Real-Time Applications of 3D Object Detection and Tracking

Thesis by

Jeremy Ma

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2010

(Defended October 19, 2009)

To mom, for teaching me to be that better person,
and to Priscilla for loving me those times when I'm not.

Acknowledgments

I would like to extend my warmest thanks and appreciation to my advisor, Joel Burdick, for his endless support and advice — not just in academic avenues but in life as well. Without his guidance, much (if not all) of this thesis would not have been possible.

I would also like to thank Richard Murray for his wise counsel and especially for allowing me the opportunity to serve in the DARPA Urban Challenge, a truly rewarding experience on so many levels. The support and instruction of my thesis committee members, Pietro Perona and Jim Beck, have been invaluable and for that I am truly appreciative.

I am thankful for the network of people in the Thomas building who have made my time at Caltech enjoyable, in particular Maria Koeper, Chris Silva, and Carolina Oseguera for the wonderful work they do and without whose efforts my defense of this thesis would not have been possible to schedule. A special thanks to Alejandro Delgado for helping me to maintain my office and the lab. The support of the SOPS community is also much appreciated for the soda, the candy, and the social hours.

The fellowship of the robotics research group is greatly appreciated, for the collaboration and the Ernie's lunches. In particular, I am especially grateful for the friendship and support of my fellow researcher and officemate Noel DuToit.

I would like to thank my family for their love and their support, which is beyond measure. Thank you for making me the person I am today. A special thanks to Patricia for letting me relive my high school years when graduate school seemed too difficult.

Finally, my deepest gratitude and love goes to my wife and better half, Priscilla — without her, I have nothing and with her I have everything.

Abstract

Robot perception is a fundamental aspect of any autonomous system. It gives the robot the capacity to make sense of vast amounts of data and gain an understanding of the world around it. An active problem in the area of robot perception is real-time detection and pose estimation of 3D objects. This thesis presents an approach to 3D object detection and tracking utilizing a stereo-camera sensor. Geometric object models are learned in short order time via a training phase and real-time detection and tracking is made possible by performing sparse stereo calculations on the chosen features within an adaptive region of interest of the camera image. The experimental results obtained by using this method will show the effectiveness of the approach as compared against ground truth measures in real-time. Using that framework as a basis, extensions to two other problems in robot sensing are then considered: (1) sensor-planning for model identification, and (2) sensor-planning for object-search. In the former, a novel algorithm for determining the *next-best-view* for a mobile sensor to identify an unknown 3D object from among a database of known models is presented and tested across two experiments involving real robotic systems. An information theoretic approach is taken to quantify the utility of each potential sensing action and the validity of the algorithm is discussed. In the latter area, a novel approach is presented that allows an autonomous mobile robot to search for a 3D object using an onboard stereo camera sensor mounted on a pan-tilt head. Search efficiency is realized by the combination of a coarse-scale global search coupled with a fine-scale local search, guided by a grid-based probability map. Obstacle avoidance during the search is naturally integrated into the method with additional experimental results on a mobile robot presented to illustrate and validate the proposed search strategy. All presented experiments were carried out in real-time processing with modest computation done by a single laptop computer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis develops new methodologies for 3D object detection and tracking in vision with applications to sensor planning and object search. Motivation for this research will be discussed, highlighting the need for the main contributions of this work. Previous works will briefly be reviewed and an outline of the thesis contributions made will be given.

1.1 Motivation

A large part of robot autonomy is characterized by *perception*, the ability of a robot to sense its environment and interpret it in an intelligent way. While humans have been able to solve this problem with only a few types of proprioceptive and exteroceptive sensors, the robotic systems of today typically require a large number of exteroceptive sensors coupled with a significant source of computing power. Nonetheless, significant advances in perception have still been made with applications in military defense, assisted driving, and even home living.

One of the rapidly maturing areas of robot perception has been in the field of computer vision where the primary sensor is a camera and sensory data comes in the form of images — grayscale and color. Historically, limitations in computing bandwidth have restricted many algorithms to off-line *batch* implementation, with real-time processing issues less well explored. Not until recently, the expense of high-quality, precision cameras limited many algorithms to using a single monocular camera — sacrificing the richness of data available from a second camera (such as in stereovision applications).

With the recent advent of faster computers and cheaper cameras, real-time processing of vision data from multiple cameras has become much more common in robotics research over the past decade. A particular problem enabled by the availability of modern hardware is *3D object-detection with tracking*. While past methods have achieved robust object-detection in static images ([18] provides a good review), the following section will discuss the recent advances that have extended existing methods to real-time 6D pose estimation.

The focus of the research presented here is to consider the areas of robot perception that can benefit from the capabilities of real-time 3D object detection and tracking. Of particular interest is the area of sensor-planning for model identification and object-search. Sensor-planning for model identification considers the problem of planning sensor movements to acquire improved sensory data in order to identify an unknown model. As the next section will discuss, many existing methods have yet to incorporate full object pose estimation into the analyses and as such, model identification is often limited to only static 2D images. Similarly, in the area of object-search — i.e., the problem of finding a specific object in a given environment — current methods have yet to fully integrate complete 6D pose estimation of the searched object, thus limiting the possibility of accurate object placement in a global coordinate frame, a useful capability for service robots.

1.2 Background and Related Works

The problem of 3D object detection and tracking is not a new one and has seen its fair share of proposed solutions. However, much of the prior work on object detection has focused on off-line computation thereby limiting the real-time "tracking" aspect. In this context of *3D object recognition*, most approaches first detect and recognize an object, then estimate its 6D pose. Of the developed methods, a majority of them can be classified as either *appearance based* ([23], [26], [43], [45]) where descriptors (i.e., features) derived from images of the object are utilized, or *geometrical based* ([7], [28], [36], [50]) where geometric models are constructed for each object and shape-matching techniques (e.g., aspect-graphs) are then used. In either case, a *training-phase* is first implemented whereby the object is first learned from varying *viewpoints* – a process that accounts for detection from any number of possible viewing angles. While a majority of research has shifted towards *appearance-based* methods due to the success of various approaches in recent years, the related problem of *feature-correspondence* (i.e., matching features detected in the image with the correct features on the known object) has become equally important ([9]).

Quite notable among appearance-based methods is the work by Ponce et al. ([43], [45]) who present an off-line method based on a collection of small planar patches (a type of visual invariant that incorporates the 3D spatial relationships to other object features as collected from varying viewpoints around the object). Ferrari et al. in [15] and Kushal and Ponce in [23] improve on the work of [45] by utilizing multiple model views and groups of aggregate matches (*GAMs*) – groups of matched sets that serve as the unit of matching between model views and the test image. For each initial match set, the authors implement a series of *expansion* and *contraction* phases that explore nearby regions in the model view and the test image to collect more matches and prune away mismatches. While the work of these authors is notable, the training phase associated with these methods is an involved process that takes on the order of hours to generate models for each object.

While much ground has been gained in the area of object recognition in recent years, it is not to say that previous efforts ignored real-time approaches to object tracking with pose estimation altogether. Most notable is the early work by DeMenthon and Davis in [10] who first developed the POSIT algorithm, a method of determining the 6D pose of an object given four or more noncoplanar feature points in the image matched to geometric points on the object. While their approach is fast and simple and can be implemented with a basic monocular camera, it is highly susceptible to instability caused by mismatches.

More recently, new advances are still being made such as the work by Najafi et al. in [36]. These authors propose an approach similar to DeMenthon and Davis, but which is more robust to mismatches. Their method utilizes a set of calibrated images of the object with the 3D geometric model, and image feature correspondence is achieved via a combination of PCA and a Bayesian classification framework where a 1.5–3 Hz tracking rate is possible with a monocular camera. Vacchetti et al. in [54] take an approach similar to [36], yet achieve tracking rates at about 25–30 Hz for 320×240 sized images from a monocular camera. In their approach, batch methods are used to generate a 3D object model from a series of *keyframes,* and object pose estimation is realized by matching features between the test image and the closest *keyframe*. Drawbacks of their approach, however, are the requirement that the object be “close” to a known keyframe to ensure robust object registration and the algorithm’s sensitivity to scale and rotation changes — which is a direct result of the corner patch features chosen by the authors.

With the introduction of SIFT features in 1999 by David Lowe [29], many previous problems associated with feature mismatches caused by scale or rotational variance have been alleviated, allowing greater accuracy and stability. Panin and Knoll [40] introduce a real-time solution that is invariant to scale and rotation via SIFT. Their approach uses a monocular camera and SIFT features to detect and initialize an object with sustained tracking achieved by switching to a contour-based tracking algorithm. Choi et al. in [5] extend the work of Panin and Knoll by demonstrating real-time 6D pose estimation and tracking for both monocular and stereo cameras. Rather than applying SIFT-base feature matching on each frame, the object pose is only initialized using SIFT (similar to the approach of [40]), and then a Lucas-Kanade (LK) tracker [31] is used. Pose estimation for the monocular case is done using the POSIT algorithm [10], while a closed-form solution involving unit quaternions handles the stereo case. However, their proposed system is susceptible to large delays caused by SIFT reinitialization when tracks are lost due to large object motions.

These recent works have made clear the fact that the quality of pose estimation and the speed of tracking is directly related to the type of feature used. SIFT features offer a high probability of accurate feature correspondence, but come at the cost of computational speed. Similarly while other features like Harris corner patches offer a significant speedup in computational speed, they come at a cost of reduced quality in feature correspondences. This posits the need for a real-time object pose estimation method that is both accurate in feature correspondence and fast in tracking rate. The contributions of chapter 3 will highlight a particular approach that does exactly this.

One important research area that can benefit from real-time 3D object detection and tracking is sensor planning. While the subject of sensor planning itself is vast, two topics are relevant to this thesis: *model identification* and *object search*. Sensor planning for model identification considers the situation when not all of an object model may be visible, and the visible parts may not be discernible from another known object. It investigates solutions to the question: where does one move next to improve the quality of sensing and best discern what object is being seen? Sensor planning for object search considers the problem of finding a known object in a given environment with or without prior information on its whereabouts. It investigates solutions to the similar question: how should one move next to improve the probability of object detection and localization?

In previous years, the problem of sensor planning has been considered in the context of various applications: autonomous driving ([52], [41]), object search ([56]), visual mapping ([8],[51]), multiple target detection and tracking ([20], [48], [22]). The application of sensor planning to the problem of *model identification* has also been investigated to some extent. In computer vision, the problem is often categorized as an *active recognition* problem ([4], [44]), i.e., determining the appropriate set of actions to execute for a visual agent to gather enough evidence to disambiguate initial object hypotheses. Recently, the application of information theoretic concepts has been investigated as a possible solution in off-line and on-line methods to this problem.

Arbel et al. [1] consider an approach that discretizes the viewing sphere around an unidentified object and populates grid cells with entropy values. Notable in their work is that sensor planning for object recognition is achieved by selection of the most informative view based on the precomputed entropy maps. Paletta et al. in [39] take a more practical approach and present a Bayesian fusion method that incorporates the temporal context of observations by integrating multiple recognition results. Through a Markov Decision process, a planning scheme is selected that minimizes the expected entropy loss. Their proposed real-time algorithm shows interesting results for object classification using Bayesian sequential recognition. However, their work lacks in its ability to track and estimate the 6D continuous pose of model objects.

In [11], Denzler and Brown choose mutual-information as their cost metric and present a similar sequential decision-making process for choosing both optimal gaze-control inputs and viewpoint action selections. Monte Carlo sampling is employed to approximate an analytical solution, which partly adds to the computational complexity of their approach but yields rather accurate results for their chosen features – which are the same features as [39]. More recently Eidenberger et al. in [12] present a sequential Bayesian method for active object recognition with a cost metric defined as the upper bound of the differential entropy. Their Bayesian analysis is done in such a way that the prior and posterior distributions of the state variable are framed as mixtures of Gaussians, allowing for fast parametric updates. Additionally, their framework allows for (planar) pose estimation of each object that is shown in simulation. However, their presented experiments on real data are limited and real-time capabilities with 6D pose estimation have yet to be explored.

Aside from information theoretic approaches to the active recognition problem, advances have been made with other approaches as well. In recent years, Laporte et al. in [25] considered the active recognition problem in the context of Fisher's linear discriminant analysis. The authors incorporated a *separability measure* between the expected observations of one model and the observations of all the other models in the set for all possible viewpoints. As later detailed in [24], their introduced measure can be computed off-line which facilitates fast viewpoint selection. However, their presented approach suffers when applied to real data images, as opposed to simulated views.

It is clear that the problem of sensor planning for model identification has yet to incorporate simultaneous, on-line 6D pose estimation in any of the above methods. Chapter 4 will present an information theoretic approach to sensor planning that shows via Bayesian analysis that object pose estimation is not only an added feature but a necessary step to the next-best-view calculation. Furthermore, the object itself is no longer confined to a stationarity assumption and active recognition in a dynamic setting is explored.

The problem of *object-search* has been considered as an element of sensor planning research in recent years as well. Ye and Tsotsos in [56] developed one of the first systematic frameworks for object search that incorporated both sensor planning and object recognition. Their method was developed for a two-wheeled robot equipped with a pan-tilt-zoom camera and a laser eye. Their spherically arranged training data set encodes the probability that a given sensor movement on a sphere surrounding the object will improve detection. Their computationally expensive method can be tedious to implement given its need for the experimental construction of a detection function for all sensing parameters (pan, tilt, zoom, robot orientation) under various lighting conditions, object orientations, and background effects. Furthermore, the object recognition function is limited to a 2D technique using a blob finder based on pixel intensity.

More recently, Saidi et al. ([46], [47]) extend the work of Ye and Tsotsos to a humanoid robot where object recognition is carried out via 3D SIFT features. They present a visual attention framework that relies upon pan-tilt-zoom capabilities to generate 3D data of the sensed environment. They formulate search as the problem of optimizing sensor actions and trajectories with respect to a utility function that incorporates target detection probability, new information gain, and motion cost. A visibility map similar to the sensed sphere of [56] filters uninformative sensing actions. While their approach is a significant improvement on the work of Ye and Tsotsos, the visibility map calculations are computationally expensive and their utility function lacks a formal Bayesian framework.

The probabilistic approach used by Chung et al. in [6] to solve an abstract object search problem provides the Bayesian framework lacking in [47]. There the authors develop a recursive Bayes' Filter for updating the probability of object existence in each cell of a grid map, and various different search strategies are considered with the *saccadic* search method yielding the minimum average search time – an approach that mimics the search patterns in human visual attention ([19]). Nonetheless, their method must be further developed for any specific implementation.

Petersson et al. in [42] consider the problem of object search in the context of grasping and manipulation. They use a support vector machine for object recognition on a robotic platform equipped with an arm, a laser scanner, sonar, a torque sensor, and a color camera. Once recognized, the object is tracked in a *window of attention*. Though the problem of object search *with* manipulation is addressed, a crude object recognition system is used, and object pose estimation is limited to the process of aligning the current object image with a predefined reference image – an approach that works only on piecewise planar objects in positions that match the reference image and pose.

Building and improving upon the work of Petersson et al., Ekvall et al. [14] and Lopez et al. [27] decompose the object search problem into global and local search stages. Their coarse global search employed Receptive Field Coocurrence Histograms [13] to identify potential object locations. A mobile robot equipped with laser, sonar, and a pan-tilt-zoom camera then zooms into each hypothesized location to apply a localized object search algorithm (based on SIFT features). An a priori map built via SLAM is used to establish likely locations of known objects. Navigation is restricted to planning over a graph of predetermined "free-space" nodes. This approach simplifies the methods of [56] and [47] and allows for simultaneous search of multiple objects. However, their approach is limited to planar objects whose pose is crudely approximated by a single laser scan point in [14] and later moderately refined in [27] to a distance measure based on comparing the number of occupied pixels in the image against a reference image. Furthermore, much prior information is assumed given or computed off-line (e.g., the SLAM-based map and the set of navigation nodes). The contributions of chapter 6 will improve on the work of [56] by using a 3D object detector and will also simplify the method of [47] by replacing the computationally expensive 3D visibility map and rating function with a global and local search technique that updates the grid-based probability map incorporated from [6].

1.3 Contributions and Thesis Organization

The contributions and organization of this thesis are as follows. Chapter 2 establishes technical background that supports the contributions in later chapters. It presents existing techniques and strategies used in control theory and computer vision.

Chapter 3 presents a novel approach to real-time 6D pose estimation and tracking of 3D objects using stereo. In the majority of the works noted above, the focus has been on achieving pose estimation with monocular cameras – mainly due to the computational (and monetary) costs associated with high-fidelity stereo cameras at the time. The contributions in this chapter will extend the work of previous authors by showing how high-fidelity stereo-cameras can be an efficient means to obtaining robust, real-time object pose tracking. This chapter also describes how 3D geometric object models can be created in relatively short order time (minutes) using a stereo-camera. Furthermore, the experimental results show that robust tracking can be achieved using SIFT features at each time step (not just pose initialization, as was done in [40] and [5]), increasing the fidelity of the overall system to handling occlusions, changes in intensity, rotation, and scale. Lastly, a novel method for determining ground truth 6D pose estimates will also be introduced which does not require the purchase of third-party software as in [54], simulated experimental results as in [40], or distinguishing markers for computer vision toolkits as in [5].

Chapter 4 presents a novel approach to sensor planning that incorporates the 6D pose estimation algorithm developed in chapter 3. Using Bayesian analysis, the Sensor-Planning-for-Pose-Estimation-and-Model-Identification (SPPEMI) algorithm is derived. It solves for the optimal action for the next-best-view which optimizes an information gain metric. While the previous works of [1], [11], [12], [25], [38] and others considered static objects in confined workspaces, the derived approach makes no assumptions on the workspace, allowing for mobile objects sensed from mobile platforms.

Chapter 5 presents two specific experiments to highlight the applicability of the algorithm in chapter 4 in real-time. The first experiment considers a mobile object and mobile agent both of which are constrained in their movements to the projected viewing sphere, illustrating the capabilities of the overall SPPEMI algorithm to consider mobile objects. In the second experiment, unconstrained motion of the object and the viewing agent are considered demonstrating the success of the overall method to initialize a potential object, plan a sensor motion for improved sensing, and simultaneously track and estimate an object's pose all while executing the planned sensing action.

Chapter 6 considers another application of the 3D tracking algorithm of chapter 3 to the *object-search* problem, where a global and local search decomposition is used to not only locate the object but accurately estimate its 6D pose as well. A method is presented whereby an autonomous mobile robot searches for a 3D object using an onboard stereo camera sensor mounted on a pan-tilt head and estimates the 6D pose of the object once found. Search efficiency is realized by the combination of a coarse-scale global search coupled with a fine-scale local search over a grid-based probability map which is updated using Bayesian recursion methods. A grid-based costmap is also populated from stereo and used to facilitate obstacle avoidance and path-planning. Experimental results obtained from the use of this method on a mobile robot are also presented in this chapter to validate the approach, confirming that the search strategy can be carried out with modest computation.

Finally, chapter 7 summarizes the thesis and discusses directions of future work and research.

Chapter 2

Background

This chapter reviews fundamental concepts, algorithms, and necessary equations that support the overall work of this thesis. Since this work entails a great deal of 3D position estimation, section 2.1 presents a brief outline of the Bayes' Filter, a recursive tracking filter derived from the principles of Bayes' Law and is a form of Bayesian Sequential Updating. Section 2.2 presents an overview of SIFT (Scale-Invariant-Feature-Transform) features, a vision-based feature set first developed by David Lowe ([30]) and used in this work (and by other researchers as well) for their strong discriminating capabilities that greatly simplify the data association problem between measurements in images and the state elements to which they measure. Section 2.3 presents an overview of another type of vision-based feature developed by C. Harris and M. Stephens in [17], known as the Harris corner detector.[1] Harris corners are computationally affordable (and thus faster), though less robust to scale and rotation variance than SIFT features. Lastly, a major element of this thesis relies on accurate stereo-reprojection of 2D pixels to their corresponding 3D location in Cartesian space. Section 2.4 presents a brief review of epipolar geometry as applied to two optically aligned cameras which enables stereo vision in the most common sense.

2.1 Bayes' Filter

Bayes' Filter is a recursive algorithm derived from Bayes' Law that allows the state of a system to be represented by a probability distribution or a *belief.* In the field of robotics, the state of the system is often defined as the position and orientation of the robot, and the Bayes' Filter is a means of incorporating sensory data to update and continuously estimate the *pose* of the robot.

Let $\mathbf{X}_k \in \mathbb{R}^{n\times 1}$ be a continuous random variable representing the system state at the k-th timestep, $\mathbf{D}_{1:k} \in \mathbb{R}^{m\times 1}$ a continuous random variable representing the direct or indirect measurements of the system from timestep 1 up to and including timestep k, and $\mathbf{u}_{1:k} \in \mathbb{R}^{b\times 1}$ the corresponding control inputs to the system.[2] Then Bayes Filter is a recursive algorithm which cycles

[1]Harris corners are also sometimes referred to as Kanade-Tomasi corners, where the latter authors developed an improved version of the original algorithm by Harris and Stephens, by replacing the original corner measure with the minimum eigenvalue.

[2]Throughout the remainder of the thesis, the notation of $1:k$ will always be used to indicate the set of measure-

between the following two steps:

DYNAMIC PREDICTION:

$$p(\mathbf{X}_k|\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1}) = \int p(\mathbf{X}_k|\mathbf{X}_{k-1},\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})p(\mathbf{X}_{k-1}|\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})d\mathbf{X}_{k-1}\,, \tag{2.1}$$

MEASUREMENT UPDATE:

$$p(\mathbf{X}_k|\mathbf{D}_{1:k},\mathbf{u}_{1:k-1}) = \frac{p(\mathbf{D}_k|\mathbf{X}_k,\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})p(\mathbf{X}_k|\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})}{\int p(\mathbf{D}_k|\hat{\mathbf{X}}_k,\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})p(\hat{\mathbf{X}}_k|\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1})d\hat{\mathbf{X}}_k}\,. \tag{2.2}$$

2.1.1 The Kalman Filter

As an exercise, one can consider the particular case when the state is governed by linear dynamics and the measurements linearly dependent on the state of the system with only Gaussian noise. In this particular case, two propositions will prove useful in analytically solving the expressions of Bayes Filter. The propositions will be stated here and the proofs can be found in [53].

The first proposition deals with the integral of the product of two multivariate normal distributions and is useful in solving the dynamic prediction step of the filter:

Proposition 2.1: *Let $N(\mathbf{x};\mathbf{f}(\mathbf{y},\mathbf{u}),\mathbf{\Sigma}_x)$ be a normal distribution over the continuous random variable $\mathbf{x}$ with mean $\mathbf{f}(\mathbf{y},\mathbf{u})$ and covariance $\mathbf{\Sigma}_x$. Let $\mathbf{f}(\mathbf{y},\mathbf{u})$ be a linear function in $\mathbf{y}$ and $\mathbf{u}$ of the following form:*

$$\mathbf{f}(\mathbf{y},\mathbf{u}) = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{u} + \mathbf{C}\,.$$

Similarly, let $N(\mathbf{y};\mu_y,\mathbf{\Sigma}_y)$ be a normal distribution over the continuous random variable $\mathbf{y}$ with mean μ_y and covariance $\mathbf{\Sigma}_y$. Then the integral over $\mathbf{y}$ of the product of both normal distributions is itself a normal distribution, with mean and covariance given by

$$\begin{aligned} N(\mathbf{x};\mu^*,\mathbf{\Sigma}^*) &= \int N(\mathbf{x};\mathbf{f}(\mathbf{y},\mathbf{u}),\mathbf{\Sigma}_x)\cdot N(\mathbf{y};\mu_y,\mathbf{\Sigma}_y)d\mathbf{y}\,, \\ \mu^* &= \mathbf{f}(\mu_y,\mathbf{u})\,, \\ \mathbf{\Sigma}^* &= \mathbf{A}\mathbf{\Sigma}_y\mathbf{A}^T + \Sigma_x\,. \end{aligned}$$

The second proposition provides an analytical solution to the product of two multivariate normal distributions, and is useful in solving the measurement update step:

Proposition 2.2: *Let $N(\mathbf{y};\mathbf{f}(\mathbf{x},\mathbf{u}),\mathbf{\Sigma}_y)$ be a normal distribution over the continuous random variable $\mathbf{y}$ with mean $\mathbf{f}(\mathbf{x},\mathbf{u})$ and covariance $\mathbf{\Sigma}_y$. Let $\mathbf{f}(\mathbf{x},\mathbf{u})$ be a linear function in $\mathbf{x}$ and $\mathbf{u}$ of the*

ments, controls, etc. from timestep 1 up to and including timestep k.

following form:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}\,.$$

Similarly, let $N(\mathbf{x};\mu_x,\mathbf{\Sigma}_x)$ be a normal distribution over the continuous random variable $\mathbf{x}$ with mean μ_x and covariance $\mathbf{\Sigma}_x$. Then the product of both normal distributions is itself a normal distribution:

$$N(\mathbf{x};\mu^*,\mathbf{\Sigma}^*) = N(\mathbf{y};\mathbf{f}(\mathbf{x},\mathbf{u}),\mathbf{\Sigma}_y)\cdot N(\mathbf{x};\mu_x,\mathbf{\Sigma}_x)\,,$$

where:

$$\begin{aligned}\mu^* &= \mu_x + \mathbf{K}(\mathbf{y} - \mathbf{f}(\mu_x,\mathbf{u}))\,,\\ \mathbf{\Sigma}^* &= (\mathbf{I} - \mathbf{K}\cdot\mathbf{C})\mathbf{\Sigma}_x\,,\\ \mathbf{K} &= \Sigma_x\mathbf{C}^T(\mathbf{C}\cdot\Sigma_x\cdot\mathbf{C}^T + \Sigma_y)^{-1}\,.\end{aligned}$$

With the above propositions defined, now consider the following system dynamic model and measurement model:

$$\mathbf{X}_k = \mathbf{A}\cdot\mathbf{X}_{k-1} + \mathbf{B}\cdot\mathbf{u}_{k-1} + \mathbf{w}_{k-1}\,, \tag{2.3}$$

$$\mathbf{D}_k = \mathbf{C}\cdot\mathbf{X}_k + \mathbf{v}_k\,, \tag{2.4}$$

where $\mathbf{A}\in\mathbb{R}^{n\times n}$, $\mathbf{B}\in\mathbb{R}^{n\times b}$, and $\mathbf{C}\in\mathbb{R}^{m\times n}$ are state independent matrices and $\mathbf{w}_{k-1}\in\mathbb{R}^{n\times 1}$ and $\mathbf{v}_{k-1}\in\mathbb{R}^{m\times 1}$ represent Gaussian white process and measurement noise governed by covariance matrices $\mathbf{Q}\in\mathbb{R}^{n\times n}$ and $\mathbf{R}\in\mathbb{R}^{m\times m}$, respectively. The Bayes' Filter as defined by equations (2.1) and (2.2) can be analytically solved using propositions 2.1 and 2.2. Beginning with the dynamic prediction step, the belief of the state of the system is predicted forward as follows:

$$\begin{aligned}p(\mathbf{X}_k|\mathbf{D}_{1:k-1},\mathbf{u}_{1:k-1}) &= \int N(\mathbf{X}_k|\mathbf{A}\cdot\mathbf{X}_{k-1} + \mathbf{B}\cdot\mathbf{u}_{k-1};\mathbf{Q})\cdot N(\mathbf{X}_{k-1}|\mu_{k-1};\Sigma_{k-1})d\mathbf{X}_{k-1}\\ &= N(\mathbf{X}_k|\overline{\mu}_k;\overline{\Sigma}_k)\,,\end{aligned}$$

where the terms $\overline{\mu}_k$ and $\overline{\Sigma}_k$ are given by:

$$\overline{\mu}_k = \mathbf{A}\cdot\mu_{k-1} + \mathbf{B}\cdot\mathbf{u}_{k-1}\,, \tag{2.5}$$

$$\overline{\Sigma}_k = \mathbf{A}\cdot\Sigma_{k-1}\cdot\mathbf{A}^T + \mathbf{Q}\,. \tag{2.6}$$

The measurement update step of Bayes Filter works to integrate received sensor measurements into the belief of the state of the system:

$$\begin{aligned} p(\mathbf{X}_k|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) &= \eta \cdot N(\mathbf{D}_k|\mathbf{C} \cdot \mathbf{X}_k; \mathbf{R}) \cdot p(\mathbf{X}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &= \eta \cdot N(\mathbf{D}_k|\mathbf{C} \cdot \mathbf{X}_k; \mathbf{R}) \cdot N(\mathbf{X}_k|\overline{\mu}_k; \overline{\Sigma}_k) \\ &= N(\mathbf{X}_k|\mu_k; \Sigma_k)\,, \end{aligned}$$

where the terms μ_k and Σ_k are given by

$$\mu_k = \overline{\mu}_k + \mathbf{K}_k(\mathbf{D}_k - \mathbf{C} \cdot \overline{\mu}_k)\,, \tag{2.7}$$

$$\Sigma_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C})\overline{\Sigma}_k\,, \tag{2.8}$$

$$\mathbf{K}_k = \overline{\Sigma}_k \mathbf{C}^T (\mathbf{C} \cdot \overline{\Sigma}_k \cdot \mathbf{C}^T + \mathbf{R})^{-1}\,. \tag{2.9}$$

As should be obvious, the Bayes Filter reduces to the well-known Kalman Filter under linear system, Gaussian noise assumptions. Equations(2.5) and (2.6) implement the Kalman prediction step and equations (2.7) and (2.8) implement the Kalman update step with the dynamic Kalman gain given by equation (2.9). Note that the Kalman gain is a matrix that optimally weights the term $\mathbf{D}_k - \mathbf{C} \cdot \overline{\mu}_k^-$ (often referred to as the *innovation* or the *measurement residual*) such as to minimize the covariance of the error between the true state of the system and the best estimate of that state. While the derivation of the Kalman Filter as shown here is brief, references [32] and [55] provide more rigorous derivations.

2.1.2 The Extended Kalman Filter

Now consider the case when the system to be estimated does not have linear dynamics nor are the measurements linearly dependent on the state. Bayes' Filter can still be applied by approximating the system as locally linear.[3] Since the system is assumed non-linear, the system dynamic model and measurement model are different and generalized to be

$$\mathbf{X}_k = \mathbf{F}(\mathbf{X}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})\,, \tag{2.10}$$

$$\mathbf{D}_k = \mathbf{H}(\mathbf{X}_k, \mathbf{v}_k)\,, \tag{2.11}$$

where the variables $\mathbf{X}_k$, $\mathbf{u}_{k-1}$, $\mathbf{w}_{k-1}(\sim N(\mathbf{0}; \mathbf{Q}))$, and $\mathbf{v}_k(\sim N(\mathbf{0}; \mathbf{R}))$ are as previously defined for the linear case and $\mathbf{F}(\cdot) \in \mathbb{R}^{n\times 1}$ and $\mathbf{H}(\cdot) \in \mathbb{R}^{m\times 1}$ are non-linear functions that depend on the state and control variables and process/measurement noise. By applying the linear approximations,

[3]The particle filter is another method of applying Bayes' Filter to non-linear systems. However, they can be computationally expensive, often requiring a large number of particles for accurate results which may not always be well suited for real-time applications. As such, only the Extended Kalman Filter will be discussed here since that was the method used to implement Bayes' Filter in the experimental results of later chapters. Reference [53] provides a good review of particle filters for state estimation.

equations (2.10) and (2.11) can be reduced to

$$\mathbf{X}_k \approx \mathbf{A}_k \cdot \mathbf{X}_{k-1} + \mathbf{B}_k \cdot \mathbf{u}_{k-1}\,,$$
$$\mathbf{D}_k \approx \mathbf{C}_k \cdot \mathbf{X}_k\,,$$

where the matrices $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{C}_k$ are given by

$$\mathbf{A}_k = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1}}, \quad \mathbf{B}_k = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1}}, \quad \mathbf{C}_k = \left.\frac{\partial \mathbf{H}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}}.$$

With the linear approximations as defined above, the Bayes' Filter is then carried out in a manner analogous to the linear case. Beginning with the dynamic prediction step, the belief of the state of the system is predicted forward as follows:

$$\begin{aligned} p(\mathbf{X}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}) &= \int N(\mathbf{X}_k|\mathbf{F}(\mathbf{X}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}); \mathbf{Q}) \cdot N(\mathbf{X}_{k-1}|\mu_{k-1}; \Sigma_{k-1}) d\mathbf{X}_{k-1} \\ &= N(\mathbf{X}_k|\overline{\mu}_k; \overline{\Sigma}_k)\,, \end{aligned}$$

where the terms $\overline{\mu}_k$ and $\overline{\Sigma}_k$ are given by

$$\overline{\mu}_k = \mathbf{F}(\mu_{k-1}, \mathbf{u}_{k-1}, \mathbf{0})\,, \tag{2.12}$$
$$\overline{\Sigma}_k = \mathbf{A}_{k-1} \cdot \Sigma_{k-1} \cdot \mathbf{A}_{k-1}^T + \mathbf{Q}\,. \tag{2.13}$$

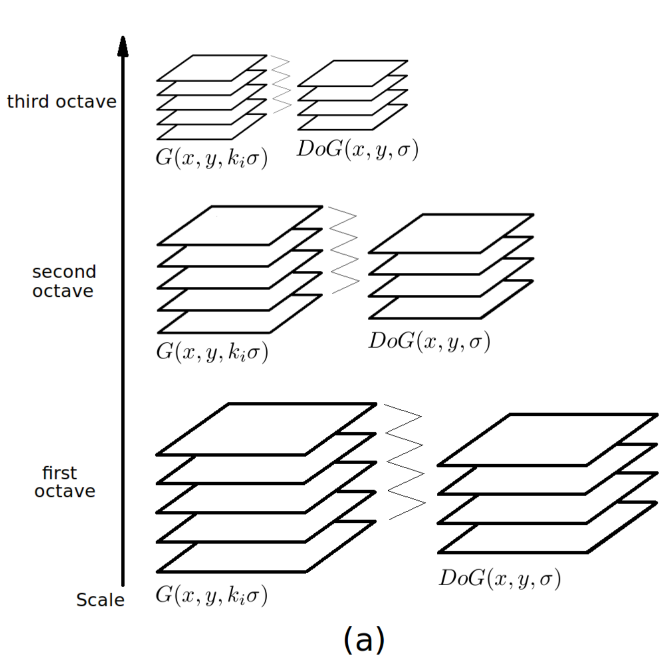
The measurement update step of Bayes Filter is then implemented with linear approximations as follows:

$$\begin{aligned} p(\mathbf{X}_k|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) &= \eta \cdot N(\mathbf{D}_k|\mathbf{H}(\mathbf{X}_k, \mathbf{0}); \mathbf{R}) \cdot p(\mathbf{X}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &= \eta \cdot N(\mathbf{D}_k|\mathbf{H}(\mathbf{X}_k, \mathbf{0}); \mathbf{R}) \cdot N(\mathbf{X}_k|\overline{\mu}_k; \overline{\Sigma}_k) \\ &= N(\mathbf{X}_k|\mu_k; \Sigma_k)\,, \end{aligned}$$

where the terms μ_k and Σ_k (and the Kalman Gain $\mathbf{K}_k$) are given by

$$\mu_k = \overline{\mu}_k + \mathbf{K}_k(\mathbf{D}_k - \mathbf{H}(\overline{\mu}_k, \mathbf{0}))\,, \tag{2.14}$$
$$\Sigma_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C})\overline{\Sigma}_k\,, \tag{2.15}$$
$$\mathbf{K}_k = \overline{\Sigma}_k \mathbf{C}^T(\mathbf{C} \cdot \overline{\Sigma}_k \cdot \mathbf{C}^T + \mathbf{R})^{-1}\,.$$

As was obvious in the linear case, the non-linear system (locally approximated as linear) reduces to the Extended Kalman Filter (EKF) algorithm. equations (2.12) and (2.13) correspond to the EKF



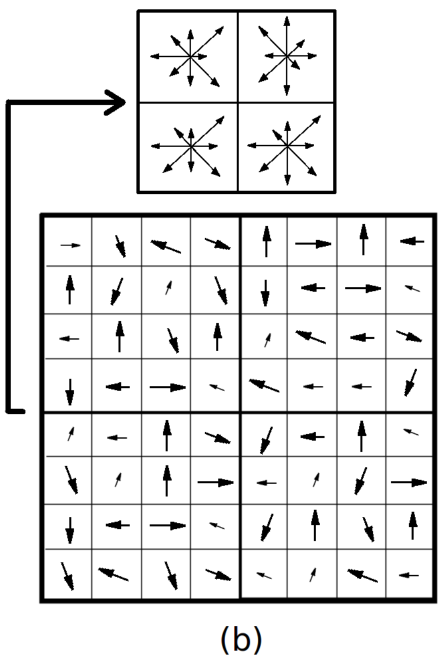


Figure 2.1: (a) Interest point detection is achieved by applying the Difference of Gaussians (DoG) operator in between adjacent levels of Gaussian blurred images within the same scale. The set of Gaussian blurred images within a scale define an octave. Each additional octave is then created by downsampling the image and repeating the calculations of determining the DoG images. (b) The feature descriptor is composed of an orientation histogram generated over a 4x4 sample region computed from a 16x16 sample array, though shown in the figure is a 2x2 region from an 8x8 sample array.

prediction step and equations (2.14) and (2.15) correspond to the EKF update step. It is important to note that the EKF is an adhoc approach of approximating non-linear systems as linear. In many cases, the system does not behave locally linear and the EKF should not be applied. Furthermore, the system and measurement noise will not necessarily follow a normal distribution after going through the non-linear transforms of $\mathbf{F}(\cdot)$ and $\mathbf{H}(\cdot)$ and is fundamentally inaccurate. Nonetheless, rather good results can be achieved for state estimation of non-linear systems with these linear approximations.

2.2 SIFT Features

SIFT features are distinctive, invariant features extracted from images that allow for efficient matching with various other viewpoints of the extracted features that may exist in the same or different scene. The features are invariant to scale, rotation, illumination, and noise and have been shown to yield a high probability of matching to sets of extracted (or known) features. As the invention of SIFT features came about from David Lowe's seminal paper in 1999 [29], much of the review presented here is a summary of the work presented in that paper. For further details, the reader is encouraged to read through Lowe's discussion and experimental results in the paper [30].

The calculation of SIFT features is done in four basic steps:

1. **Interest point detection in scale-space:** Candidate interest points are selected from the image by selecting those image pixels which yield either a local maxima or minima when compared across several image scales.

2. **Interest point selection and localization:** Of the set of candidate interest points found in Step 1, a stable set is selected and further localized in image space.

3. **Orientation assignment:** For all stable interest points selected, an orientation based on local image gradient directions and magnitudes is assigned. If multiple viable orientations exist, then new interest points are generated with identical localized positions and scale, yet differing orientation.

4. **Keypoint descriptor:** To handle invariance to rotation, each interest point is rotated up to 360° in 8 steps, relative to the interest point dominant orientation. For each stepped rotation of the interest point, an orientation histogram is generated over a 4x4 sample region computed from a 16x16 sample array. This yields a 128-element descriptor.

Each of the above steps will now be discussed in further detail to provide a better understanding of how the exact features achieve invariance properties.

2.2.1 Interest Point Detection in Scale-Space

Interest point detection in scale-space is achieved by applying a Difference of Gaussians (DoG) operator between various scales of the image. This is done by first considering the original image and creating an *octave* of Gaussian blurred copies of the original image, with each copy convolved with a Gaussian at a differing variance offset by a multiplicative constant k; that is

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)\ ,$$

where $G(x,y,k_i\sigma)$ is given by

$$G(x,y,k_i\sigma) = \frac{1}{2\pi\sigma_i^2} e^{-(x^2+y^2)/(2(k_i\sigma)^2)}\ .$$

In between adjacent copies of the image (yet within the same scale) the DoG operator is then applied, as seen in Figure 2.1(a). Once an octave of images is completed with the associated DoG images generated, the image is downsampled and a new octave is generated by repeating the above process. In creating these octaves of images, interest points are easily highlighted in each DoG image as local maxima and minima when compared with adjacent scales.

2.2.2 Interest Point Selection and Localization

Once a set of candidate interest points is generated, the location of each point is then interpolated via a second-order Taylor expansion to pinpoint the exact maxima or minima location, $\hat{\mathbf{x}}$, in image space:

$$D(x) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}\,,$$
$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}\,.$$

Before determining a candidate point as a valid interest point, the ratio of eigenvalues of principal curvature is checked for each point, to ensure the interest point does not lie simply on an edge. To avoid direct computation of the eigenvalues, Lowe borrows from the approach of Harris and Stephens in [17] and needs only to calculate the trace and determinant of the Hessian matrix and subsequently check the ratio of the two against some threshold, r, i.e.,

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix},$$
$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r},$$

which in Lowe's implementation, the ratio r is set to $r = 10$. Once the interest points are selected, the orientation of the interest point is next set.

2.2.3 Orientation Assignment

For each interest point selected with some associated scale, the local orientation of image gradients is then considered. That is, for the interest point (x_i, y_i), the corresponding Gaussian blurred image closest in scale is selected. A histogram of gradient orientations (organized into 36 bins to span the 360° of orientation) is then generated for a small neighborhood of pixels surrounding the interest point. Each entry to the histogram is weighted by the magnitude of the gradient and a Gaussian-shaped circular window. Once the histogram is generated, the peak orientation is selected and tagged to the interest point. In the event that the histogram has multiple peaks, a check is done to see if additional peaks are within 80% of the peak value. If that condition is satisfied, a second interest point is generated with the same maxima/minima pixel location and scale, yet differing orientation. The purpose of this step is to identify those features which may have large image gradients in several directions; enough to merit the creation of another descriptor to reduce the effects of a feature mismatch later on.

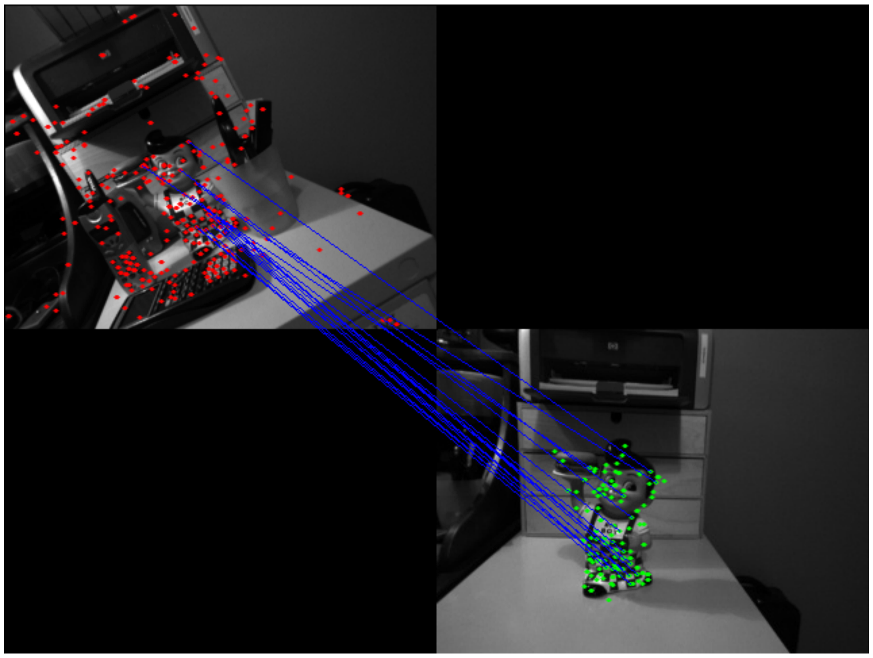


Figure 2.2: The above display shows the quality of feature matching achieved with SIFT features and the Best-Bin-First search algorithm. The object to recognize is displayed in the bottom right image. In the top left image, a cluttered and rotated scene containing the object is presented. Note the large number of matches (shown in blue) that exist despite the distractors, rotation, and partial occlusion.

2.2.4 Keypoint Descriptor

In this final step, invariance to rotation is achieved via careful construction of the interest point descriptor. This is done by rotating the interest point 360° in 8 rotation steps, relative to the interest point orientation assigned in the previous step. For each stepped rotation, an orientation histogram is generated over a 4x4 sample region computed from a 16x16 sample array. This yields a 128-element descriptor which improves the probability of accurately establishing feature correspondences across various rotations of the image scene. Figure 2.1(b) illustrates schematically how the orientation histograms might be computed for a given orientation and what the resulting sample region might look like.

2.2.5 Matching

Establishing feature correspondences is typically done by comparing the Euclidean distance between the 128-element interest point descriptors of currently detected SIFT features and those SIFT features known to exist in some database. The correspondence which yields the lowest Euclidean error is typically selected as the correct correspondence and the matched database feature is then eliminated from further correspondence consideration. However, the lowest Euclidean distance is not always the best match criterion to use, as some features may prove more discriminating than others with poor discriminating features matching quite often to spurious features or just noise. Furthermore, exhaustive search can be rather computationally expensive, especially when the database of features reaches an order of several hundred. Thus, to improve the rate of determining feature correspondences, two simplifications are often proposed when implementing matching of SIFT features:

1. a match criteria based not on lowest Euclidean distance, but the ratio of Euclidean distance of the nearest neighbor and the second nearest neighbor,

2. using a Best-Bin-First search algorithm that replaces exhaustive search.

The motivation behind the first simplification is that often low discriminating SIFT features can yield many mismatches that are undesirable. By comparing how well a feature matches to its nearest neighbor *and* its second nearest neighbor allows one to consider how far isolated the nearest neighbor match truly is (i.e., if the ratio of the two matched distances is small enough, then a valid correspondence is likely to have been found and should be kept).

The motivation behind the second simplification is more practical for implementation purposes. The Best-Bin-First search algorithm returns with high probability a feature's nearest neighbor by utilizing an ordered k-d tree. The search order itself requires the use of a heap-based priority queue that yields a speedup of search time by about two orders of magnitude. Figure 2.2 presents an example of the matching results for SIFT features used on a given object. The object to be matched is shown in the bottom right image and the presented scene image is shown in the top left. Note that despite the rotation of the scene image and large number of distractors, robust matching is still achieved.

2.3 Harris Corner Detectors

Corner detection is perhaps one of the first forms of feature extraction from images used in the field of computer vision. It began with the seminal work of Hans Moravec [34] whose approach to corner detection was to consider for each pixel a patch centered on that pixel and computing a weighted Sum-of-Squared-Differences (SSD) against a small set of neighboring patches generated by slight variations of the center pixel in x and y:

$$SSD = \sum_u \sum_v w(u,v)(I(u,v) - I(u+x, v+y))^2 \,,$$

where $w(u,v)$ is some nominal weight applied to the calculated Sum-of-Squared Difference, calculated as a function of pixel location. Figure 2.3 illustrates the concept behind Moravec's approach. Pixels at corners typically yield large values in all translations of x and y whereas pixels at non-corner locations typically yield low overall values. With regards to edges, the SSD will be prominent in

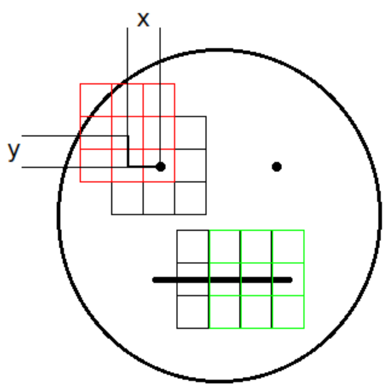


Figure 2.3: Corners are detected by generating a patch at a certain pixel location and calculating a Sum-of-Squared-Difference (SSD) between that patch and a set of neighboring patches. Pixels at corners (e.g., the red patch) typically yield large values in all translations of x and y whereas pixels at non-corner locations (i.e., noise) typically yield low overall values. With regards to edges (e.g., the green patch), the SSD can be prominent in translations of x and y perpendicular to the direction of the edge and low for translations of x and y along the edge.

translations of x and y perpendicular to the direction of the edge and low for translations of x and y along the edge until it reaches the ends of the edge when it reaches a maximal SSD value. One of the major drawbacks of Moravec's approach was that his operator was not *isotropic* in the sense that if an edge ran a direction not consistent with a translation to a neighboring pixel, then it would fail to be picked up as a corner.

In 1988, Harris and Stephens improved on the approach of Moravec in their seminal paper [17] by making the operator *isotropic* by performing an analytical expansion about the shift origin:

$$\begin{aligned} SSD &= \sum_u \sum_v w(u,v)(I(u,v) - I(u+x, v+y))^2 \\ &\approx \sum_u \sum_v w(u,v)(I_x(u,v)\cdot x + I_y(u,v)\cdot y)^2\,, \end{aligned} \tag{2.16}$$

where $I_x = \frac{\partial I}{\partial x}$and $I_y = \frac{\partial I}{\partial y}$ represent the image gradients in x and y at the (u,v) pixel. Note that equation (2.16) is often re-written in the following matrix form:

$$\begin{aligned} SSD &\approx [\begin{array}{cc} x & y \end{array}]\mathbf{A}\left[\begin{array}{c} x \\ y \end{array}\right], \\ \mathbf{A} &= \sum_u \sum_v w(u,v)\left[\begin{array}{cc} I_x & I_xI_y \\ I_xI_y & I_y \end{array}\right] \quad . \end{aligned}$$

With the SSD so formed, Harris and Stephens noted that the existence of corners can be easily deduced by the eigenvalues of the matrix $\mathbf{A}$, namely that if the eigenvalues (α,β) are both large, then the curvature of the image gradient is peaked in both directions and thus a corner exists. Note

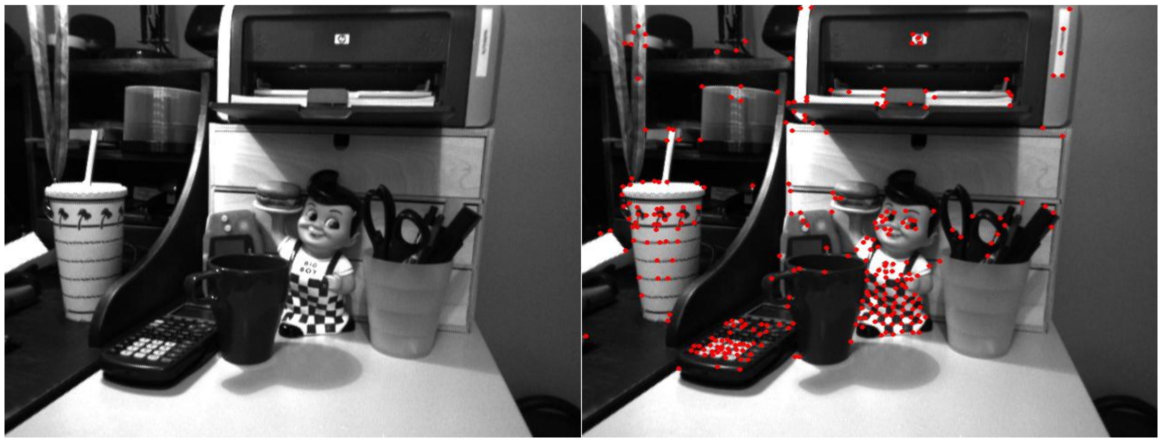


Figure 2.4: The above figure illustrates the application of the Harris corner detector to a given image shown on the left and the resultant detected corners shown in red on the right.

that this calculation holds only for one proposed shift in the center pixel by (x, y). In their proposed operator, Harris and Stephens took a conservative approach and declared a corner exists only if the response to their operator for all 8 possible neighboring shifts yielded a local maximum.

Lastly, to avoid computationally expensive operations for eigenvalue decomposition, an alternative metric is used when calculating Harris corners termed the "corner response" and is defined by

$$CR = Det(\mathbf{A}) - k \cdot trace(\mathbf{A})^2 \, .$$

This metric serves as a feasible alternative to the eigenvalue comparison, with the constant k often chosen in the range of $0.04 - 0.15$.

One of the benefits of the Harris corner feature, when compared to more sophisticated features like SIFT, is its speed. Figure 2.4 illustrates a simple application of the Harris operator to a given image scene, which is a 640×480 image that took 0.04 s to compute roughly 300 corners on a modest laptop computer with a 1.86 GHz processor.

2.4 Stereo Vision

Stereo vision calculates the 3D point location in Euclidean space of the object(s) causing the pixels in an image. While there exist many different approaches to stereo-vision, the method reviewed here will consider the common case of two cameras with coplanar image planes and parallel optical axes with aligned *x-axes*, a configuration that simplifies much of the epipolar geometry.

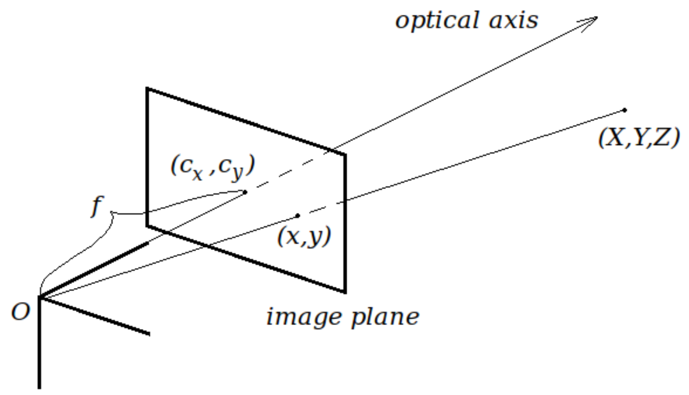


Figure 2.5: The pinhole camera model commonly used to describe the interworkings of most CMOS/CCD cameras.

2.4.1 Pinhole Camera Model

A common model used for most cameras is the pinhole-camera model, as shown in Fig. 2.5. The model consists of an image plane, some distance f – known as the focal length and typically measured in pixels – from the center of projection, denoted by O. Image coordinates are represented in pixels (x, y) measured relative to the image center (c_x,c_y). Using similar triangles, a point in 3D Euclidean space (X, Y, Z) can be transformed into image coordinates using the following relation:

$$x = \frac{f \cdot X}{Z}\,, \tag{2.17}$$

$$y = \frac{f \cdot Y}{Z}\,. \tag{2.18}$$

Note that going from 3D point locations to 2D image locations is a straightforward process, yet the converse is not possible unless the depth Z of the (x, y) pixel is known, or another equation is introduced. Alternatively, with a second camera, this slight complication can be remedied.

2.4.2 Two Pinhole Camera Models and Epipolar Constraints

Consider the diagram of Fig. 2.6(a) which shows two pinhole camera models. Let the reference camera in this case be the right camera (an arbitrary choice). Note that in the figure both cameras do not have coplanar image planes. Nonetheless, this serves as a good opportunity to briefly discuss epipolar geometry. An *epipole* is defined as the point of intersection with the image planes of the line joining the optical centers of both cameras (e_l and e_r in Fig. 2.6a). When a point $\mathbf{P}_0 \in \mathbb{R}^3$ is seen in the reference image, its location in the other image can potentially be anywhere along the line (called the *epipolar line*) generated by the intersection of the *epipolar plane* (i.e., the plane formed by the epipoles e_r and e_l and the point $\mathbf{P}_0$) and the image plane of the other camera.

Now consider the case of two pinhole cameras which have coplanar image planes, parallel optical

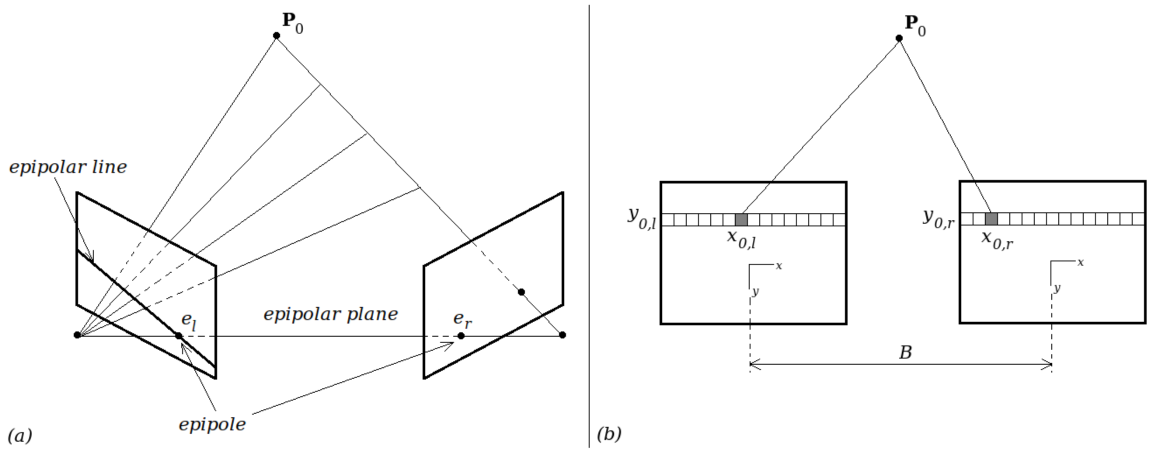


Figure 2.6: (a) A schematic explaining the epipolar geometry associated with two pinhole camera models. (b) For two cameras that have coplanar image planes and co-linear $x - axes$, the epipolar line associated with a point $\mathbf{P}_0$ is reduced to the same horizontal row at which the point was detected in the reference image.

axes, and colinear *x-axes* separated by a distance B as shown in Fig. 2.6b. In this particular case, the epipolar geometry reduces the epipoles of both cameras to lying on the same rows in each image plane. As such, the epipolar lines coincide with horizontal rows of the image and image point correspondences can be reduced to search along rows.

The pixel distance between the image point of $\mathbf{P}_0$ in the reference camera ($\mathbf{p}_{0,r} = (x_{0,r}, y_{0,r})$) and the found correspondent point in the secondary image ($\mathbf{p}_{0,l} = (x_{0,l}, y_{0,l})$) is defined as the *disparity*, d_0:

$$d_0 = x_{0,l} - x_{0,r} \, .$$

Note that $x_{i,l} > x_{i,r}$ always. Knowing the disparity and the separation distance between cameras (baseline) B, the following relation can be used to determine the depth associated with $\mathbf{p}_{0,r}$ (using similar triangles):

$$Z_0 = \frac{f \cdot B}{d_0} \, ,$$

and with Z_0 known, the distance in x and y can be deduced from equations (2.17) and (2.18):

$$X_0 = \frac{x_{0,r} \cdot Z_0}{f} \, ,$$
$$Y_0 = \frac{y_{0,r} \cdot Z_0}{f} \, .$$

By repeating the above calculations for all pixels $p_{i,r}$ in the reference image, a 3D depth image can be easily produced.

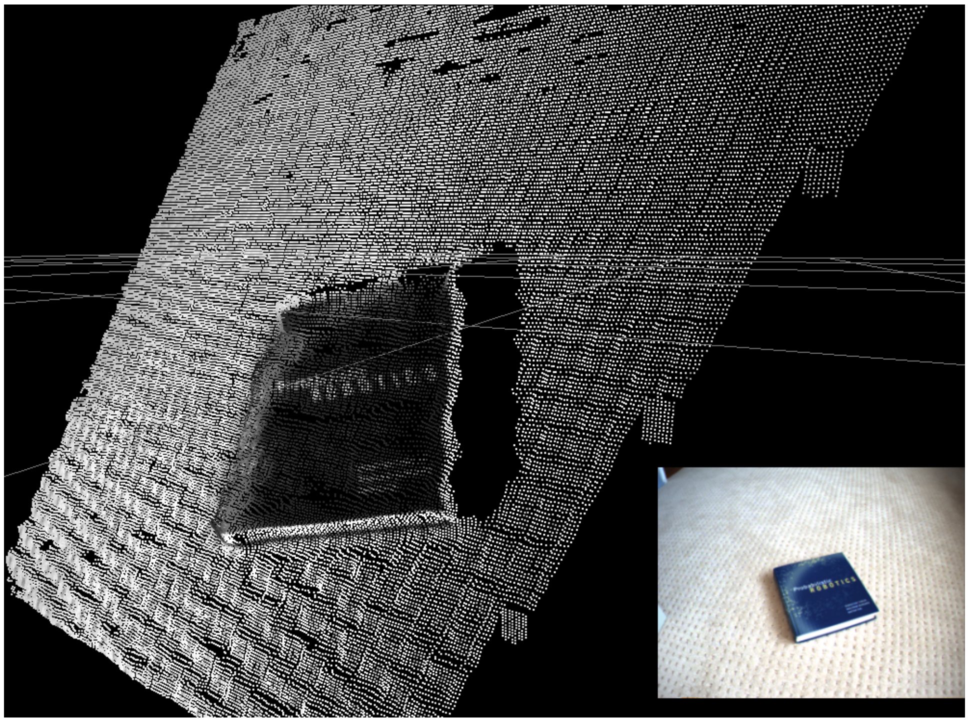


Figure 2.7: Shown is a plot of 3D stereo reprojected data using a stereo camera with a baseline of 0.12 m. The reference image is shown in the bottom right.

2.4.3 Practical Considerations

For the two pinhole-camera models described above, the simple case of both cameras having coplanar image planes and parallel optical axes was considered. However, in all real-life situations, cameras in that configuration are never perfectly aligned no matter how well calibrated a stereo-camera pair may be. Furthermore, there are barrel affects associated with all types of lenses that produce distortion around the periphery of the image resulting in non-linear effects which invalidate the horizontal epipolar line assumption. Lastly, the CMOS or CCD chip of the camera is never perfectly aligned with the optical axis. As such, the focal length f is rather difficult to come by, typically resulting in two components that need to be accounted for: the focal length in x (f_x) and the focal length in y (f_y).

Oftentimes much of these practical issues can be accounted for by calibrating the cameras and applying transformations to the image in the form of distortion coefficients that serve to rectify the image and align the focal length. This process of rectification is generally carried out using a calibration image (i.e., a checkerboard). Once calibrated and rectified, the equations derived earlier for the nominal case of two pinhole cameras can be applied. Reference [18] provides a good review of this standard process.

Lastly, it should be noted that one of the most difficult steps of stereo calculations is establishing image point correspondences between points in the reference image and points in the secondary image. Despite having the epipolar constraint to aid in the overall correspondence search, there are typically many pixels in a single row that can still match to the pixel in question. As such, a common solution is to consider the Sum-of-Squared Difference (SSD) or Sum-of-Absolute Difference (SAD) between the neighborhood of the pixel in the reference image and neighborhoods of all the candidate pixels of the same row of the secondary image. By introducing a contextual aspect to the correspondence search, the probability of accurate correspondences is increased. The underlying assumption with this approach, however, is that the context around corresponding pixels in one image and another are not that different. This highlights some of the differences between wide-baseline stereo and short-baseline stereo. With wide-baseline stereo, objects which are very close to the camera will have a very different context when considered in the reference and secondary images separately. However, objects rather far away will have much more similar neighborhoods and thus yield reliable 3D measurements. Conversely with short-baseline stereo, close objects will tend to have more contextual similarities as opposed to further objects which will have too much similarity between many candidate pixels.

Chapter 3

Object Detection with 3D Pose Estimation

This chapter considers the problem of real-time 3D-object detection with tracking. Much of the prior work done in this area has focused on off-line methods. Only in recent years have advances been made toward real-time applications. In the sections to follow, a tracking filter derived from Bayes' Filter will be presented that enables 6D pose estimation given accurate data association between measurements in the world and measurements from the known model(s).

3.1 Object Detection Context and Contribution

As mentioned in chapter 1, much prior work on object pose estimation has been developed in the context of 3D object recognition, where the goal is to first detect and recognize an object, then estimate its 6D pose ([23], [26], [43], [45]). While advances have been made toward real-time applications, the existing approaches have either sacrificed feature correspondence accuracy (and thus scale and rotation invariance) for speed ([54], [5]) or computational speed for feature correspondence accuracy ([36], [40]). The contributions in this chapter will show that both a high accuracy of feature correspondence and fast computational speeds can be achieved using a stereo camera with SIFT feature extraction. The details of this chapter will also illustrate how 3D object models can be created in relatively short order time (minutes) using a stereo. Lastly, a novel method for determining ground truth 6D pose estimates will be presented.

3.2 Framework

Consider a mobile agent equipped with a stereo-camera sensor head and some model representation of an object (e.g., some database of features known to correspond to the model) and tasked with detecting and tracking the object using the measurements provided only by the stereo camera in real-time. Understanding that the object itself is free to move about the environment workspace uncoupled from the mobile agent, this problem can be solved with a straightforward application of

Bayes' Filter. The following subsections will detail the application of Bayes' Filter to this problem in the most general sense, with implementation details provided as they become relevant. Experimental results supporting the real-time capabilities of the approach will also be presented.

3.2.1 Bayes' Filter: Dynamic Prediction

Let the state $\mathbf{X}_k \in \mathbb{R}^{6\times 1}$ be defined as the 3D pose of the object in the camera-reference-frame:[1]

$$\mathbf{X}_k = \left[\begin{array}{cc} \mathbf{x}_R^T & \mathbf{x}_\theta^T \end{array} \right]^T \in \mathbb{R}^6 \,, \tag{3.1}$$

where $\mathbf{x}_R = \left[\begin{array}{ccc} x & y & z \end{array} \right]$ describes the translational displacement of the object relative to the camera and $\mathbf{x}_\theta = \left[\begin{array}{ccc} \alpha & \beta & \gamma \end{array} \right]$ describes the orientation of the object relative to the camera (yaw about the z-axis of the camera, pitch about the y-axis, and roll about the *x-axis*, respectively) .

Let the object be free to move in any direction at any given moment. As such, the motion-model of the object can best be described by a 0^{th}-order random-walk model:[2]

$$\mathbf{X}_k = \mathbf{A}{\cdot}\mathbf{X}_{k-1} + \eta \,, \tag{3.2}$$

where $\mathbf{A} = \mathbf{I}^{6\times 6}$ the identity matrix, $\eta \in \mathbb{R}^{6\times 1}$ is Gaussian white process noise, with covariance $\mathbf{Q} \in \mathbb{R}^{6\times 6}$. However, understanding that the object itself is limited by a maximum velocity, $\mathbf{v}_{max} = [\ v_{x,max} \quad v_{y,max} \quad v_{z,max}\ ]^T$, and maximum angular velocity, $\omega_{max} = [\ \alpha_{max} \quad \beta_{max} \quad \gamma_{max}\ ]^T$, the variance associated with matrix $\mathbf{Q}$ is upper bounded by $\mathbf{v}_{max}\Delta t$ and $\omega_{max}\Delta t$, where $\Delta t = t_k - t_{k-1}$. In keeping with the conservative framework, let $\mathbf{Q}$ vary each timestep to be

$$\begin{aligned} \mathbf{Q}_k &= diag(\sigma^2_{v_{max}}, \sigma^2_{\omega_{max}})\,, \\ \sigma^2_{v_{max}} &= [\ (v_{x,max}\Delta t)^2 \quad (v_{y,max}\Delta t)^2 \quad (v_{z,max}\Delta t)^2\ ]\,, \\ \sigma^2_{\omega_{max}} &= [\ (\alpha_{max}\Delta t)^2 \quad (\beta_{max}\Delta t)^2 \quad (\gamma_{max}\Delta t)^2\ ]\,. \end{aligned}$$

The dynamic prediction step of Bayes Filter follows the form of section 2.1.1, with the predicted state and covariance, $(\overline{\mathbf{X}}_k, \overline{\mathbf{P}}_k)$, given by equations (2.5) and (2.6):

$$\overline{\mathbf{X}}_k = \mathbf{A} \cdot \mathbf{X}_{k-1}\,, \tag{3.3}$$

$$\overline{\mathbf{P}}_k = \mathbf{A} \cdot \mathbf{P}_{k-1} \cdot \mathbf{A}^T + \mathbf{Q}_k\,, \tag{3.4}$$

[1]The camera-reference-frame is a reference frame coincident with the frame of the pinhole camera model. Typically either the right or left camera is chosen as the reference. Throughout this thesis, the reference camera is always chosen to be the right camera.

[2]While other random walk models exist, the state of the system – which does not include velocity – limits the chosen model to be 0^{th}-order.

where $\mathbf{X}_{k-1}$ and $\mathbf{P}_{k-1}$ define the pose estimate and covariance of the previous timestep. With the dynamic prediction model defined, Bayes' Filter is made complete with appropriate specification of a measurement model for the measurement update step. Recall the measurement update step will incorporate the predicted state and covariance terms solved in equations (3.3) and (3.4) to yield a state and covariance estimate for the k^{th} timestep.

3.2.2 Bayes' Filter: Measurement Update

Let a current-feature, $\mathbf{y}_j$, be defined as the j-th feature[3] found in the current stereo image frame and let a database-feature, $\mathbf{b}_n$, be defined as the n-th feature belonging to the object model stored in the database. Consider $\mathbf{y}_j$ and $\mathbf{b}_n$ to have the following forms:

$$\mathbf{y}_j = \begin{bmatrix} y_x \\ y_y \\ y_z \\ \mathbf{d}_y \end{bmatrix}, \quad \mathbf{b}_n = \begin{bmatrix} b_x \\ b_y \\ b_z \\ \mathbf{d}_b \end{bmatrix}$$

where for $\mathbf{y}_j$, terms $\{y_x,\, y_y,\, y_z\}$ are the 3D coordinates of the j-th feature as defined in the camera-reference-frame, and $\mathbf{d}_y \in \mathbb{R}^U$ is a feature descriptor, U elements in length, used primarily to facilitate feature correspondence. Similarly for $\mathbf{b}_n$, terms $\{b_x, b_y, b_z\}$ are the 3D coordinates of the n-th feature as defined in the object-reference-frame, and $\mathbf{d}_b \in \mathbb{R}^U$ is a feature descriptor used to facilitate feature correspondences[4].

Let the set of all database-features be defined by:

$$\mathbf{B} = \{\mathbf{b}_n \,|\, \forall n \in [1, \ldots, N]\},$$

where N is the total number of features stored in the database defining the object. Let n_k be the number of currently detected features in the k-th camera image known to belong to the object, and $\mathbf{J} \in \mathbb{Z}^{n_k}$, a correspondence variable indicating which n_k current-features match to the database-features of $\mathbf{B}$. The notation here will be such that the j-th current feature, $\mathbf{y}_j$, corresponds to the $\mathbf{J}(j)$ database-feature of $\mathbf{B}$, denoted by $\mathbf{b}_{\mathbf{J}(j)}$. Note that the variable $\mathbf{J}$ is a measure of the best fit of the model's database features to the current ones. It is typically determined by comparing the Euclidean distance between the feature descriptors, $\mathbf{d}_y$ and $\mathbf{d}_b$, and establishing correspondences between features that yield low distances (in descriptor space) and are below some threshold. Features in the database are generated off-line during a *training phase* and then catalogued by camera viewpoint.

[3]"feature" in this sense is defined as a 3D interest point with a descriptor of a number of possible types: SIFT descriptor, shape context descriptor, optical flow descriptor, etc.

[4]In previous approaches to object recognition and detection, the features themselves omit the 3D coordinates since stereo camera data was not considered. Instead, the 2-D image coordinates of the feature were often used.

Details on how the training phase is implemented and how the correspondence variable $\mathbf{J}$ is acquired will be discussed later in sections 3.2.3 and 3.2.4.

Now let a data measurement be a set of 3D features, defined in the camera reference frame:

$$\mathbf{D}_k = [\ \mathbf{y}_1^T \quad \mathbf{y}_2^T \quad \cdots \quad \mathbf{y}_{n_k}^T\ ]^T ,$$

where each 3D point measurement, $\mathbf{y}_j$, corresponds to a model database feature as defined by the correspondence variable $\mathbf{J}$. The measurement model can then be defined by the following set of non-linear geometrical transforms:

$$\begin{aligned} \mathbf{D}_k &= \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{n_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(1)} \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(2)} \\ \vdots \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(n_K)} \end{bmatrix} + \xi , \\ &= \mathbf{H}(\mathbf{X}_k, \mathbf{B}) + \xi \end{aligned} \tag{3.5}$$

where $\xi \in \mathbb{R}^{(3\times n_k)\times 1}$ is Gaussian white measurement noise with covariance given by $\Sigma_m \in \mathbb{R}^{3n_k \times 3n_k}$, $\mathbf{x}_{R,k}$ is the translational pose of the object relative to the camera at the k^{th} timestep (as defined in equation (3.1)), and $\mathbf{R}_{CO_k} \in SE(3)$ is the rotation matrix defining the orientation of the object-frame relative to the camera-frame:

$$\mathbf{R}_{CO_k} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}_k .$$

Note that in the expression for equation (3.5), only the Cartesian coordinates are used in $\mathbf{y}_j$ and for $\mathbf{b}_{\mathbf{J}(i)}$ when multiplying by $\mathbf{R}_{CO_k}$. For brevity, the following abbreviations have been used: $c(\cdot) \triangleq cos(\cdot)$ and $s(\cdot) \triangleq sin(\cdot)$.

With the non-linear measurement model defined, the measurement update step of Bayes' Filter follows the form of section 2.1.2, assuming that a linear approximation to equation (3.5) is valid. The updated state and covariance terms, $(\mathbf{X}_k,\ \mathbf{P}_k)$, are then given by equations (2.14) and (2.15):

$$\begin{aligned} \mathbf{X}_k &= \overline{\mathbf{X}}_k + \mathbf{K}_k(\mathbf{D}_k - \mathbf{H}(\overline{\mathbf{X}}_k, \mathbf{B})) , \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C}_k)\overline{\mathbf{P}}_k , \end{aligned}$$

where the terms $\overline{\mathbf{X}}_k$ and $\overline{\mathbf{P}}_k$ come from the dynamic prediction step of Bayes' Filter as defined in equations (3.3) and (3.4), and the matrix $\mathbf{C}_k \in \mathbb{R}^{(3\times n_k)\times 6}$ is the linearized measurement matrix

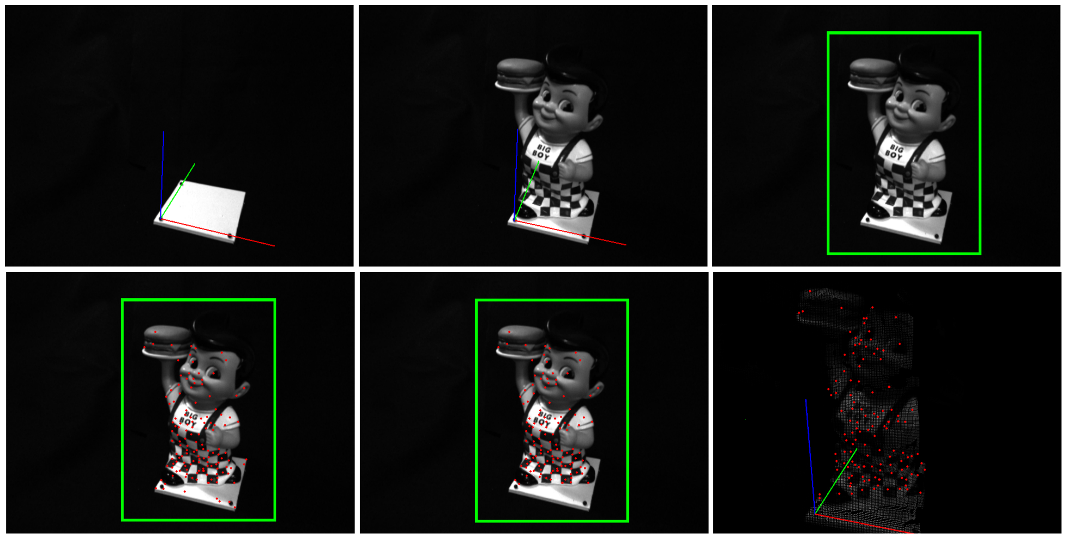


Figure 3.1: The database of features for the object is generated during a training phase. Beginning with the top row, far left, the training phase begins by establishing a set of coordinate axes for the object using a calibration platform. Once established, the object axes is aligned with the axes of the platform (top row, middle image). A bounding box is then defined by the user around the object in the image (top row, far right), where feature extraction is to be applied (bottom row, far left). Features within the bounding box though not part of the image are then removed as shown in the bottom row, middle image (note the features on the white platform have been removed). In the final step, full stereo is applied and nearest-neighbor data association is applied to assign each extracted feature a corresponding 3D point, as defined in the *object-reference-frame*.

evaluated at the predicted pose location:

$$\mathbf{C}_k = \left.\frac{\partial \mathbf{H}}{\partial \mathbf{X}_k}\right|_{\overline{\mathbf{X}}_k} \quad .$$

The full analytical expression for $\mathbf{C}_k$ can be found in Appendix A. The matrix $\mathbf{K}_k$ is the dynamic Kalman gain as defined in equation (2.9) and reproduced here for reference:

$$\mathbf{K}_k = \overline{\mathbf{P}}_k \mathbf{C}_k^T (\mathbf{C}_k \cdot \overline{\mathbf{P}}_k \cdot \mathbf{C}_k^T + \Sigma_m)^{-1} \quad .$$

With the dynamic prediction step and measurement update step of Bayes' Filter defined, the recursive algorithm can be implemented in a sequential fashion as discussed in section 2.1.2. However, it is important to note that the measurement update is contingent upon accurate specification of the feature correspondence variable, $\mathbf{J}$, which is discussed in section 3.2.4.

3.2.3 Training Phase

A key component of the presented framework thus far presumes that a database of 3D descriptive features, defined in the object reference frame, are available available to compare against current features. Kushal and Ponce [23] and Rothganger et al. [45] capture a series of viewpoints around the object and use the viewpoints to reconstruct a 3D object model. The approach considered here – referred to as the *training-phase* – follows a similar vein:

1. Using a rigidly fixed stereo camera head, the coordinate axes associated with an object reference frame are first established. This is done by using a series of calibration points detected in stereo and placed against a uniform background (Fig. 3.1, top row, left).
2. Once the object reference frame is established, the object to be identified is placed in front of the stereo camera, aligned with the coordinate axes just generated (Fig. 3.1, top row, middle).
3. When the object is stabilized and properly aligned with the anchored coordinate axes, the live image from the reference camera is displayed and a user defined bounding box is drawn around the object (Fig. 3.1, top row, right).
4. Feature extraction is then applied to the region of interest defined by the bounding box (Fig. 3.1, bottom row, left).
5. Once the set of features in the region of interest are generated, features not necessarily belonging to the object (though still within the bounding box) are removed manually in a touch-up step (Fig. 3.1, bottom row, middle).
6. Once extraneous extracted features are removed, the remaining features are assigned nearest neighbor 3D stereo data points, where "nearest-neighbor" is determined via Euclidean distance in pixel coordinates between the feature location and the projected stereo location in the image plane. However, before tagging features with their corresponding 3D stereo location, the 3D stereo data points are transformed into the object reference frame using the coordinate axes defined in step 1 (Fig. 3.1, bottom row, right).
7. Steps 1–6 are then repeated for each additional viewpoint.

Following the training phase, the set of database features is stored to a series of data files indexed by viewpoint and object ID. Thus the above steps need only be executed once for each object since any subsequent queries for detection and tracking for that object can be done by loading the appropriate set of data files.

3.2.4 Feature Matching

The purpose of extracting features in the training-phase is to establish a baseline set of features to be used in feature-matching in the measurement update step. The sole aim of feature matching is to determine the correspondence variable $\mathbf{J} \in \mathbb{Z}^{n_k}$, of which many different techniques exist. At the core of all feature-matching techniques however is the uniqueness associated with the descriptor itself. The more descriptive a feature is, the greater the tendency to reliably match that feature in an image from another viewpoint.

While many different descriptors have been proposed over the years (e.g., SIFT features by [30], PCA-SIFT features by Ke and Sukthankar in [21], steerable filters by Freeman and Adelson in [16], differential invariants by Schmid and Mohr in [49], and shape context descriptors by Belongie et al. in [3]), a recent review by Moreels and Perona in [35] considered the performance of a set of the most popular detectors and descriptors for use with images of 3D objects and published their recommendations.[5] Based on their findings – which considered viewpoint and lighting changes, frequency of false alarms and mismatches, and computational cost – the best overall choice reduced to features with SIFT descriptors but detected using an affine-rectified detector (as defined by Mikolajczyk and Schmid in [33]) instead of the Difference of Gaussians detector proposed by Lowe in [30]. However, the authors acknowledge that for increased computational speed, the Difference of Gaussians detector can be used since its performance is comparable to the affine-rectified detector. As such, the approach presented here considers SIFT features detected using the Difference of Gaussians detector.

With regards to finding a method of searching through potentially thousands of features to find the correspondences, exhaustive search is guaranteed to be optimal. However, its computational expense has promoted other approaches. A popular method – and the one adopted in this framework – is the Best-Bin-First search method first presented by Beis and Lowe in [2]. In their search method, the authors define an approach that uses a k-d tree to bin up the feature space so that features are searched in order of distance from the query location. Note that their approach is not an exhaustive search since the tree search is cut off after a certain number of bins have been explored. Additionally, valid matches are only considered if the ratio of the current best match to the second best match is less than a threshold (in their case, 0.80). While their approach does not necessarily always return the true correspondence, it will at least return the true feature correspondence with high probability.

Given that inaccurate feature correspondences can lead to erroneous pose estimates, a remedy commonly implemented by others is to add an additional coherency check to validate correspondences ([54], [40], [5]). In the approach presented, a geometric model check is similarly employed. Supposing an object is already being tracked, then for each correspondence found from SIFT feature matching, the matched 3D database feature is transformed into the camera-reference frame and the Mahalanobis distance between the transformed feature and its matched current feature (in Euclidean space) is checked against a threshold. If the distance exceeds the threshold, the correspondence is rejected. If an object is not yet initialized, the geometric constraint is not applied, since the object

[5]It is important to note the authors' distinction between a "detector" and a "descriptor". A detector is any interest point operator that extracts an identifiable feature, such as a corner or an edge. A descriptor is a signature associated with the extracted detector that aids in establishing feature correspondence.

can really be positioned and oriented in any which way relative to the camera assuming no prior knowledge given. In this manner, the object pose is allowed to jump large distances (in Euclidean 6-DOF space) until initialized, at which point large jumps in pose are then attributed to feature mismatches.

3.2.5 Real-Time SIFT

As noted by Choi et al. in [5] and Panin and Knoll in [40], SIFT feature extraction is a rather expensive operation and a bottleneck to real-time pose tracking capabilities. As such, authors that utilize SIFT feature extraction often limit its utility to pose initialization at the beginning of the algorithm or pose re-initialization after a lost track. Once a track is established, SIFT is essentially turned off and sustained tracking is realized by faster, computationally less-expensive algorithms (i.e., POSIT, Lucas-Kanade Tracker, *Contracting Curve Density* (CCD) algorithm, etc.). To get a perspective on the delays associated with SIFT feature extraction (without sparse stereo), a typical 320×240 image will take on average 200 ms to compute 70+ features in an image on a 2.40 GHz processor (and $>$200 ms for high context images that can contain several hundred SIFT features); that means at most, a 5 Hz performance rate can be achieved assuming the remaining computations associated with sparse stereo, feature matching, and the steps of Bayes' Filter are negligible in computational costs.

In the approach presented here, SIFT feature extraction is used for track initialization *and* also for sustained tracking. Real-time calculations with SIFT are achieved by noting that much of the expensive operations involved with feature extraction are spent on areas of the image not containing the object. Assuming the object pose is known, a region of interest (ROI) can be specified that bounds a subset of the image known to contain the object. By specifying a ROI in the image, SIFT feature extraction can be reduced to operating in a window size typically 10%-40% of the original image, thus making real-time pose tracking with SIFT possible. Of course the ROI cannot be specified if the object pose is it not known and SIFT must be applied to the entire image – reducing the proposed approach to an initialization step no different than the approaches of Choi et al. [5] and Panin and Knoll [40].

Assuming an object track has been initialized, the ROI is determined as follows: let $\mathbf{o} \in \mathbb{R}^{2\times 1}$ be the projected object origin onto the image plane of the reference camera, $B = [x\ y\ w\ h]$ represent a bounding box defining the ROI of the image (x and y the center of the ROI, and w and h the width and height), $d_{ref} \in \mathbb{R}$ be the reference distance defined as the nominal Euclidean distance between the camera and the object reference frames during training – as described in section 3.2.3, and $d_{curr} \in \mathbb{R}$ be the current distance between the camera origin and the object origin (Figure 3.2 illustrates the defined variables). The ROI as defined by B requires the parameters x, y, w, and h

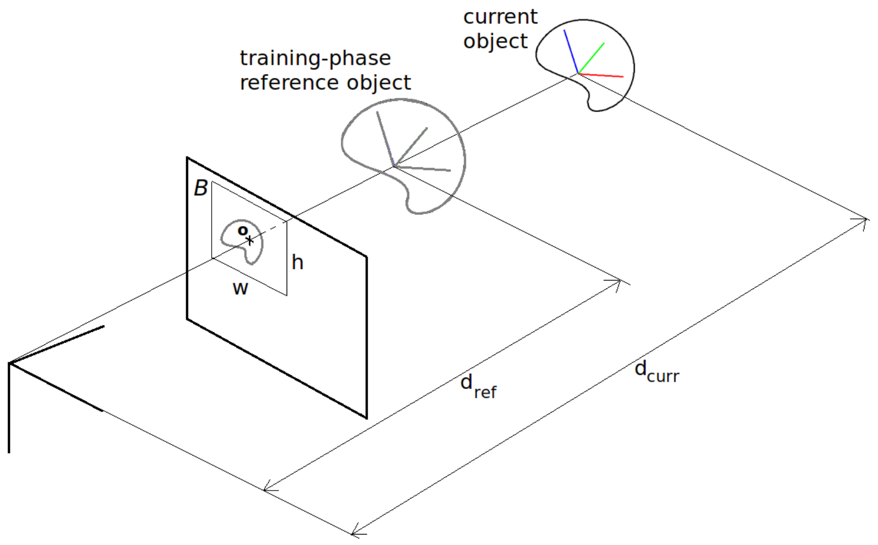


Figure 3.2: The ROI of the current image is centered on the object origin (projected into the image plane), with width and height computed based on the scaled ratio of the reference object distance to the current object distance.

to be determined. x and y will come from the projection of the object origin onto the image plane:

$$x = \frac{f \cdot X}{Z} + c_x \,,$$
$$y = \frac{f \cdot Y}{Z} + c_y \,,$$

where $(X,\ Y,\ Z)$ represent the current 3D location of the object origin in the camera-reference frame, f the focal length of the camera, and $(c_x,\ c_y)$ the image center of the reference image. As for the width w and height h, these parameters of the ROI are defined as follows:

$$w = k_x \cdot \sigma_s \,,$$
$$h = k_y \cdot \sigma_s \,,$$

where $k_x$and k_y are constants (dependent on the size of the object) and σ_s a scaling factor defined as

$$\sigma_s = \frac{d_{ref}}{d_{curr}} \quad .$$

Having the ROI defined as above allows the ROI to change in size as the object-to-camera transform changes with time. The resulting effect is a tracking window that keeps the SIFT calculations localized on the object and real-time tracking with modest computation thus feasible.

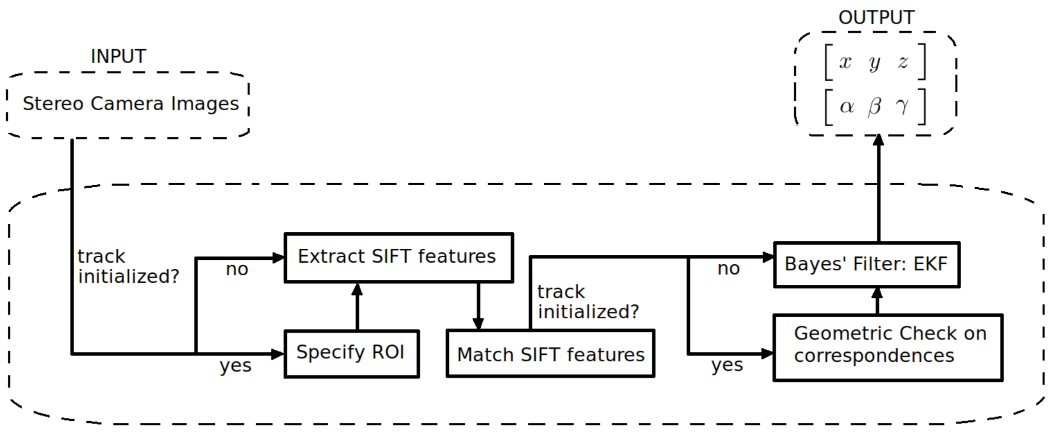


Figure 3.3: The overall approach for the presented algorithm is summarized in the above figure which describes the series of steps taken from input to output for one iteration.

3.3 Approach Summary

To summarize the overall approach, Fig. 3.3 outlines the basic steps for one iteration of the algorithm. It begins with the input, which is the pair of stereo camera images. SIFT feature extraction is then applied to the reference image of the stereo pair using the Region of Interest (ROI) if the object track has already been initialized. This step also incorporates 3D coordinates being assigned to each extracted feature using sparse stereo calculations. Following SIFT feature extraction, feature matching is then applied using the Best-Bin-First algorithm of Beis and Lowe [2]. Geometric constraint checks are then applied depending on whether the object track has been initialized. The purpose of this step is to remove geometrically infeasible correspondences that may have resulted from the matching step. The final step of the algorithm is to apply the recursive Bayes' Filter equations (EKF), which yields a pose estimate of all 6 Euler parameters for the k^{th} timestep.

3.4 Experimental Results

Based on the analysis presented thus far, two types of experiments were considered to validate the real-time capabilities of the 3D pose tracking algorithm. An object – in this case a Bob's Big Boy model bank – was trained using the training phase discussed in section 3.2.3 and used as the test object in both experiments. Eight viewpoints were considered during training, with each viewpoint positioned at 45° intervals about the object spanning a full 360° along the equidistant ring – yielding a total of 1482 features stored in the database. For a stereo camera, a PointGrey Research BumbleBee2 stereo color camera – with a native resolution of 640×480 downsampled to 320×240 – was used for both experiments. The computing platform used was a laptop running Linux with an Intel Core2 Duo CPU 2.40 GHz processor and the algorithm written in C/C++ for

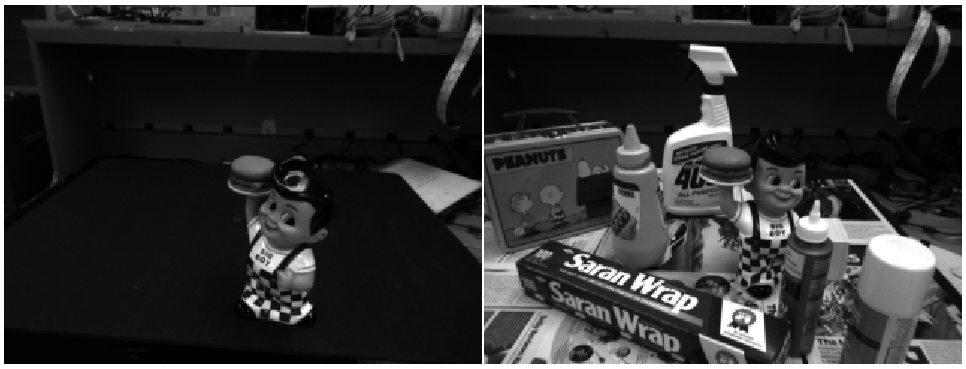


Figure 3.4: The image on the left shows the setup for the first experiment considered (uncluttered environment). The image on the right shows the setup for the second experiment considered (cluttered environment with noise).

optimized speed. The open source Intel library, OpenCV,[6] and the 3D rendering library, OpenGL,[7] were also used to develop the system. Fig. 3.8 shows a screenshot of the visual interface used in the experiments.

In the first experiment, the trained object was positioned in an uncluttered environment (uniform background with no other objects) and placed under continuous rotations and translations in all degrees of freedom with the camera stationary and measurements made in the camera reference frame. The purpose of this experiment was to test the baseline capabilities of the presented approach in tracking all six Euler parameters. In the second experiment, the trained object was placed in a cluttered environment (noisy background with other similarly sized objects) and again placed under continuous rotations and translations with the camera stationary. The purpose of this experiment was to test the capabilities of the approach to robustly track 3D pose estimates against non-uniform noisy backgrounds and partial occlusions. Fig. 3.4 shows the setup for both experiments.

In both experiments, a measure of ground truth was essential in comparing the accuracy of the estimated results. The next section describes an approach to acquire ground truth estimates of all six Euler parameters. In comparison to prior work, this method does not require the purchase of third party software as in [54], simulated experimental results as in [40], or distinguishing markers for computer vision toolkits as in [5] – which in itself is just another form of a position estimate with noisy data.

3.4.1 Ground Truth

Ground truth for both experiments was determined by using a controllable stepper motor – in this case, a Directed Perception^{TM,8} PTU-D46-17 pan-tilt-unit with an accuracy of 0.0514° in both pan and tilt positions (though tilt positioning was not used). The base of the motor (reference frame B)

[6]http://opencv.willowgarage.com/wiki/
[7]http://www.opengl.org
[8]www.directedperception.com

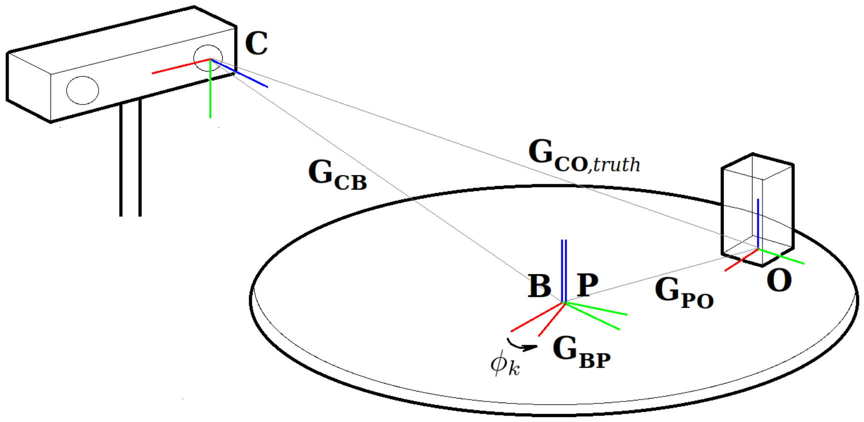


Figure 3.5: The various frame transformations between camera (C), platform (P), and object (O) can be measured to establish a means of object ground truth, as measured in the camera reference frame.

was rigidly mounted to a table and a platform was fixed to the pan axis of the motor (reference frame P), rigid enough to support the weight of the object being detected. The object (reference frame O) was then placed on the platform, a fixed distance from the pan-axis, though with the object's vertical axis aligned in the same direction. The stereo camera (reference frame C) was then placed some fixed distance away from the motor base-frame, pitched looking slightly downwards, and kept stationary (see Fig. 3.5 for a clear depiction of all various reference frames). Commanded rotations resulted in rotations and translations of the object as seen in the camera reference frame.

With this particular setup, all 6 Euler parameters would be known to high accuracy since the transformations between all frames could be easily measured. Because the camera was kept stationary in both experiments, the transformation between the base of the motor frame and the camera frame was a constant, measurable value:

$$G_{CB} = \begin{bmatrix} \mathbf{R}_{CB} & \mathbf{d}_{CB} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad .$$

For each pan rotation commanded to the stepper motor, the transformation between the platform frame and the base frame could be determined by the motor angle read out (ϕ_k):

$$G_{BP} = \begin{bmatrix} \mathbf{R}_{BP}(\phi_k) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad .$$

Since the object frame was displaced some slight amount from the pan-axis (though with the vertical axes aligned) and fixed relative to the platform frame, the transformation between the object frame and the platform frame was a constant value that could be determined be measuring the translation

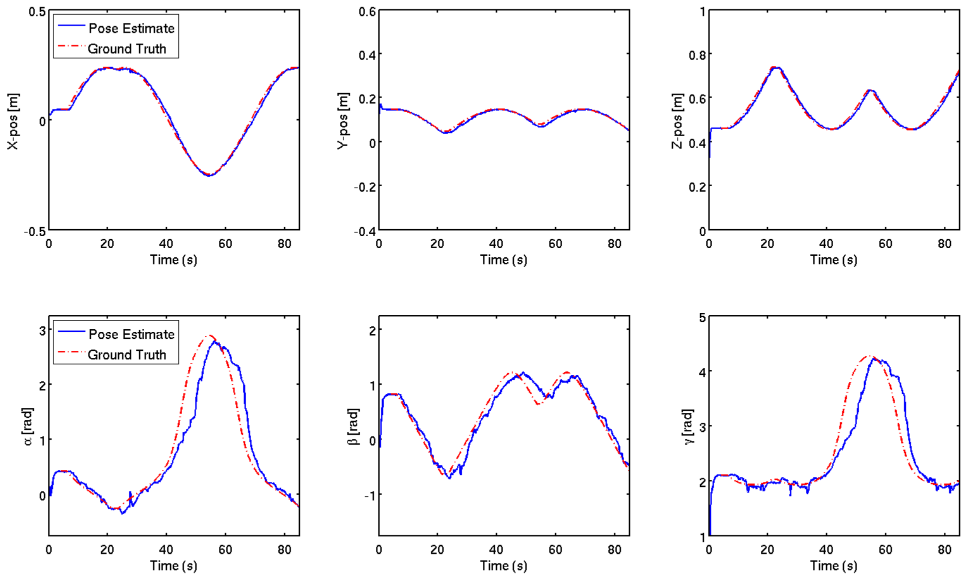


Figure 3.6: Shown above are the 6 Euler parameter estimates using the algorithm described for the first trial experiment where the trained object was placed in an uncluttered, uniform environment. The top row shows the tracking results in (x,y,z) and the bottom row shows the results in (α,β,γ). Shown in the red dotted line is the ground truth measurements of the Euler parameters, and in blue the EKF-estimated Euler parameters.

(needed only in x and y) and the rotation (needed only about the z-axis) between frames:

$$G_{PO} = \begin{bmatrix} \mathbf{R}_{PO} & \mathbf{d}_{PO} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad .$$

Finally, by measuring the transformations G_{CB}, G_{BP}, and G_{PO}, the ground truth transformation between the object and camera can easily be determined as

$$G_{CO,truth} = G_{CB} \cdot G_{BP} \cdot G_{PO} \, .$$

3.4.2 Uncluttered Environment

As mentioned earlier, in this experiment the trained object was positioned in an uncluttered environment having a uniform background with no other distractors present. The trained object was placed under continuous rotations and translations in a manner as described in section 3.4.1. The platform itself oscillated between $+90°$ and $-90°$ at a slew rate of $6°$/s. The purpose of this experiment was to test the baseline capabilities of the presented approach in tracking all six Euler parameters.

Shown in Fig. 3.6 are the results for this experiment. As can be seen in the figure, in this most basic scenario the algorithm does an extremely good job of tracking the object in (x,y,z). With regards to orientation, the estimated values of (α,β,γ) are slightly noisier and subject to minor delays at some points. This effect is most likely due to certain features coming into view as the

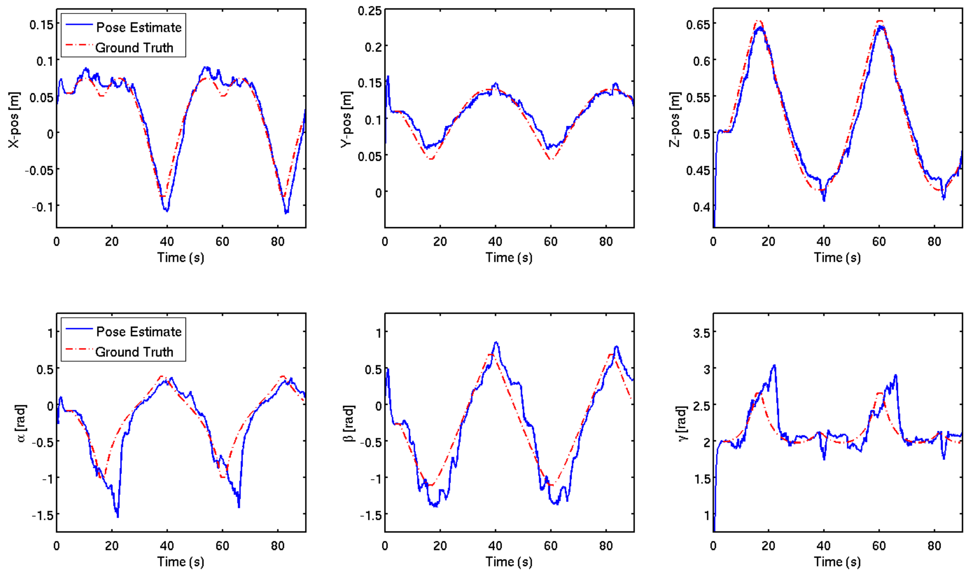


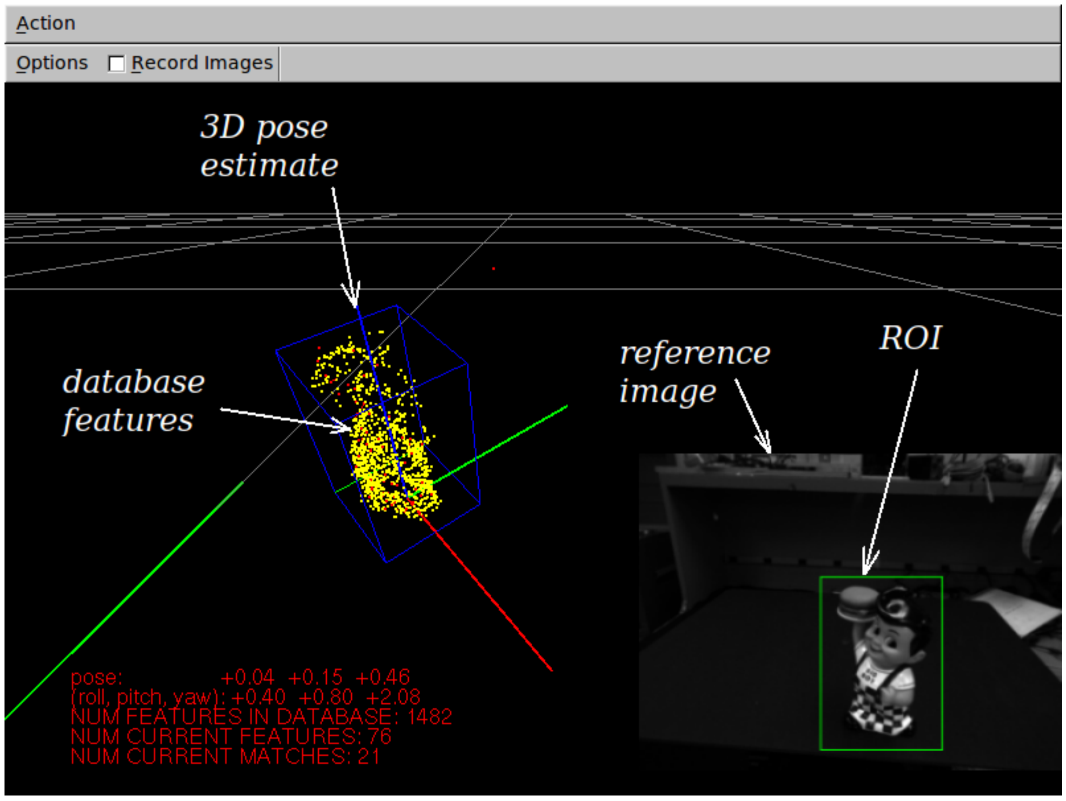
Figure 3.7: Shown above are the 6 Euler parameter estimates using the algorithm described for the second trial experiment where the trained object was placed in a cluttered, noisy environment. The top row shows the tracking results in (x,y,z) and the bottom row shows the results in (α,β,γ). Shown in the red dotted line is the ground truth measurements of the Euler parameters, and in blue the EKF-estimated Euler parameters.

platform rotates, which when matched, causes the EKF to improve the overall estimate and reduce the state covariance.

With regards to real-time tracking capabilities, the algorithm ran at an average rate of 12.12 Hz, with a maximum rate reaching 21.40 Hz. Because of the manner in which the ROI is computed, the algorithm actually performs faster as the object moves away from the camera (smaller ROI) reaching a nominal 20 Hz frequency rate, and slows down to about 8 Hz when the object comes closer (larger ROI).

3.4.3 Cluttered Environment

In this second experiment, the trained object was placed in a cluttered environment (noisy background with other similarly sized objects) and again placed under continuous rotations and translations with the camera stationary. As stated earlier, the purpose of this experiment was to test the capabilities of the presented approach to robustly track 3D pose estimates against non-uniform noisy backgrounds and partial occlusions. Using the same trained object, the platform was rotated this time between $+60°$ and $-60°$ at a slew rate of $6°/s$ to generate the necessary translations and rotations of the object. To add noise, the platform itself was covered with magazine pages of various text and pictures. To generate occlusions, various objects of similar size to the training object were placed around the object (see right image of Fig. 3.4). As the platform moved, certain objects would occlude parts of the training object and thus test the ability of the algorithm to track during partial



```
pose:              +0.04  +0.15  +0.46
(roll, pitch, yaw): +0.40  +0.80  +2.08
NUM FEATURES IN DATABASE: 1482
NUM CURRENT FEATURES: 76
NUM CURRENT MATCHES: 21
```

Figure 3.8: A screenshot of the algorithm running in real-time. For clarity, aspects of the developed visual interface have been annotated to illustrate the details of the algorithm working.

occlusions.

Shown in Fig. 3.7 are the results for this experiment. As can be seen in the figure, the tracking results in (x,y,z) and in (α,β,γ) are definitely noisier than in the first trial experiment, which is expected given the noisier image scene and partial occlusions. There is also a similar delay in the orientation estimates which is prominent at the peaks of the oscillating platform. Similar to the first experiment, this effect is most likely due to certain features coming into view as the platform rotates, which when matched, causes the EKF to improve the overall estimate and reduce the state covariance. It is more prominent in this experiment mainly due to the occlusions caused by the other objects surrounding the object being detected.

With regards to real-time tracking capabilities, the algorithm ran at an average rate of 9.50 Hz, with a maximum rate reaching 13.52 Hz. The slower rates are expected given the spurious features on the platform floor which add computation cycles to the SIFT extraction step of the overall algorithm. Nonetheless, the results of this experiment show robustness to noise and partial occlusions, while illustrating real-time detection capabilities of the presented algorithm.

3.5 Conclusions

Based on the experiments of sections 3.4.2 and 3.4.3, the presented algorithm is capable of real-time 3D pose tracking of all 6 Euler parameters. The novel specification of the ROI in the image enables SIFT calculations at each timestep, an improvement on works of previous authors who have reserved SIFT calculations only for emergencies (i.e., pose re-initialization after lost track) because of the computational expenses involved. The detailed approach presented for obtaining ground truth of all 6 Euler parameters presents an additional novel contribution that allows for more accurate standards by which to obtain ground truth estimates without the need to purchase third party software or intricate simulations. In the chapter to follow, extensions of the presented approach will be applied to a more complicated problem, that of sensor planning for pose estimation and model identification.

Chapter 4

Sensor Planning for Model Identification

Building on the framework for pose estimation presented in chapter 3, the additional problem of sensor planning for object/model identification is considered. This chapter addresses the combined problem of sensor planning for object identification *with* 3D pose estimation by contributing a novel algorithm derived from basic Bayesian principles. Optimal control actions for sensor placement are achieved via an information gain metric based on current and future measurements conditioned on estimated 3D object poses for all possible models. The following sections present the proposed algorithm in full detail. Experimental validation of the algorithm is presented in chapter 5, which will illustrate the success of the approach and the novelty of the contribution. This chapter will focus on the derivation and details of the algorithm.

4.1 Sensor Planning Context and Contribution

As mentioned in chapter 1, the problem of sensor planning for object identification has been considered in previous works with information theoretic approaches: i.e., where should the robot move next to get the most relevant information to identify some unknown object? To help illustrate the problem being investigated, Figure 4.1 shows a hypothetical situation in which a robot is attempting to identify a person from a back view. As shown, the object is indistinguishable from a finite set of similar models, requiring the robot to contemplate the next viewpoint to move to that best acquires the most distinguishing information. The contributions of this chapter will be a Bayesian analysis of the sensor planning problem that incorporates as a necessary step the estimation of the pose of each potential model object. The resultant Sensor-Planning-for-Pose-Estimation-and-Model-Identification (SPPEMI) algorithm makes no assumptions on the workspace of the viewing agent or the object itself, allowing for identification of mobile objects from potentially mobile platforms – a major extension of previous works that have until now only considered stationary objects in controlled environments.

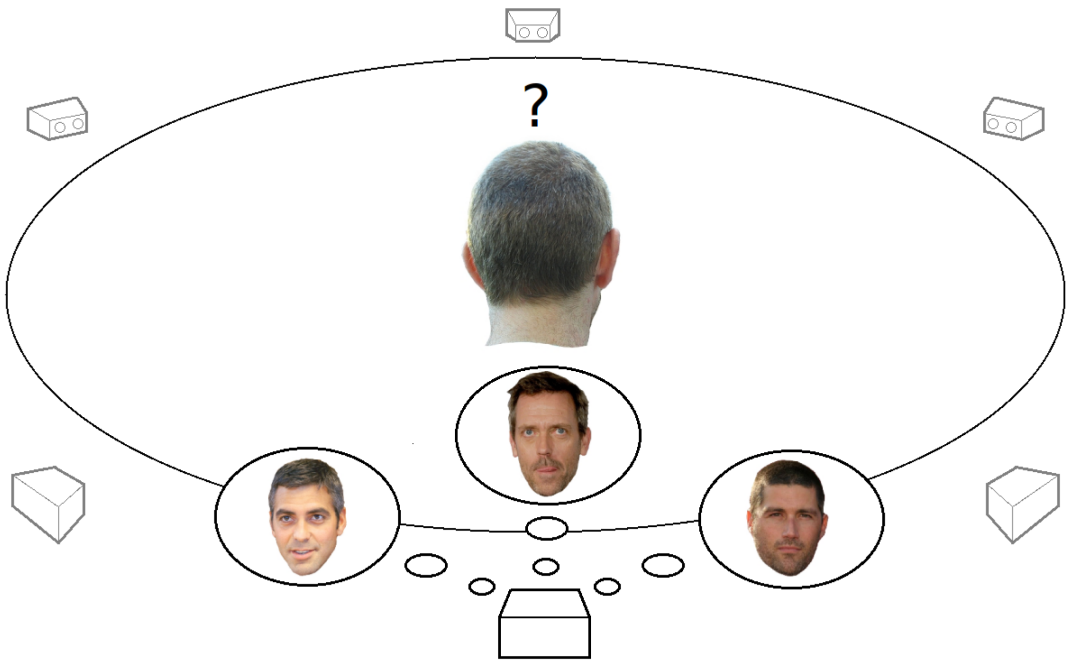


Figure 4.1: An illustration motivating the problem addressed in this chapter. A robot is tasked with identifying an unknown object (in this case a person) from a set of known objects. If the object identity is indistinguishable from a finite set of similar models, the robot must then decide where to move next to best acquire the most relevant information that helps resolve the identity of the unknown object.

4.2 Bayesian Sequential Analysis and Formulation

Consider the case of an autonomous robot which is given a database of known models for objects that may exist in its environment. Now suppose an unidentified object, though belonging to the database of known objects, is presented to the robot which is now tasked with identifying the correct object model and estimating its 3D pose. The question then arises: if the robot is unable to identify the correct model in the initial presented view, how should it optimally move to identify the object?

4.2.1 Approach

Let $\mathbf{D}_k$ denote visual (and possibly other) data obtained at t_k (e.g., this data might consist of SIFT features [30], Harris corners, 3D range points, etc.) and let $\mathbf{u}_k$ be the control input at t_k.[1] Let $\mathbf{X}_{k,i}$ be the system state vector at t_k for the i^{th} model, M_i (this state could be the state of the object relative to the camera, or the state of the camera relative to the object, etc.). For a given model M_i, state $\mathbf{X}_{k,i}$, and measurement data $\mathbf{D}_k$, suppose its evolution in time is governed by the following

[1]The notation $1 : k$ will be used to indicate the set of measurements, inputs, etc. from t_1 up to and including t_k.

discrete-time system equation and measurement model:

$$\mathbf{X}_{k,i} = \mathbf{A}(\mathbf{X}_{k-1,i}) + \mathbf{B}(\mathbf{u}_{k-1}) + \eta\,, \tag{4.1}$$

$$\mathbf{D}_k = \mathbf{C}(\mathbf{X}_{k,i}) + \xi\,, \tag{4.2}$$

where η, ξ are white Gaussian noise, and allow for $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ to be linear or non-linear functions.

In determining the optimal control action to execute for the next-best-view, $\mathbf{u}^*$, a useful utility metric often considered is the information gain function (shown by [39], [11], and others to yield promising results):

$$I_{\mathbf{u}_k} = H(M|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) - E_{\mathbf{D}_{k+1}}\left[H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})\right]\,. \tag{4.3}$$

The term, $H(M|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1})$, is the conditional entropy of the model identity, conditioned on the data up until t_k and the control actions up until t_{k-1}. The term, $E_{\mathbf{D}_{k+1}}\left[H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})\right]$, is the expected conditional entropy, with the expectation taken over the future data at the next timestep, i.e., $\mathbf{D}_{k+1}$, after hypothetical control $\mathbf{u}_k$ is applied. Note that with this definition of the expected conditional entropy, a myopic approach to sensor planning is assumed (i.e., a one step lookahead).

Entropy is understood to represent the uncertainty associated with a random variable, which in this case is the model identity, M. Thus the larger the entropy, the greater the uncertainty associated with the model being identified. It make sense, then, that the optimal control action is the one which maximizes the expected information gain:

$$\mathbf{u}^* = \underset{\mathbf{u}_k}{argmax}\ I_{\mathbf{u}_k} \quad , \tag{4.4}$$

since information gain occurs when $H(M|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) > E_{\mathbf{D}_{k+1}}\left[H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})\right]$, or when the next applied control results in an expected reduction in the uncertainty of the model identity.

Considering the information gain as defined above, the remainder of this section expands the expression for $I_{\mathbf{u}_k}$. It will be seen that the state of the system (the object 6D pose) becomes a necessary estimation step in determining the optimal sensing action.

The core entropy terms can be expanded as follows:

$$H(M|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) = -\sum_{i=0} P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) \log P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1})\,, \tag{4.5}$$

$$H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k}) = -\sum_{i=0} P(M_i|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k}) \log P(M_i|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})\,, \tag{4.6}$$

$$E_{\mathbf{D}_{k+1}}\left[H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})\right] = \int H^+ \cdot p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}) d\mathbf{D}_{k+1}\,, \tag{4.7}$$

where $H^+ \triangleq H(M|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})$. Calculation of equations (4.5), (4.6), and (4.7) require the terms:

$P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1})$, $P(M_i|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k})$, and $p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k})$. These terms can be developed into computable expressions with appropriate applications of Bayes Rule:

$$P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}) = \frac{\overbrace{p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)}^{I} \cdot \overbrace{P(M_i|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1})}^{II}}{\sum\limits_{j=0} p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_j)P(M_j|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1})}$$

$$P(M_i|\mathbf{D}_{1:k+1}, \mathbf{u}_{1:k}) = \frac{\overbrace{p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}, M_i)}^{III} \cdot \overbrace{P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k})}^{IV}}{\sum\limits_{j=0} p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}, M_j)P(M_j|\mathbf{D}_{1:k}, \mathbf{u}_{1:k})}$$

$$p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}) = \sum_{i=0} p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}, M_i)P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k})$$

where I, II, III, and IV are the remaining terms needing further specification. Close inspection of these terms shows that III and IV have an identical form to I and II, with a simple index shift; thus expressions for I and II will suffice to define III and IV.

4.2.2 Bayes Filter and Model Probabilities

Note that by marginalizing over the state of the i^{th} model, $\mathbf{X}_{k,i}$, term I can be rewritten as

$$p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) = \int p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{X}_{k,i}, M_i)p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)d\mathbf{X}_{k,i}\,,$$

where $p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{X}_{k,i}, \mathbf{u}_{1:k-1}, M_i)$ is derived from the measurement model (equation (4.2)). The second term of the integral results from the prediction step of a Bayes' Filter (as discussed in section 2.1). Recall that Bayes' Filter cycles recursively between the following two steps:

$$\begin{aligned} p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) &= \int p(\mathbf{X}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) \cdot \\ &\qquad p(\mathbf{X}_{k-1,i}|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)d\mathbf{X}_{k-1,i} \\ p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, M_i) &= \frac{p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{X}_{k,i}, \mathbf{u}_{1:k-1}, M_i)p(\mathbf{X}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)}{\int p(\mathbf{D}_k|\hat{\mathbf{X}}_{k,i}, \mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)p(\hat{\mathbf{X}}_{k,i}|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)d\hat{\mathbf{X}}_{k,i}}\,. \end{aligned}$$

The terms involved in the Bayes Filter can be solved for from the system state and measurement equations, equations (4.1) and (4.2), provided the following assumption is true:

$$p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k}, M_i) = p(\mathbf{X}_k|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, M_i)\,. \tag{4.8}$$

equation (4.8) presumes that the current state, $\mathbf{X}_{k,i}$, is only dependent on data up until the current timestep and the control actions up until the *previous* timestep. This assumption is valid since the current control action, $\mathbf{u}_k$, will not have a direct effect on the *current* state and can thus safely be omitted.

With regards to term II, a similar argument can also be made as was done in equation (4.8) by stating the following:

$$P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k}) = P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k-1})\,, \tag{4.9}$$

where the conditional dependence on $\mathbf{u}_k$ has been omitted. Inspection of equation (4.9) thus shows that $P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k-1})$ can be solved for recursively, provided that at the first timestep, the model probabilities are set to a priori values, $P_{o,i}$, *i.e.*,

$$P(M_i|\mathbf{D}_0,\mathbf{u}_0) = P_{o,i}\,.$$

4.2.3 Monte Carlo Sampling

Though terms III and IV can be solved for in a like manner as I and II, they require further analysis. Note that the expected entropy of equation (4.7) is taken with respect to the probability of future data, $\mathbf{D}_{k+1}$, conditioned only on past data measurements and control inputs. By marginalizing over the models with some simplification, the expected model entropy can be rewritten as follows:

$$\begin{aligned}
E_{\mathbf{D}_{k+1}}[H^+] &= \int H^+ \cdot p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k},\mathbf{u}_{1:k})d\mathbf{D}_{k+1} \\
&= \int H^+ \cdot \sum_{i=1} p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k},\mathbf{u}_{1:k},M_i)P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k})d\mathbf{D}_{k+1} \\
&= \sum_{i=0} P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k}) \int H^+ \cdot p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k},\mathbf{u}_{1:k},M_i)d\mathbf{D}_{k+1}\,.
\end{aligned} \tag{4.10}$$

The remaining integral is analytically intractable since the space of future measurements can extend into higher and higher dimensions. It can be approximated, however, using Monte Carlo sampling. Consider the following definitions:

$$\begin{aligned}
g(\mathbf{D}_{k+1}) &= H^+\,, \\
f(\mathbf{D}_{k+1}) &= p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k},\mathbf{u}_{1:k},M_i)\,,
\end{aligned}$$

where recall that $H^+ \triangleq H(M|\mathbf{D}_{1:k+1},\mathbf{u}_{1:k})$. Then equation (4.10) reduces to

$$\begin{aligned}
E_{\mathbf{D}_{k+1}}[H^+] &= \sum_{i=0} P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k}) \int g(\mathbf{D}_{k+1})\cdot f(\mathbf{D}_{k+1})d\mathbf{D}_{k+1} \\
&\approx \sum_{i=0} \left(P(M_i|\mathbf{D}_{1:k},\mathbf{u}_{1:k})\,\frac{1}{N}\sum_{n=0}^{N} g(\tilde{\mathbf{D}}) \right)\,,
\end{aligned}$$

where $\tilde{\mathbf{D}} \sim f(\mathbf{D}_{k+1})$. Thus, the expected model entropy, $E_{\mathbf{D}_{k+1}}[H^+]$, is calculated as a weighted sum of all model entropies with the weight chosen as the model probability. The model entropies are found by simulating N future measurements at timestep $k+1$ for which N model entropies are then calculated and averaged to yield a measure of the expected information associated with the control action, $\mathbf{u}_k$. Depending on the type of data measurements expected, simulation of future measurements may be as straightforward as application of equation (4.2). However, if expected measurement data is more complicated (e.g., SIFT measurements have 128-dimensional feature descriptors that are innately coupled with a correspondence vector that cannot be so easily simulated), other methods – such as lookup tables – can be employed. Chapter 5 provides a detailed discussion of how SIFT features can be sampled through the use of lookup tables, which is a novel contribution to the implementation framework of the SPPEMI algorithm.

4.3 SPPEMI Algorithm

Algorithm 4.1 SPPEMI

```
1: collect data $\mathbf{D}_k$ from sensor
2: $I_{max} = 0$
3: $\mathbf{u}^* = NULL$
4: $\overline{H}_k = calcCurrentEntropy(\mathbf{X}_{k-1}, \mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, M)$
5: $\mathbf{u}^+ = calcPossibleControlActions(\mathbf{X}_{k-1,i}, \mathbf{u}_{1:k-1})$
6: for all $\mathbf{u}_m^+$ do
7:    $H_m^+ = calcExpectedEntropy(\mathbf{X}_{k-1,i}, \mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M)$
8:    $I_{gain} = \overline{H}_k - H_m^+$
9:    if $I_{gain} > I_{max}$ then
10:      $I_{max} = I_{gain}$
11:      $\mathbf{u}^* = \mathbf{u}_m^+$
12:    end if
13: end for
14: execute $\mathbf{u}^*$
```

The Sensor Planning for Pose Estimation and Model Identification (SPPEMI) Algorithm 4.1 details how to implement the above analysis in a concise recursive form. Note that the algorithm makes use of the following additional definitions to simplify some of the expressions:

$$\begin{aligned}
bel_{k,i} &= p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, M_i)\,, \\
\overline{bel}_{k,i} &= p(\mathbf{X}_{k,i}|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i)\,, \\
\overline{p}_{k,i} &= P(M_i|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1})\,, \\
\overline{H}_k &= H(M|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1})\,.
\end{aligned}$$

During each algorithm cycle, the current model entropy is calculated and the expected model entropy is estimated for the set of all possible future actions, $\mathbf{u}^+$. The future action which yields the greatest information gain is then selected and executed.

4.3.1 Function: *calcCurrentEntropy(·)*

The function *calcCurrentEntropy*(·) computes the current model entropy based on the available data and robot state. It is responsible for updating Bayes' Filter and the model probabilities for all models, i.e., the internal terms: $\overline{bel}_{k,i}$, $bel_{k,i} \rightarrow \mathbf{X}_{k,i}$, and $\overline{p}_{k,i}$. The following pseudocode provides an outline of how this function should be implemented.

Function 4.2 *calcCurrentEntropy*$(\mathbf{X}_{k-1,i}, \mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, M)$

```
1: $\overline{H}_k = 0$
2: for all $M_i$ do
3:     $\overline{bel}_{k,i} = \int p(\mathbf{X}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) bel_{k-1,i} d\mathbf{X}_{k-1,i}$
4:     $bel_{k,i} = p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{X}_{k,i}, \mathbf{u}_{1:k-1}, M_i)\overline{bel}_{k,i} \cdot \eta_0$
5:     $p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) = \int p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{X}_{k,i}, \mathbf{u}_{1:k-1}, M_i)\, \overline{bel}_{k,i} d\mathbf{X}_{k,i}$
6:     $\overline{p}_{k,i} = p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) \cdot \overline{p}_{k-1,i} \cdot \eta_1$
7:     $\overline{H}_k = -\overline{p}_{k,i} \log \overline{p}_{k,i} + \overline{H}_k$
8: end for
9: return $\overline{H}_k$
```

Note that lines 3-4 of Function 4.2 implement the Bayes' Filter over the robot state. Lines 5-6 use the predicted belief state to update the model probability and line 7 calculates the current model entropy based on the calculated model probability. The terms η_0 in line 4 and η_1 in line 6 are normalizing constants that ensures the pdfs integrate/sum to 1.

4.3.2 Function: *calcPossibleControlActions(·)*

The function *calcPossibleControlActions*(·) depends on the limits of the actuating hardware. Additional criteria, such as the costs of control actions in proportion to power consumption, can be added but those details are omitted here. In general, the set of potential control actions, $\mathbf{u}^+ = \left\{\mathbf{u}_m^+ \in \mathbb{R}^d | 0 < m < M_u\right\}$, for M_u total actions, is a discretized set of controls, each of which is potentially executable at timestep k due to the underlying myopic assumption associated with the cost function. Potentially at the very next timestep, the process will repeat and new control actions will be generated. The control values themselves are dependent on the dynamics of the system and are thus difficult to generalize into a pseudocode.

4.3.3 Function: *calcExpectedEntropy(·)*

The function *calcExpectedEntropy*$(\cdot)$ computes the expected model entropy for a potential future control input, $\mathbf{u}_m^+$. It makes use of the $bel_{k,i}$, $\overline{p}_{k,i}$, and $\mathbf{X}_{k,i}$ terms computed in the *calcCurrentEntropy*$(\cdot)$ function to predict forward the future state of the object pose for all possible models under the potential action $\mathbf{u}_m^+$. Once calculated, Monte Carlo sampling is then performed for each model to extract expected data measurements and to determine model entropies. The resultant entropies are then averaged over all models to yield an expected model entropy. The following pseudo-code provides an outline of how this function should be implemented:

Function 4.3 *calcExpectedEntropy*$(\mathbf{X}_{k-1,i}, \mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M)$

```
1: $H_m^+ = 0$
2: for all $M_i$ do
3:     $G = 0$
4:     $\overline{bel}_{k+1,i} = \int p(\mathbf{X}_{k+1,i}|\mathbf{X}_{k,i}, \mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_i)\, bel_{k,i} d\mathbf{X}_{k,i}$
5:     $p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_i) = \int p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{X}_{k+1,i}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_i)\overline{bel}_{k+1,i} d\mathbf{X}_{k+1,i}$
6:     for $n = 1$ to $N$ do
7:         $h = 0$
8:         sample $\tilde{\mathbf{D}} \sim p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_i)$
9:         for all $M_j$ do
10:            $P(M_j|\mathbf{D}_{1:k}, \tilde{\mathbf{D}}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+) = p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_j) \cdot \overline{p}_{k,j} \cdot \eta_2$
11:            $h = -P(M_j|\mathbf{D}_{1:k}, \tilde{\mathbf{D}}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+) \cdot \log P(M_j|\mathbf{D}_{1:k}, \tilde{\mathbf{D}}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+) + h$
12:        end for
13:        $G = h + G$
14:    end for
15:    $G = \frac{1}{N} \cdot G$
16:    $H_m^+ = G \cdot \overline{p}_{k,i} + H_m^+$
17: end for
18: return $H_m^+$
```

Note that Line 4 makes use of the dynamic prediction step of Bayes' Filter to predict the state of the system, $\overline{bel}_{k+1,i}$, if the potential control action $\mathbf{u}_m^+$ is executed. Line 5 uses the predicted state to generate the probability distribution over the future data, $p(\mathbf{D}_{k+1}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_i)$. Lines 6-15 then implement the Monte Carlo Sampling step discussed in section 4.2.3. Line 8 samples a future data measurement ($\tilde{\mathbf{D}}$) N times from the pdf generated in Line 5. Lines 10-11 calculate the resultant model entropy based on $\tilde{\mathbf{D}}$, and lines 13 and 15 apply the averaging over all N entropies calculated. Line 16 applies the expectation over model ID and weights the resultant entropy (G) by the model probability, $\overline{p}_{k,i}$. Note that in line 10, the term η_2 is a normalizing constant to ensure

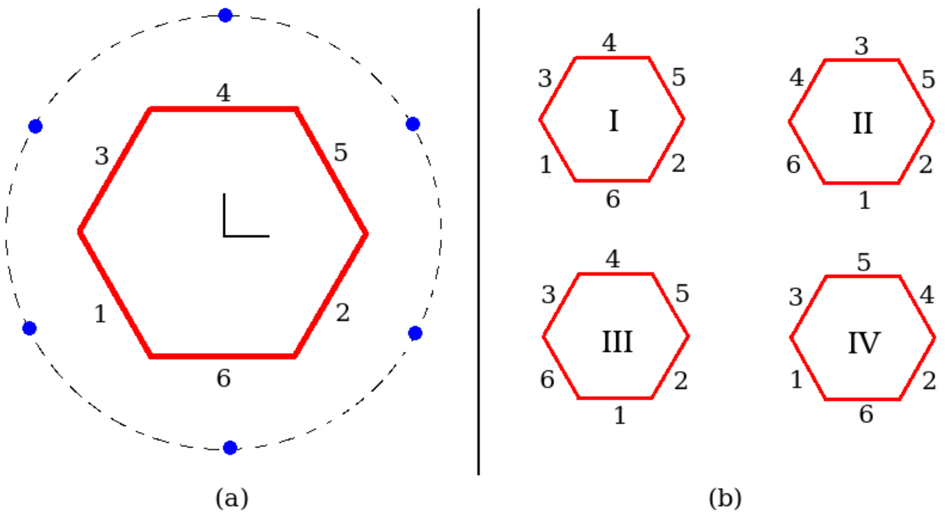


Figure 4.2: (a) An example of an n-sided polygon ($n = 6$) with viewpoints shown by dots. (b) An example set of known polygon models, with edges marked with one of six colors: red, orange, yellow, green, blue, violet.

$P(M_j|\mathbf{D}_{1:k}, \tilde{\mathbf{D}}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+)$ sums to 1.

4.4 Case Study

To illustrate how Algorithm 4.1 can be implemented, and to highlight the limits of the proposed algorithm, consider the following case study: a mobile agent is tasked with identifying a given n-sided convex polygon as belonging to a known set of unique n-sided polygons. Each polygon has edges that are uniquely marked by an edge number (not necessarily sequential) and the agent is equipped with a sensor that can identify an edge number with some amount of noise.

To keep the problem tractable, the following assumptions are also made for this particular case study:

- The n-sided polygon is static and aligned with the global origin.[2]
- Since the polygon is static, the state $\mathbf{X}_{k,i}$ is best defined as the pose of the robot relative to the i-th polygon model at the k-th timestep.
- The robot is confined to a ring of radius, R, about the center of the polygon—allowing the state $\mathbf{X}_{k,i}$ to be defined by a bearing in the range of $[0, 2\pi)$.
- The control actions, $\mathbf{u}_k$, are chosen to move the agent between any one of n *viewpoints* (where a viewpoint is defined as a viewing position orthognal to an edge and some fixed distance along

[2]While the static assumption is not a requirement, it makes analysis of the overall system more tangible in this particular case study. Later real experiments in chapter 5 remove the static assumption and illustrate the success of the algorithm for a dynamic object. Furthermore, the assumption of alignment with an existing reference frame can be resolved by applying an object detection schema such as the method discussed in chapter 3.

the perpendicular bisector of that edge). Note that due to added noise, the robot pose may not be exactly centered on a desired viewpoint.

- The initial state $\mathbf{X}_{0,i}$ will be at one of the n viewpoint locations.
- All control actions are equal in cost (i.e. moving to viewpoint v_1 is no different than moving to viewpoint v_2).
- No sensing is performed while the robot is moving from one location to the next.
- A Markov assumption is assumed where past and future data are independent if the current state is known.

Figure 4.2(a) illustrates an example polygon with viewpoints represented by the surrounding dots, as well as four other distinct polygon models. Note that viewpoints are located at increments of $\frac{\pi}{3}$ around the ring of radius R and each polygon is unique and marked with an edge number between 1 and n.

4.4.1 Motion Model

Since the robot is confined to a planar ring about the unknown static polygon , the state can be defined by a scalar bearing measure on that ring, $\mathbf{X}_{k,i} = X_{k,i} \in [0, 2\pi)$. Similarly, since the control actions move the robot to one of n discrete locations, the control can be defined as a scalar bearing offset from the current pose to a desired viewpoint, $\mathbf{u}_k = u_k \in [-\pi, \pi]$. This construct yields the following system state equation:

$$X_{k,i} = X_{k-1,i} + u_{k-1} + \eta \,, \tag{4.11}$$

where η is Gaussian white noise (i.e. $\eta \sim N(\eta; 0, \sigma_s^2)$) to account for imperfect robot control execution.

4.4.2 Measurement Model

Consider the case now that the robot is equipped with a sensor which returns a value $D_k \in \mathbb{R}$ corresponding to the edge being detected — so that a measure of $D_k = 2.0$ indicates a measurement of edge number 2 and so forth. Note that D_k returns a floating point value, indicating the possibility of measuring a value in-between known edge values. Since each polygon model has a different sequence of numbered edges, a unique measurement model can be defined for each which will be a function of robot bearing position. To allow for a continuous range of return measurements, D_k is linearly interpolated between neighboring edge values for those bearing positions found in between

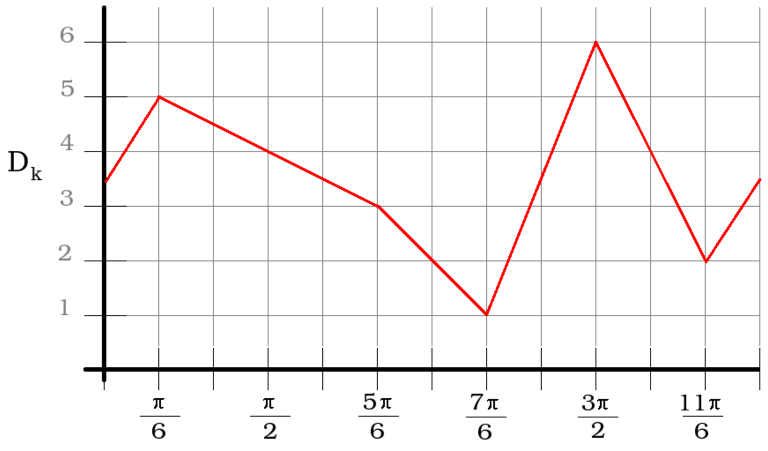


Figure 4.3: An example measurement model, plotted as a function of robot bearing for Model I as shown in Figure 4.2 with $n = 6$.

viewpoint poses. Lastly, to incorporate the notion of an imperfect sensor, Gaussian white noise is added to the received measurement value, leading to the following overall measurement model:

$$D_k = C_i \cdot X_{k,i} + F_i + \xi\,, \tag{4.12}$$

where $\xi \sim N(\xi; 0, \sigma_m^2)$ and $(C_i,\, F_i)$ are bearing dependent parameters specific to the i-th model polygon. Figure 4.3 shows an example measurement model for the Model I polygon as shown in Figure 4.2(b).

4.4.3 Applying the Algorithm

With the motion and measurement models both defined, Algorithm 4.1 can be realized with the appropriate implementation of the functions calls *calcPossibleControlActions(·)*, *calcCurrentEntropy(·)*, and *calcExpectedEntropy(·)*.

The function *calcPossibleControlActions(·)* is understood to generate a set of the possible control commands $\mathbf{u}^+$ that can potentially be executed at timestep k. For this case study, since the robot is limited to discrete poses on a ring of radius R, the set of possible control actions are simply bearing displacements corresponding to each viewpoint pose allowed for the given n-sided polygon.

The function *calcCurrentEntropy(·)* is responsible for determining the current model entropy, H_k, based on the set of perceived measurements, D_k. In particular, Lines 3 and 4 of Function 4.2 define the Bayes Filter implementation which reduces to a Kalman Filter because of the linear motion and measurement models with white Guassian noise as was shown in section 2.1.1:

DYNAMIC PREDICTION

$$\begin{aligned}\overline{bel}_{k,i} &= N(X_{k,i}; \overline{\mu}_{k,i}, \overline{\sigma}^2_{k,i})\,, \qquad (4.13)\\ \overline{\mu}_{k,i} &= \mu_{k-1,i} + u_{k-1}\,,\\ \overline{\sigma}^2_{k,i} &= \sigma^2_{k-1,i} + \sigma^2_s\,,\end{aligned}$$

MEASUREMENT UPDATE

$$\begin{aligned}bel_{k,i} &= N(X_{k,i}; \mu_{k,i}, \sigma^2_{k,i})\,,\\ \mu_{k,i} &= \overline{\mu}_{k,i} + K_{k,i}(D_k - (C_i\overline{\mu}_{k,i} + F_i))\,,\\ \sigma^2_{k,i} &= (1 - K_{k,i}C_i)\cdot\overline{\sigma}^2_{k,i}\,,\\ \Longrightarrow K_{k,i} &= \frac{C_i\overline{\sigma}^2_{k,i}}{C_i^2\overline{\sigma}^2_{k,i} + \sigma^2_m}\,.\end{aligned}$$

However, before the model entropy H_k can be fully expressed, the data likelihood must be resolved in line 5 of Function 4.2:

$$\begin{aligned}p(D_k|D_{1:k-1}, u_{1:k-1}, M_i) &= \int p(D_k|D_{1:k-1}, u_{1:k-1}, X_k, M_i)\cdot\overline{bel}_{k,i}dX_k \qquad (4.14)\\ &= \int N(D_k; C_iX_k + F_i, \sigma^2_m)\cdot N(X_k; \overline{\mu}_{k,i}, \overline{\sigma}^2_{k,i})dX_k\\ &= N(D_k; \mu_{D_k,i}, \sigma^2_{D_k,i})\,,\end{aligned}$$

where by the linear nature of the measurement model, the data probability pdf of equation (4.14) is analytically solved for by an application of Proposition 2.1, with $\mu_{D_k,i}$ and $\sigma^2_{D_k,i}$ given by

$$\begin{aligned}\mu_{D_k,i} &= C_{k,i}\overline{\mu}_{k,i}\,,\\ \sigma^2_{D_k,i} &= C_{k,i}\overline{\sigma}^2_{k,i}C^T_{k,i} + \sigma^2_m\,.\end{aligned}$$

Note that the function *calcExpectedEntropy(·)* is responsible for determining the expected model entropy H^+_m associated with the potential future control action $\mathbf{u}^+_m$. Close inspection of Function 4.3 will show that a majority of the terms are no different than terms already solved for from determining the current model entropy H_k in the *calcCurrentEntropy(·)* function (with the exception of a simple index shift) and with only two additional pdfs needing to be solved for to complete the implementation of the algorithm: $p(D_{k+1}|D_{1:k}, u_{1:k-1}, u^+_m, M_i)$ and $\overline{bel}_{k+1,i}$.

The first pdf, $p(D_{k+1}|D_{1:k}, u_{1:k-1}, u^+_m, M_i)$, corresponds to the probability distribution over future sensed data at timestep $k+1$ that would be sensed based on executing potential control action u^+_m. While this is a generally difficult distribution to come by, the manner in which this particular case study was constructed makes the prediciton of future data fairly straight forward from the measurement model of equation (4.12).[3] The term can be resolved by an application of proposition 2.1:

$$\begin{aligned}
p(D_{k+1}|D_{1:k}, u_{1:k-1}, u_m^+, M_i) &= \int p(D_{k+1}|D_{1:k}, X_{k+1,i}, u_{1:k-1}, u_m^+, M_i) \cdot \overline{bel}_{k+1,i} dX_{k+1,i} \\
&= \int N(D_{k+1}; C_i X_{k+1,i} + F_i) \cdot N(X_{k+1,i}; \overline{\mu}_{k+1,i}, \overline{\sigma}^2_{k+1,i}) dX_{k+1,i} \\
&= \frac{1}{\sigma_{D_{k+1}}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(D_{k+1} - \mu_{D_{k+1},i})^2}{\sigma^2_{D_{k+1},i}}\right),
\end{aligned}$$

where $\mu_{D_{k+1},i}$ and $\sigma_{D_{k+1},i}$ are given by

$$\begin{aligned}
\mu_{D_{k+1},i} &= C_i \overline{\mu}_{k+1,i} + F_i\,, \\
\sigma^2_{D_{k+1},i} &= \overline{\sigma}^2_{k+1,i} C_i^2 + \sigma_m^2\,.
\end{aligned}$$

The second pdf, $\overline{bel}_{k+1,i}$, corresponds to the future state estimate of the system, $X_{k+1,i}$, as a result of executing the potential control action u_m^+. It can be resolved in a manner analogous to equation (4.13):

$$\begin{aligned}
\overline{bel}_{k+1,i} &= \int p(X_{k+1,i}|X_{k,i}, D_{1:k}, u_{1:k-1}, u_m^+, M_i) \cdot bel_{k,i} dX_{k,i} \\
&= \int N(X_{k+1,i}; X_{k,i} + u_m^+, \sigma_s^2) \cdot N(X_{k,i}; \mu_{k,i}, \sigma^2_{k,i}) dX_{k,i} \\
&= \frac{1}{\overline{\sigma}_{k+1,i}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(X_{k+1,i} - \overline{\mu}_{k+1,i})^2}{\overline{\sigma}^2_{k+1,i}}\right),
\end{aligned}$$

where $\overline{\mu}_{k+1,i}$ and $\overline{\sigma}^2_{k+1,i}$ are given by

$$\begin{aligned}
\overline{\mu}_{k+1,i} &= \mu_{k,i} + u_m^+\,, \\
\overline{\sigma}^2_{k+1,i} &= \sigma^2_{k,i} + \sigma_s^2\,.
\end{aligned}$$

4.4.4 Simulated Results

In this particular case study, simulations were run to observe what the limits of the algorithm would be in terms of the number of steps taken to positively identify the correct model. The tunable parameters were chosen to be:

- system noise standard deviation, $\sigma_s \in [0.05, 2.00]$

[3]Chapter 5 will present an approach to determining the future data likelihood ($p(D_{k+1}|D_{1:k}, u_{1:k-1}, u_m^+, M_i)$ for a real experiment in which an exact predictive measurement model is not easily determined as was the case in this particular case study.

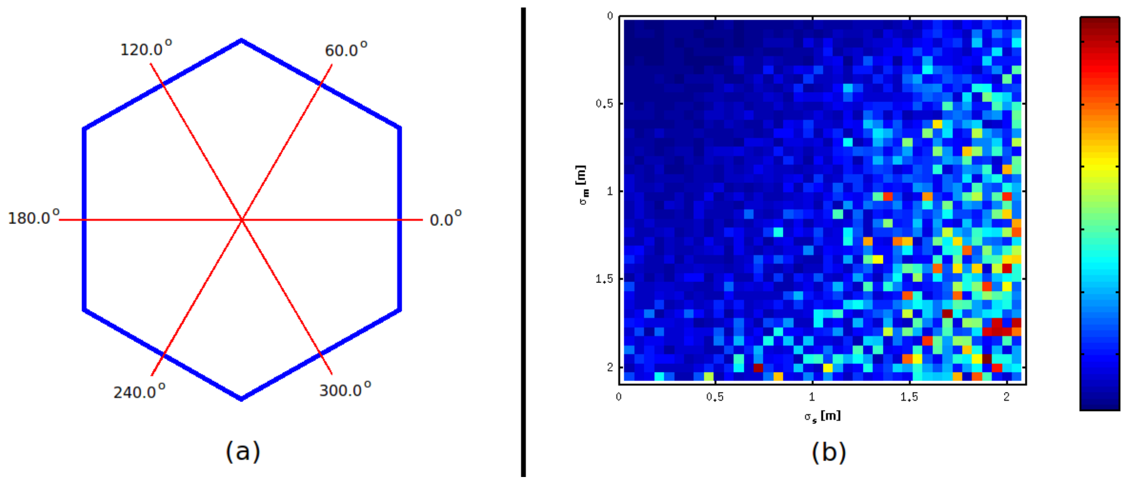


Figure 4.4: (a) For the case of m=4 and n=6, the shown polygon illustrates the overall shape of the object to be identified as well as the angles associated with the n viewpoints. (b) The shown plot displays a color coding of the number of steps taken by the algorithm to positively identify the given model for each setting of $\sigma_s \in [0.05, 2.00]$ and $\sigma_m \in [0.05, 2.00]$. The color bar on the right of the plot displays the mapping of colors to number of steps.

- measurement noise standard deviation, $\sigma_m \in [0.05, 2.00]$
- number of known models, $m \in$ 4, 6, 8
- number of polygon edges, $n \in$ 6, 8, 10

where the allowable ranges have been specified. In each simulated case, a specific m and n were chosen and a database of models was then generated. The choice of database models was such that enough overlap between models existed so that the minimal number of steps required to positively identify the unknown model was greater than 1. Once a valid database of models was generated, Algorithm 4.1 was run recursively for varying values of the noise parameters σ_s and σ_m and the number of steps taken to positively identify the model were recorded for each set of (σ_s, σ_m).

4.4.4.1 m=4, n=6

In the case of m=4 and n=6, Figure 4.4(a) shows the general shape of the type of polygon being identified, with the viewpoint angles denoted by the red lines. Table 4.1 lists the unique edge labels for each model in the database, categorized by viewpoint angle. Note that each model is in fact unique, yet similar enough that from certain viewpoints (e.g. 0°, 60°, 300°), a number of models in the database would be indistinguishable unless an additional measurement were taken from another viewpoint. This particular choice of database models was designed so that sensor planning actions would be required to test the performance of the algorithm. The model with an asterisk denotes the true model used in the simulated test runs to generate the measurement data. Close inspection of the table will show that the true model can in fact be determined by as few as 1 step (e.g. moving from 0° to 120°). However, without knowledge of the true model, the expected number of steps is in fact 3.

Model ID	0°	60°	120°	180°	240°	300°
0	1	5	6	4	2	3
1	1	5	2	4	6	3
2*	1	5	4	2	6	3
3	1	5	6	2	4	3

Table 4.1: The above table indicates the specific edge numbers (categorized by viewpoint angle) for each model used in this particular case study. The (*) indicates the true model used in the simulations to generate measurement data.

Simulated runs were performed by applying Algorithm 4.1 to the presented case study, with varying values of measurement and system variance, σ_m and σ_s respectively. In particular, each case run considered a specific value of $\sigma_m \in [0.05, 2.00]$ and $\sigma_s \in [0.05, 2.00]$ and recorded the number of steps taken by the algorithm to positively identify the given model, with an initial viewpoint position always at 0°. Figure 4.4(b) shows a color coded histogram of the number of steps required by the algorithm for the varying values of σ_m and σ_s. Note that for $\sigma_m = 0.05$ and $\sigma_s = 0.05$, the algorithm identifies the correct model in the expected number of steps, 3. For values of $\sigma_m < 1.0$ and $\sigma_s < 1.0$, the algorithm performs fairly well and correctly identifies the true model in less than 10 steps. However, once σ_m and σ_s increase past 1.0, the algorithm starts to take a significantly large number of steps to accurately identify the model. Part of that result is dependent on the manner in which the measurement model for this case study was designed. As σ_s increases, the estimated pose of the robot starts to deviate significantly from expected viewpoint locations and towards areas in between edges. Consequently, because of the linear interpolation between edge values in the measurement model and the increased value of σ_m, perceived measurements start to resemble measurements of other sides of the polygon and it becomes slightly unclear which edge is actually being sensed. Multiple steps then need to be taken by the robot to revisit previously observed edges to reduce the overall uncertainty of the robot pose and improve the estimated model probabilities before the true model can be identified.

4.4.4.2 m=6, n=8

In the case of m=6 and n=8, Figure 4.5(a) shows the general shape of the type of polygon being identified, with the viewpoint angles denoted by the red lines. Table 4.2 lists the unique edge labels for each of the 6 models in the database, categorized by viewpoint angle. As in the previous test case, note that each model is in fact unique, yet similar enough that from certain viewpoints (e.g. 0°, 45°, 315°), a number of models in the database would be indistinguishable unless an additional measurement were taken from another viewpoint. The model with an asterisk again denotes the true model used in the simulated test runs to generate the measurement data. Close inspection

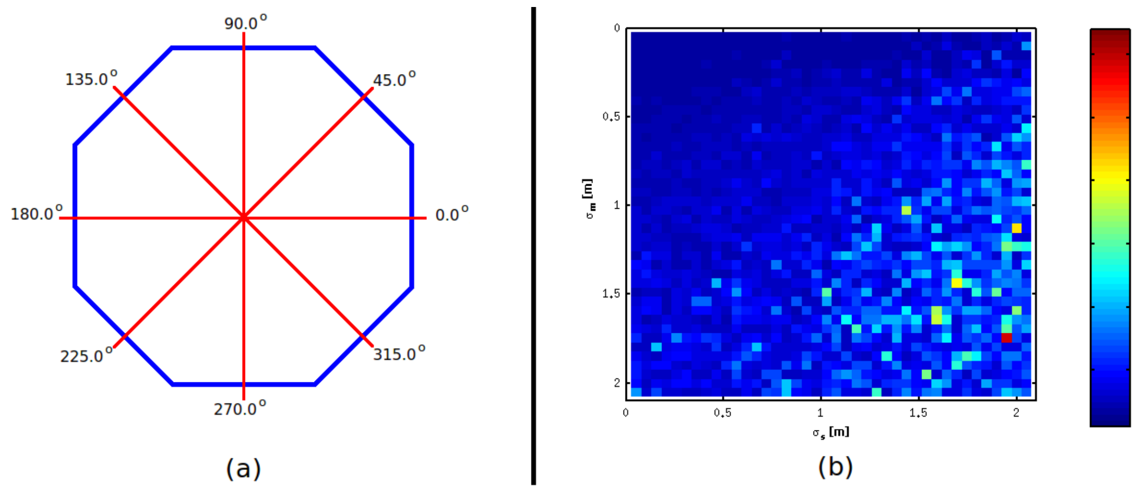


Figure 4.5: (a) For the case of m=6 and n=8, the shown polygon illustrates the overall shape of the object to be identified as well as the angles associated with the n viewpoints. (b) The shown plot displays a color coding of the number of steps taken by the algorithm to positively identify the given model for each setting of $\sigma_s \in [0.05, 2.00]$ and $\sigma_m \in [0.05, 2.00]$. The color bar on the right of the plot displays the mapping of colors to number of steps.

of the table will show that if starting at initial viewpoint angle 0°, the true model can in fact be determined by as few as 1 step (e.g. moving from 0° to 180° and observing edge 8). However, without knowledge of the true model, the expected number of steps is again in fact 3.

Model ID	0°	45°	90°	135°	180°	225°	270°	315°
0	6	5	3	8	7	4	1	2
1	6	5	1	7	4	8	3	2
2*	6	5	7	3	8	4	1	2
3	6	5	1	3	4	7	8	2
4	6	5	8	7	3	4	1	2
5	6	5	7	8	3	4	1	2

Table 4.2: The above table indicates the specific edge numbers (categorized by viewpoint angle) for each model used in this particular case study. The (*) indicates the true model used in the simulations to generate measurement data.

As in the previous test case, simulated runs were performed by applying Algorithm 4.1 with varying values of measurement and system variance, σ_m and σ_s respectively. Using the same range of values of $\sigma_m \in [0.05, 2.00]$ and $\sigma_s \in [0.05, 2.00]$, the number of steps taken by the algorithm to positively identify the given model (with an initial viewpoint position always at 0°) was again recorded. Figure 4.5(b) shows a color coded histogram of the number of steps required by the algorithm for the varying values of σ_m and σ_s. Note that the results illustrate the same pattern of behavior as in the previous case study, i.e., for small values of σ_m and σ_s, the algorithm identifies the correct model in the expected number of steps and as σ_m and σ_s gradually increase, the algorithm takes significantly larger number of steps to accurately identify the model.

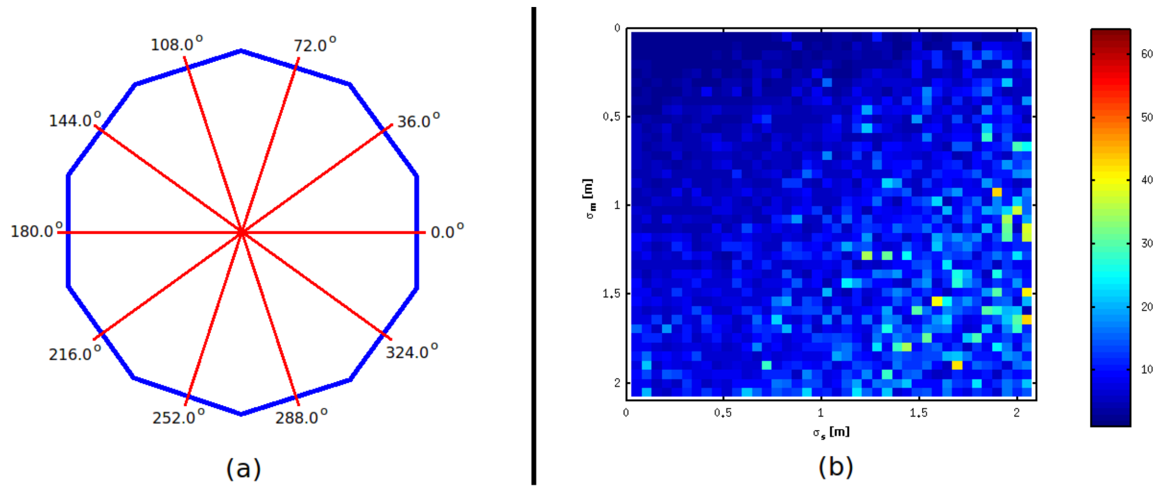


Figure 4.6: (a) For the case of m=8 and n=10, the shown polygon illustrates the overall shape of the object to be identified as well as the angles associated with the n viewpoints. (b) The shown plot displays a color coding of the number of steps taken by the algorithm to positively identify the given model for each setting of $\sigma_s \in [0.05, 2.00]$ and $\sigma_m \in [0.05, 2.00]$. The color bar on the right of the plot displays the mapping of colors to number of steps.

4.4.4.3 m=8, n=10

In the case of m=8 and n=10, Figure 4.6(a) shows the shape of the polygon being identified, with the viewpoint angles denoted by the red lines. Table 4.3 lists the unique edge labels for each of the 8 models in the database, categorized by viewpoint angle.

Model ID	0°	36°	72°	108°	144°	180°	216°	252°	288°	324°
0	5	6	9	4	1	2	7	10	3	8
1	5	6	9	1	7	10	2	4	3	8
2*	5	6	9	2	1	4	7	10	3	8
3	5	6	9	1	4	10	7	2	3	8
4	5	6	9	10	1	2	7	4	3	8
5	5	6	9	10	7	2	4	1	3	8
6	5	6	9	10	2	4	1	7	3	8
7	5	6	9	2	7	10	4	1	3	8

Table 4.3: The above table indicates the specific edge numbers (categorized by viewpoint angle) for each model used in this particular case study. The (*) indicates the true model used in the simulations to generate measurement data.

As in the previous test cases, simulated runs were performed by applying Algorithm 4.1 with varying values of measurement and system variance, σ_m and σ_s respectively. Using the same range of values of $\sigma_m \in [0.05, 2.00]$ and $\sigma_s \in [0.05, 2.00]$, the number of steps taken by the algorithm to positively identify the given model (with an initial viewpoint position always at 0°) was again recorded. Figure 4.6(b) shows the color coded histogram of the number of steps required by the algorithm for the varying values of σ_m and σ_s. Note that the results illustrate the same pattern of behavior as in the previous two simulations, i.e., for small values of σ_m and σ_s, the algorithm

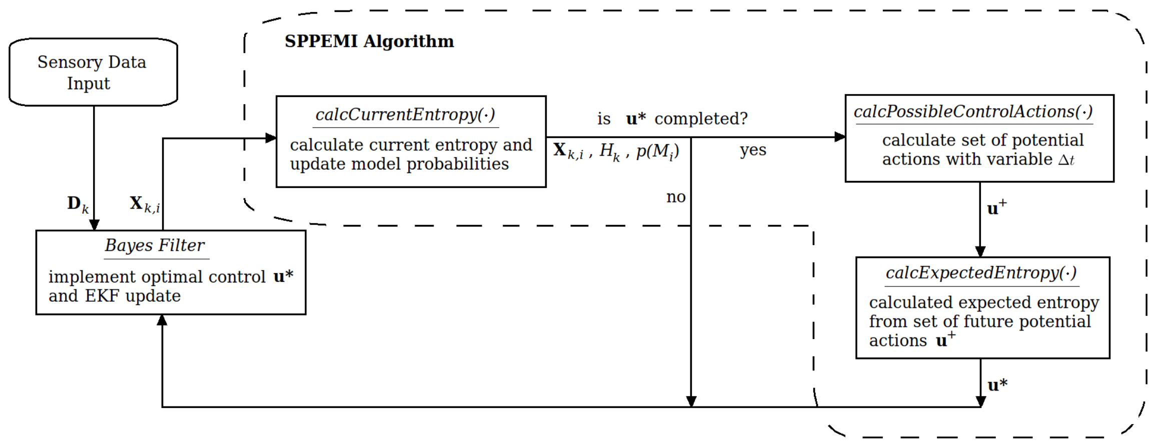

Sensory Data Input
SPPEMI Algorithm
calcCurrentEntropy(·)
calculate current entropy and update model probabilities
is u* completed?
$\mathbf{X}_{k,i}$, H_k, $p(M_i)$
yes
no
calcPossibleControlActions(·)
calculate set of potential actions with variable Δt
$\mathbf{u}^+$
calcExpectedEntropy(·)
calculated expected entropy from set of future potential actions $\mathbf{u}^+$
$\mathbf{u}^*$
$\mathbf{D}_k$
$\mathbf{X}_{k,i}$
Bayes Filter
implement optimal control $\mathbf{u}^*$ and EKF update

Figure 4.7: The shown block diagram illustrates a typical implementation of the SPPEMI Algorithm which limits the information gain calculations to being computed when the system state has reached the state determined by the optimal control action $\mathbf{u}^*$.

identifies the correct model in the expected number of steps and as σ_m and σ_s gradually increase, the algorithm takes significantly larger number of steps to accurately identify the model.

Based on the results of these simulated cases, it becomes clear that the algorithm is sensitive to large noise parameters in both the motion model and the measurement model. While the case study itself is not necessarily a real-life example, it does highlight the fact that with an accurate sensor and fairly good pose estimate, the algorithm can be expected to perform in a manner which successfully identifies the true object in a minimal number of steps.

4.5 Practical Considerations

Given the layout of Algorithm 4.1, it would seem plausible that a typical implementation could be achieved by recursively applying the steps listed. However, as will be discussed further in chapter 5, the computational expenses associated with running Algorithm 4.1 at each timestep are impractical. A more practical approach would be to limit function calls to *calcPossibleControlActions*(·) and *calcExpectedEntropy*(·) until the final state prescribed by the control action $\mathbf{u}^*$ is achieved. This leads to a behavior whereby the agent is committed to completing a chosen action (once selected) before being allowed to consider the next set of actions to take.

Shown in Figure 4.7 is a block diagram illustrating a practical implementation of the algorithm. Note that while the information gain calculations are limited from being executed each timestep, the current model entropy and model probabilities can still be computed while $\mathbf{u}^*$ is being executed.

In the following chapter, the algorithm and theory relating to the SPPEMI Algorithm will be tested on two experiments. For each experiment, the entire implementation details will be presented. Additional discussion will be added where approximations have been made to simplify expressions.

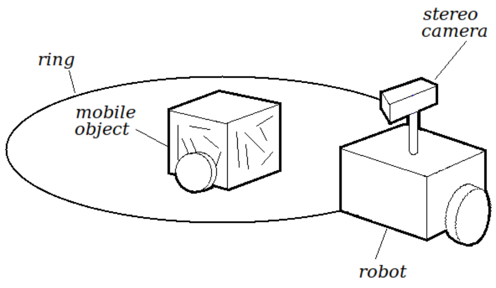
Chapter 5

Implementation of SPPEMI

The content of this chapter will focus on the implementation details and experimental results relating to two experiments that were carried out to test the Sensor Planning for Pose Estimation and Model Identification (SPPEMI) algorithm proposed in the previous chapter to not only plan for future sensor actions but also to simultaneously calculate 6D pose estimates and identify an unknown object model. In the first experiment, the robot and the unidentified object were allowed to move about in the working environment. However, as was the case in the case study of the previous chapter, the robot motions were confined to a ring about the object and the object motions (which were operator controlled) were confined to within the field of view of the robot stereo camera. In the second experiment, a similar setup to the first experiment was considered, although the robot was no longer confined to a ring about the object and the object was allowed to freely move about the workspace. In this chapter, the implementation details and experimental results of both experiments are presented and discussed.

5.1 Experiment 1: Constrained Mobile Agent and Object

In this experiment a two-wheeled, non-holonomic robot is tasked with visually identifying an unknown mobile object (whose movements are operator controlled) from a finite database of M known models. For simplicity, the mobile agent is confined to move on a ring initially centered about the unknown moving object. Similar to the case study examined in the previous chapter, control actions of the agent are restricted to a discrete set of wheel speeds and turn rates corresponding to locations along the circumference of the ring. This reduces the agent's pose to a single degree of freedom, which is parameterized by the angle of its position on the ring circumference (where the angle is measured with respect to a global coordinate frame placed at the center of the circle) – see Figure 5.1. The agent is equipped with onboard wheel odometry for localization and a fixed stereo camera head for sensing. SIFT features augmented with 3D sparse stereo data are used, and object identification is done by comparing known model features in a database to current features in the image, as was detailed in chapter 3. The following subsections will discuss the application of Algorithm 4.1



(a)

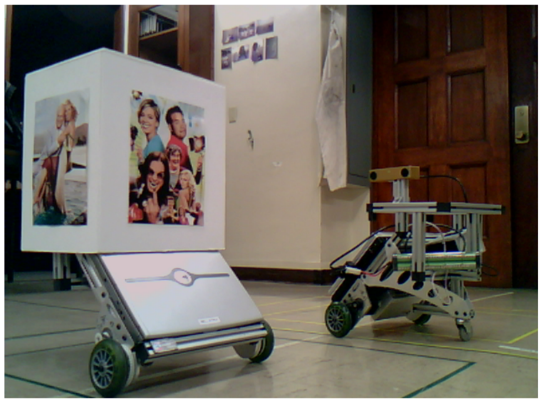

(b)

Figure 5.1: (a) A schematic of the experiment carried out where the agent is mobile yet confined to a ring about the object being identified. The object itself is also mobile and allowed to freely move within the confines of the ring. (b) A picture of the actual setup in the laboratory for this experiment.

and the implementation details for this particular experiment.

5.1.1 Motion Model

The state of the system is defined as the pose of the object relative to the camera reference frame:

$$\mathbf{X}_{k,i} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_\theta \end{bmatrix} \in \mathbb{R}^6 ,$$

where the object translational pose is given by $\mathbf{x}_R = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and the Euler angle parameters given by $\mathbf{x}_\theta = [\ \alpha \quad \beta \quad \gamma\ ]^T$. Note that because the object itself is dynamic, an individual pose estimate is required for each model in the known database being tracked, making the subscript i necessary (whereas in the case study of the previous chapter, because of the static object assumption with a globally coincident reference frame, the i subscript could be dropped). Figure 5.2(b) provides a schematic detailing the motion of the robot-camera system between timesteps $k-1$ and k. Note the various reference frames and transforms involved in this experiment: $\{G_{C_{k-1}O_{k-1}}, G_{R_{k-1}C_{k-1}}, G_{R_{k-1}R_k}, G_{R_kC_k}, G_{C_kO_k} \in SE(3)\}$ where (R) refers to the robot reference frame, (C) the camera reference frame, and (O) the object reference frame. Because the camera is rigidly mounted on top of the robot, the transform $G_{R_kC_k}$ is constant, i.e., $G_{R_kC_k} = G_{R_{k-1}C_{k-1}} = G_{RC}$. Using the appropriate frame transformations, the following kinematic relation can be derived:

$$G_{C_k,O_k} = G_{RC}^{-1} \cdot G_{R_{k-1}R_k}^{-1} \cdot G_{RC} \cdot G_{C_{k-1},O_{k-1}} \cdot G_{O_{k-1}O_k} \quad . \tag{5.1}$$

equation (5.1) defines the motion model of the robot, provided the relevant terms can be expressed.

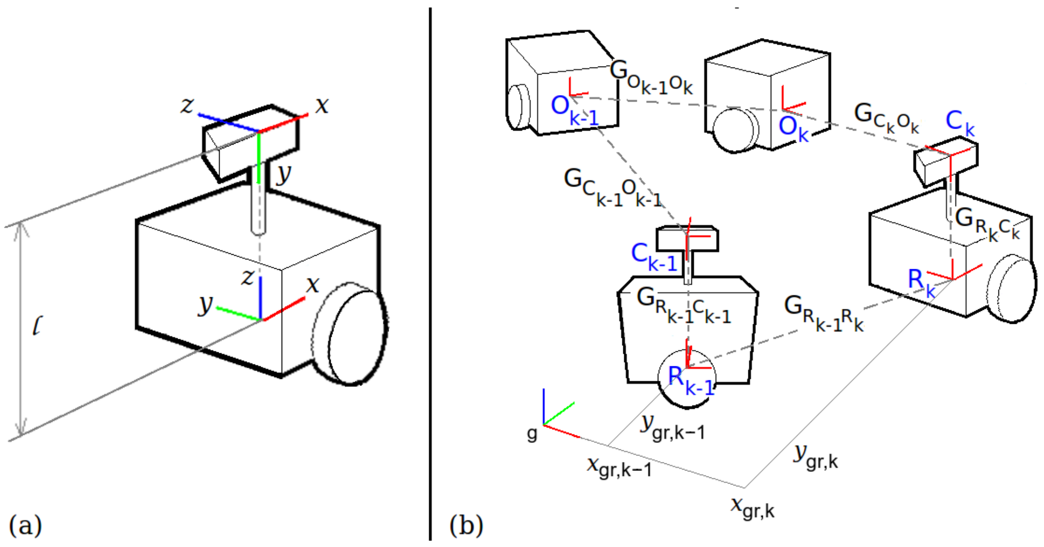


Figure 5.2: (a) The camera reference frame is offset from the robot transform by a constant translation (l) and a rotation of $\frac{\pi}{2}$ about the $x-axis$. (b) A schematic showing the various transforms involved in the experiment, when the robot moves from one timestep k to the next timestep $k+1$. Note the reference frame notation: the camera reference frame (C), the object reference frame (O), and the robot reference frame (R).

In this experiment, the origin of the camera is elevated above the robot origin by a distance l along the robot *z-axis* with the camera *z-axis* aligned with the optical axis (see Figure 5.2(a)). This configuration yields an inward-facing camera as the robot traverses the path of the ring, and G_{RC} can be found to be

$$G_{RC} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.2}$$

The transform $G_{R_{k-1}R_k}$ can be determined using a traditional two-wheeled non-holonomic kinematic model of the robot body – an approach used in determining robot odometry. Let the robot control input at time k be defined as follows:

$$\mathbf{u}_k = \begin{bmatrix} u_s \\ u_\omega \end{bmatrix}_k \in \mathbb{R}^2 ,$$

where u_s represents a speed command to the robot (measured in $[m/s]$) and u_ω represents an angular rate command to the robot (measured in $[rad/s]$). Because the robot motion is confined to a plane, the robot body pose can be defined in a global reference frame[1] by three variables (see Fig. 5.2):

$$\mathbf{x}_{gr,k} = \begin{bmatrix} x_{gr} & y_{gr} & \theta_{gr} \end{bmatrix}_k^T ,$$

where x_{gr} and y_{gr} represent the translation component of the robot in the global x-y plane, and θ_{gr} represents the heading of the robot relative to the global coordinate frame. The robot kinematic model can thus be defined as

$$\begin{bmatrix} x_{gr} \\ y_{gr} \\ \theta_{gr} \end{bmatrix}_k = \begin{bmatrix} x_{gr} \\ y_{gr} \\ \theta_{gr} \end{bmatrix}_{k-1} + \underbrace{\begin{bmatrix} \delta_{x,k-1} \\ \delta_{y,k-1} \\ \delta_{\theta,k-1} \end{bmatrix}}_{\overline{\delta}_{k-1}} , \tag{5.3}$$

where $\overline{\delta}_{k-1} \in \mathbb{R}^3$ represents the change in motion of the robot body between timesteps $k-1$ and k and is given by

$$\overline{\delta}_{k-1} = \begin{bmatrix} u_{s,k-1}\Delta t \cdot cos(\theta_{gr,k-1}) \\ u_{s,k-1}\Delta t \cdot sin(\theta_{gr,k-1}) \\ u_{\omega,k-1}\Delta t \end{bmatrix} ,$$

where $\Delta t = t_k - t_{k-1}$. The transformation $G_{R_{k-1}R_k}$ can thus be expressed by the following:

$$G_{R_{k-1}R_k} = \begin{bmatrix} \mathbf{R}_{R_{k-1}R_k} & \mathbf{d}_{R_{k-1}R_k} \\ \mathbf{0}^T & 1 \end{bmatrix} , \tag{5.4}$$

with $\mathbf{R}_{R_{k-1}R_k}$ and $\mathbf{d}_{R_{k-1}R_k}$ given by:

$$\mathbf{R}_{R_{k-1}R_k} = \begin{bmatrix} cos(\delta_{\theta,k-1}) & -sin(\delta_{\theta,k-1}) & 0 \\ sin(\delta_{\theta,k-1}) & cos(\delta_{\theta,k-1}) & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$

$$\mathbf{d} = \begin{bmatrix} cos(\theta_{g,k-1})\delta_{x,k-1} + sin(\theta_{g,k-1})\delta_{y,k-1} \\ -sin(\theta_{g,k-1})\delta_{x,k-1} + cos(\theta_{g,k-1})\delta_{y,k-1} \\ 0 \end{bmatrix} .$$

The transform $G_{O_{k-1}O_k}$ is derived from the object's motion model which is assumed to be a

[1]The global reference frame is defined in this case to be coincident with a coordinate frame centered at the ring origin.

random walk perturbed by noise at the velocity level, i.e.,

$$G_{O_{k-1}O_k} = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} . \tag{5.5}$$

Substituting equations (5.2), (5.4), and (5.5) into equation (5.1), the translational and rotational pose parameters can be extracted to yield the following non-linear motion model equation:

$$\begin{aligned} \mathbf{X}_{k,i} &= \begin{bmatrix} \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T \\ \begin{bmatrix} f_\alpha & f_\beta & f_\gamma \end{bmatrix}^T \end{bmatrix}_{k-1} + \eta \\ &= \mathbf{F}(\mathbf{X}_{k-1,i}, \mathbf{u}_{k-1}) + \eta \,, \end{aligned} \tag{5.6}$$

where the terms f_x, f_y, and f_z define the kinematic relations used to predict the object translational pose and are given by

$$\begin{aligned} f_x =& c\delta_{\theta,k}x + s\delta_{\theta,k}z - c\delta_{\theta,k}(c\theta_{gr,k}\delta_{x,k} + s\theta_{gr,k}\delta_{y,k}) - s\delta_{\theta,k}(-s\theta_{gr,k}\delta_{x,k} + c\theta_{gr,k}\delta_{y,k}) \,, \\ f_y =& y \,, \\ f_z =& -s\delta_{\theta,k}x + c\delta_{\theta,k}z + s\delta_{\theta,k}(c\theta_{gr,k}\delta_{x,k} + s\theta_{gr,k}\delta_{y,k}) - c\delta_{\theta,k}(-s\theta_{gr,k}\delta_{x,k} + c\theta_{gr,k}\delta_{y,k}) \,, \end{aligned}$$

where $c(\cdot) \triangleq cos(\cdot)$ and $s(\cdot) \triangleq sin(\cdot)$. The terms f_α, f_β, and f_γ define the kinematic relations used to predict the orientation of the object and are given by

$$\begin{aligned} f_\alpha =& tan^{-1}\left(\frac{s\alpha c\beta}{c\delta_{\theta,k}c\alpha c\beta - s\delta_{\theta,k}s\beta}\right) \,, \\ f_\beta =& -sin^{-1}\left(-s\delta_{\theta,k}c\alpha c\beta - c\delta_{\theta,k}s\beta\right) \,, \\ f_\gamma =& tan^{-1}\left(\frac{-s\delta_{\theta,k}(c\alpha s\beta s\gamma - s\alpha c\gamma) + c\delta_{\theta,k}c\beta s\gamma}{-s\delta_{\theta,k}(c\alpha s\beta c\gamma + s\alpha s\gamma) + c\delta_{\theta,k}c\beta c\gamma}\right) \,, \end{aligned}$$

where $\eta \in \mathbb{R}^6$ is Gaussian white system noise with covariance given by $\mathbf{\Sigma}_s \in \mathbb{R}^{6\times 6}$.

5.1.2 Measurement Model

In this experiment 3D SIFT features were chosen as the type of measurement data since the working environment required highly descriptive features to ensure robust feature matching. SIFT features are widely used for their invariance to rotations and scale and high probability of matching; as such, it was an obvious choice to use for this experiment. This allows for the measurement model of this experiment to be identical to equation (3.5) developed in chapter 3 (reproduced here for convenience):

$$
\begin{aligned}
\mathbf{D}_k &= \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(0)} \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(1)} \\ \vdots \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(n_K)} \end{bmatrix} + \xi\,, \\
&= \mathbf{H}(\mathbf{X}_{k,i}, \mathbf{B}_i) + \xi \qquad (5.7)
\end{aligned}
$$

where the terms $\mathbf{D}_k$, $\mathbf{y}_j$, $\mathbf{b}^i_{\mathbf{J}(j)}$, and $\mathbf{J}_i(j)$ are the same as was defined in section 3.2.2. Recall the correspondence variable $\mathbf{J}_i(j)$ is found by employing a Best-Bin-First Search algorithm as first described by Lowe in [30] and optimized using a *k-d* tree to yield a high probability of accurate correspondences. Features in the database are generated off-line during a *training phase* and then catalogued according to each object, identical to the approach discussed in section 3.2.3. It should be noted that the 3D SIFT features are only extracted from a fixed subwindow of the image to increase computational efficiency. While an ROI approach could have been used in this experiment as discussed in section 3.2.4, it was discovered that because of the multiple model state estimates involved, an independent ROI would have to be used for each model which was not conducive to a multi-threaded framework and thus only added unnecessary computation.

5.1.3 Applying the Algorithm

With the motion and measurement models both defined, Algorithm 4.1 can be implemented in a similar manner as was done in the previous chapter, i.e., with appropriate implementation of the functions calls *calcPossibleControlActions(·)*, *calcCurrentEntropy(·)*, and *calcExpectedEntropy(·)* as outlined in Figure 4.7 (where because of the computational expenses involved with running an iteration of the SPPEMI algorithm each timestep, information gain calculations are executed only after a commanded control action has been completed).

For this experiment, much of the robot dynamics are simplified because the robot is constrained to motion on a ring and the robot inputs u_s and u_ω are related by the ring radius: $u_s = \frac{R_{ring}}{u_\omega}$. By specifying u_ω, u_s is also specified and for some Δt, a future robot position on the ring can be predicted. As such, the set of possible control actions determined by the function *calcPossibleControlActions(·)* can be reduced to a set of discrete future locations on the ring. In this experiment, the set of controls corresponding to robot positions on the ring ranging from 0° to 360° in steps of 10° are considered. The radius of the ring R_{ring} is chosen specifically to be the same nominal distance between the object and the robot during the training phase – thus increasing the probability of detection and feature matching overall.

Recall that the function *calcCurrentEntropy(·)* is responsible for determining the current model entropy, H_k, based on the set of perceived measurements, $\mathbf{D}_k$. As was shown in the previous chapter, this can be achieved by an application of a Bayes Filter. In particular, Lines 3 and 4 of Function 4.2 define the Bayes Filter implementation and can be resolved if a Markov assumption is made and an assumption of Gaussian white noise on both the motion and measurement models of equations (5.6) and (5.7) considered. Because of the non-linearity of the measurement model in equation (5.7), Bayes Filter reduces to an Extended Kalman Filter as was shown in section 2.1.2:

DYNAMIC PREDICTION

$$\begin{aligned} \overline{bel}_{k,i} &= N(\mathbf{X}_{k,i}; \overline{\mu}_{k,i}, \overline{\Sigma}_{k,i})\,, \\ \overline{\mu}_{k,i} &= \mathbf{F}(\mu_{k-1,i}, \mathbf{u}_{k-1})\,, \\ \overline{\Sigma}_{k,i} &= \mathbf{A}_{k-1,i}\Sigma_{k-1,i}\mathbf{A}^T_{k-1,i} + \Sigma_s \\ \Longrightarrow \mathbf{A}_{k-1,i} &= \left.\frac{\partial \mathbf{F}}{\partial \mathbf{X}_{k-1,i}}\right|_{\overline{\mu}_{k-1,i},\mathbf{u}_{k-1}}\,, \end{aligned} \tag{5.8}$$

MEASUREMENT UPDATE

$$\begin{aligned} bel_{k,i} &= N(\mathbf{X}_{k,i}; \mu_{k,i}, \Sigma_{k,i})\,, \\ \mu_{k,i} &= \overline{\mu}_{k,i} + \mathbf{K}_{k,i}(\mathbf{D}_k - \mathbf{H}(\overline{\mu}_{k,i}, \mathbf{B}_i))\,, \\ \Sigma_{k,i} &= (\mathbf{I} - \mathbf{K}_{k,i}\mathbf{C}_{k,i})\overline{\Sigma}_{k,i} \\ \Longrightarrow \mathbf{K}_{k,i} &= \overline{\Sigma}_{k,i}\mathbf{C}^T_{k.i}(\mathbf{C}_{k,i}\overline{\Sigma}_{k,i}\mathbf{C}^T_{k,i} + \Sigma_m)^{-1} \end{aligned} \tag{5.9}$$

$$\Longrightarrow \mathbf{C}_{k,i} = \left.\frac{\partial \mathbf{H}}{\partial \mathbf{X}_{k,i}}\right|_{\overline{\mu}_{k,i}}\,. \tag{5.10}$$

However, before the model entropy H_k can be fully expressed, the data likelihood must be resolved in line 5 of Function 4.2:

$$\begin{aligned} p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) &= \int p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{X}_{k,i}, M_i) \cdot \overline{bel}_{k,i} d\mathbf{X}_{k,i} \\ &= \int N(\mathbf{D}_k; \mathbf{H}(\mathbf{X}_{k,i}, \mathbf{B}_i), \mathbf{\Sigma}_m) \cdot N(\mathbf{X}_{k,i}; \overline{\mu}_{k,i}, \overline{\Sigma}_{k,i}) d\mathbf{X}_{k,i}\,, \end{aligned} \tag{5.11}$$

where equation (5.8) has been substituted in place of $\overline{bel}_{k,i}$ and $p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{X}_{k,i}, M_i)$ has been replaced with $N(\mathbf{D}_k; \mathbf{H}(\mathbf{X}_{k,i}, \mathbf{B}_i), \mathbf{\Sigma}_m)$ from the measurement model of equation (5.7). Note that by making a linear approximation to $\mathbf{H}(\mathbf{X}_{k,i}, \mathbf{B}_i)$ as was done in equation (5.10), the data probability pdf of equation (5.11) can be analytically solved with application of Proposition 2.1:

$$p(\mathbf{D}_k|\mathbf{D}_{1:k-1}, \mathbf{u}_{1:k-1}, M_i) = N(\mathbf{D}_k; \mu_{D_k,i}, \Sigma_{D_k,i})\,, \tag{5.12}$$

True Model: M_2	M_0			M_1			M_3		
Pose $[x \quad y \quad z \quad \alpha \quad \beta \quad \gamma]$	$P^0_{\mathbf{b}_0}$	$\cdots$	$P^0_{\mathbf{b}_{N_0-1}}$	$P^1_{\mathbf{b}_0}$	$\cdots$	$P^1_{\mathbf{b}_{N_1-1}}$	$P^3_{\mathbf{b}_0}$	$\cdots$	$P^3_{\mathbf{b}_{N_3-1}}$
0.19 0.32 $\cdots$ 0.72	1.00	...	0.00	0.92	...	0.00	0.93	...	0.01
0.18 0.33 $\cdots$ 0.81	0.92	...	0.00	0.95	...	0.02	1.00	...	0.03
0.19 0.32 $\cdots$ 0.81	0.90	...	0.00	0.99	...	0.01	0.95	...	0.05
0.20 0.34 $\cdots$ 0.82	0.89	...	0.00	0.98	...	0.03	0.91	...	0.03
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0.21 0.32 $\cdots$ 1.09	0.00	...	0.00	0.00	...	0.00	0.00	...	0.00

Table 5.1: Probability lookup table generated by considering a true model M_i at various relative poses, and recording how every other model's database of features compares to the true model set.

where $\mu_{D_k,i}$ and $\Sigma_{D_k,i}$ are given by

$$\begin{aligned} \mu_{D_k,i} &= \mathbf{C}_{k,i}\overline{\mu}_{k,i}\,, \\ \mathbf{\Sigma}_{D_k,i} &= \mathbf{C}_{k,i}\overline{\Sigma}_{k,i}\mathbf{C}^T_{k,i} + \Sigma_m\,. \end{aligned}$$

Recall that the function *calcExpectedEntropy(·)* is responsible for determining the expected model entropy H^+_m associated with the potential future control action $\mathbf{u}^+_m$. Close inspection of Function 4.3 will show that a majority of the terms are no different than terms already solved for from determining the current model entropy H_k in the *calcCurrentEntropy(·)* function. The only term that requires further specification is the sampled future data measurement, $\tilde{\mathbf{D}}$, which is not as straightforward as may seem – how does one sample a set of 3D SIFT features, each with a 128-dimensional descriptor that determines the appropriate correspondence? In the case study of the previous chapter, the measurement model constructed had simplified away many of these details making the prediction of future data measurements possible from a detailed measurement model — which is not available in this particular experiment. To find a solution to this problem, Function 4.3 is examined in further detail.

Note that the purpose of generating future measurement data $\tilde{\mathbf{D}} \in \mathbb{R}^{(3\times n)\times 1}$ (for n future measurements) from model M_i according to $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}^+_m, M_i)$ is to consider the sampled data against all other models M_j according to $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}^+_m, M_j)$. In other words, if control action $\mathbf{u}^+_m$ is executed and $\tilde{\mathbf{D}}$ potentially observed (assuming model M_i is the true model), the utility of that action and the discriminability of the data when compared to the set of all other models $\{M_j\}$ is determined by evaluating the expression: $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}^+_m, M_j)$. This is a necessary step to Monte Carlo Sampling so that the model entropy associated with executing control action $\mathbf{u}^+_m$ can be accurately determined. Consider then the term $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}^+_m, M_j)$ (which was shown to have a normal distribution via application of Proposition 2.1):

$$p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}^+_m, M_j) = N(\tilde{\mathbf{D}}; \mu_{D_k,j}, \Sigma_{D_k,j}) \quad .$$

If each SIFT measurement, $\tilde{\mathbf{y}}_l \in \mathbb{R}^{3\times 1}$ from the sampled set $\tilde{\mathbf{D}} = [\tilde{\mathbf{y}}_1^T\,\tilde{\mathbf{y}}_2^T\,\cdots\,\tilde{\mathbf{y}}_n^T]^T$ is assumed mutually exclusive from all other features in the same set, then $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_j)$ becomes

$$\begin{aligned} p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_j) &= \prod_{l=1}^{n} N(\tilde{\mathbf{y}}_l; \mu_{y_l,j}, \Sigma_{y_l,j}) \\ &= \prod_{l=1}^{n} \frac{1}{(2\pi)^{(3/2)}|\Sigma_{y_l,j}|^{1/2}} exp(-\frac{1}{2}(\tilde{\mathbf{y}}_l - \mu_{y_l,j})^T \Sigma_{y_l,j}^{-1}(\tilde{\mathbf{y}}_l - \mu_{y_l,j}))\,. \end{aligned}$$

The mean and covariance of the l-th measurement are given by

$$\begin{aligned} \mu_{y_l,j} &= \mathbf{C}_{l,j}\overline{\mu}_{k,j} \\ \mathbf{\Sigma}_{y_l,j} &= \mathbf{C}_{l,j}\overline{\Sigma}_{k,j}\mathbf{C}_{l,j}^T + \Sigma_m \end{aligned}$$

where $\mathbf{C}_{l,j}$ corresponds to the linearized measurement model for the single measurement $\tilde{\mathbf{y}}_l$ of model M_j, $\overline{\mu}_{k,j}$ and $\overline{\Sigma}_{k,j}$ correspond to the prediction step of the Bayes Filter, and Σ_m is the covariance matrix associated with the noise of measurement $\tilde{\mathbf{y}}_l$. Note that the vector $[\tilde{\mathbf{y}} - \mu_{y_l,j}] \in \mathbb{R}^{3\times 1}$ is the residual of the sampled future measurement from model M_i and the linearized expected location of that matched feature in model M_j in 3D Euclidean space. Also, $\Sigma_{y_l,j}$ is defined as a sum of the uncertainty in object pose and measurement uncertainty. Suppose the following assumptions are now made

1. $[\tilde{\mathbf{y}}_l - \mu_{y_l,j}] \approx \bar{\epsilon}_0 \in \mathbb{R}^{3\times 1}$,
2. $|\Sigma_m| \gg |\mathbf{C}_{y_l,j}\overline{\Sigma}_{k,j}\mathbf{C}_{y_l,j}^T| \quad \Longrightarrow \quad \Sigma_{y_l,j} \approx \Sigma_m = \Sigma_0 \in \mathbb{R}^{3\times 3}$.

The first assumption claims that the "residuals" between all sampled measurements of model M_i and their expected linearized matched locations in model M_j are always the same value. Obviously that is not the case but for the most part the sampled measurements and expected locations are on the same order and in situations when feature distances in the object reference frame are small relative to feature distances in the camera reference frame, this assumption can be valid. The second assumption claims that the sensor measurement uncertainty is dominant, and approximated by a constant matrix value, which can be argued as a valid claim provided the object is far enough from the camera origin. With these assumptions then made, the data probability $p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_j)$ can be reduced to

$$p(\tilde{\mathbf{D}}|\mathbf{D}_{1:k}, \mathbf{u}_{1:k-1}, \mathbf{u}_m^+, M_j) \approx \frac{1}{(2\pi)^{3n/2}|\Sigma_0|^{1/2}} exp(-\frac{n}{2}\bar{\epsilon}_0\Sigma_0^{-1}\bar{\epsilon}_0^T)\,, \tag{5.13}$$

where again n is the number of features matched between the data $\tilde{\mathbf{D}}$ sampled from M_i and the set of database features in M_j. With this approximation, expected entropy calculations can thus be carried out without requiring specific generated future data. The only information needed is the

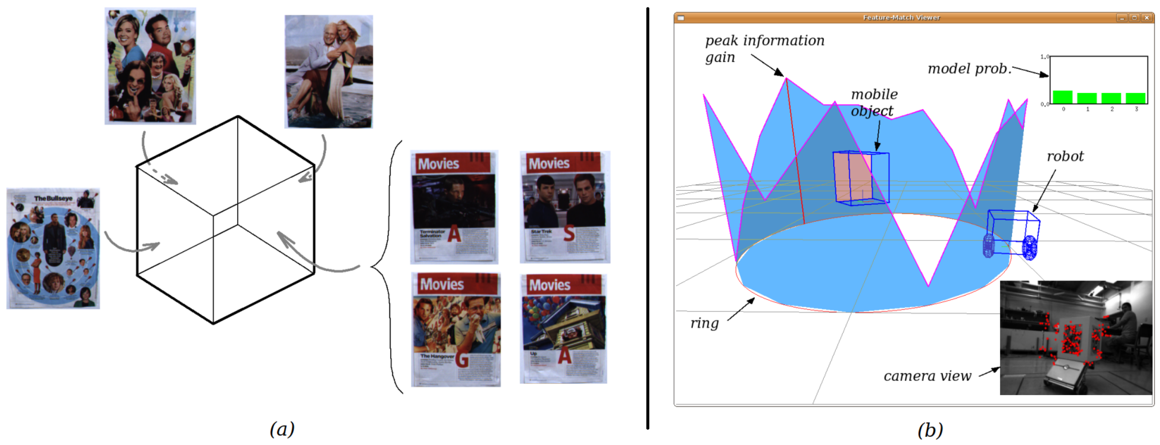


Figure 5.3: (a) Four similar models of a box were used in Experiment 2. Each box model shared the same three faces while the fourth face was different for each model. The faces themselves were described by magazine cutouts. (b) A screenshot of the algorithm's visual interface. Shown is the information gain for various future robot locations (limited to a circle passing through robot's current pose). The optimal action moves to the peak of the information gain metric, which brings the discriminating face (shown in pink) into view.

number of measurements, n, matched between the database of features in model M_j and a potential data set $\tilde{\mathbf{D}}$ of features in model M_i expected to be observed based on some potential action $\mathbf{u}_m^+$. Furthermore, this approximation does not involve any matrix inversions and reduces computational complexity greatly.

As a result of the above approximations, probability lookup tables can be used. Since the feature correspondences need to be considered for each true model against every other model, the probability lookup tables take on the form shown in Table 5.1.

The lookup table is generated by considering a true model M_i at various relative poses, and recording how every other M_j model's database features matches to the true model set. The probability values, $P^j_{\mathbf{b}_l}$ represent the probability of the l-th database feature of the j-th model finding a match to some feature in the i-th model at the listed relative pose between robot and object. Shown in the table are values used for a true model $M_i = M_2$, though note that probability lookup tables were generated for all models considered as the true model and kept in a database of files loaded at run time. Thus the variable n in equation (5.13) is determined by using the lookup table for true model M_i and counting the number of positively sampled features from the probability columns associated with model M_j.

5.1.4 Results

The experimental system (see Figure 5.1(b)) used a PointGrey BumbleBee2 stereo color camera (downsampled to 320×240 resolution) mounted on an Evolution Robotics ER-1 mobile robot. Wheel odometry was used as a crude measure of robot pose estimation. Computations were performed on

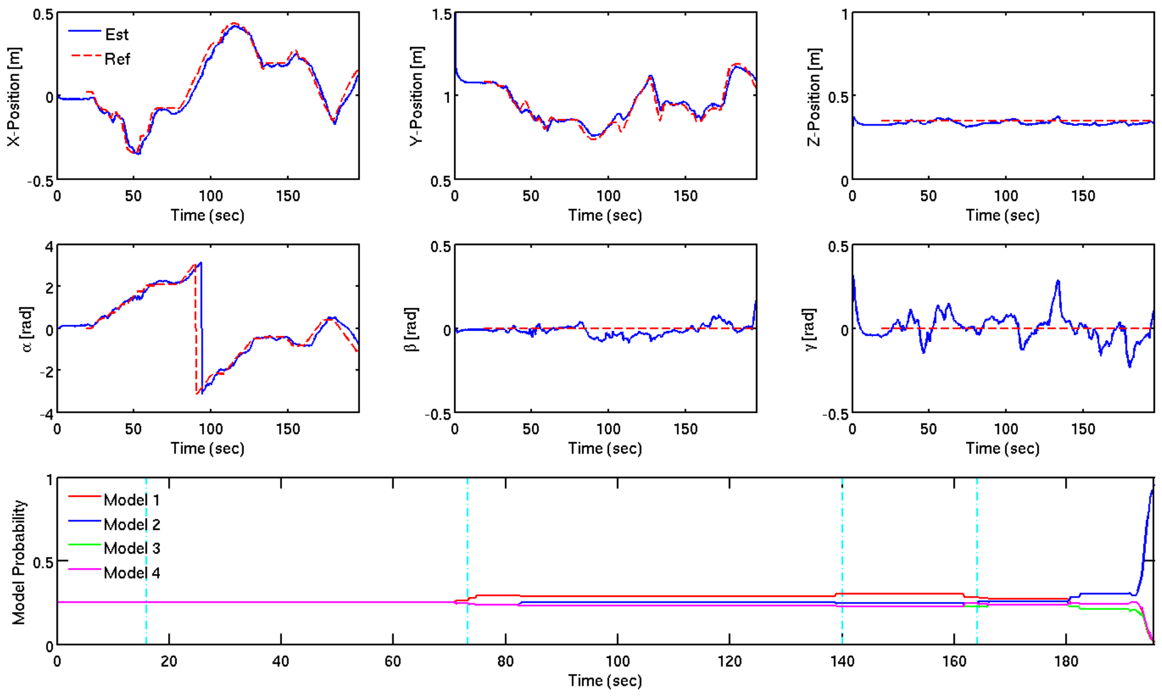


Figure 5.4: The position estimates of the object (as defined in a global reference frame) are shown in the top row and the orientation estimates are shown in the middle row. The robot estimated pose using the proposed algorithm is shown in the solid-blue line and the reference pose is shown in the dotted-red line. The last row shows the model probability estimates for all models, plotted against time. The vertical dotted lines indicate instances of when the information gain calculations were executed for sensor planning.

a laptop (Intel Pentium(R) M 1.86 GHz processor) running Linux. The algorithm was written in C/C++.

For the mobile object, the database consisted of 4 models of a box attached to a moveable base. Each lateral side of the box was marked with a distinct pattern (a magazine cutout), and it was decided that all 4 models would share 3 identical faces, with only the last face being different for each model (see Figure 5.3(a)). The database of features for each model was generated during an off-line *training phase* in which the robot was allowed to learn each object from eight varying *viewpoints*, all kept at the same nominal distance from the object. Motion of the object was remotely controlled by a human operator in such a manner so as to prevent the discriminating model face from being observed by the robot. This strategy required repeated sensing and planning operations on the part of the robot until the model probability of one model peaked to near 100% certainty.

The top two rows of Figure 5.4 show the position parameters $(x, y, z, \alpha, \beta, \gamma)$ estimated by the SPPEMI algorithm, plotted as a function of time. Though poses were estimated for all database models, only the parameters for the true model are shown. Plotted against the estimated parameters is the model object reference pose as determined by wheel odometery. Considering that the algorithm assumes a simple random walk model for the object, the results show that it does a fairly good job

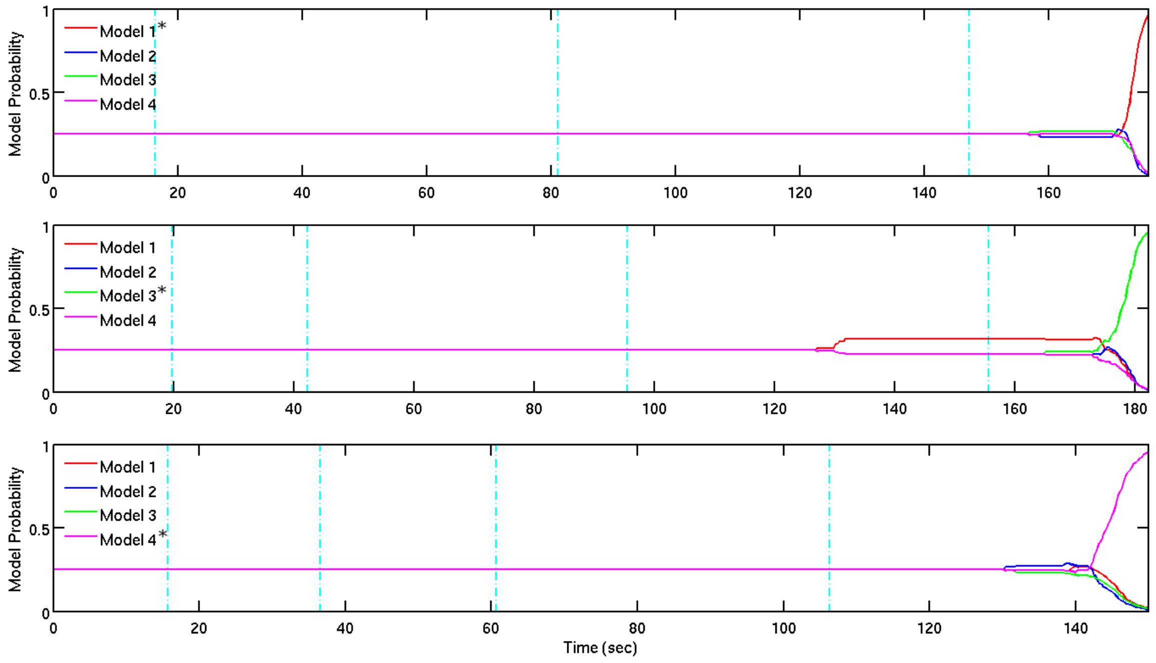


Figure 5.5: Other trials were also considered for different models. Shown are those model probability estimates for each trial plotted against time. The true models are indicated with the asterisk. As can be seen, the algorithm accurately identifies the true model in each case. The vertical dotted lines indicate instances of when information gain calculations were executed for sensor planning.

of tracking. Note however the noise associated with the roll angle estimate in the far right plot of the second row – a variance of roughly $0.3 rad$. This is largely attributed to poor stereo projection of certain features and also a small set of feature mismatches.

The bottom row of Figure 5.4 shows the calculated model probabilities for each model during the same trial. Note that up until $t = 180\,s$, the model probabilities are roughly equal at 0.25. This is expected since all models share the same three faces, and until the discriminating model face is observed (which happens at $t = 180\,s$), each model is equally likely to be the unknown object. Note that once that face is observed, the model probability of the correct model increases rapidly to 1.0 while the other probabilities decrease to 0.0. The dotted vertical lines correspond to the time instances when the information gain is calculated and the optimal action chosen. Figure 5.3(b) illustrates a screenshot of the system's visual interface taken at such an updating event. It is important to note that the information gain calculations are not performed during each timestep, but are executed once the robot has completed the prior sensing action.

Subsequent experiment trials were carried out for the other objects in the database. Figure 5.5 shows the results of those trials (with the pose estimate plot comparisons omitted for brevity), with the true models indicated by the asterisk. As can be seen in the figure, the algorithm accurately identifies the true models in each trial run and shows consistent behavior across all trials. This experiment validates the SPPEMI algorithm in a more complex, noisier environment. It also shows

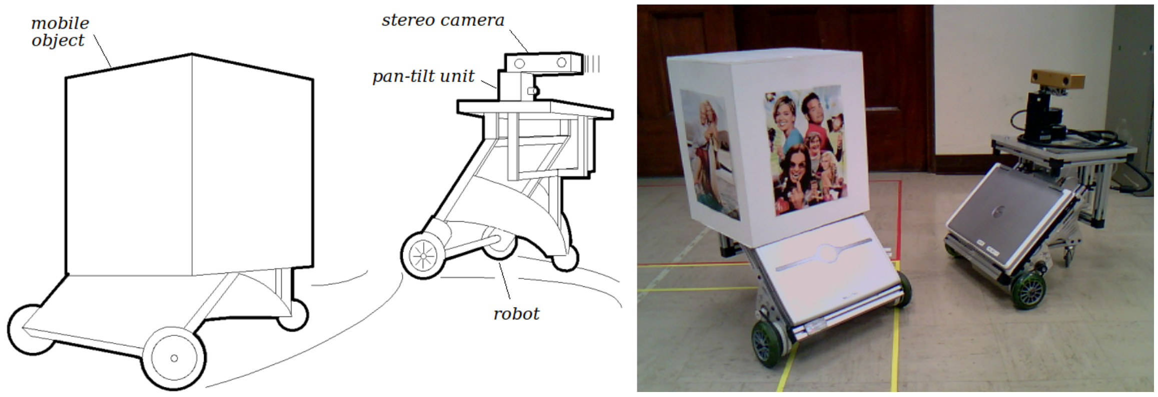


Figure 5.6: Experimental setup for Experiment 3. Both the agent and the object are free to move about the $x-y$ plane. The stereo camera head on the mobile agent is also controlled to swivel about the pan axis.

an extension of the work done in chapter 3 to larger scale objects. However, the constraints involved in this experiment (i.e., the robot motion being confined to a ring as well as the fixed camera orientation relative to the robot frame) are limiting and not truly illustrative of the robustness of the proposed algorithm. An implementation of the SPPEMI algorithm that removes the constraints of this experiment is discussed next.

5.2 Experiment 2: Mobile Agent with Mobile Object

In this experiment, the setup is very similar to Experiment 1 though with some slight differences. The mobile agent is again tasked with identifying a given object model, where the models in this experiment are identical to those used in Experiment 1. The robot considered is again a two-wheeled non-holonomic robot, though this time free to move about in the x-y plane, with its motions determined by a combination of the output of the SPPEMI algorithm and a point-to-point controller designed to keep the robot within a reasonable vicinity of the object. The agent is also equipped with a stereo camera head mounted on a pan-tilt-unit allowing the camera head to rotate independently of the robot base orientation. The purpose of the pan-tilt unit controlled stereo head is to simplify the motion planning associated with the nonholonomic constraints of the robot base and to allow the robot to freely move in the x-y plane, unconstrained by the field of view of the camera. As was the case in the previous experiment, the motions of the object are determined by operator control. The schematic in Fig. 5.6 illustrates the proposed setup for this experiment. The chosen features are again SIFT features augmented with 3D stereo data and object identification is done by comparing known model features in a database to current features in the image.

5.2.1 Motion Model

The state of the system is the same as was in Experiment 1, and the motion model is also similar. In this experiment, however, the transform $G_{R_k C_k}$ in equation (5.1) changes from timestep to timestep because of the motion of the pan-tilt unit (PTU) and thus cannot be assumed constant. The kinematic relations of equation (5.1) are then re-written as

$$G_{C_k,O_k} = G^{-1}_{R_k C_k} \cdot G^{-1}_{R_{k-1} R_k} \cdot G_{R_{k-1} C_{k-1}} \cdot G_{C_{k-1},O_{k-1}} \cdot G_{O_{k-1} O_k} \, , \tag{5.14}$$

where $G_{R_{k-1} C_{k-1}}$ and $G_{R_k C_k}$ define the camera to robot frame transform at timestep $k-1$ and k. These transforms can be defined by knowing the pan position of the PTU at both timesteps. The pan position at timestep $k-1$ will come from the pan angle position read directly from the unit (ϕ_k) while the pan position at timestep k will come from the pan slew rate at timestep $k-1$, ($\dot{\phi}_{k-1}\Delta t$). In this experiment, $G_{R_{k-1} C_{k-1}}$ and $G_{R_k C_k}$ are found to have the following forms:

$$G_{R_{k-1} C_{k-1}} = \begin{bmatrix} sin(\phi_{k-1}) & 0 & cos(\phi_{k-1}) & -b \\ -cos(\phi_{k-1}) & 0 & sin(\phi_{k-1}) & 0 \\ 0 & -1 & 0 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} , \tag{5.15}$$

$$G_{R_k C_k} = \begin{bmatrix} sin(\dot{\phi}_{k-1}\Delta t) & 0 & cos(\dot{\phi}_{k-1}\Delta t) & -b \\ -cos(\dot{\phi}_{k-1}\Delta t) & 0 & sin(\dot{\phi}_{k-1}\Delta t) & 0 \\ 0 & -1 & 0 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} , \tag{5.16}$$

where b and l refer to the displacement offset in x and z between the robot reference frame and origin of the camera as shown in Figure 5.7.

Because the PTU has its own set of control commands that can be governed by the robot, there is added complexity to the overall system if the pan controls are considered in the SPPEMI algorithm in addition to the robot odometry controls. The approach taken in this experiment is to reduce complexity by treating the control loop governing the PTU independently from the SPPEMI algorithm. That is, the PTU control loop will operate in a separate thread from the SPPEMI algorithm and in such a manner that ensures the object is in the camera field of view. (Section 5.2.3.1 will discuss the PTU control loop in further detail.) Thus the only controls to the overall system are controls to the robot motion:

$$\mathbf{u}_k = \begin{bmatrix} u_s & u_\omega \end{bmatrix}^T_k \in \mathbb{R}^2 \, ,$$

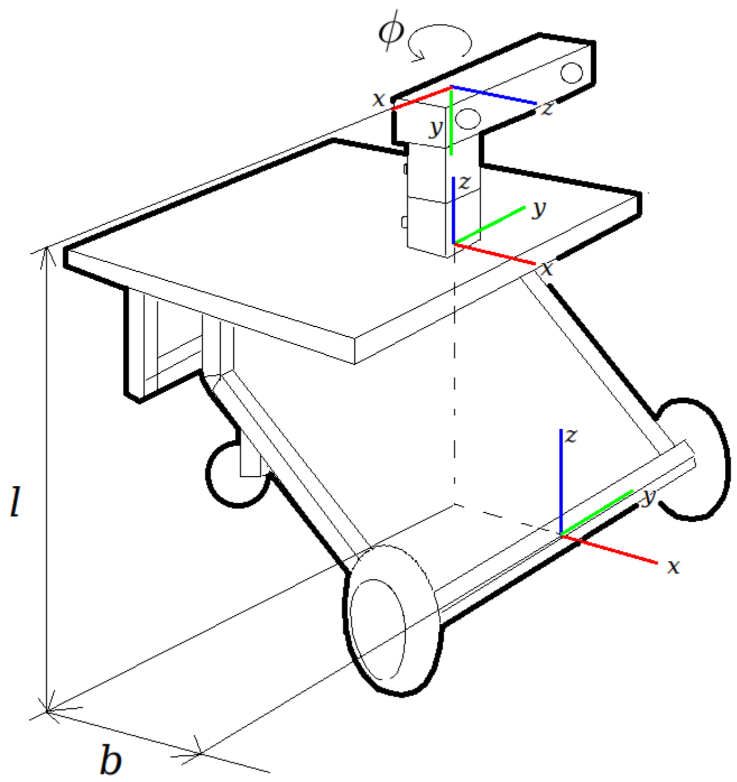


Figure 5.7: The camera to robot transform in this experiment was determined by considering the offset between the PTU base and the robot base in x (distance b) and z (distance l), as well as the pan angle of the camera, ϕ.

where again u_s and u_ω correspond to the wheel speed and turnrate of the robot.

Aside from the expressions for $G_{R_kC_k}$ and $G_{R_{k-1}C_{k-1}}$, the motion model for the system in this experiment thus follows the same approach as in Experiment 2, with the transforms $G_{R_{k-1}R_k}$ and $G_{O_{k-1}O_k}$ being the same:

$$\begin{aligned}\mathbf{X}_{k,i} &= \left[\begin{array}{c}\left[\begin{array}{ccc} f_x & f_y & f_z \end{array}\right]^T \\ \left[\begin{array}{ccc} f_\alpha & f_\beta & f_\gamma \end{array}\right]^T\end{array}\right]_{k-1} + \eta \\ &= \mathbf{F}(\mathbf{X}_{k-1,i}, \mathbf{u}_{k-1}) + \eta\,, \end{aligned} \tag{5.17}$$

however $\mathbf{F}(\mathbf{X}_{k-1}, \mathbf{u}_{k-1})$ takes on a different form from equation (5.6) due to the changes in $G_{R_kC_k}$ and $G_{R_{k-1}C_{k-1}}$. The exact expression can be solved for by substitution of equations (5.4), (5.15), and (5.16) into equation (5.14) and extracting out the relevant Euler parameters (see Appendix B for the mathematical details). The terms f_x, f_y, f_z are thus given by

$$\begin{aligned}
f_x &= c(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot x + s(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot z + \\
&\quad - b \cdot s(\dot{\phi}\Delta t + \delta_\theta) + -s(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) + \\
&\quad c(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) + b \cdot s(\dot{\phi}\Delta t)\,, \\
f_y &= y\,, \\
f_z &= -s(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot x + c(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot z + \\
&\quad - b \cdot c(\dot{\phi}\Delta t + \delta_\theta) + -c(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) + \\
&\quad - s(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g))\,,
\end{aligned}$$

and the terms f_α, f_β, and f_γ are given by

$$\begin{aligned}
f_\alpha &= tan^{-1}\left(\frac{s\alpha s\beta}{c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha c\beta - s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\beta}\right)\,, \\
f_\beta &= -sin^{-1}(-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha c\beta - c(\dot{\phi}\Delta t + \delta_\theta - \phi)s\beta)\,, \\
f_\gamma &= tan^{-1}\left(\frac{-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha s\beta s\gamma + s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\alpha c\gamma + c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\beta s\gamma}{-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha s\beta c\gamma - s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\alpha s\gamma + c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\beta c\gamma}\right)\,.
\end{aligned}$$

5.2.2 Measurement Model

The measurement model for this experiment is identical to the model developed for Experiment 2 (see section 5.1.2). 3D SIFT features are used with a fixed subwindow of the image to increase computational speed and feature correspondence is achieved using Lowe's Best-Bin-First Search algorithm that utilizes a *k-d* tree. The measurement model is repeated again here for convenience:

$$\begin{aligned}
\mathbf{D}_k &= \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(0)} \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(1)} \\ \vdots \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}^i_{\mathbf{J}(n_K)} \end{bmatrix} + \xi\,, \\
&= \mathbf{H}(\mathbf{X}_{k,i}, \mathbf{B}_i) + \xi
\end{aligned}$$

where the terms $\mathbf{D}_k$, $\mathbf{y}_j$, $\mathbf{b}^i_{\mathbf{J}(j)}$, and $\mathbf{J}_i(j)$ are the same as was defined in section 3.2.2.

5.2.3 Applying the Algorithm

Application of the SPPEMI algorithm to this experiment is similar to Experiment 2 in that information gain calculations are computed only after the mobile robot has reached the location determined by the optimal control action, $\mathbf{u}^*$. Lookup tables are also similarly used for generating future data using the same approximation of equation (5.13). However, the application of the SPPEMI algorithm to this experiment is slightly more involved because of the control loop governing the pan-tilt unit, the unconstrained motion of the object, and the unconstrained motion of the robot. Each of these factors will be considered in full detail, in relation to how implementation of the SPPEMI algorithm is ultimately achieved.

5.2.3.1 Pan-Tilt-Unit Control Loop

As was mentioned in section 5.2.1, the complexity of the SPPEMI algorithm in the overall system is reduced if the PTU control is implemented separately from the robot motion control. The control loop of the PTU revolves around two states: a "tracking state" and a "scanning state." The tracking state is concerned with keeping the object origin in the field of view of the camera at all times once detected. However, because the object model ID may not be known with 100% certainty at initial detection, the object's exact origin location is a vague concept: an Extended Kalman Filter is running for each of the M models in the database and there are essentially M potential origin locations. As such, the average origin of all models (weighted by model probability) is chosen as the origin and the "tracking state" aims to keep the origin in the field of view of the camera using a simple state-feedback control law. That is, if the weighted origin, $\mathbf{P}_{obj} \in \mathbb{R}^3$, is found to be $\mathbf{P}_{obj} = [x_{obj}\, y_{obj}\, z_{obj}]$, then the projection of the origin on the image plane, $\mathbf{p}_{obj} \in \mathbb{R}^2$, is defined as

$$\mathbf{p}_{obj} = \begin{bmatrix} x_{img} \\ y_{img} \end{bmatrix} = \begin{bmatrix} \dfrac{f \cdot x_{obj}}{z_{obj}} + c_x \\ \dfrac{f \cdot y_{obj}}{z_{obj}} + c_y \end{bmatrix},$$

where f is the focal length of the camera and (c_x, c_y) the image center. A simple proportional feedback law is then used to control the pan-angle slew rate:

$$u_{pan} = -K \cdot \delta_{pan} = -K(c_x - x_{img}) = -K\frac{f \cdot x_{obj}}{z_{obj}},$$

for constant gain $K > 0$ and u_{pan} in units of $\frac{rad}{sec}$. A similar method can be used for the PTU tilt, though tilt control was found to be unnecessary in this experiment.

The "scanning state" of the PTU is concerned with moving the camera head in a scanning pattern so as to cover regions of the environment outside the original field of view. This state is typically entered during the initial startup of the algorithm when the robot is *searching* for any known object model or whenever the robot has lost existing tracks of object models. In situations relating to the latter, a signal is sent to the robot odometry to pause until the track is re-established. Depending on the re-initialized state of the object, a new iteration of the SPPEMI algorithm may be computed for a new optimal control action, or the control action originally sought before the track was lost is

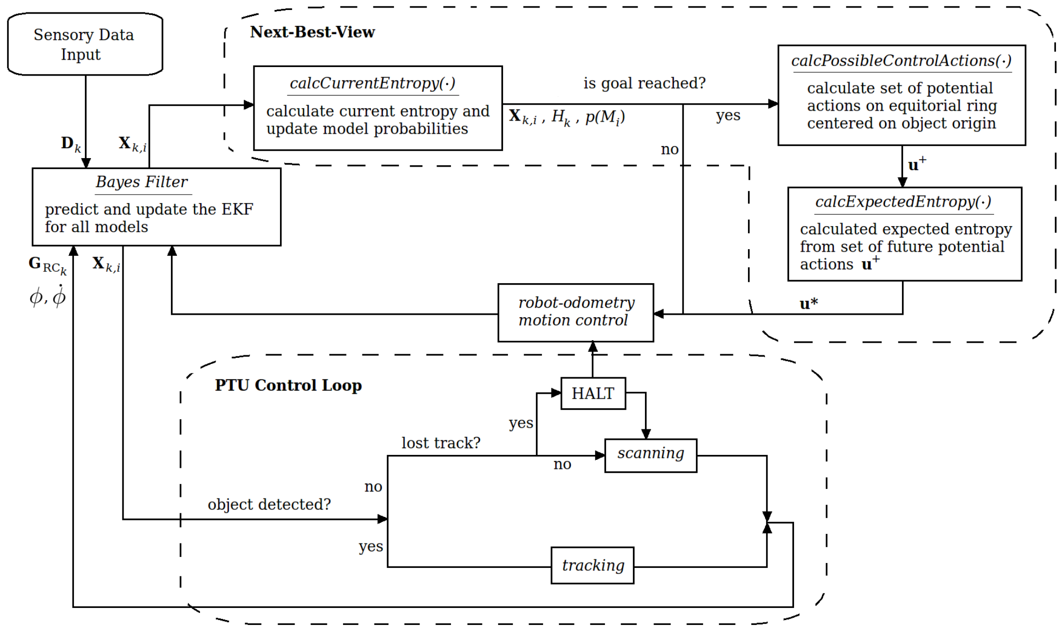

Sensory Data Input
Next-Best-View
calcCurrentEntropy(·)
calculate current entropy and update model probabilities
is goal reached?
$X_{k,i}$, H_k, $p(M_i)$
yes
no
calcPossibleControlActions(·)
calculate set of potential actions on equitorial ring centered on object origin
u^+
calcExpectedEntropy(·)
calculated expected entropy from set of future potential actions u^+
u^*
D_k
$X_{k,i}$
Bayes Filter
predict and update the EKF for all models
G_{RC_k}
$\phi, \dot{\phi}$
$X_{k,i}$
robot-odometry motion control
PTU Control Loop
HALT
yes
lost track?
no
scanning
no
object detected?
yes
tracking

Figure 5.8: The shown block diagram illustrates the specific implementation of the SPPEMI Algorithm in this experiment. Note the information gain calculations are limited to being computed when the system state has reached the state determined by the optimal control action $\overline{\mathbf{u}}^*$, which is a modified control action output from the point-to-point controller. Note also the PTU control block which acts as a separate thread.

then resumed.

5.2.3.2 Robot Motion Control

Because the object itself is mobile and the robot free to move about in the x-y plane, the set of possible control actions to consider in the *calcPossibleControlActions*$(\cdot)$ function is vast. Furthermore, the object itself may be in a position far from the robot where simple motion in a circular path (such as in Experiment 2) may not be optimal. To circumvent these issues, an approach can be taken which constructs the current experiment in a framework similar to Experiment 1, allowing all the methodologies previously described to be implemented in the same manner.

Consider the situation where the mobile object's weighted origin is found to be at $\mathbf{P}_{obj} \in \mathbb{R}^3$ and the mobile robot is deciding where to move next based on the information gain metric of the SPPEMI algorithm. Given that the object may be at some far distance (or even a close distance), the current robot pose can be projected onto a ring of radius R_{radius} centered on $\mathbf{P}_{obj}$, where R_{radius} is of the nominal distance between the object and the robot set during the training phase when the object was first learned. In so doing, a set of discrete control actions along the perimeter of the ring can then be considered and the function *calcPossibleControlActions*$(\cdot)$ computed in a manner analogous to Experiment 1.

Once the optimal control action is determined from the SPPEMI algorithm, note that $\mathbf{u}^*$ presumes a motion along a circular path of the hypothetical ring which is inaccurate since: (1) the *projection* of the mobile robot pose was used to generate $\mathbf{u}^*$ and not the true robot pose which may be some further distance away; (2) the motions of the mobile robot are no longer confined to circular paths on the ring and more optimal paths may exist that achieve the same outcome. As such, the predicted final robot pose on the ring is extracted from the optimal control action $\mathbf{u}^*$ and fed into a motion controller. The motion controller determines a new control action (or set of), $\bar{\mathbf{u}}^*$, that is equivalent to the desired location of $\mathbf{u}^*$ yet does not confine motion to a ring and will take more direct paths.

Figure 5.8 shows a block diagram of the entire system, which is similar to the block diagram of Fig. 4.7, yet includes the PTU control loop and robot motion controller. Note in the PTU control loop the HALT state is directly responsible for pausing the motion controller should the object track be lost. Also note that the SPPEMI algorithm loop remains virtually unchanged, with the exception that the state $\mathbf{X}_{k,i}$ is replaced with the projected state on the ring of radius R_{ring}. As such, the probability lookup tables generated in Experiment 2 can still be used and the information gain calculations carried out in the same fashion. The results of this experiment are presented in the next section.

5.2.4 Results

In this experiment, the database of 4 box models that were used in Experiment 1 were also used (see Figure 5.3) – with all four models being nearly identical except for one discriminating face. The mobile agent used was again an Evolution Robotics ER-1 two-wheeled robot with onboard wheel odometry and the mobile object (which was operator controlled) was another ER-1 robot (affixed with the box model) and equipped with wheel odometry that was used as a measure of ground truth in analyzing the pose estimates from the experiment. As in the first and second experiments, the stereo camera consisted of a PointGrey BumbleBee2 stereo color camera with a native resolution of 640×480, downsampled to 320×240. The pan-tilt unit was a Directed Perception[2] PTU-D46-17 pan-tilt unit with an accuracy of 0.0514° in both pan and tilt positions. The computing platform was a laptop running Linux with an Intel Core2 Duo 2.40 GHz processor. The algorithm was written in C/C++ using the Intel OpenCV library.

Model 2 was used as the true model for the mobile object, with remote operation of the object controlled by an external operator and in such a manner that prevented the discriminating face from being observed too quickly. Recall the purpose of this strategy was to require the mobile robot to recursively sense and plan until the model probability of one model had peaked to near 100% certainty. Unlike Experiment 1, however, the initial object location was placed arbitrarily in the laboratory and no information of its initial location was provided to the mobile agent. The purpose

[2]www.directedperception.com

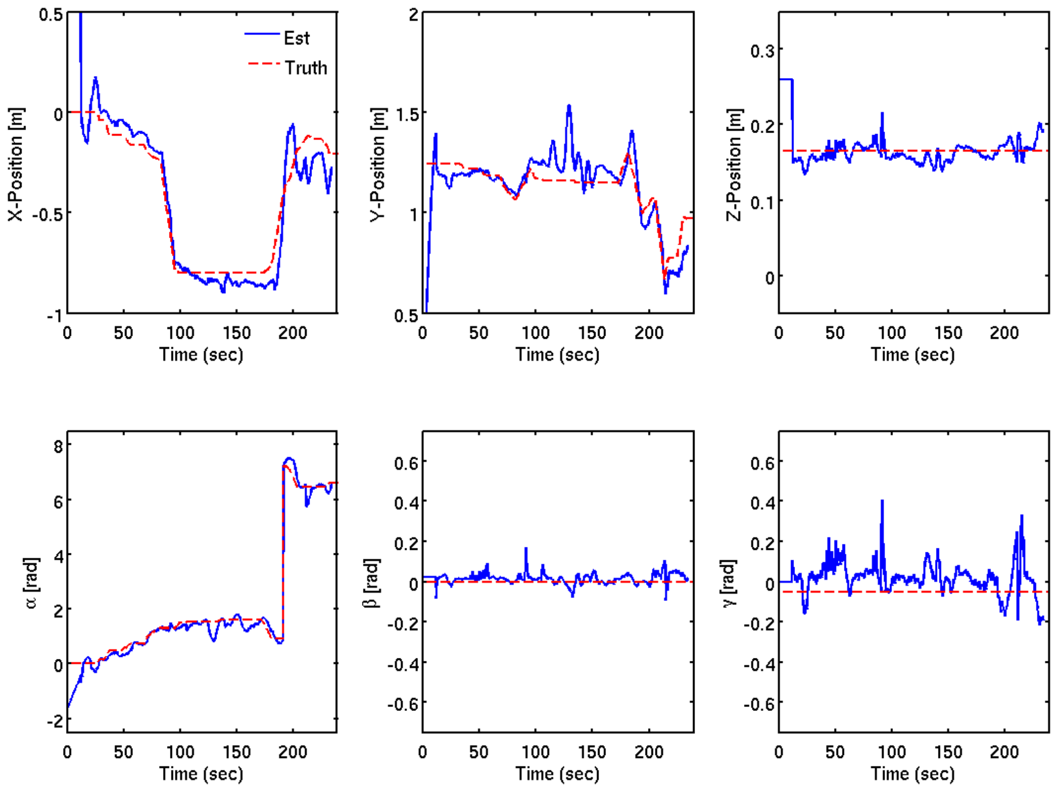


Figure 5.9: The position estimates of the object (as defined in the global reference frame) are shown in the top row and the orientation estimates are shown in the bottom row. The robot estimated pose using the SPEMMI algorithm is shown in the solid-blue line and the ground-truth pose is shown in the dotted-red line. The ground-truth pose was determined by the wheel odometry on the mobile object.

of this was to force the PTU control loop into the "scanning mode" and to test the robustness of the overall system to detect and initialize a pose for the object.

Figure 5.9 shows the pose estimation results of this experiment, where the translational pose estimates are shown in the top row of plots and the orientation estimates on the bottom row of plots. While multiple pose estimates exist for all potential models in the database, only the pose estimate for the true model is shown. Plotted against the pose estimate for reference is the mobile object's pose estimation from the wheel odometer. Overall the system does a fairly good job of estimating the pose of the model object, though with some spikes in uncertainty in the roll and pitch orientation estimates which can be attributed mainly to visual feature mismatches. Note also that there are some displacement offsets in x at around $t = 150s$ and $t = 220s$ and similarly in y at $t = 100s$ and $t = 220s$. Those offsets, while eventually rectified, correspond to the periods of "tracking" states of the PTU control loop. While efforts were made to keep the timestamps between images and pan angles/slew-rates synchronized, there was a slight delay that was unaccounted for which led to some images being tagged with unsynchronized pan angles/slew-rates. Nonetheless the EKF was still able to adjust to the more accurate state estimates.

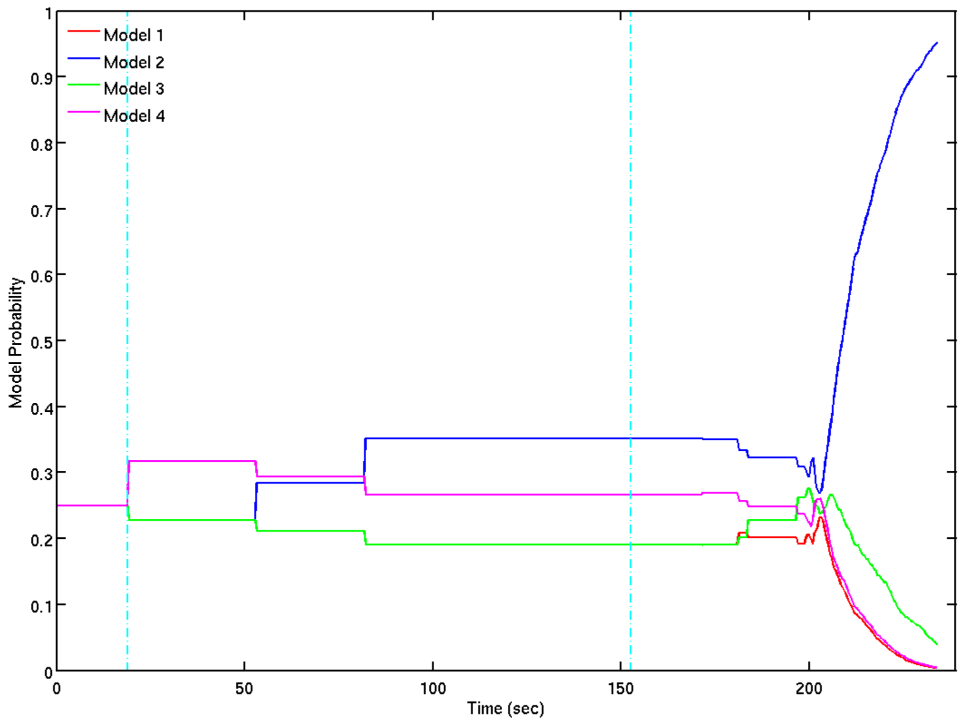


Figure 5.10: The model probabilities plotted against time for this experiment. Model 2 was correctly identified as the true model, despite an initial growing confidence in Model 4.

In Figure 5.10, the model probabilities are plotted against time to indicate the history of model confidence throughout the experiment. Despite some growing confidence in Model 4 initially, the system accurately determines the true model of the object once the discerning face of the object is seen which occurs around $t = 205s$ when the final sensor planning action has been executed. Figure 5.11 shows a screenshot of the visual interface developed for the implemented system. Note the equatorial ring is centered on the object with the information gain plotted at various locations along the ring. The peak information gain is denoted by the vertical yellow line, and is marked as the future goal location.

This experiment validates the SPPEMI algorithm in a more complex, noisier environment and shows the extensions of the SPPEMI algorithm to mobile robots in dynamic environments with real-time processing capabilities.

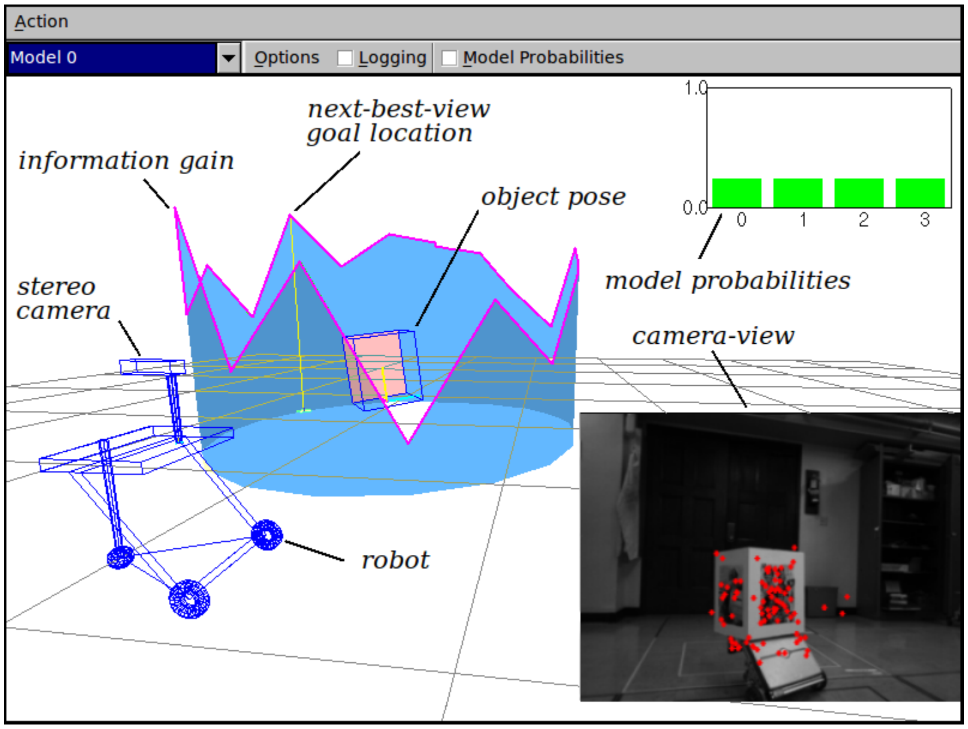


Figure 5.11: An annotated screenshot of the visual interface developed for the SPPEMI algorithm running in real-time. Note the generated information gain is plotted at various locations along the perimeter of the ring using the projected location of the robot onto the ring, while the robot itself is not located on the ring.

Chapter 6

Sensor Planning for 3D Object Search

The previous two chapters considered the problem of sensor planning for object identification and pose estimation with the underlying assumption that the object being tracked and identified could be easily found (either directly in the field of view of the mobile robot or within a reachable field of view by appropriate control of a pannable unit on which the camera is mounted). However, such an assumption is not always necessarily true and touches on a growing area of research known as *object search*. This chapter considers the problem of 3D object search with full 6D pose estimation using stereo vision in a probabilistic framework.

6.1 3D Object Search Context and Contribution

As mentioned in chapter 1, much of the prior work in this field has focused on developing a systematic approach to the sensor planning problem by enumerating all possible sensing actions and quantifying the utility of each action via optimization of constructed cost functions ([56], [46], [47]). While these methods have traditionally not considered the coupled problem of object pose estimation (once the object is found), recent works have touched on it ([13], [14], [27]). With advances made for applications to servicing robots, the problem is generally divided into a global (coarse) search and a local (refined) search approach, where the local search attempts to solve the problem of pose estimation for grasping. However, a large number of sensors is typically required to implement most current methods and the pose estimation in the local search is limited to simple planar objects in well-defined orientations. In addition, existing methods lack a formal Bayesian framework in their implementation. The contributions of this chapter will show that accurate 6D pose estimation can be realized using a single stereo camera mounted on a pan-tilt unit (PTU) of a mobile robot. The details of this chapter will present several useful features in a common framework: a two-scale search strategy, a grid-based probability map that governs the search process, a recursive Bayesian map updating process, 6D pose estimation of the found object, and a simple integration of obstacle avoidance into the search planning method.

6.2 Framework

Consider a (possibly nonholonomic) mobile robot equipped with a stereo camera mounted on top of a pan-tilt unit (see Figure 6.1(a)). Let the robot localization problem be resolved via one of many available methods (e.g., vision-based SLAM, onboard odometry, indoor GPS, etc.). While the position of the object to be found is not known to the robot, assume that the object's position is stationary throughout the search process. The object is assumed to be learned during a *training phase* in which various viewpoints of the object are captured in stereo. Registered features are recorded in the object reference frame along with a reference image of the object taken at each viewpoint. The resultant set of recorded features and images constitute a feature database specific to the object being searched (identical to the methodology presented in chapter 3), thus contributing to a dictionary of known objects in the robot's memory. This facilitates the setup for object search by allowing the user to simply specify which object in the dictionary to search for. Finally, suppose that a known object is placed somewhere in the workspace. The question can now be asked: how does the mobile robot search for the object and place its position in the global environment once found?

6.2.1 Probability Map

While the methods of previous approaches to the 3D object search problem have lacked in a formal Bayesian analysis, the work of [6] presented a recursive Bayesian updating schema for a grid-based probability map applied to the specific problem of object search. While their presented results were limited to simulation, the approach presented here borrows from that framework and applies it to a real robotic platform.

Assume that the workspace can be divided into cells whose coordinates are known in some global coordinate frame. Let $c_{i,j}$ represent the $(i,j)^{th}$ cell of the discretized search space, $\mathcal{M}$, and let $y_{i,j}^k \in \{0,1\}$ be a stochastic binary variable indicating a positive or negative detection of the object in cell $c_{i,j}$ at timestep k. Let $\mathbf{Y}_{1:k}$ denote the set of binary measurements from timestep 1 up to and including timestep k: $\mathbf{Y}_{1:k} = [y^1\, y^2\, \cdots\, y^k]$. Let $h_{i,j}^k \in \{0,1\}$ define a hypothesis of object existence in cell $c_{i,j}$ at timestep k, such that $h_{i,j}^k = 1$ is the hypothesis that the object exists in $c_{i,j}$ at time k and $h_{i,j}^k = 0$ the hypothesis that it does not.

Suppose that a known object is placed somewhere in the workspace, but its location is unknown to the searching robot. The probability (or belief) that the object resides in $c_{i,j}$ at time k given the binary measurements, $\mathbf{Y}_{1:k}$, is $P(h_{i,j}^k = 1|\mathbf{Y}_{1:k})$. Applying Bayes' Rule, the cell probability can be expanded as

$$P(h_{i,j}^k = 1|\mathbf{Y}_{1:k}) = \frac{P(y_{u,v}^k|h_{i,j}^k = 1, \mathbf{Y}_{1:k-1})P(h_{i,j}^k = 1|\mathbf{Y}_{1:k-1})}{P(y_{u,v}^k|\mathbf{Y}_{1:k})}\,, \tag{6.1}$$

where $y^k_{u,v}$ indicates the measurement at time k was of the $(u,v)^{th}$ cell which is not necessarily the same as $c_{i,j}$.

The first term of the numerator defines the sensor detection model. Many different forms of detection models exist and vary depending on the type of sensor and object recognition algorithm used. The detection model of [6] is appropriate for object detection on a grid-based map and is thus used when updating the probability map after a global or local search:

$$P(y^k_{u,v}|h^k_{i,j}=1,\mathbf{Y}_{1:k-1}) = \begin{cases} P(y^k_{u,v}=0|h^k_{i,j}=1,\mathbf{Y}_{1:k-1}) = \beta & (u,v=i,j) \\ P(y^k_{u,v}=1|h^k_{i,j}=1,\mathbf{Y}_{1:k-1}) = 1-\beta & (u,v=i,j) \\ P(y^k_{u,v}=0|h^k_{i,j}=1,\mathbf{Y}_{1:k-1}) = 1-\alpha & (u,v\neq i,j) \\ P(y^k_{u,v}=1|h^k_{i,j}=1,\mathbf{Y}_{1:k-1}) = \alpha & (u,v\neq i,j) \end{cases}, \quad (6.2)$$

where α and β represent the detection error probabilities for false alarms and missed detections, respectively, and are dependent on the sensor quality and the recognition modality (e.g., SIFT, support vector machine, color histogram, etc.).

Since the object is assumed stationary, the second term of the numerator can be simplified: $P(h^k_{i,j}=1|\mathbf{Y}_{1:k-1}) = P(h^{k-1}_{i,j}=1|\mathbf{Y}_{1:k-1})$, which equals the cell probability value at the previous time step, thus enabling the recursive nature of the probability map update. Note that at initial run time when $k=0$, $P(h^k_{i,j}=1|\mathbf{Y}_{1:k-1}) = P_0^{i,j}$, an initial prior distribution on the probability map. This allows for the possibility of specifying a prior belief of object location in the given geometric workspace before the search has begun. In turn, this leads to a natural extension toward search in multiple rooms; e.g., a cup object is more likely to be found in the kitchen as opposed to the bedroom so P_0 could be constructed to have more probability mass in the workspace of the kitchen than the bedroom (see section 6.4.4 for further discussion on multi-room search).

Lastly, the denominator of equation (6.1) is obtained by marginalizing over the object cell location:

$$P(y^k_{u,v}|\mathbf{Y}_{1:k}) = \sum_{m,n} P(y^k_{u,v}|h^k_{m,n}=1,\mathbf{Y}_{1:k-1})P(h^k_{m,n}=1|\mathbf{Y}_{1:k-1})\,. \quad (6.3)$$

With the various terms of equation (6.1) expressed, a Bayesian recursion scheme is thus resolved for each cell $c_{i,j}$ of the probability map, $\mathcal{M}_p$. In effect, the recursive update of the probability map redistributes probability mass from the explored cells where no object is found to the unexplored cells.

6.2.2 Global Search Method

As noted by [14], the search process can be divided into a global search based on a coarse detection methodology which operates over longer ranges, and local detailed search that operates well at close

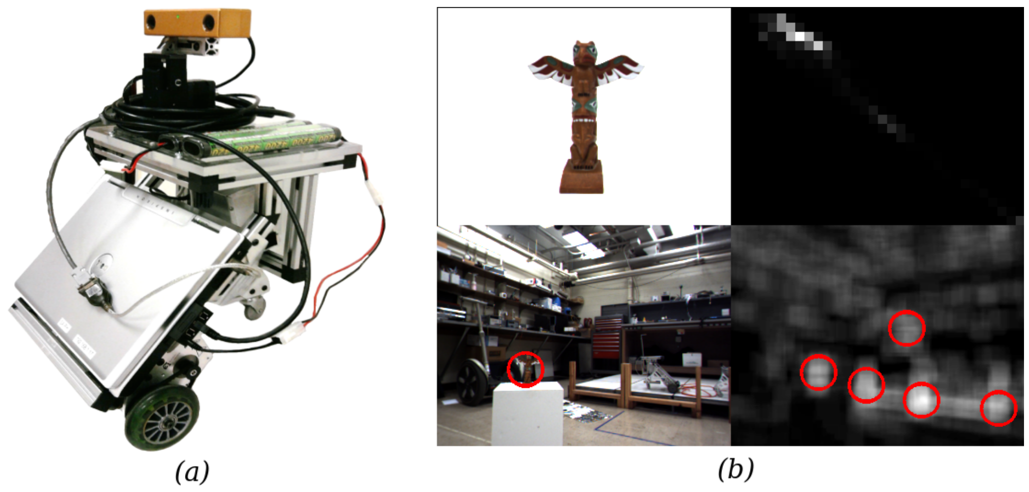


Figure 6.1: (a) The presented experiments for 3D object search used an Evolution Robotics ER-1 (a two-wheeled differential drive robot) equipped with a pan-tilt unit and a stereo camera. (b) The coarse global search uses a color histogram of the object to identify likely regions in the current image that resemble the object. Shown in the top left is a sample object. The top right shows the corresponding color histogram in RG (red-green) space. The bottom left shows one image from a stereo pair (with the true object location circled in red) while the bottom right shows the back projection of histogram comparison to subwindows of the current image. Probable object locations in the image are shown in red circles. The 3D location of the putatative object is next estimated from sparse stereo.

ranges. For the approach considered here, a common global search method is implemented that utilizes color histograms.

During a training phase (as mentioned earlier), color images of the object (e.g., see top left of Figure 6.1(b)) taken from various viewpoints are used to construct a cumulative model histogram using the red and green image channels, (e.g., the top-right image of Figure 6.1(b)). The model histogram is stored in a dictionary which can be queried during search. At the start of a search, the robot scans its environment at a set of pan-tilt angles that cover its viewing sphere. For each view, the image is back projected against the model using a fixed scanning window size (see bottom two images of Figure 6.1(b)). The resultant image is normalized to yield a distribution over the image pixels of the conditional probability that a given pixel is a member of the model histogram. Next, the N_{hist} highest probability pixel clusters are selected (e.g., the red circles in the bottom-right image of Figure 6.1(b)). Using sparse stereo the 3D locations to the peak probability values, $P'_n \; \forall n \in \{1, \cdots, N_{hist}\}$, are estimated. This process is repeated for all discrete pan angles.

The set of peak probability locations is then projected onto the probability map, $\mathcal{M}_p$. Each peak probability location, P'_n, is then treated as an individual measurement of the cell, $c_{u,v}$, to which it is projected. As such, the probability of object detection at that cell (if the object exists there) is set to $P(y^k_{u,v} = 1 | h^k_{u,v} = 1, \mathbf{Y}_{1:k-1}) = P'_n$ and the probability map $\mathcal{M}_p$ is updated according to equation (6.1), which incorporates prior cell probabilities from the term $P(h^k_{i,j} = 1 | \mathbf{Y}_{1:k-1})$.

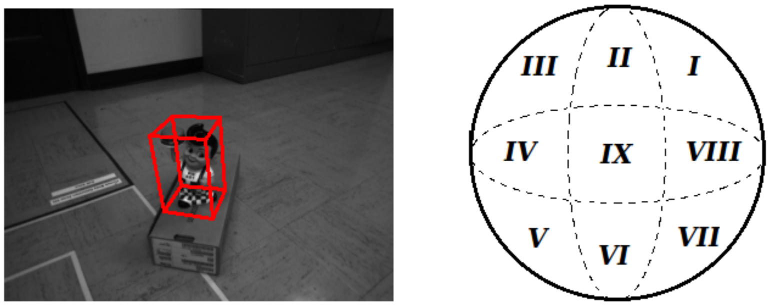


Figure 6.2: The 3D SIFT-based local search and pose estimation process is applied to each of quadrants of the viewing sphere. The diagram on the right depicts the arrangement of search regions while the left image is taken from an experiment where positive detection and pose estimation occurred in quadrant VI.

Because the global search is a coarse detection model, there is also a likelihood of the object existing in cells adjacent to the projected cell $c_{u,v}$. To account for this possibility, a variable β missed detection rate parameter is used, initialized to $1 - P_n'$ and increased exponentially for neighboring cells of $c_{u,v}$. This has the effect of incorporating high likelihood of object existence in a neighborhood of cells, as opposed to a single cell if β were kept constant (the second column of Figure 6.6 shows a series of probability maps generated from experiments).

Once a global search is executed and peak probability locations from the back-projected color histogram are integrated into $\mathcal{M}_p$, a path is then planned to the location of the peak probability, where a more refined local sensing method is executed. Thereafter the local search procedure provides the main method to update map probabilities, except in the case where the peak probability is below a threshold, meaning there is no strong evidence for the presence of the object (see section 6.4).

6.2.3 Local Search Method with 6D Pose Estimation

A more refined local search is used when the robot lies within a fixed distance of a peak in the search probability map. The local sensing method uses the same EKF framework of 3D object detection and tracking as developed in chapter 3, that is, sparse stereo SIFT feature matching, optimized by using a *k-d* tree with Lowe's Best-Bin-First Search schema ([2]). However, as opposed to the methods in [27] and [14], which also use SIFT features for object identification, the local search procedure as presented in chapter 3 estimates the full 6D pose of the identified object (see chapter 3 for a review of the detailed analyses and notation used).

It is important to note that while the EKF framework developed in chapter 3 considered potentially moving objects, in this particular problem the object is assumed to be stationary. Nonetheless, because the object position and orientation relative to the robot are not given a priori, an estimator that accurately rectifies the position and orientation of the object for any possible initial pose is essential. On this front, the Extended Kalman Filter of chapter 3 works fairly well in establishing accurate pose estimates (see left image of Figure 6.2).

Because in this particular implementation the robot is paused during the local search, the motion model of equation (3.2) is used (shown here for convenience), with $\mathbf{X}_k$ being the 6D state of the object, as seen in the camera reference frame:

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \eta \,. \tag{6.4}$$

Note that the robot dynamics are not integrated into the motion model. Should object pose tracking be needed while the robot moves and/or pans/tilts the camera while running the EKF, the motion model can be easily augmented to include the robot dynamics, similar to the motion models of equations (5.6) and (5.1.1). The measurement model also follows the same expression as equation (3.5) (shown here for convenience), with $\mathbf{D}_k$ being the set of 3D SIFT measurements from stereo, $\mathbf{B} = [\mathbf{b}_0\ \mathbf{b}_1\ \cdots\ \mathbf{b}_N]$ the set of database features generated during training, and $\mathbf{x}_{R,k}$ and $\mathbf{R}_{CO_k}$ the appropriate translational pose and rotation matrix formed from the elements of $\mathbf{X}_k$ to bring $\mathbf{b}_i$ into the same frame as $\mathbf{y}_i$.

$$\begin{aligned} \mathbf{D}_k &= \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(0)} \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(1)} \\ \vdots \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(n_K)} \end{bmatrix} + \xi \quad . \\ &= \mathbf{H}(\mathbf{X}_k, \mathbf{B}) + \xi \end{aligned} \tag{6.5}$$

As discussed in chapter 3, for increased computational efficiency, the burdensome SIFT calculations are limited to a region of interest (ROI) surrounding the object once the object track has been initialized (object initialization is based on the minimum number of observed SIFT matches and the covariance of object pose). Furthermore, to handle the problem of potential mismatches, if the object is initialized a geometric feasibility check is applied to each found feature correspondence. If a feature correspondence places a matched feature too far to be feasible given the known geometry of the object and its initialized pose, it is rejected.

It is important to note that the 3D physical space associated with a cell may not be entirely visible in a single camera view. To address the issue of *where to look* in the cell, an approach similar to [56] is considered with a search hemisphere defined and divided into 9 quadrants (see Figure 6.2). The quadrants are searched in sequence (using the 3D SIFT procedure) with the pan-tilt unit. If an object is detected, the PTU is adjusted to localize the detected object in the image center of the image frame. If the object is not detected in any quadrant, the cell probability is updated using equation (6.1), and the next peak location is determined. Because the local search is generally effective at close ranges, a constant α and β parameter are used which has the effect of

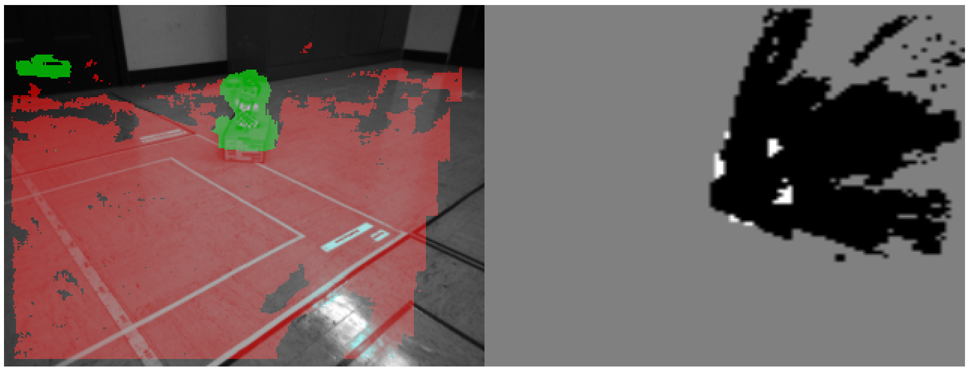


Figure 6.3: Robot navigation uses a costmap based on stereo imagery. *Left*: an image taken from an experiment, with space categorized as *free* (red) or *obstacle* (green). *Right*: the navigation costmap generated from the stereo images. White cells correspond to *obstacle*, black cells correspond to *free* and gray cells are *unknown*.

suppressing only the single searched cell when the object is not detected. Implicit in this approach is the underlying assumption that if the object lies in a given grid cell, it can be detected within the viewing hemisphere. This assumption requires the navigation system to deliver the robot to a position facing the center of the grid cell and in close enough range for the local search function to be effective.

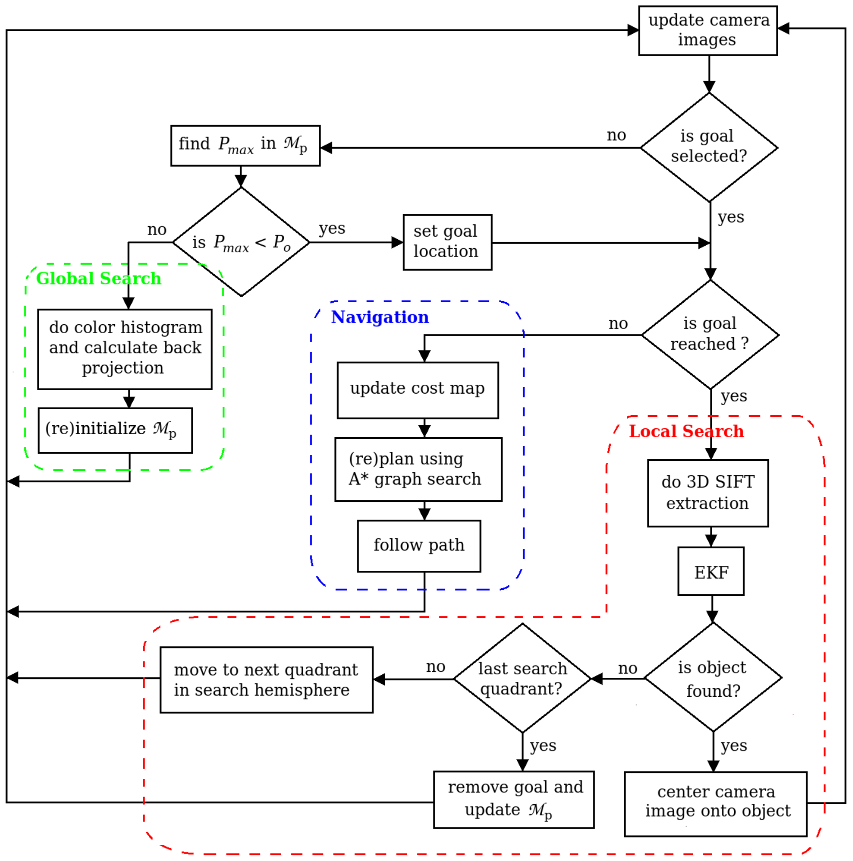
6.2.4 Robot Navigation and the Costmap

Navigation is an important part of object search. Previous work [27] [14] relied on a graph of predetermined free-space nodes to determine navigable paths. This approach limits the number of allowable robot positions and is not guaranteed to cover the search space.

To navigate, the robot maintains a grid-based costmap, $\mathcal{M}_c$, which is searched via the $A*$ algorithm to find the best feasible path to a selected search location. Each navigation map cell is classified as *free*, *unknown*, or *obstacle* by comparing points in the cell with the estimated ground plane. Points that lie within a threshold distance of the ground plane are labeled as *free*, while others are labeled as *obstacle*. All other cells are considered *unknown* until labeled as either *free* or *obstacle* (see Figure 6.3). A conservative approach is taken such that priority is given to *obstacle* if two types of points are projected into the same cell. Furthermore, when applying the $A*$ search algorithm, obstacles are grown on the map to account for the size of the robot.

To limit computational complexity of the obstacle detection process, the image is downsampled to 320×240 pixels while the robot navigates between search locations. To improve ground plane and obstacle detection, the stereo camera is tilted downwards.

It is often the case that the goal search location, as determined from peaks in the probability map, coincides with a cell marked as *obstacle* in the costmap (since the object may be placed on a box or a table which gets registered as an obstacle). This has the tendency to prevent the $A*$



update camera images
is goal selected?
no
find P_{max} in $\mathcal{M}_p$
yes
is $P_{max} < P_o$
no
yes
set goal location
Global Search
do color histogram and calculate back projection
(re)initialize $\mathcal{M}_p$
Navigation
is goal reached ?
no
yes
update cost map
(re)plan using A* graph search
follow path
Local Search
do 3D SIFT extraction
EKF
is object found?
no
yes
last search quadrant?
no
yes
move to next quadrant in search hemisphere
remove goal and update $\mathcal{M}_p$
center camera image onto object

Figure 6.4: Block diagram of the process data flow. The dashed boundaries surround the main components of our approach.

search from finding a feasible path to the goal location. In order to allow the robot to navigate close enough to apply the local search, neighboring obstacle cells of the goal cell are suppressed during the graph search.

6.3 Approach Summary

The particular experiments presented in the following section consider a uniform prior belief P_0 in a single room. Figure 6.4 summarizes the process data flow of the specific implementation. The dashed boundaries isolate the the main algorithm components discussed in section 6.2. To balance the use of a global versus a local search, the global search is invoked if the maximum probability in $\mathcal{M}_p$ is less than a threshold, P_o. Such an event can occur (1) at startup, when $\mathcal{M}_p$ is set to uniform P_0, and (2) when all cells have been visited and the peak probability of any one cell is not significantly large. This latter case occurs during a missed detection of the object over the entire global search space – thus, invoking an entirely new search process. For the majority of the search process, the local search procedure is the active mode. If a putative object location (a local peak in

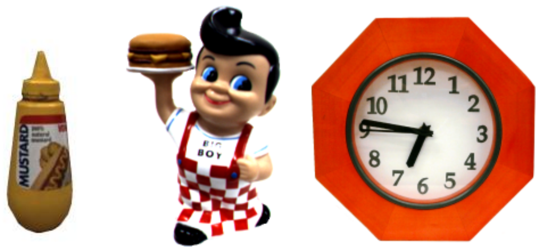


Figure 6.5: The models used in the series of trial experiments considered were (*from left to right*): a mustard bottle, a Bob's Big Boy model, and a clock.

the prior map probability) is found to be empty, the probability value in that cell is reduced (via equation (6.1)) and a new goal is selected and planned to via the navigation function.

6.4 Experimental Results

The presented approach has been implemented on an Evolution Robotics ER-1 mobile robot equipped with a Point Grey Research BumbleBee2 stereo color camera downsampled to a resolution of 320×240. The stereo camera is mounted on a Directed-Perception PTU-D46-17 pan-tilt unit. In the experiments summarized below, robot pose is estimated via wheel odometry. All computations were performed on a laptop computer (Intel Pentium M 1.86 GHz processor) running Linux. The algorithm was written in C/C++ using the Intel OpenCV library.

A Bob's Big Boy model, a mustard bottle, and a clock were initially used as search models (see Figure 6.5). For each object, a series of 4 trials were conducted. In the first three trials, the object was placed at three different heights and at random locations in the workspace. These trials aimed to test: (1) how well division of the search hemisphere into quadrants addressed the *where to look* problem; and (2) the ability of the local search method to identify and estimate object pose at various object heights. In order to test the navigation system, an obstacle is placed directly in the robot's search path during a fourth trial. In all trials, the ability of the global search procedure to generate a valid prior probability map, $\mathcal{M}_p$ is also considered.

6.4.1 Robustness to Height Variance

The three rows of Figure 6.6 show results from the first three trials with the object placed at various heights. For brevity, only one trial result from each object is shown. The far left column presents a monocular image with the object manually highlighted in red for convenience. As can be seen, the object is placed far enough such that a local search attempt would fail.

The second column of images shows a relevant portion of the probability map calculated during the initial global search (cell resolution is 20×20cm). Superimposed on the probability map is the global coordinate frame origin (green), the initial planned path (yellow), and the goal location

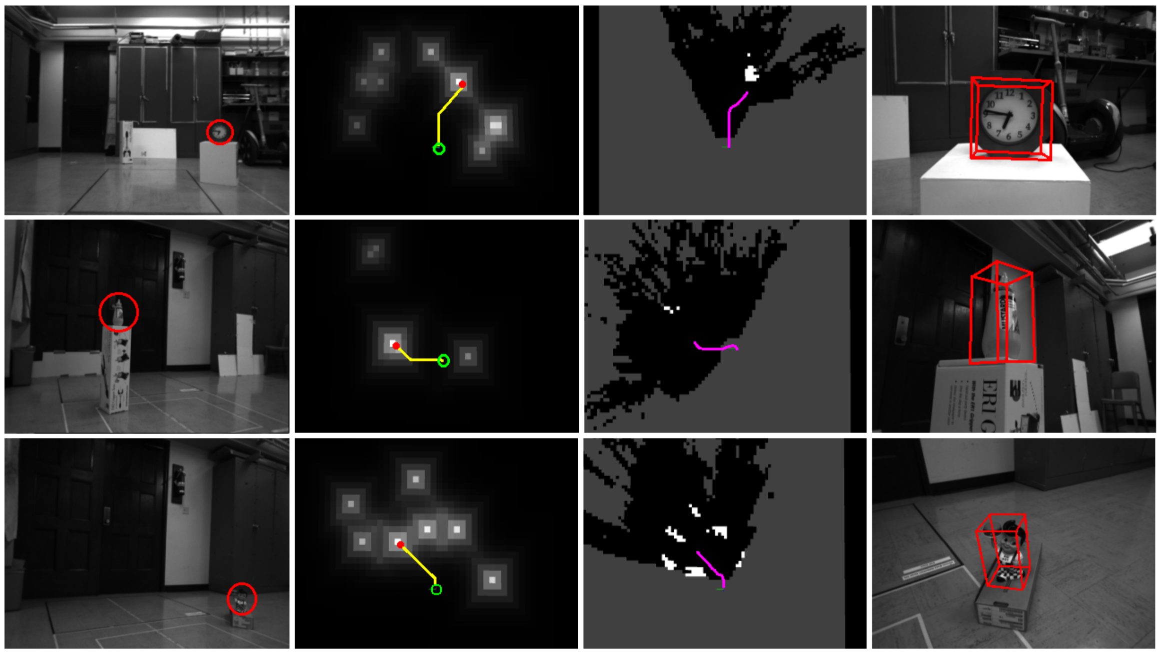


Figure 6.6: Experimental results from the set of trials that tested the robustness of the method to varied object height placements are depicted in the figure. Each row corresponds to a different trial. The first column shows an image of the environment with the true object location manually highlighted in red. The second column shows the probability map resulting from global search. The initially planned robot path is superimposed (yellow). The third column shows the costmap at the end of the trial, with the robot pose history (magenta). The final column shows the 6D pose estimation.

(red). Since path planning is executed repeatedly during navigation mode (see Figure 6.4), the initial planned path is subject to change should an obstacle appear later in that path. Note also that multiple probable object locations are typically hypothesized from the coarse search, requiring subsequent local search attempts. The first and third-row trials showed large numbers of probable locations mainly because the color histograms of the clock and Bob's Big Boy matched several objects in the space of the room. The second row trial had less peaks in the probability map mainly because of the unique yellow nature of the mustard bottle. Of the trials shown in Figure 6.6, the initial planned path always leads to the true object location – however, this is not always the case. The results discussed in section 6.4.2 will show a trial run where the initial planned path was not to the true goal location.

The third column displays the costmap at the end of the trial (cell resolution of 10×10cm). Superimposed on that map (magenta) is the history of the robot location as estimated by wheel odometry. In each of the trials, the robot ends at the true object location. Note that in the first and third-row trials, the costmap accurately identifies the boxes on which the clock and Big Boy sit as obstacles (the white cells in the costmap). In the second-row trial, the long narrow box on which the mustard bottle stands does not appear as on obstacle in the final costmap (see end of magenta

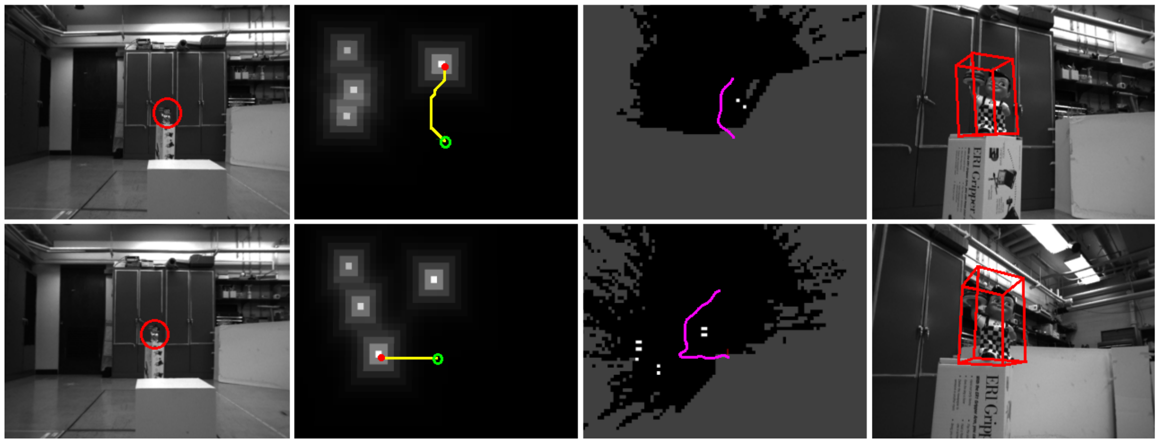


Figure 6.7: Experimental results from the set of trials that tested the robustness of the navigation function to avoid obstacles. Shown are two different results to the same trial setup of the same object.

trail in Figure 6.6). Close investigation of the logged data showed that the narrow width of the box did not always yield enough stereo returns to register an obstacle. This caused a flickering behavior of the obstacle location from one cell to the next such that by the time the robot stopped, a ground point was projected into that cell and was declared *free*. While this was not the desired behavior of the algorithm, it is a feature of the costmap generation from stereo that will be addressed in future work.

The fourth column shows the results of the local search recognition and 6D pose estimation algorithm. Even at various orientations, heights, and scales, the local search method does extremely well at locating and pose estimating the object in all trials.

6.4.2 Robustness to Obstacles and Re-planning

The two rows of Figure 6.7 show two different results of the fourth trial run considered for the Big Boy object. Note that in the first row of results, the global search identifies a peak probability location, the navigation function plans an initial path around the box obstacle as seen in the costmap, and the local search accurately identifies and estimates the object pose once the goal cell location is reached. However, note that in a second test run in a nearly identical room configuration, the global search identifies a cell to the left of the robot as the most probable object location – which in fact does not contain the object. Nonetheless, the trace of the robot pose (third column) illustrates that upon applying the local search at the first goal location and failing to find the object in any one search quadrant, the robot re-plans to the next peak cell in the probability map. The local search applied to the second goal location in fact yields the true object and pose estimation is accurately applied.

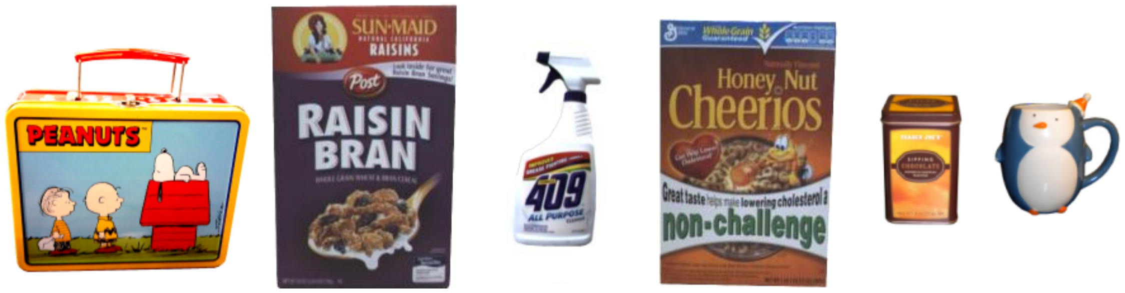


Figure 6.8: The additional models used in the series of trial experiments considered were (left to right): a lunch-box, a Raisin Bran cereal box, a window cleaner spray bottle, a Cheerios cereal box, a tin can of chocolate mix, and a penguin cup.

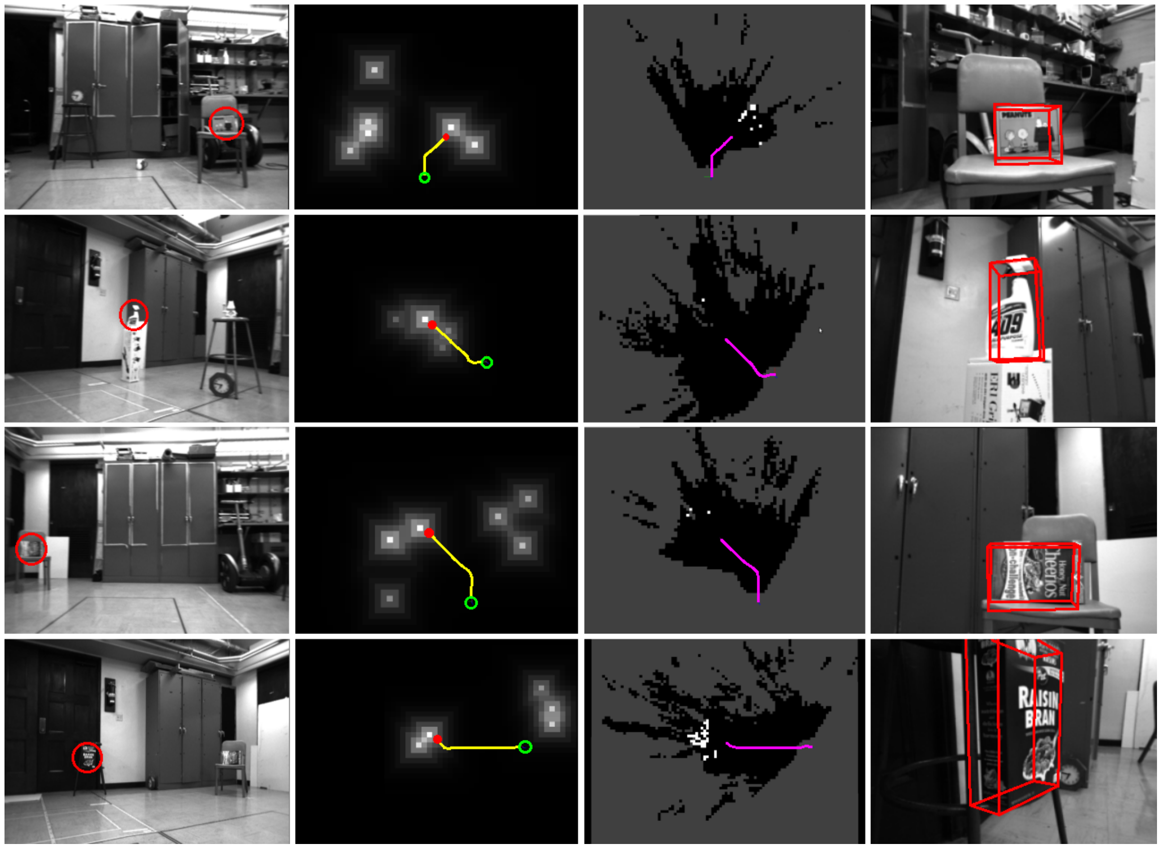


Figure 6.9: Shown above are the successful experimental results associated with 4 of the 6 additional objects considered. Following the format described in Figure 6.6, each column shows a particular stage of the presented search algorithm, with each row corresponding to a unique object.

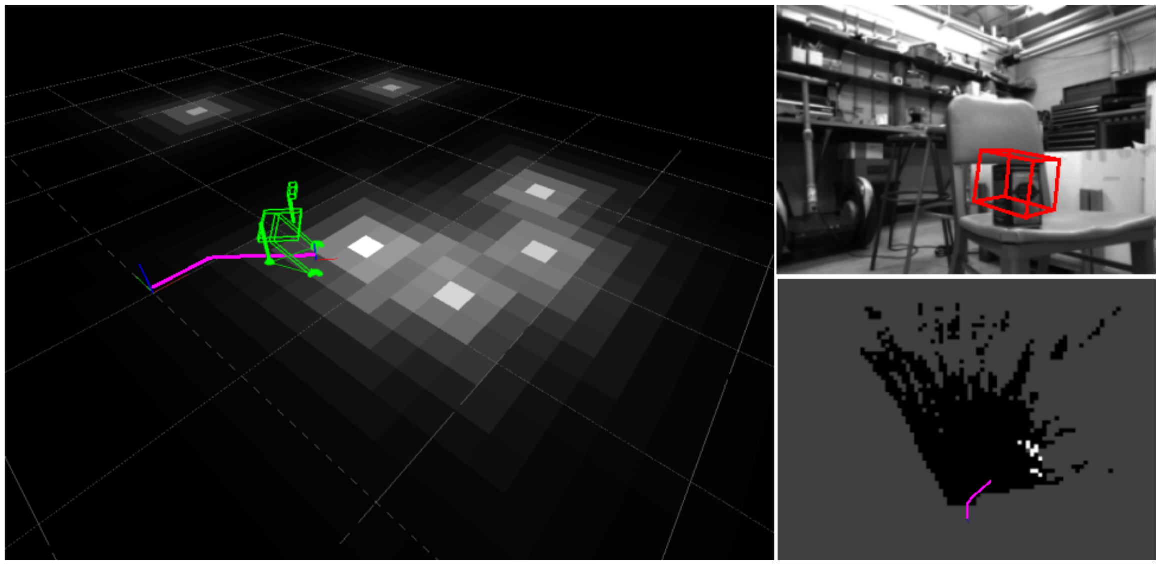


Figure 6.10: Experimental results from the object search failed attempt for the tin can of chocolate mix. Shown on the left is the grid-based probability map generated from the global-search. The peak probability cell is actually the true cell containing the object being searched for. The bottom right is the grid-based costmap generated using stereo-obstacle avoidance. Note the magenta line in both images correspond to the robot path as determined from the wheel odometry. The top right image shows the estimated pose of the object, which is incorrectly estimated.

6.4.3 Additional Tests and Search Limits

Based on the success of the three models used in the series of experiments discussed in the previous sections, an additional set of object models was used. Figure 6.8 shows the additional objects considered. The lunch box, two cereal boxes, and the spray bottle were all easily detectable using the proposed method, as the results of Figure 6.9 show. Note that consistent across these 4 models is the fairly large size and good amount of texture on each object which makes the overall conditions of the search problem somewhat favorable. The large size allows for the color histogram approach of the global search method to quickly identify the most probable location as the correct one in the probability map. The texture on the object further allows for the refined local search method to accurately estimate the pose of the object. However, the tin can of chocolate mix and penguin cup were considerably more difficult for the algorithm to find. Both objects are fairly small making the search rather challenging, particularly for the local search method.

In the case of the tin can, Figure 6.10 shows the results of one of several failed attempts to find the object. As can be seen in the left image of the figure, the grid-based probability map shows several probable locations containing the object — with the most probable cell location actually being somewhere in the vicinity of the true cell containing the object. Once the local search is applied to find the 6D pose of the object, the estimated pose fails to localize on the true position (see top right image). This is in part due to the small size of the object which makes the features

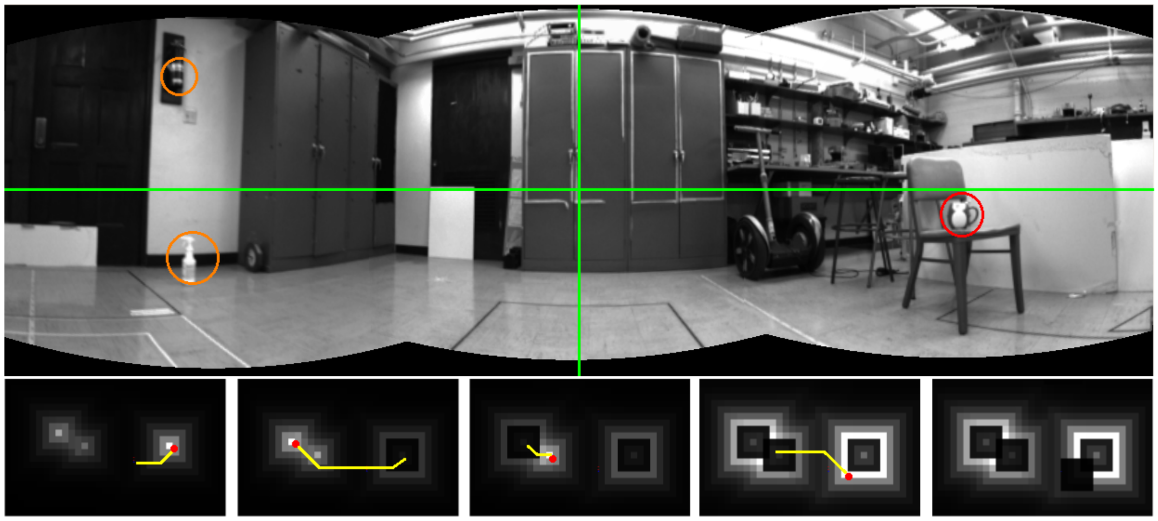


Figure 6.11: Experimental results from the object search failed attempt for the penguin cup. Shown in the top view is the panoramic view formed from the initial survey of the environment from the robot's perspective. From that, a probability map is constructed indicating the most probable locations for the penguin cup. The bottom row shows the progress of the grid-based probability map as the experiment progresses and the robot investigates more probable cells. The width of the each grid-based map shown in the bottom row matches the combined field of view of the panoramic image above.

on the object difficult to find in the local search, and resulting feature matching to be fairly poor.

In the case of the penguin cup, this particular object was the most challenging to find because of its small size and lack of distinguinshing features further removed by the polished surfaces. This caused the local search to fail to find any matched features on the object, even in situations when the object was placed directly in front of the robot. In a majority of the cases, the number of detected features on the cup would be 1 or 2 which would clearly not be enough to determine any form of accurate 6D pose. What was interesting to note in the experiments for this object was that since the local search always failed to find the object in any cells, it allowed the recursive Bayesian update of the probability map (as defined in Eq. 6.1) to run its course.

Figure 6.11 shows the results of one particular trial with the penguin cup. The top image of the figure shows a panoramic view of the evironment stitched together from the initial scan of images taken by the robot (the horizontal green line indicates the horizon and the vertical green line indicates the viewing angle associated with a forward facing robot). The bottom row of images show sequential snapshots of the probability map (left to right) taken each time after a local search had been applied and failed to find the object. The width of each grid-based map shown in the bottom row matches the combined field of view of the panoramic image above. At the beginning of the experiment, three probable locations stand out in the initial probability map (bottom row, left image). The objects in the scene corresponding to the cell locations have been highlighted in the panoramic image, with the true object circled in red and distractor objects circled in orange.

Initially, the robot plans a path to investigate a cell location to the right, corresponding to the actual location of the penguin cup. However, once the object is not found in that cell (due to a lack of features), the probability is reduced and the next probable location is planned to and visited (bottom row, second image from left), which in this case corresponds to the fire extinguisher on the wall. Upon failing to find the object there, the probability is again reduced and the next probable location visited (bottom row, third image from left), which now corresponds to the spray bottle on the floor. Note that because of the reduced probability of the cells associated with the fire extinguisher, probability mass has now been increased in the cells associated with the spray bottle and also the surrounding cells of the first visited cell. Once the spray bottle is found not to be the penguin cup, the robot reduces the probability of the cell and now revisits a neighboring cell of the first visited cell (bottom row, second from right). When the object is not found there, the robot again reduces the probability of that cell and continues to the next probable location (bottom row, right image). At this point, the experiment was terminated by the operator, but the resulting data illustrates the ability of the search algorithm to use the Bayesian update to identify probable search locations.

Based on the results of these additional experiments, it was observed that the search algorithm performs fairly well for large, textured objects with unique color distributions. The large size and unique color distribution serves to improve the generated probability map from the color-histogram based global search such that the initial most probable location is the true one. If the color distibution is not unique, the search algorithm will still find the object, yet not necessarily on the first local search attempt (see second row of Figure 6.11). The texture on the object is also fairly important as well as the results of Figures 6.10 and 6.11 have shown. Highly textured objects will have a larger number of descriptive features allowing for the local search to accurately estimate the 6D pose of the object.

6.4.4 Extensions to Multiroom Search

While the results of the experiments discussed in sections 6.4.1, 6.4.2, and 6.4.3 considered a uniform prior P_0 over a single room, the presented approach is not limited to this case. The Bayesian recursion update, as mentioned in section 6.2.1, allows for various initial distributions of P_0. If there is prior information that an object is likely to exist in one region (or room) as opposed to another, P_0 can be weighted to have more mass in a concentrated area (similar to the various initial cases considered by [6]). The distribution of P_0 will govern whether the robot searches peak locations in the immediate vicinity, or forgoes those locations for a concentrated likelihood area in P_0.

To illustrate this point, a simple simulation was run that considered a robot placed in an environment with 4 rooms (see top image of Figure 6.12). The true object location (shown in green) is in an adjacent room from where the robot currently exists. Two objects exist in the field of view of the

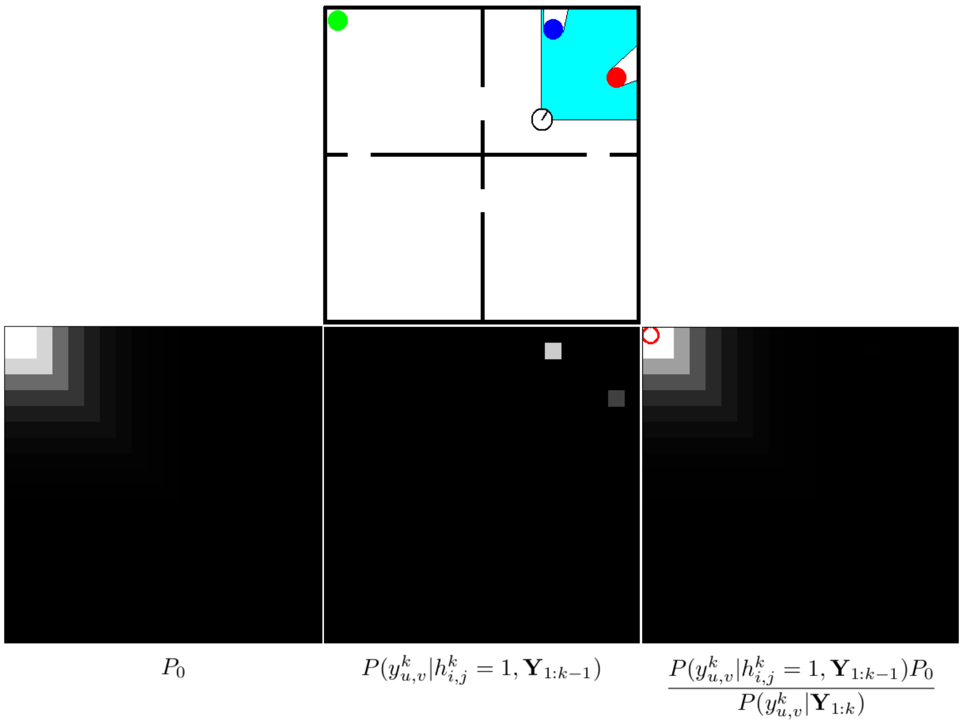

P_0
$P(y^k_{u,v}|h^k_{i,j}=1,\mathbf{Y}_{1:k-1})$
$\frac{P(y^k_{u,v}|h^k_{i,j}=1,\mathbf{Y}_{1:k-1})P_0}{P(y^k_{u,v}|\mathbf{Y}_{1:k})}$

Figure 6.12: *Simulated search in a multi-room environment.* Shown in the top image is the simulated environment with the true object in green and and the robot in an adjacent room sensing two potential objects. In the bottom row of images, the geometric prior P_0 is shown on the left, the results of a global search are shown in the middle image, and the updated probability map is shown on the right.

robot which are detected via a global search method and probabilities assigned to each projected cell location (center image of bottom row). A geometric prior, P_0, is given to the robot ahead of time indicating there is a strong probability the object exists in an adjacent room (left image of bottom row). When the Bayesian update is applied to the probability map using the global search detected probabilities in conjunction with the given prior, P_0, the effect of the global search detections is nearly unseen in the resultant probability map (right image of bottom row) due to the strong peak in the prior P_0.

However, if the distribution of P_0 is spread out such that the peak location is reduced in probability, the effect of the global search detections become more prominent. Figure 6.13 shows two additional simulations done where the spread of P_0 was gradually increased, in effect reducing the concentration of probability mass. As can be seen in the right image of the top row, one of the global search detections becomes visible and as the spread of P_0 is further increased, the peak probability then switches from a location specified by the geometric prior to one within the immediate vicinity of the robot's sensing range. Thus it becomes clear the prior P_0 can be used in such a way so as to influence the selected goal locations of the robot, if so desired.

In summary, it is evident the Bayesian update of equation (6.1) appropriately adjusts the proba-

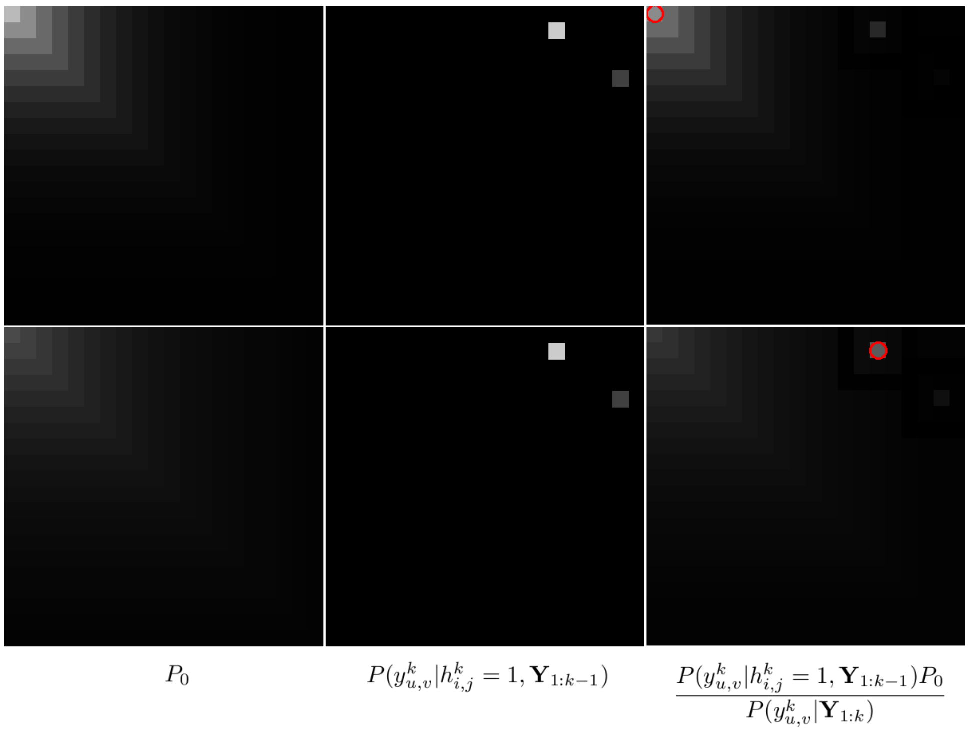


Figure 6.13: *Simulated search in a multi-room environment.* Similar to the previous figure, shown in the top row is a slightly weaker prior and the resultant updated probability map. The bottom row shows the results of an even weaker prior, resulting in a switch of the peak probability.

bility map to balance incoming measurements against prior information, such that at each planning action, the peak probability location can always be selected to yield the most probable object location.

6.5 Conclusions

An improved stereo vision search procedure was presented that uses Bayesian updating of a grid-based probability map to drive the search process. By using a global and local search method, object search with path planning, obstacle avoidance, and 6D pose estimation has been demonstrated on three different model objects using only a single stereo sensor and modest computation. Extensions to the multi-room search problem were also considered, illustrating the capability of the proposed method to longer searches in larger environments.

Chapter 7

Conclusion and Future Work

7.1 Discussion and Summary of Work

This thesis developed a new real-time 3D object detection and tracking method using stereo vision data. Extensions of the core tracking algorithm to sensor planning and object search were also considered.

In particular, the 3D object detection and tracking algorithm presented in chapter 3 introduced a novel training technique that enables fast object training, which would allow large databases of objects to be generated quickly. The contributions discussed in that chapter extended the work of previous authors ([54], [40], [45], [5]) to show that real-time robust tracking could be achieved using 3D SIFT features in stereo and tracked via an application of Bayes' Filter. A novel method for determining ground truth 6D pose estimates was also introduced that did not require the purchase of third-party software ([54]), the use of simulated experiments ([40]), nor the need for distinguishing markers for computer vision toolkits that are highly susceptible to noise ([5]).

Chapter 4 extended the framework of chapter 3 to the problem of sensor planning for model identification. Bayesian analysis was used to interpret an information gain metric that became the Sensor-Planning-for-Pose-Estimation-and-Model-Identification (SPPEMI) algorithm. Experimental results were shown in chapter 5 that illustrated the capabilities of the algorithm to plan for the *next-best-view* when the object's initial view was not discriminating enough to allow accurate identification. The results also showed the algorithm's ability to handle unconstrained motion of both the mobile agent and the object, while estimating the object's 6D pose in real-time – an improvement on the works of previous authors ([39], [11], [12], [25]) who did not consider full 6D pose estimation in their analysis.

Chapter 6 also built on the framework of chapter 3 to address the problem of *object search*. The presented approach made use of a global and local search decomposition to locate the object and accurately estimate its 6D pose – an improvement on previous methods ([13], [14], [27]) that had limited object pose estimation to crude approximations based on object size in the image without any Euler angle estimates. The overall search strategy was governed by a grid-based *probability* *map*, updated via Bayesian recursion. Obstacle avoidance was also integrated into the system by maintaining a secondary grid-based *costmap* that was updated from stereo data and searched using an A^* algorithm. Experimental implementation of the approach on a real robotic system showed the validity of the proposed method for three types of objects at various heights and locations in a large room. Simulated results were also shown that extended the framework to multi-room search as well.

7.2 Conclusions and Future Work

Future work on these methodologies should generally focus on increasing the robustness of the approaches through extended experimental studies. In particular, the 3D object-detection and tracking algorithm discussed in chapter 3 relies heavily on the accuracy of the Best-Bin-First-Search matching schema of [2] for robust tracking. This method tends to weight false matches with equal probability as regular matches. While a geometric constraint check was implemented for the initialized tracks, a more formal methodology that considers the probability of correspondence should be developed. Dellaert's [9] method – based on Expectation Maximization with an MCMC approach to learn the most probable associations in the structure from motion problem – may be useful in this regard.

The sensor planning for model identification experimental results were performed with only 4 object models, requiring 4 individual Extended Kalman Filters to estimate the position of each model object. While that approach seems limiting to the natural extension toward multiple objects, future work should consider only applying Extended Kalman Filters to a limited set of the most probable objects – e.g., the n most probable models. Work by Nister et al. in [37] suggests a large database of objects can be efficiently organized into a vocabulary tree that could also be implemented in this framework to extend the number of considered objects to several hundred.

While multiple objects should be considered in the future, a more pressing issue is to test the search strategy under harsher lighting conditions. In the experiments of chapter 6, no significant variations in lighting were considered. It would also be interesting to see in conjunction with the lighting variation, a validation of the multi-room search analysis discussed in section 6.4.4.

Finally, a joint experiment combining the frameworks of chapter 5 and 6 should be implemented to address the interesting problem of dynamic object search and identification – i.e., search for a potentially moving object for identification. The search strategy of chapter 6 combined with the EKF implementation in the local search is well equipped for tracking a moving object and the methodology of chapter 4 already describes a sensor planning strategy for improved sensing toward model identification.

Appendix A

Linearized Measurement Matrix

For the given measurement model of the stereo camera system:

$$\begin{aligned}
\mathbf{D}_k &= \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{n_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(1)} \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(2)} \\ \vdots \\ \mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(n_K)} \end{bmatrix} + \xi\,, \\
&= \mathbf{H}(\mathbf{X}_k, \mathbf{B}) + \xi
\end{aligned}$$

the matrix $\mathbf{C}_k \in \mathbb{R}^{(3\times n_k)\times 6}$ is the linearized measurement matrix evaluated at the predicted pose location:

$$\begin{aligned}
\mathbf{C}_k &= \left.\frac{\partial \mathbf{H}}{\partial \mathbf{X}_k}\right|_{\overline{\mathbf{X}}_k} \\
&= \begin{bmatrix} \frac{\partial \mathbf{h}_1}{\partial \mathbf{X}_k} \\ \frac{\partial \mathbf{h}_2}{\partial \mathbf{X}_k} \\ \vdots \\ \frac{\partial \mathbf{h}_3}{\partial \mathbf{X}_k} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{X}_k}(\mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(1)}) \\ \frac{\partial}{\partial \mathbf{X}_k}(\mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(2)}) \\ \vdots \\ \frac{\partial}{\partial \mathbf{X}_k}(\mathbf{x}_{R,k} + \mathbf{R}_{CO_k}\mathbf{b}_{\mathbf{J}(n_k)}) \end{bmatrix},
\end{aligned}$$

where $\frac{\partial \mathbf{h}_i}{\partial \mathbf{X}_k} \in \mathbb{R}^{3\times 6}$ and the measurement model for the i-th detected feature, $\mathbf{h}_i$, can be expanded as

$$\begin{bmatrix} h_1^i \\ h_2^i \\ h_3^i \end{bmatrix} = \begin{bmatrix} x_k + c\alpha c\beta \cdot b_x^{J(i)} + (c\alpha s\beta s\gamma - s\alpha c\gamma)\cdot b_y^{J(i)} + (c\alpha s\beta c\gamma + s\alpha s\gamma)\cdot b_z^{J(i)} \\ y_k + s\alpha c\beta \cdot b_x^{J(i)} + (s\alpha s\beta s\gamma + c\alpha c\gamma)\cdot b_y^{J(i)} + (s\alpha s\beta c\gamma - c\alpha s\gamma)\cdot b_z^{J(i)} \\ z_k - s\beta \cdot b_x^{J(i)} + c\beta s\gamma \cdot b_y^{J(i)} + c\beta c\gamma \cdot b_z^{J(i)} \end{bmatrix},$$

where $c(\cdot) \triangleq cos(\cdot)$ and $s(\cdot) \triangleq sin(\cdot)$, and the i-th feature measurement, $\mathbf{y}_i$, is found to correspond to the $\mathbf{J}(i)$-th database feature, $\mathbf{b}_{\mathbf{J}(i)} = [\ b_x \quad b_y \quad b_z\ ]^T$. Solving for $\frac{\partial \mathbf{h}_i}{\partial \mathbf{X}_k}$ (recall that $\mathbf{X}_k = [\ x \quad y \quad z \quad \alpha \quad \beta \quad \gamma\ ]_k^T$), the resultant submatrix for the i-th linearized feature measurement in $\mathbf{C}_k$ becomes

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{X}_k} = \begin{bmatrix} 1 & 0 & 0 & \frac{\partial h_1^i}{\partial \alpha} & \frac{\partial h_1^i}{\partial \beta} & \frac{\partial h_1^i}{\partial \gamma} \\ 0 & 1 & 0 & \frac{\partial h_2^i}{\partial \alpha} & \frac{\partial h_2^i}{\partial \beta} & \frac{\partial h_2^i}{\partial \gamma} \\ 0 & 0 & 1 & \frac{\partial h_3^i}{\partial \alpha} & \frac{\partial h_3^i}{\partial \beta} & \frac{\partial h_3^i}{\partial \gamma} \end{bmatrix}$$

where

$$\begin{aligned}
\frac{\partial h_1^i}{\partial \alpha} &= -s\alpha c\beta \cdot b_x^{J(i)} + (-s\alpha s\beta s\gamma - c\alpha c\gamma) \cdot b_y^{J(i)} + (-s\alpha s\beta c\gamma + c\alpha s\gamma) \cdot b_z^{J(i)}\,, \\
\frac{\partial h_1^i}{\partial \beta} &= -c\alpha s\beta \cdot b_x^{J(i)} + c\alpha c\beta s\gamma \cdot b_y^{J(i)} + c\alpha c\beta c\gamma \cdot b_z^{J(i)}\,, \\
\frac{\partial h_1^i}{\partial \gamma} &= (c\alpha s\beta c\gamma + s\alpha s\gamma) \cdot b_y^{J(i)} + (-c\alpha s\beta s\gamma + s\alpha c\gamma) \cdot b_z^{J(i)}\,, \\
\frac{\partial h_2^i}{\partial \alpha} &= c\alpha c\beta \cdot b_x^{J(i)} + (c\alpha s\beta s\gamma - s\alpha c\gamma) \cdot b_y^{J(i)} + (c\alpha s\beta c\gamma + s\alpha s\gamma) \cdot b_z^{J(i)}\,, \\
\frac{\partial h_2^i}{\partial \beta} &= -s\alpha s\beta \cdot b_x^{J(i)} + s\alpha c\beta s\gamma \cdot b_y^{J(i)} + s\alpha c\beta c\gamma \cdot b_z^{J(i)}\,, \\
\frac{\partial h_2^i}{\partial \gamma} &= (s\alpha s\beta c\gamma - c\alpha s\gamma) \cdot b_y^{J(i)} + (-s\alpha s\beta s\gamma - c\alpha c\gamma) \cdot b_z^{J(i)}\,, \\
\frac{\partial h_3^i}{\partial \alpha} &= 0\,, \\
\frac{\partial h_3^i}{\partial \beta} &= -c\beta \cdot b_x^{J(i)} - s\beta s\gamma \cdot b_y^{J(i)} - s\beta c\gamma \cdot b_z^{J(i)}\,, \\
\frac{\partial h_3^i}{\partial \gamma} &= c\beta c\gamma \cdot b_y^{J(i)} - c\beta \cdot s\gamma \cdot b_z^{J(i)}\,.
\end{aligned}$$

Appendix B

Motion Model for Unconstrained Motion

For the given kinematic relation between the various reference frames associated with the camera (C), the mobile object (O), and the robot (R) between timestep k and $k-1$:

$$G_{C_k,O_k} = G^{-1}_{R_k C_k} \cdot G^{-1}_{R_{k-1} R_k} \cdot G_{R_{k-1} C_{k-1}} \cdot G_{C_{k-1},O_{k-1}} \cdot G_{O_{k-1} O_k}\ ,$$

the various transforms were defined in chapter 5 as follows:

$$G_{R_k C_k} = \begin{bmatrix} s(\dot{\phi}\Delta t) & 0 & c(\dot{\phi}\Delta t) & -b \\ -c(\dot{\phi}\Delta t) & 0 & s(\dot{\phi}\Delta t) & 0 \\ 0 & -1 & 0 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}, \quad G^{-1}_{R_k C_k} = \begin{bmatrix} s(\dot{\phi}\Delta t) & -c(\dot{\phi}\Delta t) & 0 & b\cdot s(\dot{\phi}\Delta t) \\ 0 & 0 & -1 & l \\ c(\dot{\phi}\Delta t) & s(\dot{\phi}\Delta t) & 0 & b\cdot c(\dot{\phi}\Delta t) \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1},$$

$$G_{R_{k-1} C_{k-1}} = \begin{bmatrix} s(\phi) & 0 & c(\phi) & -b \\ -c(\phi) & 0 & s(\phi) & 0 \\ 0 & -1 & 0 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1},$$

$$G_{O_{k-1},O_k} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

$$G_{R_{k-1} R_k} = \begin{bmatrix} c(\delta_\theta) & -s(\delta_\theta) & 0 & c(\theta_g)\delta_x + s(\theta_g)\delta_y \\ s(\delta_\theta) & c(\delta_\theta) & 0 & -s(\theta_g)\delta_x + c(\theta_g)\delta_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1},$$

$$G^{-1}_{R_{k-1}R_k} = \begin{bmatrix} c(\delta_\theta) & s(\delta_\theta) & 0 & \nu_x \\ -s(\delta_\theta) & c(\delta_\theta) & 0 & \nu_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}$$

$$G_{C_{k-1},O_{k-1}} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & x \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & y \\ -s\beta & c\beta s\gamma & c\beta c\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}$$

where $c(\cdot) \triangleq cos(\cdot)$ and $s(\cdot) \triangleq sin(\cdot)$ and the terms ν_x and ν_y are given by

$$\nu_x = -[c(\delta_\theta)c(\theta_g)\delta_x + c(\delta_\theta)s(\theta_g)\delta_y - s(\delta_\theta)s(\theta_g)\delta_x + s(\delta_\theta)c(\theta_g)\delta_y]\,,$$
$$\nu_y = -[-s(\delta_\theta)c(\theta_g)\delta_x - s(\delta_\theta)s(\theta_g)\delta_y - c(\delta_\theta)s(\theta_g)\delta_x + c(\delta_\theta)c(\theta_g)\delta_y]\,.$$

Defining $\Phi = G^{-1}_{R_kC_k} \cdot G^{-1}_{R_{k-1}R_k}$, the transforms combine to yield

$$\Phi = \begin{bmatrix} s(\dot\phi\Delta t + \delta_\theta) & -c(\dot\phi\Delta t + \delta_\theta) & 0 & \begin{array}{c} -s(\dot\phi\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) \\ +c(\dot\phi\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) + b \cdot s(\dot\phi\Delta t) \end{array} \\ 0 & 0 & -1 & l \\ c(\dot\phi\Delta t + \delta_\theta) & s(\dot\phi\Delta t + \delta_\theta) & 0 & \begin{array}{c} -c(\dot\phi\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) \\ -s(\dot\phi\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) \end{array} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1} .$$

Now defining $\Gamma = \Phi \cdot G_{R_{k-1}C_{k-1}}$, the expression for Γ becomes

$$\Gamma = \begin{bmatrix} c(\dot\phi\Delta t + \delta_\theta - \phi) & 0 & s(\dot\phi\Delta t + \delta_\theta - \phi) & \gamma_x \\ 0 & 1 & 0 & 0 \\ -s(\dot\phi\Delta t + \delta_\theta - \phi) & 0 & c(\dot\phi\Delta t + \delta_\theta - \phi) & \gamma_z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1} .$$

where the terms γ_x and γ_z are given by

$$\gamma_x = -b \cdot s(\dot\phi\Delta t + \delta_\theta) + -s(\dot\phi\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) +$$
$$c(\dot\phi\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) + b \cdot s(\dot\phi\Delta t)\,,$$

$$\gamma_z = -b \cdot c(\dot{\phi}\Delta t + \delta_\theta) + -c(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) + \\ - s(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g))\,.$$

To simplify the expressions, the following terms are introduced

$$\begin{aligned}
\xi &= \dot{\phi}\Delta t + \delta_\theta - \phi\,,\\
\rho_1 &= -b \cdot s(\dot{\phi}\Delta t + \delta_\theta) + -s(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) + \\
&\quad c(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) + b \cdot s(\dot{\phi}\Delta t)\,,\\
\rho_2 &= -b \cdot c(\dot{\phi}\Delta t + \delta_\theta) + -c(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) + \\
&\quad - s(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g))\,.
\end{aligned}$$

Then the final expression for G_{C_k,O_k} becomes

$$G_{C_k,O_k} = \begin{bmatrix} c(\xi) & 0 & s(\xi) & \eta_1 \\ 0 & 1 & 0 & 0 \\ -s(\xi) & 0 & c(\xi) & \eta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1} \cdot \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & x \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & y \\ -s\beta & c\beta s\gamma & c\beta c\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}$$

$$= \begin{bmatrix} c\xi c\alpha c\beta - s\xi s\beta & \begin{matrix} c\xi(c\alpha s\beta s\gamma - s\alpha c\gamma) \\ +s\xi c\beta s\gamma \end{matrix} & \begin{matrix} c\xi(c\alpha s\beta c\gamma + s\alpha s\gamma) \\ +s\xi c\beta c\gamma \end{matrix} & c\xi x + s\xi z + \rho_1 \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & y \\ -s\xi c\alpha c\beta - c\xi s\beta & \begin{matrix} -s\xi(c\alpha s\beta s\gamma - s\alpha c\gamma) \\ +c\xi c\beta s\gamma \end{matrix} & \begin{matrix} -s\xi(c\alpha s\beta c\gamma + s\alpha s\gamma) \\ +c\xi c\beta c\gamma \end{matrix} & -s\xi x + c\xi z + \rho_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}.$$

Finally, extracting out the relevant Euler parameters, the motion model from the kinematic relations reduces to the following:

$$\begin{aligned}
\mathbf{X}_{k,i} &= \begin{bmatrix} \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T \\ \begin{bmatrix} f_\alpha & f_\beta & f_\gamma \end{bmatrix}^T \end{bmatrix}_{k-1} + \eta\,,\\
&= \mathbf{F}(\mathbf{X}_{k-1,i}, \mathbf{u}_{k-1}) + \eta
\end{aligned}$$

where η is Gaussian white system noise and f_x, f_y, and f_z are given by

$$\begin{aligned}
f_x &= c(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot x + s(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot z \\
&\quad - b \cdot s(\dot{\phi}\Delta t + \delta_\theta) + -s(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) \\
&\quad + c(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g)) + b \cdot s(\dot{\phi}\Delta t)\,, \\
f_y &= y\,, \\
f_z &= -s(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot x + c(\dot{\phi}\Delta t + \delta_\theta - \phi) \cdot z \\
&\quad - b \cdot c(\dot{\phi}\Delta t + \delta_\theta) + -c(\dot{\phi}\Delta t)(\delta_x c(\delta_\theta + \theta_g) + \delta_y s(\delta_\theta + \theta_g)) \\
&\quad - s(\dot{\phi}\Delta t)(-\delta_x s(\delta_\theta + \theta_g) + \delta_y c(\delta_\theta + \theta_g))\,,
\end{aligned}$$

and f_α, f_β, and f_γ are given by:

$$\begin{aligned}
f_\alpha &= tan^{-1}\left(\frac{s\alpha s\beta}{c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha c\beta - s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\beta}\right)\,, \\
f_\beta &= -sin^{-1}(-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha c\beta - c(\dot{\phi}\Delta t + \delta_\theta - \phi)s\beta)\,, \\
f_\gamma &= tan^{-1}\left(\frac{-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha s\beta s\gamma + s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\alpha c\gamma + c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\beta s\gamma}{-s(\dot{\phi}\Delta t + \delta_\theta - \phi)c\alpha s\beta c\gamma - s(\dot{\phi}\Delta t + \delta_\theta - \phi)s\alpha s\gamma + c(\dot{\phi}\Delta t + \delta_\theta - \phi)c\beta c\gamma}\right)\,.
\end{aligned}$$

Bibliography

[1] T. Arbel and F. P. Ferrie. Entropy-based gaze planning. In *Proc. 2nd IEEE Workshop on Perception for Mobile Agents*, pages 779–786, Santa Barbara, California, 1999.

[2] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbor search in high-dimensional spaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1000–1006, San Juan, Puerto Rico, 1997.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 509–522, 2002.

[4] F. G. Callari and F. P. Ferrie. Autonomous recognition: Driven by ambiguity. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 701–707, San Francisco, CA, 1996.

[5] C. Choi, S. M. Baek, and S. Lee. Real-time 3d object pose estimation and tracking for natural landmark based visual servo. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robot Systems*, pages 3983–3939, Nice, France, 2008.

[6] T.H. Chung and J.W. Burdick. A decision making framework for control strategies in probabilistic search. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4386–4393, Rome, Italy, 2007.

[7] C. Cyr and B. Kimia. 3d object recognition using shape similarity-based aspect graph. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 254–261, Vancouver, British Columbia, 2001.

[8] A. J. Davison. Active search for real-time vision. In *Proc. 10th IEEE Int. Conf. on Computer Vision*, volume 1, pages 66–73, Beijing, China, 2005.

[9] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun. Structure from motion without correspondence. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 557–564, Hilton Head, South Carolina, 2000.

[10] D. DeMenthon and L. Davis. Model-based object pose in 25 lines of code. *Int. Journal of Computer Vision*, 15:123–141, 1995.

[11] J. Denzler and C. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:145–157, 2002.

[12] R. Eidenberger, T. Grundmann, W. Feiten, and R. Zoellner. Fast parametric viewpoint estimation for active object detection. In *Proc. Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 309–314, Seoul, South Korea, 2008.

[13] S. Ekvall and D. Kragic. Receptive field cooccurrence histograms for object detection. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 84–89, Edmonton, Alberta, 2005.

[14] S. Ekvall, D. Kragic, and P. Jensfelt. Object detection and mapping for service robot tasks. *Robotica*, 25(2):175–187, 2007.

[15] V. Ferrari, T. Tuytelaars, and L. VanGool. Integrating multiple model views for object recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 105–112, Washington, DC, 2004.

[16] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[17] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. ALVEY Vision Conf.*, pages 147–151, Manchester, 1988.

[18] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2000.

[19] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[20] K. Kastella. Discrimination gain to optimize detection and classification. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 27:112–116, 1997.

[21] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 506–513, Washington, DC, 2004.

[22] C. Kreucher, K. Kastella, and A. O. Hero III. Sensor management using an active sensing approach. *Signal Processing*, 85:607–624, 2005.

[23] A. Kushal and J. Ponce. Modeling 3d objects from stereo views and recognizing them in photographs. In *Proc. European Conf. on Computer Vision*, volume 3954, pages 563–574, Graz, Austria, 2006.

[24] C. Laporte and T. Arbel. Efficient discriminant viewpoint selection for active bayesian recognition. *Int. Journal of Computer Vision*, 68:267–287, 2006.

[25] C. Laporte, R. Brooks, and T. Arbel. A fast discriminant approach to active object recognition and pose estimation. In *Proc. 17th Int. Conf. on Pattern Recognition*, volume 3, pages 91–94, Cambridge, UK, 2004.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. In *Proc. British Machine Vision Conf.*, pages 959–968, London, 2004.

[27] D. Lopez, K. Sjo, C. Paul, and P. Jensfelt. Hybrid laser and vision based object search and localization. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3881–3887, Pasadena, CA, 2008.

[28] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13:441–450, 1991.

[29] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 1150–1157, Kerkyra, Corfu, Greece, 1999.

[30] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60:91–110, 2004.

[31] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 674–679, Vancouver, British Columbia, 1981.

[32] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, 1979.

[33] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. European Conf. Computer Vision*, volume 65, pages 128–142, Copenhagen, Denmark, 2002.

[34] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-3, Robotics Institute, Carnegie-Mellon University, September 1980.

[35] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *Int. Journal of Computer Vision*, 73(3):263–284, 2007.

[36] H. Najafi, G. Yakup, and N. Nassir. Fusion of 3d and appearance models for fast object detection and pose estimation. In *Proc. Asian Conf. on Computer Vision*, volume 3852, pages 415–426, Hyderabad, India, 2006.

[37] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2161–2168, New York, NY, 2006.

[38] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31:71–86, 2000.

[39] L. Paletta, M. Prantl, and A. Pinz. Learning temporal context in active object recognition using bayesian analysis. In *Proc. 15th Int. Conf. on Pattern Recognition*, volume 1, pages 695–699, Barcelona, Spain, 2000.

[40] G. Panin and A. Knoll. Fully automatic real-time 3d object tracking using active contour and appearance models. *Journal of Multimedia*, 1:62–70, 2006.

[41] M. Pellkofer and E. D. Dickmanns. Ems-vision: Gaze control in autonomous vehicles. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 296–301, Dearborn, Michigan, 2000.

[42] L. Petersson, P. Jensfelt, D. Tell, M. Strandberg, D. Kragic, and H. I. Christensen. Systems integration for real-world manipulation tasks. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2500–2505, Washington, DC, 2002.

[43] J. Ponce, S. Lazebnik, F. Rothganger, and C. Schmid. Toward true 3d object recogntion. In *Proc. CVPR Workshop on Generic Object Recognition and Categorization, IEEE Computer Society*, Washington, DC, 2004.

[44] E. Rivlin, Y. Aloimonos, and A. Rosenfeld. Purposive recognition: An active and qualitative approach. In *Proc. SPIE*, pages 225–240, Munich, Germany, 1991.

[45] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. Journal Computer Vision*, 66(3):231–259, 2006.

[46] F. Saidi, O. Stasse, and K. Yokoi. A visual attention framework for search behavior by a humanoid robot. In *Proc. IEEE RAS/RSJ Conf. on Humanoid Robots*, pages 346–351, Genova, Italy, 2006.

[47] F. Saidi, O. Stasse, and K. Yokoi. Active visual search by a humanoid robot. In *Proc. IEEE/RAS Int. Conf. on Advanced Robotics*, pages 360–365, Jeju, South Korea, 2007.

[48] W. Schmaedeke and K. Kastella. Information-based sensor management and immkf. In *Proc. of SPIE (Int. Society for Optical Engineering)*, volume 3373, pages 390–401, San Diego, California, 1998.

[49] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[50] A. Sethi, D. Renaudie, D. J. Kriegman, and J. Ponce. Curve and surface duals and the recognition of curved 3d objects from their silhouette. *Int. Journal Computer Vision*, 58:73–86, 2004.

[51] V. A. Sujan and S. Dubowsky. Efficient information-based visual robotic mapping in unstructured environments. *Int. Journal of Robotics Research*, 24:275–293, 2005.

[52] R. Sukthankar, D. Pomerleau, and C. Thorpe. Panacea: An active sensor controller for alvinn autonomous driving system. In *Proc. Int. Symposium on Robotics Research*, Hidden Valley, Pennsylvania, 1993.

[53] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, 2005.

[54] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26:1385–1391, 2004.

[55] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR-95-041, Department of Computer Science, University of North Carolina, University of North Carolina at Chapel Hill, Chapel Hill, NC, July 2006.

[56] Y. Ye and J. K. Tsotsos. Sensor planning for 3d object search. *Comp. Vision and Image Understanding*, 73:145–168, 1999.