Australian National University

ENGN4718
Engineering Research and Development Project
2016

Rigid Body State Estimation From Sparse Range Measurements

Montiel Abello
u5162235

Supervised by Dr Jochen Trumpf and Dr Viorela Ila

Acknowledgements

I would like to thank my supervisors Dr Jochen Trumpf and Dr Viorela Ila for their guidance over the course of this project. Thank you for being so generous with your time and providing so much encouragement and support.

Abstract

The aim of this research was to design and implement an observer to estimate the state of a rigid cube from sparse range measurements. This was motivated by active research in the development of a design methodology for symmetry-preserving, infinite dimensional observers. The simplified cube estimation problem serves as a first step in this research direction, investigating the use of a sparse sensor in dense estimation and identifying effective innovation functions for rigid body state estimation.

A Matlab toolbox was developed to simulate measurements of rigid bodies by a Hokuyo UBG-04LX-F01 scanning laser range-finder. This simulation was used to assess the performance of the observer. Experimental data was collected to more realistically simulate the error distribution of the sensor and for future testing of observers under real-world conditions.

The simulation results show that the observer is almost globally convergent when correcting orientation and size error. An $\mathbf{SE}(3)$ invariant innovation function would be required to concurrently correct position and orientation, showing the importance of symmetry considerations in nonlinear observer design. This research also demonstrates that dense estimation can be performed using sparse range measurements, given a suitable sensor trajectory.

Contents

List of Figures

Aims and contributions

The aims of this research were to:

- design an observer to estimate the state of a rigid cube from range measurements;
- investigate whether sparse range measurements can be used to make a dense estimation of an infinite dimensional state;
- assess the performance of the observer through simulated and experimental testing;
- identify directions for future research in infinite-dimensional, symmetry-preserving observer design.

The outcomes of this project include:

- demonstration that sparse range measurements can be used in a dense estimation problem by designing an appropriate sensor trajectory;
- developing a noise model for the Hokuyo UBG-04LX-F01 scanning laser range-finder that is more complete under the conditions tested than those provided in existing literature;
- developing a Matlab toolbox to simulate scanning laser range-finder measurements of rigid bodies and test observer implementations;
- the design of an observer that estimates the state of a cube from sparse range measurements. Performance assessment showed that the observer is almost globally convergent when correcting the orientation and size of a stationary cube. Positive results have also been achieved with tracking rotating cubes and correcting position error in special cases. The observer does not rely on the cube's geometry and could potentially be used to estimate the states of a wider class of rigid bodies;
- the collection of real-world range measurements of a rigid body moving with known

trajectory for use in future observer performance testing;

- identifying the adaptation of the current observer update to an $\mathbf{SE}(3)$ invariant function as a logical step in developing a theory of symmetry-preserving, infinite-dimensional observers.

Functions used to compute conversions between rotation representations provided by my supervisor Dr Viorela Ila, as well as functions from the Matlab Aerospace Toolbox were used in the observer simulation. Matlab's Curve Fitting Toolbox was used to fit surfaces to simulated and experimental data. All other contributions to the outcomes described above were my own.

Chapter 1

Introduction

Advances in manufacturing and hardware design have made mobile robots more accessible for not only research and industrial purposes, but to the general public. However, autonomous robots have largely been limited to indoor, carefully controlled environments. Before autonomous robots can be more widely deployed, they must be able to accurately observe and represent unstructured, dynamic environments. Dense sensors such as light field cameras are becoming cheaper and lighter, promising to allow autonomous robots to acquire detailed measurements of complex environments. To fully utilise the potential of these advancing technologies, improvements in observer design are required to generate more accurate and detailed descriptions of these environments from dense measurements.

One method of estimating the state of a complex environment is with an *infinite-dimensional observer*. Typically, observers for infinite-dimensional systems are extensions of finite-dimensional Luenberger observers. Unfortunately, this design approach is only able to guarantee convergence for linear systems. Developing a theory of symmetry-preserving, infinite-dimensional observers would simplify the design process for nonlinear systems and result in observers with improved convergence properties.

This research project serves as an initial exploration into the design of symmetry-preserving, infinite-dimensional observers. An infinite dimensional system will be simplified and an observer will be designed to estimate a finite dimensional state. The potential for a symmetry-preserving, infinite-dimensional observer to improve performance will be explored.

This report presents the novel implementation of an observer to estimate the state of a rigid cube from range measurements. Section 1.1 reviews the current state of observer design methods for infinite-dimensional systems. Recent work in the development of design methodologies for symmetry preserving observers is described. Particular attention is paid to a dense optical flow estimator that will be particularly relevant to this research.

Chapter 2 provides theory on Lie groups, rigid body state representation and state observer concepts that will be relevant in the observer design.

The cube state estimation problem that is the focus of this research is described in detail in Chapter 3. The place of this problem within the larger area of symmetry-preserving, infinite-dimensional observer research is defined.

Chapter 4 provides a detailed description of the simulation implementation, including the observer update function design. The performance of the observer in estimating the state of stationary and moving cubes is assessed. It is shown that almost global convergence is achieved for orientation and size correction for stationary cubes, though the position update only converges in special cases.

Chapter 5 details steps taken to experimentally validate the results of the simulation. Range measurement data is collected and used to develop an error distribution model for the Hokuyo UBG-04LX-F01 sensor. Range measurements of a cube of known trajectory are taken for future performance testing.

1.1 Literature review

The use of dense sensors allows for a more accurate estimation of the state of an infinite-dimensional system such as a complex, real-world environment. The theory of infinite-dimensional observers is required to fully utilise this information. This section will review the current state of design methodologies and implementations for infinite-dimensional observers. Particular focus will be paid to an emerging avenue of research; symmetry-preserving observer design. Recent theory developments in this area have allowed limitations in the global convergence properties of nonlinear observers to be overcome.

1.1.1 Infinite-dimensional observers

In many real world systems the dependent variables are functions of one or more spatial variables. An example would be the dynamics of waves in a body of water. The height of the surface varies continuously along the x and y directions. These spatial variables vary continuously, meaning an infinite number of parameters is required to describe the state of the system. Such systems are termed *infinite-dimensional systems*, or *distributed parameter systems*. Their dynamics are modelled by a partial differential equation (PDE).

When a state estimate is required but direct measurement of the state with sensors is difficult or impossible, a *state observer* is employed. A state observer is a filter that provides an estimate of the state of a system using the difference between its measured and predicted outputs. A more detailed description of the concept of a state observer is provided in Section 2.2. An observer for an infinite-dimensional system is called an *infinite-dimensional observer*.

1.1.1.1 Linear systems

Observer theory for *linear* infinite-dimensional systems has been widely studied. The techniques used are typically extensions of Luenberger observers and Kalman filter methods used to observe finite dimensional system.

A simplified approach is to use a spatial discretisation method such as finite difference or finite element to reduce the infinite-dimensional system to a finite-dimensional one. From here, finite-dimensional observer design techniques can be used. This is known as the *early lumping* method, and was employed by Stavroulakis [1] who implemented a finite-dimensional observer as part of a control system for an infinite dimensional systems.

The early lumping approach suffers from *spillover*, a phenomenon where performance is affected by the neglected dynamics of the system[2]. Harkort [3] recently developed an observer based control scheme that reduced this effect by using modelled outputs rather than true measurements to reduce the effect of the neglected dynamics.

More accurate observers can be designed with the *late lumping* approach which uses the infinite-dimensional model of the system in the observer design. The result is an infinite-dimensional observer that is discretised later for practical implementation. These methods are typically extensions of Kalman or Luenberger methods to infinite dimensions.

Early work by Gressang [4] extended the Luenberger observer to infinite-dimensional systems whose state space was an abstract Banach space with dynamics defined by an infinitesimal generator of a semigroup. More recently, Smyshlyaev [5] developed an exponentially converging backstepping observer for systems governed by parabolic PDEs. Ramdani introduced forward and backward observers [6] whose convergence properties were investigated by Haine [7].

1.1.1.2 Nonlinear systems

There is currently no universal approach for observer design for nonlinear infinite-dimensional systems. The most common approach has been to linearise the system, then apply a linear infinite-dimensional observer design. Common linearisation methods include Lyapunov methods, extended linearisation and the Lie-algebraic approach [8].

There has been some progress in infinite-dimensional observer design for special cases of nonlinear systems. For bilinear systems, Xu [9] designed an infinite-dimensional observer that converged for certain inputs. Bounit [10] designed Kalman and Luenberger type observers for infinite-dimensional bilinear systems.

Despite these small advances in special cases of nonlinear design, the most common design methods for nonlinear infinite-dimensional systems are based on linearisation techniques. These techniques rely on the fact that differentiable functions can be approximated by a first-order Taylor expansion around a point. Luenberger and Kalman methods can be applied to linear approximations of infinite-dimensional systems around an equilibrium point. This simplification relies on the dynamics of system at the point of linearisation being representative of the entire space. In general, this is not necessarily true, and is the biggest limitation in this design technique. The result is that these linearised observers only converge if the initial state estimate is within a local neighbourhood of the true state. Global converge is not guaranteed which severely limits robustness.

Global convergence can be achieved by taking account the symmetries inherent to the system during observer design. A powerful tool for dealing with symmetries is the theory of *Lie groups*. Investigation into *symmetry-preserving* observer design for systems on Lie groups is an active area of research. It promises to produce theoretically validated design principles for nonlinear infinite-dimensional observers, though the majority of research so far has been limited to finite-dimensional observers.

1.1.2 Symmetry-preserving observers

The motivation behind symmetry-preserving observers is to take advantage of invariances in the dynamics of the system. The goal is to design an observer around an equilibrium point in such a way that it can be extended to converge around a wider set of points.

1.1.2.1 Early work

Geometry conscious observer design is not a new idea. Early investigation by Marcus [11] into algebraic and geometric methods for nonlinear filter design showed promise. A seminal work by Salcudean [12] was the design of an eventually-exponential, globally converging observer for the attitude of rigid bodies from orientation and torque measurements. This observer design took advantage of the simplicity of the quaternion rotation representation and dynamics of rigid body motion.

Another important result that is a precursor to the active research of today is a design method developed by Aghannon & Rouchon [13]. Their invariant observer construction was based on Cartan's moving frame method. Though convergence was proven for a specific problem, the observer convergence for a general case was left an open problem. Maithripala [14] demonstrated the effectiveness of Aghannon & Rouchons' method by incorporating it into the design of an intrinsic observer based controller. Performance was shown to be independent of the coordinate system used to represent the configuration space.

1.1.2.2 Active research

There are currently two groups actively researching symmetry-preserving observer design. Both have begun to apply symmetry-preserving methods to infinite-dimensional observers.

The work of Bonnabel, Auroux, Rouchon, Martin et al. is a progression of the early results from Aghannon & Rouchon. Their general approach is to first design a Luenberger type observer around an equilibrium point. An invariant frame is used to construct an invariant output error. The observer innovation term respects the symmetries of the system and thus the nonlinear observer is well behaved around a continuum of equilibrium points.

In [15], Bonnabel et a. developed an observer design procedure based on Aghannon & Rouchons' work. Asymptotic stability was achieved, though this required a design procedure tailored to specific nonlinearities of the system and did not apply in a general case. It was shown in [16] that an invariant error equation simplified convergence analysis. The observer's global behaviour improved, having a larger region of attraction in comparison to naively linearised observers. Developments were made to the theory and presented in [17]. For a particular class of invariant system it was shown that the observer converged locally around any trajectory, and global convergence behaviour was independent of trajectory.

Most recently, these invariant design methods were applied to an infinite dimensional system [18]. An observer estimating the state of fluid in a water tank where height varied with the continuous dependent variables position and time was developed. It was shown to converge more quickly and robustly than previous attempts to design infinite dimensional observers with Extended Kalman Filter (EKF) methods.

The work of Trumpf, Mahony, Hamel, Lageman et al. differs in scope. The methods of Bonnabel et al. are generalised and can be applied to a wide range of systems. In contrast, the work of Trumpf et al. is limited to two specific classes of systems but achieves stronger convergence properties. In [19] nonlinear filters on the Special Orthogonal Group $\mathbf{SO}(3)$ are used in attitude estimation and the resulting nonlinear observers achieved almost globally stable observer error. Another attitude observer [20] achieved almost globally asymptotic and locally exponential convergence. In [21], the design methodology for 2 classes of systems is presented. The approach taken is to lift the kinematic system onto its symmetry group and design an observer for the lifted system. The Lyapunov method is used to design the observer innovation term. This methodology simplifies nonlinear observer design and produces observers with strong convergence properties. This group has also begun research on symmetry-preserving infinite-dimensional observers.

The motivation behind the development of infinite-dimensional observers is to allow dense sensors to be fully utilised. In this vein, research presented in a PhD thesis by Zarrouati [25] utilised dense measurements from a camera and depth sensor. An observer was developed from rotation invariant equations for light and depth. Though the sensors took measurements of an infinite dimensional state, a finite dimensional approximation of this state is what was estimated by the observer.

Another recent work by Adarve et al.[26] also uses dense sensing in the estimation of an infinite dimensional state. This result will be examined closely as it is similar in direction to this research.

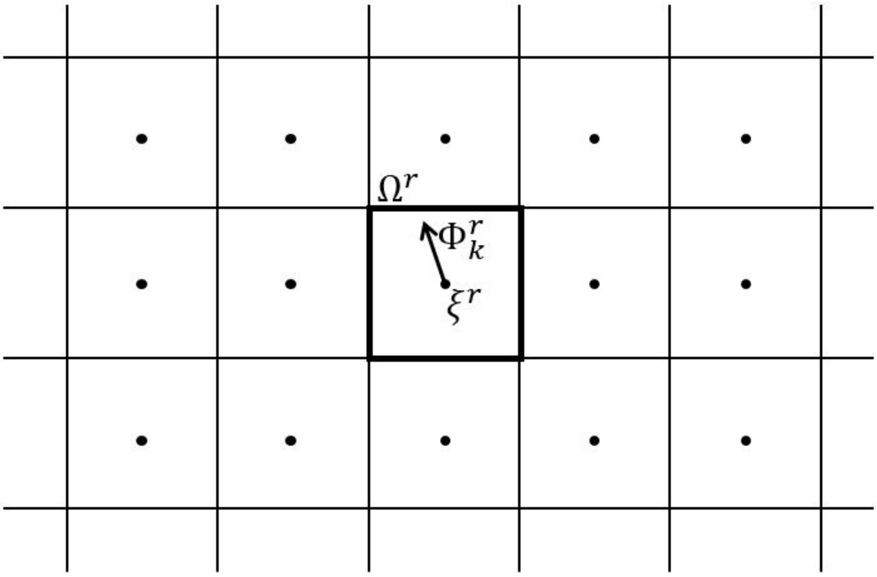


Figure 1.1: Infinite dimensional optical flow discretised and computed in separate, independent regions

Adarve et al. design an update-propagation filter to iteratively compute dense optical flow Φ from CCD camera measurements Y. In reality, this optical flow is an infinite dimensional state. Rather than computing the optical flow independently at each frame, a two-stage process is used to build it incrementally. The propagation stage uses a non-linear PDE to model the transport of the optical flow in the next time step. The update stage corrects this prediction using the current image.

The iterative filter used in this approach is an observer that estimates the state of the continuous spatio-temporal flow field. By using a dense sensor, the measured image stream can be treated as a continuous, infinite-dimensional state. This is in contrast to sparse optical flow computation where the image is modelled as a set of discrete pixel values. However, the flow field Φ is discretised and computed in r independent regions Ω around a discrete set of control points ξ as shown in Figure 1.1.2.2. Here, this approach differs from that of general infinite-dimensional invariant observer. The state is treated as a discrete set of locally continuous states which does not allow for symmetry considerations. This is because the PDE relations in the local regions are invariant to 2D rotation and translation, but the interactions between regions themselves are not.

Discretising Φ and Y at the beginning of the algorithm design makes this is an example of the early lumping design approach. Employing a late lumping approach by discretising an infinite dimensional observer would allow for the rotation and translation invariance of the flow field to be taken advantage of to improve convergence properties.

The lesson to take from this analysis is that discretisation methods must be carefully chosen in order to preserve the invariance of the observer.

Another reason to pay attention to geometric symmetries in observer design is due to limitations placed on convergence by the topology of the system. Bhat & Bernstein [22] show that global convergence cannot be achieved with a continuous observer on a state space that includes a vector bundle such as $\mathbf{SE}(3)$. Some advancements have been made with extended state-space observers [23, 24], that extend the state space. However, this scheme can produce state estimates that are incompatible with the physics of the system prior to convergence. A theory of symmetry-preserving observer design for infinite dimensional systems could simplify observer design and convergence analysis for such systems.

Chapter 2

Theoretical background

2.1 Rigid body kinematics

A rigid body is a model of a solid object whose deformation is assumed to be negligible. The distance between every pair of points on the body remains constant. Because such a body does not deform, knowledge of the orientation and position of a reference frame fixed to a rigid body constitutes knowledge of the position of all points. The position of the rigid body is thus defined by the position of a particular point in the body, most commonly its centre of mass. The orientation can be defined using a set of coordinate axes fixed to the body such that its origin coincides with the representative position point. The theory of Lie groups will be used to describe the kinematics of rigid bodies in this report.

2.1.1 Lie groups

A Lie group $\mathbf{G}$ is a group whose elements form a differentiable manifold and whose group operation and inverse operation are differentiable. As a group, $\mathbf{G}$ is a set of elements and a group operation. This group operation is a binary operation that combines two elements and is denoted by multiplication: AB or $A \cdot B$ for $A, B \in \mathbf{G}$. Because it is a group, $\mathbf{G}$ satisfies the 4 group axioms:

- **Closure:** The group operation $\mathbf{G} \times \mathbf{G} \mapsto \mathbf{G}$ is a function that maps elements of $\mathbf{G}$ onto itself; $\forall A, B \in \mathbf{G},\ AB \in \mathbf{G}$.
- **Associativity:** Elements of G are associative under the group operation; $\forall A, B, C \in \mathbf{G}$, $(AB)C = A(BC)$.

- **Identity:** There exists an identity element $I \in \mathbf{G}$ such that $\forall A \in \mathbf{G}$, $IA = AI = A$.
- **Inverse:** For all $A \in \mathbf{G}$ there exists an inverse element $A^{-1} \in \mathbf{G}$ such that $AA^{-1} = A^{-1}A = I$.

Because the Lie group $\mathbf{G}$ is a differentiable manifold, it is locally Euclidean. This means that the neighbourhood around every element of $\mathbf{G}$ can be approximated with a tangent plane. This property allows calculus to be performed on elements of $\mathbf{G}$.

2.1.1.1 Matrix Lie groups

A matrix Lie group $\mathbf{G} \subset \mathbf{GL}(n)$ is made up of group elements which are $n \times n$ matrices. This work will focus on matrix Lie groups because the form of the exponential map and Lie bracket functions provided below only apply to such Lie groups. Generalised concepts for these functions exist, but a more detailed and relevant description can be given by focusing on matrix Lie groups.

2.1.1.2 Lie algebra

The tangent space at the identity element of a Lie group is called the Lie algebra $\mathfrak{g}$. It is called the Lie *algebra* because it has a binary operation, known as the Lie bracket $[X, Y]$. For matrix Lie groups the Lie bracket is

$$[A, B] \triangleq AB - BA \tag{2.1}$$

2.1.1.3 The exponential map and logarithm map

The canonical mapping from the Lie algebra $\mathfrak{g}$ to the Lie group $\mathbf{G}$ is called the exponential map.

$$\exp : \mathfrak{g} \to \mathbf{G} \tag{2.2}$$

Similarly, the logarithm map maps elements from its domain $\mathbf{D} \subset \mathbf{G}$ to $\mathfrak{g}$

$$\log : \mathbf{D} \to \mathfrak{g} \tag{2.3}$$

such that for a group element A,

$$\exp(\log(A)) = A \tag{2.4}$$

For matrix Lie groups, the exponential map and logarithm map correspond to the matrix exponential and matrix logarithm respectively.

2.1.1.4 Infinitesimal generators

The *hat* operator $(\cdot)^\wedge$ can be used to map an n-vector to an $m \times m$ matrix representation, when $\mathbb{R}^{m\times m}$ is isomorphic to $\mathbb{R}^n$.

$$\begin{aligned} (\cdot)^\wedge : \mathbb{R}^n &\to \mathbb{R}^{m\times m} \\ x \mapsto x^\wedge &= \sum_{i=1}^{n} x_i G_i \end{aligned} \tag{2.5}$$

where the set of elements G_i form a basis for $\mathbb{R}^{m\times m}$.

Conversely, the *vee* operator $(\cdot)^\vee$ maps matrices in $\mathbb{R}^{m\times m}$ to vectors in $\mathbb{R}^n$ such that $(x^\wedge)^\vee = x$

$$\begin{aligned} (\cdot)^\vee : \mathbb{R}^{m\times m} &\to \mathbb{R}^n \\ x^\wedge &\mapsto x \end{aligned} \tag{2.6}$$

For an n-dimensional matrix Lie group, the Lie algebra $\mathfrak{g}$ is a vector space isomorphic to $\mathbb{R}^n$. The hat operator $(\cdot)^\wedge$ maps vectors $x \in \mathbb{R}^n$ to elements of $\mathfrak{g}$. For a matrix Lie group $\mathbf{G}$ whose elements are $m \times m$ matrices, the elements of $\mathfrak{g}$ will also be $m \times m$ matrices.

$$\begin{aligned} (\cdot)^\wedge : \mathbb{R}^n &\to \mathfrak{g} \\ x \mapsto x^\wedge &= \sum_{i=1}^{n} x_i G_i \end{aligned} \tag{2.7}$$

The basis elements G_i are $m \times m$ matrices known as the infinitesimal generators of $\mathbf{G}$.

2.1.1.5 Lie bracket and group operation

For Lie groups endowned with the commutative property ($\forall A, B \in \mathbf{G}, AB = BA$), vector addition in the Lie algebra maps to a group operation in the Lie group. For $C = A + B$

where $A, B, C \in \mathfrak{g}$,

$$e^C = e^{A+B} = e^A e^B \tag{2.8}$$

For non-commutative Lie groups, this relationship between the Lie algebra and Lie group operations does not hold. Instead, for $C = \log\left(e^A e^B\right)$, C is calculated with the Baker-Campbell-Hausdorff formula:

$$C = A + B + \frac{1}{2}[A, B] + \frac{1}{12}[A - B, [A, B]]\frac{1}{24}[B, [A, [A, B]]] + \dots \tag{2.9}$$

2.1.1.6 Actions

When a group action for a Lie group $\mathbf{G}$ acting on a manifold M is a differentiable map, this is known as a *Lie group action*. For example, 3D rotations act on 3D points so the Lie group $\mathbf{SO}(3)$ acts on $\mathbb{R}^3$. A left action of $\mathbf{G}$ on M is defined as a differentiable map

$$\Phi : \mathbf{G} \times M \mapsto M \tag{2.10}$$

where

- the identity element I acts as the identity on M

$$\Phi(I, m) = m \qquad \forall m \in M \tag{2.11}$$

- Group actions compose according to

$$\Phi(m, \Phi(n, o)) = \Phi(mn, o) \qquad \forall m, n, o \in M \tag{2.12}$$

2.1.2 SO(3)

A rotation represents the motion of a point about the origin of a Euclidean space. In $\mathbb{R}^3$ this is a proper isometry: a transformation that preserves distances between any pair of points and has a determinant of $+1$. The set of all rotations about the origin of $\mathbb{R}^3$ is known as the *special orthogonal group* $\mathbf{SO}(3)$. Group elements of $\mathbf{SO}(3)$ can be represented using a special subset of 3×3 invertible matrices and in this case, form a matrix Lie group. Several rotation representations are described later in this section, but the theory presented below only applies to matrix Lie groups which rely on the rotation matrix representation for group elements.

A rotation matrix $\mathbf{R}$ is a 3×3 matrix that performs a rotation operation when it acts on an element of $\mathbb{R}^3$. The properties of $\mathbf{R}$ are described in more detail in Section 2.1.2.4.

2.1.2.1 Lie algebra

The Lie algebra $\mathfrak{so}(3)$ is a vector space whose elements represent angular velocities. These elements can be represented with 3×3 skew-symmetric matrices $\boldsymbol{\omega}^\wedge$, where $\boldsymbol{\omega} \in \mathbb{R}^3$ is a 3-vector representing an angular velocity. For $\boldsymbol{\omega} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$, the skew symmetric representation is given by taking the hat representation of $\boldsymbol{\omega}$

$$\boldsymbol{\omega}^\wedge = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.13}$$

Elements of $\mathfrak{so}(3)$ are mapped to $\mathbf{SO}(3)$ according to the exponential map:

$$\begin{aligned} \exp : \mathfrak{so}(3) &\to \mathbf{SO}(3) \\ \boldsymbol{\omega}^\wedge &\mapsto \exp(\boldsymbol{\omega}^\wedge) \end{aligned} \tag{2.14}$$

where the matrix $\exp(\boldsymbol{\omega}^\wedge) \in \mathbf{SO}(3)$ is a rotation matrix $\mathbf{R}$.

Conversely, the logarithm map maps 3×3 rotation matrices of $\mathbf{SO}(3)$ to elements of $\mathfrak{so}(3)$:

$$\begin{aligned} \log : \mathbf{SO}(3) &\to \mathfrak{so}(3) \\ \exp(\boldsymbol{\omega}^\wedge) &\mapsto \boldsymbol{\omega}^\wedge \end{aligned} \tag{2.15}$$

This means that for a rotation matrix $\mathbf{R}$, $\log(\mathbf{R}) \in \mathfrak{so}(3)$ and represents an angular velocity.

2.1.2.2 Actions

By the group action, elements of $\mathbf{SO}(3)$ rotate points $\mathbf{p} \in \mathbb{R}^3$ about the origin.

$$\begin{aligned} \Phi : \mathbf{SO}(3) \times \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ (\mathbf{R}, \mathbf{p}) &\mapsto \mathbf{Rp} \end{aligned} \tag{2.16}$$

2.1.2.3 Rotation representations

There are many conventions by which elements of $\mathbf{SO}(3)$ can be represented. The representations that will be used in this report are described below.

2.1.2.4 Rotation matrices

A 3D rotation matrix $\mathbf{R}$ is an orthogonal 3×3 matrix with a determinant of $+1$. Since $\mathbf{R}$ is orthogonal, its columns and rows are respectively sets of orthogonal unit vectors and

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.17}$$

The group operation using rotation matrices is simply a matrix multiplication which concatenates the two rotations. The product of two rotation matrices $\mathbf{R}_3 = \mathbf{R}_2\mathbf{R}_1$ is a rotation matrix corresponding to left multiplication by $\mathbf{R}_1$ followed by $\mathbf{R}_2$.

The left action of a rotation matrix $\mathbf{R}$ on a point $\mathbf{p} \in \mathbb{R}^3$ is a left matrix multiplication that rotates $\mathbf{p}$ about the origin.

2.1.2.5 Scaled-axis representation

An orientation in $\mathbb{R}^3$ can also be represented by a 3-vector $\boldsymbol{\theta}$ whose direction $\mathbf{r}$ represents the axis of rotation and magnitude θ represents the angle of rotation.

$$\boldsymbol{\theta} = \theta\mathbf{r} \tag{2.18}$$

Though scaled-axis vectors are not typically used to perform rotations, Rodrigues' rotation formula efficiently converts scaled-axis vectors to rotation matrices:

$$\mathbf{R}_{\boldsymbol{\theta}} = \mathbf{I} + [\mathbf{r}]_{\times} \sin\theta + ([\mathbf{r}]_{\times})^2 (1 - \cos\theta) \tag{2.19}$$

Elements of $\mathfrak{so}(3)$ are typically represented with the hat representation $\boldsymbol{\omega}^{\wedge}$ of a scaled-axis vector $\boldsymbol{\omega}$, where the magnitude ω corresponds to the angular velocity about the axis $\mathbf{r}$.

2.1.2.6 Rotation quaternions

Quaternions are an extension of complex numbers. The set of unit quaternions can be used to represent $\mathbf{SO}(3)$, and will be referred to as the set of *rotation quaternions* or *orientation quaternions*. A rotation quaternion $\mathbf{q}$ is a 4-tuple of real numbers that encode the same information as the scaled-axis representation. $\mathbf{q}$ is often described in terms of its first element w - the scalar part, and the remaining elements x, y and z - the vector part. Given an axis of rotation $\mathbf{r}$ and an angle of rotation θ:

$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{r} \end{bmatrix} \tag{2.20}$$

In general, the quaternion inverse is given by

$$\mathbf{q}^{-1} = \frac{1}{w^2 + x^2 + y^2 + z^2} \begin{bmatrix} w \\ -x \\ -y \\ -z \end{bmatrix} \tag{2.21}$$

For unit magnitude rotation quaternions the inverse represents a rotation by $-\theta$ and is given by

$$\mathbf{q}^{-1} = \begin{bmatrix} \cos(\theta/2) \\ -\sin(\theta/2)\mathbf{r} \end{bmatrix} = \begin{bmatrix} w \\ -x \\ -y \\ -z \end{bmatrix} \tag{2.22}$$

The group operation is performed with quaternion multiplication which is defined:

$$\mathbf{q}_1\mathbf{q}_2 = \begin{bmatrix} w_1 \\ \mathbf{v}_1 \end{bmatrix} \cdot \begin{bmatrix} w_2 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 \\ w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{bmatrix} \tag{2.23}$$

As with rotation matrices, quaternion multiplication is associative but not commutative.

The group action rotates a point $\mathbf{p}_0 \in \mathbb{R}^3$ to $\mathbf{p}_1 \in \mathbb{R}^3$ by embedding it as the vector component of a quaternion and applying a conjugation operation with $\mathbf{q}$. The rotated vector $\mathbf{p}_1$ can be extracted as the vector component of the resulting quaternion.

$$\begin{bmatrix} 0 \\ \mathbf{p}_1 \end{bmatrix} = \mathbf{q} \begin{bmatrix} 0 \\ \mathbf{p}_0 \end{bmatrix} \mathbf{q}^{-1} \tag{2.24}$$

2.1.3 SE(3)

The *special Euclidean group* $\mathbf{SE}(3)$ represents rigid transformations in $\mathbb{R}^3$. This is a matrix Lie group whose elements comprise the set of all rigid transformations in $\mathbb{R}^3$ and can be represented with 4×4 matrices of the form

$$\mathbf{S} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.25}$$

where $\mathbf{R} \in \mathbf{SO}(3)$ and $\mathbf{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^\top \in \mathbb{R}^3$.

$\mathbf{SE}(3)$ is a semidirect product of $\mathbf{SO}(3)$ and $\mathbb{R}^3$. As its group elements contain a rotation matrix and translation vector, $\mathbf{SE}(3)$ has 6 degrees of freedom and is a 6-dimensional manifold.

2.1.3.1 Lie algebra

The Lie algebra $\mathfrak{se}(3)$ is a vector space whose elements are 4×4 matrices of the form

$$\begin{bmatrix} \boldsymbol{\omega}^\wedge & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \tag{2.26}$$

where $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^\top \in \mathbf{so}(3)$, representing an angular velocity in scaled axis representation, and $\mathbf{v} = \begin{bmatrix}v_x & v_y & v_z\end{bmatrix}^\top \in T\mathbb{R}^3 \equiv \mathbb{R}^3$, representing a linear velocity vector.

Elements of $\mathfrak{se}(3)$ are mapped to $\mathbf{SE}(3)$ according to the exponential map:

$$\begin{gathered}\exp : \mathfrak{se}(3) \to \mathbf{SE}(3) \\ \begin{bmatrix}\boldsymbol{\omega}^\wedge & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0\end{bmatrix} \mapsto \begin{bmatrix}\mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}\end{gathered} \tag{2.27}$$

This means that $\forall \mathbf{T} \in \mathfrak{se}(3),\ \exp(\mathbf{T}) \in \mathbf{SE}(3)$

Conversely, the logarithm map maps elements of $\mathbf{SE}(3)$ to elements of $\mathfrak{se}(3)$:

$$\begin{gathered}\log : \mathbf{SE}(3) \to \mathfrak{se}(3) \\ \begin{bmatrix}\mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix} \mapsto \begin{bmatrix}\boldsymbol{\omega}^\wedge & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0\end{bmatrix}\end{gathered} \tag{2.28}$$

This means that $\forall \mathbf{S} \in \mathbf{SE}(3),\ \log(\mathbf{S}) \in \mathfrak{se}(3)$

2.1.3.2 Actions

$\mathbf{SE}(3)$ group elements act to perform a rigid transformation on points in $\mathbb{R}^3$. This corresponds to a rotation about the origin and a translation. To apply a transformation using the 4×4 matrix elements of $\mathbf{SE}(3)$ to a point $\mathbf{p} = (x, y, z)$ in $\mathbb{R}^3$, the point must be represented with homogeneous coordinates as $\mathbf{p}'$

$$\mathbf{p}' = \begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix} = \begin{bmatrix}x \\ y \\ z \\ 1\end{bmatrix} \tag{2.29}$$

The left group action of $\mathbf{SE}(3)$ is now simply a left matrix multiplication of $\mathbf{p}$:

$$\mathbf{p}'_1 = \mathbf{S}\mathbf{p}'_0 = \begin{bmatrix}\mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}\begin{bmatrix}\mathbf{p}_0 \\ 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}\mathbf{p}_0 + \mathbf{p} \\ 1\end{bmatrix} \tag{2.30}$$

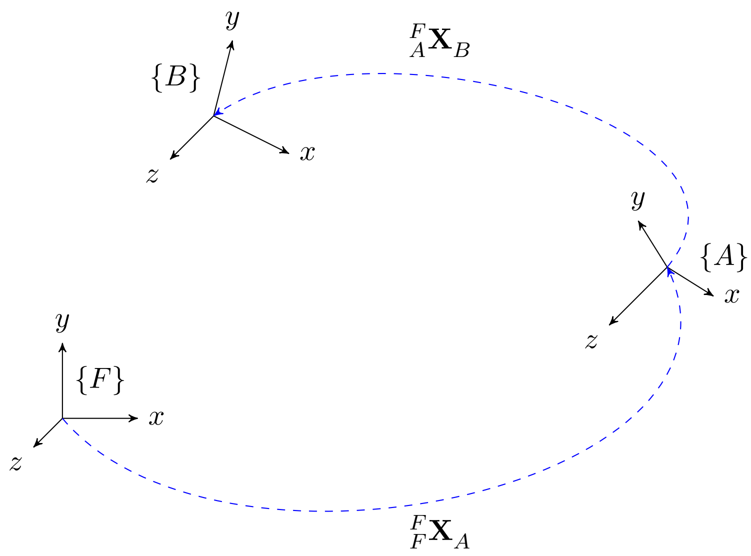


Figure 2.1: Transformations between reference frames $\{F\}$, $\{A\}$ and $\{B\}$ defined with respect to $\{F\}$

2.1.4 Reference frames

A reference frame is a system of coordinates that is used to uniquely identify points on a manifold. This report will deal with reference frames on $\mathbb{R}^3$ that are used both to define the position of a point and the pose of a rigid body in 3D space. Such a reference frame is represented by an element of $\mathbf{SE}(3)$.

The notion of an *inertial reference frame* is introduced here. This will be defined as a reference frame that is stationary for the purpose of the problem being described. The convention used will be to denote the inertial reference frame as $\{F\}$.

Consider the three reference frames shown in Figure 2.1, denoted $\{F\}$ (the inertial frame), $\{A\}$ and $\{B\}$. The notation ${}^F_A\mathbf{X}_B$ defines the transformation in $\mathbf{X}$ of the reference frame $\{B\}$ with respect to the frame $\{A\}$, defined in the frame $\{F\}$.

For example, ${}^F_A\mathbf{R}_B$ defines the rotation of $\{B\}$ with respect to $\{A\}$, defined in the inertial frame $\{F\}$.

2.1.4.1 Pose

A pose defines an orientation and position in space. The pose of a rigid body is represented by a reference frame fixed to a particular point within the body. The pose of the rigid body with respect to another reference frame is defined by the relative position and orientation between the two frames. This transformation can be defined with respect to either reference frame and is represented by an element of $\mathbf{SE}(3)$. If a rigid body has orientation aligned with a reference frame $\{B\}$ and position at the origin of $\{B\}$, then the pose of the rigid body with respect to $\{A\}$ and defined in $\{F\}$ is:

$$
{}^{F}_{A}\mathbf{S}_{B} = \begin{bmatrix} {}^{F}_{A}\mathbf{R}_{B} & {}^{F}_{A}\mathbf{p}_{B} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.31}
$$

2.1.4.2 Point

A point $\mathbf{p} \in \mathbb{R}^3$ in the frame $\{F\}$ is denoted ${}^{F}\mathbf{p}$ and is expressed as a 3-vector of the weights used to compose it from the basis vectors of $\{F\}$.

$$
{}^{F}\mathbf{p} = \begin{bmatrix} {}^{F}x \\ {}^{F}y \\ {}^{F}z \end{bmatrix} \tag{2.32}
$$

2.1.4.3 Homogeneous coordinates

To be acted on by an element of $\mathbf{SE}(3)$, a point must be expressed in homogeneous coordinates.

$$
{}^{F}\mathbf{p}' = \begin{bmatrix} {}^{F}\mathbf{p} \\ 1 \end{bmatrix} \tag{2.33}
$$

2.1.4.4 Redefining the reference frame of a point

Consider a point in $\mathbb{R}^3$ defined with in terms of the frame $\{A\}$. To redefine the point in terms of $\{F\}$, the left action of ${}^{F}_{F}\mathbf{S}_{A} \in \mathbf{SE}(3)$ is used:

$$
{}^{F}\mathbf{p}' = {}^{F}_{F}\mathbf{S}_{A}\, {}^{A}\mathbf{p}' \tag{2.34}
$$

2.1.4.5 Concatenating poses

Poses are concatenated by multiplying relative poses.

$$
{}^{F}_{F}\mathbf{X}_{B} = {}^{F}_{F}\mathbf{X}_{A}\, {}^{A}_{A}\mathbf{X}_{B} \tag{2.35}
$$

2.1.4.6 Redefining the reference frame of a pose

To define a pose transformation matrix in terms of a different reference frame, a matrix conjugation is used.

$$ {}^{A}_{B}\mathbf{X}_{C} = ({}^{A}_{A}\mathbf{X}_{F})\, {}^{F}_{B}\mathbf{X}_{C}\, ({}^{A}_{A}\mathbf{X}_{F})^{-1} \tag{2.36} $$

2.1.4.7 Inverse

Taking the inverse of a pose transformation matrix has the effect of reversing the transformation, but does not alter the frame that the transformation is defined in terms of.

$$ ({}^{F}_{A}\mathbf{X}_{B})^{-1} = {}^{F}_{B}\mathbf{X}_{A} \tag{2.37} $$

2.1.5 Rigid body state representation

The state of a rigid body moving through 3D space can be represented by its linear and angular position, velocity and acceleration. Higher derivatives could be taken but will be ignored for simplicity. The inertial frame is denoted $\{F\}$ and a frame $\{A\}$ is fixed to the pose of the moving body.

The pose of the body with respect to $\{F\}$ at time t, defined in $\{F\}$ is represented by the *screw* matrix ${}^{F}_{F}\mathbf{S}_{A}(t) \in \mathbf{SE}(3)$,

$$ {}^{F}_{F}\mathbf{S}_{A}(t) = \begin{bmatrix} {}^{F}_{F}\mathbf{R}_{A}(t) & {}^{F}_{F}\mathbf{p}_{A}(t) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.38} $$

where ${}^{F}_{F}\mathbf{R}_{A}(t) \in \mathbf{SO}(3)$ is a rotation matrix, and ${}^{F}_{F}\mathbf{p}_{A}(t) \in \mathbb{R}^3$ represents the position of the body defined in $\{F\}$.

The linear and angular velocity of the body with respect to $\{F\}$ at time t, defined in the body-fixed frame $\{A\}$ is represented by the *twist* matrix ${}^{A}_{F}\mathbf{T}_{A}(t) \in \mathfrak{se}(3)$,

$$ {}^{A}_{F}\mathbf{T}_{A}(t) = \begin{bmatrix} {}^{A}_{F}\boldsymbol{\omega}^{\wedge}_{A}(t) & {}^{A}_{F}\mathbf{v}_{A}(t) \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \tag{2.39} $$

where ${}^{A}_{F}\boldsymbol{\omega}_{A}(t) \in \mathfrak{so}(3)$ is an angular velocity in the scaled-axis representation, and the linear velocity is ${}^{A}_{F}\mathbf{v}_{A}(t) \in T\mathbb{R}^3 \equiv \mathbb{R}^3$.

The linear and angular acceleration of the body with respect to $\{F\}$ at time t, defined in the body-fixed frame $\{A\}$, is represented by the *wrench* matrix ${}^{A}_{F}\mathbf{W}_{A}(t) \in T\mathfrak{se}(3) \equiv \mathfrak{se}(3)$,

$$ {}^{A}_{F}\mathbf{W}_{A}(t) = \begin{bmatrix} {}^{A}_{F}\boldsymbol{\alpha}^{\wedge}_{A}(t) & {}^{A}_{F}\mathbf{a}_{A}(t) \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \tag{2.40} $$

where ${}^{A}_{F}\boldsymbol{\alpha}_{A}(t) \in T\mathfrak{so}(3) \equiv \mathfrak{so}(3)$ is an angular acceleration in the scaled-axis representation, and the linear acceleration is ${}^{A}_{F}\mathbf{a}_{A}(t) \in T^2\mathbb{R}^3 \equiv \mathbb{R}^3$.

From this point on, frames will dropped in the notation. For a body labelled x fixed to a frame $\{A\}$, ${}^{F}_{F}\mathbf{S}_{A}$, ${}^{A}_{F}\mathbf{T}_{A}$ and ${}^{A}_{F}\mathbf{W}_{A}$ will be denoted $\mathbf{S}_x$, $\mathbf{T}_x$ and $\mathbf{W}_x$.

2.1.6 Rigid body kinematics

The dynamics of the screw, twist and wrench matrices as they are defined in section 2.1.5 are governed by the following ordinary differential equations (ODEs),

$$ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{S}(t) = \mathbf{S}(t)\mathbf{T}(t) \tag{2.41} $$

$$ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}(t) = \mathbf{W}(t) \tag{2.42} $$

$$ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{W}(t) = \mathbf{f}(t) \tag{2.43} $$

where the function $\mathbf{f}(t)$ is known.

2.1.7 Scanning laser range-finder model

A scanning laser range-finder is a sensor that measures the distance to the nearest object in a certain direction. Consider such a sensor fixed to a moving rigid body. The pose of the sensor labelled s is defined by $\mathbf{S}_s$, $\mathbf{T}_s$ and $\mathbf{W}_s$. The scanning direction of the sensor defined in the body fixed frame $\{A\}$ is a unit vector ${}^{A}\mathbf{d}(t) \in S^2$.

The sensor provides measurement of the range $r(t) \in \mathbb{R}^{0+}$ from ${}^{F}_{F}\mathbf{p}_{A}(t)$ to the nearest object in the environment in the direction ${}^{F}\mathbf{d}(t) = {}^{F}_{F}\mathbf{R}_{A}(t)\,{}^{A}\mathbf{d}(t)$.

The function ${}^{A}\mathbf{d}(t)$ determines the scanning behaviour of the sensor and depends on the specific model used. The scanning model for the Hokuyo UBG-04LX-FO1 used in this research is provided in section 4.1.2.2.

2.2 State observers

2.2.1 Luenberger observers

An observer is a filter that is used to estimate the state of a dynamic system. The state x can be chosen as some set of variables governed by the system. In real-world conditions, the system is often infinite-dimensional.

The real system, known as the *plant*, is represented with a *model*. In most cases, some simplification of the plant must be performed to produce the model.

An observer provides an estimate $\hat{x}$ of the state $x \in \mathbb{R}^n$ of the model, given an output $y \in \mathbb{R}^m$ and a system input $u \in \mathbb{R}^p$. The dynamics of a nonlinear system are modelled with a nonlinear function f of dimension n.

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.44}$$

The state output y can also be conceptualised as a measurement of the state x, where the measurement function g is of dimension m.

$$y(t) = g(x(t), u(t)) \tag{2.45}$$

An observer employs an innovation function $L : \mathbb{R}^m \to \mathbb{R}^n$ to update a state estimate $\hat{x}$ using the difference between the measurement $y(t)$ and the predicted measurement $\hat{y}(t)$.

$$\dot{\hat{x}} = f(\hat{x}, u(t)) + L(y(t) - \hat{y}(t)) \tag{2.46}$$

$$\hat{y}(t) = g(\hat{x}(t), u(t)) \tag{2.47}$$

This form of observer was developed by Luenberger [27], and is often referred to as a *Luenberger observer*. Combining the observer equation with the measurement prediction function g gives the state observer as

$$\dot{\hat{x}} = f(\hat{x}, u(t)) + L(y(t) - g(\hat{x}(t), u(t))) \tag{2.48}$$

The goal is to choose design innovation function L that reduces the observer error $e(t) = x(t) - \hat{x}(t)$ to zero.

2.2.2 Symmetry-preserving observers

If the function f governing the dynamics of the state is invariant to the action of a Lie group $\mathbf{G}$, an *invariant* or *symmetry-preserving* observer has the form

$$\dot{\hat{x}} = h(\hat{x}, u(t), y(t) - \hat{y}(t)) \tag{2.49}$$

where the function h is also invariant to actions of $\mathbf{G}$. For nonlinear systems, the result is often improved global convergence properties in comparison to a traditional Luenberger observer.

Chapter 3

Problem statement

This project is part of a larger research direction at the ANU Research School of Engineering that will develop a theory of infinite-dimensional, symmetry preserving observers. As an initial exploration of this open problem, the central goals of this project are to:

- gain an understanding of how dense sensors can be used to estimate the state of infinite-dimensional systems;
- gain an insight into how symmetry-preserving observers can be used to better observe nonlinear infinite-dimensional systems;
- uncover pertinent directions for future research in this area - where a theory of infinite-dimensional, symmetry preserving observers would be useful;
- investigate if a sparse sensor can be used in a manner that approximates the capabilities of a dense sensor in the observation of an infinite-dimensional state.

The approach taken to achieve these goals will be to design and implement an observer for a simplified system that still captures some components of the overall research goals. The state variable to be estimated will be finite-dimensional, and thus the observer will be finite-dimensional. However, the sensor will still take measurements of an infinite dimensional state. In this way, the problem is similar to the dense optical flow estimation by Adarve et al. [26].

Initially, the observer innovation function will not be designed to be invariant. As a first step, this research will investigate correction schemes that converge locally. A future work package will be to adjust the update function to respect the symmetries of the system and achieve more global convergence properties.

Sparse range measurements have previously been used to reconstruct a depth field by Szeliski [28], who fitted a spline surface model to a cloud of points estimated from range data. The key difference in the approach used in this research is that rather than using the range measurements to directly estimate the state, the difference in measured and predicted range will be used to drive the innovation term of a state observer. In this approach the trajectory of the sensor will have a greater importance in ensuring that dense measurements can be made.

3.1 Estimating the pose and size of a cube from sparse range measurements

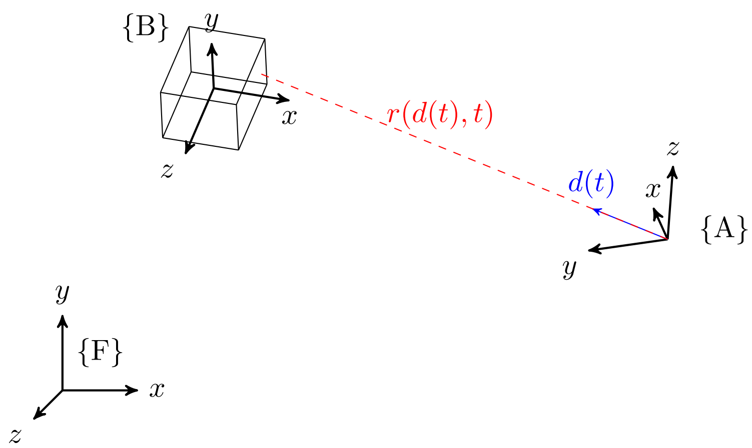


Figure 3.1: Range measurements of a cube from a depth sensor

A situation in which an infinite dimensional observer would be useful is in the estimation of the pose of an object of unknown size moving in an environment of unknown state.

For example, consider an autonomous robot deployed in an agricultural survey, which must determine the position and size of specimens of a certain crop. Using a geometric model for the general shape of the crop, an aerial vehicle that could routinely detect and characterise the position and size of specimens would be useful in monitoring growth and during harvesting.

The problem to be investigated is shown in Figure 3.1. A 2D scanning range sensor moves through an environment consisting of a target object of known shape, in this case a rigid cube, and an unknown background which may be an infinite dimensional dynamic system.

The state of the sensor is known, but the states of the cube and background environment are unknown. The goal is to use the state of the sensor and the range measurements it provides to estimate the state of the cube.

The frames used to describe the motion of the rigid bodies in this problem are:

- $\{F\}$ - the inertial (fixed) frame. For the purposes of this problem, the inertial frame is a frame whose motion is negligible. For the practical experiment this frame will be fixed to the ground.
- $\{A\}$ - the frame fixed to the sensor. The origin of this frame is the centre of rotation of the sensor's scan direction. The axes of $\{A\}$ are fixed to the sensor according to Figure 4.1 in Chapter 4. The transformation from $\{F\}$ to $\{A\}$ at time t is defined by the screw matrix of the sensor $\mathbf{S}_s(t)$.
- $\{B\}$ - the frame fixed to the cube. The origin of $\{B\}$ coincides with the centre of the cube and is aligned so that each axis intersects with the centre of a face of the cube. The transformation from $\{F\}$ to $\{B\}$ at time t is defined by the screw matrix of the cube $\mathbf{S}_c(t)$.

The sensor provides measurements of the range r to the nearest object from the sensor (either the cube or the background) in the direction $\mathbf{d}(t)$. These measurements can be considered sparse because the distance to just a single point is returned at each time step. The state of the sensor $\mathbf{X}_s(t)$ is defined as:

$$\mathbf{X}_s(t) = \{{}^F_F\mathbf{S}_A(t), {}^A_F\mathbf{T}_A(t), {}^A_F\mathbf{W}_A(t), {}^A\mathbf{d}(t)\} \tag{3.1}$$

The screw matrix represents the transformation from $\{A\}$ to $\{F\}$, defined in $\{F\}$. The twist and wrench matrices, as well as the scan direction $\mathbf{d}(t)$ are defined in terms of$\{A\}$. For simplicity, this will be denoted

$$\mathbf{X}_s(t) = \{\mathbf{S}_s(t), \mathbf{T}_s(t), \mathbf{W}_s(t), {}^A\mathbf{d}(t)\} \tag{3.2}$$

The direction of measurement ${}^A\mathbf{d}(t)$ varies as a rotation about the z-axis of $\{A\}$. This 2D scanning motion depends on the model of the sensor used and is described in more detail in section 4.1.2.2. For simplification, the motion of the sensor itself with respect to $\{F\}$ will be limited to rotation about the y-axis of $\{F\}$.

The state of cube $\mathbf{X}_c(t)$ is defined as

$$\mathbf{X}_c(t) = \{{}^F_F\mathbf{S}_B(t), {}^B_F\mathbf{T}_B(t), {}^B_F\mathbf{W}_B(t), s\} \tag{3.3}$$

For simplicity, this will be denoted

$$\mathbf{X}_c(t) = \{\mathbf{S}_c(t), \mathbf{T}_c(t), \mathbf{W}_c(t), s\} \tag{3.4}$$

The range measurements do not indicate whether the object detected is the cube or the background. Though the state of the cube and environment remain unknown, for simplification, it is assumed that either:

- the cube is within a distance r_{max} from the sensor and the background is at least a distance r_{max} away
- these target object and background do not touch or overlap and their surfaces are continuous functions on $\mathbb{R}^3$

These assumptions will be used to separate range measurements corresponding to the cube from those corresponding to the background. Only range measurements corresponding to the cube will be used in the observer innovation step. For simulated data, only the first assumption is necessary. For experimental data sets the environment is more complex so the second assumption is required to identify range measurements corresponding to the cube.

The aim is to design a nonlinear observer function f which estimates the state of the cube from the pose of the sensor, scan direction $\mathbf{s}$ and range measurement $\tilde{r}$ and measurement prediction $\hat{r}$.

$$\hat{\mathbf{X}}_c(k+1) = f(\mathbf{X}_s(t), \hat{\mathbf{X}}_c(k), \tilde{r}(t), \hat{r}(t)) \tag{3.5}$$

This observer formulation differs from that provided in equation 2.46 in that no state input is present in the system and more importantly, $\tilde{r}$ and $\hat{r}$ are provided as separate terms. Though a true Luenberger observer is driven by the output difference $\tilde{r} - \hat{r}$, such a scheme is not possible due to the way the problem has been simplified. Since range measurements corresponding to the background are discarded, the term $\tilde{r} - \hat{r}$ is undefined unless both ranges correspond to the cube. Such a limitation would make correcting differences in position and size particularly difficult.

A simulation toolbox will be implemented to simulate range measurements of rigid bodies using a scanning range sensor. The observer implementation will be implemented and its performance tested under a range of conditions.

Experimental validation will be performed by taking measurements of a known environment using the Hokuyo UBG-04LX-F01 scanning laser range-finder. These measurements will be used to quantify the performance of the observer under real-world conditions.

The observer implementation will be considered successful if it is able to converge to the true cube state around a local neighbourhood. Since an invariant observer is not being implemented, global convergence is not expected.

3.2 Deliverables

The project deliverables are to:

- implement a toolbox to simulate range measurements of rigid bodies;
- design an observer to estimate the cube state from sparse range measurements;
- produce and test a software implementation of the observer;
- validate the observer performance by collecting experimental data;
- present the research in a report and presentation.

Chapter 4

Observer simulation

4.1 Implementation

A simulation toolbox was implemented in Matlab to model scanning laser range-finder measurements of rigid bodies and test observer schemes. The main components of the simulation are:

- rigid body trajectory computation;
- solid object modelling;
- range measurement simulation;
- noise modelling;
- observer implementation.

The notation employed here and throughout the rest of the report uses the following conventions:

- Single values are denoted by plain text;
- Vectors are denoted in bold lowercase;
- Matrices are denoted in bold uppercase;
- An array formed by replicating a variable $\mathbf{a}$ in an $n \times m$ block array is denoted $\mathbf{a}_{n\times m}$.
- In many cases, a variable such as the sensor position $\mathbf{p}_s(t)$ represents a set of elements, each corresponding to the value at a certain time t. These will be referred to by the form of the value at a single time. For example, a matrix where each column represents a different position vector is used to represent $\mathbf{p}_s(t)$, but it will be described as a vector as this is the form of a single position.

Algorithm 1: Scanning range-finder and state observer simulation

Data:

n_{steps} - number of time steps in simulation

$\mathbf{X}_s$ - sensor pose and scan direction

$\mathbf{X}_e$ - cube and background pose, points and triangles

$\hat{\mathbf{X}}_c$ - estimate of the pose and size of the cube

c - (`true`/`false`) current range measurement is of cube or not

$\mathbf{r}$ - ground truth range

$\tilde{\mathbf{r}}$ - measured range

$\hat{\mathbf{r}}$ - range predicted from state estimate

$\boldsymbol{\alpha}$ - angle of incidence for each range measurement

$\mathbf{m}$ - index of triangle measured

$\boldsymbol{\theta}$ - scan angle in sensor frame

$\boldsymbol{\Theta}$ - set of scan angles that return range measurement

begin
    $settings \leftarrow$`loadsettings`
    $\mathbf{X}_s \leftarrow$ `initialisesensor`$(settings)$
    $\mathbf{X}_e \leftarrow$ `initialiseenvironment`$(settings)$
    `initialiseobserver`
    for $ii \leftarrow 1$ **to** n_{steps} **do**
        if $\boldsymbol{\theta}[ii] \in \boldsymbol{\Theta}$ **then**
            $[\mathbf{r}[ii], \boldsymbol{\alpha}[ii], \mathbf{m}[ii]] \leftarrow$ `computerange`$(\mathbf{X}_s[ii], \mathbf{X}_e[ii])$
        end
    end
    $\tilde{\mathbf{r}} =$ `addnoise`$(\mathbf{r}, \boldsymbol{\alpha}, \mathbf{m}, settings)$
    for $ii \leftarrow 1$ **to** n_{steps} **do**
        $\hat{\mathbf{X}}_c[ii+1] \leftarrow$ `estimatestate`$(\hat{\mathbf{X}}_c[ii])$
        if $\boldsymbol{\theta}[ii] \in \boldsymbol{\Theta}$ **then**
            $\hat{\mathbf{r}}[ii] \leftarrow$ `computerange`$(\mathbf{X}_s[ii], \hat{\mathbf{X}}_c[ii])$
            $c \leftarrow$ `identifyobject`$(c, \tilde{\mathbf{r}})$
            if c **then**
                $\hat{\mathbf{X}}_c[ii+1] \leftarrow$ `updatestate`$(\hat{\mathbf{X}}_c[ii+1], \mathbf{X}_s[ii], \hat{\mathbf{r}}, \tilde{\mathbf{r}})$
            end
        end
    end
end

A high level description of the simulation is provided in Algorithm 1. First, a settings file is loaded. The most important settings determined here are the trajectories of the sensor and environment objects, the scanning behaviour of the sensor and the observer update function. Next, the sensor class instance is initialised with `initialisesensor`. This requires computation of the pose and scanning directions of the sensor over time. Similarly, initialisation of the environment through `initialiseenvironment` requires computation of the pose of each rigid body comprising it. The surfaces of the bodies are then represented with a set of points and corresponding triangles. The position of each point with respect to the inertial frame $\{F\}$ is computed at each time step. The settings file provides the initial conditions with which the observer is initialised in `initialiseobserver`. Beginning on line 6, the state of the sensor and environment are used to compute the ground truth range measurements $\mathbf{r}$ at each time step. The incidence angle between the scan direction and object, as well as the index of the triangle hit are also stored as they will be required for sensor noise modelling. This is performed with a parallel `for` loop to speed up computation. Line 7 ensures ranges are only computed when the current scan direction is within the sensor's field of view. In line 11, noise is simulated and added to the ground truth ranges to produce the measured ranges $\tilde{\mathbf{r}}$. The `for` loop beginning on line 12 begins the observer simulation. At each time step, `estimatestate` estimates the state of the cube $\hat{\mathbf{X}}_c$ from the previous state with the kinematics model in Section 2.1.6. From the sensor state $\mathbf{X}_s$ and the estimated state of the cube $\hat{\mathbf{X}}_c$, `computerange` is used to determined the predicted range measurement $\hat{\mathbf{r}}$. The variable c indicates whether the current range measurement corresponds to the cube or the background. On line 16 the measured ranges and previous value of c are used to determine whether the current measurement corresponds to the cube. If it does, the cube state estimate $\hat{\mathbf{X}}_c$ is updated using the previous state estimate, current sensor state, and the predicted and measured ranges.

4.1.1 Rigid body motion

To simulate range measurements the pose of the sensor and the objects comprising the environment must be computed at each time step. The computations required to do so can be reduced by taking into account the kinds of motion that must be simulated. This section details how the pose of the sensor and objects is represented and computed, which is used in the functions `initialisesensor`, `initialiseenvironment` and `estimatestate` referenced in Algorithm 1.

The observer actually computes the *relative* position between the sensor and cube and simply uses knowledge of the sensor pose to determine the pose of the cube in the inertial frame. There is no need to simulate complex sensor motions because the motion of the cube can be adjusted to achieve the same result. The only requirement of the sensor motion is that measurements of a large range of the environment are acquired to ensure that the entire target object can be viewed. The scanning behaviour of the sensor is to rotate about the z-axis of the body fixed frame $\{A\}$. To provide a rectangular field of view, the motion of the sensor is therefore limited to constant velocity rotation about the y-axis of the inertial frame $\{F\}$.

The environment is modelled with two rigid bodies: a cube to be observed as the target object, and a stationary rectangular prism enclosing the sensor and cube which acts as the background. The various cube motions that will be simulated to test the observer's performance can be classed in terms of the wrench matrix of the cube as either

1. $\mathbf{W}_c = \mathbf{0}$
2. $\mathbf{W}_c \neq \mathbf{0}$

For case 1. the wrench and screw are constant so only the initial values are required. It is more efficient to represent the pose of a rigid body with just position and orientation in this case. The pose can be quickly computed by interpolating between an initial and final pose. For case 2. the screw, twist and wrench must be integrated numerically.

4.1.1.1 Interpolation

For the case of zero wrench, the pose of the body can be represented with a position vector $\mathbf{p}_i$ and an orientation quaternion $\mathbf{q}_i$. A trajectory of k poses at times $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & t_3 & \dots & t_k \end{bmatrix}$, is computed by interpolating from an initial pose $\{\mathbf{p}_1, \mathbf{q}_1\}$ to a final pose $\{\mathbf{p}_k, \mathbf{q}_k\}$.

The array of position vectors $\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \dots & \mathbf{p}_k \end{bmatrix}$ are computed with:

$$\mathbf{P} = (\mathbf{1}_{3\times k})\mathbf{p}_1 + (\mathbf{p}_k - \mathbf{p}_1)\frac{\mathbf{t} - t_1(\mathbf{1}_{1\times k})}{t_k - t_1} \tag{4.1}$$

Spherical linear interpolation is used to compute the array of orientation quaternions

$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \dots & \mathbf{q}_k \end{bmatrix}$ at each time step:

$$\mathbf{Q} = \frac{\mathbf{q}_1 \sin((\mathbf{1}_{[1\times k]} - \mathbf{t})\theta) + \mathbf{q}_k \sin(\mathbf{t}\theta)}{\sin(\theta)} \tag{4.2}$$

where

$$\theta = \cos^{-1}(\mathbf{q}_1 \cdot \mathbf{q}_k) \tag{4.3}$$

This interpolation method is used to compute the trajectory of the sensor. To acquire multiple views of the entire cube, the sensor must pan back and forth several times. This is achieved by first reversing the trajectory and concatenating with the original to produce the looped trajectories $\mathbf{P}_{loop}$ and $\mathbf{Q}_{loop}$:

$$\mathbf{P}_{loop} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \dots & \mathbf{p}_k & \mathbf{p}_k & \mathbf{p}_{k-1} & \mathbf{p}_{k-2} & \dots & \mathbf{p}_1 \end{bmatrix} \tag{4.4}$$

$$\mathbf{Q}_{loop} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \dots & \mathbf{q}_k & \mathbf{q}_k & \mathbf{q}_{k-1} & \mathbf{q}_{k-2} & \dots & \mathbf{q}_1 \end{bmatrix} \tag{4.5}$$

This looped trajectory is repeated k times to produce multiple back and forth scans:

$$\mathbf{P} = (\mathbf{P}_{loop})_{1\times k} \tag{4.6}$$

$$\mathbf{Q} = (\mathbf{Q}_{loop})_{1\times k} \tag{4.7}$$

4.1.1.2 Numerical integration

The time evolution of the screw, twist and wrench is computed iteratively from initial conditions by numerically integrating the ODEs in Section 2.1.6. For a rigid body with an associated reference frame $\{X\}$, moving with constant acceleration:

$$\mathbf{S}_X(t+\delta t) = \mathbf{S}_X(t)\exp(\delta t \mathbf{T}_X(t)) \tag{4.8}$$

$$\mathbf{T}_X(t+\delta t) = \mathbf{T}_X(t) + \delta t \mathbf{W}_X(t) \tag{4.9}$$

$$\mathbf{W}_X(t+\delta t) = \mathbf{W}_X(t) \tag{4.10}$$

Though a higher order integration method such as Runge-Kutta could be used to compute a trajectory that more accurately represents a constant acceleration, this is not strictly necessary. The observer performance is unlikely to be affected by how constant the acceleration is. Furthermore, it is likely that the experimentally collected data will have even larger variations in acceleration.

To simplify the code, the position vector and orientation quaternion are computed from the screw matrix. This allows the same functions to be used in either the interpolation or numerical integration cases. Rotating the points that make up the rigid objects can also be done more compactly with quaternions.

4.1.2 Sensor modelling: `initialisesensor`

The motion model below is used to compute the pose of the sensor. The scanning model is used to compute the set of scanning directions. These actions are performed in the `initialisesensor` function in Algorithm 1

4.1.2.1 Motion

The state of the sensor $\mathbf{X}_s(t)$ consists of terms corresponding to its motion and scanning operation. Since the motion of the sensor is restricted to zero acceleration, the state of sensor can be computed with the interpolation method and represented with position, orientation and scanning direction.

$$\mathbf{X}_s(t) = \{\mathbf{p}_s(t), \mathbf{q}_s(t), {}^{A}\mathbf{d}(t)\} \tag{4.11}$$

Since it has a stationary position, the position of the sensor over time is fixed at the origin of the inertial frame $\{F\}$.

$$\mathbf{p}_1 = \mathbf{p}_2 = \mathbf{p}_3 = \cdots = \mathbf{p}_k = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.12}$$

The sensor rotates between $-\phi$ and ϕ about the y-axis of inertial frame $\{F\}$. Thus, its orientation is computed by interpolating between $\mathbf{q}_1$ and $\mathbf{q}_k$ with equation 4.2.

$$\mathbf{q}_1 = \begin{bmatrix} \cos(-\phi/2) \\ \sin(-\phi/2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \cos(\phi/2) \\ 0 \\ -\sin(\phi/2) \\ 0 \end{bmatrix} \tag{4.13}$$

$$\mathbf{q}_k = \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \cos(\phi/2) \\ 0 \\ \sin(\phi/2) \\ 0 \end{bmatrix} \tag{4.14}$$

4.1.2.2 Scanning

The scanning behaviour of the sensor depends on the particular model used. The *Hokuyo UBG-04LX-F01* scanning laser range-finder will be modelled as it was the sensor used to conduct experiments in this project. This sensor produces a 785nm laser beam, projected at a precise direction. It measures the characteristics of the reflected beam to determine the position to the nearest object in the direction of the beam. The beam direction is varied by reflecting it off a rotating mirror. The rotation means that the beam direction effectively rotates with a constant velocity in a single plane. A portion of the field of view of the laser beam is obscured, so measurements will not be returned in a certain region of each revolution.

The vector ${}^{A}\mathbf{d}(t)$ will be used to model this scanning behaviour. To accurately model this, the following parameters are used:

- field of view $\mathbf{\Theta}$: The vector ${}^{A}\mathbf{d}(t)$ rotates anti-clockwise about the z axis of the sensor frame $\{A\}$. Measurements are only taken when the scan angle about is between $-\theta$ and θ about the $-z$-axis of the sensor frame $\{A\}$. In practice, the field of view is implemented as the start angle $-\theta$, direction of rotation and angular range 2θ.
- number of scans n_{scans}: This represents the number of scan angles in a single revolution. Since measurements are limited by the field of view of the sensor, the actual number of measurements per second is $n_{ranges} = \frac{2\theta}{2\pi} n_{scans}$. The angular resolution is $d\theta = \dfrac{2\pi}{n_{scans}}$.
- revolutions per second Ω: This is measured in Hz and gives the length of each time

step $d\tau = \dfrac{1}{n_{scans}\Omega}$

- n_{loops}: The number of back and forth repeats of the sensor trajectory.

From these parameters the scanning direction ${}^A\mathbf{d}(t)$ is created. At each time t, ${}^A\mathbf{d}(t)$ is either a unit vector indicating the direction of measurement in the sensor frame, or has $\mathbf{0}$ magnitude, corresponding to when ${}^A\mathbf{d}(t)$ is outside the field of view and the sensor is not returning a measurement.

$$
{}^A\mathbf{d}(t) = \begin{cases} \begin{bmatrix} \cos(-\theta + 2\pi t') \\ -\sin(-\theta + 2\pi t') \\ 0 \end{bmatrix} & \text{if } t' \leq \theta/\pi,\ t' = k\delta\tau\ \forall k \in \mathbb{N} \\ \mathbf{0}_{3\times 1} & \text{if } t' > \theta/\pi,\ t' \neq k\delta\tau\ \forall k \in \mathbb{N} \end{cases} \tag{4.15}
$$

where

$$
t' = \mod(t, 1/d\theta)\, d\theta \tag{4.16}
$$

Figure 4.1 shows the frame $\{A\}$ fixed to the sensor and the scan direction ${}^A\mathbf{d}(t)$. At time $t' = 0$, the first scan direction ${}^A\mathbf{d}_0$ has an angular displacement of $-\theta$ about the z-axis from the forward facing x-direction. After each time step $d\tau$, the scan direction rotates by $d\theta$ about the z-axis. There are n_{ranges} scan directions within the field of view of the sensor. The entire revolution is divided into n_{scans} scan directions.

To simulate range measurements, the scan direction is required in the inertial frame $\{F\}$. This is computed by multiplying with the screw matrix of the sensor.

$$
\begin{aligned} {}^F\mathbf{d}'(t) &= \mathbf{S}_s(t)\, {}^A\mathbf{d}'(t) \\ &= {}^F_F\mathbf{S}_A(t)\, {}^A\mathbf{d}'(t) \end{aligned} \tag{4.17}
$$

4.1.2.3 Hokuyo UBG-04LX-F01 model

To achieve a panning motion, the sensor trajectory is defined as a rotation between $\pi/8$ and $\pi/8$ radians about the y-axis of the sensor frame $\{A\}$. The sensor rotates with a constant angular speed of $\pi/4$ radians/s.

The sensor has a field of view of π radians, providing range measurements when the scan direction is between $\pi/2$ and $-\pi/2$ radian about the z-axis of the sensor frame $\{A\}$. The

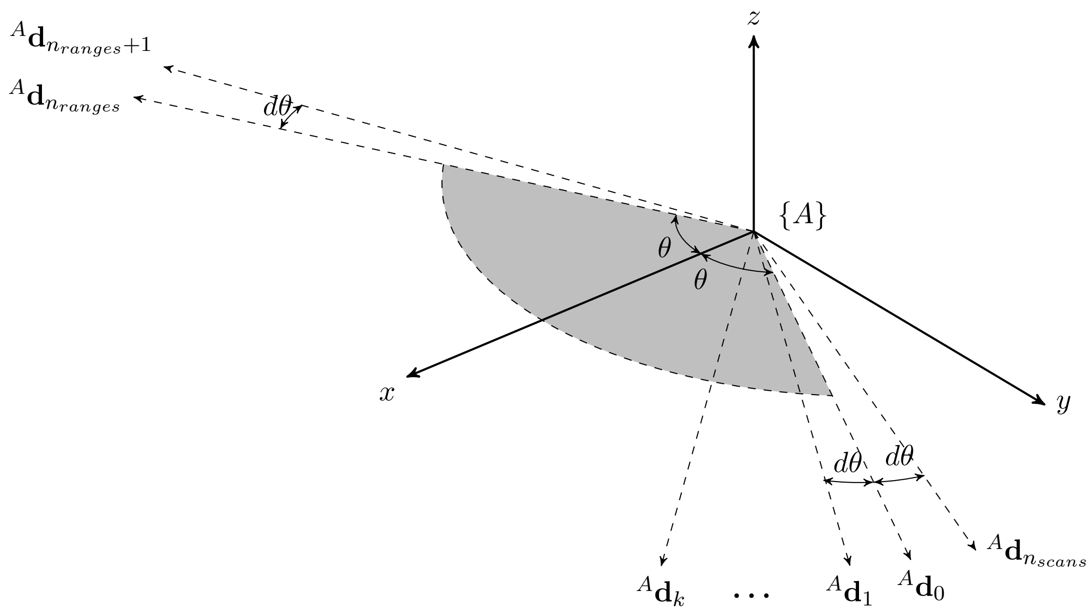


Figure 4.1: Scanning behaviour of the UBG-04LX-F01 and parameters used to model it

laser beam rotates in an anti-clockwise direction about the z-axis of $\{A\}$. The beam rotates with an angular speed of 24Hz, providing 24 scans per second. Measurements are returned at half of the time steps ineach revolution, so each scan contains 512 measurements at increments of $pi/512$ radians.

4.1.3 Environment modelling: `initialiseenvironment`

The environment described in the problem formulation in Chapter 3 was an infinite-dimensional system. Since the state variable to be estimated consists of just the cube pose and size, range measurements corresponding to the background are filtered out. Thus, the state of the background has no effect on the performance of the observer. To reduce computational load, the background will be modelled as a single object; a rigid rectangular prism enclosing both the sensor and cube. The entire state - cube and background - is now modelled as a finite dimensional state. However, the sensor is still taking measurements of a dense depth field on S^2. From its perspective, there would be no difference between the rectangular prism background and a more complex surface model.

4.1.3.1 Motion

The pose of each object represents the pose of its centre of mass. As described in Section 4.1.1 the pose is computed with interpolation for the case of zero wrench, and numerical integration for the case of non-zero wrench. The pose of the object making up the environment is computed with the function `initialiseenvironment` in Algorithm 1. This function also creates the points and triangles that model the surface of the object which is described below.

4.1.3.2 Rigid objects

The environment is represented with two rectangular prisms; the cube and a larger rectangular prism enclosing both the sensor and cube, to represent the background. These objects are modelled as an ordered set of 8 points in the inertial reference frame and an ordered set of 12 triangles formed by these points. Each triangle is represented by a set of 3 integers, indicating the index of the three points that make up its vertices.

The cube points in body frame $\{B\}$ are represented with the matrix ${}^{B}\mathbf{P}$.

$$ {}^{B}\mathbf{P} = \frac{1}{2}s \begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} \tag{4.18} $$

To represent these points in the inertial frame $\{F\}$, ${}^{F}\mathbf{P}$ is computed by rotating each point with the orientation quaternion of frame $\{B\}$ using Equation 2.24 before adding the vector representing the translation of $\{B\}$ from $\{F\}$.

The triangles are represented with the matrix $\mathbf{T}$. Each triangle is represented by a row. The elements of these rows are the indexes of the points in ${}^{F}\mathbf{P}$ that form the three vertices

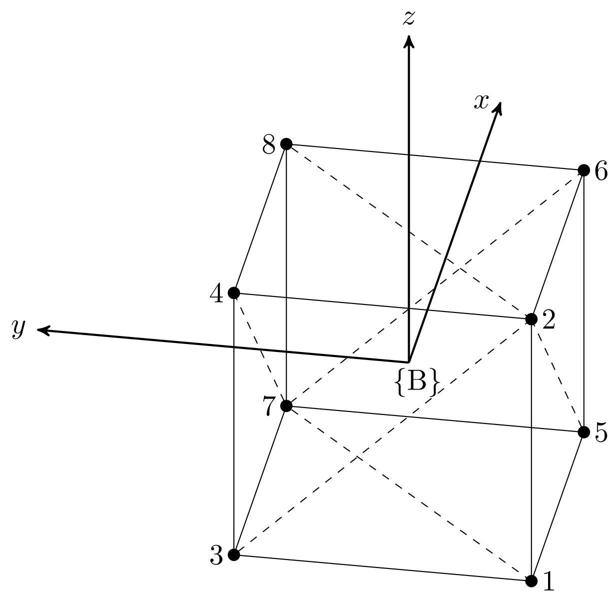


Figure 4.2: Cube modelled with an ordered set of points and corresponding triangles

of the triangle.

$$\mathbf{T} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 3 \\ 4 & 3 & 7 \\ 4 & 8 & 7 \\ 5 & 6 & 7 \\ 8 & 6 & 7 \\ 2 & 6 & 5 \\ 2 & 1 & 5 \\ 2 & 6 & 8 \\ 2 & 4 & 8 \\ 1 & 5 & 7 \\ 1 & 3 & 7 \end{bmatrix} \tag{4.19}$$

The points and triangles are shown in Figure 4.2.

4.1.4 Measurement modelling

4.1.4.1 Range computation: `computerange`

This section describes the implementation used in the `computerange` function in Algorithm 1.

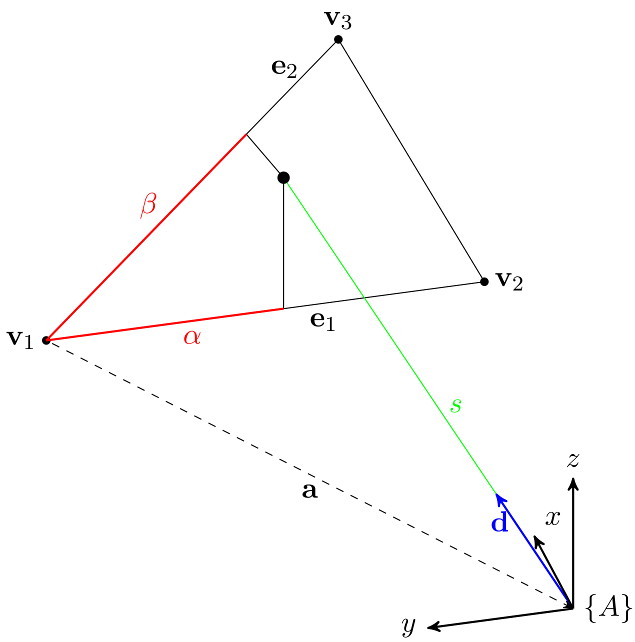


Figure 4.3: Ray-triangle intersection

Given its screw matrix and scan direction (in the body fixed frame), the position of the sensor and its scan direction in the inertial frame are determined. The distance to the nearest environment object from the sensor along the scan direction is determined with the Möller-Trumbore ray-triangle intersection algorithm, shown in Algorithm 2.

Figure 4.3 shows a simplified scenario involving the intersection of a ray with a single triangle. In practice, the algorithm is vectorised to compute the intersections with a *set* of triangles. In this vectorised implementation, many variables represent a matrix whose columns are each vectors. These variables will still be described as vectors to emphasise the operation of the Möller-Trumbore algorithm, rather than the specific implementation details.

The output variables are first initialised to the case that there is no intersection with the scanning direction. x is set to `false` and the range, angle and triangle index outputs are set to return `NaN`.

On line 8, the set of points $\mathbf{P}$ are indexed using the columns of the triangle matrix $\mathbf{T}$ to extract the three vertices corresponding to each triangle. The vectors representing the edges sharing vertex $\mathbf{V}_1$ are computed. from triangles and points, extract vertices of each triangle. The vector $\mathbf{A}$ computed on line 14 represents the translation from the ray origin $\mathbf{o}$ to $\mathbf{V}_1$.

The collection of vectors $\mathbf{B}$ is computed on line 16 by taking the cross product of the scan direction $\mathbf{d}$ and each edge $\mathbf{E}_2$. The determinant $\boldsymbol{\delta}$ of the matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{d} & \mathbf{e}_2 \end{bmatrix} \tag{4.20}$$

is computed on line 16. This is first used to determine if the scan direction $\mathbf{d}$ lies in the plane of the triangle by checking if the determinant is close to zero. If so, no intersection can occur. The zero determinant values are then set to `NaN` to avoid a division by zero later.

Beginning on line 21, determinant $\boldsymbol{\delta}$ is used to compute the barycentric coordinates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and the distance $\mathbf{s}$ from the origin to the triangle plane along the scan direction $\mathbf{d}$.

The barycentric coordinates are used to determine if the intersection between the scan direction $\mathbf{d}$ and the plane of the triangle lies within the triangle itself.

The vector $\mathbf{x}$ on line 27 now indicates which triangles intersected with the scan direction. $\mathbf{x}$ is used to mask the ranges to the triangles $\mathbf{s}$, to give $\mathbf{r}$; the range to each intersecting triangle.

The minimum range r and triangle index m are determined before computing the angle of incidence θ between $\mathbf{d}$ and the closest triangle.

4.1.4.2 Sensor noise: `addnoise`

In Algorithm 1, the function `addnoise` takes the ground truth range measurements $r(t)$ and produces the noisy range measurements $\hat{r}(t)$ that the observer will actually receive. The noise function f_s is

$$\hat{r}(t) = f_s(r(t), \theta(t), \phi(k)) \tag{4.21}$$

where $\theta(t)$ is the incidence angle of the measurement at time t, ϕ represents the surface properties of the object k that was measured.

For the Hokuyo UBG-04LX-F01 sensor used, range measurements taken at various distances and incidence angles were used to estimate the noise model f_{UBG} which is provided in Section 5.1.

Algorithm 2: Möller-Trumbore ray-triangle intersection

input : $\mathbf{o}$ - ray origin
$\mathbf{d}$ - ray direction vector
$\mathbf{P}$ - cube in inertial frame
$\mathbf{T}$ - triangle matrix

output : x - True/False - measurement corresponds to object
r - distance to object in m
θ - incidence angle in rad
m - index of triangle hit

begin
  `/* initialise outputs */`
  $x \longleftarrow 0$
  $r \longleftarrow NaN$
  $\theta \longleftarrow NaN$
  $m \longleftarrow NaN$
  `/* triangle vertexes and edges */`
  $\mathbf{V}_1 \longleftarrow \mathbf{P}[\mathbf{T}[:,1]]$
  $\mathbf{V}_2 \longleftarrow \mathbf{P}[\mathbf{T}[:,2]]$
  $\mathbf{V}_3 \longleftarrow \mathbf{P}[\mathbf{T}[:,3]]$
  $\mathbf{E}_1 \longleftarrow \mathbf{V}_2 - \mathbf{V}_1$
  $\mathbf{E}_2 \longleftarrow \mathbf{V}_3 - \mathbf{V}_1$
  $m = \texttt{size}(\mathbf{V}_1, 1)$
  $\mathbf{A} \longleftarrow \mathbf{o}_{[m\times 1]} - \mathbf{V}_1$
  `/* determinant */`
  $\mathbf{B} \longleftarrow \mathbf{d}_{[m\times 1]} \times \mathbf{E}_2$
  $\boldsymbol{\delta} \longleftarrow \mathbf{E}_1 \cdot \mathbf{B}$
  $\mathbf{y} \longleftarrow |\boldsymbol{\delta}| \leq \mathbf{0}$
  $\boldsymbol{\delta}[\mathbf{y}] \longleftarrow \mathbf{NaN}$
  `/* barycentric coordinates */`
  $\boldsymbol{\alpha} \longleftarrow (\mathbf{A} \cdot \mathbf{B})/\boldsymbol{\delta}$
  $\mathbf{Q} \longleftarrow \mathbf{A} \times \mathbf{E}_1$ *(along dim 2)
  $\boldsymbol{\beta} \longleftarrow (\mathbf{d}_{[n\times 1]} \cdot \mathbf{Q})/\boldsymbol{\delta}$ *(along dim 2)
  $\mathbf{s} \longleftarrow (\mathbf{E}_2 \cdot \mathbf{Q})/\boldsymbol{\delta}$
  `/* intersection vector */`
  $\mathbf{z} \longleftarrow \mathbf{y}$ **and** $(\boldsymbol{\alpha} \geq \mathbf{0})$ **and** $(\boldsymbol{\beta} \geq \mathbf{0})$ **and** $(\boldsymbol{\alpha} + \boldsymbol{\beta} \leq \mathbf{1})$
  $\mathbf{x} \longleftarrow \mathbf{z}$ **and** $(\mathbf{s} \geq \mathbf{0})$
  if $\texttt{any}(\mathbf{x})$ **then**
    $x \longleftarrow 1$
    $\mathbf{x}[\textbf{not } \mathbf{x}] \longleftarrow \mathbf{NaN}$
    $\mathbf{r} = \mathbf{s} \circ \mathbf{x}$
    $r = \texttt{min}(\mathbf{r})$
    $m = \texttt{find}(\mathbf{r} = r, 1)$
    $\mathbf{e}_1 \longleftarrow \mathbf{E}_1[t,:]$
    $\mathbf{e}_2 \longleftarrow \mathbf{E}_2[t,:]$
    $\mathbf{n} = \mathbf{e}_1 \times \mathbf{e}_2$
    $\theta = \texttt{atan2}(|\mathbf{d} \times \mathbf{n}|, \mathbf{d} \cdot \mathbf{n})$
    $\theta = \texttt{min}(\theta, \pi - \theta)$
  end
end

4.1.5 Observer implementation

4.1.5.1 Estimate: `estimatestate`

The state of the cube at each time step is estimated using the numerical integration method described in Section 4.1.1.2. This state estimation is implemented in the function `estimatestate` in Algorithm 1.

4.1.5.2 Object/background separation: `identifyobject`

The observer update function uses range measurements to estimate the state of the cube $\mathbf{X}_c$. In order to perform accurately, the observer must only use range measurements that correspond to the cube. The function `identifyobject` in Algorithm 1 uses the range measurements and knowledge of the configuration of the environment to separate measurements of the cube and background.

The binary variable c indicates whether the current range measurement corresponds to cube ($c =$ `true`) or the background ($c =$ `false`). It is assumed that initially the sensor will be observing the background, so $c_0 =$ `false`.

The scheme used to identify range measurements corresponding to the cube is shown in Algorithm 3. There are two assumptions that may be used.

1. The *difference assumption* relies on the assumption that the cube and background objects are continuous. Differences in consecutive range measurements larger than Δ_{max} indicate a discontinuity, implying that a new object is being measured. When this occurs, the value of c changes.

2. The *range assumption* is used when the maximum distance to the cube and minimum distance to the background are restricted. Range measurements within r_{max} correspond to the cube while larger ranges correspond to the background.

4.1.5.3 Update: `updatestate`

If the `identifyobject` function identifies a range measurement as corresponding to the cube, the `updatestate` function is used to update the state estimate of the cube $\hat{\mathbf{X}}_c$. $\hat{\mathbf{X}}_c$ is updated using the previous cube state estimate, current sensor state, and sets of measured

Algorithm 3: Target/background object separation

```
input  : differenceAssumption - true/false
         rangeAssumption - true/false
         Δ_max - max diff between measurements of same object
         r_max - max range for cube
         c - true/false - current measurement is of cube
         r_{i+1} - distance to object at t = i + 1
         r_i - distance to object at t = i
output : c - true/false
begin
    if differenceAssumption then
        if |r_{i+1} − r_i| > Δ_max then
            c = mod (c + 1, 2)
        end
    end
    if rangeAssumption then
        if r_{i+1} > r_max then
            c = 0
        end
    end
end
```

and predicted range measurements chosen according to a set of indexes $\mathbf{u}(t)$.

$$\hat{\mathbf{X}}_c(k+1) = f(\mathbf{X}_s(t), \hat{\mathbf{X}}_c(k), \mathbf{r}(\mathbf{u}(t)), \hat{\mathbf{r}}(\mathbf{u}(t))) \tag{4.22}$$

The pose of $\hat{\mathbf{X}}_c$ is corrected by adjusting $\hat{\mathbf{S}}_c$, $\hat{\mathbf{T}}_c$ or $\hat{\mathbf{W}}_c$. The orientation is adjusted by rotating about an axis $\mathbf{r}_{update}$. The position is adjusted by translating in the direction of $\mathbf{p}_{update}$ $\mathbf{r}_{update}$ and $\mathbf{p}_{update}$ are scaled differently, depending on whether they are applied to $\hat{\mathbf{S}}_c$, $\hat{\mathbf{T}}_c$ or $\hat{\mathbf{W}}_c$. The size update s_{update} is independent of the pose update scheme used.

Input ranges:

A set of four range measurements forming a quadrilateral are used in the state update. The four ranges are chosen with an ordered sequence of indexes $\mathbf{u}(t)$. At a time step ii, the set of time steps used is in the update function is

$$\mathbf{u}(ii) = \{ii, (ii-1), (ii-n_{scans}), (ii-1-n_{scans})\} \tag{4.23}$$

It is possible that the measured or predicted ranges do not exist for some time steps in $\mathbf{u}(ii)$, as the range may have corresponded to the background rather than the cube. Thus, the measurement indexes $\tilde{\mathbf{u}}(ii)$ and estimation indexes $\hat{\mathbf{u}}(ii)$ will be subsets of, but not necessarily congruent to $\mathbf{u}(ii)$.

Orientation update:

The method used to correct the orientation of the cube state estimate $\hat{\mathbf{X}}_c$ is shown in Figure 4.4.

In order to estimate the orientation of the cube, at least 3 ranges are required from both the prediction and measurement: $|\hat{\mathbf{u}}| \geq 3$ and $|\tilde{\mathbf{u}}| \geq 3$. If all four indexes are present, the range from the last time step $(ii - 1 - n_{scans})$ is ignored.

For both the measurement and prediction, the points of intersection ${}^F\mathbf{P}(\mathbf{u})$ between the set of scanning directions ${}^F\mathbf{D}(\mathbf{u})$ and the cube are computed using the set of range measurements $\mathbf{r}(\mathbf{u})$.

$$ {}^F\mathbf{P}(\mathbf{u}) = {}^F\mathbf{D}(\mathbf{u})\mathbf{r}(\mathbf{u}) \tag{4.24} $$

The normal to the plane formed by the three points is then computed.

$$ \mathbf{n} = [{}^F\mathbf{P}(u_2) - {}^F\mathbf{P}(u_1)] \times [{}^F\mathbf{P}(u_3) - {}^F\mathbf{P}(u_1)] \tag{4.25} $$

Because a cube has 24 regular isometries, it is not necessary to exactly align the reference frames of the estimated and true cubes. Any face of the estimated cube can be aligned with any face of the true cube, and the maximum rotation correction required will be $\pi/4$ radians.

The angle between the two normals ψ is computed as

$$ \psi = \mathtt{atan2}(|\hat{\mathbf{n}} \times \tilde{\mathbf{n}}|, \hat{\mathbf{n}} \cdot \tilde{\mathbf{n}}) \tag{4.26} $$

The axis $\mathbf{r}_{update}$ that the estimated cube orientation $\hat{\mathbf{R}}_c$ will be rotated by is computed by taking the cross product of the predicted and measured normals. The direction of rotation is changed if the angle ψ between the two normals is greater than $\pi/4$ radians.

$$ \mathbf{r}_{update} = \mathtt{sign}\left(\frac{\pi}{4} - \psi\right)(\hat{\mathbf{n}} \times \tilde{\mathbf{n}}) \tag{4.27} $$

where

$$ \mathtt{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{4.28} $$

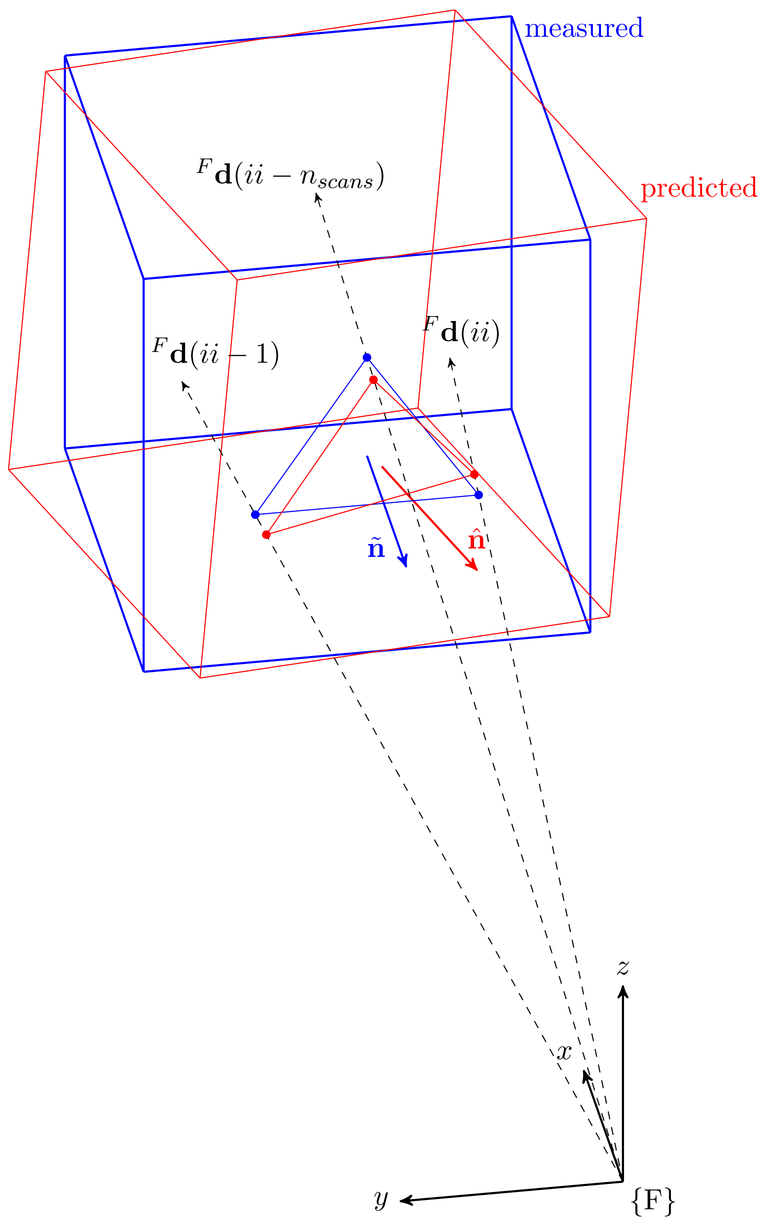


Figure 4.4: Orientation update scheme: intersection of the scan directions with the surfaces of the measured and predicted cubes are used to determine the surface normals.

To update the screw matrix $\hat{\mathbf{S}}_c$, $\mathbf{r}_{update}$ is converted to a rotation matrix $\mathbf{R}_{update}$ with Rodrigues' rotation formula (equation 2.19). The correction is then applied as:

$$\hat{\mathbf{R}}_c(k+1) = R_{scale}\mathbf{R}_{update}\hat{\mathbf{R}}_c(k) \tag{4.29}$$

To update via the screw or wrench, $\mathbf{r}^{\wedge}_{update}$ is scaled and then added to the angular velocity or angular acceleration respectively.

$$\hat{\boldsymbol{\omega}}^{\wedge}_c(k+1) = \hat{\boldsymbol{\omega}}^{\wedge}_c(k) + \omega_{scale}\mathbf{r}^{\wedge}_{update} \tag{4.30}$$

$$\hat{\boldsymbol{\alpha}}^{\wedge}_c(k+1) = \hat{\boldsymbol{\alpha}}^{\wedge}_c(k) + \alpha_{scale}\mathbf{r}^{\wedge}_{update} \tag{4.31}$$

The chosen scale factor depends on whether $\mathbf{r}_{update}$ is applied via $\hat{\mathbf{S}}_c$, $\hat{\mathbf{T}}_c$ or $\hat{\mathbf{W}}_c$.

Position update:

The method used to correct the position of $\hat{\mathbf{X}}_c$ is shown in Figure 4.5.

In order to estimate the position of the cube, at least 1 range measurement is required from both the prediction and measurement: $|\hat{\mathbf{u}}| \geq 1$ $|\tilde{\mathbf{u}}| \geq 1$. Additionally, scan directions at time steps $ii, (ii-1)$ and $(ii-1-n_{scans})$ are required. The scan direction ${}^F\mathbf{d}(t)$ must be within the sensor's field of view at these time steps.

The points of intersection are calculated using Equation 4.24. In Figure 4.5 all four scan directions intersect with the predicted cube state, but only one intersects with the measured cube. The average of these points is computed to give the mean estimated position $\hat{\boldsymbol{\mu}}_{\mathbf{p}}$ and the mean measured position $\tilde{\boldsymbol{\mu}}_{\mathbf{p}}$.

The x, y and z components of the update vector may very significantly in size due to the scanning behaviour of the sensor. It is necessary to scale the position update vector according to these components. The mean of all predicted and measured ranges μ_r is computed. Four points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ are computed to be used in scaling:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{p}_s(t) = {}^F_F\mathbf{p}_A(t) \\
\mathbf{p}_1 &= \mu_r {}^F\mathbf{d}(ii) \\
\mathbf{p}_2 &= \mu_r {}^F\mathbf{d}(ii-1) \\
\mathbf{p}_3 &= \mu_r {}^F\mathbf{d}(ii-1-n_{scans})
\end{aligned} \tag{4.32}$$

The update vector is computed by scaling the mean intersection points with these four

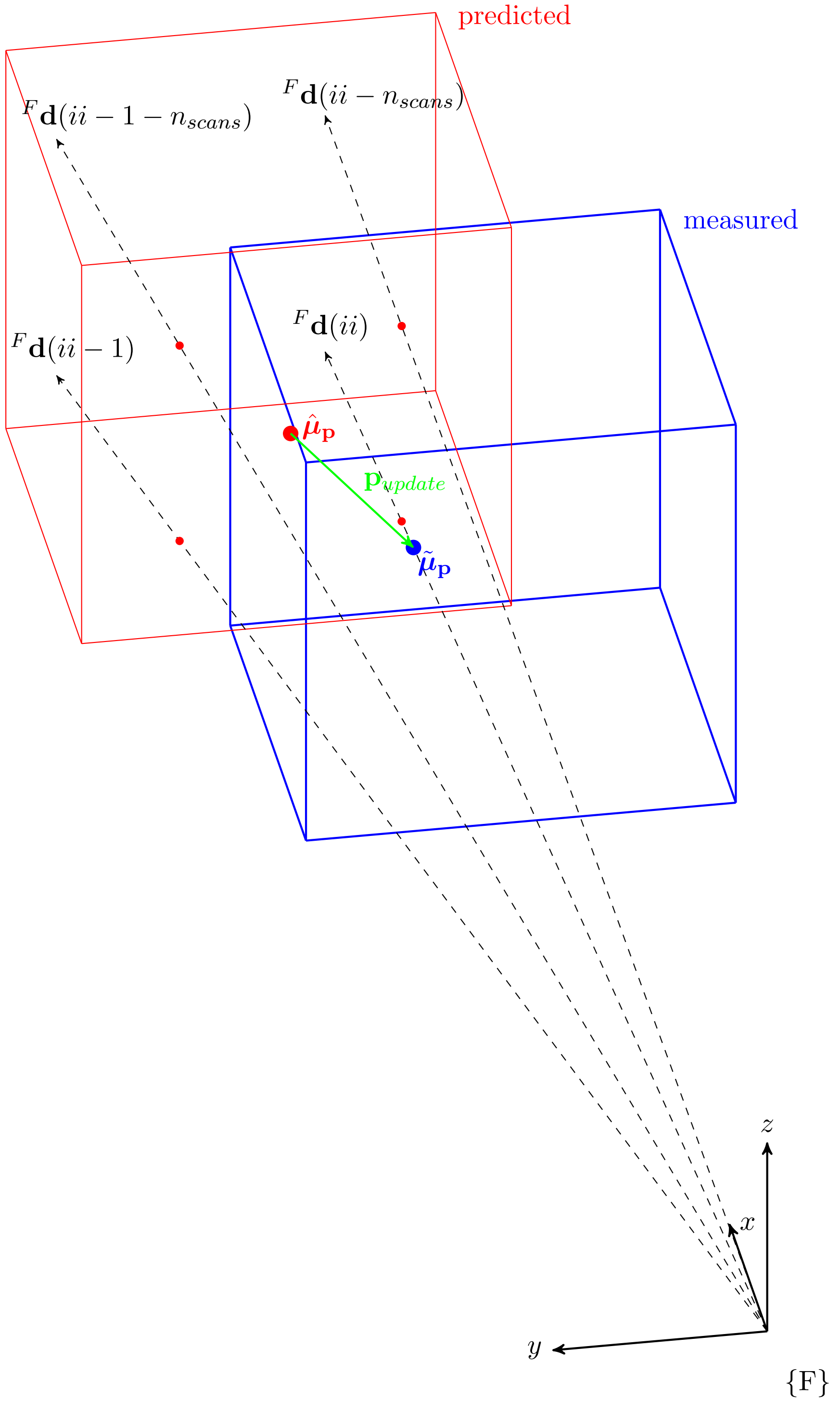


Figure 4.5: Position update: centre of mass of predicted and measured intersection points used to determine update direction vector.

points:

$$\mathbf{p}_{update} = \begin{bmatrix} \frac{1}{|\mathbf{p}_1-\mathbf{p}_0|} & 0 & 0 \\ 0 & \frac{1}{|\mathbf{p}_2-\mathbf{p}_1|} & 0 \\ 0 & 0 & \frac{1}{|\mathbf{p}_3-\mathbf{p}_2|} \end{bmatrix} (\tilde{\boldsymbol{\mu}}_{\mathbf{p}} - \hat{\boldsymbol{\mu}}_{\mathbf{p}}) \tag{4.33}$$

The screw, twist and wrench are corrected using $\mathbf{p}_{update}$. The scaling factor used depends on whether the update is performed via the screw, twist or wrench.

$$\hat{\mathbf{p}}_c(k+1) = \hat{\mathbf{p}}_c(k) + p_{scale}\mathbf{p}_{update} \tag{4.34}$$

$$\hat{\mathbf{v}}_c(k+1) = \hat{\mathbf{v}}_c(k) + v_{scale}\mathbf{p}_{update} \tag{4.35}$$

$$\hat{\mathbf{a}}_c(k+1) = \hat{\mathbf{a}}_c(k) + a_{scale}\mathbf{p}_{update} \tag{4.36}$$

Size update:

In order to correct the size of the cube, at least 1 range measurement is required from both the prediction and measurement: $|\hat{\mathbf{u}}| \geq 1$ $|\tilde{\mathbf{u}}| \geq 1$. The size update scheme also differs based on the sets predicted and measured ranges.

For the case where a different pattern of ranges is observed ($\hat{\mathbf{u}} \not\equiv \tilde{\mathbf{u}}$), the update method is shown in Figure 4.6.

The dot product from the vector $\mathbf{p}_{update}$ computed for the position update and the current scan direction is taken:

$$s_{update} = \mathbf{p}_{update} \cdot {}^F\mathbf{d}(ii) \tag{4.37}$$

For the case where the same pattern of ranges is observer ($\hat{a} \equiv \tilde{a}$), the update method is shown in Figure 4.7.

The size update is taken as the difference in the means of the measured and predicted ranges:

$$s_{update} = \tilde{\mu}_r - \hat{\mu}_r \tag{4.38}$$

In both cases, the cube size estimate is updated by scaling s_{update} and adding this to the previous estimate:

$$s(k+1) = s(k) + s_{scale}s_{update} \tag{4.39}$$

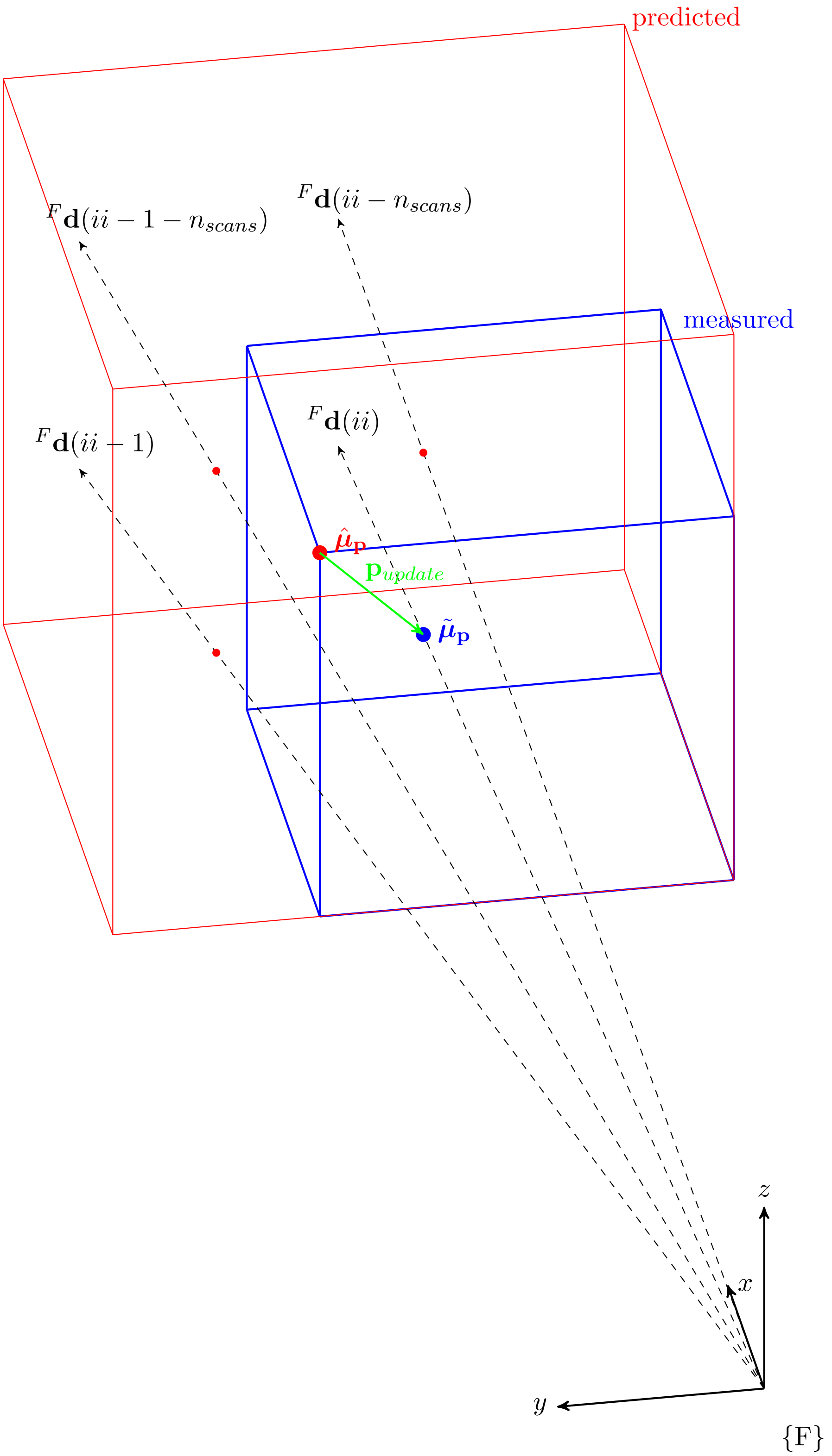


Figure 4.6: Size update - case 1: centre of mass of intersection points used to determine size update.

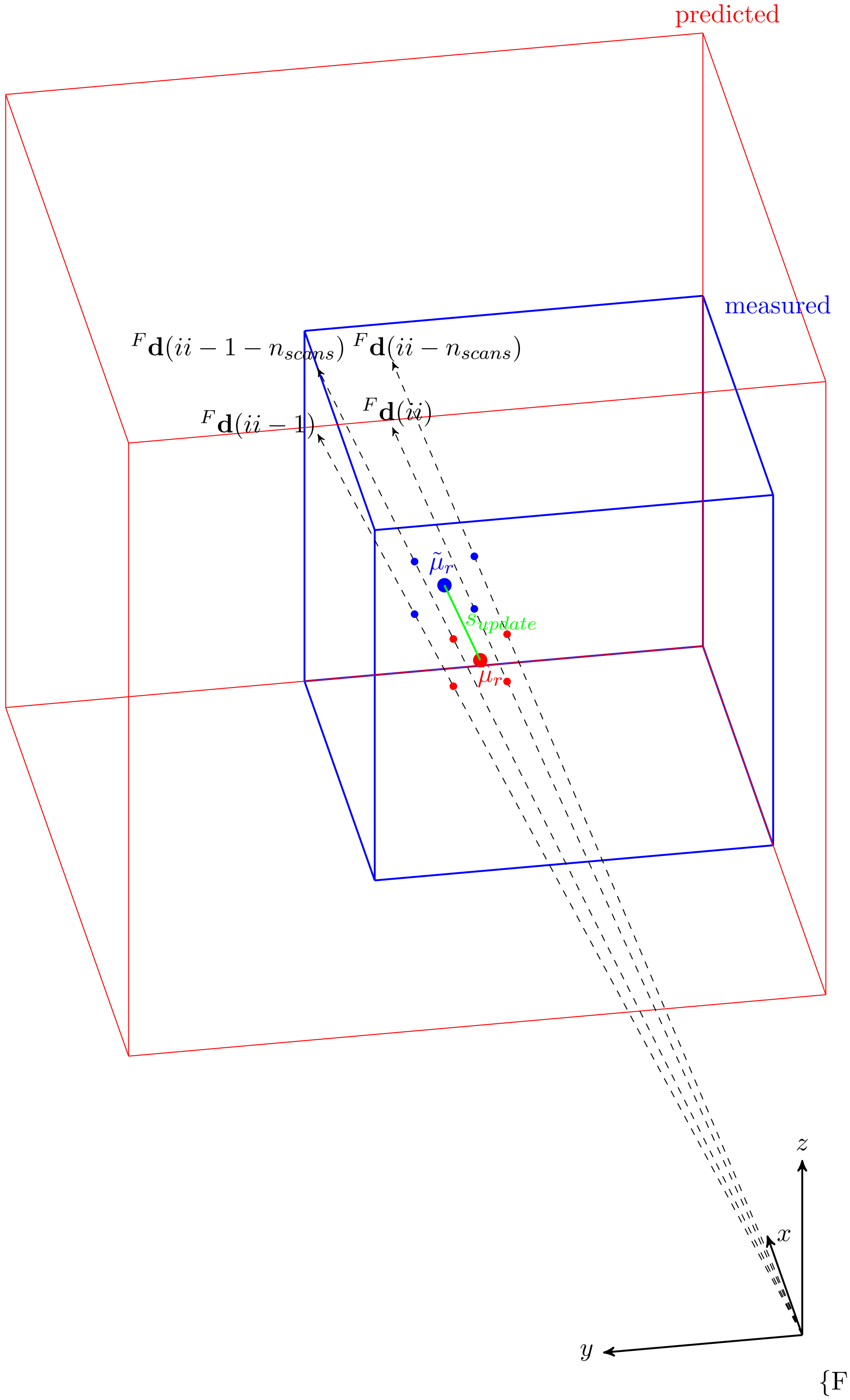


Figure 4.7: Size update - case 2: difference in mean ranges used to determine size update.

4.2 Results

The ability of the observer to estimate the state of a cube undergoing stationary, rotating, and translating motions was tested. For these classes of motion, the trajectory was further defined to assess the performance of the observer when one, two or three faces of the cube were visible to the sensor. Initial conditions and the scaling factors of the orientation, position and size update functions were defined to assess the individual and combined performance of these update functions. Key results of this analysis and an overall assessment of the observer's performance are presented in this section.

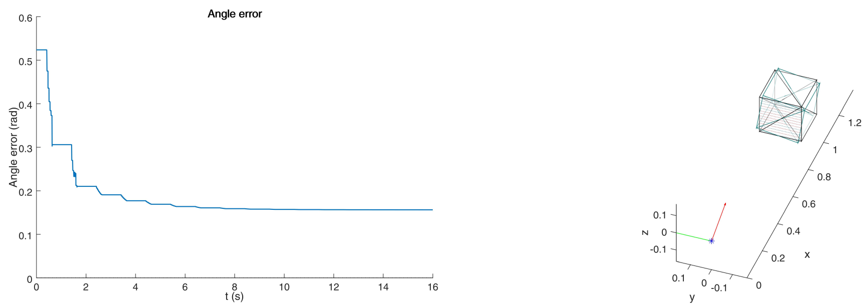
4.2.1 Orientation correction

4.2.1.1 Stationary cube

When testing the orientation update alone, the position and size update functions were turned off by setting their scaling factors to zero. The initial size and position of the state estimate were assigned the ground truth values. The *angle error* between the predicted and ground truth orientation was computed to quantify performance. To compute this, the rotation matrix required to map the orientation of the predicted cube to that of the ground truth cube was computed over time. This was converted to the scaled-axis representation. The angle error was defined as the magnitude of this scaled-axis vector, representing the angular error between the predicted and ground truth orientation.

The results presented in Figure 4.9 are from trials where the cube was stationary and three faces were visible. Similar performance was achieved when two faces were visible. When no noise was present, the angle error quickly converged to zero. In fact, for the case of orientation correction alone, the observer guarantees global convergence. When noise was present, the error converged to the noise floor and exhibited a stable fluctuation about this limit. Given the mean and standard deviation of the Gaussian distributed range error, the angle error would be expected to cross zero once the error converged to the noise floor. The reason this does not occur is due to small error in the orientation axis between the prediction and ground truth.

A limitation of the orientation update scheme was revealed when only a single cube face was visible to the sensor. The normals ($\hat{\mathbf{n}}$ and $\tilde{\mathbf{n}}$ in Figure 4.4) estimated from the predicted and measured intersection points were used to determine the rotation correction axis. However,



(a) (b)

Figure 4.8: Orientation update when a single cube face visible leads to (a) angle error, depicted in final the state estimate (b).

this information is insufficient to correct for angular error about these normals themselves, as shown in Figure 4.8.

An approach considered to account for this was to apply a rotation to the predicted cube about the normal axis if the visibility pattern indicted by the set of indexes $\mathbf{u}(t)$ was not congruent for the prediction and measurement. This would effectively rotate the predicted cube until the faces aligned. However, this scheme was not used as it would later interfere with the performance of the orientation and size updates. Furthermore, it was unstable under noisy conditions and caused rotation away from the correct orientation. The recommendation for future work is to use the visibility pattern to estimate the actual rotation of the predicted and measured cubes about this normal axis before applying the necessary correction.

4.2.1.2 Moving cube

Figure 4.10 shows the angular error for a trial where the ground truth cube rotated with an angular velocity of approximately 0.0327 rad/s such that 3 faces were visible to the sensor. The observer was given the ground truth initial orientation but an incorrect initial angular velocity of zero. Figure 4.10(a) shows that for noiseless measurements where orientation update is performed via the screw matrix, the observer is able to estimate the angular velocity of the cube and match its orientation. Again for noiseless measurements, Figure 4.10(b) shows that updating via the twist matrix initially provides a slower but smoother estimation of orientation. In trying to match the rotation of the cube, the angular velocity estimate overshoots the ground truth angular velocity. Initially, the angle error grows so the angular velocity increases to correct this. However, there is a delay in the

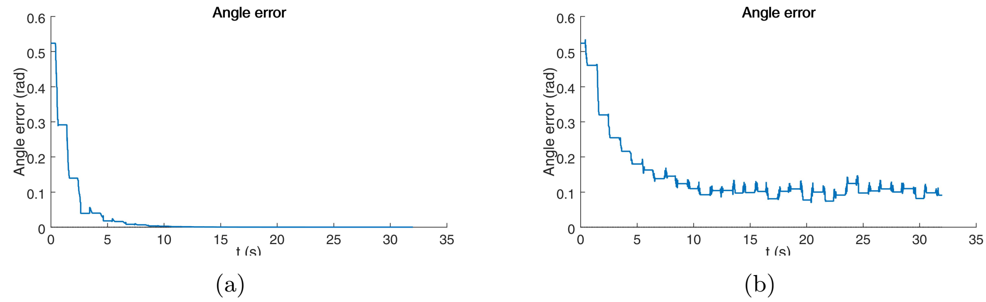


Figure 4.9: Angle error for (a) noiseless and (b) noisy measurements of a stationary cube with three faces visible to the sensor

response of the orientation correction that causes the angular velocity to grow too large. The overcompensation then occurs in the other direction and the error diverges. For noisy measurements, update via the screw in Figure 4.10(c) and twist in Figure 4.10(b) is unable to closely track the rotation of the cube. Updating via the twist matrix again results in overshoot.

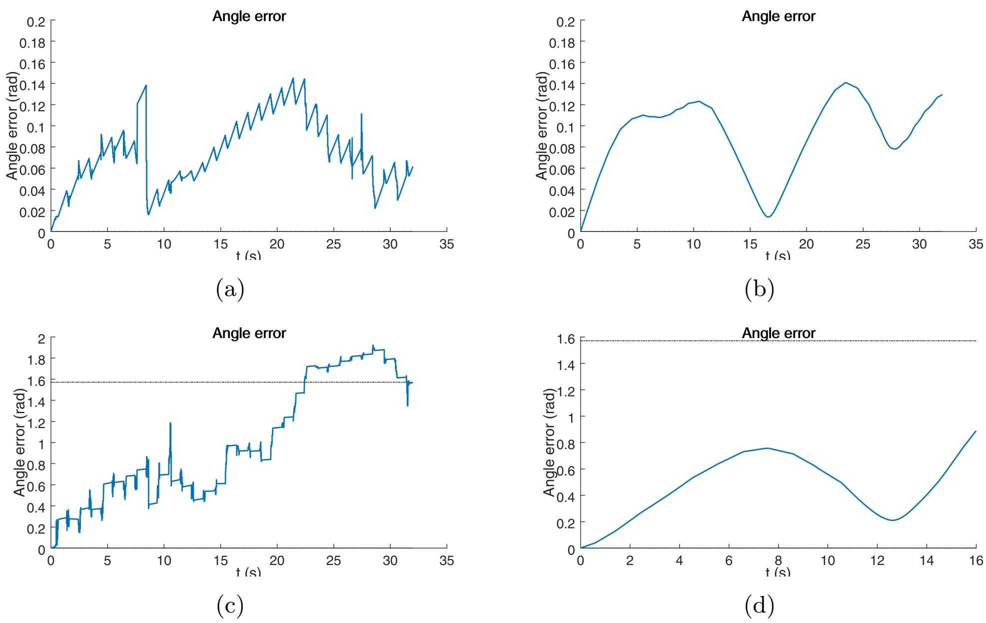


Figure 4.10: Angle error when cube was rotating with angular velocity of approximately 0.0327 rad/s such that three faces were visible to the sensor. Noiseless measurements were used to update the orientation via the (a) screw matrix and (b) twist matrix. Noisy measurements were used to update the orientation via the (c) screw matrix and (d) twist matrix.

4.2.2 Position correction

Figure 4.11 shows the effect of the scale p_{scale} on the convergence behaviour of the position error. In these trials, one face of the cube was visible to the sensor. Figure 4.11(a) shows that for noiseless measurements, the position error converges to a region where it begins to oscillate. Due to the dynamics of the scanning sensor, the size of the position update vector is too large and the estimate is unable to exactly align with the ground truth. Eventually, the estimate happens to align with the ground truth and the position error drops to zero.

Figure 4.11(b) shows that for noisy measurements, the update gain $p_{scale} = 0.01$ used in (a) is too large. A large random disturbance causes the position error to diverge so far that there is no longer any overlap between the estimated and ground truth cubes. At this point, no update can be computed. By reducing the gain to $p_{scale} = 0.001$, the position error for noisy measurements is able to converge slowly towards the noise floor.

In Figure 4.12, noiseless measurements are taken of a stationary cube with two faces visible to the sensor. Though the error shrinks slightly at first, the basin of attraction points away from the ground truth position. This occurs because the observer attempts to align the cube state estimate with the wrong face of the ground truth cube. The computed update vector will point in a direction orthogonal to the desired direction. The position update function is only successful when a single cube face is visible to the sensor. Future improvements should focus on designing a position update that is invariant to orientation.

4.2.3 Size correction

The size correction function is extremely robust, able to globally converge to the noise floor regardless of the number of cube faces visible. Figure 4.13 shows that the *size error ratio* (the ratio between the size error and ground truth size) converges to zero for noisy measurements of a stationary cube where three faces are visible to the sensor. In Figure 4.14, noisy measurements were taken of a cube rotating at 0.0327 rad/s and translating at 0.0094 m/s such that three faces were visible to the sensor. The ground truth twist was given as the initial twist estimate to allow the predicted cube to match the motion of the ground truth cube. In this case, the size error ratio still converged to the noise floor. The speed of the convergence suggests that the correction is dominated by case 2 of the size update function. This update scheme uses the mean range difference effectively measures the size difference. On the other hand, case 1 relies on the rarer occurrence of misaligned edges between the prediction and ground truth being observed to determine. However, case 1 will be required when there is non-zero position error, since a situation may arise

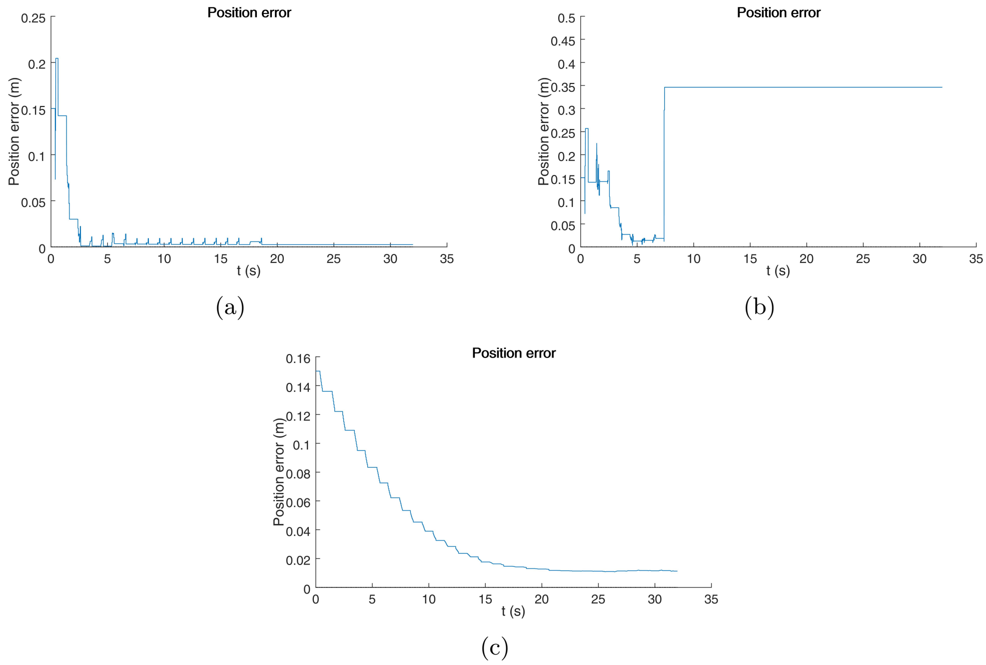


Figure 4.11: Position error for stationary cube with a single face visible to the sensor. Position error (a) converges quickly for noiseless measurements when $p_{scale} = 0.01$. Position error (b) is unstable for noisy measurements if $p_{scale} = 0.01$. Position error (c) convergences to noise floor for noisy measurements when $p_{scale} = 0.001$.

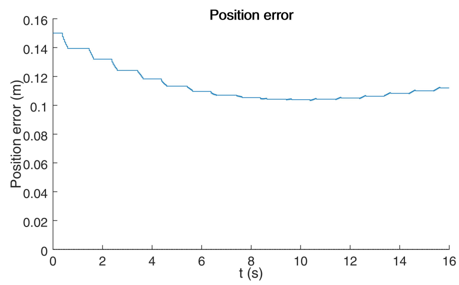


Figure 4.12: Position error does not converge when observing stationary cube with two faces visible to the sensor.

where a combination of position and size error means that the predicted and ground truth cube faces lie the same distance from the sensor.

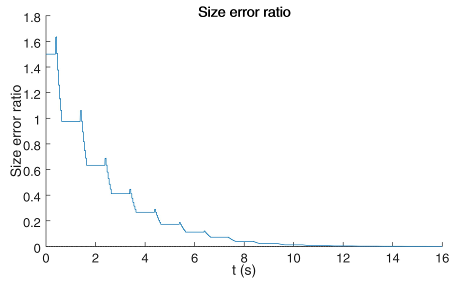


Figure 4.13: For noisy measurements of a stationary cube with three faces visible to the sensor, the size error ratio converges to the noise floor

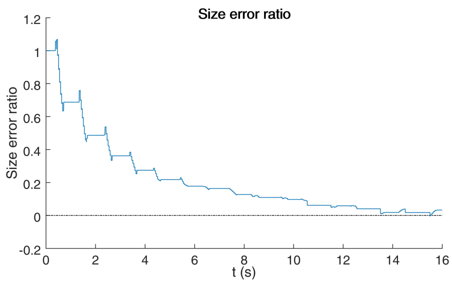


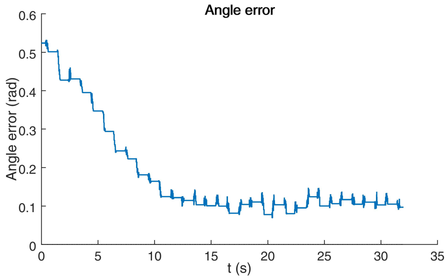
Figure 4.14: For noisy measurements of a cube rotating at 0.0327 rad/s and translating at 0.0094 m/s such that three faces are visible to the sensor, the size error ratio converges to the noise floor.

4.2.4 Orientation and size correction

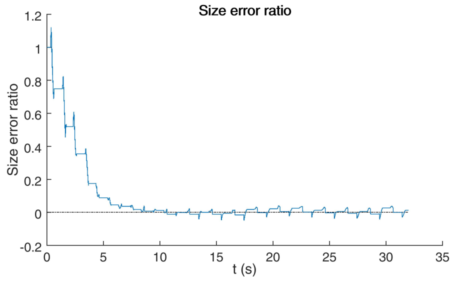
As the orientation and size update functions were effective individually, their ability to work simultaneously was tested.

For stationary cubes, the observer was able convergent for noiseless and noisy results, regardless of the number of cube faces visible to the sensor. Figure 4.15 shows the angle error and size error ratio converging to the noise floor for noisy measurements of a stationary cube where three faces were visible to the sensor.

A more systematic test was conducted with a wide range of initial conditions to verify this global convergence. Orientation and size correction were tested for initial angular error ranging from a minimum of 0 to a maximum of $\pi/4$ radians, and a minimum size error ratio of -0.5 to a maximum of 2. Noiseless range measurements were used. Figure 4.16 shows the time taken to converge to within 1% error. Comparing Figures 4.16(a) and 4.16(b) shows that the size error ratio always converges before the angle error. The magnitude of

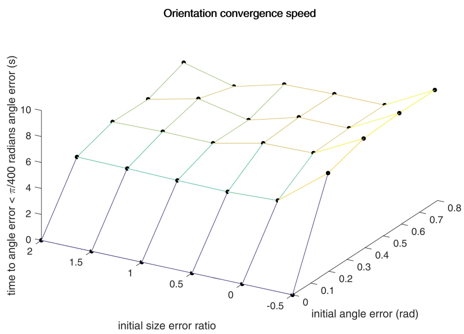


(a)

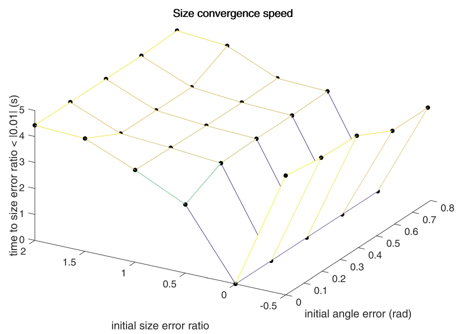


(b)

Figure 4.15: For noisy measurements of stationary cube with 3 faces visible to sensor, the angle error (a) and size error ratio (b) both converge to the noise floor.



(a)



(b)

Figure 4.16: Time taken to converge to (a) angle error $< \pi/400$ radians (1% of maximum $\pi/4$ angle error) and (b) size error ratio $< |0.01|$ for range of initial conditions

the initial angle error has little effect on the speed of the size error ratio convergence. The size correction only has a significant impact on the speed of orientation correction when the initial size estimate is smaller than the ground truth. This global convergence was also observed when estimating the state of a stationary cube with two faces visible, but not one - due to the limitation in the orientation update described in Section 4.2.1.1.

For noiseless measurements of a rotating cube, Figures 4.17(a) and 4.17(b) show that the observer is able to track the orientation of the cube when updating via the screw, and the size error ratio converges to the noise floor. Though the size error ratio in Figure 4.17(d) converges to the noise floor, Figure 4.17(c) shows that updating orientation via the twist results in overshoot. This also occurred when size correction was not being performed. Because the convergence of the size error ratio is so rapid, the performance of the observer when correcting both orientation and size is limited only by the effectiveness of the orientation update.

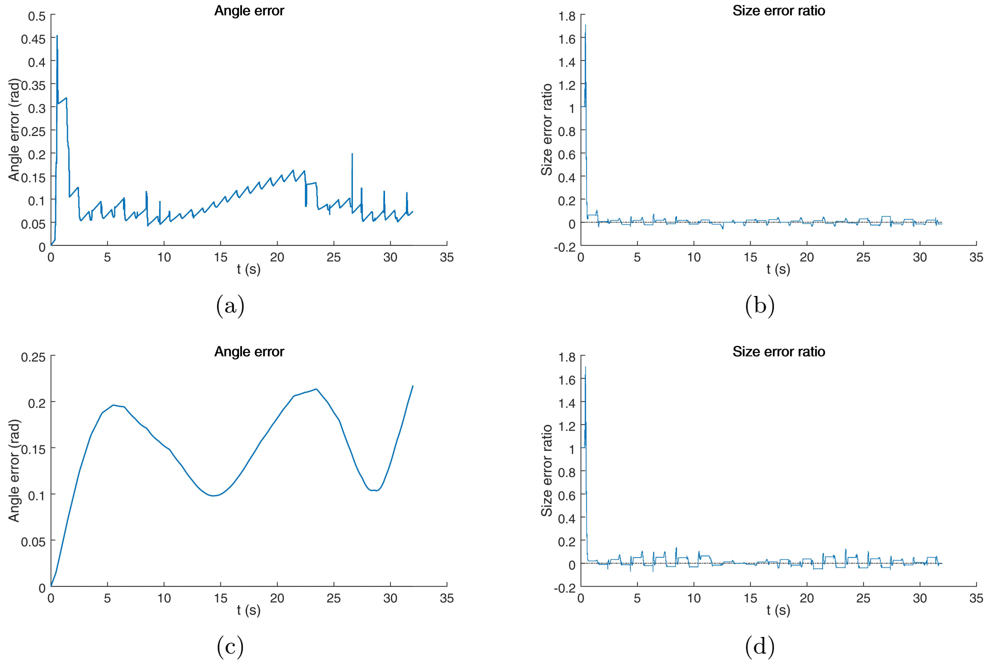


Figure 4.17: Noiseless measurements of cube rotating at 0.0327 rad/s such that 3 faces visible to sensor. Angle error (a) and size error ratio (b) when orientation updated via screw. Angle error (c) and size error ratio (d) when orientation updated via wrench.

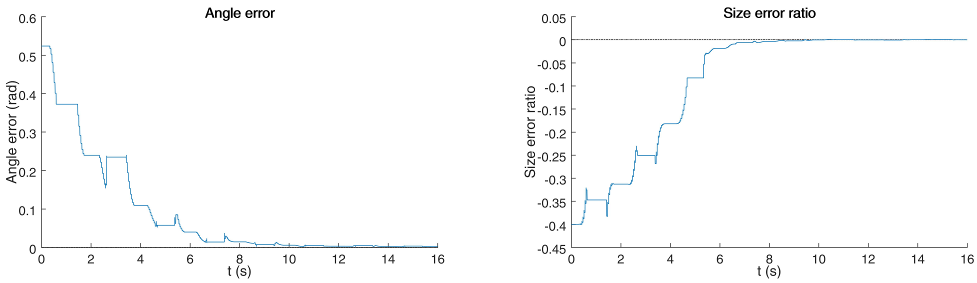
4.2.5 Discussion

4.2.5.1 Performance assessment

The analysis in Figure 4.16 shows that the observer is globally convergent when the position error is zero. This is only the case if the position update is turned off by setting $p_{scale} = 0$. Even with a correct initial position, attempting to correct orientation and size while the position update is active leads to divergence: the position update is unstable about the point $p_{error} = 0$.

To achieve global convergence, or even a larger basin of attraction for the position update, the update function must be invariant to actions of $\mathbf{SE}(3)$. The current position update is not invariant to actions of $\mathbf{SO}(3)$, as demonstrated by its divergence when more than a single cube face was visible.

Another strength of the observer design is that it does not actually rely on the specific geometry of the target object. Figure 4.18 shows that the orientation and size correction converges when the object to be estimated is a tetrahedron with side length $s = 0.5$m. The observer will likely give sensible results if the target object is a platonic solid due to their



(a) (b)

Figure 4.18: Angle error (a) and size error ratio (b) for the observer using noiseless measurements to estimate the state of a tetrahedron of side length $s = 0.5$m

rotational symmetry.

These results show that sparse range measurements from a scanning sensor can be indeed used to densely measure an infinite dimensional state. The dynamics of the sensor are particularly important in ensuring that the sparse measurements are form an arrangement suitable for dense estimation.

4.2.5.2 Improvements and future work

Though updating the orientation via the twist gives a smoother estimation in comparison to the screw update, this results in overshoot. In control theory, this problem is solved with *feed-forward control* which anticipates overshoot and corrects for it before it can occur. A topic of future research would be to investigate if an observer theory analogue to feed-forward control can be designed. A possible starting point could be to investigate applying combinations of screw and twist updates that combine the responsiveness of the screw update and the smoothness of the twist update.

These simulated results showed that the size and orientation update functions in this initial observer implementation show promise. To improve performance, a more robust position update function must be designed. A possible approach would be to augment the cube state with a measure of the centre of mass of a history of measured points. Over time, the difference in the estimate and measurement of this variable would produce a more accurate position update vector. This scheme would also be much more robust to noise.

Once a working position update function is implemented, the next step is to combine the position and orientation update functions. Such a function would act on the screw, twist or wrench matrices as a whole, rather than correcting only the linear or angular terms.

Designing this innovation function to be invariant under actions of $\mathbf{SE}(3)$ would result in improved global convergence properties.

A symmetry preserving observer would allow design methods for linear, infinite-dimensional observers to be utilised in nonlinear systems. This would then allow the environment to be represented as infinite dimensional state. The observer would not need to separate the target object and the background. Instead, an estimate of the entire depth field could be computed.

Chapter 5

Experimental data

In order to validate the performance of the observer implementation, experimental data was collected with a Hokuyo UBG-04LX-F01 scanning laser range-finder.

Measurements were taken to:

- build a model of the noise characteristics of the Hokuyo UBG-04LX-F01 in order to more accurately simulate the performance of the observer;
- observe the motion of a moving cube of known state to test the observer in real-world conditions.

Section 5.1 details how measurements were taken to develop the noise model. Section 5.2 describes how experimental range measurements were taken and how the ground truth cube state was determined. This work is still ongoing as the data must be calibrated before it can be used to assess the performance of the observer.

5.1 Sensor noise characterisation

An accurate range sensor simulation must include a model for the error distribution of the measurements. A noise model for the Hokuyo UBG-04LX-F01 was developed by Park et al. [29]. The effect of range and incidence angles on the error was measured, but a unified model combining both was not provided. Furthermore, [29] showed that the measurement error depends highly on the texture and colour of the surface measured. To accurately model the Hokuyo UBG-04LX-F01 for the usage case of this research, a wide

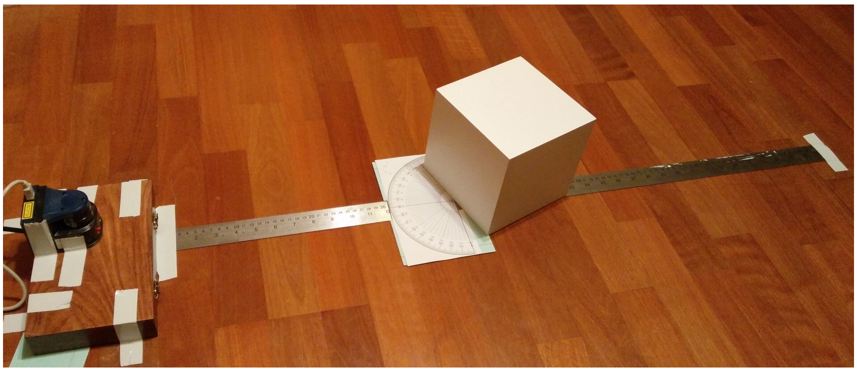


Figure 5.1: Experimental setup to measure noise at a different ranges and angle (lights turned off during measurement to eliminate error from variation in lighting conditions).

set of measurements using a specific surface were taken to determine the effect of range and incidence angle on the error distribution.

5.1.1 Measurement setup

A flat surface was painted matte white. The surface was placed perpendicular to the ground and at a known distance and angle with respect to the range sensor. 1200 samples of the measured distance to the surface were taken.

For this research, the cube is likely to be placed within 1.5m from the sensor and at any orientation. The range error distribution for these conditions should be measured. The distance from the sensor to the measurement surface was thus varied in 50mm increments between 250mm and 1750mm, to an accuracy of ± 1mm. At each of these ranges the incidence angles was varied in $20°$ increments from $0°$ to $80°$ to an accuracy of $\pm 0.5°$. The physical setup is shown in Figure 5.1.

5.1.2 Results

The range error $e_r = r - \tilde{r}$ was computed. The distributions of this error for varying ranges and angles is shown in Figure 5.2. The range error is approximately normally distributed.

The mean range error as function of r and θ is shown in Figure 5.3. The standard deviation of the range errors as function of r and θ is shown in Figure 5.4.

Figures 5.3(a) and 5.4(a) show that the mean error and error standard deviation increase significantly when $\theta > 75°$ and $r > 0.8$m. This can be explained by considering what

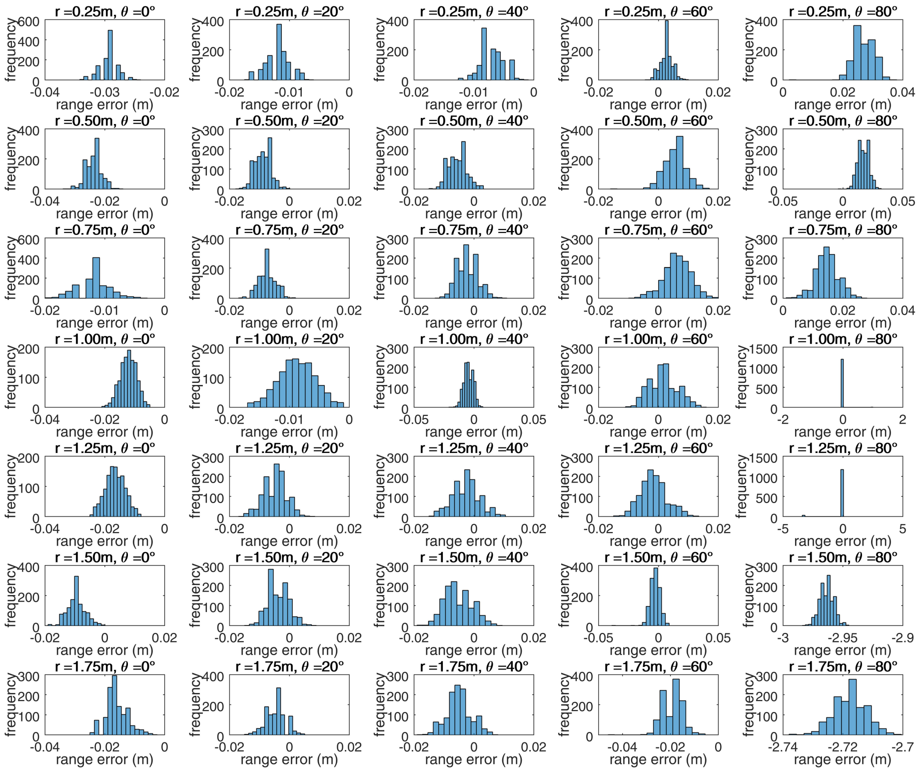
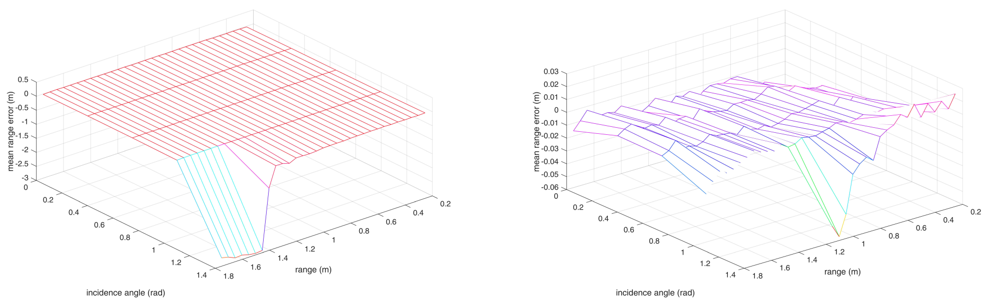
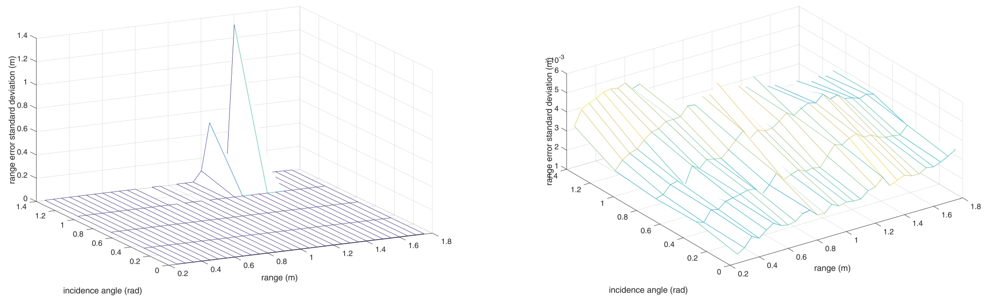


Figure 5.2: Sensor noise function $f_{UBG}(r, \theta)$ approximately normally distributed



(a) (b)

Figure 5.3: Mean range error vs (r, θ) showing (a) large error at high angles and range, and (b) overall trend.



(a) (b)

Figure 5.4: Range error standard deviation σ vs (r, θ) showing (a) outliers/large σ at high angles and range, and (b) overall trend.

happens to the laser beam under these conditions. Though it has been idealised as a ray in the simulation, the laser has a nonzero beam width. Thus, as θ increases, one side of the beam will encounter the surface before the centre of the beam. A portion of the light will reach the sensor earlier, though the total amount of light will be reduced as the angle increases. This earlier reflected light will result in a shorter range measurement, but less reflected light will cause a longer range measurement. For angles greater than 75° and ranges greater than 0.8m, the reflected light is insufficient to allow a range measurement. The sensor returns the maximum possible range measurement of 4095mm. In modelling the noise, range measurements for $\theta > 75°$ and $r > 0.8$m are discarded.

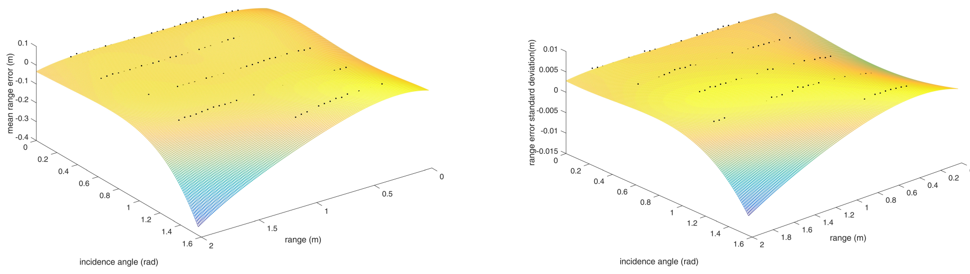
These results are corroborated by [29] who reported difficulty in acquiring measurements for high angles and modelled the noise distribution as Gaussian.

It was assumed that the noise distribution for an incidence angle θ would be identical to $-\theta$. However, the fact that the sensor's scan direction rotates in a single direction may mean this is not the case. Including the effect of incidence angles from $\pi/2$ to $\pi/2$ would provide a more accurate noise model for future work.

The 4th degree polynomial surfaces in equations 5.2 and 5.3 were fitted to the adjusted set of data points using Matlab's curve fitting tool. The surfaces and the goodness of fit are shown in Figure 5.5.

5.1.2.1 Gaussian noise model

$$\tilde{r}(r, \theta) = \begin{cases} f_s(r, \theta, \phi(k)) = r + \mathcal{N}(\mu, \sigma) & \theta \leq 75^\circ \text{ or } r \leq 0.8 \\ NaN & \theta > 75^\circ \text{ and } r > 0.8 \end{cases} \tag{5.1}$$



(a) (b)

Figure 5.5: polynomials fitted to range error mean & standard deviation data points to model noise. (a) SSE: 0.003234, R-square: 0.8447, Adjusted R-square: 0.8278, RMSE: 0.005027(b) SSE: 7.592e-06, R-square: 0.9196, Adjusted R-square: 0.9103, RMSE: 0.0002515

where

$$\begin{aligned}\mu =& a_{00} + a_{10}r + a_{01}\theta + a_{20}r^2 + a_{11}r\theta + a_{02}\theta^2 \\ &+ a_{30}r^3 + a_{21}r^2\theta + a_{12}r\theta^2 + a_{03}\theta^3 + a_{40}r^4 \\ &+ a_{31}r^3\theta + a_{22}r^2\theta^2 + a_{13}r\theta^3 + a_{04}\theta^4\end{aligned} \tag{5.2}$$

$$\begin{aligned}\sigma =& b_{00} + b_{10}r + b_{01}\theta + b_{20}r^2 + b_{11}r\theta + b_{02}\theta^2 \\ &+ b_{30}r^3 + b_{21}r^2\theta + b_{12}r\theta^2 + b_{03}\theta^3 + a_{40}r^4 \\ &+ b_{31}r^3\theta + b_{22}r^2\theta^2 + b_{13}r\theta^3 + b_{04}\theta^4\end{aligned} \tag{5.3}$$

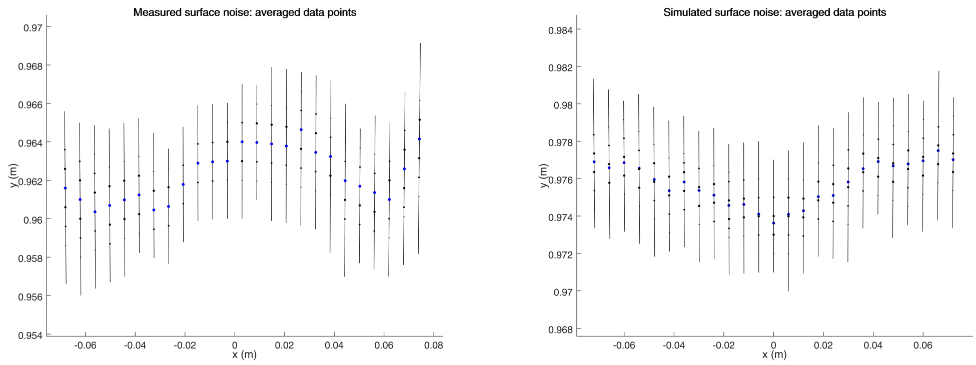
and coefficients a_{ij} and b_{ij} are provided in tables 5.1 and 5.2 respectively.

Table 5.1: a_{ij} coefficients

	j_0	j_1	j_2	j_3	j_4
i_0	-0.06529	0.2126	-0.533	0.4629	-0.1223
i_1	0.2024	-0.1906	0.4006	-0.1791	0
i_2	-0.3074	0.0228	-0.0716	0	0
i_3	0.2053	0.01455	0	0	0
i_4	-0.04912	0	0	0	0

Table 5.2: b_{ij} coefficients

	j_0	j_1	j_2	j_3	j_4
i_0	0.001242	0.2126	-0.01128	0.01162	-0.002746
i_1	0.00352	0.006146	0.01021	-0.007316	0
i_2	-0.005138	-0.00626	-0.0005068	0	0
i_3	0.004067	0.001337	0	0	0
i_4	-0.001092	0	0	0	0



(a) (b)

Figure 5.6: Comparision of (a) measured and (b) simulated surface noise. Point distribution along radial lines is shown as quintiles of error.

5.1.2.2 Surface noise

An additional source of range error was observed and found to be mostly independent of r and θ. This may be caused by surface properties of the environment, though the error is larger than expected in this case. A possible explanation is compensation performed by the sensor to produce globally straight lines. While flat surfaces do appear flat from a distance, locally there are regular variations in depth as shown in Figure 5.6(a).

This surface noise was modelled with a random walk function

$$e_{surface} = a \left(\sum_{n=1}^{n_{Steps}} -1 + 2 \ \lfloor \mathcal{R} \rfloor \right) \tag{5.4}$$

where $\mathcal{R}$ is a random variable following a uniform distribution on [0,1]. A step size $a = 0.0005m$ was used. Figure 5.6 shows that this model accurately models the measured surface variations. It should be noted that the measured variation appears concave while the simulated noise appears convex. This is due to the nature of the random walk noise. Over a large sample, both concave and convex surface noise is observed in real-world measurements and the simulated random walk.

5.2 Collection of observer performance testing data

Experimental data was collected to assess the performance of the observer under real-world, less-than-ideal conditions.

Figure 5.7: setup to collect experimental data

5.2.1 Setup

The experimental setup is shown in Figure 5.7.

The Hokuyo UBG-04LX-F01 sensor was mounted to a tripod. The sensor was panned up and down manually while the measured range and time recorded. This panning along trajectory with the sensor's scanning behaviour is what allows it to densely estimate the infinite-dimensional environment depth field. To compute the elevation angle of the sensor, the range measurements of a portion of the blank wall at a known distance were used. The range to this wall was measured at elevations increasing from $-25°$ to $25°$ in $5°$ increments for calibration purposes.

The target object was a cube of 100mm side length made from medium-density fibreboard. The cube was spray painted matte white - the same surface used for the measurements in the sensor noise modelling in Section 5.1.

The cube was placed in the gripper of a Kinova Jaco robotic arm. The arm was manually manipulated to produce stationary, rotating, translating and combined motions. The joint angles of the sensor over time were recorded. From this data, the forward kinematic model for the arm was used to compute the ground truth pose of the cube over time.

5.2.2 Results

Calibration of the data to determine the elevation angle of the sensor is yet to be completed. However, it is unlikely that the current implementation of the observer will be able to estimate the cube state from this data. Because the cube is held by the gripper of the arm, neither the range or continuity assumptions in Section 4.1.5.2 hold.

An infinite-dimensional observer measuring the entire depth field would give a better estimate of the state of the cube. A symmetry-preserving observer would likely be more robust to the significant levels of noise in the data set.

Conclusion

An observer has been designed to estimate the state of a rigid cube from sparse range measurements. It has been shown that by appropriately defining the sensor trajectory, the sparse range measurements can be used to approximate those of a dense sensor.

Performance testing suggests that the observer is almost globally convergent when correcting the orientation and size of stationary cubes for trajectories where two or three cube faces are visible. This implementation also reveals the limitations of an observer update function that does not consider symmetry. The position update only works in the special case that a single cube face is visible. In order to simultaneously correct position and orientation, the update function must be invariant to actions of $\mathbf{SE}(3)$. It is recommended that the position of the cube be updated by computing the centre of mass of a history of predicted and measured points, and defining the position update as the difference of these. This history of points would be added as an augmented state variable. A similar approach could be used to correct the orientation and size of the cube state estimate. Such an approach would be far more resistant to noise.

It is recommended that the position and orientation updates are combined into a function that acts on either the screw, twist or wrench as a whole, rather than extracting the rotation matrix or position vector. Such an update function should be invariant to actions of $\mathbf{SE}(3)$, leading to improved convergence properties over a wider region of trajectories.

A major simplification of this observer design was the representation of the infinite-dimensional environment as a single target object and background object. Future work should attempt to model a more complex environment that is a better representation of an infinite-dimensional system. The triangular mesh method of modelling rigid bodies in the simulation implementation would allow for complex, deformable surfaces to be represented with few changes required to the code. Rather than separating the range measurements corresponding to the cube and background, all measurements should be used. The observer update function could then be driven by the difference in measured and predicted ranges, rather than the separate values. The benefit of this addition is that the observer would have a form more like the traditional Luenberger observer that is used in existing symmetry-preserving observer design methodologies [17, 21].

Bibliography

[1] P. Stavroulakis and P. Sarachik, "Design of optimal controllers for distributed systems using finite dimensional state observers," in *Decision and Control including the 12th Symposium on Adaptive Processes, 1973 IEEE Conference on.* IEEE, 1973, pp. 105–109.

[2] L. Meirovitch and H. Baruh, "On the problem of observation spillover in self-adjoint distributed-parameter systems," *Journal of Optimization Theory and Applications*, vol. 39, no. 2, pp. 269–291, 1983.

[3] C. Harkort and J. Deutscher, "Finite-dimensional observer-based control of linear distributed parameter systems using cascaded output observers," *International Journal of Control*, vol. 84, no. 1, pp. 107–122, 2011.

[4] R. V. Gressang and G. B. Lamont, "Observers for systems characterized by semigroups," *Automatic Control, IEEE Transactions on*, vol. 20, no. 4, pp. 523–528, 1975.

[5] A. Smyshlyaev and M. Krstic, "Backstepping observers for a class of parabolic pdes," *Systems & Control Letters*, vol. 54, no. 7, pp. 613–625, 2005.

[6] K. Ramdani, M. Tucsnak, and G. Weiss, "Recovering the initial state of an infinite-dimensional system using observers," *Automatica*, vol. 46, no. 10, pp. 1616–1625, 2010.

[7] G. Haine, "Recovering the observable part of the initial data of an infinite-dimensional linear system with skew-adjoint generator," *Mathematics of Control, Signals, and Systems*, vol. 26, no. 3, pp. 435–462, 2014.

[8] J. Primbs, "Survey of nonlinear observer design techniques," 1996.

[9] C.-Z. Xu, P. Ligarius, and J.-P. Gauthier, “An observer for infinite-dimensional dissipative bilinear systems,” *Computers & Mathematics with Applications*, vol. 29, no. 7, pp. 13–21, 1995.

[10] H. Bounit and H. Hammouri, “Observers for infinite dimensional bilin ear, systems,” *European journal of control*, vol. 3, no. 4, pp. 325–339, 1997.

[11] S. I. Marcus, “Algebraic and geometric methods in nonlinear filtering,” *SIAM Journal on Control and Optimization*, vol. 22, no. 6, pp. 817–844, 1984.

[12] S. Salcudean, “A globally convergent angular velocity observer for rigid body motion,” *Automatic Control, IEEE Transactions on*, vol. 36, no. 12, pp. 1493–1497, 1991.

[13] N. Aghannan and P. Rouchon, “On invariant asymptotic observers,” in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2. IEEE, 2002, pp. 1479–1484.

[14] D. S. Maithripala, W. P. Dayawansa, and J. M. Berg, “Intrinsic observer-based stabilization for simple mechanical systems on lie groups,” *SIAM journal on control and optimization*, vol. 44, no. 5, pp. 1691–1711, 2005.

[15] S. Bonnabel and P. Rouchon, “On invariant observers,” in *Control and observer design for nonlinear finite and infinite dimensional systems.* Springer, 2005, pp. 53–65.

[16] S. Bonnabel, P. Martin, and P. Rouchon, “Symmetry-preserving observers,” *Automatic Control, IEEE Transactions on*, vol. 53, no. 11, pp. 2514–2526, 2008.

[17] ——, “Non-linear symmetry-preserving observers on lie groups,” *Automatic Control, IEEE Transactions on*, vol. 54, no. 7, pp. 1709–1713, 2009.

[18] D. Auroux and S. Bonnabel, “Symmetry-based observers for some water-tank problems,” *Automatic Control, IEEE Transactions on*, vol. 56, no. 5, pp. 1046–1058, 2011.

[19] R. Mahony, T. Hamel, J. Trumpf, and C. Lageman, “Nonlinear attitude observers on so (3) for complementary and compatible measurements: A theoretical study,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on.* IEEE, 2009, pp. 6407–6412.

[20] J. Trumpf, R. Mahony, T. Hamel, and C. Lageman, "Analysis of non-linear attitude observers for time-varying reference measurements," *Automatic Control, IEEE Transactions on*, vol. 57, no. 11, pp. 2789–2800, 2012.

[21] R. Mahony, J. Trumpf, and T. Hamel, "Observers for kinematic systems with symmetry," in *Proceedings of the 9th IFAC symposium on nonlinear control systems*, vol. 9, 2013, pp. 617–633.

[22] S. P. Bhat and D. S. Bernstein, "A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon," *Systems & Control Letters*, vol. 39, no. 1, pp. 63–70, 2000.

[23] Y. Huang and J. Han, "Analysis and design for the second order nonlinear continuous extended states observer," *Chinese science bulletin*, vol. 45, no. 21, pp. 1938–1944, 2000.

[24] S. E. Talole, J. P. Kolhe, and S. B. Phadke, "Extended-state-observer-based control of flexible-joint system with experimental validation," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 4, pp. 1411–1419, 2010.

[25] N. Zarrouati-Vissière, "Augmented reality: the fusion of vision and navigation," Ph.D. dissertation, Ecole Nationale Supérieure des Mines de Paris, 2013.

[26] J. D. Adarve, D. J. Austin, and R. Mahony, "A filtering approach for computation of real-time dense optical-flow for robotic applications."

[27] D. G. Luenberger, "Observers for multivariable systems," *Automatic Control, IEEE Transactions on*, vol. 11, no. 2, pp. 190–197, 1966.

[28] R. Szeliski, "Estimating motion from sparse range data without correspondence." in *ICCV*, vol. 88, 1988, pp. 207–216.

[29] C.-S. Park, D. Kim, B.-J. You, and S.-R. Oh, "Characterization of the hokuyo ubg-04lx-f01 2d laser rangefinder," in *RO-MAN, 2010 IEEE*. IEEE, 2010, pp. 385–390.