Selection of an Optimal Set of Landmarks for Vision-Based Navigation

by

Pablo L. Sala

A thesis submitted in conformity with the requirements
for the degree of Master in Computer Science
Graduate Department of Computer Science
University of Toronto

Abstract

Selection of an Optimal Set of Landmarks
for Vision-Based Navigation

Pablo L. Sala
Master in Computer Science
Graduate Department of Computer Science
University of Toronto
2004

Recent work in the object recognition community has yielded a class of interest point-based features that are stable under significant changes in scale, viewpoint, and illumination, making them ideally suited to landmark-based navigation. Although many such features may be visible in a given view of the robot's environment, only a few such features are necessary to estimate the robot's position and orientation. In this thesis, we address the problem of automatically selecting, from the entire set of features visible in the robot's environment, the minimum (optimal) set by which the robot can navigate its environment. Specifically, we decompose the world into a small number of maximally sized regions such that at each position in a given region, the same small set of features is visible. We introduce a novel graph theoretic formulation of the problem and prove that it is NP-complete. Next, we introduce a number of approximation algorithms and evaluate them on both synthetic and real data.

A mamá.

Acknowledgements

I would like to start by thanking my supervisor Sven Dickinson for all his continuous support, encouragement, and guidance throughout this work, as well as his invaluable comments on grammar and style on the manuscript of this thesis. I also want to thank my second reader, Allan Jepson, for his thoughtful observations and suggestions from his careful reading of the thesis.

I likewise wish to thank Robert Sim for providing me with the image datasets that I used in the experiments and for the stimulating discussions that we shared which, like those that I had with Ali Shokoufandeh, helped to shape and enrich the work in this thesis.

I would also like to take this opportunity to express my gratitude to Debbie Gutman, Sebastian Ferro and Mario Bianchetti. Several years ago while we were still students at the University of Buenos Aires, they encouraged me to pursue graduate studies overseas, a prospect I perceived to be unattainable.

I am grateful to a large number of other people who as well have contributed in numerous ways to fulfill my vocation. The list is long and it would be impossible to name everyone here, but I would like to especially mention my former supervisor in Argentina, Hugo Scolnik, and my dear friends, Gene and Leilani Pierson, who helped me in particular ways through the first steps on this path. I cannot forget my Linear Algebra course professors at the University of Buenos Aires: Juan Sabia, Susana Tesauri and Juan José Della Barca who, through an enjoyable and exceptionally taught course, opened my eyes to all the beauty of this exciting field of mathematics, and thus started building the basics of my knowledge on this subject, so useful in my research today. If I am on this track today, it is certainly due to all these people.

I also wish to thank my dear friends in the Young Adults Fellowship at The Peoples Church who have made me feel at home in Toronto. I must certainly not fail to mention my friends at the University of Toronto who have made my lunchtimes enjoyable: Diego

Macrini, Patricio Simari, Blas Melissari, Ariel Fuxman, Sebastian Sardiña, Andres Lagar Cavilla, and Flavio Rizzolo.

I have been so blessed by my family. I am truly thankful to my mother, my brothers Jorge and Mario, including their families for all their understanding, support, and love.

Finally, and most importantly, I am grateful to God, with Whom I enjoy a personal, living relationship. Without merit He has blessed me in many ways and has granted me the talent to achieve this accomplishment.

> *Praise be to the name of God for ever and ever; wisdom and power are his. He changes times and seasons; he sets up kings and deposes them. He gives wisdom to the wise and knowledge to the discerning. He reveals deep and hidden things; he knows what lies in darkness, and light dwells with him.*

Daniel 2:20-22

Contents

List of Figures

Chapter 1

Introduction

A very important issue in mobile robotics research is that of robot navigation. For an autonomous mobile robot to be able to accomplish the tasks it has been given, without the intervention of humans, it needs to have a way to determine its current pose (i.e., its position and orientation) in the environment as well as being able to determine the steps to take in order to move to another desired position. Hence the navigation problem has been summarized by Leonard and Durrant-Whyte [21] with three questions: "Where am I?", "Where am I going" and "How do I get there?". The term *localization* refers to the set of methods for answering the first question, i.e., the robot's pose estimation. The second and third questions refer to the *goal identification* and *path-planning* problems, respectively.

Borenstein *et al.* [5] classify robot localization solutions into two main categories. The first is *relative position measurement*, which includes *odometry*, consisting of measuring the wheel rotation and steering orientation of the robot. The problem with this approach is that due to wheel slippage, collisions, etc., the error in the measurement increases without bound, and therefore there is a need to periodically use some independent reference to correct the error [10]. *Inertial navigation* is the other approach also included under this category. It consists of the use of gyroscopes and accelerometers to estimate the robot displacements and changes in orientation. These devices have the drawback that they are not suitable for accurate positioning over an extended period of time since sensor data drifts with time, increasing the error in the estimation without bound.

The other main group of robot localization methods identified by Borenstein *et al.* is that of *absolute position measurement*, which includes the use of: *active beacons*, *landmark recognition*, and *model matching*. In the first approach, the robot uses the direction of incidence of (or the measured distance to) three or more beacons, which are transmitted from known locations, to estimate its absolute position. The landmark recognition approaches can be subdivided into those that use artificial landmarks, and those that use natural landmarks. The first methods place artificial landmarks, designed to be highly distinctive and easily detectable, in known locations of the environment. Pose estimation is achieved when three or more landmarks are recognized, just as in the active beacons method. This approach has the advantage that errors are bounded. When using natural landmarks, natural distinctive features in the environment are selected as landmarks. In the case of a visual approach, landmarks are interesting locally unique image features with particular characteristics which the robot is capable of quickly recognizing in images of the environment under different environmental conditions (e.g., changes in perspective, illumination, etc.).

Localization using natural landmark recognition is a less reliable method than the previous one, but it has the advantage that it does not require an artificial modification of the environment, which is not always possible or desirable. In model matching, or *map positioning*, as it is also known, the data acquired by the robot sensors is compared to a model of the environment. The location can be estimated if features from the sensed data and the map match. There are two classes of maps: *geometric* and *topological*. A global coordinate system is used to represent the world in geometric maps, while in topological maps, the environment is represented as a graph.

There are several good reasons to use vision for navigation. Cameras are low cost sensors that capture a large amount of distinctive information. Also being passive sensors, they do not suffer from interference problems, such as those experienced by light- or sound-based proximity sensors. Additionally, if robots are to navigate populated environments, it seems reasonable to base their perceptual skills for localization on vision, just as humans do.

In a natural landmark recognition approach for robot localization using vision, two possible approaches are possible. One is object-based, in which the model of the environment consists of a database with the computed 3D location of each landmark. The second is a view-based approach, in which the database contains a collection of (landmarks observed in) 2D images of the environment and the position and orientation from which each image was acquired. Although in this thesis we focus on the problem of automatic optimal landmark selection for robot localization using a view-based approach, our idea can also be applied to object-based localization as a means to organize the landmark's database to allow for efficient landmark storage and retrieval.

1.1 A Framework for View-Based Navigation

A view-based navigation framework involves three stages, as shown in Figure 1.1: an off-line exploration of the world, the construction of a landmark database, and on-line localization. In the off-line exploration phase, images of the environment are collected at discrete point locations, and the position and orientation at which they are acquired is computed. Image features are extracted from each image, and correspondence between features from different images is computed at this point. Image features from different images corresponding to the same scene feature are assigned the same feature id. Finally, using the feature correspondences, a *feature visibility vector* is generated for each sampled pose, specifying what scene features are visible from that pose.

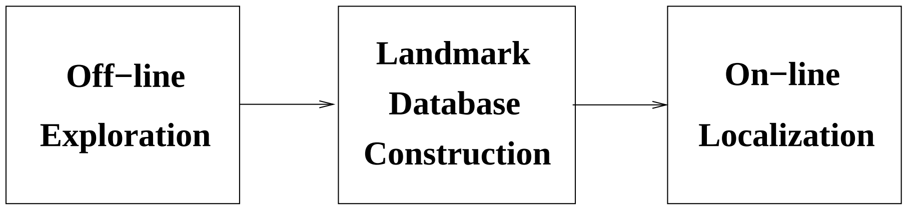


Figure 1.1: View-based localization framework.

The landmark database construction phase uses as its input the collection of images and the set of feature visibility vectors obtained in the previous stage. The purpose of this phase is to create a database of landmarks useful for reliable navigation. (A landmark is defined as a set of features.) The generated database has to specify which landmark is to be used for localization at each pose, and the locations of all the features that form that landmark in two *model views*. A model view for a particular landmark is one of the images collected in the previous stage in which all the features that form the landmark are simultaneously visible. The database must also record the position and orientation at which each model view was acquired.

Finally, the on-line localization process is that in which the robot estimates its actual 3D location using as its input the database of landmarks produced in the previous stage. Features visible in the current robot's image have to be matched to those in the database. The locations of the matched features in the two corresponding model views are then used as input to a method to estimate the 3D position and orientation of the robot, relative to that of the model views.

This thesis will primarily focus on the second stage of the framework, namely, the problem of constructing the database of landmarks. Our main contribution is a practical and optimal solution to the problem of landmark selection, and it assumes that we have obtained the set of feature visibility vectors from a previous off-line exploration process. We include chapters in which we propose methods to accomplish the tasks involved in the off-line exploration and on-line localization phases, as to offer the whole picture of a practical framework. However, we do not provide experimental results for the methods that we propose to realize these tasks. The experiments performed in the Results chapter in this thesis will only include tests of our proposed solution to the problem of optimal landmark selection.

1.2 Interest-Point Based Features as Landmarks

Several natural image features have been used as natural visual landmarks, ranging from very simple features such as points and lines [30, 40], to more complex patterns [29] or 3D models of world objects. In the domain of exemplar-based (as opposed to generic) object recognition, the computer vision community has recently adopted a class of interest point-based features, e.g., [25, 14, 8]. Such features typically encode a description of image appearance in the neighbourhood of an interest point, such as a detected corner or scale-space maximum. The appeal of these features over their appearance-based predecessors is their invariance to changes in illumination, scale, image translation and rotation, and minor changes in viewpoint (rotation in depth). These properties therefore make them ideally suited to the problem of landmark-based navigation. If we can define a set of invariant features that uniquely define a particular location in the environment, these features can, in turn, define a visual landmark.

To use these features, we could, for example, adopt a localization approach proposed by Basri and Rivlin [2] and Wilkes *et al.* [46], based on the LC (*linear combination of views*) technique. (See section 7.3.) During a training phase, the robot is manually "shown" two views of each landmark in the environment by which the robot is to later navigate. These views, along with the positions at which they were acquired, form a database of landmark views. At runtime, the robot takes an image of the environment and attempts to match the visible features to the various landmark views it has stored in its database. Given a match to some landmark view, the robot can compute its position and orientation in the world.

1.3 Selection of an Optimal Set of Landmarks

There are two major challenges with this approach. First, from any given viewpoint, there may be hundreds or even thousands of such features. The union of all pairs of landmark views may therefore yield an intractable number of distinguishable features that must be indexed in order to determine which landmark the robot may be viewing.[1] Fortunately, only a small number of features are required (in each model view) to compute the robot's pose. Therefore, of the hundreds of features visible in a model view, which small subset should we keep?

The second challenge is to automate this process and let the robot automatically decide on an optimal set of visual landmarks for navigation. What constitutes a good landmark? A landmark should be both distinguishable from other landmarks (a single floor tile, for example, would constitute a bad landmark since it's repeated elsewhere on the floor) and widely visible (a landmark visible only from a single location will rarely be encountered and, if so, will not be persistent). Therefore, our goal can be formulated as partitioning the world into a minimum number of maximally sized contiguous regions, such that the same set of features is visible at all points within a given region.

There is an important connection between these two challenges. Specifically, given a region, inside of which all points see the same set of features (our second challenge), what happens when we reduce the set of features that must be visible at each point (first challenge)? Since this represents a weaker constraint on the region, the size of the region can only increase, yielding a smaller number of larger regions covering the environment. As mentioned earlier, there is a lower bound on the number of features that can define a region, based on the pose estimation algorithm and the degree to which we want to

[1]Worst-case indexing complexity would occur during the kidnapped localization task, in which the robot has no prior knowledge of where it is in the world. Under normal circumstances, given the currently viewed landmark and the current heading, the space of landmark views that must be searched can be constrained. Still, even for a small set of model views (landmarks), this may yield a large search space of features.

overconstrain its solution.

Combining these two challenges, we arrive at the main problem addressed by this thesis: from a set of views acquired at a set of sampled positions in a given environment, partition the world into a minimum set of maximally sized regions, such that at all positions within a given region, the same set of k features is visible, where k is defined by the pose estimation procedure (or some overconstrained version of it).

1.4 Outline

This thesis is organized as follows: In Chapter 2, we present a discussion of related work for robot navigation using visual landmarks. The definition of the optimal landmark selection problem, the main focus of this thesis, is presented in Chapter 3, where we introduce a novel, graph theoretic formulation of this problem. Its intractability is proven in Chapter 4. In the absence of optimal, polynomial-time algorithms, in Chapter 5, we introduce six different approximation algorithms for solving the problem. Methods for estimating robot pose during the off-line image collection phase and computing feature correspondences between images are presented in Chapter 6. In Chapter 7, we discuss methods for view-based localization. We have constructed a simulator that can generate thousands of worlds with varying conditions, allowing us to perform exhaustive empirical evaluation of the six algorithms. In Chapter 8, following a comparison of the algorithms on synthetic environments, we adopt the most effective algorithm, and test it on real world imagery of a real environment. Finally, in Chapter 9, we conclude with a discussion of the main contributions made and possible directions for future work.

Chapter 2

Related Work

In this section, we briefly discuss some previous work done for robot localization using landmark recognition.

2.1 Localization from Landmark Recognition

If a robot is provided with an a priori map of the 3D locations of known model landmarks, its pose can be estimated through a triangulation process after the correspondences between the model landmarks and those sensed by the robot are found. Betke and Gurvits [3] presented an efficient solution to this problem, when the robot is navigating on a plane, in which they estimate the position and orientation of the robot from an overdetermined set of bearings of the sensed landmarks. By representing the landmark locations using the complex-domain, they came up with an algorithm that runs in time linear in the number of visible landmarks.

2.2 Dealing With Uncertainty

Since, in the estimated robot's position, there is always present a certain amount of uncertainty, some authors have considered the problem of landmark selection for the purpose of minimizing uncertainty in the computed pose estimate. Sutherland and Thompson [42] demonstrate that the precision of a pose estimate derived from point features in 2D is dependent on the configuration of the observed features, and provide an algorithm for selecting an appropriate set of observed features from which to compute a pose estimate.

Methods have also been developed to combine multiple unreliable observations into a more reliable estimate. Measurements from various sensors, data acquired over time, and previous estimates are integrated to compute a more accurate estimate of the current robot's pose. In every sensor update, previous data is weighted according to how accurately it predicts the current observations. This technique, called *sensor fusion*, has generally been implemented through the use of *Kalman filters* and *Extended Kalman Filters (EKF)*. It has been applied to the problem of localization by Leonard and Durrant-Whyte [21] from sonar data obtained over time. A disadvantage of Kalman filters and EKF is that since they realize a local linear approximation to the exact relationship between the position and observations, they depend on a good a priori estimate, therefore suffering from robustness problems.

Markov localization is a statistical localization approach utilizing Kalman filtering which models uncertainty using a probability distribution over pose space. As evidence is collected from the sensors, it is used to update the current state of belief of the robot's pose. In [44], Thrun presents an approach based on Markov localization in which neural networks are trained to discover landmarks that will minimize the localization error. The proposed algorithm has the advantage of being widely applicable, since the robot customizes its localization algorithm to the sensors' characteristics and the particular environment in which it is navigating. The localization error achieved by the automatically selected landmarks is shown to outperform the error achieved with landmarks carefully selected by human experts. On the other hand, this approach has the drawback that the training of the neural networks can take several hours, though this process generally needs to be performed only once in an off-line stage.

Another set of approaches that make use of Kalman Filtering is that of Simultaneous Localization and Mapping (SLAM), in which after each new measurement, it is attempted to not only improve the estimation of the current robot's pose, but also reduce the uncertainty of the 3D map of landmarks computed so far. Davison's work in this direction basically computes a solution to the structure-from-motion problem on-line [11]. A problem with this technique is that the time complexity of each sensor update step increases with the square of the number of landmarks in the database. To deal with this scalability problem, some authors suggested dividing the global map into sub-maps, within which the complexity can be bounded [6, 39]. Other researchers [23, 20, 9, 13] have proposed hierarchical approaches to SLAM, in which a topological map is maintained, organizing the landmarks in smaller regions where feature-based mapping strategies are applied.

2.3 Natural vs Artificial Landmarks

The landmarks can be either natural distinctive features in the environment, or artificial landmarks designed to be easily recognized and distinguished. Numerous researchers have taken this latter approach, e.g., [22, 34, 43, 7]. However, it has the drawback that an artificial modification of the environment to include the landmarks, besides requiring a prior and generally human intervention, is not always possible or desirable.

Some researchers have chosen as natural landmarks simple features such as lines. For example, Moon *et al.* [30] used vertical lines, under the assumption that they can be reliably extracted from image edge information, and if using a camera vertically aligned, vertical structures will be observed as vertical lines from all viewpoints. Although these features remain invariant to changes in perspective, they lack distinctiveness. Furthermore, these assumptions can break down if the terrain is not flat, the camera is not firmly mounted, or if there is a scarcity of straight, vertical structures.

Recent work in the object recognition community has yielded a class of interest point-based features that are stable under significant changes in scale, viewpoint, and illumination, making them ideally suited to landmark-based navigation [25, 8, 4]. Lowe, Se and Little [36] have used scale- and rotation-invariant features as landmarks, extracted using Lowe's scale-invariant feature transform (SIFT) [25]. The robot automatically updates a 3D landmark map with the reliable landmarks seen from the current position using Kalman filtering techniques. The position of the robot is estimated using the odometry of the robot as an initial guess, and is improved using the map. Trinocular vision is used to estimate the 3D locations of landmarks and their regions of confidence, with all reliable landmarks stored in a dense database.

2.4 View-Based Approaches to Localization

Navigation by landmark recognition is still possible without knowledge of the locations of the landmarks in a map of the environment. Localization can be accomplished in a view-based fashion, in which the robot knows only the image location of the landmarks in a collection of model views of the environment acquired at known positions and orientations. One such approach is the linear combination of views (LC) technique, which was first introduced by Ullman and Basri for object recognition, and later applied to vision-based navigation by Basri and Rivlin [2]. (See section 7.3.) The authors proved that if a scene is represented as a set of 2D views, each novel view of the scene can be computed as a linear combination of the model views. From the value of the linear coefficients, it is possible to estimate the position from which the novel view was acquired, relative to that of the model views. Wilkes et al. [46] described a practical robot navigation system that used the LC technique. Their method consists of decomposing the environment into regions within which a set of model views of a particular piece of the environment may be used to determine the position of the robot. In these original applications of the LC method, the features comprising the model views were typically linear features extracted from the image.

The view-based approach of Sim and Dudek [37] consists of an off-line collection of monocular images sampled over a space of poses. The landmarks consist of PCA encodings of the neighborhoods of salient points in the images, obtained using an attention operator. Landmarks are tracked between contiguous poses and added to a database if stable through a region of reasonable size and sufficiently useful for pose estimation according to an a priori utility measure. Each stored landmark is parameterized on the basis of a set of computed landmark attributes. The localization is performed by finding matches between the candidate landmarks visible in the current image and those in the database. A position estimate is obtained by merging the individual estimates yielded by each computed attribute of each matched candidate landmark.

While all of these approaches demonstrate how robot localization can be performed from a set of landmark observations, none consider the issue of eliminating redundancy from the landmark-based map, which at times can grow to contain tens of thousands of landmark models. Additionally, little or no attention has been given to the number of landmark look-ups required for localization. In this thesis, we will study the problem of automatically selecting a minimum size subset of landmarks such that reliable navigation is still possible. While maximizing precision is clearly an important issue, in this thesis we are concerned primarily with selecting landmarks that are widely visible.[1]

[1]The algorithms presented in this work can be easily extended to select sets of features that fulfill any given additional constraints.

Chapter 3

Landmark Selection Problem Definition

In this chapter, we present a definition of the particular problem of view-based navigation that we try to solve in this thesis. We would like to address the question: What is an optimal set of visual landmarks by which a mobile robot can reliably navigate in a given environment?

3.1 Optimal Landmark Selection

This section starts by explaining the processes involved in a view-based navigation approach: the collection of images of the environment, and how the view-based localization process makes use of those images. Finally, we introduce the problem that is the main focus of this thesis: how to select an optimal set of landmarks for practical, reliable, view-based navigation.

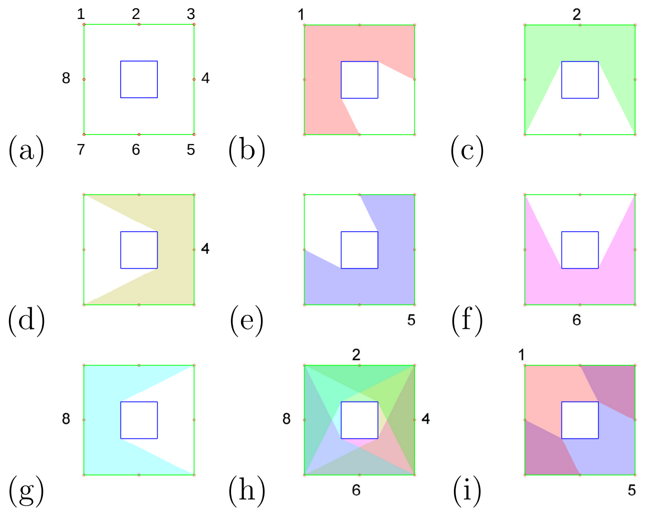


Figure 3.1: (a) A simple world with a square perimeter (in green), a square (blue) obstacle in its center and eight features (red circles on its perimeter). (b)-(g) Visibility areas of some features, (h) A covering of the world using 4 features. (i) A covering of the world using 2 features.

3.1.1 Off-line Image Collection

In view-based navigation, there is an off-line training phase, in which images are first collected at known discrete points in pose space, e.g., the accessible vertices (points) of a virtual grid overlaid on the floor of the environment. During collection, the known pose of the robot is recorded for each image, and a set of interest point-based features are extracted and stored in a database. For each of the grid points, we therefore know exactly which features in the database are visible. Conversely, for each feature in the database, we know from which grid points it is visible. Consider the example shown in Figure 3.1. Figure 3.1 (a) shows a simple 2-D world having a square perimeter, a square obstacle in its center, and eight features evenly distributed along its perimeter. In figures 3.1 (b) - 3.1 (g), the area of visibility of some of the features is shown as a coloured region. The feature visibility areas, computed from a set of images acquired at a set of grid points in the world, constitute the input to our problem.

3.1.2 View Based Localization

In a view-based localization approach, the current pose of the robot is estimated using, as input, the locations of a small number of features in the current image matched against their locations in the training images. This set of simultaneously visible features constitutes a landmark. The minimum number of features necessary for this task depends on the method employed for pose estimation. For example, three features are enough for localization in Basri and Rivlin's linear combination of views technique [2], which uses a weak perspective projection imaging model. (See section 7.3 for a detailed explanation on this localization method.) The essential matrix method [48], that properly models perspective projection in the imaging process, requires at least eight features to estimate pose.

To reduce the effect of noise, a larger number of features can be used to overconstrain the solution. This presents a trade-off between the accuracy of the estimation and the size (in features) of the landmark. Requiring a larger number of features for localization, will yield better pose estimation. However, the more constrained a landmark is, the smaller its region of visibility becomes. We will define the parameter k as the number of features that will be employed to achieve pose estimation with the desired accuracy, i.e., the number of features constituting a landmark.

3.1.3 An Optimal Set of Landmarks for View-Based Navigation

Robot localization from a given position is possible if, from the features extracted from an image taken at that position, there exists a subset of k features that exist in the database and that are simultaneously visible from at least two known locations. For a large environment, the database may be large, and such a search may be costly. For each image feature, we would have to search the entire database for a matching feature until not only k such matches were found, but that those k features were simultaneously visible from at least two separate positions (grid points).

Recalling that k is typically far less than the number of features in a given image, one approach to reducing search complexity would be to prune features from the database subject to the existence of a minimum of k features visible at each point, with those same k features being visible at one or more other positions. Unfortunately, this is a complex optimization problem whose solution still maintains all the features in a single database, leading to a potentially costly search. A more promising approach is to partition the pose space into a number of *regions*, i.e., sets of contiguous grid points, such that for each region, there are at least k features simultaneously visible from all the points in the region. Such a partitioning of the world, in turn, partitions the database of features into a set of smaller databases, each corresponding to what the robot sees in a spatially coherent region. In this latter approach, since k is small, the total number of features (corresponding to the union of all the databases) that need to be retained for localization is much smaller than that of the unique database in the previous approach. Therefore, even without prior knowledge of the region in which the robot is located, the search results to be far less costly.

Let's return to the simple world depicted in Figure 3.1. In this example, we will assume, for sake of clarity, that a single ($k = 1$) feature is sufficient for reliable navigation. However the reader must note that in practice a k greater than 1 is generally required for localization, its particular minimum value depending on the method employed. Under this assumption, one possible decomposition of the world into a set of regions (such that each pose of the world sees at least one feature) is achieved using features 2, 4, 6, and 8, as shown in Figure 3.1 (h). It is clear that all four features in this set are needed to cover the world, since removing any one of them will yield some portion of the world from which the remaining three features are not visible, meaning that the robot is blind in this area. However, this decomposition is not optimal, since other decompositions with fewer regions are possible. Our goal is to find the minimum decomposition of the world which,

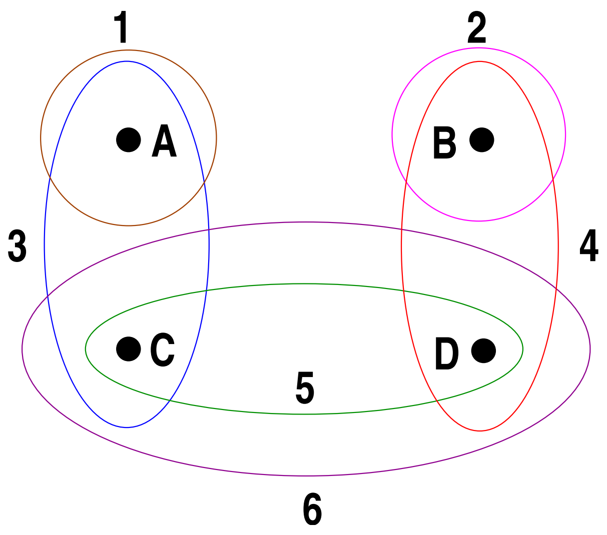


Figure 3.2: Example of a world with with four poses (A, B, C, D), in which the minimum decomposition in regions with two features each, does not correspond to minimizing the total number of active features.

in this case, has only two regions, corresponding to the areas of visibility of features 1 and 5, (or its symmetric solution using features 3 and 7), as shown in Figure 3.1 (i). This minimum set of maximally sized regions is our desired output, and allows us to discard from the database all but features 1 and 5. Since at least one of these two features is seen from every point in pose space, reliable navigation through the entire world is possible.

3.1.4 Advantages of This Approach

Besides reducing the total number of features to be stored, a partitioning of the world into regions offers additional advantages. While navigating inside a region, the corresponding k features are easily tracked between the images that the robot sees. If the expected k features are not all visible in the current image, this may indicate that the robot has left the region in which it was navigating and is entering a new region. In that case, the visible features can vote for the regions they belong to, if any, according to a membership relationship computed off-line. The new region(s) into which the robot is likely moving will be those with at least k votes. Input features would therefore be matched to the

k model features defining each of the candidate regions. This approach also provides a solution to the *kidnapped robot problem*, i.e., if the robot is blindfolded and released at an arbitrary position, it can estimate its current pose.

3.1.5 Minimizing Regions $\neq$ Minimizing Features

From the example in Figure 3.1, it may seem to some readers that the problem of minimizing the number of regions is equivalent to that of minimizing the total number of used features. However, although these two problems are equivalent when the number of features required per regions is $k = 1$, that is not the case when $k > 1$. In Figure 3.2, we show a world with four poses (A, B, C, D) and six features whose regions of visibility are pictured as Venn diagrams, numbered from 1 to 6. We attempt to find a decomposition of this world into regions that see at least $k = 2$ features per region. There is a unique decomposition into four regions (one per pose), if features 1 through 5 are used. However, if all 6 features are employed, we can achieve a decomposition with three regions (features C and D belong to the same region now, commonly seeing features 5 and 6.)

3.2 A Graph Theoretic Formulation

In this section, we introduce a novel, graph theoretic formulation of the problem of optimal landmark selection.

3.2.1 Definition of Terms

Before we formally define the minimization problem under consideration, we will introduce some terms.

Definition 3.1. *The set of positions at which the robot can be at any time is called the* pose space*. The discrete subset of the pose space from which images were acquired is modeled by an undirected planar graph $G = (V, E)$, where each node $v \in V$ corresponds* *to a sampled pose, and two nodes are adjacent if the corresponding poses are contiguous in 2D space.*

Definition 3.2. *Let F be the set of computed features from all collected images. The* visibility-set of v *is the set $\mathcal{F}_v \subset F$ of all features that are visible from pose $v \in V$.*

Definition 3.3. *A* world instance *consists of a tuple $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$, where the graph G models a discrete set of sampled poses, F is a set of features, and $\{\mathcal{F}_v\}_{v \in V}$ is a collection of visibility-sets.*

Definition 3.4. *A set of poses $R \subset V$ is said to be a* region *iff for all poses $u, v \in R$, there is a path between u and v completely contained in R, i.e.,*
$\forall u, v \in R : \exists \{u = v_0, \ldots, v_h = v\} \subseteq R$, *such that* $(v_i, v_{i+1}) \in E$ *for all* $0 \leq i < h$.

Definition 3.5. *A set of regions $D = \{R_1, \ldots, R_d\} \subset 2^V$ is said to be a* decomposition of V *iff* $\bigcup_{1 \leq i \leq d} R_i = V$.

Definitions 3.1 to 3.5 define the set of inputs and outputs of interest to our problem. In view of our optimization problem, for a given world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$, one would like to create a minimum cardinality D. In addition, it will be desirable for a given solution to the optimization problem to satisfy a variety of properties. One property of interest is that of ensuring a minimum amount of overlap between regions in the decomposition. The purpose of overlap is to ensure smooth transitions between regions, as different sets of features become visible to the robot. When one region's features start to fade at its border, the robot can be assured to be within the boundary of some other region, where the new region's landmark is clearly visible. The following definitions formalize this property:

Definition 3.6. *The ρ-neighborhood of a pose $v \in V$ is the set $N_\rho(v) = \{u \in V : \delta(u, v) \leq \rho\}$, where $\delta(u, v)$ is the length of the shortest path between nodes u and v in G.*

Definition 3.7. *A decomposition $D = \{R_1, \ldots, R_d\}$ of V is said to be* ρ-overlapping *iff* $(\forall v \in V)(\exists i) : N_\rho(v) \subset R_i$.

3.2.2 The Minimum Overlapping Region Decomposition Problem

With these definitions in hand, the problem can now be formally stated as follows:

Definition 3.8. *Let k be the number of features required for reliable localization at each position, according to the localization method employed. The* ρ-Minimum Overlapping Region Decomposition Problem (ρ-MORDP) *for a world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$ consists of finding a minimum-size ρ-overlapping decomposition $D = \{R_1, \ldots, R_d\}$ of V into regions, such that $\forall i : |\bigcap_{v \in R_i} \mathcal{F}_v| \geq k$.*

Note that given a solution of size d to this problem, the total number of features needed for reliable navigation is bounded by $d \cdot k$.

Chapter 4

Complexity Analysis

In the previous chapter, we introduced a novel, graph theoretic formulation of the problem of optimal landmark selection. In the present chapter, we will proceed to prove its intractability.

4.1 ρ-MORDP = 0-MORDP

Before we consider the complexity of ρ-MORDP, we will present two theorems indicating that ρ-MORDP can be reduced to 0-MORDP ($\rho = 0$), and that a solution to the reduced 0-MORDP can be transformed back into a solution of the more general ρ-MORDP. The first of the following two theorems states that if there is a ρ-overlapping decomposition, such that k features are visible in each region for a certain world instance, then there is a 0-overlapping decomposition for the related problem also with k features visible in each region. This theorem guarantees that if a solution exists for the ρ-MORDP, then there is also a solution to the related 0-MORDP.

The second theorem states that whenever the related 0-MORDP has a solution $\tilde{D}$, then the ρ-MORDP has a solution too, and it presents the method to construct it from $\tilde{D}$. The theorems depend on three lemmas which will be proven below. It should be noted that while the transformation from ρ-MORDP to 0-MORDP and back to ρ-MORDP may create a different ρ-overlapping decomposition, the cardinality of the decomposition under this two-step transformation will remain the same, hence the optimality will not be affected.

Theorem 4.1. *If $D = \{R_1, \ldots, R_d\}$ is a ρ-overlapping decomposition of V for a world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V}\rangle$, such that $|\bigcap_{v \in R_i} \mathcal{F}_v| \geq k$ for all $i = 1, \ldots, d$, then $\tilde{D} = \{\tilde{R}_1, \ldots, \tilde{R}_d\}$, where $\tilde{R}_i = \{v \in R_i : N_\rho(v) \subseteq R_i\}$, is a 0-overlapping decomposition for a world instance $\langle G = (V, E), F, \{\tilde{\mathcal{F}}_v\}_{v \in V}\rangle$, where $\tilde{\mathcal{F}}_v = \bigcap_{w \in N_\rho(v)} \mathcal{F}_w$, such that $|\bigcap_{v \in \tilde{R}_i} \tilde{\mathcal{F}}_v| \geq k$ for all $i = 1, \ldots, d$.*

Proof: According to Lemma 4.1, we know that $\tilde{D}$ is a 0-overlapping decomposition of V. By Lemma 4.3, we know that $\bigcap_{v \in R_i} \mathcal{F}_v \subseteq \bigcap_{v \in \tilde{R}_i} \tilde{\mathcal{F}}_v$, for all $i = 1, \ldots, d$. Therefore, $|\bigcap_{v \in \tilde{R}_i} \tilde{\mathcal{F}}_v| \geq |\bigcap_{v \in R_i} \mathcal{F}_v| \geq k$, for all $i = 1, \ldots, d$. $\square$.

Theorem 4.2. *If $\tilde{D} = \{\tilde{R}_1, \ldots, \tilde{R}_d\}$ is a solution to 0-MORDP for a world instance $\langle G = (V, E), F, \{\tilde{\mathcal{F}}_v\}_{v \in V}\rangle$, then $D' = \{R'_1, \ldots, R'_d\}$, where $R'_i = \bigcup_{v \in \tilde{R}_i} N_\rho(v)$, is a solution to ρ-MORDP for the world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V}\rangle$.*

Proof: We have to show that:

1. D' is a ρ-overlapping decomposition of V, i.e., $(\forall v \in V)(\exists i) : N_\rho(v) \subset R'_i$. (This is direct from Lemma 4.2.),

2. $|\bigcap_{v \in R'_i} \mathcal{F}_v| \geq k$ for all $i = 1, \ldots, d$. (Direct from Lemma 4.3 and the facts that $\tilde{D}$ is a 0-MORDP solution, and $R'_i = \bigcup_{v \in \tilde{R}_i} N_\rho(v)$.), and

3. D' is minimum size.

   (We'll prove this by contradiction. We'll suppose that there is solution D'' to ρ-MORDP that has size less than D', and will show that from this, we can construct a 0-MORDP decomposition $\tilde{D''}$ for the original problem of size smaller than $\tilde{D}$ with the property $|\bigcap_{v\in\tilde{R''}_i}\tilde{\mathcal{F}}_v| \geq k$. This is a contradiction, since $\tilde{D}$ was a decomposition of minimum size with that property.

Suppose $D'' = \{R''_1, \ldots, R''_h\}$ is a decomposition for the original ρ-overlapping problem such that $h < t$ and $|\bigcap_{v\in R''_i}\mathcal{F}_v| \geq k$ for all $i = 1, \ldots, h$.
Let $\tilde{D''} = \{\tilde{R''}_1, \ldots, \tilde{R''}_h\}$, where $\tilde{R}''_i = \{v \in R''_i : N_\rho(v) \subseteq R''_i\}$ for all $i = 1, \ldots, h$. By Lemma 4.1, we know that $\tilde{D''}$ is a 0-overlapping decomposition of V, and by Lemma 4.3, we can affirm that $|\bigcap_{v\in\tilde{R''}_i}\tilde{\mathcal{F}}_v| \geq |\bigcap_{v\in R''_i}\mathcal{F}_v| \geq k$.)

$\square$.

Lemma 4.1. $\{R_1, \ldots, R_d\}$ *is a* ρ*-overlapping decomposition of* V *if and only if* $\{\tilde{R}_1, \ldots, \tilde{R}_d\}$ *is a* 0*-overlapping decomposition of* V*, where* $\tilde{R}_i = \{v \in R_i : N_\rho(v) \subseteq R_i\}$ *for all* $i = 1, \ldots, d$.

Proof: $\{R_1, \ldots, R_d\}$ is a ρ-overlapping decomposition of $V \iff (\forall v \in V)(\exists i : 1 \leq i \leq d) N_\rho(v) \subseteq R_i \overset{(*)}{\iff} (\forall v \in V)(\exists i : 1 \leq i \leq d) v \in \tilde{R}_i \iff (\forall v \in V)(\exists i : 1 \leq i \leq d) N_0(v) \subseteq \tilde{R}_i \iff \{\tilde{R}_1, \ldots, \tilde{R}_d\}$ is a 0-overlapping decomposition of V.
In (*) we use the definition of $\tilde{R}_i$.
$\square$.

Lemma 4.2. $\{\tilde{R}_1, \ldots, \tilde{R}_d\}$ *is a* 0*-overlapping decomposition of* V *if and only if* $\{R'_1, \ldots, R'_d\}$ *is a* ρ*-overlapping decomposition of* V*, where* $R'_i = \bigcup_{v\in\tilde{R}_i} N_\rho(v)$ *for all* $i = 1, \ldots, d$.

Proof: First, observe that R'_i is a region, since $\tilde{R}_i$ is a region and $N_\rho(v)$ is path connected, as it can be inferred from its definition. Now, $\{\tilde{R}_1, \ldots, \tilde{R}_d\}$ is a 0-overlapping decomposition of $V \iff (\forall v \in V)(\exists i) : N_0(v) \subset \tilde{R}_i \iff (\forall v \in V)(\exists i) : v \in \tilde{R}_i \overset{(*)}{\iff} (\forall v \in V)(\exists i) : N_\rho(v) \subseteq R'_i \overset{(**)}{\iff} \{R'_1, \ldots, R'_d\}$ is a ρ-overlapping decomposition of V.

In (*) we use that $(\forall v \in \tilde{R}_i) : N_\rho(v) \subseteq R'_i$, which is a direct implication of the definition of R'_i. In (**) we use that R'_i is a region.

□.

Lemma 4.3. *If $\{R_1, \ldots, R_d\}$ is a ρ-overlapping decomposition of V, $\tilde{R}_i = \{v \in R_i : N_\rho(v) \subseteq R_i\}$ for all $i = 1, \ldots, d$, and $\tilde{\mathcal{F}}_v = \bigcap_{w \in N_\rho(v)} \mathcal{F}_w$, then for all $i = 1, \ldots, d$: $\bigcap_{v \in R_i} \mathcal{F}_v \subseteq \bigcap_{v \in \tilde{R}_i} \tilde{\mathcal{F}}_v$, with equality holding if $R_i = \bigcup_{v \in \tilde{R}_i} N_\rho(v)$.*

Proof: From the definition of $\tilde{R}_i$, we know that $(\forall v \in \tilde{R}_i) : N_\rho(v) \subseteq R_i$, and hence $\bigcup_{v \in \tilde{R}_i} N_\rho(v) \subseteq R_i$. Therefore $\bigcap_{w \in R_i} \mathcal{F}_w \subseteq \bigcap_{w \in \left(\bigcup_{v \in \tilde{R}_i} N_\rho(v)\right)} \mathcal{F}_w$, and the equality holds when $R_i = \bigcup_{v \in \tilde{R}_i} N_\rho(v)$. Now, $\bigcap_{w \in \left(\bigcup_{v \in \tilde{R}_i} N_\rho(v)\right)} \mathcal{F}_w = \bigcap_{v \in \tilde{R}_i} \left(\bigcap_{w \in N_\rho(v)} \mathcal{F}_w\right) = \bigcap_{v \in \tilde{R}_i} \tilde{\mathcal{F}}_w$.

□.

The transformation applied in Theorem 4.1 from a ρ-overlapping to a 0-overlapping solution effectively shrinks the regions of D by ρ, and reduces the visibility-set of each vertex v to correspond to only those features that are visible over the entire neighborhood $N_\rho(v)$ of v.[1] Theorem 4.2 assumes that the collection of visibility-sets $\tilde{\mathcal{F}}$ input to 0-MORDP is defined by a reduction of the ρ-overlapping instance of the problem to a 0-overlapping instance using the transformation described in Theorem 4.1.

4.2 Complexity of 0-MORDP

We will now show that 0-MORDP is NP-complete. The proof is by reduction from the Minimum Set Cover Problem.

Definition 4.1. *Given a set U, and a set of subsets $S = \{S_1, \ldots, S_m\}$ of U, the* Minimum Set Cover Problem *(MSCP) consists of finding a minimum set $C \subset S$ such that each*

[1]Strictly speaking, the region reduction is impervious to boundary effects at the boundary of G, due to the definition of $N_\rho(v)$.

$$\begin{aligned} U &= \{A, B, C, D\} \\ S &= \{\{A, B\}, \{C\}, \\ &\quad \{A, D\}, \{C, D\}\} \end{aligned}$$

Figure 4.1: An instance of the *Minimum Set Cover Problem*

element of U is covered at least once, i.e., $\bigcup_{S_i \in C} S_i = U$.

Figure 4.1 presents an instance of MSCP. The optimal solution for this instance is $C = \{\{A, B\}, \{C, D\}\}$ and, in fact, this solution is unique. An instance $\langle U, S, r\rangle$ of the Set Cover *decision* problem, where r is an integer, consists of determining if there is a set cover of U, by elements of S, of size at most r. The decision version of SCP was proven to be NP-complete by Karp [19], with the size of the problem measured in terms of $|S|$.

Theorem 4.3. *The decision problem* $\langle$0-*ORDP, d*$\rangle$ *is NP-complete.*

Proof: It is clear that 0-MORDP is in NP, i.e., given a world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V}\rangle$ and $D = \{R_1, \ldots, R_d\}$, it can be verified in time polynomial in $\max(|V|, |F|)$ if D is a ρ-overlapping decomposition of V such that $\forall i : |\bigcap_{v \in R_i} \mathcal{F}_v| \geq k$. We now show that any instance of SCP can be reduced to an instance of 0-ORDP in time polynomial in $|V|$. Given an instance $\langle U, S = \{S_1, \ldots, S_m\}\rangle$ of the Minimum Set Cover Problem, we construct a 0-ORDP for the world instance $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V}\rangle$ in the following way:

- Let v^* be an element not in U; then $V = U \cup \{v^*\}$
- $E = \{(u, v^*) : u \in U\}$ (Note that the graph G thus generated is planar.)
- $F = \{f_1, \ldots, f_m\}$ where $f_i = S_i \cup \{v^*\}$
- $\mathcal{F}_v = \{f \in F : v \in f\}$
- $k = 1$

The introduction of the dummy vertex v^* will be used in the proof to ensure that elements of U that belong to the same subset S_i can be part of the same region in the decomposition, by virtue of their mutual connection to v^*. Each visibility-set $\mathcal{F}_v$ in the transformed problem instance corresponds to a list of the sets S_i in the SCP instance that element v is a member of.

Now we show that from a solution to 0-ORDP of size d, we can build a SC of size d. Let $D = \{R_1, \ldots, R_d\}$ be a solution to the transformed 0-ORDP instance, i.e.,

1. $R_i \subseteq V$ is a region, for $i = 1, \ldots, d$,
2. $\bigcup_{1 \leq i \leq d} R_i = V$, and
3. $|\bigcap_{v \in R_i} \mathcal{F}_v| \geq k = 1$, for $i = 1, \ldots, d$.

Claim: $C = \{C_1, \ldots, C_d\}$, with $C_i = \mathsf{first}_{lex}(\bigcap_{v \in R_i} \mathcal{F}_v) - \{v^*\}$ is a Set Cover for the original problem, where $\mathsf{first}_{lex}(A)$ returns the first element in lexicographical order from the non-empty set A. (For each C_i, the choice of an element f from $\bigcap_{v \in R_i} \mathcal{F}_v$ is arbitrary in that any such f yields a valid solution.) Note that C_i is well-defined, since $|\bigcap_{v \in R_i} \mathcal{F}_v| \geq 1$.

Proof: We must show that:

1. $\forall i = 1, \ldots, d : C_i \in S$:

   From the definition of C_i we can affirm that $(\exists j) : [1 \leq j \leq m$ and $C_i = f_j - \{v^*\}]$. Hence $C_i = S_j \in S$.

2. $\bigcup_{1 \leq i \leq d} C_i = U$:

   From the definition of $\mathcal{F}_v$:

$$\begin{aligned} \bigcap_{v \in R_i} \mathcal{F}_v &= \bigcap_{v \in R_i} \{f \in F : v \in f\} \\ &= \{f \in F : R_i \subseteq f\} \end{aligned}$$

Therefore, from the definition of C_i:

$$\begin{aligned} C_i &= \mathsf{first}_{lex}\{f \in F : R_i \subseteq f\} - \{v^*\} \\ \implies & R_i \subseteq C_i \cup \{v^*\} \\ \implies & V = \bigcup_{1 \le i \le d} R_i \subseteq \bigcup_{1 \le i \le d} C_i \cup \{v^*\} \subseteq V \\ \implies & \bigcup_{1 \le i \le d} C_i \cup \{v^*\} = V \\ \implies & \bigcup_{1 \le i \le d} C_i = V - \{v^*\} = U, \end{aligned}$$

Finally, we have to show that if there is a set cover of size d, then there is a decomposition of size d for the 0-ORDP. Let $C' = \{C'_1, \ldots, C'_d\}$ be a set cover for the original SCP instance.

Claim: $D' = \{R'_1, \ldots, R'_d\}$, where $R'_i = C'_i \cup \{v^*\}$, is a 0-overlapping region decomposition such that $|\bigcap_{v \in R'_i} \mathcal{F}_v| \ge 1$.

Proof: We must show that:

1. Each $R'_i \subseteq V$ is a region[2]:

   $\forall i : 1 \le i \le d$, since $C'_i \subseteq U$, then $R'_i = C'_i \cup \{v^*\} \subseteq V$.

   R'_i is a region because $v^* \in R'_i$ and, by the definition of the graph G, v^* is connected to all other nodes in R_i.

2. $\bigcup_{1 \le i \le d} R'_i = V$:

$$\bigcup_{1 \le i \le d} R'_i = \bigcup_{1 \le i \le d} C'_i \cup \{v^*\} = U \cup \{v^*\} = V$$

[2]Recall that a region corresponds to a subset R of vertices in V for which a path exists between any two vertices in R that lies entirely within R.

3. $|\bigcap_{v \in R'_i} \mathcal{F}_v| \geq 1$:

$$\begin{aligned}
&C'_i \text{ is a set cover} \\
\implies\ & C'_i \in S \\
\implies\ & \exists j = 1, \ldots, m : C'_i = S_j \\
\implies\ & R'_i = S_j \cup \{v^*\} = f_j \in F \\
\implies\ & 1 \leq |\{f \in F : R'_i \subseteq f\}| \\
& = |\bigcap_{v \in R'_i} \{f \in F : v \in f\}| = |\bigcap_{v \in R'_i} \mathcal{F}_v|
\end{aligned}$$

$\square$.

Chapter 5

Searching for an Approximate Solution

The previous chapter established the intractability of our problem. Fortunately, the full power of an optimal decomposition is not necessary in practice. A decomposition with a small number of regions is sufficient for practical purposes. We therefore developed and tested six different greedy approximation algorithms, divided into two classes, to realize the decomposition.

5.1 Limitations in the Real World

In real world visibility data, there are usually sampled poses at which the count of visible features is less than the required number k. This is generally the case for poses that lie close to walls and object boundaries, as well as for areas that are located far from any visible object and are therefore beyond the visibility range of most features. For this reason, the set of poses that should be decomposed into regions has to include only the *k-coverable poses,* i.e., those sampled poses whose visibility-set sizes are at least k.

5.2 Growing Regions From Seeds

The A.x class of algorithms decomposes pose space by greedily growing new regions from poses that are selected according to three different criteria. Once a new region has been started, each growth step consists of adding the pose in the vicinity of the region that has the largest set of visible features in common with the region. This growth is continued until adding a new pose would cause that region's visibility set to have a cardinality less than k.

The pseudocode of this class of algorithms is shown in Figure 5.1. Algorithms A.1, A.2 and A.3 implement each of three different criteria for selecting the pose from which a new region is grown. These three algorithms differ only in the implementation of line 3 (Figure 5.1):

- A.1 selects the pose $v \in U$ at which the least number of features is visible, i.e., $v = \arg\min_{u \in U} |\mathcal{F}_u|$.
- A.2 selects the pose $v \in U$ at which the greatest number of features is visible, i.e., $v = \arg\max_{u \in U} |\mathcal{F}_u|$.
- A.3 randomly selects a pose $v \in U$.

In cases of ties in line 3, they are broken randomly.

The set U, which is initialized in line 1 of the algorithm, contains the k-coverable poses which are still *unassigned* to some region. The set D that will contain the regions in the achieved decomposition, is also initialized to be empty. The main loop starts in line 2, and it is executed while there are unassigned poses. In lines 3 and 4, a pose v is selected from U according to the criteria given above, and a new region R containing only v is created. The loop that starts in line 5 adds neighboring poses to the region R, until the addition of a new pose would cause the set of features commonly visible in the region to have cardinality less than k. An iteration of this loop is realized in the

```
Input: world $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$
Output: decomposition $D$
 1: $U = \{v \in V : |\mathcal{F}_v| \geq k\}, D = \emptyset$
 2: while $U \neq \emptyset$ do
 3:   Select $v \in U$ (See text)
 4:   $R = \{v\}$
 5:   repeat
 6:     $W = \{u \in \{N_1(v) : v \in R\} - R : |\mathcal{F}_u \cap [\bigcap_{v \in R} \mathcal{F}_v]| \geq k\}$
 7:     if $W \neq \emptyset$ then
 8:       if $W \cap U \neq \emptyset$ then
 9:         $W := W \cap U$
10:       end if
11:       $u = \arg\max_{w \in W} |\mathcal{F}_w \cap [\bigcap_{v \in R} \mathcal{F}_v]|$
12:       $R = R \cup \{u\}$
13:     end if
14:   until $W = \emptyset$
15:   $U = U - R$
16:   $D = D \cup \{R\}$ (See Section 5.5)
17: end while
```

Figure 5.1: Algorithm A.x

following way: In line 6, the set W is formed by all poses u in the vicinity of the region R (i.e., the set of poses not in R that are at distance exactly 1 from a pose in R), such that u together with the poses in R commonly see at least k features.

In lines 8 through 10, if W contains unassigned poses, then W is restricted to those poses. Since the region R is going to grow with a pose selected from W, this step is intended to give priority to the growth of R with poses that still have not been assigned to any other region. In lines 11 and 12, the pose from W that together with the poses in R commonly sees the maximum number of features, is added to R. In case of a tie, it is broken randomly. Finally in lines 15 and 16, the poses in R are removed from the set of unassigned poses U, and the new region R is added to the decomposition set D.

5.3 Shrinking Regions Until k Features are Visible

Algorithms B.x and C take an incremental approach to defining the k features, starting with a large region that "sees" one feature, and iteratively shrinking the region as additional features (up to k) are added. The resulting region is added to the decomposition, a new region is started, and the process continued until the world is covered. These algorithms select as a new region the set of poses from which the most widely

```
Input: world $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$
Output: decomposition $D$
1: $U = \{v \in V : |\mathcal{F}_v| \geq k\}, D = \emptyset$
2: while $U \neq \emptyset$ do
3:    $R = U, L = \emptyset$
4:    for $i = 1$ to $k$ do
5:       $f = \arg\max_{\phi \in (F-L)} |\{v \in R : \phi \in \mathcal{F}_v\}|$
6:       $R = \{v \in R : f \in \mathcal{F}_v\}$
7:       $L = L \cup \{f\}$
8:    end for
9:    $R = \{v \in V : L \subseteq \mathcal{F}_v\}$
10:   $U = U - R$
11:   $D = D \cup \{R\}$ (See Section 5.5)
12:   Purge $D$ (See text)
13: end while
```

Figure 5.2: Algorithm B.x

visible feature, taken from a set $\mathcal{F}$, is seen among the poses that are not yet assigned to a region. Algorithms B.x and C differ in the criteria by which $\mathcal{F}$ is defined, as shown in Figures 5.2 and 5.3, respectively. In the case of algorithm B.x, $\mathcal{F}$ is just the set of all features, while algorithm C systematically selects as $\mathcal{F}$ the set of features commonly visible in a circular area centered at each pose $v \in V$. If the number of unassigned poses in the circular area is less than a certain fraction α of the size of the circular area, or the size of $\mathcal{F}$ is less than k, then no further processing is performed for pose v, and the next pose is processed.

The class B.x comprises two algorithms, B.1 and B.2, that differ only in their treatment of the decomposition D after adding to it a new region R (line 12). While Algorithm B.1 leaves D as it is, Algorithm B.2 greedily eliminates regions from D as long as the total number of poses that become unassigned, after the regions are removed from D, is less than the number of cells that the recently added region R has covered but were unassigned before.[1] This algorithm is adapted from the algorithm "Altgreedy", appearing in [17], where it is empirically shown to achieve very good approximation results for the set cover problem.

In line 1 of algorithm B.x, the sets U and D are initialized as in algorithm A.x. The

[1]Notice that this discarding rule ensures that the number of poses assigned to regions strictly increases with each iteration, so that the algorithm always terminates.

main loop starts in line 2, and it is executed while there are unassigned poses. In line 3, a new region R is initialized containing all unassigned poses, and the set L, which will contain features that all poses in the region commonly see, is initialized to be empty. Each iteration of the for-loop in lines 4 to 8 greedily selects the feature f not in L that is most widely visible in the region R, shrinks R to be formed only by those poses, and extends L to include f. At the exit of the for-loop, which is executed k times, R contains at least one pose, (since R entered the loop containing k-coverable poses), and the set L contains the k features that greedily decreased the least the size of the region R. In line 9, R is set to be the set of all poses (not only the unassigned ones) that at least see the k features in L. Finally, in lines 10 and 11, the poses in R are removed from the set of unassigned poses U, and the region R is added to the decomposition D.

Algorithm C, in line 1 initializes the set of unassigned poses U and the decomposition set D, in the same way that algorithms A.x and B.x do. In line 2, the variable r is assigned the maximum natural number such that at least half of the k-coverable poses have a r-neighborhood such that the k-coverable poses of the neighborhood commonly see at least k features. The main loop of this algorithm starts in line 3 and is executed for every pose $v \in V$. In line 4, $\mathcal{C}$ is assigned the set of unassigned poses in the r-neighborhood of v, and in line 5, $\mathcal{F}$ is assigned the set of features commonly visible in all poses of $\mathcal{C}$. The condition verified in line 6 is if the proportion of unassigned poses in the r-neighborhood of the current pose v is greater or equal than a constant α (defined by the user), and if the number of features commonly visible from all unassigned poses in the r-neighborhood of v is at least k. If this condition is true, then the process continues in a way similar to lines 3 to 11 of algorithm B.x: a for-loop greedily select the k "most visible features" from the set of unassigned poses, and finally a region containing all poses seeing those k features is created. The only difference is in the fact that in this algorithm, in the for-loop, the features are greedily selected from the set $\mathcal{F} - L$, while in algorithm B.x such features are selected from $F - L$. With this difference, algorithm C ensures that

```
Input: world $\langle G = (V, E), F, \{\mathcal{F}_v\}_{v \in V} \rangle$
Output: decomposition $D$
 1: $U = \{v \in V : |\mathcal{F}_v| \geq k\}, D = \emptyset$
 2: $r = \max\{\rho \in \mathbb{N} : |\{u \in U : |\bigcap_{w \in N_\rho(u) \cap U} \mathcal{F}_w| \geq k\}| \geq \frac{|U|}{2}\}$
 3: for all $v \in V$ do
 4:   $\mathcal{C} = N_r(v) \cap U$
 5:   $\mathcal{F} = \bigcap_{u \in \mathcal{C}} \mathcal{F}_u$
 6:   if $\frac{|\mathcal{C}|}{|N_r(v)|} \geq \alpha$ and $|\mathcal{F}| \geq k$ then
 7:     $R = U, L = \emptyset$
 8:     for $i = 1$ to $k$ do
 9:       $f = \arg\max_{\phi \in (\mathcal{F} - L)} |\{v \in R : \phi \in \mathcal{F}_v\}|$
10:       $R = \{v \in R : f \in \mathcal{F}_v\}$
11:       $L = L \cup \{f\}$
12:     end for
13:     $R = \{v \in V : L \subseteq \mathcal{F}_v\}$
14:     $U = U - R$
15:     $D = D \cup \{R\}$ (See Section 5.5)
16:   end if
17: end for
```

Figure 5.3: Algorithm C

the for-loop will exit with a region R that has a minimum number (which depends on r and α) of newly assigned poses that are in $N_r(v)$. This algorithm may terminate leaving some poses unassigned to a region. A process (not shown in the pseudocode) is therefore applied to cover those areas. Such a process is equivalent to Algorithm B.1, but with line 1 making U equal to the set of unassigned poses.

Algorithms B.x and C are based on the assumption that the set of poses from which each feature is visible form a connected region, and that the intersection of such feature visibility areas is also a connected region. This assumption is true if all feature visibility areas are simple and convex. In our experiments with real data, we have observed that the feature visibility regions are not always convex or connected, and they sometimes have some small holes. Since the number of extracted features is quite large, we can afford to exclude from the decomposition process those features with significant holes in their visibility regions. Visibility regions with many concavities can also be trimmed to the set of poses that have a more or less convex shape. Also, if a visibility region has more than one connected component, each component of significant size can be considered to be the visibility region of a different feature.

5.4 Elimination of Redundant Regions

All algorithms, except B.2, can terminate with a solution that is not minimal. Redundancy is therefore eliminated from their solutions by discarding regions one by one until a minimal solution is obtained. This process greedily selects for elimination the region R from the solution D with the largest *minimum-overlapping-count* ω value, where $\omega = \min\{|\{R' \in D : v \in R'\}| : v \in R\}$, i.e., it is the minimum number of regions that overlap at a pose contained in the region. The worst-case running time complexity of algorithm A.x is bounded by $O(|V|^2|F|)$, while algorithms B.x and C are bounded by $O(k|V|^2|F|)$.

5.5 Relaxing the Requirement for a Complete Decomposition

A decomposition that tries to cover all k-coverable poses may include a large number of regions in total, since many regions will serve only to cover small "holes" that could not be otherwise covered by larger regions. These holes generally lie in areas for which the size of the visibility-set is very close to k, leaving very few features to choose from. In order to avoid the inclusion of regions that are only covering small holes, our implementations of the algorithms add a region to the decomposition only if its number of otherwise uncovered poses is greater than a certain value σ.[2]

[2]The presence of a few small holes does not prevent reliable navigation. In general, whenever the robot is at a point for which the number of visible features is less than k, advancing a short distance in most directions will get it to a point that is assigned to some region.

Chapter 6

Off-line Exploration

The off-line exploration stage consists of the following tasks:

- Image sampling at discrete locations on the floor.
- Computation of camera position and orientation at each sampled location.
- Extraction of image features and their classification according to what scene feature gave rise to them, and construction of a feature visibility vector for each sampled position, specifying the scene features visible from that location.

In this chapter, we suggest methods to perform the last two tasks. We refer the interested reader to [38] for a discussion of methods for automatically performing the image collection task and links to related work on this topic.

6.1 Pose Estimation of Collected Images

View-based localization consists of estimating the current robot's position given image point correspondences between the image currently seen by the robot and two model views that were acquired at known positions and orientations. The set of model views used in the on-line localization stage are a subset of the images collected during the offline exploration phase. The particular selection of images that will be used as model views is done during the landmark database construction phase. In order to build the database of landmarks, the database construction phase also needs to know the topology of the pose space of sampled images, and the set of scene features visible at each pose. These requirements of the successive stages of the view-based framework justify the need for computing the position and orientation at which each sampling image has been acquired, and the generation of a feature visibility vector at each sampling pose.

Several methods have been proposed to compute the camera pose for each of the sampled images. One is simply using the robot's odometry to measure pose, but as it has been mentioned in the introduction chapter to this thesis, this approach has the problem that the error in the estimate grows limitless. Another possibility is using range data, but this approach only works in restricted environments, generally indoors, where walls and obstacles are within the sensor range. Interferences are also a cause of measurement error for this type of sensor. Collaborative exploration has also been suggested by some authors, e.g., [33, 32], in which two or more robots that see each other act together to reduce odometry errors.

All these approaches have the problem that either their measurements are unreliable, the proposed methods are restricted to particular environments, or more than one robot or non-standard sensors are required for the task. In this section, we propose a method to estimate the camera pose using nothing else but the set of collected images, when the camera calibration parameters are known.

A set of approaches that is related to the problem of computing the pose of the robot during image collection is that of Simultaneous Localization and Mapping (SLAM). In SLAM, a map of the environment is built on-line at the same time that robot localization is performed. Most of the work available in this area addresses this problem by performing data fusion (generally via Kalman filtering) of the information provided by all the available robot sensors. These approaches solve the problem of map construction from an object-based point of view, where the 3D position of each landmark is estimated. Recent work in this area can be found in [11, 35, 12, 32].

In our case, we need to solve a less ambitious problem, namely, an off-line estimation of the locations and orientations from which each image was taken. Since, in images taken from close locations, there is redundant visual information, we can take advantage of that redundancy to find the best set of camera poses that minimizes the total residual error. Here, we propose a rather simple methodology to construct the map of sampled images without the need to compute the 3D positions of the landmarks. Our proposal is based on a weighted least squares approximation of the camera pose of each collected image, by minimizing the total residual error when taking into account all the landmark data from all images.

6.1.1 Noise Filtering

In chapters 3 and 5, we showed how the robot environment can be decomposed into a set of overlapping regions, such that a certain number (k) of features are commonly visible from any point in a region. An extra benefit from the clustering of features into regions that we can take advantage of is the fact that the locations of the k features in all the images of the region are correlated. Therefore, we can exploit lower rank approximation techniques to de-noise the location measurements of features in each region. Stewart [41] presents an analysis of the effects of noise on the singular values and singular vectors of a matrix. It is shown that small singular values tend to grow under perturbation.

Based on the fact that in the absence of noise, a matrix of highly correlated measurements is rank deficient and noise in the measurements has the effect of increasing the rank of such a matrix, Muijtjens *et al.* [31] used lower rank approximation to de-noise image data measurements. They were interested in de-noising image data used to perform a 3D reconstruction of the motion of a wall of a live heart. They tracked radiopaque markers implanted on a live heart through a sequence of images acquired in a time interval. A matrix was then created containing the measured image locations of the markers. Each row of the matrix corresponded to a particular marker, and each column corresponded to the image taken at a particular instant in time. Here, we follow their same idea, but instead of dealing with images of a dynamic object acquired over time, we are working with images of a rigid 3D object (the "landmark" formed by the k features) taken from different locations and orientations. Our problem is equivalent to theirs in the sense that the images of the features could be thought as being acquired from a fixed camera with the 3D structure of scene features changing pose through time.

Here is the de-noising procedure. Assume the number of images in a given region is r. We need to create a block matrix P such that $P_{ij} = \overrightarrow{p}_i^j$, with $\overrightarrow{p}_i^j$ being the 2D location of the i-th feature in the j-th image, where $i = 1, \ldots, k$ and $j = 1, \ldots, r$. Formally, let

$$U \cdot \begin{pmatrix} s_1 & 0 & \ldots & 0 & 0 \\ 0 & s_2 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & s_{r-1} & 0 \\ 0 & 0 & \ldots & 0 & s_r \end{pmatrix} \cdot V^t = P$$

be the singular value decomposition of P. The "de-noised" version of P will be the matrix

$$\widetilde{P} = U \cdot \begin{pmatrix} s_1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & s_{r-h} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{pmatrix} \cdot V^t,$$

where h determines the rank reduction.

```
Input: Set of images $\{I^i\}_{i=1,\ldots,n}$
Output: A set $\mathbf{Y}$ of indices of pairs of images for which motion parameters were computed, a set of rotation matrices $\{R^{ij}\}_{(i,j)\in\mathbf{Y}}$, and a set of unitary translation vectors $\{\overrightarrow{\hat{T}^{ij}}\}_{(i,j)\in\mathbf{Y}}$ encoding the camera motion between pairs of views.
1: $\mathbf{Y} = \emptyset$
2: for i = 1, ..., n do
3:   Compute set of feature points $\mathcal{F}_{v_i}$ from image $I^i$
4: end for
5: for $1 \le i < j \le n$ do
6:   Compute correspondences $\mathcal{F}_{v_i \wedge v_j} = \mathcal{F}_{v_i} \cap \mathcal{F}_{v_j}$ between features in images $I^i$ and $I^j$
7:   if $|\mathcal{F}_{v_i \wedge v_j}| \ge m$ then
8:     Estimate motion parameters $R^{ij}$ and $\overrightarrow{\hat{T}^{ij}}$ using the locations of $\mathcal{F}_{v_i \wedge v_j}$ in $I^i$ and $I^j$.
9:     $\mathbf{Y} = \mathbf{Y} \cup \{(i,j)\}$
10:  end if
11: end for
```

Figure 6.1: Algorithm to compute the motion parameters between all pairs of collected images.

6.1.2 Motion Parameters Between Views

We will first use weighted least squares to find the set of rotations between all pairs of views that minimize the total rotation error of the whole set of images. Then we will use weighted least squares again to find the set of camera locations that minimizes the total translational error of the complete set. To apply these ideas we first need, given a set of calibrated sampled images $\{I^i\}_{i=1,\ldots,n}$ corresponding to sampling positions $v_1, \ldots, v_n$, respectively, to estimate the rotation and translation of the camera between pairs of images. Depending on the method used and the desired accuracy, there is a minimum number m of point correspondences needed to estimate the motion parameters. Algorithm 6.1 computes and stores the motion parameters between all pairs of views that commonly see at least m feature points. In line 8 of this algorithm, a robust method has to be employed to estimate the motion parameters (a rotation matrix R^{ij} and unitary translation vector $\overrightarrow{\hat{T}^{ij}}$) in the presence of outliers, i.e., wrong point correspondences between views. In chapter 7, we present several methods to estimate these parameters from the locations of feature point correspondences in both views.

6.1.3 Best Fit of Rotations Between Views

The set of estimated parameters, although redundant, is also inconsistent due to the presence of noise and outliers in the measurements. In this section, we present a method for computing an estimate of the camera rotations that minimizes the total residual error. The set of resulting rotations that minimizes the error will be the solution to a linear system of equations with its elements in the set of *quaternions*, subject to a certain constraint.

Quaternions are a non-commutative extension of the complex numbers. A quaternion q can be represented by a real number α and a vector $\overrightarrow{v} \in \mathbb{R}^3$, and is noted as $q = \alpha + \overrightarrow{v}$ or $q = (\alpha, \overrightarrow{v})$. α is called the *real part* of q, and $\overrightarrow{v}$ is called the *imaginary part* of q. The *conjugate* of a quaternion q is defined as $\overline{q} = \alpha + (-\overrightarrow{v})$. If $q_1 = \alpha_1 + \overrightarrow{v}_1$ and $q_2 = \alpha_2 + \overrightarrow{v}_2$ are two quaternions, the sum and product between quaternions are defined as $q_1 + q_2 = (\alpha_1 + \alpha_2) + (\overrightarrow{v}_1 + \overrightarrow{v}_2)$, and $q_1 \cdot q_2 = (\alpha_1\alpha_2 - \overrightarrow{v}_1^t \overrightarrow{v}_2) + (\alpha_1 \overrightarrow{v}_2 + \alpha_2 \overrightarrow{v}_1 + \overrightarrow{v}_1 \times \overrightarrow{v}_2)$, respectively. The *norm* of q is defined as $\|q\| = \sqrt{a\overline{a}} = \sqrt{\overline{a}a} = \sqrt{\alpha^2 + \overrightarrow{v}^t\overrightarrow{v}}$.

The set of quaternions $\mathbb{H}$ with the sum and product defined over them $\langle \mathbb{H}, +, \cdot \rangle$ form a division algebra, that is, a non-commutative field. Since $\langle \mathbb{H}, +, \cdot \rangle$ is a division algebra, and in particular a non-commutative ring, then $\langle \mathbb{H}^n, +, \cdot \rangle$ is a module over $\mathbb{H}$. A module over a ring is a more general algebraic structure than a vector space over a field, in which most of the desirable properties of linear algebra on vector spaces also work. This means that we can solve systems of linear equations with its variables and vectors in $\mathbb{H}^n$ using the same mathematical tools used to solve systems of equations over $\mathbb{R}$ or $\mathbb{C}$, taking care of replacing sum and product by the appropriate definition of those operations in $\mathbb{H}$. And since also $\langle \mathbb{H}, +, \cdot \rangle$ is isomorphic to the subspace

$$\left\langle \{ \begin{pmatrix} a & -b & d & -c \\ b & a & -c & -d \\ -d & c & a & b \\ c & d & b & a \end{pmatrix} \in \mathbb{R}^{4\times 4} \}, +, \cdot \right\rangle,$$

(through isomorphism $h : \mathbb{H} \rightarrow \mathbb{R}^{4\times 4}$ defined by

$$h((a,(b,c,d)^t) = \begin{pmatrix} a & -b & d & -c \\ b & a & -c & -d \\ -d & c & a & b \\ c & d & b & a \end{pmatrix}),$$

all matrix operations involving quaternions can be realized using block matrices of reals.

An interesting fact that we will exploit is that rotations can be represented by quaternions of norm 1. This means that there is an isomorphism between the set of rotations in 3D and the set of quaternions lying on the four-dimensional unitary sphere S^3. This is a continuous mapping, i.e., close points on the sphere correspond to close rotations and vice versa. The rotation of angle θ around the axis $\overrightarrow{v} \in \mathbb{R}^3$ is represented by the unitary quaternion:

$$q_v = \cos\tfrac{\theta}{2} + \sin\tfrac{\theta}{2}\tfrac{\overrightarrow{v}}{\|\overrightarrow{v}\|}.$$

To rotate a 3D vector $\overrightarrow{x}$ using the quaternion q_v, we must compute $q_v\widetilde{x}\overline{q_v}$, where $\widetilde{x}$ is a quaternion with real part zero and imaginary part $\overrightarrow{x}$. The benefit of representing 3D rotations using quaternions is that only one non-linear constraint has to be enforced on a quaternion in order for it to correspond to a rotation, i.e., its norm being 1. When rotations are represented using 3×3 matrices of reals, there are six non-linear constraints that a matrix has to satisfy in order to correspond to a rotation, i.e., the orthonormality of its rows and columns. Since our problem can be posed as the solution to a linear system of equations in which the unknowns are rotations, using quaternions means just solving the system subject to the constraint that each unknown has norm 1.

We then propose the following method to find an estimate of the orientation of the cameras during the image collection phase that minimizes the total residual error of all the computed rotations between pairs of views. Let q^{ij} be a unitary quaternion encoding the 3D rotation R^{ij} between cameras i and j, obtained, for example, from the factorization of the essential matrix E^{ij} [28]. Let q^i be a unitary quaternion representing the orientation of the i-th camera during the image collection phase. We wish to obtain an estimate of q^i, for $i = 1, \ldots, n$. From each measured rotation q^{ij} between cameras, we can create the equation

$$q^{ij} \cdot q^i - q^j = 0 \tag{6.1}$$

We create a vector of unknowns $\overrightarrow{q} = (q^1, \ldots, q^n)^t \in \mathbb{H}^n$, and a matrix Q with all the equations generated from each R^{ij}, weighted by a scalar inversely proportional to the uncertainty of the estimation of R^{ij}. (See next section for an explanation of how to compute this weight.) The solution to our problem is therefore the minimizer vector $\overrightarrow{q} \in \mathbb{H}^n$ of $\|Q\overrightarrow{q}\|^2$ subject to the constraints $\|q^i\| = 1$, for $i = 1, \ldots, n$. Since the constraints are in the Cartesian product of the four-dimensional unit sphere, they form a compact set. (It is closed because each component is defined by an equality of a continuous function, and it is bounded because each component is a unit sphere.) Therefore, since $\|Q\overrightarrow{q}\|^2$ is a continuous function, then it must achieve its minimum on the constraint set.

An iterative minimization method is not guaranteed to converge to a global minimum in our problem, due to the non-linearity of the function being optimized. However, the minimization process would at least be guaranteed to improve on the current solution, and if we could start from a fairly good initial guess of the solution $\overrightarrow{q}$, it could eventually converge to the global minimum. We actually can construct such a fair initial guess from the measurements we have for the rotations between cameras. We can set an arbitrary orientation for the first camera, e.g., $q^1 = 1$, and then set all other orientations in terms of this one by means of an arbitrary selection of a consistent subset of constraints, as defined by Equation 6.1, until all orientations q^i have been initially assigned.

6.1.4 Best Fit of Translations Between Views

Each estimated translation $\overrightarrow{\hat{T}^{ij}}$ is expressed in the coordinate frame of camera i. We need to express all translations using the same coordinate frame, e.g., the one corresponding to camera 1. This can be accomplished by $\overrightarrow{T^{ij}} = R^{1i^t}\overrightarrow{\hat{T}^{ij}}$. If the measurements had been free of error, then the unitary translation vector $\overrightarrow{T_{ij}}$ has the direction of translation between locations v_i and v_j. That is, there exists a magnitude $a_{ij} \in \mathbb{R}$, such that $a_{ij}\overrightarrow{T^{ij}} = \overrightarrow{v_j} - \overrightarrow{v_i}$. We would like to recover a good approximation of these image locations from the noisy measurements of the translation vectors that we have. So we propose to use a weighted least squares approach to do this, taking advantage of the redundancy present in the relationship between all the measured translations between images.

For each $(i, j) \in Y$ we create the vector equation

$$\overrightarrow{v_j} - \overrightarrow{v_i} - a_{ij}\overrightarrow{T^{ij}} = \overrightarrow{0}, \tag{6.2}$$

which gives rise to three linear equations, one per vector component. We choose one of the image locations, e.g., $\overrightarrow{v_1}$, as the center of our coordinate frame, i.e., we fix $\overrightarrow{v_1} = \overrightarrow{0}$. Our unknowns are then the locations $\overrightarrow{v_i}, 1 < i \leq n$, and the magnitudes of the translation vectors $a_{ij}, (i, j) \in \mathbf{Y}$. We set the unknowns as components of a vector

$$\overrightarrow{x} = (v_2^x, v_2^y, v_2^z, v_3^x, v_3^y, v_3^z, \ldots, v_n^x, v_n^y, v_n^z, a_{y_1}, \ldots, a_{y_r})^t \in \mathbb{R}^{(3(n-1)+|\mathbf{Y}|)},$$

with $\mathbf{Y} = \{y_1, \ldots, y_r\}$ From the system of equations 6.2, we create a matrix A of size $(3|\mathbf{Y}|) \times (3(n-1) + |\mathbf{Y}|)$, such that rows $3k + 1, 3k + 2$, and $3k + 3$ of A encode the three components of vector equation $(i, j) = y_{k+1}, k = 0, \ldots, r - 1$. The linear system to be solved becomes then $A\overrightarrow{x} = \overrightarrow{0}$.

Knowledge of the uncertainty present in the the estimation of T^{ij} from the image point correspondences can be used to weight the equations in the system, giving a heavier weight to those equations that come from more reliable measurements. The uncertainty can be computed from some measure of the residual error in the estimation of the translations from the inliers (i.e., the point correspondences that *agreed* on the value of the estimated translation), and from the proportion of inliers in the set of point correspondences. If $w_{ij}, (i,j) \in \mathbf{Y}$, are weights inversely proportional to the uncertainty in the estimation of T^{ij}, the solution to the weighted system is the vector $\overrightarrow{x}$ that minimizes $\sum_{k=1}^{3r} w_{y_{\lceil k/3 \rceil}} (A_k \overrightarrow{x})^2$, subject to the constraint $\|\overrightarrow{x}\| = 1$, where A_k is the k-th row of matrix A. The solution to this weighted least squares system lies in the null space of $\widehat{A} = WA$, where W is a diagonal matrix such that $W_{3k+1,3k+1} = W_{3k+2,3k+2} = W_{3k+3,3k+3} = \sqrt{w_{y_{k+1}}}$, $k = 0, \ldots, r-1$.

The null space of $\widehat{A}$ will have dimension one in the case of the existence of a unique solution to the system, up to a scale factor. The vector spanning the null space of the matrix can be obtained from its singular value decomposition $\widehat{A} = UDV^t$, as the last column of V corresponding to the smallest singular value in the diagonal of D. Although the matrix of the system is large, it is sparse, since each equation only relates unknown position between two images that commonly see a certain minimum number of 3D points. Yet this will only happen between images acquired from close locations and looking at the same scene. Therefore the matrix can be built in a band diagonal shape, and sparse linear algebra algorithms can be employed to solve the system [1].

An incremental method to compute the translation between cameras is the following: Start by estimating the translation direction between two arbitrary cameras, e.g., $\overrightarrow{T^{1,2}}$ between cameras 1 and 2, and create the set $\mathcal{I} = \{1, 2\}$ containing the indexes of the cameras for which translations were already computed. Then, iteratively compute a set of translations $\mathbf{T} = \{\overrightarrow{T^{k,i_1}}, \ldots, \overrightarrow{T^{k,i_h}}\}$ between a new camera $k \notin \mathcal{I}$ and cameras $i_1, \ldots, i_h \in \mathcal{I}$, such that the dimension of the space generated by the vectors in $\mathbf{T}$ is at least 2; and then add k to $\mathcal{I}$.

It is a sufficient condition for the existence of a unique solution, up to a scale factor, (in the absence of noise), to generate the system of equations from the translations between cameras computed in this incremental way. This can be proven inductively:

It is obvious that for the base case of two cameras related by a translation, there is a unique solution, up to scaling, for the magnitude of the translation and location of the camera centers. Inductively, assume the property is true for the system of equations corresponding to the structure $\mathcal{S}$ of translations and camera locations so far computed, and now at least two non-parallel translations are computed between a new camera k and cameras i, j in $\mathcal{S}$. A change in scale to any of the translations or camera locations in $\mathcal{S}$, by inductive hypothesis, will proportionally scale the distance between the locations $\overrightarrow{v_i}$ and $\overrightarrow{v_j}$, and therefore the vertex $\overrightarrow{v_k}$ and the other two sides of the triangle $\overrightarrow{v_i}\ \overset{\triangle}{\overrightarrow{v_k}}\ \overrightarrow{v_j}$, (i.e., the translations with directions $\overrightarrow{T^{k,i}}$ and $\overrightarrow{T^{k,j}}$), will have to be proportionally scaled in order to preserve their directions. Analogously, if the camera location $\overrightarrow{v_k}$ or the translations with directions $\overrightarrow{T^{k,i}}$ or $\overrightarrow{T^{k,j}}$ change their magnitude, then the triangle $\overrightarrow{v_i}\ \overset{\triangle}{\overrightarrow{v_k}}\ \overrightarrow{v_j}$ will have to be proportionally scaled in order to preserve the directions of its sides, and hence the camera location $\overrightarrow{v_k}$ and the translations $\overrightarrow{T^{k,i}}$ and $\overrightarrow{T^{k,j}}$ will be scaled, and also the distance between $\overrightarrow{v_i}$ and $\overrightarrow{v_j}$ will change proportionally, which will in turn scale all translations and camera locations in $\mathcal{S}$. $\Box$.

As evidence that the way that this method computes the translations is not a necessary condition for the existence of a unique solution, see Figure 6.2. In the graph shown in this figure, the nodes represent camera locations, and the edges represent translations between cameras. The translations of this example cannot be computed in the suggested incremental way, however, there is a unique solution for the translation magnitudes and camera locations, up to a scale factor. It is easy to see that if the magnitude of one of the edges is changed, then all other edges have to change their magnitudes proportionally, in order to preserve their directions and still remain with the same adjacency relationship.

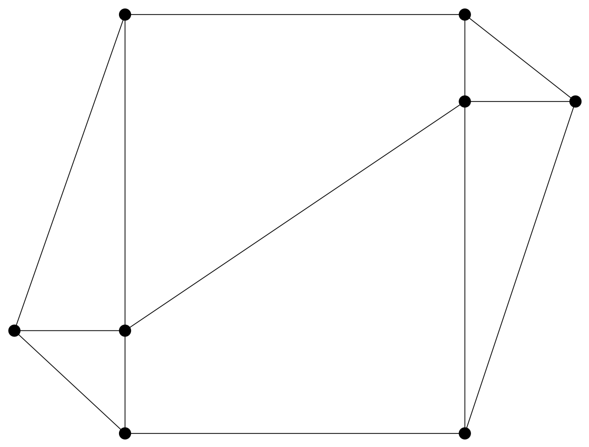


Figure 6.2: Example of a camera layout that has a unique solution, up to a scale factor, for camera locations (nodes) and magnitude of the translations (edges), but cannot be computed using the suggested incremental method.

6.2 Generation of the Feature Visibility Vectors

The feature visibility vector of a pose specifies the scene features that are visible from the image taken at that pose. In order to build these vectors, we need to create classes of images features. Each class will group the image features that correspond to the same scene feature. In order to realize this classification, we have to be able to compute reliable correspondences between image features of different views, and to discard the image features that are ambiguous, that is, those that may seem to belong to more than one class. This section proposes a method to do these tasks.

6.2.1 Computing Feature Correspondences

For a particular family of image features to be useful, a function $\hat{\delta}(\cdot, \cdot)$ has to be provided to measure the "similarity" between two features. The similarity accounts for how likely the two features are of being images of the same scene feature, i.e., the smaller the value of $\hat{\delta}$ is, the more likely the same scene feature gave rise to both image features. (Ideally $\hat{\delta}$ should be a *distance* function in a metric space; in particular it should be positive, commutative, $\hat{\delta}(f, f) = 0$ for all image features f, and the triangular inequality should hold.) For example, in the case of Lowe's SIFT features [25], and the phase-based features of Carneiro and Jepson [8], an image feature is implemented as a vector in a high dimensional space. Similarity between SIFT features is measured by the Euclidean distance between feature-vectors, while for phase-based features, similarity is measured using phase-correlation.

We want to group the image features f into classes $\overline{f}$, such that there is exactly one class per scene feature, and such that all image features due to that scene feature belong to that class. However, not only do we want to separate image features into classes, we likewise want to discard all the features that are not distinctive enough, i.e., that can be easily confused as belonging to a different class. This, in turn, suggests that *all* image features belonging to a class for which there is at least one ambiguous feature have to be discarded as well. We take this drastic approach, because during on-line localization, we will match visible features to those in the database and we want to keep only one exemplar image feature representing the class.

Other less strict approaches to matching are possible, e.g., instead of just retaining one image feature per class, the database can keep a representative collection of the various appearances that the scene feature has in pose space. The correspondence between image features in the sample images can be done via tracking the features between adjacent viewpoints, in which their changes in appearance are small. Such an approach will achieve larger areas of visibility for each scene feature than the method proposed here, but it will be more susceptible to incorrect matchings during on-line localization. We aim to study these ideas as future work to this thesis.

In the proposed method that follows, we will assume that there is exactly one image per sampling position. If two or more images are available at a particular position, they should all be integrated into a unique panorama, to guarantee that for each scene feature, at most one corresponding image feature is visible at each sampling position. Let $G = (V, E)$ be an undirected planar graph describing the topology of the set of sampling poses, with each pose being a node of the graph. Let $\hat{F}$ be the set of all image features. Let $\hat{\mathcal{F}}_v \subset \hat{F}$ be the set of image features visible from the image acquired at pose $v \in V$. Two different image features from the same image have to necessarily correspond to different scene features. Then it is reasonable to specify that the equivalence relation between image features satisfy the *local uniqueness* property:

$$(\forall v \in V)(\forall f, g \in \hat{\mathcal{F}}_v) : \overline{f} = \overline{g} \iff f = g, \tag{6.3}$$

where the notation $\overline{f}$ means "the class of f", and $=$ refers to identity. If more than one scene feature having the same appearance are visible from the same pose, it is possible to have repeated identical image features in the image. And since we will enforce the above property on the equivalence relation, those features will be excluded from the relation, although they do correspond to *different* scene features. That is actually something good, since in our localization framework, we are interested only in image features corresponding to scene features that are rather unique.

We will now define a relation between image features that uses the notion of similarity defined by $\hat{\delta}$ and with the property specified by (6.3). Given a threshold $\tau \in \mathbb{R}_{>0}$, we define the relation $=_F$ between two image features $f \in \hat{F}$ and $g \in \hat{\mathcal{F}}_w$ by

$$f =_F g \iff \hat{\delta}(f, g) \leq \tau \ \wedge \ g = \operatorname*{arg\,min}_{\phi \in \hat{\mathcal{F}}_w} \hat{\delta}(f, \phi) \tag{6.4}$$

This is not an equivalence relation over all $\hat{F}$, but just a nearest-neighbor relation. Although it is clearly reflexive, it is not symmetric or transitive, in general. We will turn it into an equivalence relation by restricting it to the maximal subset of image features $\widetilde{F} \subset \hat{F}$ for which the symmetry and transitivity properties hold. Clearly, the features that break the property of symmetry and/or transitivity are ambiguous in the sense that they can be confused as belonging to a different class than the one that they actually belong to, depending on the "direction" in which the relation is verified, or to what particular feature in the class they are compared to. We therefore are interested in using only the image features in $\widetilde{F}$ for localization.

Although the set $\widetilde{F}$ will contain only those image features that are distinctive enough *among all images collected during the off-line phase*, there is no guarantee that symmetry will hold when a feature $f \in \widetilde{F}$ is matched, using $=_F$ in one direction, to an image feature g visible in a novel view during localization. However, since we only retain in $\widetilde{F}$ the features that *statistically* belong to clearly defined classes among all image features visible in pose space (where our sample was the set of images collected during the off-line exploration phase), it is more likely that g will belong to the same class as f. (Actually, *for the purpose of matching verification* during on-line localization, we could even retain all image features visible in each model view, to use them to verify the symmetry of $=_F$ between f and g. If symmetry does not hold, then g should be discarded as ambiguous.)

Algorithm 6.3 computes $\widetilde{F}$ by verifying the symmetry of $=_F$ between image features from different poses, and by computing the transitive closure of $=_F$ over $\hat{F}$. Whenever two features are found such that $=_F$ is not symmetric, all the elements in the classes to which they both belong are excluded from $\widetilde{F}$. Finally, after transitive closure, all elements in the classes of features for which 6.3 does not hold are excluded from $\widetilde{F}$. The algorithm takes as input a collection $\{\hat{\mathcal{F}}_v\}_{v \in V}$ of sets of image features extracted from the images acquired at each pose, and outputs a collection of *feature class identifiers* $\{\mathcal{F}_v\}_{v \in V}$, where $\mathcal{F}_v$ is the set of class identifiers of the image features in $\widetilde{F}$ that are visible from the image taken at pose v.

To soften the drastic nature of the suggested approach to compute the equivalence classes, we introduced a parameter $\gamma \in \mathbb{Z}_{>0}$ in our algorithm. This parameter specifies to what extent violations to the symmetry and transitivity properties of the equivalence relation are going to be checked. The symmetry and transitive closure between features is not verified *for all pairs of poses*, but this verification is realized only between the poses that are separated at most by a distance γ, where the distance between poses is

```
Input: $\{\hat{\mathcal{F}}_v\}_{v \in V}$
Output: A set $\widetilde{F}$, over which $=_F$ is a relation of equivalence; and a collection $\{\mathcal{F}_v\}_{v \in V}$, where
$\mathcal{F}_v = \{class(f) : f \in \hat{\mathcal{F}}_v \cap \widetilde{F}\}$.
 1: $\perp = \hat{F}$
 2: $classesToExclude = \emptyset$
 3: for all $v, w \in V : \delta(v,w) \leq \gamma$ do
 4:   for all $f \in \hat{\mathcal{F}}_v$ do
 5:     if $f \in \perp$ then
 6:       $class(f) =$ new class
 7:       $\perp = \perp - \{f\}$
 8:     end if
 9:     $f' = \arg\min_{\phi \in \hat{\mathcal{F}}_w} \hat{\delta}(f, \phi)$
10:     if $\hat{\delta}(f', f) \leq \tau$ then
11:       if $f = \arg\min_{\phi \in \hat{\mathcal{F}}_v} \hat{\delta}(f', \phi)$ then
12:         if $f' \in \perp$ then
13:           $class(f') = class(f)$
14:           $\perp = \perp - \{f'\}$
15:         else if $class(f') \neq class(f)$ then
16:           if $class(f) \in classesToExclude \vee class(f') \in classesToExclude$ then
17:             $classesToExclude = classesToExclude - \{class(f), class(f')\}$
18:             Merge $class(f)$ and $class(f')$ into a unique class.
19:             $classesToExclude = classesToExclude \cup \{class(f)\}$
20:           else
21:             Merge $class(f)$ and $class(f')$ into a unique class.
22:           end if
23:         end if
24:       else
25:         $classesToExclude = classesToExclude \cup \{class(f)\}$
26:         if $f' \in \perp$ then
27:           $class(f') =$ new class
28:           $\perp = \perp - \{f'\}$
29:         end if
30:         $classesToExclude = classesToExclude \cup \{class(f')\}$
31:       end if
32:     end if
33:   end for
34: end for
35: for all $v \in V$ do
36:   $classesToExclude = classesToExclude \cup \{class(f) : f \in \hat{\mathcal{F}}_v \wedge (\exists g \in \hat{\mathcal{F}}_v) : f \neq g \wedge class(f) = class(g)\}$
37: end for
38: $\widetilde{F} = \{f \in \hat{F} : class(f) \notin classesToExclude\}$
39: for all $v \in V$ do
40:   $\mathcal{F}_v = \{class(f) : f \in \hat{\mathcal{F}}_v \cap \widetilde{F}\}$
41: end for
```

Figure 6.3: Algorithm to compute image features equivalence.

measured in terms of the shortest distance function $\delta(\cdot,\cdot)$ between nodes in the graph G. The greater γ is, the lower the probability of having a class with ambiguous image features in $\widetilde{F}$ at the end of the algorithm's execution, with that probability being zero if γ is greater or equal to the maximum distance between two poses in pose space. Actually, to achieve an exact solution, it suffices that γ is as large as the radius of the maximum feature visibility region. On the other hand, if $\gamma = 1$ is used, the equivalence classes are computed by relating similar features from images acquired at adjacent poses (i.e., *tracking* image features between adjacent poses), with the symmetry and transitivity checked only for features extracted from images acquired from sets of poses in which each pose is at distance 1 from any other pose in the set.

This more relaxed approach to compute the equivalence classes is reasonable, because we expect the image features to tend to preserve their appearance in images acquired from a set of contiguous poses. There is a trade-off between the computational cost and the accuracy of the solution, since the approximate computation is $O(\frac{|V|}{\gamma^2})$ times faster than the exact one. In our Results chapter, we applied this algorithm and obtained excellent results. We used $\gamma = 5$ to compute the correspondences between SIFT features extracted from images acquired on a 6m by 3m sampling grid at intervals of 25 cm, i.e., a lattice of 25 by 12 poses. In a visual analysis of the results, we were able to determine that only one class, out of a total of 897, contained an incorrectly classified image feature from one particular image out of a total of 1152 images.

6.2.2 Retaining Only Good Features

If the number of available features is large, which is generally the case, we can afford the luxury of discarding features that are not suitable for our needs. Once the classification of image features by scene feature correspondence has been performed, not all of the obtained features have desirable properties when using our proposed method to build the landmark database. A bad feature is one which is not widely or consistently visible, i.e., either it is not visible from a large number of *contiguous* poses, or its region of visibility is too narrow, it has many holes, or it is fragmented into several non-connected components.

We have to avoid using features with these bad visibility properties in our decomposition method. Since a region in the decomposition is obtained from the intersection of the visibility regions of individual features, the result of the intersection of a region with the mentioned characteristics with another arbitrary region, is a region with generally the same bad characteristics, which is not good for our proposed method of navigation. Features with almost convex and wide visibility regions should be preferred over others to be used in the region decomposition stage.

Chapter 7

On-line Localization

This chapter explains how to perform the on-line localization stage of the view-based navigation framework. We discuss how to use the decomposition of the environment in regions that commonly see k features to achieve efficient robot localization. The localization stage involves two main tasks: the robot needs to recognize the features that constitute a landmark in the current view, and based on the recognized landmark, it has to compute its pose in the world. We start this chapter with a high-level explanation of the steps involved in the landmark recognition and pose computation tasks, and then we discuss in depth different techniques to compute the robot's position and orientation.

7.1 Recognition of Landmarks

The landmark recognition in the current view can be performed in two possible scenarios: it is either the case that the robot knows in what region it is located, or it simply does not, i.e., it is starting navigation, or it was *kidnapped*. If the robot does not know in what region it is, the features visible in the current image can vote for the regions they belong to, if any, according to a membership relationship computed off-line. The region in which the robot is will be assumed to be one of those with at least k votes.

On the other hand, when the robot is navigating inside a known region, the k features that form the landmark commonly visible from all locations inside that region can be easily tracked between the images that the robot sees. If at any point the expected k features are not all visible in the current image, this may indicate that the robot has left the region in which it was navigating and is entering a new region. In that case, using a graph map of the regions, and knowing the direction of movement of the robot, we can constrain the number of possible regions to which the robot might have moved. In the worst case, the same voting scheme mentioned above, using all regions, can be applied.

The region overlapping can also be exploited to avoid the voting scheme, when the robot is approaching the boundary of the current region, by switching to the overlapping region to which it is headed. Before a motion command is executed, the current location and direction of motion of the robot should be used to predict the position to which the robot will arrive after the execution of the motion step. If such a position is near the boundary of the current region or outside of it altogether, the robot should switch, before moving, to a region that contains both the current and the predicted new position. Such a region is guaranteed to exist if the overlapping value ρ used for the region decomposition is greater than or equal to the magnitude of the motion steps of the robot. If more than one region containing both positions exist, the one that spans most of the path that the robot will follow should be selected.

7.2 Computation of the Robot's Pose

Each region has associated with it two model views, i.e., two images acquired at two extremes of the region, in which all k features that form the landmark corresponding to the region are simultaneously visible. To compute the robot's position and orientation, the locations of each of the k features in each of the model views and their locations in the current image are used as input to a method that estimates the position and orientation of the current view relative to that of the model views.

When two or more calibrated model views are available, and the position and orientation at which each image was acquired is known, there are several possible approaches to estimate the position and orientation of a calibrated novel view, relative to that of the model views that see the same scene. (By *calibrated view*, we mean that the intrinsic camera parameters at the moment of image acquisition are known.) In the following sections, we discuss different techniques to solve this problem: Basri and Rivlin's linear combination of views approach, methods that use the essential matrix, and a technique that we devised for the case of panoramic images when the robot undergoes planar motion.

7.3 Localization Using Linear Combination of Views

Here we explain Basri and Rivlin's scheme for localization using their Linear Combination of Views approach. The material in this section is taken from [2]. From a list of feature points in an image, two view vectors are constructed containing the x and y coordinates of the points in the image, in order of correspondence. The environment is modeled by a set of such views. The location of a feature point in a novel view can be obtained as a linear combination of its locations in the model views. The position and orientation at which the novel view was acquired, with respect to that of the model views, is recovered from the coefficients of the linear combination. Formally: Let $p_i = (x_i, y_i, z_i)$, $1 \leq i \leq n$ be a set of n object points. The position p'_i of these points in an image I, under weak-perspective projection is given by

$$x'_i = sq_{11}x_i + sq_{12}y_i + sq_{13}z_i + t_x \tag{7.1}$$

$$y'_i = sq_{21}x_i + sq_{22}y_i + sq_{23}z_i + t_y, \tag{7.2}$$

where q_{ij} are the elements of a 3×3 matrix Q, and s is a scale factor.

Let $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{x}', \mathbf{y}' \in \mathbb{R}^n$ be the vectors of x_i, y_i, z_i, z'_i and y'_i coordinates respectively, and $\mathbf{1} = (1, 1, \ldots, 1)$. Then Equations 7.1 and 7.2 can be rewritten in vector form as:

$$\begin{aligned}\mathbf{x}' &= sq_{11}\mathbf{x} + sq_{12}\mathbf{y} + sq_{13}\mathbf{z} + t_x\mathbf{1} \\ \mathbf{y}' &= sq_{21}\mathbf{x} + sq_{22}\mathbf{y} + sq_{23}\mathbf{z} + t_y\mathbf{1}.\end{aligned}$$

Therefore $\mathbf{x}', \mathbf{y}' \in span\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{1}\}$, i.e., $\mathbf{x}$ and $\mathbf{y}$ belong to a subspace of $\mathbb{R}^n$ of dimension four. This subspace is spanned by any four linearly independent vectors in the subspace. Let I^1 and I^2 be two views of the scene. Let $\mathbf{x}_1, \mathbf{y}_1$ and $\mathbf{x}_2, \mathbf{y}_2$ be the location vectors of the n feature points in I^1 and I^2, respectively. These four vectors are linearly independent and belong to the subspace in question; therefore, there exist coefficients a_1, a_2, a_3, a_4 and $b_!, b_2, b_3, b_4$, such that

$$\begin{aligned}\mathbf{x}' &= a_1\mathbf{x}_1 + a_2\mathbf{y}_1 + a_3\mathbf{x}_2 + a_4\mathbf{1} \\ \mathbf{y}' &= b_1\mathbf{x}_1 + b_2\mathbf{y}_1 + b_3\mathbf{x}_2 + b_4\mathbf{1}.\end{aligned}$$

Note that vector $\mathbf{1}$ is a generator of the four-dimensional subspace, so we only need three other linearly independent vectors to have a complete base for it. Therefore we chose to disregard vector $\mathbf{y}_2$. Assume that I^2 is obtained from I^1 by a rotation R, translation $T = (T_x, T_y, T_z)$, and scaling s. The previous equations can then be rewritten as

$$\begin{aligned}\mathbf{x}' &= a_1\mathbf{x}_1 + a_2\mathbf{y}_1 + a_3(sr_{11}\mathbf{x}_1 + sr_{12}\mathbf{y}_1 + sr_{13}\mathbf{z}_1 + T_x) + a_4\mathbf{1} \\ \mathbf{y}' &= b_1\mathbf{x}_1 + b_2\mathbf{y}_1 + b_3(sr_{11}\mathbf{x}_1 + sr_{12}\mathbf{y}_1 + sr_{13}\mathbf{z}_1 + T_x) + b_4\mathbf{1}.\end{aligned}$$

Re-arranging those equations, we arrive at

$$\mathbf{x}' = (a_1 + a_3sr_{11})\mathbf{x}_1 + (a_2 + a_3sr_{12})\mathbf{y}_1 + (a_3sr_{13})\mathbf{z}_1 + (a_3T_x + a_4\mathbf{1}) \tag{7.3}$$

$$\mathbf{y}' = (b_1 + b_3sr_{11})\mathbf{x}_1 + (b_2 + b_3sr_{12})\mathbf{y}_1 + (b_3sr_{13})\mathbf{z}_1 + (b_3T_x + b_4\mathbf{1}). \tag{7.4}$$

Assuming I was obtained from I^1 by a rotation U, translation t_n, and scaling s_n, we have

$$\begin{aligned}\mathbf{x}' &= s_n u_{11}\mathbf{x}_1 + s_n u_{12}\mathbf{y}_1 + s_n u_{13}\mathbf{z}_1 + t_{n_x}\mathbf{1} &(7.5)\\ \mathbf{y}' &= s_n u_{21}\mathbf{x}_1 + s_n u_{22}\mathbf{y}_1 + s_n u_{23}\mathbf{z}_1 + t_{n_y}\mathbf{1}, &(7.6)\end{aligned}$$

and due to the orthonormality of the rows in U, we can derive the following constraint from Equation 7.3:

$$\begin{aligned}s_n^2 &= (s_n u_{11})^2 + (s_n u_{12})^2 + (s_n u_{13})^2\\ &= (a_1 + a_3 s r_{11})^2 + (a_2 + a_3 s r_{12})^2 + (a_3 s r_{13})^2\\ &= a_1^2 + a_2^2 + a_3^2 s^2 + 2a_3 s(a_1 r_{11} + a_2 r_{12}). &(7.7)\end{aligned}$$

Analogously, from Equation 7.4, we can derive

$$s_n^2 = b_1^2 + b_2^2 + b_3^2 s^2 + 2b_3 s(b_1 r_{11} + b_2 r_{12}). \qquad (7.8)$$

And due to orthogonality, we can also derive from 7.3 and 7.4 the constraint

$$\begin{aligned}0 &= (s_n u_{11})(s_n u_{21}) + (s_n u_{12})(s_n u_{22}) + (s_n u_{13})(s_n u_{23})\\ &= (a_1 + a_3 s r_{11})(b_1 + b_3 s r_{11}) + (a_2 + a_3 s r_{12})(b_2 + b_3 s r_{12}) + (a_3 s r_{13})(b_3 s r_{13})\\ &= a_1 b_1 + a_2 b_2 + a_3 b_3 s^2 + (a_1 b_3 + a_3 b_1) s r_{11} + (a_2 b_3 + a_3 b_2) s r_{12}. &(7.9)\end{aligned}$$

If the weak perspective approximation is valid, Equations 7.7 and 7.8 should give similar results, and Equation 7.9 should hold.

Comparing Equation 7.5 with 7.3, and Equation 7.6 with 7.4, we can derive the x and y components of the translation t_n between the position where the model view I^1

was acquired and the position from which image I was taken:

$$\begin{aligned} t_{n_x} &= a_3 T_x + a_4 \mathbf{1} \\ t_{n_y} &= b_3 T_x + b_4 \mathbf{1}. \end{aligned}$$

The z component of this translation can be computed from the z component of the translation T, between model views I^1 and I^2, in a value proportional to the change in scale between the object in image I and I^2:

$$t_{n_z} = \frac{1 - \frac{1}{s_n}}{1 - \frac{1}{s}} T_z.$$

Comparing Equation 7.5 with 7.3, and Equation 7.6 with 7.4, and because U is a rotation matrix, we infer

$$\begin{array}{lll} u_{11} = \frac{a_1 + a_3 s r_{11}}{s_n} & u_{12} = \frac{a_2 + a_3 s r_{12}}{s_n} & u_{13} = \frac{a_3 s r_{13}}{s_n} \\ u_{21} = \frac{b_1 + b_3 s r_{11}}{s_n} & u_{22} = \frac{b_2 + b_3 s r_{12}}{s_n} & u_{23} = \frac{b_3 s r_{13}}{s_n} \end{array}$$
$$(u_{31}, u_{32}, u_{33}) = (u_{11}, u_{12}, u_{13}) \times (u_{21}, u_{22}, u_{23}).$$

U, t_n and s_n are recovered in terms of R, T and s, assuming a calibrated camera model. Erroneous correspondences and/or bad orthographic approximations will yield erroneous alignment coefficients, which will result in incorrect solutions to the positioning problem. t_{n_x} and t_{n_y} depend linearly on the errors in the coefficients, while t_{n_z} is inversely dependent on them. A better estimate of the coefficients can be achieved by using a large number of point correspondences in the computation.

7.4 Localization from the Essential Matrix

In contrast to the linear combination of views method, which approximates perspective projection using a scaled orthographic projection camera model, we can estimate the

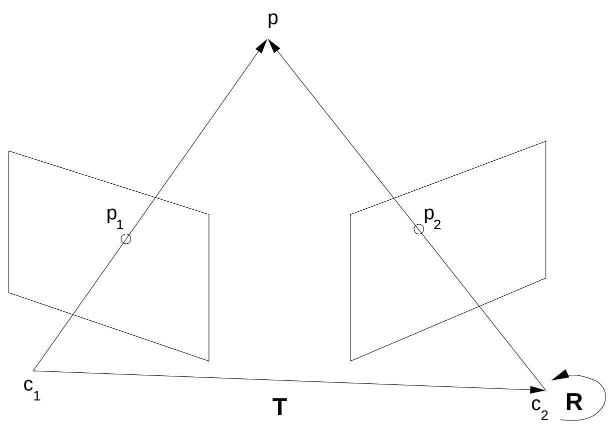


Figure 7.1: Basic geometry of the camera motion problem.

position and orientation of a novel view with respect to that of the model views using an exact perspective projection camera model. Most of the approaches to compute the relative position between cameras from a set of point correspondences exploit the so-called *epipolar constraint*, which relates the locations of the feature points in both images. The basic geometry of the problem under discussion consists of two cameras with their optical centers located at 3D points c_1 and c_2, as shown in Figure 7.1. The second camera has been translated $\mathbf{T}$ from the first camera, and has been rotated $\mathbf{R}$ with respect to the orientation of the first camera. A 3D scene point p is projected onto the image planes of both cameras at locations p_1 and p_2, respectively, measured in the coordinate frame of each camera.

We'll assume that the images are calibrated, i.e., $p_i = P_{i_{\mathrm{int}}}^{-1} p_{i_{\mathrm{im}}}, i = 1, 2$, where $P_{i_{\mathrm{int}}}$ is the projection matrix of intrinsic coordinate parameters of the i-th camera, and $p_{i_{\mathrm{im}}}$ is the location of the projection of p on the non-calibrated image. As seen on the schematic of Figure 7.1, the points p, c_1 and c_2 define a plane, called the *epipolar plane*. The vector $\mathbf{T} \times \mathbf{R}p_1$ is normal to the plane, and hence we can derive the *epipolar constraint* $p_2^t(\mathbf{T} \times \mathbf{R}p_1) = 0$. For all vectors $\overrightarrow{x}$ we can write $\mathbf{T} \times \overrightarrow{x}$ as the product $\mathbf{T}_\times \cdot \overrightarrow{x}$, where

$$\mathbf{T}_\times = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}.$$

Then, the epipolar constraint can be rewritten as $p_2^t(\mathbf{T}_\times \cdot \mathbf{R}p_1) = 0$. The matrix $E = \mathbf{T}_\times \cdot \mathbf{R}$ is called the *essential matrix*. There is a vast literature on how to compute the essential matrix from point correspondences and how to factor it in its corresponding rotation and translation; see, for example, [28]. It must be observed that the translation is computed up to a scale factor, since in the image projection process the depth information is lost.

We can estimate the position and orientation of a novel view I^u with respect to that of the model views via estimating the essential matrices E^{vu} and E^{wu} between the novel view and each of two model views I^v and I^w, respectively, from the correspondence of point-features between the images. Each of the two essential matrices E^{vu} and E^{wu} can then be factored into a rotation matrix (R^{vu} and R^{wu}) and a translation vector ($\overrightarrow{T}^{vu}$ and $\overrightarrow{T}^{wu}$) representing the camera motion between the model images I^v, I^w and the novel view I^u.

Since the position and orientation of the model views is known, the position $\overrightarrow{u}$ of the novel image I^u can be computed as the intersection of the lines that go through $\overrightarrow{v}$ and $\overrightarrow{w}$, with directions given by the translation vectors $\overrightarrow{T}^{vu}$ and $\overrightarrow{T}^{wu}$, respectively, when both vectors are expressed in the same coordinate system. Such intersection $\overrightarrow{u} = (\overrightarrow{v} + \lambda\overrightarrow{T}^{vu}) \cap (\overrightarrow{w} + \mu(R^{vw})^t\overrightarrow{T}^{wu})$, (expressed in the coordinate system of image I^v), where $\lambda, \mu \in \mathbb{R}$ and R^{vw} is the rotation between images I^v and I^w, can be computed from the solution of the linear system of equations

$$\begin{pmatrix} \overrightarrow{T}^{vu} & -(R^{vw})^t\overrightarrow{T}^{wu} \end{pmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \overrightarrow{w} - \overrightarrow{v}.$$

Since this method accurately models perspective projection, it has the advantage over the linear combination of views that there is no possibility of errors due to camera model approximation when the camera distance to objects in the scene is too small. However, this approach is still very sensitive to noise in the values of the coordinates of the matched points. Several methods have been proposed for the computation of the essential matrix. (See [27, 48] for a description and comparison of several algorithms.) Among all the algorithms, the method introduced by Longuet-Higgins [24] is the most remarkably simple one. The *eight-point algorithm*, as it is called, computes the components of the essential matrix by solving a linear system of equations.

For many years, this algorithm was highly criticized for its excessive sensitivity to noise in the coordinates of the matched points. In 1997, Hartley [18] presented a mechanism to normalize the coordinates before running the eight-point algorithm, and showed that the results so achieved are comparable with the ones obtained using the best iterative algorithms. Robust estimation methods such as RANSAC (Random sampling and consensus) [16], or M-estimators [47, 45] can also be employed in the computation of the essential matrix to overcome the effect of outliers due to incorrect point correspondences. Recently, Feng and Hung [15] presented a robust method that finds inliers using random minimum sets, and the estimated matrix is evaluated over the entire data set using the 2D re-projection error as a measure. Their method is shown to perform better than previous robust methods.

A drawback of using the essential matrix for localization, however, is the fact that a larger number of point correspondences are needed to apply it, in contrast with just three required for the linear combination of views approach. (At least eight correspondences are needed to apply the eight point algorithm, although there is an algorithm that only uses five points, but it requires solution of the roots of a 10th degree polynomial and is quite sensitive to noise.) In practice, the algorithms should be applied using more than the minimum number of points required so as to reduce the effects of noise. In the next section, we present a localization method using the essential matrix, in which only three point correspondences are needed when the camera is restricted to motion on a plane.

7.5 Estimating Camera Motion on a Plane

When the camera motion is restricted to the $\langle x, z\rangle$ plane, i.e., translations have a zero y component and rotations are limited to be around the y-axis, we can estimate the motion parameters using as few as three image point correspondences. In the following sections, we present two solutions to this problem, depending on the imaging model used to acquire the images. In the first approach we assume that images were obtained using a pinhole camera and we propose a method to estimate a simplified version of the essential matrix under this kind of motion, including a simple way to extract the translation and rotation parameters from it without any ambiguities. We as well propose a novel approach to estimate the motion parameters when the images were acquired using 360 degree panoramic cameras.

7.5.1 Pinhole Camera Model

The essential matrix E encodes the motion between two calibrated images as $E = [T]_\times \cdot R$, where

$$[T]_\times = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}.$$

In the case of planar motion on the $\langle x, z\rangle$ plane, since $T_y = 0$ and R is a rotation around the y-axis, it follows that

$$E = \begin{pmatrix} 0 & -T_z & 0 \\ T_z & 0 & -T_x \\ 0 & T_x & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{7.10}$$

$$= \begin{pmatrix} 0 & -T_z & 0 \\ T_z\cos\theta + T_x\sin\theta & 0 & T_z\sin\theta - T_x\cos\theta \\ 0 & T_x & 0 \end{pmatrix}. \tag{7.11}$$

Let p_i and p'_i be the location of the i-th image point in the calibrated model images I^1 and I^2, respectively, for $i = 1, \ldots, n$. The epipolar constraint given by the essential matrix is $p'^t_i E p_i = 0$ for $i = 1, \ldots, n$. With four points, we can find an estimate of E by solving a linear system of equations, following the same idea behind the"8-point algorithm", as follows:

$$\begin{aligned} 0 &= p'^t E p \\ &= (p'_x p'_y p'_z) \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \\ &= p'_x(p_x e_{11} + p_y e_{12} + p_z e_{13}) + p'_y(p_x e_{21} + p_y e_{22} + p_z e_{23}) + p'_z(p_x e_{31} + p_y e_{32} + p_z e_{33}). \end{aligned}$$

Hence, each constraint gives rise to a linear equation that has the form $\overrightarrow{c}^t \overrightarrow{e} = 0$, where

$$\begin{aligned} \overrightarrow{c} &= \begin{pmatrix} p'_x p_x & p'_x p_y & p'_x p_z & p'_y p_x & p'_y p_y & p'_y p_z & p'_z p_x & p'_z p_y & p'_z p_z \end{pmatrix}^t \\ \overrightarrow{e} &= \begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33} \end{pmatrix}^t. \end{aligned}$$

In our planar motion model $e_{11} = e_{13} = e_{31} = e_{33} = 0$, therefore we can reduce the constraint vector $\overrightarrow{c}$ and the unknown vector $\overrightarrow{e}$ to

$$\begin{aligned} \overrightarrow{c} &= \begin{pmatrix} p'_x p_y & p'_y p_x & p'_y p_z & p'_z p_y \end{pmatrix}^t \\ \overrightarrow{e} &= \begin{pmatrix} e_{12} & e_{21} & e_{23} & e_{32} \end{pmatrix}^t. \end{aligned}$$

From Equation 7.11, we know that

$$\begin{aligned} e_{12} &= -T_z \\ e_{21} &= T_z\cos\theta + T_x\sin\theta \\ e_{23} &= T_z\sin\theta - T_x\cos\theta \\ e_{32} &= T_x. \end{aligned}$$

From the components of E, we can obtain $T = (e_{32}, 0, -e_{12})$, and compute $\sin\theta$ and $\cos\theta$ as follows:

$$\begin{aligned} \frac{e_{32}e_{21} - e_{12}e_{23}}{e_{32}^2 + e_{12}^2} &= \frac{T_x(T_z\cos\theta + T_x\sin\theta) + T_z(T_z\sin\theta - T_x\cos\theta)}{T_x^2 + T_z^2} \\ &= \frac{T_xT_z\cos\theta + T_x^2\sin\theta + T_z^2\sin\theta - T_zT_x\cos\theta}{T_x^2 + T_z^2} \\ &= \frac{T_x^2\sin\theta + T_z^2\sin\theta}{T_x^2 + T_z^2} = \sin\theta \\ -\frac{e_{32}e_{23} + e_{12}e_{21}}{e_{32}^2 + e_{12}^2} &= -\frac{T_x(T_z\sin\theta - T_x\cos\theta) - T_z(T_z\cos\theta + T_x\sin\theta)}{T_x^2 + T_z^2} \\ &= -\frac{T_xT_z\sin\theta - T_x^2\cos\theta - T_z^2\cos\theta - T_zT_x\sin\theta}{T_x^2 + T_z^2} \\ &= -\frac{-T_x^2\cos\theta - T_z^2\cos\theta}{T_x^2 + T_z^2} = \cos\theta. \end{aligned}$$

Observe that, in reality, our problem only has three unknowns (T_x, T_z and θ.) We therefore could express $\overrightarrow{c}^t\overrightarrow{e} = 0$ as a non-linear system of equations by replacing $\overrightarrow{e}$ with its definition in terms of T_x, T_z and θ, obtaining

$$0 = -p'_x p_y T_z + p'_y p_x(T_z\cos\theta + T_x\sin\theta) + p'_y p_z(T_z\sin\theta - T_x\cos\theta) + p'_z p_y T_x.$$

We could attempt to solve this system with as few as three independent equations. However, it doesn't seem reasonable for the sake of just using one less point correspondence to embark on this path that not only requires much more complex minimization techniques, but also for which there is no guarantee of converging to the optimal solution due to the

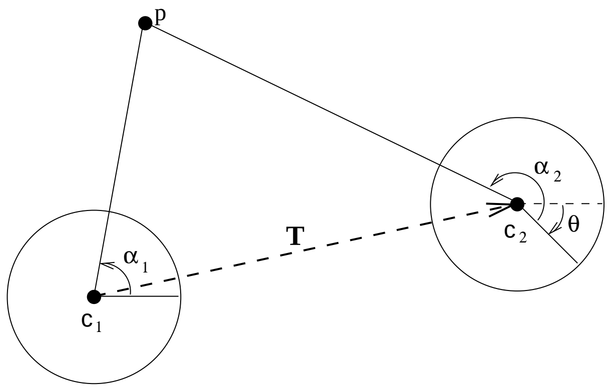


Figure 7.2: Two panoramic cameras centered at points c_1 and c_2. The second camera has been translated T, and has rotated an angle θ with respect to the orientation of the first camera. The point p is seen with a deviation angle of α_1 from the orientation of camera 1, and with a deviation angle α_2 from the orientation of camera 2.

presence of local minima in the function under consideration.

7.5.2 Panoramic Cameras

The epipolar constraint, as explained in section 7.4 can be applied to the case of cylindrical panoramic cameras, by noting that a point with cylindrical coordinates (α, y) on a panoramic camera has 3D coordinates $(\cos\alpha, y, \sin\alpha)$. In this section, we introduce an alternative method to compute the relative motion between a pair panoramic cameras. Let C_1 and C_2 be two cylindrical panoramic cameras located on the plane $\langle x, z\rangle$ centered at points c_1 and c_2, respectively, such that there exists a rotation θ around the y-axis between them. Each camera defines a *camera direction* that corresponds to its azimuth 0. See Figure 7.2.

Let I^1 and I^2 be the images corresponding to cameras centered at c_1 and c_2, respectively. The projection of the 3D point p in each image is defined by a pair of angles $(\alpha_i, \gamma_i), i = 1, 2$, where α_i corresponds to the azimuth of the projection of p on the i-th camera, and γ_i measures the elevation of the projection of p over the horizon of the i-th camera.

Given a point correspondence $(\alpha_1, \gamma_1); (\alpha_2, \gamma_2)$, we can deduce the relationship between the angles and the translation and rotation in the following way: Let p' be the projection of p onto the plane $< X, Z >$. Let $A \in \Re$ be the length of the ray between c_1 and the 3D point p. The distance between c_1 and p' is $D_1 = A \cdot \cos\gamma_1$, and the distance between p and p' is $p_y = A \cdot \sin\gamma_1$. The distance between p' and c_2 is $D_2 = p_y \cdot \cot\gamma_2$. Therefore, the translation between camera centers (in the coordinate system of c_1) is

$$\begin{aligned}
\overrightarrow{T} &= D_1(\cos\alpha_1, \sin\alpha_1)^t - D_2(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t \\
&= A\cos\gamma_1(\cos\alpha_1, \sin\alpha_1)^t - A\sin\gamma_1\cot\gamma_2(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t \\
&= A\sin\gamma_1(\cot\gamma_1(\cos\alpha_1, \sin\alpha_1)^t - \cot\gamma_2(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t) \\
&\propto \cot\gamma_1(\cos\alpha_1, \sin\alpha_1)^t - \cot\gamma_2(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t. \qquad (7.12)
\end{aligned}$$

Computing $\overrightarrow{T}$ and θ

We will present in this section a method for computing the translation $\overrightarrow{T}$ and rotation angle θ between cameras. It will be shown that two point correspondences determine, in a closed form, two possible solutions for the value of θ. The proposed method consists of computing these solutions for each pair of correspondences, and then robustly deciding for the more likely estimate of θ, given the set of computed values. Finally, $\overrightarrow{T}$ can be computed from the values of θ and the point correspondences.

Let $\{((\alpha_1^j, \gamma_1^j), (\alpha_2^j, \gamma_2^j))\}_{j=1,\ldots,n}$ be a set of point correspondences between images I^1 and I^2. Let $\overrightarrow{v}^j = \cot\gamma_1^j(\cos\alpha_1^j, \sin\alpha_1^j)^t$ and $\overrightarrow{w}^j = -\cot\gamma_2^j(\cos\alpha_2^j, \sin\alpha_2^j)^t$. With these definitions, Equation 7.12 can be written as

$$\overrightarrow{T} \propto \overrightarrow{v}^j + R_\theta \overrightarrow{w}^j \in \mathbb{R}^2, \qquad (7.13)$$

where R_θ is a 2×2 rotation matrix of angle θ, with $\overrightarrow{T}$ and θ being the unknowns. Equation 7.13 states that if we knew θ and computed the vectors $\overrightarrow{v}^j + R_\theta \overrightarrow{w}^j \in \mathbb{R}^2$, they would all have the same orientation, that is, the orientation of $\overrightarrow{T}$. That suggests that θ

can be computed as the angle of rotation η that makes all vectors $\overrightarrow{v}^j + R_\eta \overrightarrow{w}^j$ to be the closest in orientation. A set of vectors can be tested for their agreement in orientation by normalizing them and setting them as rows of a matrix A_η, and checking how close A_η is to having rank one. In our particular case, because the vectors are in $\mathbb{R}^2 \setminus \{\overrightarrow{0}\}$, this is equivalent to measuring how close A_η is to being rank deficient, i.e., its null space having dimension one.

One way to measure this is by evaluating how close to zero the smallest singular value of A_η is, which can be computed through a singular value decomposition of the matrix. However, this process is not robust in the presence of outliers in the form of incorrect image point correspondences. An approach that is less affected by outliers is one in which the likelihood of an angle θ to be the correct camera rotation is measured in terms of the number of vectors that get aligned (within a certain threshold) when using such an angle. Therefore, we search for the angle that maximizes the number of vectors agreeing on a direction.

Formally, given a threshold τ, θ could be found as

$$\theta = \arg\max_{\eta} |\psi(\{\overrightarrow{v}^i + R_\eta \overrightarrow{w}^i\}_{i=1,\dots,n})|,$$

where the function $\psi_\tau : 2^{\mathbb{R}^2} \rightarrow 2^{\mathbb{R}^2}$, when applied to a set of vectors, returns the largest subset of vectors in the set that agree on a direction within the threshold τ. Even more sophisticated techniques could be applied to robustly measure the agreement in direction between the vectors. For example, if an adequate distribution is assumed for outliers and inliers, Expectation Maximization could be utilized, and the agreement would be measured by the estimated standard deviation of the distribution of inliers in the mix. Once θ is estimated, the orientation of $\overrightarrow{T}$ can be computed as the principal direction of the set $\psi(\{\overrightarrow{v}^i + R_\theta \overrightarrow{w}^i\}_{i=1,\dots,n})$ via SVD.

There is a problem that makes this approach impractical: a search via steepest ascent is doomed to fail to find the global maximum of the function being optimized, due to its high non-linearity. A possible way around this problem, since we know that $\theta \in [0, 2\pi)$, is to initially exhaustively sweep the whole interval at a certain arbitrary precision, and then perform a local search in the vicinity of the minimum point(s) (within a certain threshold) found during the exhaustive search. The total running time of this whole process can be easily calculated to be $O(k(n^2 + 7n) + t)$, where k is the number of discrete points in which the interval was discretized, t is the running time of the final local search, and $O(n^2)$ is the running time of the proposed method to measure agreement at a particular θ. Yet, we'll present an alternative way to estimate θ that has a total running time of $37n^2 + O(n^2)$, and since k would typically need to be fairly greater than 37 to apply the exhaustive search idea, it doesn't make sense to apply this approach at all.

To derive the proposed method that effectively computes the motion parameters, we model the plane $\langle x, z \rangle$ by means of the complex plane. Hence, vectors are represented by complex numbers, and a rotation matrix is modeled by a complex number with absolute value one. Thus, Equation 7.13 becomes

$$\widetilde{T} \propto \widetilde{v^j} + \widetilde{R_\theta}\widetilde{w^j} \in \mathbb{C}, \tag{7.14}$$

where $\widetilde{T}, \widetilde{v^j}, \widetilde{R_\theta}$ and $\widetilde{w}$ are complex numbers defined as $\widetilde{T} = T_x + iT_y$, $\widetilde{v^j} = v_x^j + iv_y^j$, $\widetilde{R_\theta} = \cos\theta + i\sin\theta$ and $\widetilde{w^j} = w_x^j + iw_y^j$.

Given the image point location corresponding to two point correspondences j and k, from Equation 7.14 we know that $\exists r \in \mathbb{R}$ such that

$$\begin{aligned}
\widetilde{v}^j + \widetilde{R_\theta}\widetilde{w}^j &= r(\widetilde{v}^k + \widetilde{R_\theta}\widetilde{w}^k) \Leftrightarrow \\
\widetilde{v}^j - r\widetilde{v}^k &= \widetilde{R_\theta}(r\widetilde{w}^k - \widetilde{w}^j) \Rightarrow \\
\left|\widetilde{v}^j - r\widetilde{v}^k\right|^2 &= \left|r\widetilde{w}^k - \widetilde{w}^j\right|^2 \Leftrightarrow \\
\left|\widetilde{v}^j\right|^2 - 2(v_x^j v_x^k + v_y^j v_y^k)r + \left|\widetilde{v}^k\right|^2 r^2 &= \left|\widetilde{w}^k\right|^2 r^2 - 2(w_x^k w_x^j + w_y^k w_y^j)r + \left|\widetilde{w}^j\right|^2 \Leftrightarrow
\end{aligned} \tag{7.15}$$

$$0 = (\left|\widetilde{v}^k\right|^2 - \left|\widetilde{w}^k\right|^2)r^2 + 2((w_x^k w_x^j + w_y^k w_y^j) - (v_x^j v_x^k + v_y^j v_y^k))r + (\left|\widetilde{v}^j\right|^2 - |\widetilde{w}^j|^2).$$

Solving this quadratic polynomial yields two solutions for r which, in turn, defines two possible values for $\widetilde{R_\theta}$. From Equation 7.15, we see how to compute $\widetilde{R_\theta}$ from r:

$$\widetilde{R_\theta} = \frac{\widetilde{v}^j - r\widetilde{v}^k}{r\widetilde{w}^k - \widetilde{w}^j} = \frac{(\widetilde{v}^j - r\widetilde{v}^k)\overline{(r\widetilde{w}^k - \widetilde{w}^j)}}{\left|r\widetilde{w}^k - \widetilde{w}^j\right|^2}. \tag{7.16}$$

There are two possible values for $\widetilde{R_\theta}$ from each pair of image point correspondences, but only one of those two values is the correct estimate. To find it, robust estimation has to be applied to the set of all pair of values computed from each pair of point correspondences. We know that at least half of the values are "outliers", for which we can assume a uniform distribution. The remainder are the inliers and should be concentrated around the correct value of the parameter θ, so we can assume that they have a normal distribution. The measured angles belong to a mixture distribution with known proportions, and hence we could try to estimate the mean of the Gaussian applying Expectation Maximization. We can easily compute the running time of this approach as $37n^2 + t + O(n^2)$, where t is the running time of the final robust estimation of θ (typically $O(n^2)$). Once θ is estimated, $\overrightarrow{T}$ can be computed as the principal direction of the set of vectors $\psi(\{\overrightarrow{v}^i + R_\theta \overrightarrow{w}^i\}_{i=1,\ldots,n})$.

Conditions for the Existence of Solutions

We will study here under what conditions a pair of image point correspondences is sufficient to estimate the motion parameters between cameras. According to Equation 7.16, two solutions for $\widetilde{R_\theta}$ can be obtained from two image point correspondences when $\left|r\widetilde{w}^k - \widetilde{w}^j\right|^2 \neq 0$. When this condition does not hold, there are an infinite number of solutions for the motion parameters, since Equation 7.15 will hold for any rotation angle θ. Let's analyze what particular relation between the image point locations will make this condition hold, and therefore produce exactly two solutions. First, observe that

$$\left|r\widetilde{w}^k - \widetilde{w}^j\right|^2 = 0 \iff r\widetilde{w}^k = \widetilde{w}^j. \tag{7.17}$$

And since r is such that $\left|r\widetilde{w}^k - \widetilde{w}^j\right|^2 = \left|\widetilde{v}^j - r\widetilde{v}^k\right|^2$, then

$$0 = \left|r\widetilde{w}^k - \widetilde{w}^j\right|^2 = \left|\widetilde{v}^j - r\widetilde{v}^k\right|^2 \iff r\widetilde{v}^k = \widetilde{v}^j. \tag{7.18}$$

From Equations 7.17 and 7.18, it must be the case that

$$\frac{\widetilde{w}^j}{\widetilde{w}^k} = \frac{\widetilde{v}^j}{\widetilde{v}^k}.$$

And using the definitions of $\widetilde{v}^j, \widetilde{v}^k, \widetilde{w}^j$ and $\widetilde{w}^k$, and the fact that $r \in \mathbb{R}$, this in turn means that

$$\frac{\cot \gamma_2^j}{\cot \gamma_2^k} = \frac{\cot \gamma_1^j}{\cot \gamma_1^k}. \tag{7.19}$$

Pairs of image point correspondences that satisfy Equation 7.19 should be avoided. A particular example of a configuration of two 3D points that satisfies this equation is one in which both points have the same x and z value, but different $y > 0$. It is easy to mentally picture this configuration and intuitively see that any translation and rotation between two cameras that see these points is possible. Let $p^j = (x, y, z^j), j = 1, 2$ be the two 3D points seen from both cameras, such that $z^j > 0$. Let d_i^j be the distance from the i-th camera center to the projection of p^j onto the $\langle x, z\rangle$ plane for $i = 1, 2$. (Note that $d_i^1 = d_i^2$ because the x and z components of p^1 and p^2 are the same.) Then $\cot\gamma_i^j = \frac{d_i^j}{z_j}$. Equation 7.19 for this point and camera configuration therefore becomes

$$\frac{\cot\gamma_2^1}{\cot\gamma_2^2} = \frac{\cot\gamma_1^1}{\cot\gamma_1^2} \quad\Longleftrightarrow\quad \frac{\left(\frac{d_2^1}{z_1}\right)}{\left(\frac{d_2^2}{z_2}\right)} = \frac{\left(\frac{d_1^1}{z_1}\right)}{\left(\frac{d_1^2}{z_2}\right)}$$

which holds, since $d_1^1 = d_1^2$ and $d_2^1 = d_2^2$. In practice, this is not a problem, since it is very unlikely that many image point correspondences would satisfy Equation 7.19.

Analysis of Numerical Stability

Note that points having an image location with a null elevation angle cannot be used because in that case $\cot\gamma$ would be ∞, and therefore Equation 7.12 wouldn't be well defined. In general it should be avoided to use points with γ close to 0, since as we will show, the errors in the computed equations (7.12) relating θ and $\overrightarrow{T}$ rapidly grow as γ tends to 0.

The absolute difference between the cotangent of an angle γ and itself perturbed by a small value ϵ is

$$\begin{aligned}
|\cot(\gamma) - \cot(\gamma+\epsilon)| &= \left|\frac{\cos\gamma}{\sin\gamma} - \frac{\cos(\gamma+\epsilon)}{\sin(\gamma+\epsilon)}\right| \\
&= \left|\frac{\cos\gamma}{\sin\gamma} - \frac{\cos\gamma\cos\epsilon - \sin\gamma\sin\epsilon}{\sin\gamma\cos\epsilon + \sin\epsilon\cos\gamma}\right| \\
&= \left|\frac{\sin\gamma\cos\gamma\cos\epsilon + \sin\epsilon\cos^2\gamma - \sin\gamma\cos\gamma\cos\epsilon + \sin^2\gamma\sin\epsilon}{\sin\gamma(\sin\gamma\cos\epsilon + \sin\epsilon\cos\gamma)}\right| \\
&= \left|\frac{\sin\epsilon(\sin^2\gamma + \cos^2\gamma)}{\sin^2\gamma\cos\epsilon + \sin\epsilon\sin\gamma\cos\gamma)}\right| \\
&= \frac{1}{\left|\frac{\sin^2\gamma\cos\epsilon + \sin\epsilon\sin\gamma\cos\gamma}{\sin\epsilon}\right|} \\
&= \frac{1}{\left|\sin^2\gamma\cot\epsilon + \sin\gamma\cos\gamma\right|}
\end{aligned}$$

If the amount ϵ of error in the measured angle is bounded, then $\cot\epsilon$ is bounded, and hence

$$\lim_{\gamma\to 0}|\cot(\gamma) - \cot(\gamma+\epsilon)| = \lim_{\gamma\to 0}\frac{1}{\left|\sin^2\gamma\cot\epsilon + \sin\gamma\cos\gamma\right|} = +\infty. \tag{7.20}$$

By definition, $\overrightarrow{v}^j = \cot\gamma_1^j(\cos\alpha_1^j, \sin\alpha_1^j)^t$, therefore

$$\|\overrightarrow{v}^j\| = \|\cot\gamma_1^j(\cos\alpha_1^j, \sin\alpha_1^j)^t\| = \left|\cot\gamma_1^j\right|. \tag{7.21}$$

In a similar manner, $\overrightarrow{w}^j = -\cot\gamma_2^j(\cos\alpha_2^j, \sin\alpha_2^j)^t$, and hence

$$\|\overrightarrow{w}^j\| = \left|\cot\gamma_2^j\right|. \tag{7.22}$$

Equation 7.20 tells us that for small measured values of γ, even a small error can produce an unbounded difference between the cotangent of the measured angle and the cotangent of the real angle γ. According to equations 7.21 and 7.22, these unbounded differences will translate into unbounded changes in the values of $\|\overrightarrow{v}^j\|$ or $\|\overrightarrow{w}^j\|$.

Let's consider the case in which the measured value of γ_1 is close to 0, while the measured γ_2 is not. A large increase in $\|\overrightarrow{v}^j\|$ due to errors in the measurement of γ_1 will make the orientation of vector $\overrightarrow{v}^j + R_\theta \overrightarrow{w}^j$ to be highly influenced by the direction of $\overrightarrow{v}^j$ itself, while a significant decrease of $\|\overrightarrow{v}^j\|$ will make the orientation of $\overrightarrow{v}^j + R_\theta \overrightarrow{w}^j$ close to that of $R_\theta \overrightarrow{w}^j$. In either case, the orientation of $\overrightarrow{v}^j + R_\theta \overrightarrow{w}^j$ will be quite different from the correct one. Therefore, the angle θ that aligns $\overrightarrow{v}^j + R_\theta \overrightarrow{w}^j$ to $\overrightarrow{T}$ will differ greatly from the correct one.

Advantages of Using Panoramic Cameras with Planar Motion

Besides the obvious fact that a larger number of feature correspondences between images can be computed when using panoramic cameras (since the field of view in a panoramic camera is 360 degrees, several times the field of view of a perspective camera), there is an even greater advantage to using this type of camera when the planar motion model can be applied. In this case, if the focal length and y-scaling factor of both cameras is the same, then the only image calibration parameter that needs to be known is the y-coordinate y_0 corresponding to elevation 0 in the image, i.e., the level of the horizon.

To compute the azimuth of an image point, a particular image x-coordinate has to be chosen as azimuth 0, i.e., the orientation at which the camera is "heading", e.g., $x = 0$. Since the width w of the panoramic image corresponds to a field of view of 360 degrees, the azimuth of an image point (p_x, p_y) is $\alpha = \frac{360 p_x}{w}$ degrees. When the focal length f and the y-scaling factor s_y is the same in both cameras, (which is the case when both images where acquired by the same robot camera at different locations), we can compute the motion between cameras without explicitly knowing those calibration parameters. Let γ_i be the elevation angle of an image point $p_i = (p_{i_x}, p_{i_y})$. Then $\cot \gamma_i = \frac{f}{s_y(p_{i_y} - y_0)}$, and so f and s_y can be extracted as part of a common factor in Equation 7.12:

$$\begin{aligned}
\overrightarrow{T} &\propto \cot\gamma_1(\cos\alpha_1, \sin\alpha_1)^t - \cot\gamma_2(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t \\
&\propto \frac{f}{s_y(p_{1_y}-y_0)}(\cos\alpha_1, \sin\alpha_1)^t - \frac{f}{s_y(p_{2_y}-y_0)}(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t \\
&\propto \frac{f}{s_y}\left(\frac{1}{p_{1_y}-y_0}(\cos\alpha_1, \sin\alpha_1)^t - \frac{1}{p_{2_y}-y_0}(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t\right) \\
&\propto \frac{1}{p_{1_y}-y_0}(\cos\alpha_1, \sin\alpha_1)^t - \frac{1}{p_{2_y}-y_0}(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t \\
&\propto (p_{2_y}-y_0)(\cos\alpha_1, \sin\alpha_1)^t - (p_{1_y}-y_0)(\cos(\alpha_2-\theta), \sin(\alpha_2-\theta))^t.
\end{aligned}$$

As a result, y_0 is the only calibration parameter that needs to be known in order to estimate θ and $\overrightarrow{T}$.

Chapter 8

Results

In this thesis, we have proposed methods to accomplish the three main tasks involved in a view-based navigation system, i.e., off-line exploration, landmark database construction, and on-line localization. We are only going to demonstrate the results obtained for what has been the main contribution of this work, namely, our proposed method for optimal landmark selection by region decomposition. We start by briefly describing the simulator that we used to generate synthetic worlds and to run decomposition algorithms on visibility data. We finish by commenting on the results obtained when we applied the heuristic algorithms presented in previous chapter to decompose both synthetic and real worlds.

8.1 Simulator Description

Synthetic visibility data was produced using a simulator we developed. A world consists of a 2-D top view of the pose space defined by a polygon, with internal polygonal obstacles, and a collection of features on the polygons (both external and internal). Each feature is defined by two parameters, an angle (*visibility angle extent*), and a range of visibility (*visibility range*), determining the span of the area on the floor from which the feature is visible. An example of a randomly generated world and the visibility area of some of its

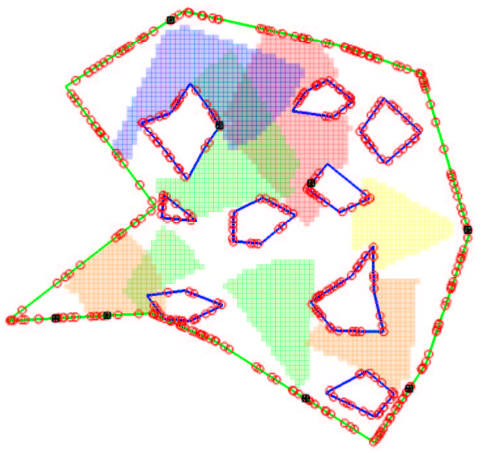
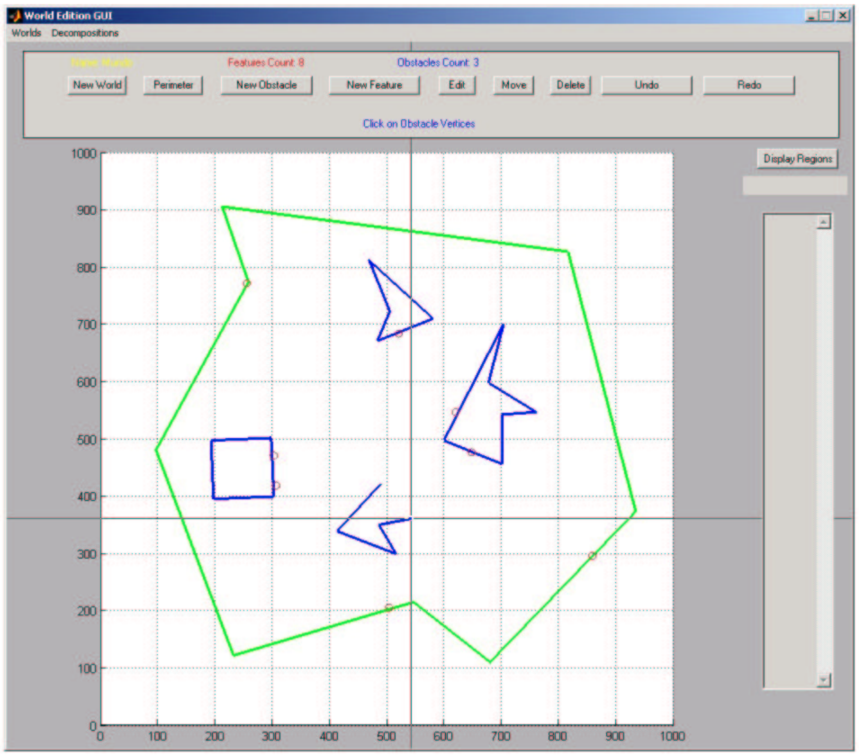


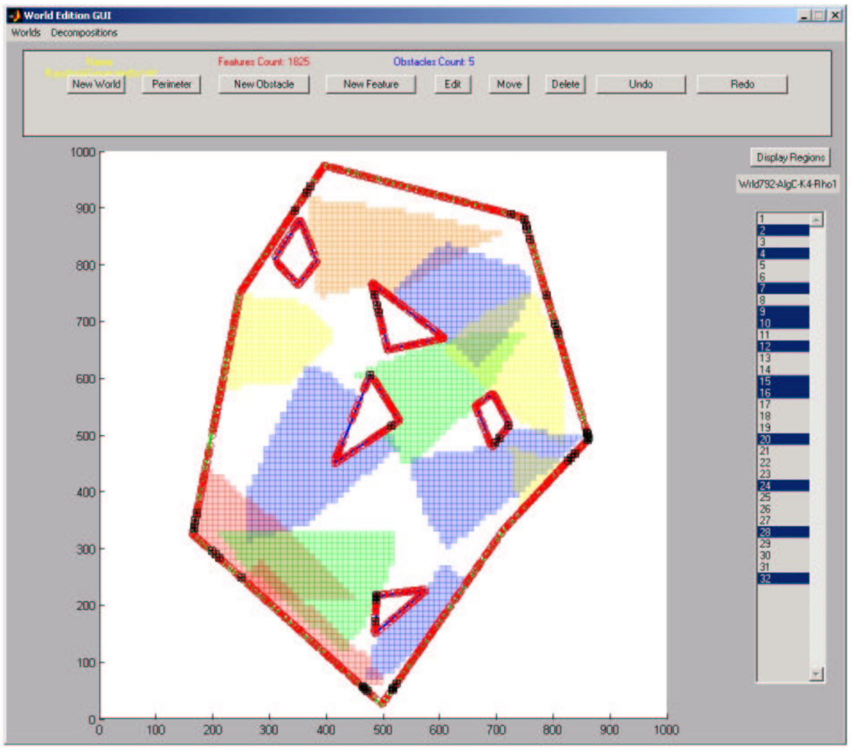
Figure 8.1: A randomly generated world. The green polygon defines the perimeter of the world. The blue polygons in the interior define the boundaries of obstacles. The small red circles on the polygons are the features. As an illustration, the visibility areas of selected features are shown as coloured regions.

features is illustrated in Figure 8.1.

The worlds can either be designed manually through a graphical user interface shown, in Figure 8.2, or generated randomly. In the latter case, a mixture probability distribution has to be specified for each of the defining parameters of the world, which are shown in Table 8.1. Given a sampling grid, the simulator creates the visibility data of a synthetic world by considering that a feature is visible from a grid vertex if the line connecting the vertex to the feature does not intersect an obstacle or the perimeter, and if that line is entirely contained in the visibility region of the feature. The simulator provides, as well, an object-oriented framework, in which a decomposition algorithm can be built as a new class, implementing a class method that returns the decomposition that the algorithm achieves for the visibility data provided as its input. Such visibility data can either be the one computed in a synthetic world, or come from real world imagery. Finally the simulator includes a visualization tool that allows the user to display the decomposition produced by an algorithm on a particular world by selecting a set of regions to display.



(a)



(b)

Figure 8.2: Simulator graphic user interface. (a) World edition. (b) Visualization of a region decomposition.

8.2 Decomposition of Synthetic Worlds

Independent tests of the algorithms on synthetic data were performed for four different world settings. The settings combined different feature visibility properties with different shape complexities for the world and obstacle boundaries. Two types of features were used, having visibility ranges: $\mathcal{N}(0.65, 0.2)$ to $\mathcal{N}(12.5, 1)$m with an angular range $\mathcal{N}(25, 3)$ degrees for Type 1, and $\mathcal{N}(0.65, 0.2)$ to $\mathcal{N}(17.5, 2)$m with an angular range $\mathcal{N}(45, 4)$ degrees for Type 2 (where $\mathcal{N}(\mu, \sigma)$ is normally distributed with mean μ and variance σ^2). Two classes of shapes were tested for the world and obstacles: irregular and rectangular. For the case of irregular worlds, the number of sides of its perimeter was generated from the mixture distribution $\{\mathcal{U}(4, 4)$ with $p = 0.1$; $\mathcal{N}(5, 0.5)$ with $p = 0.45$; $\mathcal{N}(7, 2)$ with $p = 0.45]\}$, and the number of obstacles from the distribution $\{\mathcal{U}(5, 9)$ with $p = 0.5$; $\mathcal{N}(8, 2)$ with $p = 0.5\}$. The number of obstacles in each rectangular world was generated from the mixture distribution $\{\mathcal{U}(6, 9)$ with $p = 0.5$; $\mathcal{N}(10, 2)$ with $p = 0.5\}$. The generated worlds had an average diameter of 40m, and feature visibility was sampled in pose space at points spaced at 50cm intervals.

Table 8.1: Average number of features visible from a pose

Component	**Parameters**
Perimeter	• Sides count • Vertex radius
Obstacles	• Total obstacles count • Sides count • Vertex radius
Features	• Total features count • Visibility angular extent • Visibility range

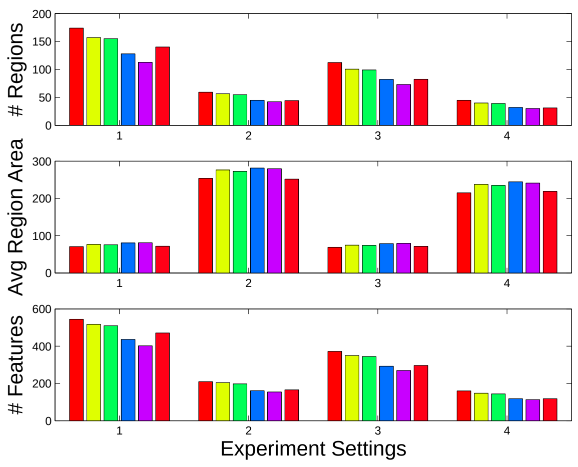


Figure 8.3: Results for Experiments on Synthetic Data. The x-axes of the charts represent the four world settings used in the experiments. (Rectangular worlds were used in settings 1 and 2, while irregularly shaped worlds in settings 3 and 4. Type 1 features were used in settings 1 and 3, and Type 2 features in settings 2 and 4.) The y-axes correspond to the average value of 300 experiments for the total number of regions, average number of poses per region, and total number of used features in each decomposition. From left to right, the bars at each setting correspond to Algorithms A.1, A.2, A.3, B.1, B.2, and C.

The parameters used in the experiments were overlapping $\rho = 1$, and features commonly visible per region $k = 4$. (Basri and Rivlin [2] showed that reliable localization can be accomplished using their linear combination of model views method with as few as three point correspondences between the current image and two stored model views.) The allowed maximum area of a hole was set to $\sigma = 9$ poses, i.e., on average, a hole has a diameter of at most 1.5m. The parameter α of algorithm C was set to 0.85.

Figure 8.3 shows the results of the experiments on synthetic data. The performance of each algorithm in the four settings described above is compared in terms of the number of regions in the decomposition, the average area of a region in a decomposition, and the size of the set formed by the union of the k features commonly visible from each region in a decomposition. Each value in the figure is the average computed over a set of 300 randomly generated worlds. The decomposition of each world took only a few seconds for each algorithm.

Unsurprisingly, the average size of a region is strongly dependent on the stability of its defining features in pose space. Also as expected, the total number of regions in each decomposition increases as the average size of the regions decreases. Tables 8.2 and 8.3 show the number of regions and the average number of poses in a region, respectively, achieved by each algorithm and setting, averaged over all the randomly generated worlds. In the case of worlds with widely visible features (settings 2 and 4), the best results, in terms of minimum number of regions in the decomposition, are achieved by Algorithm B.2, closely followed by algorithms B.1 and C. For the worlds with less visible features (settings 1 and 3), Algorithm B.2 outperformed the rest.

Table 8.2: Average number of regions in a decomposition

Setting	**A.1**	**A.2**	**A.3**	**B.1**	**B.2**	**C**
1	173.81	156.96	154.97	127.76	112.63	140.10
2	59.30	56.45	54.72	44.74	42.10	44.17
3	112.40	100.46	98.97	82.11	73.08	82.29
4	44.71	40.00	39.11	31.99	30.02	31.11

Table 8.3: Average number of poses per region

Setting	**A.1**	**A.2**	**A.3**	**B.1**	**B.2**	**C**
1	70.76	76.49	75.74	80.60	80.99	71.85
2	253.88	276.37	272.83	281.63	279.81	251.86
3	69.04	74.60	73.95	78.63	79.29	71.61
4	215.15	237.68	234.67	244.44	241.26	218.56

In our simulations, we obtained fairly big regions, as seen in Table 8.3. Each pose corresponds to a sampled area of 0.25m^2 (50cm by 50cm), so the averages achieved by the best algorithm correspond to region areas of 20m^2 for features of Type 1, and 65m^2 for features of Type 2. These results were achieved with only a few features visible per pose, as shown in Table 8.4, where the average number of features visible per pose was on the order of a hundred. In real image data, however, the number of stable features visible per pose is on the order of several hundred, and each feature has a visibility range close to that of our simulated features of Type 1 (see [25], for example). These findings lead us to predict that this technique will successfully find decompositions useful for robot navigation in real environments.

Table 8.4: Average number of features visible from a pose

Setting	**Average Number of Features**
1	30
2	95
3	41
4	117

8.3 Region Decomposition Using Real Data

We took Algorithm B.2, the algorithm that achieved the best results on synthetic data, and as a further evaluation we applied it to two real datasets of real imagery.

8.3.1 Decomposition of a 2m by 2m World

We first applied Algorithm B.2 to visibility data collected in a 2m by 2m space sampled at 20 cm intervals, with a total of 46 visible features. All images were taken with the camera in a fixed orientation (looking forward), and features were extracted using the Kanade-Lucas-Tomasi (KLT) operator [4] and tracked between images. The parameters used in the decomposition were $k = 4$, $\rho = 0$, $\sigma = 3$. The four regions of the decompositions can be seen in figure 8.4. The larger gray area present in each one of the images of the regions corresponds to the set of k-coverable poses. As can be seen from the figure, the union of the four regions covers almost completely the k-coverable area of the world.

8.3.2 Decomposition of a 6m by 3m World

Our next experiment involved visibility data acquired in a 6m by 3m grid sampled at 25 cm, (i.e., a lattice of 25 by 12 poses), from Room 408, McConnell Engineering Building,

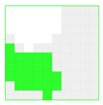
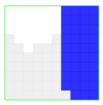
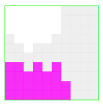
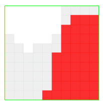

(a)

(b)

(c)

(d)

Figure 8.4: (a)-(d) The 4 regions of the decomposition of real visibility data collected in a 2m by 2m space, sampled at 20 cm intervals.

in McGill University. Images were taken with the robot's camera orientation fixed in four different orientations at 0, 90, 180 and 270 degrees. Each image's position was measured using a laser tracker and a target mounted on the robot [32]. Figure 8.5 shows some images of the employed dataset where the variations in image scale can be appreciated. The images shown for 0 and 180 degree orientations correspond to poses that are furthest back along the orientation. The images at 90 and 270 degree orientations correspond to poses that are furthest front, center, and furthest back along the orientations.

We extracted SIFT features from the images in the dataset using David Lowe's implementation [26]. On average, about 420 SIFT feature vectors were extracted from each image. We then used the method that we proposed in section 6.2.1 to match the feature vectors from different images, and to discard those that were ambiguous. We ended up with a total of 897 classes of image features that are visible from at least 16 different poses. An example of the typical feature visibility regions that we obtained after we ran Algorithm 6.3 can be seen in Figure 8.6. Each of these images represents the visibility region of a particular feature in the 25 by 12 pose sampling grid. Each thumbnail corresponds to the appearance of a (30 by 30 pixels) context around the feature point extracted from the image taken at the corresponding grid position in pose space.

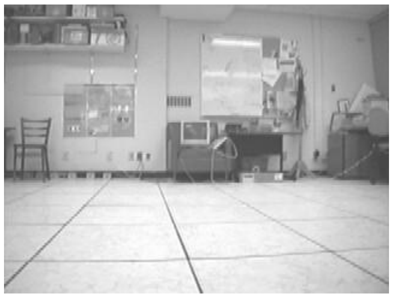
Following our suggestions in section 6.2.2, from the set of distinctive features that remained after the grouping into classes, we only retained those that were widely and consistently visible, that is, those that were visible from at least 16 poses, whose visibility regions had few small holes, and that contained at least one connected component of at

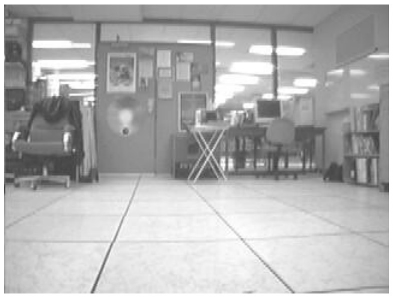
(a)

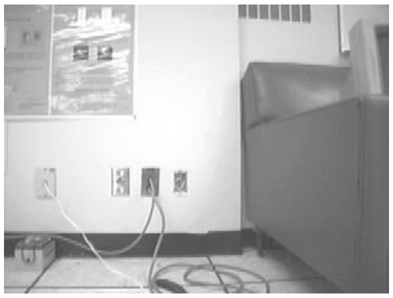
(b)

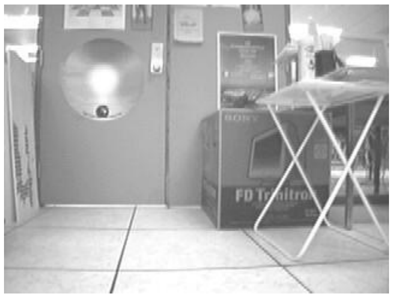
(c)

(d)


(e)


(f)


(g)



(h)

Figure 8.5: Examples of the images used in the experiments on visibility data collected at a 6m × 3m space. The images shown were acquired in orientations: (a) 0 degrees; (b) 180 degrees; (c), (e), (g) 90 degrees; and (d), (f), (h) 270 degrees.

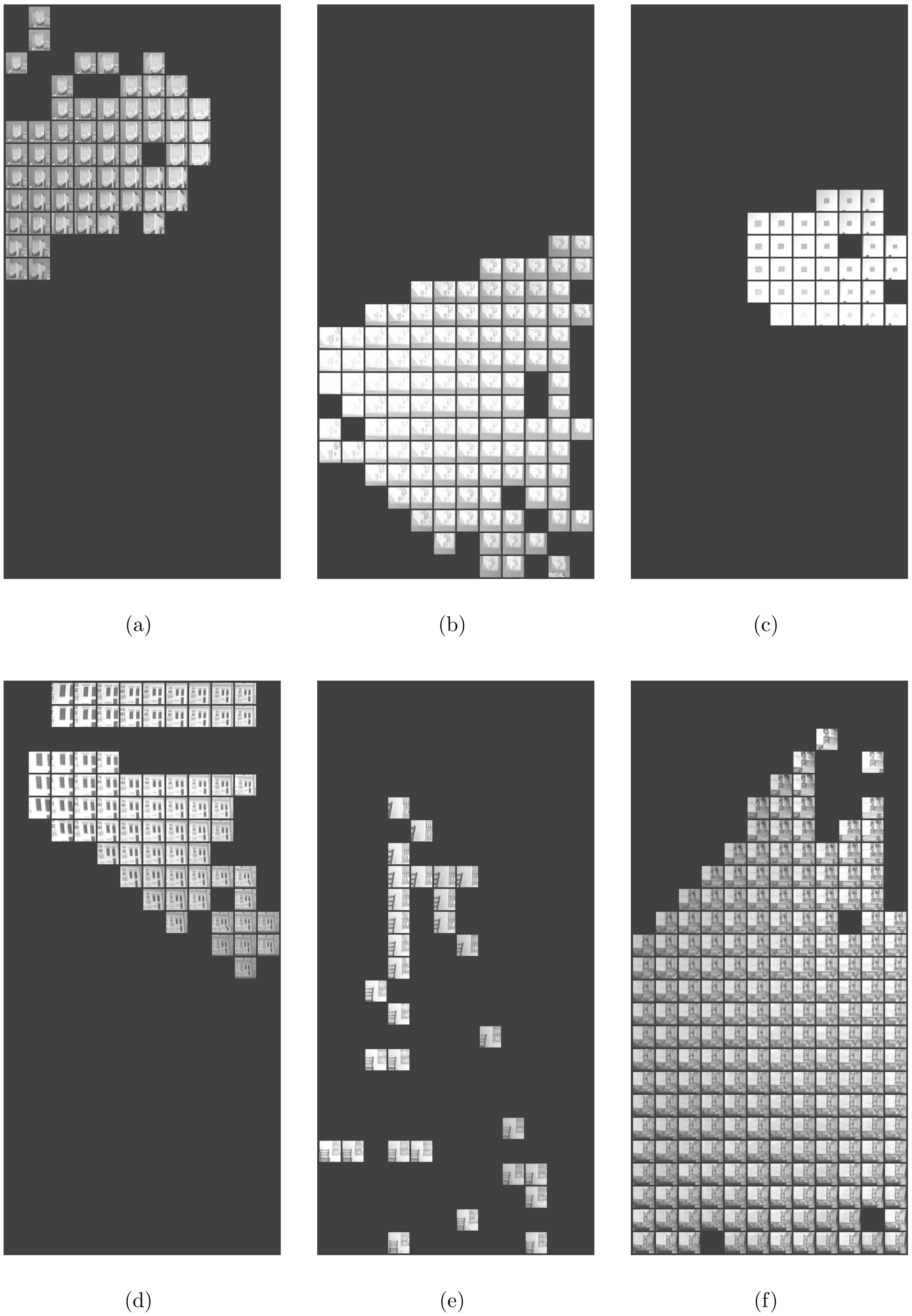
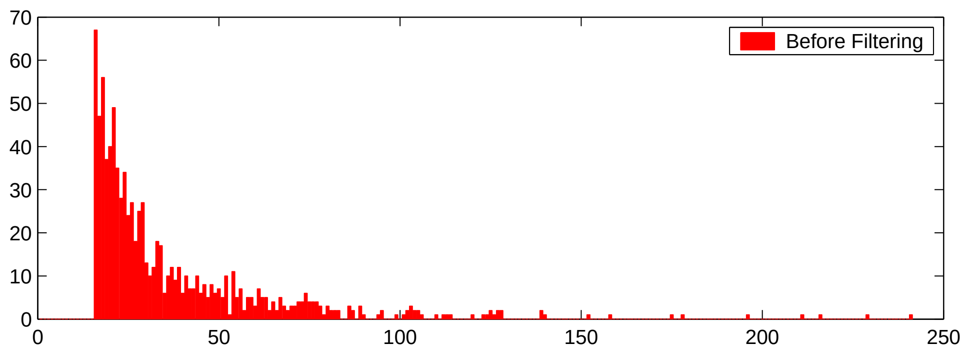


Figure 8.6: Typical examples of feature visibility regions obtained after executing Algorithm 6.3.



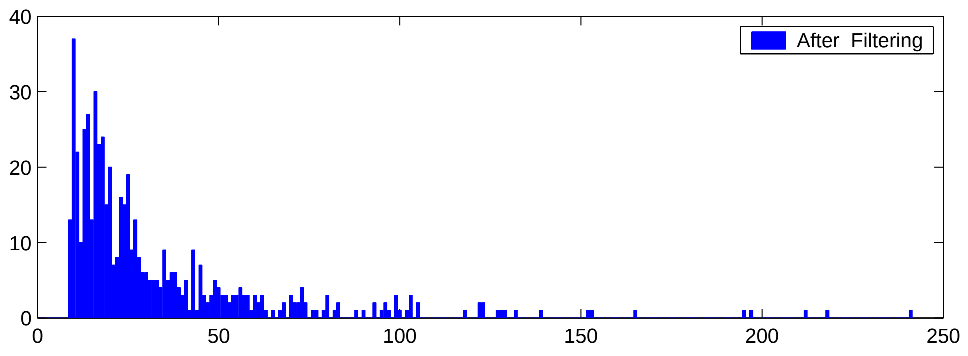
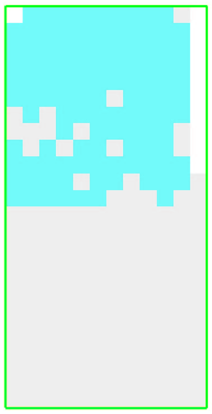


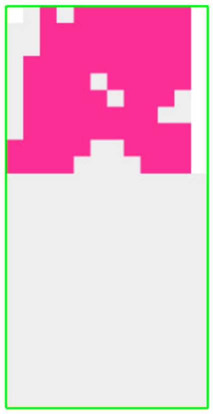
Figure 8.7: Distribution of Feature Visibility Regions by Size (i.e., number of poses).

least 3 by 3 poses. The set of poses of each of these feature visibility regions was further reduced to a subset that had a fairly convex shape. This was achieved by first retaining only the poses in the largest connected component of the visibility region. Secondly, poses were then removed from this component which did not have a neighbor with at least 7 out of 8 of its neighbor poses in the region. After these steps, the feature visibility regions of each class not only reduced in size, but also the total number of image feature classes decreased to 554, since many of the visibility regions became empty as a result of the mentioned operations. Figure 8.7 shows the distribution of feature visibility regions by size before and after this filtering process. The visibility regions, after filtering, had an average size of 33 poses, and a median of 23.

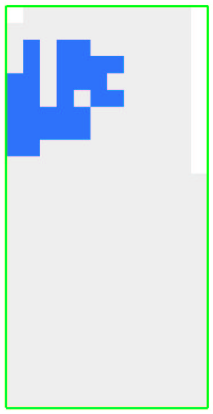
In Figure 8.8 we can see the 7 regions obtained in the decomposition when we used these visibility regions as input to Algorithm B.2, using parameteres $k = 4$, $\rho = 0$, and $\sigma = 3$. The decomposition obtained using these same parameters but with $\rho = 1$ has 9



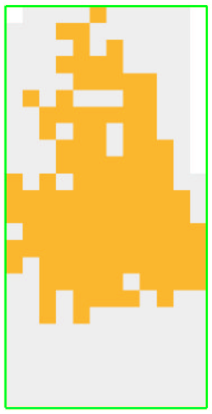
(a)



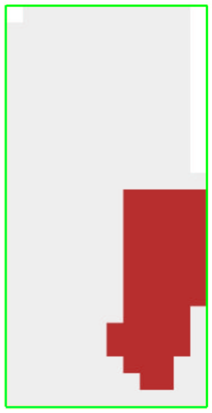
(b)



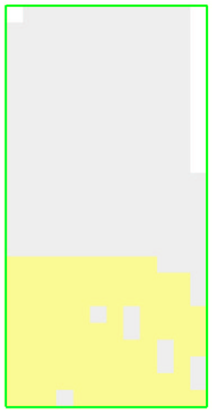
(c)



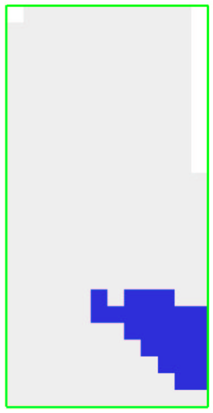
(d)



(e)



(f)



(g)

Figure 8.8: Region decomposition of the 6m by 3m Real world for $k = 4$ and $\rho = 0$ using Algorithm B.x.

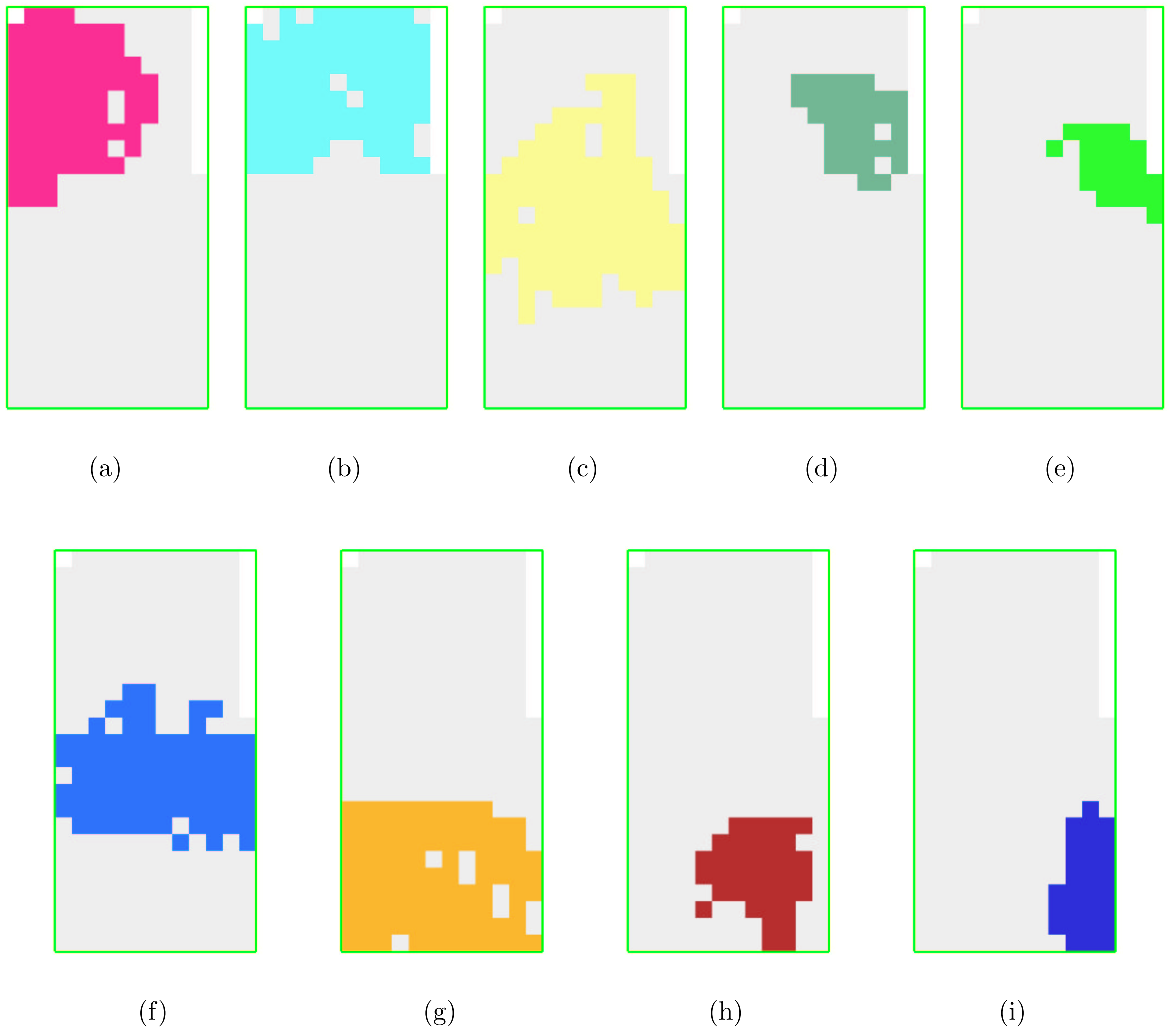
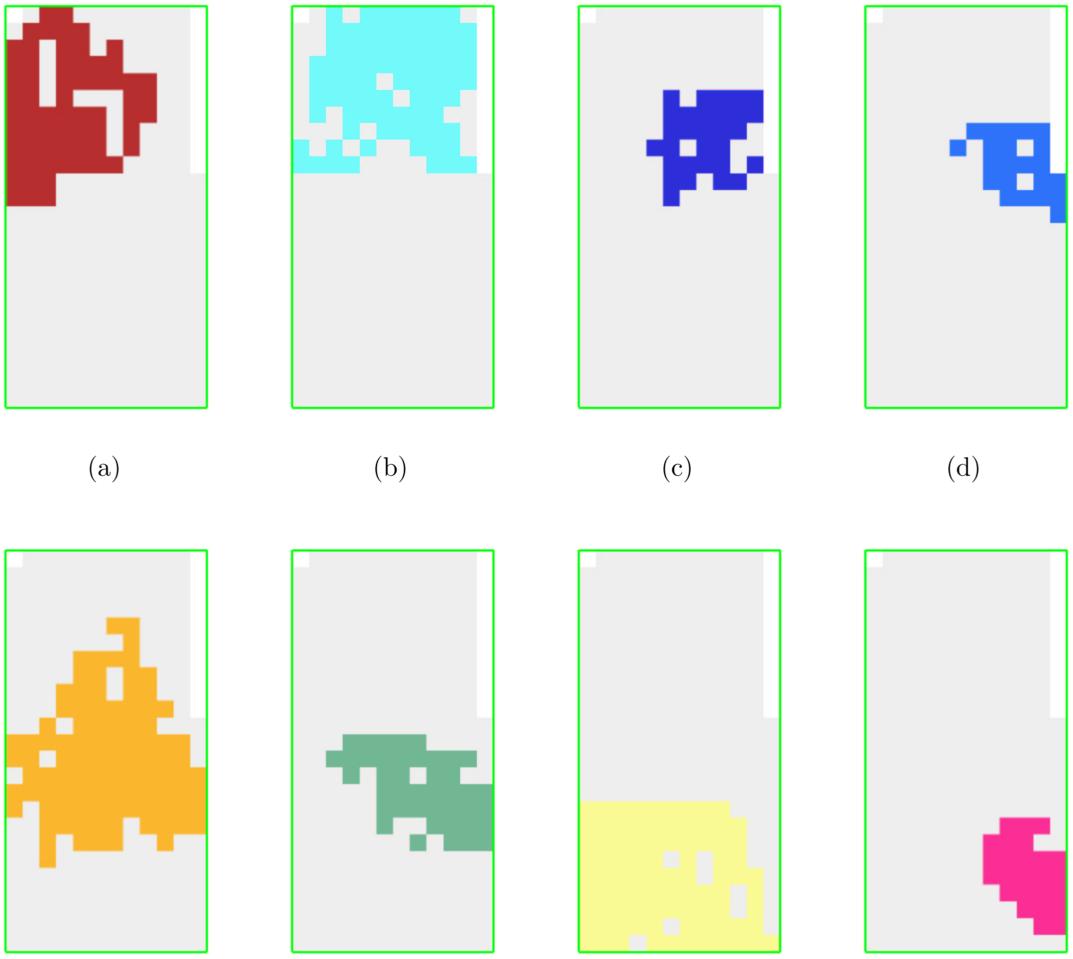


Figure 8.9: Region decomposition of the 6m by 3m Real world for $k = 4$ and $\rho = 1$ using Algorithm B.x.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8.10: Region decomposition of the 6m by 3m Real world for $k = 6$ and $\rho = 0$ using Algorithm B.x.

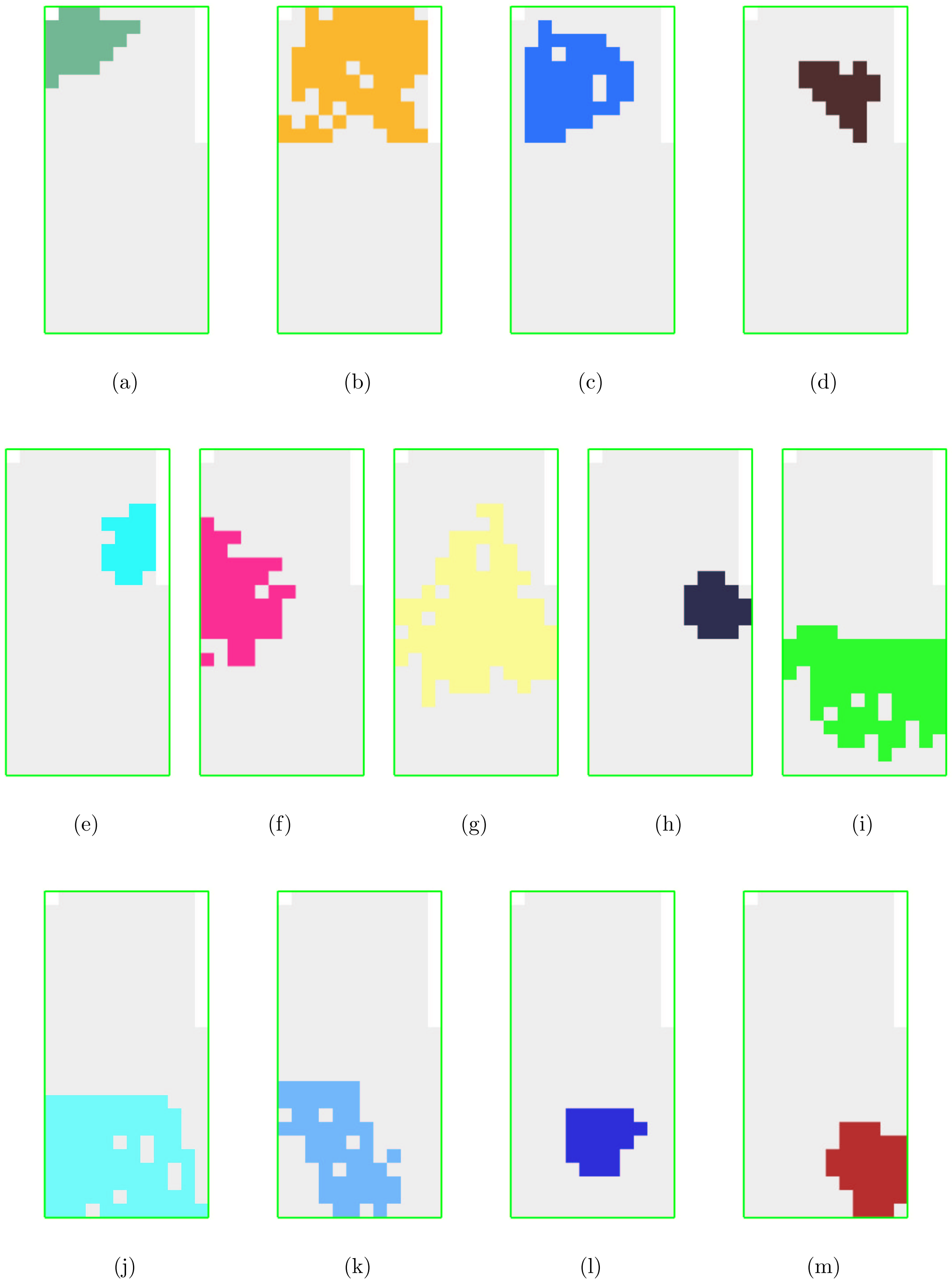


Figure 8.11: Region decomposition of the 6m by 3m Real world for $k = 6$ and $\rho = 1$ using Algorithm B.x.

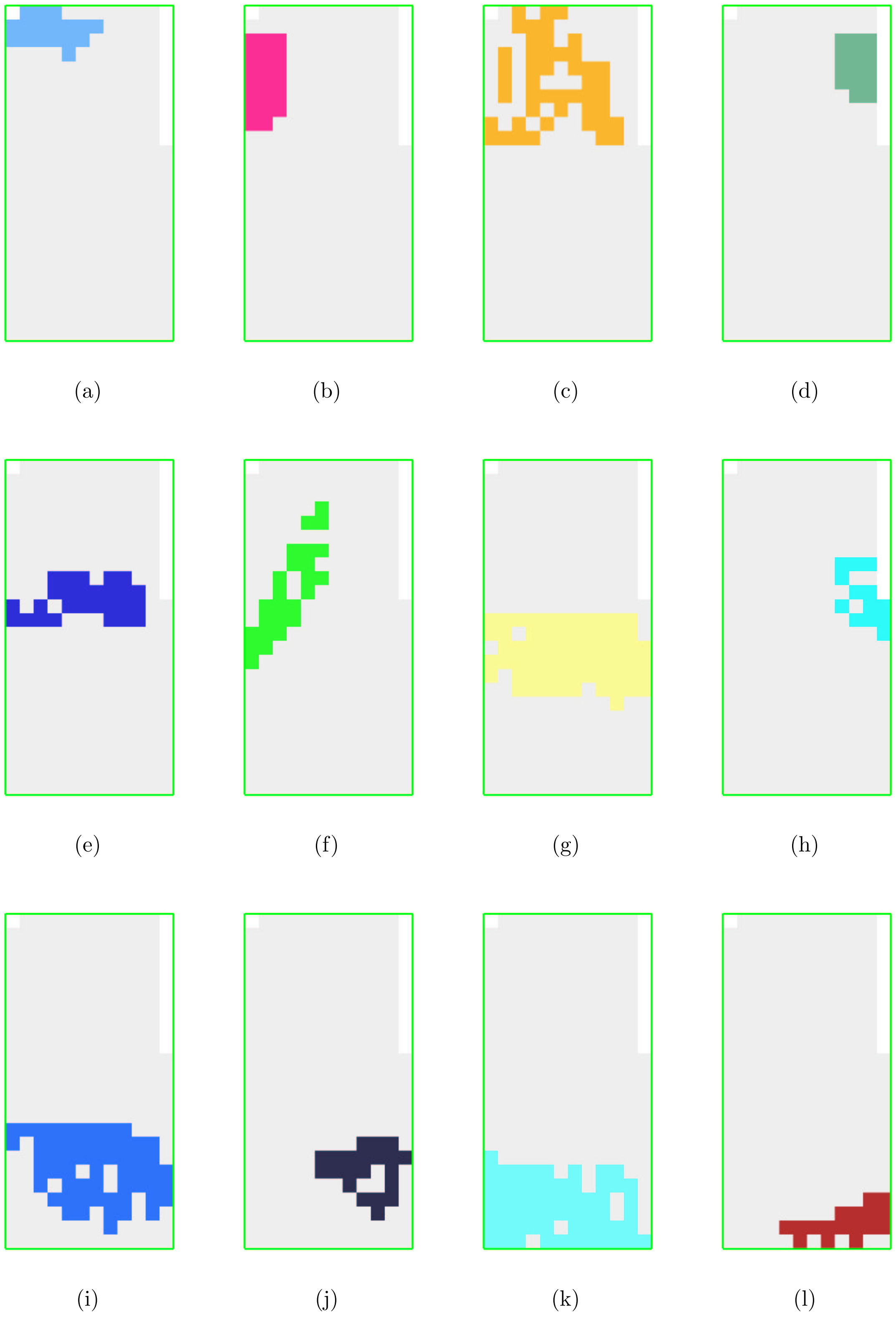


Figure 8.12: Region decomposition of the 6m by 3m Real world for $k = 8$ and $\rho = 0$ using Algorithm B.x.

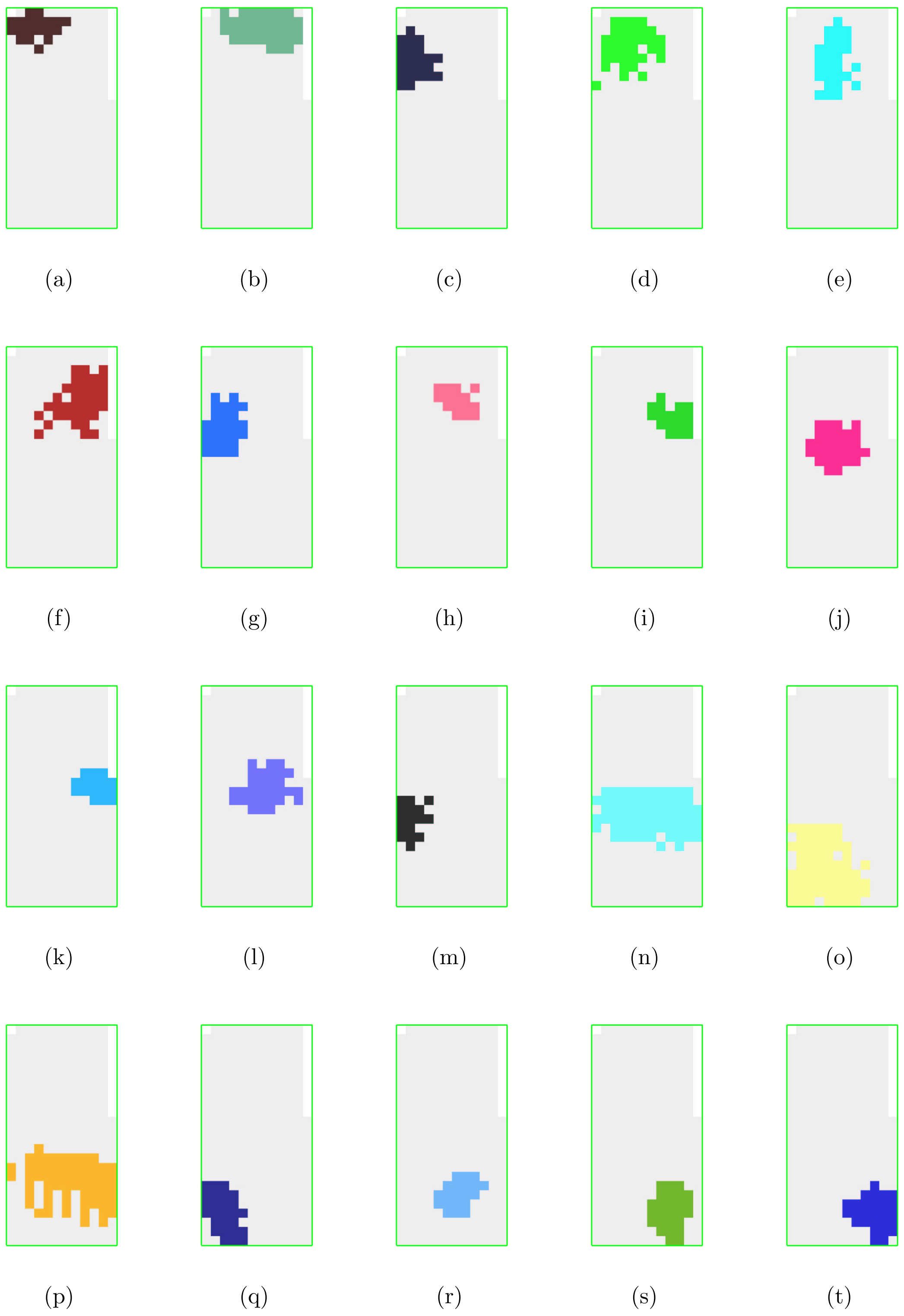


Figure 8.13: Region decomposition of the 6m by 3m Real world for $k = 8$ and $\rho = 1$ using Algorithm B.x.

regions, as shown in Figure 8.9. The decompositions obtained when using values of 6, 8 and 10 for k, and 1 and 2 for ρ can be seen in Figures 8.10, 8.11, 8.12, 8.13, 8.14, and 8.15. As expected, the decompositions for larger values of k contain a larger number of regions of smaller size. As an example of this, notice that some of the regions in Figure 8.14 are too small or irregularly shaped, and therefore not likely useful for navigation purposes. It can also be observed in the figures that the 1-overlapping decompositions had a larger number of regions than when no overlapping was required, which is a reasonable thing to expect. Also it is interesting to note that the regions of the 1-overlapping decompositions are generally more regularly shaped than their 0-overlapping counterparts. This is a natural consequence of the method used to obtain these type of decompositions, which imposes a minimum diameter to the obtained regions. As an example compare Figures 8.12 and 8.13, and Figures 8.14 and 8.15, in which the regions in the 1-overlapping decomposition seem more suitable for navigation than those obtained for $\rho = 0$.

These results, using real feature visibility data, appear to be satisfactory for robot navigation, since regions of acceptable size and without many holes were achieved in the decompositions for sufficiently large values of k. As mentioned earlier, the size of the regions in the decomposition is strongly influenced by the size of the feature visibility regions. From the results that we obtained in our experiments, we can assert that if distinctive features that remain stable through larger areas of pose space are available, and it is employed a matching method that achieves larger visibility regions without compromising feature distinctiveness during the on-line localization stage, then the proposed method of decomposition into regions is a promising and practical technique to accomplish reliable view-based robot navigation.

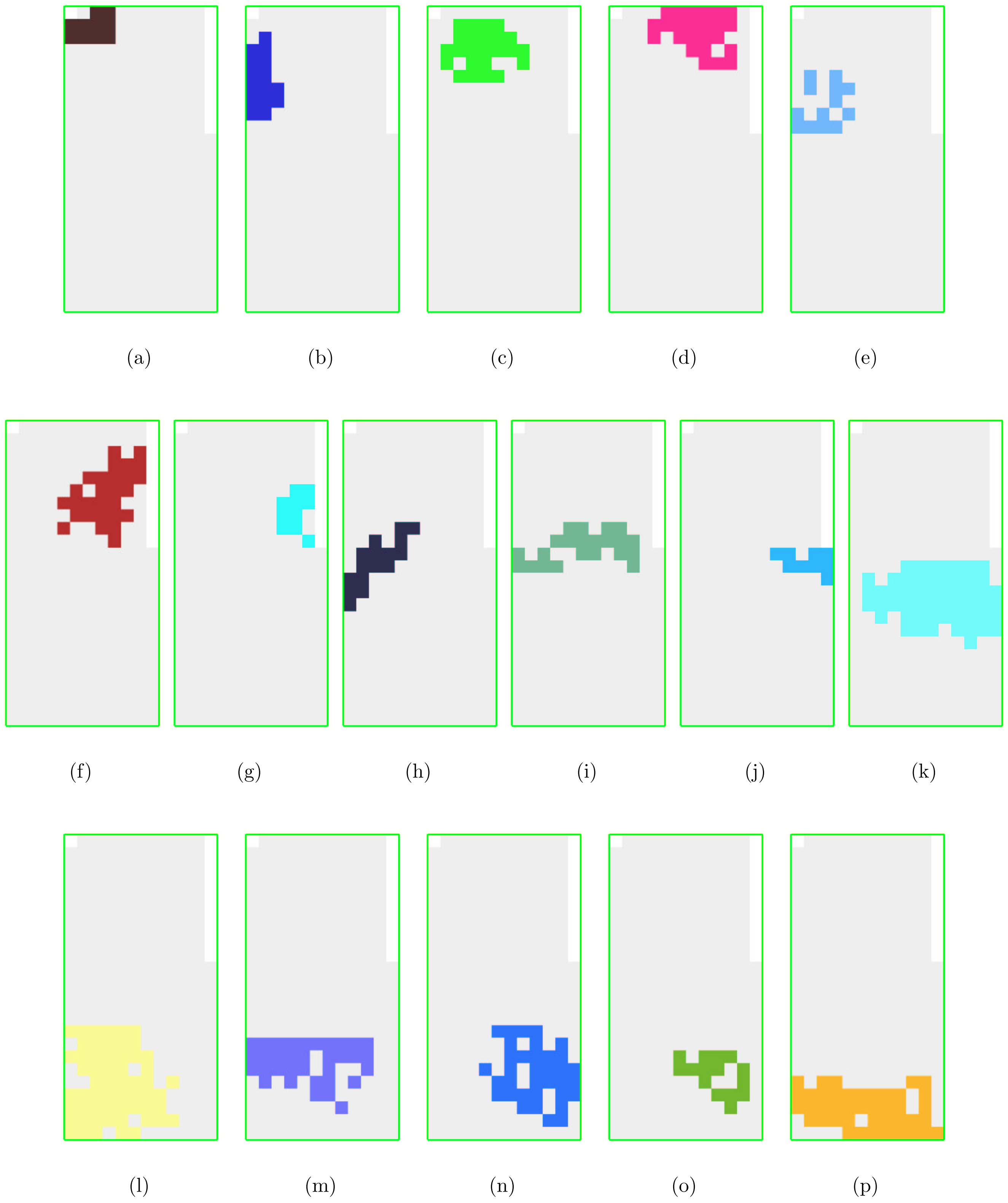


Figure 8.14: Region decomposition of the 6m by 3m Real world for $k = 10$ and $\rho = 0$ using Algorithm B.x.

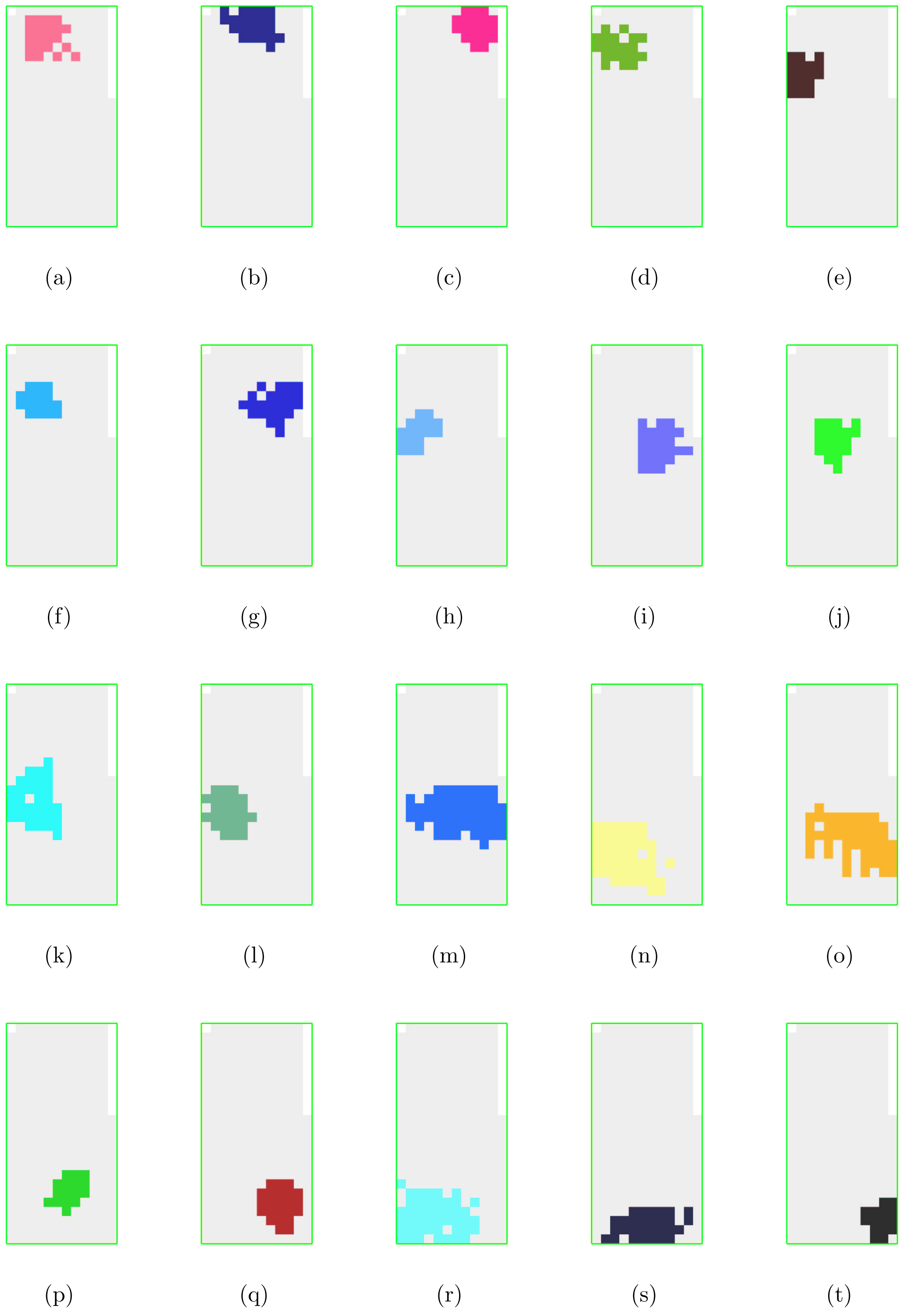


Figure 8.15: Region decomposition of the 6m by 3m Real world for $k = 10$ and $\rho = 1$ using Algorithm B.x.

Chapter 9

Conclusions

We conclude this thesis reviewing the main contributions made and we discuss possible directions for future research.

9.1 Contributions

The main contributions of this thesis can be outlined as follows:

1. We have presented a novel graph theoretic formulation of the problem of automatically extracting an optimal set of landmarks from an environment for visual navigation.

2. We have analyzed its complexity and shown that the problem is intractable.

3. We have developed six algorithms to solve for approximate solutions to the problem.

4. We have developed a simulator on which these six, and any other such algorithm, can be evaluated.

5. We have evaluated the algorithms on both synthetic and real data.

6. We have developed a technique for computing the input grid graph, including a feature tracking method and a grid point localization method.

7. We have developed a technique for computing the position and orientation at which a novel view was acquired, i.e., for localizing the robot.

9.2 Future Work

There are a number of extensions to this work that we plan to pursue:

- Integrating the image collection phase with the region decomposition stage into a unique on-line process as the robot is exploring its environment, as in a view-based SLAM fashion.

- Path planning through decomposition space, minimizing the number of region transitions in a path.

- Extend the proposed framework to detect and cope with environmental change.

- Compute the performance guarantee of our heuristic methods and provide tight upper bounds on the quality of our solution compared to those of optimal decompositions.

- Study the use of feature tracking during the image collection stage, to achieve larger areas of visibility for each feature, since tracking the features between images taken from adjacent viewpoints allows for tracking small variations of appearance (which may integrate to large ones over large areas). Such a framework would require maintaining equivalence classes of features in the database.

Bibliography

[1] C. Ashcraft and R. Grimes. SPOOLES: An Object-Oriented Sparse Matrix Library. In *Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing*, San-Antonio, Texas, USA, 1999.

[2] Ronen Basri and Ehud Rivlin. Localization and Homing Using Combinations of Model Views. *Artificial Intelligence*, 78(1–2):327–354, October 1995.

[3] Margrit Betke and Leonid Gurvits. Mobile Robot Localization using Landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, April 1997.

[4] Stan Birchfield. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. http://vision.stanford.edu/∼birch/klt.

[5] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Wellesley, MA, USA, 1996. http://www-personal.engin.umich.edu/∼johannb/my_book.htm.

[6] M. Bosse, P. Newman, J. Leonard, and S. Teller. An Atlas Framework for Scalable Mapping. In *IEEE International Conference on Robotics and Automation*, Taiwan, September 2003.

[7] A. Briggs, D. Scharstein, and S. Abbott. Reliable Mobile Robot Navigation From Unreliable Visual Cues. In *Fourth International Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, USA, March 2000.

[8] Gustavo Carneiro and Allan D. Jepson. Multi-scale Phase-based Local Features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 736–743, Madison, WI, USA, June 2003.

[9] H. Choset and K. Nagatani. Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, April 2001.

[10] I. J. Cox. Blanche - An Experiment in Guidance and Navigation of an Autonomous Mobile Robot. *IEEE Transactions on Robotics and Automation*, 7(3):193–204, 1991.

[11] Andrew Davison. Real-Time Simultaneous Localization and Mapping with a Single Camera. In *IEEE International Conference on Computer Vision*, Nice, France, 2003.

[12] M. Deans and M. Hebert. Experimental Comparison of Techniques for Localization and Mapping Using a Bearing Only Sensor. In *International Conference on Experimental Robotics*, Honolulu, Hawaii, December 2000.

[13] Gregory Dudek, Paul Freedman, and Souad Hadjres. Using Multiple Models for Environmental Mapping. *Journal of Robotic Systems*, 13(8):539–559, August 1996.

[14] Gregory Dudek and Deeptiman Jugessur. Robust Place Recognition using Local Appearance based Methods. In *IEEE International Conference on Robotics and Automation*, pages 1030–1035, San Francisco, CA, USA, April 2000.

[15] C. L. Feng and Y. S. Hung. A Robust Method for Estimating the Fundamental Matrix. In *International Conference on Digital Image Computing*, pages 633–642, 2003.

[16] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381–395, 1981.

[17] T. Grossman and A. Wool. Computational Experience with Approximation Algorithms for the Set Covering Problem. *European Journal of Operational Research*, 101(1):81–92, August 1997.

[18] Richard I. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[19] R. Karp. Reducibility among Combinatorial Problems. In Miller and Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, USA, 1972.

[20] Benjamin Kuipers and Yung-Tai Byun. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Robotics and Autonomous Systems*, 8:46–63, 1991.

[21] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic, Boston, MA, USA, 1992.

[22] C. Lin and R. Tummala. Mobile Robot Navigation Using Artificial Landmarks. *Journal of Robotic Systems*, 14(2):93–106, 1997.

[23] Brad Lisien, Deryck Morales, David Silver, George Kantor, Ioannis Rekleitis, and Howie Choset. Hierarchical Simultaneous Localization and Mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, USA, October 2003.

[24] H. C. Longuet-Higgins. A Computer Algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[25] David Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, Kerkyra, Corfu, Greece, September 1999.

[26] David G. Lowe. Demo Software: SIFT Keypoint Detector. http://www.cs.ubc.ca/∼lowe/keypoints/, September 2003.

[27] Q.-T Luong, R. Deriche, O. Faugeras, and T. Papadopoulo. On Determining the Fundamental Matrix: Analysis of Different Methods and Experimental Results. Technical Report RR-1894, INRIA, 1993.

[28] Q.-T. Luong and O. Faugeras. The fundamental matrix : Theory, algorithms and stability analysis. *International Journal on Computer Vision*, 17(1):43–76, 1996.

[29] M. Mata, J. M. Armingol, A. de la Escalera, and M.A. Salichs. Learning Visual Landmarks for Mobile Robot Navigation. In *15th World Congress of IFAC International Federation of Automatic Control*, Barcelona, Spain, July 2002.

[30] Inhyuk Moon, Jun Miura, and Yoshiaki Shirai. Automatic Extraction of Visual Landmarks for a Mobile Robot under Uncertainty of Vision and Motion. In *IEEE International Conference on Robotics and Automation*, pages 1188–1193, Seoul, Korea, May 2001.

[31] A.M.M. Muijtjens, J.M.A. Roos, Th. Arts, A. Hasman, and R.S. Reneman. Tracking Markers With Missing Data by Lower Rank Approximation. *J. Biomechanics*, 30(1):95–98, 1997.

[32] I. Rekleitis, R. Sim, G. Dudek, and E. Milios. Collaborative Exploration for the Construction of Visual Maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1269–1274, Maui, HI, USA, October 2001.

[33] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001.

[34] J. Salas, J. Gordillo, and C. Tomasi. Visual routines for mobile robots. *Expert Systems with Applications*, 14(1–2):187–197, January 1998.

[35] S. Se, D. Lowe, and J. Little. Vision-Based Mobile Robot Localization and Mapping Using ScaleIinvariant Features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058, Seoul, Korea, May 2001.

[36] Stephen Se, David Lowe, and Jim Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, 21(8):735–758, August 2002.

[37] Robert Sim and Gregory Dudek. Learning and Evaluating Visual Features for Pose Estimation. In *IEEE International Conference on Computer Vision*, pages 1217–1222, Kerkyra, Corfu, Greece, September 1999.

[38] Robert Sim and Gregory Dudek. Effective Exploration Strategies for the Construction of Visual Maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2003.

[39] Saul Simhon and Gregory Dudek. A Global Topological Map formed by Local Metric Maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, October 1998.

[40] Kristian T. Simsarian, Thomas J. Olson, and N. Nandhakumar. View-Invariant Regions and Mobile Robot Self-localization. *IEEE Transactions on Robotics and Automation*, October 1996.

[41] G. W. Stewart. Perturbation theory for the singular value decomposition. Technical Report CS-TR-2539, University of Maryland, College Park, MD, USA, 1990.

[42] Karen T. Sutherland and William B. Thompson. Inexact Navigation. In *IEEE International Conference on Robotics and Automation*, pages 1–7, Atlanta, GA, USA, May 1993.

[43] K. Tashiro, J. Ota, Y.C. Lin, and T. Arai. Design of the Optimal Arrangement of Artificial Landmarks. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 407–413, 1995.

[44] Sebastian Thrun. Finding Landmarks for Mobile Robot Navigation. In *IEEE International Conference on Robotics and Automation*, pages 958–963, Leuven, Belgium, May 1998.

[45] P.H.S. Torr and D.W. Murray. The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix. *International Journal on Computer Vision*, 24(3):271–300, September/October 1997.

[46] D. Wilkes, S. Dickinson, E. Rivlin, and R. Basri. Navigation Based on a Network of 2D Images. In *ICPR-A*, pages 373–378, 1994.

[47] Z. Zhang, R. Deriche, Q.-T. Luong, and O. Faugeras. A Robust Approach to Image Matching: Recovery of the Epipolar Geometry. In *International Symposium of Young Investigators on Information\Computer\Control*, pages 7–28, Beijing, China, February 1994.

[48] Zhengyou Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal on Computer Vision*, 27(2):161–198, 1998.