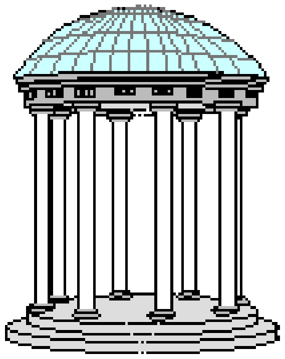
SCAAT: Incremental Tracking with Incomplete Information

TR96-051
October 1996

Gregory Francis Welch

Department of Computer Science
CB #3175, Sitterson Hall
UNC-Chapel Hill
Chapel Hill, NC 27599-3175



SCAAT: Incremental Tracking with Incomplete Information

by

Gregory Francis Welch

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill

1996

Approved by:

Gary Bishop
Dr. Gary Bishop, Adviser

[illegible]
Dr. Henry Fuchs

[illegible]
Vernon Chi, Reader

[illegible]
Dr. Anselmo Lastra

[illegible]
Dr. Russell Taylor, Reader

[illegible]
Dr. John Poulton

ABSTRACT

Gregory Francis Welch
SCAAT: Incremental Tracking with Incomplete Information
(Under the direction of T. Gary Bishop)

The Kalman filter provides a powerful mathematical framework within which a minimum mean-square-error estimate of a user's position and orientation can be tracked using a sequence of *single* sensor observations, as opposed to *groups* of observations. We refer to this new approach as *single-constraint-at-a-time* or SCAAT tracking. The method improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently *autocalibrating* mechanical or electrical devices. The method facilitates user motion prediction, multisensor data fusion, and in systems where the observations are only available sequentially it provides estimates at a higher rate and with lower latency than a multiple-constraint approach.

Improved accuracy is realized primarily for three reasons. First, the method avoids mathematically treating truly sequential observations as if they were simultaneous. Second, because each estimate is based on the observation of an individual device, perceived error (statistically unusual estimates) can be more directly attributed to the corresponding device. This can be used for concurrent autocalibration which can be elegantly incorporated into the existing Kalman filter. Third, the Kalman filter inherently addresses the effects of noisy device measurements. Beyond accuracy, the method nicely facilitates motion prediction because the Kalman filter already incorporates a model of the user's dynamics, and because it provides smoothed estimates of the user state, including potentially unmeasured elements. Finally, in systems where the observations are only available sequentially, the method can be used to weave together information from individual devices in a very flexible manner, producing a new estimate as soon as each individual observation becomes available, thus facilitating multisensor data fusion and improving the estimate rates and latencies.

The most significant aspect of this work is the introduction and exploration of the SCAAT approach to 3D tracking for *virtual environments*. However I also believe that this work may prove to be of interest to the larger scientific and engineering community in addressing a more general class of tracking and estimation problems.

ACKNOWLEDGEMENTS

Thanks to

Gary Bishop for being my advisor and friend, for begin creative, insightful, and humorous, and for sharing with me his original idea for *one-LED-at-a-time tracking*;

Gary Bishop, Vernon Chi, Henry Fuchs, Anselmo Lastra, John Poulton, and Russell Taylor for serving on my doctoral dissertation committee;

Vernon Chi and Russell Taylor for their extremely valuable contributions to my dissertation while acting as official readers;

Russell Taylor for consenting to replace Henry Fuchs as an official reader following Henry's unexpected surgery just prior to my defense;

John Poulton for his support and humor, and for agreeing to serve on my committee following Henry Fuchs' surgery (in case Henry could not fully participate);

Henry Fuchs for remaining on my committee (in a limited capacity) in spite of orders to "drop everything" following his surgery;

Dr. Gil Crouse, Jr. of BBN Systems and Technologies (Arlington, VA) for introducing me to the bearings-only tracking work done in the 1970s and 1980s;

Mark Mine for enticing me to come to UNC-CH, for Mark and his wife Sandra sharing their home with me and my wife Linda for many months after our arrival in Chapel Hill, and Mark most recently for helping arrive at the phrase *single-constraint-at-a-time*;

and most of all

Linda, my wife, for her support, patience, and warmth, and in particular for commuting between our home in Carrboro and her job in Clayton each day (55 miles, approximately one hour) so that I might spend less of *my* time commuting. I will happily chauffeur us around for the rest of my life.

Financial support for this work came from the following grants:

DARPA/ETO contract no. DABT 63-93-C-0048, "Enabling Technologies and Application Demonstrations for Synthetic Environments", Principle Investigators Frederick P. Brooks Jr. and Henry Fuchs (UNC Chapel Hill);

NSF Cooperative Agreement no. ASC-8920219 and DARPA: "Science and Technology Center for Computer Graphics and Scientific Visualization", Center Director A. van Dam (Brown University), Principle Investigators Andy van Dam, Al Barr (California Institute of Technology), Don Greenberg (Cornell University), Henry Fuchs (UNC Chapel Hill), and Rich Riesenfeld (University of Utah).

(This page intentionally left blank.)

TABLE OF CONTENTS

SCAAT: Incremental Tracking with Incomplete Information 1

ABSTRACT iii

. . . . iii

. . . . iii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS vi

LIST OF TABLES xiii

LIST OF FIGURES xiv

LIST OF ABBREVIATIONS xviii

GENERAL NOTATION xix

LIST OF SYMBOLS xx

GREEK SYMBOLS xxviii

Chapter 1. Introduction 34

1.1 Narrative, Names, and Nomenclature 34

1.2 Observability and Incomplete Information 36

1.3 An Unusual Approach to Tracking 42

1.4 Contribution (Thesis Statement) 43

Chapter 2. Motivation 44

2.1 Why a Kalman Filter? 44

2.1.1 Filtering 44

2.1.2 Data Fusion 46

2.1.3 Prediction 47

2.1.4 Calibration 48

2.2 Temporal Concerns 48

2.2.1 Fewer Observations, Improved Timing and Accuracy 48

2.2.2 Estimation Rates and Latencies 50

2.2.3 Measurement Collection 50

2.2.4 Computational Complexity 52

2.3 The Simultaneity Assumption 53

2.3.1 Mathematics Unaware that Target is in Motion 53

2.3.2 When is it a Problem? 54

2.4 Hybrid Systems and Multisensor Data Fusion 56

2.4.1 Hybrid Systems, the Past and the Future 56

2.4.2 Multisensor Data Fusion 56

2.4.3 A Simple Hybrid Example 57

2.5 Source and Sensor Autocalibration 58

2.5.1 What to Calibrate? 58

2.5.2 When and Where to Calibrate? 60

2.5.3 Calibration vs. Autocalibration 61

2.5.4 Why SCAAT? 61

2.6 Broad Applicability 62

2.7 Putting Things In Perspective 63

Chapter 3. Related Work 64

3.1 Bearings-Only Tracking 64

3.2 Sequential Updates 65

3.3 The Iterated Kalman Filter 67

3.4 The Collinearity Method 68

3.5 Autocalibration (Gottschalk & Hughes) 69

3.6 The SELSPOT System 70

3.7 MCAAT Estimators 70

3.8 Kalman Filters and GPS 71

Chapter 4. The SCAAT Method 72

4.1 The Kalman Filter in Brief 72

4.2 The SCAAT Kalman Filter 73

4.2.1 The Dynamic (Process) Model 73

4.2.2 The Model State and Discrete Transitions 75

4.2.3 The Process Noise 76

4.2.4 The Measurement Model 78

4.2.5 The Measurement Noise 81

4.2.6 The Measurement Jacobian 82

4.2.7 The State Error Covariance 83

4.3 The SCAAT Algorithm for Tracking 83

4.3.1 Time Update. 83

4.3.2 Measurement Prediction 84

4.3.3 The Kalman Gain 87

4.3.4 The Measurement Residual 92

4.3.5 Correct the State and Update the Error Covariance 94

4.3.6 Update External Orientation . . . 96

4.3.7 The SCAAT Algorithm Summary . . . 97

4.3.8 Discussion . . . 98

4.4 SCAAT Autocalibration . . . 99

4.4.1 An Overview . . . 99

4.4.2 Device Filters . . . 100

4.4.3 Revised Tracking Algorithm . . . 100

4.4.4 Discussion . . . 102

4.5 Code Optimization Strategies . . . 103

4.5.1 Jacobian Evaluation . . . 103

4.5.2 Sparse Matrices . . . 104

4.5.3 Symmetry . . . 105

4.5.4 Matrix Inversion . . . 105

4.5.5 Small Angles . . . 105

4.6 Source & Sensor Ordering Schemes . . . 106

Chapter 5. Mathematical Analysis . . . 107

5.1 Filter Stability . . . 107

5.1.1 General Conditions for Kalman Filter Stability . . . 107

5.1.2 Locally Unobservable Measurement Systems . . . 107

5.1.3 Complete Conditions for SCAAT Filter Stability . . . 108

5.1.4 SCAAT Filter With Calibration . . . 111

5.1.5 Inaccuracy of the Dynamic Model . . . 111

5.2 Computational Complexity . . . 112

5.2.1 EKF Methods (SCAAT and MCAAT) 113

5.2.2 The Collinearity Method 116

5.2.3 Collinearity vs. the EKF Methods 116

Chapter 6. Experiments. 119

6.1 SCAAT Filter Configuration 119

6.1.1 The HiBall Filter 120

6.1.2 The Beacon Filters 120

6.1.3 Process Noise 121

6.1.4 Initial Error Covariance 122

6.1.5 Measurement Function. 123

6.1.6 Measurement Noise 124

6.2 Simulation Results 126

6.2.1 EKF Parameters 126

6.2.2 Error Nomenclature 130

6.2.3 EKF and Collinearity Tracking 131

6.2.4 EKF Beacon Autocalibration. 142

6.2.5 Simultaneous Observations 144

6.2.6 Moving-Window Observations 147

6.2.7 Cold Start 148

6.2.8 Blocked Cameras 149

6.2.9 Low Data Rates 149

6.2.10 Low Dynamics 152

6.2.11 Single Scalar Measurements 153

Chapter 7. Future Work 155

7.1 An Improved Dynamic Model 155

7.2 Model Parameter Sensitivity. 155

7.3 An Adaptive or Multiple-Model Filter 156

7.4 A Better Estimator 157

7.5 Source and Sensor Ordering 158

7.6 Solving a System of Simultaneous Equations 158

7.7 Other Tracking Implementations 159

APPENDIX A. THE SIMULTANEITY ASSUMPTION 161

A.1 An Example 2D Tracker 161

A.2 The Target in Motion. 162

A.3 Further Illustrations 164

APPENDIX B. THE KALMAN FILTER 167

B.1 The Discrete Kalman Filter 167

B.1.1 The Process to be Estimated 167

B.1.2 The Computational Origins of the Filter. 168

B.1.3 The Probabilistic Origins of the Filter. 170

B.1.4 The Discrete Kalman Filter Algorithm 170

B.1.5 Filter Parameters and Tuning 172

B.2 The Extended Kalman Filter (EKF). 174

B.2.1 The Process to be Estimated 174

B.2.2 The Computational Origins of the Filter. 175

B.3 Example: Estimating a Random Constant 179

B.3.1 The Process Model 179

B.3.2 The Filter Equations and Parameters 180

B.3.3 The Simulations. 180

APPENDIX C. ACTUAL PROBABILITY DENSITIES 184

APPENDIX D. THE UNC HIBALL TRACKER 191

D.1 The Ceiling. 192

D.2 The Hiball 192

APPENDIX E. THE SIMULATION ENVIRONMENT 194

E.1 User Motion Data Sets 194

E.2 Beacon Observations 194

E.3 EKF and Collinearity Implementations 195

E.3.1 SCAAT and MCAAT 196

E.3.2 Collinearity 196

E.3.3 Moving-Window Observations 196

E.4 EKF Parameters 197

E.5 Assumptions 198

REFERENCES 201

LIST OF TABLES

Table 2.1: The practical timing characteristics of a few existing tracking systems. 54

Table 2.2: Some example technologies and SCAAT observations. 63

Table 4.1: Output, correlation, and spectral density units 74

Table 4.2: Variance units for the elements 78

Table 5.1: Upper bound on EKF floating-point operations. 113

Table 5.2: Added floating-point operations with autocalibration. 115

Table 6.4: EKF parameter sets for perfect beacons. 127

Table 6.5: EKF parameter sets for 1.7 mm beacon error without autocalibration. 127

Table 6.6: EKF parameter sets for 1.7 mm beacon error with autocalibration. 128

Table 6.7: SCAAT parameter sets for low sampling rates. 129

Table 6.8: SCAAT parameters for “still” target dynamics. 130

Table B.1: Discrete Kalman filter time update equations. 171

Table B.2: Discrete Kalman filter measurement update equations. 172

Table B.3: Extended Kalman filter time update equations. 177

Table B.4: Extended Kalman filter measurement update equations. 177

Table E.1: Characteristics of the “typical” data set. 195

LIST OF FIGURES

Figure 1.1: Sensors vs. sensing units. 36

Figure 1.2: A simple estimator example. 37

Figure 1.3: SCAAT and constraints on a system of simultaneous equations. 38

Figure 1.4: Camera position and orientation from four scene points. 39

Figure 1.5: Camera position and orientation from two scene points. 40

Figure 1.6: Camera position and orientation from one scene point. 41

Figure 1.7: Putting the pieces together. 42

Figure 2.1: An ideal filter. 45

Figure 2.2: State versus parameter estimation. 49

Figure 2.3: Timing diagram for a Polhemus Fastrak magnetic tracker. 51

Figure 2.4: Timing diagram for a (hypothetical) SCAAT magnetic tracker. 52

Figure 2.5: The simultaneity assumption. 53

Figure 2.6: Error caused by the simultaneity assumption. 55

Figure 2.7: Timing diagram for a hypothetical conventional hybrid tracking system. . . 57

Figure 2.8: Timing diagram for a SCAAT inertial-acoustic hybrid tracking system. . . . 58

Figure 2.9: Intrinsic parameters. 59

Figure 2.10: Extrinsic parameters. 60

Figure 2.11: Autocalibration and attribution of measurement error. 62

Figure 3.1: The circumstances surrounding the bearings-only tracking problem. 64

Figure 3.2: A timing diagram for a conventional Kalman filter. 66

Figure 3.3: A timing diagram for a Kalman filter using the sequential updates method. 66

Figure 3.4: A timing diagram for a Kalman filter using the SCAAT method. 67

Figure 3.5: A timing diagram for an iterated Kalman filter. 68

Figure 4.1: Continuous-time position-velocity dynamic model. 74

Figure 4.2: Geometric view of state change during time update step.. 85

Figure 4.3: Geometric view of error covariance change during time update step. 85

Figure 4.4: Measurement prediction from the state and associated parameters. 86

Figure 4.5: The measurement Jacobian 87

Figure 4.6: Project state-space uncertainty into measurement space. 89

Figure 4.7: Completing the denominator of the Kalman gain. 90

Figure 4.8: The measurement residual. 93

Figure 4.9: Geometric view of state change after measurement correction. 95

Figure 4.10: Geometric view of error covariance change after measurement update. . . 95

Figure 4.11: Complete sequence of filter state coordinate frame transitions. 97

Figure 4.12: The revised tracking algorithm for autocalibration. 102

Figure 4.13: A timing diagram for autocalibration with d devices.. 103

Figure 5.1: Floating-point operation comparison: Collinearity vs. EKF. 118

Figure 6.1: Measurement signal strength. 125

Figure 6.2: Collinearity vs. EKF for perfect beacons 132

Figure 6.3: Collinearity vs. EKF without autocalibration 133

Figure 6.4: EKF with autocalibration 134

Figure 6.6: Summary—Collinearity vs. SCAAT with autocalibration. 135

Figure 6.5: Collinearity vs. SCAAT with autocalibration 136

Figure 6.7: Error spectra for perfect beacons 138

Figure 6.8: Error spectra without autocalibration 139

Figure 6.9: SCAAT error spectra. 140

Figure 6.10: Peaks in the error spectrum. 141

Figure 6.11: Final beacon error for EKF runs with autocalibration 143

Figure 6.12: Final beacon error for SCAAT with autocalibration. 143

Figure 6.13: Simultaneous observations, perfect beacons. 145

Figure 6.14: Simultaneous observations with autocalibration. 146

Figure 6.15: Autocalibration with truly simultaneous observations 147

Figure 6.16: Cold start 150

Figure 6.17: Blocked HiBall cameras 150

Figure 6.18: Error spectra for SCAAT with low sampling rates 151

Figure 6.19: Collinearity vs. SCAAT with autocalibration (“still” data set). 152

Figure 6.20: Error spectra for the “still” data set. 154

Figure A.1: A simple 2D inside-looking-out optoelectronic tracker. 161

Figure A.2: A close up of the 2D target platform and the two 1D pinhole cameras. . . 162

Figure A.3: Sample track. 163

Figure A.4: Estimated position versus measurement delay. 165

Figure A.5: A parametric version of figure A.4. 165

Figure A.6: Error caused by the simultaneity assumption. 166

Figure B.1: The ongoing discrete Kalman filter cycle. 171

Figure B.2: The complete Kalman filter algorithm. 173

Figure B.3: The complete extended Kalman filter operation. 178

Figure B.4: The first simulation:.. 182

Figure B.5: Error covariance, first simulation. 182

Figure B.6: Second simulation: 183

Figure B.7: Third simulation: 183

Figure C.1: Actual state error density before and after time update step. 185

Figure C.2: Multiple viewpoints of densities from figure C.1. 186

Figure C.3: Actual measurement Jacobian H. 187

Figure C.4: Actual measurement-space error densities. 188

Figure C.5: Actual state error density before and after complete filter update. 189

Figure C.6: Multiple viewpoints of densities from figure C.5. 190

Figure D.1: An outward-looking optoelectronic tracking system. 191

Figure D.2: The original UNC optoelectronic ceiling tracker. 192

Figure D.3: The new HiBall camera cluster. 193

Figure E.1: Moving vs. fixed-window observations. 197

Figure E.2: Primary and secondary HiBall views. 199

LIST OF ABBREVIATIONS

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
6D	six-dimensional
DOF	degree of freedom
EKF	extended Kalman filter
FP	floating-point
GPS	Global Positioning System
Hz	Hertz
HMD	head-mounted display
IKF	iterated Kalman filter
KF	Kalman filter
LED	light emitting diode
ms	millisecond
MCAAT	multiple-constraints-at-a-time
PV	position-velocity
RMS	root-mean-square
SCAAT	single-constraint-at-a-time
UNC	University of North Carolina
VE	virtual environment

GENERAL NOTATION

x	A scalar variable (lower case, italic).
$\vec{x}$	A general column vector (lower case, italic, arrow)—indexed as $\vec{x}[r]$.
$\hat{x}$	A vector of estimated elements (lower case, italic, hat).
$\widehat{x}$	An augmented version of a particular $\hat{x}$ vector, i.e. a version with additional elements (wide hat).
A	A matrix (upper case, italic)—indexed as $A[r, c]$.
A^{-1}	A matrix inverse.
$\beta(t)$	A matrix/vector at time t.
$\beta^{-}(t)$	A matrix/vector prediction prior to a correction at time t (super minus). This is also commonly referred to as the *a priori* value of the matrix/vector.
β^{T}	A matrix/vector transpose (super T).
β_{α}	A matrix/vector/scalar identifier (subscript). This is used to discriminate between multiple similar matrices/vectors/scalars.
$E\{\bullet\}$	Denotes mathematical expectation, i.e. the probabilistic value of •.
$F\{\bullet\}$	Denotes the Fourier transform of •.
$F^{-1}\{\bullet\}$	Denotes the inverse Fourier transform of •.
$f(\bullet)\|_{\text{cond}}$	The function $f(\bullet)$ evaluated under the given conditions (cond).

LIST OF SYMBOLS

$A(t_k)$	The $n \times n$ state-transition matrix at discrete time step t_k. This relates the state vector $\vec{x}(t_k)$ to the state vector $\vec{x}(t_{k+1})$ in the absence of noise. First used in section 2.1.3 on page 47.
$A(\delta t)$	The $n \times n$ state-transition matrix as a function of the discrete time interval δt. This relates the state vector $\vec{x}(t)$ to the state vector $\vec{x}(t + \delta t)$ in the absence of noise. First used in section 4.2.2 on page 75.
$\widehat{A}(\delta t)$	The augmented state-transition matrix as a function of the discrete time interval δt. First used in section 4.4.3 on page 100.
A_k	State transition matrix at discrete time t_k (upper case, italic, subscript). This abbreviation for $A(t_k)$, which is common in the Kalman filter literature, is only used in appendix B. It is used to simplify the notation.
B	The $n \times l$ control matrix which relates the l-dimensional control vector $\vec{u}$ to the n-dimensional state $\vec{x}$. First used in section B.1.1 on page 167.
$\vec{b}(t)$	Source device parameter vector at time t. First used in section 4.2.4 on page 78.
C	The minimal number of independent simultaneous constraints necessary to uniquely determine a solution for a given estimation problem. First used in “A Single Constraint” on page 37.
c	Scalar camera baseline distance in [meters]. Used in appendix A.
$\vec{c}(t)$	Sensor device parameter vector at time t. First used in section 4.2.4 on page 78.

d_b	Distance from a beacon to a viewing camera. Units are [meters]. Used in section 6.1.6 on page 124.
FP(N)	The number of floating-point operations required to compute an estimate given a set of constraints of size N. First used in section 5.2.1 on page 113.
$\vec{f}(t)$	The n-dimensional function representing the actual (unknown) target dynamics at time t. Used in section 5.1.5 on page 111.
$\hat{f}(t)$	The n-dimensional function representing an *estimate* of the actual (unknown) target dynamics at time t. Used in section 5.1.5 on page 111.
$H(t_k)$	The $n \times m$ measurement matrix at discrete time step t_k. This relates the n-dimensional process state vector $\vec{x}(t_k)$ to the m-dimensional measurement vector $\vec{z}(t_k)$. First used in section 2.1.2 on page 46.
$\vec{h}_\sigma(\bullet)$	The m-dimensional measurement function for a sensor of type σ. First used in section 4.2.4 on page 78.
I	The identity matrix. First used in equation (4.20) on page 94.
K	The $n \times m$ Kalman gain matrix. First used in section 4.3.3 on page 87.
K_k	The $n \times m$ Kalman gain matrix at discrete time step t_k. This abbreviation for $K(t_k)$, which is common in the Kalman filter literature, is only used in appendix B. It is used to simplify the notation
M	The *observability matrix*. Used in section 5.1.2 on page 107.
m	The scalar dimension of the filter's measurement vector. First used in section 5.1.2 on page 107.

m_σ	The scalar dimension of the filter's measurement vector for a sensor of type σ. First used in section 4.2.4 on page 78.
N	The number of constraints actually employed to estimate a solution for a given estimation problem. First used in "A Single Constraint" on page 37.
N_{ind}	The number of *independent* constraints that can be formed from the N constraints actually employed to estimate a solution. First used in "A Single Constraint" on page 37.
n	The scalar dimension of the filter's state vector. First used in section 2.2.4 on page 52.
n_π	The scalar dimension of the device parameter vector $\vec{\pi}$ for the device π. First used in section 4.4.2 on page 100.
$\vec{o}_k$	Bearings-only observer position at time k. Used in section 3.1 on page 64.
$P(t)$	The $n \times n$ *error covariance matrix* at time t, where n is the dimension of the state vector. The units are described in the text. First used in section 4.2.7 on page 83.
$\widehat{P}(t)$	The augmented *error covariance matrix* at time t. The units are described in the text. First used in section 4.4.3 on page 100.
$P_\pi(t)$	The $n_\pi \times n_\pi$ *device* error covariance matrix at time t for the device π, where n_π is the dimension of the device state vector. The units are intentionally unspecified as they depend on the device and the particular parameters. First used in section 4.4.2 on page 100.
$P_b(t)$	The $n_b \times n_b$ *source* error covariance matrix at time t. See $P_\pi(t)$. First used in section 4.4.3 on page 100.

$P_c(t)$	The $n_c \times n_c$ *sensor* error covariance matrix at time t. See $P_\pi(t)$. First used in section 4.4.3 on page 100.
$\vec{p}$	Known scene point. Units in [meters]. First used in "A More Concrete Example" on page 39.
Q	The $n \times n$ continuous time *process noise covariance matrix*, where n is the dimension of the state vector. The units depend on the specific context and are provided where appropriate in the text. First used in section 4.2.3 on page 76.
$Q(\delta t)$	The $n \times n$ sampled *process noise covariance matrix* with sample time δt, where n is the dimension of the state vector. The units depend on the specific context and are provided where appropriate in the text. First used in section 4.2.3 on page 76.
$\widehat{Q}(\delta t)$	The augmented *process noise covariance matrix* with sample time δt. First used in section 4.4.3 on page 100.
$Q_\pi(\delta t)$	The $n_\pi \times n_\pi$ sampled *device* process noise covariance matrix with sample time δt for the device π, where n_π is the dimension of the device state vector. The units are intentionally unspecified as they depend on the device and the particular parameters. First used in section 4.4.2 on page 100.
$Q_b(\delta t)$	The $n_b \times n_b$ sampled *source* process noise covariance matrix with sample time δt. See $Q_\pi(\delta t)$. First used in section 4.4.3 on page 100.
$Q_c(\delta t)$	The $n_c \times n_c$ sampled *sensor* process noise covariance matrix with sample time δt. See $Q_\pi(\delta t)$. First used in section 4.4.3 on page 100.

$R_\sigma(t)$	The $m_\sigma \times m_\sigma$ *measurement noise covariance matrix*, where m_σ is the dimension of the measurement vector. The units depend on the specific context. First used in section 4.2.5 on page 81.
$s(t)$	Time-varying scalar true signal. Used in figure 2.1 on page 45.
t_k	The k^{th} time step in a regular discrete sequence. First used in section 2.1.2 on page 46.
u, v	Scalar positions along the U and V axes of a camera's image plane. The units are [meters]. First used in “A More Concrete Example” on page 39.
V	The Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to $\vec{v}_k$. Used in appendix B.
W	The Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to $\vec{\omega}_k$. Used in appendix B.
x	Scalar position along the X axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\dot{x}$	Scalar velocity in the direction of the X axis of the respective coordinate system. The units are [meters/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
x_b	Scalar position of a beacon along the X axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 6.1.1 on page 120.

$\vec{x}_\pi$	A source or sensor device parameter vector. The units are intentionally unspecified as they depend on the device and the particular parameters. First used in section 4.4.2 on page 100.
$\hat{x}_\pi(t)$	The n_π-dimensional device state vector *estimate* at time t, for the device π. The units are intentionally unspecified as they depend on the device and the particular parameters. First used in section 4.4.2 on page 100.
$\hat{x}_b(t)$	The n_b-dimensional *source* state vector *estimate* at time t. See $\hat{x}_\pi(t)$. First used in section 4.4.3 on page 100.
$\hat{x}_c(t)$	The n_b-dimensional *sensor* state vector *estimate* at time t. See $\hat{x}_\pi(t)$. First used in section 4.4.3 on page 100.
$\vec{x}(t_k)$	The *n*-dimensional tracker process state vector at discrete time step t_k. The units are described in the text. First used in section 2.1.2 on page 46.
$\vec{x}(t)$	The *n*-dimensional process state vector at time t. The units are described in the text. First used in section 4.2.2 on page 75.
$\hat{x}(t)$	The *n*-dimensional process state vector *estimate* at time t. The units are described in the text. First used in section 4.2.7 on page 83.
$\widehat{x}(t)$	The *augmented* process state vector *estimate* at time t. The units are described in the text. First used in section 4.4.3 on page 100.
$\hat{x}(t_{k+1} \mid t_k)$	The estimated *n*-dimensional process state vector at discrete time step t_{k+1} given only measurements and estimates through time step t_k. The units are intentionally unspecified. First used in section 2.1.3 on page 47.

$\hat{x}(t_k \mid t_k)$	The estimated *n*-dimensional process state vector at discrete time step t_k given all measurements and estimates up through that time. The units are intentionally unspecified. First used in section 2.1.3 on page 47.
$\vec{x}_k$	State vector at discrete time t_k. This abbreviation for , which is common in the Kalman filter literature, is only used in appendix B. It is used to simplify the notation.
y	Scalar position along the *Y* axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\dot{y}$	Scalar velocity in the direction of the *Y* axis of the respective coordinate system. The units are [meters/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
y_b	Scalar position of a beacon along the *Y* axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 6.1.1 on page 120.
z	Scalar position along the *Z* axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\dot{z}$	Scalar velocity in the direction of the *Z* axis of the respective coordinate system. The units are [meters/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.

z_b	Scalar position of a beacon along the Z axis of the respective coordinate system. The units are [meters]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 6.1.1 on page 120.
$\vec{z}$	Sensor measurement or observation vector. First used in “A More Concrete Example” on page 39.
$\vec{z}(t_k)$	The *m*-dimensional measurement vector at discrete time step t_k. First used in section 2.1.2 on page 46.
$\hat{z}_\sigma(t)$	The *m*-dimensional measurement vector *estimate* for a sensor of type σ at time t. First used in section 4.2.4 on page 78.
$\vec{z}_\sigma(t)$	The actual *m*-dimensional measurement vector for a sensor of type σ at time t. First used in section 4.2.5 on page 81.
†	Indicates a *second* experimental run through a particular data set, inheriting the autocalibrated beacon position estimates from the *first* run. First used in section 6.2.3 on page 131 and figure 6.4 on page 134.
$\Re^n$	n-dimensional real number space. Used in appendix B.

GREEK SYMBOLS

$\vec{\alpha}$	The four-element external orientation quaternion with the elements $\alpha_w, \alpha_x, \alpha_y, \alpha_z$. First used in section 4.2.2 on page 75.
α_b	Angle between the primary axis of a beacon and the primary axis of a viewing camera. Units are [radians]. Used in section 6.1.6 on page 124.
α_w	The scalar magnitude component of the external orientation quaternion $\vec{\alpha}$. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
α_x	The scalar *X* direction component of the external orientation quaternion $\vec{\alpha}$. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
α_y	The scalar *Y* direction component of the external orientation quaternion $\vec{\alpha}$. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
α_z	The scalar *Z* direction component of the external orientation quaternion $\vec{\alpha}$. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
α_1	The scalar lower process noise stability bound constant. Used in section 5.1.3 on page 108.
α_2	The scalar upper process noise stability bound constant. Used in section 5.1.3 on page 108.
β_1	The scalar lower measurement noise stability bound constant. Used in section 5.1.3 on page 108.

β_2	The scalar upper measurement noise stability bound constant. Used in section 5.1.3 on page 108.
Γ	The measurement noise stability matrix. Used in section 5.1.3 on page 108.
δt	Inter sample time interval. Units are [seconds]. First used in section 4.2.1 on page 73.
$\Delta\theta$	Scalar incremental rotation about the *Y* axis of the respective coordinate system. Units are [radians]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\Delta\phi$	Scalar incremental rotation about the *X* axis of the respective coordinate system. Units are [radians]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\Delta\psi$	Scalar incremental rotation about the *Z* axis of the respective coordinate system. Units are [radians]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\overrightarrow{\Delta z}$	The *m*-dimensional measurement residual vector. First used in section 4.3.4 on page 92.
$\Delta\vec{f}(t)$	The *n*-dimensional difference vector between the actual unknown signal $\vec{f}(t)$ and an estimate of it $\hat{f}(t)$. Used in section 5.1.5 on page 111.
ε	Small time increment. Units are [seconds]. First used

$\vec{\eta}$	The process noise source vector. The elements are the magnitudes, i.e. spectral densities, of zero-mean white noise sources. The units depend on the specific context and are provided where appropriate in the text. First used in section 4.2.1 on page 73.
$\dot{\theta}$	Scalar angular velocity about the Y axis of the respective coordinate system. Units are [radians/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
λ	A unitless identifier for the single lateral (x and y) translation element of $\vec{\eta}$, the process noise source vector. First used in section 6.1.3 on page 121.
$\vec{\mu}[i]$	The autocorrelation vector for the noise sources $\vec{\eta}[i]$ shown in figure 4.1 on page 74. The formal definition is given in equation (4.9) on page 77. The units depend on the element of $\vec{\eta}$ of interest and can be obtained from table 4.1 on page 74 using the relationship defined by equation (4.9).
$\nu(t)$	Time-varying scalar noise signal. The units are intentionally unspecified. Used in figure 2.1 on page 45.
$\vec{\nu}(t_k)$	The m-dimensional measurement noise vector at discrete time step t_k. First used in section 2.1.2 on page 46.
$\vec{\nu}_{\sigma}(t)$	The m_{σ}-dimensional measurement noise vector at time t. First used in equation (4.10) on page 81 and defined in section 4.2.5 on page 81.
$\hat{\xi}(t)$	The state error vector. The units are those of the state vector and are described in the text. First used in section 4.2.7 on page 83.
ξ_b	Initial uncertainty (variance) in the beacon position parameters. Units are $[\text{meters}]^2$. First used in section 6.1.4 on page 122.

ξ_c	Camera measurement uncertainty (variance). Units are $[\text{meters}]^2$. First used in section 6.1.6 on page 124.
ξ_0	The standard deviation of the measurement noise for a beacon sighting with $\alpha_b = 0$ degrees and $d_b = 1$ meter. Units are $[\text{meters}]^2$. Used in section 6.1.6 on page 124.
$\Pi(N)$	The complete simulation parameter vector for use with N constraints. First used in section 6.2.1 on page 126, defined in section E.4 on page 197.
π	Device parameter vector identifier. First used in section 4.4.2 on page 100.
ρ_e	The estimate rate. Units are [estimates/second]. First used in section 2.2.3 on page 50.
σ	Sensor type (general identifier). First used in section 4.2.4 on page 78.
τ_m	The amount of time needed to perform one sensor measurement. Units are [seconds]. First used in section 2.2.3 on page 50.
$\tau_c(N)$	The amount of time needed to compute an estimate as a function of N constraints. Units are [seconds]. First used in section 2.2.3 on page 50.
τ_e	The estimate latency, i.e. the total amount of time needed to collect constraints and to compute an estimate. Units are [seconds]. First used in section 2.2.3 on page 50.
$\dot{\phi}$	Scalar angular velocity about the X axis of the respective coordinate system. Units are [radians/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.

Ξ	The Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to . Used in appendix B.
χ	A single constraint. First used in “A Single Constraint” on page 37.
ψ	Scalar angular velocity about the Z axis of the respective coordinate system. Units are [radians/second]. Also used as a unitless identifier for the corresponding element of vectors or matrices. First used in section 4.2.2 on page 75.
$\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon)$	The correlation kernel for the random for the random process vector $\vec{\eta}$, where ε is the time correlation interval: $\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon) = E\{\vec{\eta}(t)\vec{\eta}^T(t+\varepsilon)\}$. (See [Maybeck70, p. 137].) First used in section 4.2.1 on page 73. The units depend on the element of $\vec{\eta}$ of interest and are described in table 4.1 on page 74.
$\overline{\Psi}_{\vec{\eta}\vec{\eta}}(\omega)$	The spectral density of the random process vector $\vec{\eta}$, where ω is the complex frequency. This is the Fourier transform of the correlation kernel $\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon)$: $\overline{\Psi}_{\vec{\eta}\vec{\eta}}(\omega) = F\{\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon)\}$. (See [Maybeck70, p. 140].) First used in section 4.2.1 on page 73. The units depend on the element of $\vec{\eta}$ of interest and are described in table 4.1 on page 74.
Ω	The maximum target motion bandwidth in $[\text{cycles/second}]$. First used in “Sufficient Constraints” on page 109.
ω	When used as a scalar (not a vector) this represents a complex frequency. First used with $\overline{\Psi}_{\vec{\eta}\vec{\eta}}(\omega)$ in table 4.1 on page 74 in particular.
$\vec{\omega}(t_k)$	The n-dimensional process noise vector at discrete time step t_k. First used in section 2.1.3 on page 47.

$\vec{\omega}_k$	The n-dimensional process noise vector at discrete time step t_k. This abbreviation for $\vec{\omega}(t_k)$, which is common in the Kalman filter literature, is only used in appendix B. It is used to simplify the notation.
$\vec{\omega}(t)$	The n-dimensional process noise vector at time t. First used in section 4.2.2 on page 75.

Chapter 1. Introduction

1.1 Narrative, Names, and Nomenclature

Credit Where Credit is Due

The idea for *one-LED-at-a-time* tracking for *virtual environments* was originally conceived of by my academic advisor Dr. Gary Bishop, and has been the subject of our joint research since August of 1994. Our work has led to the method being implemented in the new UNC "HiBall" optoelectronic tracking system, and it led me to this dissertation. As such when I refer to "we" or "our" I am referring to collaboration with Dr. Gary Bishop.

What's in a Name?

Relatively early on, the name *One-Step-at-a-Time* (OSAAT) was suggested by Dr. Anselmo Lastra in discussions about how the approach might generalize to other tracking or navigation systems, e.g. systems that employ GPS receivers, magnetic devices or inertial devices. Later reviewer comments from a paper submission led me to believe that as part of an overall improved explanation of our work, the name needed to be more descriptive. While bouncing around ideas, the phrase "incremental tracking" was suggested by Vernon Chi, while "incomplete information" was suggested by Dr. Fred Brooks. I personally decided that *constraint* was a better word than *step* because the word *step* is easily (understandably) confused with a Kalman filter *time step*, and more importantly because the word *constraint* better conveys the notion of information content which is central to this thesis. Hence the phrase "one-step-at-a-time" became "one-*constraint*-at-a-time" or OCAAT. Finally, Mark Mine (a fellow graduate student and long-time friend) suggested changing "one-constraint-at-a-time" to "*single*-constraint-at-a-time", thus transforming the lackluster OCAAT to the more colorful SCAAT or "scat".

The “scat” pronunciation fits well: like scat singing where meaningless syllables are sung to a tune, in SCAAT tracking incomplete information is woven into a trajectory. And so we are left with the title of this dissertation, *SCAAT: Incremental Tracking with Incomplete Information*. The relevance of these phrases, and the title of this dissertation, should become more clear in the remainder of this introduction.

Defining the Nomenclature

Throughout this dissertation I refer to “sources”, “sensors”, or “devices”, and it’s particularly important to understand what I mean by these words not only to avoid confusion, but to understand the basic SCAAT approach. I use the words “source” or “sensor” to refer to the single (minimal) electrical or mechanical component used to excite or sense a particular physical medium. Using this nomenclature, some example ***sources*** include a single-axis electromagnetic dipole, an infrared LED, and a single GPS satellite. Some example ***sensors*** include a single-axis electromagnetic coil, a single camera, and a GPS receiver. I will use the word “device” to refer to either a source or sensor. On the other hand, when I want to refer to a mechanical fixture that incorporates *multiple* sources or sensors, I will use phrases such as “source unit” or “sensing unit”.

For example, the Polhemus magnetic tracker [Raab79] employs a fixed source unit which contains a three-axis electromagnetic dipole. Because each of the three axes can be individually excited, I would say that each ***source unit*** contains three distinct *sources*. The Polhemus also employs target-mounted sensing units, each of which contains another three-axis electromagnetic device. Because the induced voltage corresponding to each axis of the sensing unit can be individually measured, I would say that each single ***sensing unit*** contains three distinct *sensors*. Figure 1.1 depicts this distinction for such a magnetic sensing unit, and for an inertia sensing unit.

Finally, I also frequently interchange the words “measurement” and “observation”, depending on the context. In either case I am generally referring to the combined act of source excitation (when necessary) and sensor reading. For systems that employs active sources, e.g. optoelectronic or magnetic tracking systems, I generally use ***measurement*** or

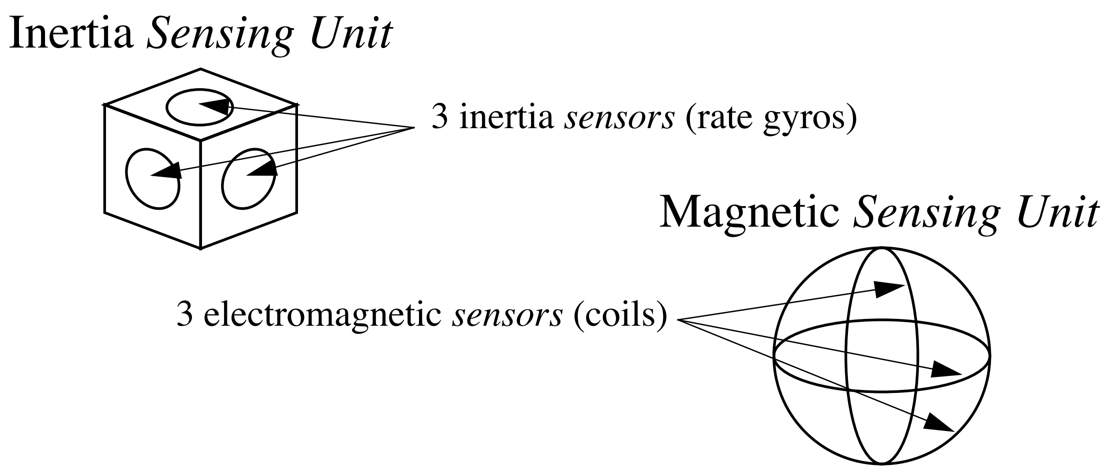

Inertia *Sensing Unit*
3 inertia *sensors* (rate gyros)
Magnetic *Sensing Unit*
3 electromagnetic *sensors* (coils)

Figure 1.1: Sensors vs. sensing units. This diagram shows two common sensing units that are frequently (loosely) referred to as "sensors". In this dissertation I make a distinction between *sensors* and *sensor units*.

observation to refer to a combined excitation/sensing act: a single source excitation followed by the corresponding sensor measurement. For passive systems, e.g. a purely inertial system, I use these words to refer to the act of reading a single sensor.

This nomenclature should be contrasted with the frequent practice of loosely referring to multi-sensor units as "sensors" and multi-source units as "sources". This practice is a result of the historical tendency to group the multiple source excitations or sensor measurements corresponding to a complete unit, into what is loosely termed "a measurement". While the distinction may at first seem minor, it is central to the understanding of the SCAAT approach to tracking.

1.2 Observability and Incomplete Information

The idea that one might build a tracking system that generates a new estimate with each individual sensor measurement or *observation* is a very interesting one. After all, individual observations usually provide only partial information about a target's complete state, i.e. they are "incomplete" observations. For either sensing unit in figure 1.1, only limited information is obtained from measurements of any single sensor within the corresponding sensing unit. In terms of control theory, systems that attempt to operate with incomplete measurements are characterized as *unobservable* because the state cannot be observed (determined) from the measurements.

A Simple Estimator

Figure 1.2 depicts a simple example of an estimator that produces estimates of $\hat{x}$ and $\hat{y}$ by observing *noisy* measurements of the *true* parameters x and y over time. This example is particularly simple because there is a direct one-to-one correspondence between the state $[x, y]^T$ and the noisy measurements.

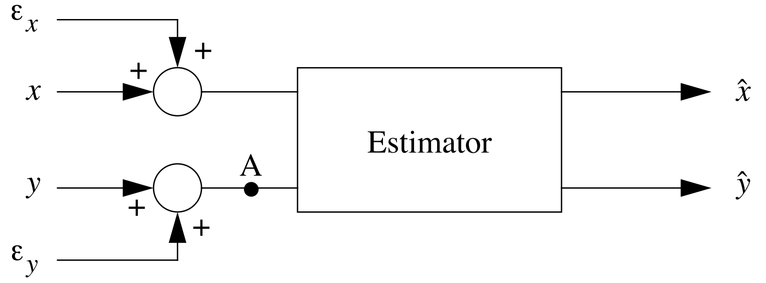


Figure 1.2: A simple estimator example. The system produces estimates $\hat{x}$ and $\hat{y}$ by observing *noisy* measurements of the *true* parameters x and y over time (noise represented by ε_x and ε_y respectively).

Now consider the same system with the connection from the (noisy) measurement of y severed at point A. In this case, the estimator would continue to estimate both x and y, but would do so with only measurements of x. Because y is no longer being observed, y and $\hat{y}$ are free to vary independently of each other. Such a system is unobservable because one can not determine the state $[x, y]^T$ from measurements of x alone.

A Single Constraint

The notion of observability can also be described in terms of *constraints* on the unknown parameters of the system being estimated, e.g. constraints on the unknown elements of the system state. In the purest mathematical sense, a single constraint consists of a single scalar equation that describes a known relationship between the unknown elements. However, sometimes multiple equations are collectively described as providing a single "constraint" in a more general sense. Such is the case in geometry where people often discuss *geometric* constraints involving points, lines, planes, etc. For example a 3D point might be constrained to lie on the surface of a sphere, or a 3D line might be constrained to intersect a 2D plane at a particular point. Furthermore, with respect to tracking systems, some implementations allow for simultaneous multi-dimensional sensor measurements. Thus I allow for, or even suggest (under certain circumstances) consideration of a multi-dimensional constraint. Depending on the circumstances, discussed in “The Measurement Model” on page 78, one might adopt either the single or the multi-dimensional notion of a single constraint. As long as one remains consistent, discussion such as that in the next section—and indeed throughout this dissertation—remains valid. In any case, the novelty of the SCAAT approach remains.

Constraints and Observability

Given a particular system, and the corresponding set of unknowns that are to be estimated, let C be defined as the minimal number of independent simultaneous constraints (single or multi-dimensional) necessary to uniquely determine a solution, let N be the number actually used to generate a new estimate, and let N_{ind} be the number of *independent* constraints that can be formed from the N constraints. For any $N \geq N_{\text{ind}}$ constraints, if $N_{\text{ind}} = C$ the problem is *well constrained*, if $N_{\text{ind}} > C$ it is *over constrained*, and if $N_{\text{ind}} < C$ it is *under-constrained*. This is depicted in figure 1.3.

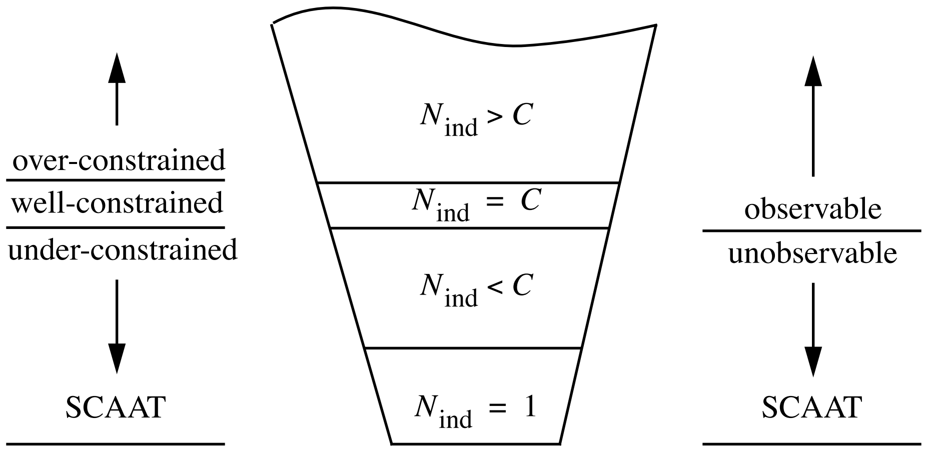


Figure 1.3: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{\text{ind}}$ and $N_{\text{ind}} \geq C$, i.e. to use enough measurements to well-constrain or even over-constrain the estimate. The SCAAT approach is to employ the smallest number of constraints available at any one time, generally $N = N_{\text{ind}} = 1$ constraint. From this viewpoint, each SCAAT estimate is severely under-constrained.

A More Concrete Example

Figures 1.4 through 1.6 depict a more concrete (and realistic) situation in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates $\vec{z}_i = [u, v]$ of 3D scene points $\vec{p}_i = [x, y, z]$, for $i = 0, 1, \ldots$. Note that the vector $\vec{z}_i$ represents the u, v image-plane coordinates of scene point i, while the scalar element z of $\vec{p}_i$ represents the distance of the scene point along the Z axis in figures 1.4 through 1.6. Given the internal camera parameters, a set of four known coplanar scene points $\vec{p}_0, \ldots, \vec{p}_3$ and the corresponding image coordinates $\vec{z}_0, \ldots, \vec{z}_3$, the camera position and orientation can be uniquely determined in closed-form [Ganapathy84]. In other words if $N = C = 4$ constraints (image points) are used to estimate the camera position and orientation, the system is completely observable. This situation is depicted in figure 1.4.

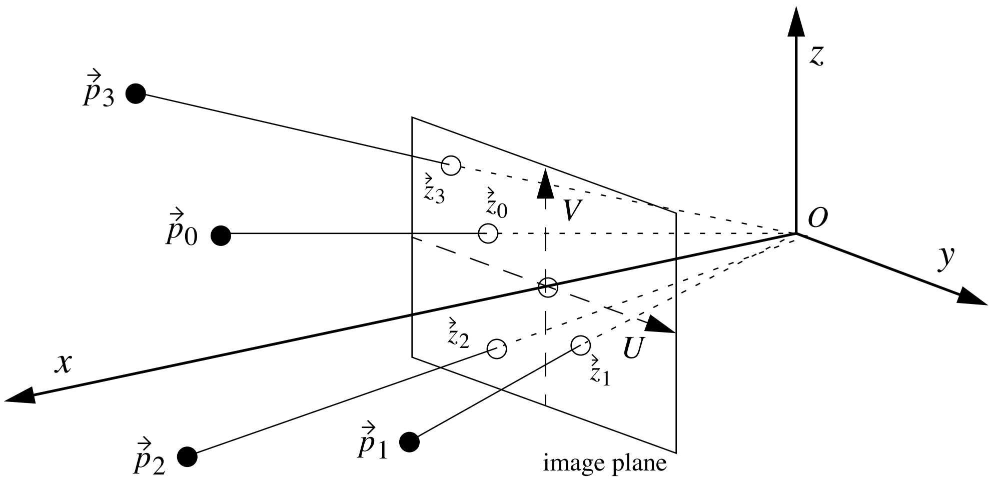
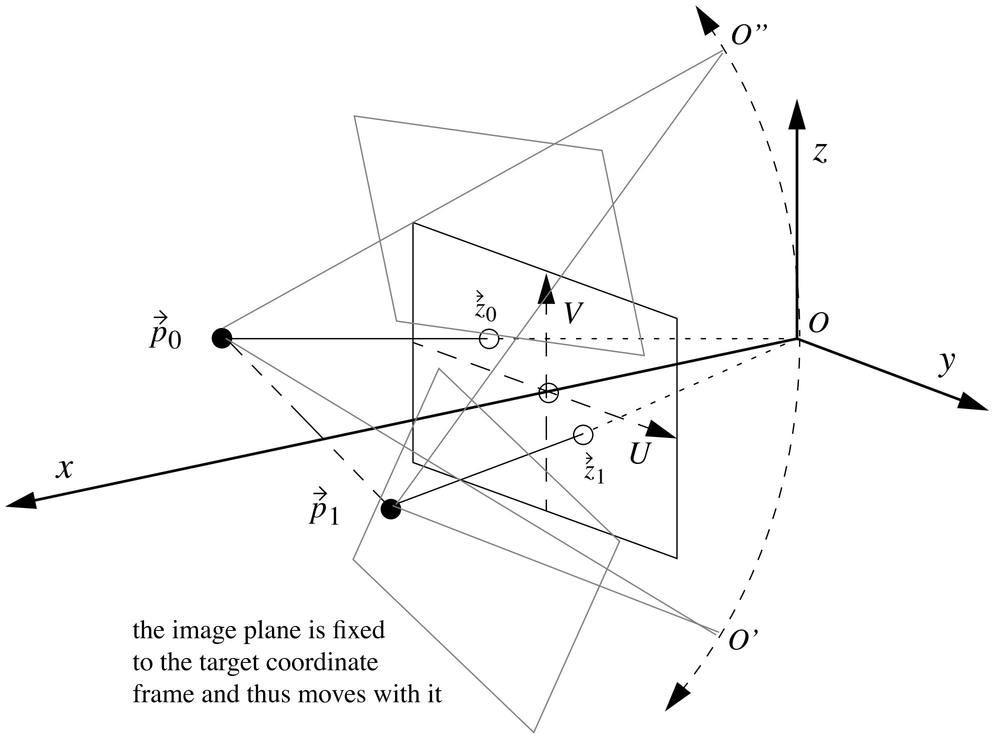


Figure 1.4: Camera position and orientation from <u>four</u> scene points. Given the internal camera parameters, $N = C = 4$ known (coplanar) scene points $\vec{p}_0, \ldots, \vec{p}_3$, and the corresponding images $\vec{z}_0, \ldots, \vec{z}_3$, the camera position and orientation can be uniquely determined in closed form. Such a system is completely observable.

On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 1$ or $N = 2$ points, there are infinite combinations of position and orientation that could result in the same camera images. (See figure 1.5 and figure 1.6.) Any single-camera system that attempts to compute a unique camera position and orientation using known coplanar scene points must employ observations of four or more such points or the state will be unobservable.



O''
z
$\vec{z}_0$
V
O
$\vec{p}_0$
y
x
U
$\vec{z}_1$
$\vec{p}_1$
the image plane is fixed to the target coordinate frame and thus moves with it
O'

Figure 1.5: Camera position and orientation from <u>two</u> scene points. The camera position and orientation is not observable. The camera origin (attached to the image plane) could be anywhere along the great circle around the segment $\vec{p}_0\vec{p}_1$, e.g. at O' or O''. As long as the camera was "properly" pointing at the segment, the resulting image would be the same.

In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices must be excited and/or sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes (discussed in section 2.2.1) they can not. The Polhemus magnetic tracker, for example, performs three *sequential* source excitations, each in conjunction with a complete sensor unit observation.

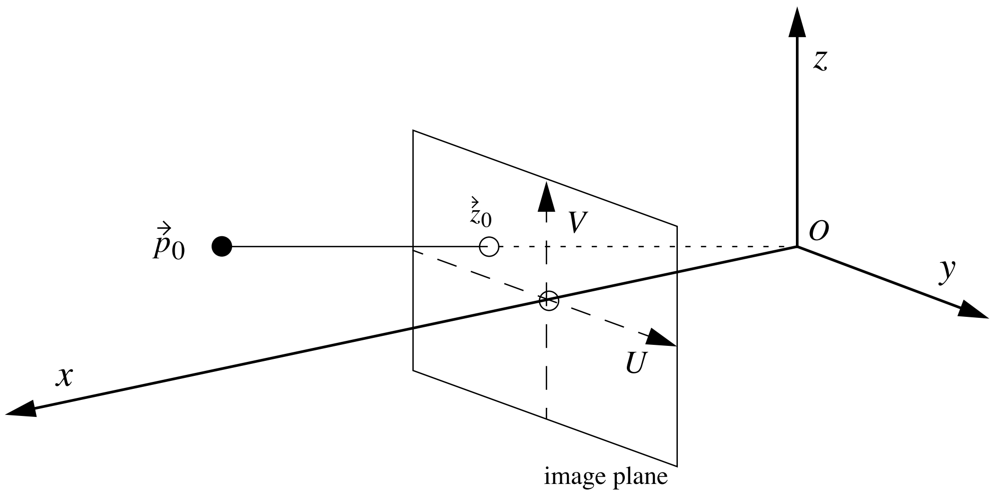


Figure 1.6: Camera position and orientation from one scene point. The camera position and orientation is clearly not observable. The camera origin could be *anywhere*. As long as the camera is "properly" pointing at $\vec{p}_0$, the resulting image would be the same.

Putting the Pieces Together

For a system that employs multiple "incomplete" measurements (each affording an under-constrained observation) the *measurement model* corresponding to any single incomplete measurement can be characterized as *locally unobservable*. Such a system must collectively incorporate a sufficient set of individual measurements such that the resulting overall system is *observable*.[1] The corresponding aggregate measurement model can then be characterized as *globally observable*. Figure 1.7 depicts this relationship in terms of a puzzle. Individually the puzzle pieces (the locally unobservable measurements) do not offer a complete picture, but when assembled together they do. To see the result, one might (somehow) collect and incorporate all of the pieces simultaneously. More likely, one would assemble the puzzle one piece at a time. Likewise, to form a globally observable system for tracking, one might simultaneously observe a sufficient set of sensors, or sequentially observe the individual sensors over time. In other words, global observability can be obtained over *space* or over *time*. The SCAAT method adopts the latter scheme, even in some cases where the former is possible.

1. If not completely observable, any unobservable states must be stable, i.e. convergent.

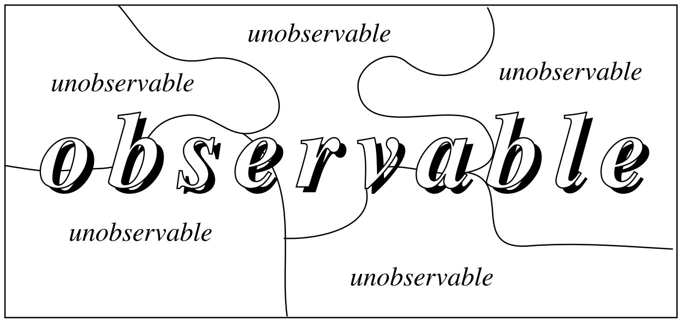


Figure 1.7: Putting the pieces together. A globally observable system can be formed from a collection of locally unobservable systems. The individual pieces offer only partial information, while the aggregate depicts a complete picture.

For more complete discussion about the characterization and detection of unobservability in control systems see [Brown92, Jacobs93, Maybeck70].

1.3 An Unusual Approach to Tracking

A key point of this dissertation is that even incomplete observations provide *some* information, and that the partial information can be used incrementally to improve an existing estimate of the solution. As long as individual measurements are incomplete in complementary ways, they can be used together over time to progressively improve an existing estimate. Rather than attempting to compute precise closed-form estimates of *points* in state space, we can instead push the current state estimate along the *track* indicated by the most recent (incomplete) observation. By carefully blending an ongoing sequence of incomplete observations in this way, one can generate an improved state estimate with each new constraint (observation)—guiding the sequence of estimates along the target track, using a single constraint at a time.

Some readers may recognize that this method for tracking is similar to methods explored in the late 1970’s and early 1980’s to solve what is often called the “bearings-only” tracking problem. The relationship to the bearings-only problem is discussed in section 3.1, but it’s worth mentioning here that the bearings-only circumstances essentially *dictated* the method: the only available 2D target data was a sequence of relatively infrequent noisy bearing measurements from a *single* 1D passive sensor. On the other hand, most modern tracking or navigation systems employ multiple (relatively fast) devices, so measurement data is more readily available. Despite these favorable circumstances, modern tracking systems are often forced to use a group of *sequentially* collected individual (incomplete) measurements to compute each estimate. This is sometimes dictated by the physical nature of the sensing medium, and sometimes by concerns for a cost-effective implementation. For these systems, and surprisingly even for systems that have truly simultaneous measurements available, I present (1) specific motivation for returning to a single observation per estimate strategy, and (2) a specific method for doing so.

1.4 Contribution (Thesis Statement)

As far as I know, no one has ever advocated the use of a Kalman filter to estimate the state of a globally-observable multiple-sensor system by sequentially incorporating only measurements of locally-unobservable single-sensor systems as described herein. I believe that the SCAAT method is not only attractive from a purely theoretical or mathematical standpoint, but that its use is justified by some very real and heretofore unidentified practical advantages.

Thesis Statement

A Kalman filter can be used to estimate a *globally-observable* process by sequentially incorporating only measurements of *locally-unobservable* processes. The use of a Kalman filter in such a manner offers several advantages: (1) a flexible framework for heterogeneous multisensor data fusion; (2) a unique opportunity to perform concurrent device *autocalibration*; and in a system that allows only sequential measurements, (3) significantly improved estimate rates and latencies; and (4) avoidance of the incorrect *simultaneity assumption*.

While I believe that the application of the method may be justified for a broader class of stochastic estimation problems, in this dissertation I have chosen to present the work in the context of 3D tracking for virtual environments, thus forming a concrete basis for explanation and analysis. Applications to other problems should be straight-forward.

Chapter 2. Motivation

The primary purpose of this chapter is to justify the use of the SCAAT method. A side-effect is to introduce some work by other researchers that is related to the subject in terms of motivation. (The primary discussion of related work is presented in chapter 3.) I begin in section 2.1 by motivating the use of a Kalman filter in general. In the remaining sections of the chapter, I assume the use of a Kalman filter and argue for the SCAAT method in particular. If necessary, see the beginning of chapter 4 for some brief discussion of the Kalman filter, and/or appendix B for a more complete introduction.

2.1 Why a Kalman Filter?

In this dissertation I claim that there are significant benefits to using a Kalman filter in what I call a SCAAT fashion. But why use a Kalman filter in the first place? There are four typical reasons why people employ Kalman filters in systems where a signal (often continuous) is to be estimated with a sequence of discrete measurements or observations: (1) filtering; (2) data fusion; (3) prediction; and (4) calibration. In this section I briefly examine each of these in the context of tracking or navigation. Once I've argued that a Kalman filter is a good thing in general, I spend the remainder of the chapter explaining how the SCAAT approach uniquely and elegantly addresses these four issues as well as other concerns.

2.1.1 Filtering

To *track* means by definition "to observe or monitor the course of" [American81]. As it turns out, the only means we humans have of observing or monitoring a target is by somehow physically sensing its presence via one of several media, e.g. through electromagnetic waves, mechanical linkage, or inertial sensing devices. In practice, all available means of sensing are susceptible to noise, e.g. electrical, mechanical, or optical noise. Thus an *ideal* system would be able to somehow perfectly separate the true signal from the inherently noisy observations so that the target could be tracked perfectly. See figure 2.1.

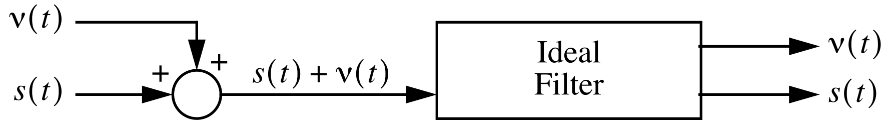


Figure 2.1: An ideal filter. Such a filter would be able to perfectly separate the signal $s(t)$ from the noise $\nu(t)$.

There is another practical limitation imposed by our common desire to use digital computers to perform tracking. Because these computers operate in discrete time steps, we are limited to discrete observations of the target. Thus some uncertainty is introduced in terms of the target dynamics in between our discrete observations. If we somehow knew what the target was doing while we weren't "looking", we might be able to better identify the noise when we do finally "look" again. However, given that our world is not perfect, we would like some means to best estimate the target motion, i.e. we would like an *optimal estimator.*

> *"An optimal estimator is a computational algorithm that processes measurements to deduce a minimum error*[1] *estimate of the state of a system by utilizing: knowledge of system and measurement dynamics, assumed statistics of system noises and measurement errors, and initial condition information."* [Gelb74]

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [Kalman60]. The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means of using noisy measurements to estimate the state of a linear system, while minimizing the *expected* mean-squared estimation error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and can do so even when the precise nature of the modeled system is unknown; it is inherently discrete (not derived from a continuous model) and thus well suited to implementation on a digital computer; it

1. In accordance with some stated criterion of optimality.

can be extended to model systems with continuous dynamics; and (because it is recursive) it makes use of past information in an efficient manner.

The derivation of the filter, and its stated optimality, depends on some assumptions about the statistical properties of the measurement and target dynamics. While in practice these assumptions are rarely absolutely correct, the Kalman filter has often been found to perform extremely well anyway. As such it has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.

2.1.2 Data Fusion

The Kalman filter assumes that the system being estimated has a *measurement equation* of the form given in equation (2.1) where the matrix $H(t_k)$ relates the state vector $\vec{x}(t_k)$ to the measurement vector $\vec{z}(t_k)$, and the vector $\vec{v}(t_k)$ represents the measurement noise. Further details are presented in chapter 4 and can be found in Kalman filter texts such as [Brown92, Gelb74, Jacobs93, Lewis86].

$$\vec{z}(t_k) = H(t_k)\vec{x}(t_k) + \vec{v}(t_k) \tag{2.1}$$

If the system being estimated has multiple forms of observation, there would be multiple corresponding measurement equations equation (2.1), i.e. multiple instances of the matrix $H(t_k)$, each representing a different relationship. By using the appropriate $H(t_k)$ for each type of measurement, the filter effectively combines, blends, or *fuses* the information contained in the heterogeneous measurements.

This capability for heterogeneous *data fusion*, combined with the properties discussed in section 2.1.1, has made the Kalman filter a very popular means of data fusion. For example the Kalman filter has been used for navigation [Geier87, Mahmoud94, Watanabe94], for virtual environment tracking [Azuma95, Emura94, Foxlin96], and for 3D scene modeling [Grandjean89, VanPabst95].

2.1.3 Prediction

The Kalman filter assumes that the system dynamics can be modeled as in equation (2.2) where $A(t_k)$ relates the state at time step *k* to the state at step *k*+1 (in the absence of noise), and $\vec{\omega}(t_k)$ represents the *driving* or *process noise*.

$$\vec{x}(t_{k+1}) = A(t_k)\vec{x}(t_k) + \vec{\omega}(t_k) \tag{2.2}$$

In the classical discrete Kalman filter implementation, the time between filter steps *k* and *k*+1 is constant, and thus there is one instance of $A(t_k)$ that is used at each step of the filter to predict the state for that step as in equation (2.3). The best state estimate of the state at time step *k*+1 given measurements and corresponding estimates through time step *k* is $A(t_k)$ times the previous estimate.

$$\hat{x}(t_{k+1} \mid t_k) = A(t_k)\hat{x}(t_k \mid t_k) \tag{2.3}$$

For obvious reasons, equation (2.3) reflects what is normally called a *one-step prediction*. To predict more than one step into the future, one can then simply use equation (2.3) with an appropriate change to $A(t_k)$. For a continuous-time model, $A(t_k)$ can be implemented as a function of the time *t* so that predicting ahead different amounts of time involves simply changing *t*. For nonlinear relationships an *extended Kalman filter* can be used. For the EKF $A(t_k)$ becomes the Jacobian of the nonlinear function, and the (nonlinear) predictions are realized by integration. (See section B.2 of appendix B, page 174.)

Even if another prediction scheme is used, the Kalman filter can often offer assistance by estimating states that one cannot directly measure. A common example is that of optimally estimating velocities when one only has measures of position. (Note that the alternative of directly computing derivatives from position measurements is inherently susceptible to noise.) With velocities in hand one can better predict where the target *will* be than if the only information is where the target currently *is*. As stated by Henry Fuchs, “A tracker shouldn’t tell us where the user is, it should tell us where the user *will be* [for example] two image frames from now.”

A classic example of prediction in virtual environments is the work done by Rebo at the Air Force Institute of Technology [Rebo88]. So and Griffin [So92] attempted to compensate for system latencies by using a “signal processing algorithm” to predict the viewing position and then “deflecting” the image appropriately. Kalman filters have been employed for similar purposes by Azuma [Azuma95], Liang et al. [Liang91], and more recently by Mazuryk and Gervautz [Mazuryk95]. In particular, [Holloway95] showed that prediction is necessary to address otherwise insurmountable latency-induced errors, and [Azuma95] showed how it can be done with a Kalman filter.

2.1.4 Calibration

Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems; thus efforts are often undertaken to measure components against a known standard in order to improve the accuracy of their estimated characteristics. A particularly relevant example of calibration is that of the work by Gottschalk and Hughes [Gottschalk93]. This work is interesting and particularly relevant because my experiments (chapter 6) reflect calibration of a similar system, using a different scheme.

The Kalman filter is generally presented as a way of estimating values of stochastic variables (the states) of linear systems whose associated system parameters (e.g. model dynamics and noise characteristics) have known values. Interestingly enough, the filter can just as well be turned around and used to estimate values of unknown system parameters when the states are known [Jacobs93]. In fact, it can even be used to estimate both system states and parameters as represented in figure 2.2.

In particular, the characteristics outlined in sections 2.1.1 and 2.1.2 make the Kalman filter an attractive and popular option for calibration. Referring to figure 2.2, the situation depicted in (a) is that of a Kalman filter being used only to estimate the system states, i.e. there is no calibration. Examples of the situation depicted in (b) include [Wefald84] and [Foxlin96]. Recent examples of the situation depicted in (c) include [Azarbayejani95] and the work presented in this dissertation.

2.2 Temporal Concerns

2.2.1 Fewer Observations, Improved Timing and Accuracy

Current commercial and experimental 3D tracking systems obtain some minimum fixed sequence of measurements of one or more sensors to compute a single position and/or orientation estimate. Multiple sensor measurements are usually necessary because the individual measurements offer only partial position and orientation information. The

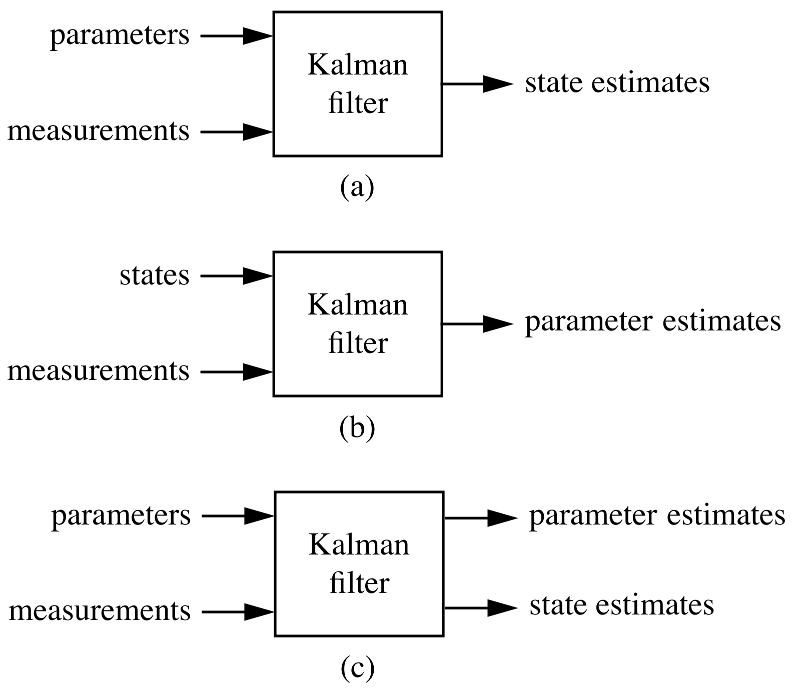


Figure 2.2: State versus parameter estimation. (a) The system parameters are known, and are used with the measurements to estimate the states. (b) For calibration purposes, known states can be used with measurements to estimate the system parameters. (c) A set of parameters can be used in conjunction with measurements to estimate another (not necessarily disjoint) set of parameters *and* the states.

measurements are sequential either because they must be made independently or because replicating the hardware is not cost or space effective. This situation results in several temporal problems when estimating human head and hand motion in virtual environment systems: (a) the estimation rates are relatively low; (b) the latencies are relatively high; and (c) error is introduced when sequential measurements are treated as if they had occurred simultaneously. By estimating with individual sensor measurements, the SCAAT method improves data rates and latencies (a & b) as discussed in this section, and as a bonus it inherently avoids what I refer to as the *simultaneity assumption* (c) as discussed in section 2.7.

2.2.2 Estimation Rates and Latencies

Per Shannon's sampling theorem [Jacobs93] the measurement or *sampling* frequency should be at least twice the true target motion bandwidth, or an estimator may track an alias of the true motion. Given that common arm and head motion bandwidth specifications range from 2 to 20 Hz [Neilson72, Fischer90, Foxlin93], the *sampling* rate should ideally be greater than 40 Hz. Furthermore, the *estimation* rate should be as high as possible so that slight (expected and acceptable) estimation error can be discriminated from the unusual error that might be observed during times of significant target dynamics.

In addition to increasing the estimation rate, we want to reduce the latency associated with generating an improved estimate, thus reducing the overall latency between target motion and visual feedback in virtual environment systems [Mine93]. If too high, such latency can impair adaptation and the illusion of presence [Held87], and can cause motion discomfort or sickness. Increased latency also contributes to problems with head-mounted display registration [Holloway95] and user-motion prediction [Liang91, Friedman92, Azuma94]. Common tracker latencies, excluding communication overhead, range from 10 to 50 milliseconds [Mine93, NRC94].

2.2.3 Measurement Collection

With these requirements in mind, let us examine the effect of sequential measurement collection on the estimation latency and rate. Let τ_m be the time needed to perform one sensor measurement, i.e. to obtain one constraint. Repeating the notation of section 1.2, figure 1.3 in particular, let C be the number of sequential non-degenerate constraints necessary to compute a unique estimate and let N be the *actual* number of sequential constraints (measurements) used to compute each estimate. In addition, let $\tau_c(N)$ be the time needed to compute an estimate given N observations. For example, for the system depicted in figure 1.6, $C = 4$ known coplanar scene points must be observed to compute an estimate in closed form, however $N \geq 4$ independent points may actually be used in an attempt to overcome measurement noise problems. The estimation latency is given by

$$\tau_e = N\tau_m + \tau_c(N), \tag{2.4}$$

and the corresponding estimation rate is given by

$$\rho_e \le \frac{1}{\tau_e} = \frac{1}{N\tau_m + \tau_c(N)}. \tag{2.5}$$

As the number of sequential measurements N increases, equations (2.4) and (2.5) show how the estimation latency and rate increase and decrease respectively. For the Selspot optical tracking system [Selspot], $N = C = 3$ beacons are observed sequentially per estimate (for position and orientation of a single target). For the Polhemus Fastrak magnetic tracking system, $N = C = 3$ sequential excitations are performed and sensed per estimate [Raab79]. For the University of North Carolina (UNC) wide-area optoelectronic tracking system, $10 \le N \le 20$ beacons are observed sequentially per estimate ($C = 3$ for the implemented iterative approach) [Ward92, Azuma91].

A diagram depicting the basic timing of the Polhemus Fastrak magnetic tracker is given in figure 2.3. Each estimation occurs only after sensing an *excitation pattern*. The excitation pattern is composed of three linearly independent *excitation vectors* that are sourced and sensed sequentially in time. The complete excitation pattern contains information sufficient to determine the 3D position and orientation of the target [Raab79].

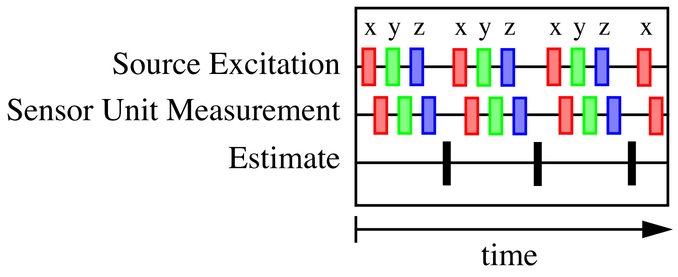


Figure 2.3: Timing diagram for a Polhemus Fastrak magnetic tracker. The state estimate is updated only after measuring a group of linearly independent excitation vectors.

Instead of grouping measurements, the SCAAT method allows the updating of the current estimate with each *individual* sensor measurement as soon as it becomes available. This improves the latency and data rate by effectively letting $N = 1$. A diagram depicting the timing of a (hypothetically) modified SCAAT Polhemus Fastrak is given in figure 2.4. In this case an estimate is available after sensing each individual excitation *vector*. Each such vector does not contain information sufficient to determine the 3D position and orientation of the sensor, but it does contain some information that can be used to improve the current state estimate.

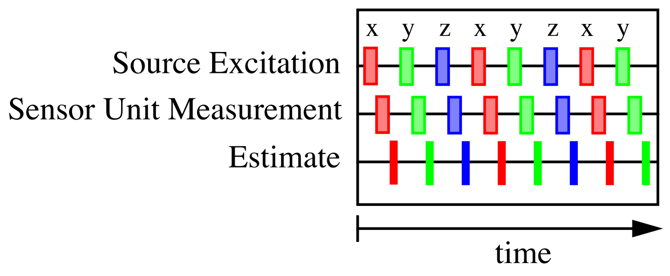


Figure 2.4: Timing diagram for a (hypothetical) SCAAT magnetic tracker. The state estimate is updated after sensing each individual excitation vector.

2.2.4 Computational Complexity

There is a further measurement-related impact on the estimation latency and rate that is indicated by equation (2.4): the computation time τ_c for any algorithm is a function of the number of measurements used to compute a single estimate. In other words, the amount of computation is a function of the amount of measurement information.

For example, given an image-based tracking system (such as that described in section 1.2) with an n-dimensional target state vector (n estimated parameters) and N pairs of observed scene points, a typical Kalman filter implementation would have to perform several $n \times n$ matrix multiplications, and a $2N \times 2N$ matrix inversion to formulate a new estimate.[2] As a function of N the asymptotic computation time for such a system is

$$\tau_c(N) \in O(n^3) + O(N^3) . \tag{2.6}$$

As can be seen by equation (2.6), letting $N = 1$ reduces not only the per-estimate measurement-collection time $N\tau_m$ in equation (2.4), but also the computation time

2. There are many variations on the set of Kalman filter equations, some more computationally efficient than others under certain circumstances. The expression (2.6), which agrees with [Maybeck70] (pp. 322, 356), is an upper bound for the implementation presented in chapter 4.

$\tau_c(N)$. (Note that n is constant for any particular implementation.) A more complete time complexity analysis is presented in section 5.2.

2.3 The Simultaneity Assumption

2.3.1 Mathematics Unaware that Target is in Motion

Most current tracking systems collect sensor measurements sequentially, and then assume (mathematically) that they were collected simultaneously. I refer to this as the *simultaneity assumption*. If the target remains motionless this assumption introduces no error. However if the target is moving the violation of the assumption does introduce error. This situation is depicted in figure 2.5.

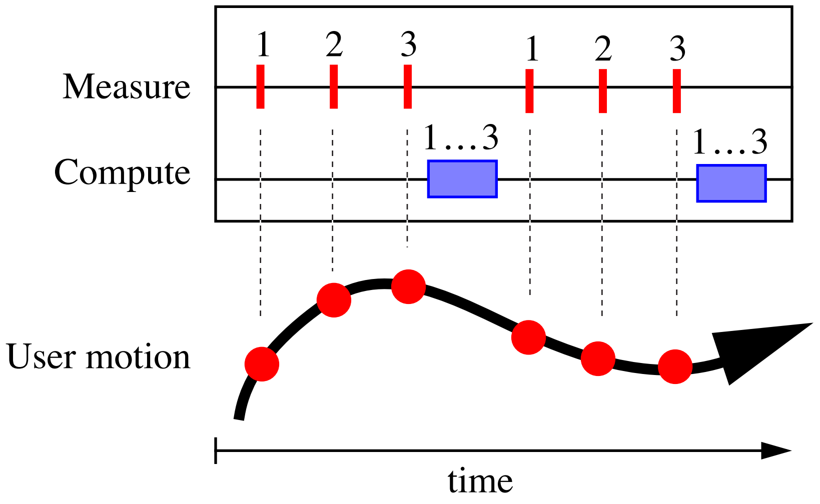


Figure 2.5: The simultaneity assumption. Most current tracking systems collect sensor measurements sequentially, and then assume (mathematically) that they were collected simultaneously. If the user is moving the violation of the assumption introduces estimate error.

It is possible to modify existing closed-form mathematical solutions to address the problem by rectifying the measurements to a common time. However to begin with, such a scheme requires ongoing estimates of the target dynamics, which are inherently maintained by a Kalman filter (including a SCAAT implementation). But more important, such rectification requires some assumptions about the *change* in those dynamics over the period of measurement collection. The longer the measurement collection period, the less likely that *any* assumptions about dynamics will be true. In the case of a SCAAT implementation, not only is the measurement period minimized, but the target dynamics (e.g. the time derivatives of position and orientation) can be maintained as part of the KF state, and thus their estimates will also be improved with each single observation.

2.3.2 When is it a Problem?

In his Ph.D. dissertation, Burton discusses the problems related to “nonsimultaneities” in measurements for his novel 3D position tracker (see [Burton73], pp. 70-75). While he concluded that in practice the problem could be ignored for his system, I would argue that this is not the case for modern tracking systems, given our current goals for accuracy in VE systems (see [NRC94], pp. 189-190). For the purpose of illustration, table 2.1 lists the practical timing characteristics of a few existing tracking systems.

Table 2.1: The practical timing characteristics of a few existing tracking systems. The data is based on [Woltring74, Selspot], [Raab79, NRC94, Kuipers80], and [Ward92, Azuma91] respectively.

	Selspot*	Polhemus Fastrak	UNC ceiling
Latency (ms)	$\tau_e \approx 3$	$20 \le \tau_e \le 30$	$26 \le \tau_e \le 36$
Rate (Hz)	$\rho_e \approx 300$	$30 \le \rho_e \le 120$	$28 \le \rho_e \le 39$

* Estimates are based on estimating position and orientation (three markers per target) for 10 targets.

Typical arm and wrist motion can occur in as little as 1/2 second, with typical “fast” wrist tangential motion occurring at three meters per second [Atkeson85]. For the systems shown in table 2.1, such “fast” motion corresponds to approximately one to ten centimeters of translation *throughout* the sequence of *m* measurements used for a single estimate. For systems that attempt sub-millimeter accuracies, even slow motion occurring during a sequence of sequential measurements impacts the accuracy of the estimates. For example, in a multiple-measurement system with 30 millisecond total measurement time, motion of only three centimeters per second corresponds to approximately one millimeter of target translation throughout the sequence of *m* measurements for one estimate. Figure 2.6 presents the results of a simulation from appendix A (page 161). It shows how estimates can be pulled away from the truth as the simultaneity assumption is violated.

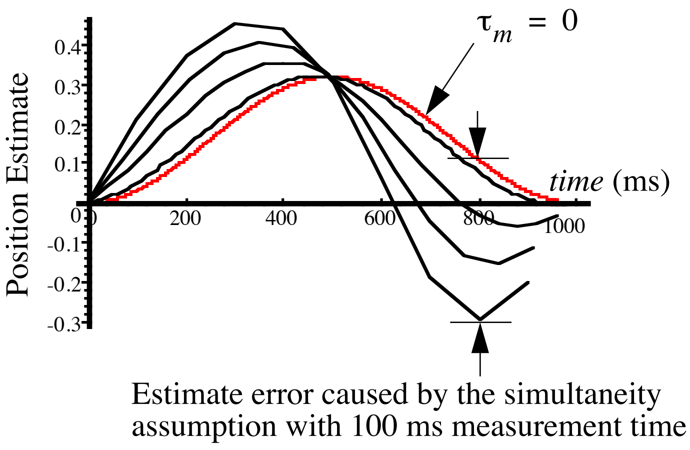


Figure 2.6: Error caused by the simultaneity assumption. The family of curves shows how simulated position estimates become skewed by the simultaneity assumption as a target undergoes motion with a one Hertz sinusoidal velocity. Note the skewing of the estimation with sensor measurement times of $\tau_m \in \{0, 10, 40, 70, 100\}$ milliseconds. This figure is a copy of figure A.6 on page 166. Details of the simulation are presented in appendix A beginning on page 161.

The error introduced by the simultaneity assumption is of even greater concern when attempting any form of autocalibration (discussed further in section 2.5). Gottschalk and Hughes state that motion during autocalibration must be severely restricted in order to avoid such errors [Gottschalk93]. Consider for example a multiple-measurement system with 30 ms total measurement time. With such a system, motion would have to be restricted to approximately 1.5 centimeters per second to confine translation throughout a measurement sequence to 0.5 millimeters. For complete calibration of a large (wide-area) multi-device system such as the UNC optoelectronic tracker, this restriction results in lengthy specialized autocalibration sessions. Appendix A presents a simple but concrete example of a 2D tracking system, and how estimates are adversely affected when the simultaneity assumption is violated.

Surprisingly, even in cases where multiple measurements are truly available simultaneously, experiments have shown that the SCAAT method can still perform better than other conventional approaches. (See section 6.2.5, page 144.)

2.4 Hybrid Systems and Multisensor Data Fusion

2.4.1 Hybrid Systems, the Past and the Future

Tracking systems that employ only one form of sensing all suffer inherent drawbacks. For example, purely inertial trackers suffer from drift, optical trackers require a clear line of sight, and magnetic trackers are affected by ferromagnetic and conductive materials in the environment [Raab79]. To maintain more consistent performance throughout a working environment, across the frequency spectrum, and over a wide range of dynamics, researchers have sought to develop *hybrid tracking systems*.

For example, Foxlin has developed a system that is primarily inertial, but aided by angular position sensors [Foxlin93], both Canadian Aerospace Electronics [NRC94] and the University of North Carolina [Meyer92] have pursued systems that are primarily optical, but aided by inertial sensors (for prediction), researchers at the University of Tokyo have sought to improve the data rate of the Polhemus tracker by augmenting it with rate gyros [Emura94], while researchers at the University of North Carolina have sought to improve the accuracy of the Ascension magnetic tracker by augmenting it with a passive image-based system that observes known fiduciary marks in the real world [State96].

Beyond improved performance in limited working volumes, future systems that will allow people to "move from room to room in a building without loss of tracking" [NRC94, p. 202, *Research Needs*] will almost certainly involve large-scale hybrid systems, necessitating the blending of data from heterogeneous sensors or sensor units. This process of blending data is often referred to as *data fusion* or *multisensor data fusion* (see for example [Crowley93]).

2.4.2 Multisensor Data Fusion

The SCAAT method represents an unusual approach to Kalman filter based multisensor data fusion. Because the filter operates on single sensor measurements, new estimates can be computed as soon as measurements from an individual sensor of any type become available, in virtually any order, and at any (possibly varying) rate. Such flexibility allows measurements from any combination of devices to be interlaced in the most convenient and expeditious fashion, ensuring that each estimate is computed from the most recent data offered by any combination of devices. The information from the various observations can then be blended using either a single SCAAT KF with multiple measurement models, or separate SCAAT filters (and thus statistics) each with single measurement models, in which case the estimates from the various models can be blended using a statistical multi-model approach (see for example [Bar-Shalom93]).

2.4.3 A Simple Hybrid Example

For the sake of illustration, imagine an inertial-acoustical hybrid tracking system composed of three accelerometers, three rate gyros, and an acoustical line-of-sight system (to control drift from the inertial sensors). Note that not only will the acoustical measurements take longer than the inertial measurements, the acoustical measurement times will vary with distance between the source and sensors. A conventional method for data fusion might repeat the fixed pattern shown in figure 2.7. Notice that an estimate is produced only after collecting an orthogonal group of measurements from each (any one) subsystem.

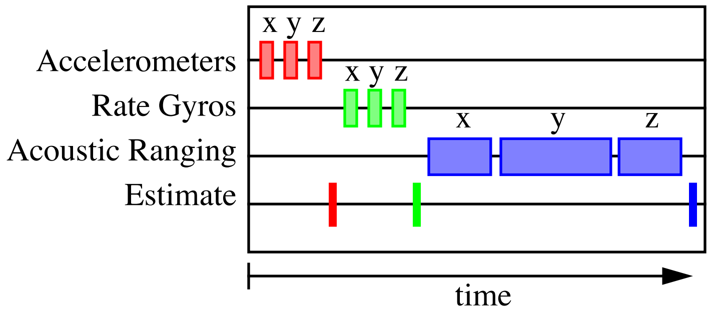


Figure 2.7: Timing diagram for a hypothetical conventional hybrid tracking system. The state estimate is updated only after each group of 3 homogeneous measurements.

In contrast, a SCAAT implementation might interleave sensor measurements as depicted in figure 2.8. Note the flexibility of measurement type, availability, rate, and ordering.[3] Again because an estimate is produced with every sensor measurement, latency is reduced and the estimation rate is increased.

3. The ordering here is round-robin within subsystem. In chapter 4 I explain how the method facilitates other (possibly dynamic) ordering schemes.

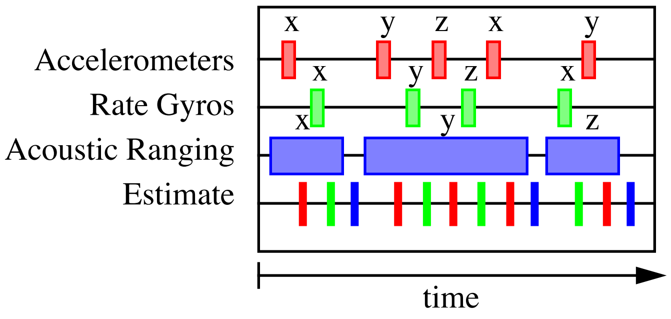


Figure 2.8: Timing diagram for a SCAAT inertial-acoustic hybrid tracking system. The state estimate is updated whenever an individual measurement becomes available.

2.5 Source and Sensor Autocalibration

2.5.1 What to Calibrate?

Following the terminology used in the fields of computer vision and image processing (see for example [Nalwa93] p. 48) I term *intrinsic* those parameters that are internal to a device or a device unit. In general these parameters are associated with a manufacturer's design specifications, but are impossible or unreasonable to control with sufficient precision during manufacture. For example, given the inertia sensing unit depicted in figure 1.1, the precise positions and orientations of the individual inertia sensors within the unit would be considered intrinsic parameters (see figure 2.9). Other geometric examples include the relationships between electromagnetic sources in a magnetic source unit, electromagnetic sensors in a magnetic sensor unit, and of course a camera's lens and image plane. While these parameters are generally static, it is possible for an intrinsic parameter to change with time. For example, a drifting electrical bias may significantly impact the usefulness of a device if not sufficiently monitored and corrected.

I term *extrinsic* those parameters that are external to a device or unit. In general these variations are associated with a specific end user implementation or working environment. These parameters can generally only be determined in the field by the end user. Some extrinsic parameters can be determined one time (statically) during or after installation, others must be or are best determined concurrently (dynamically) during

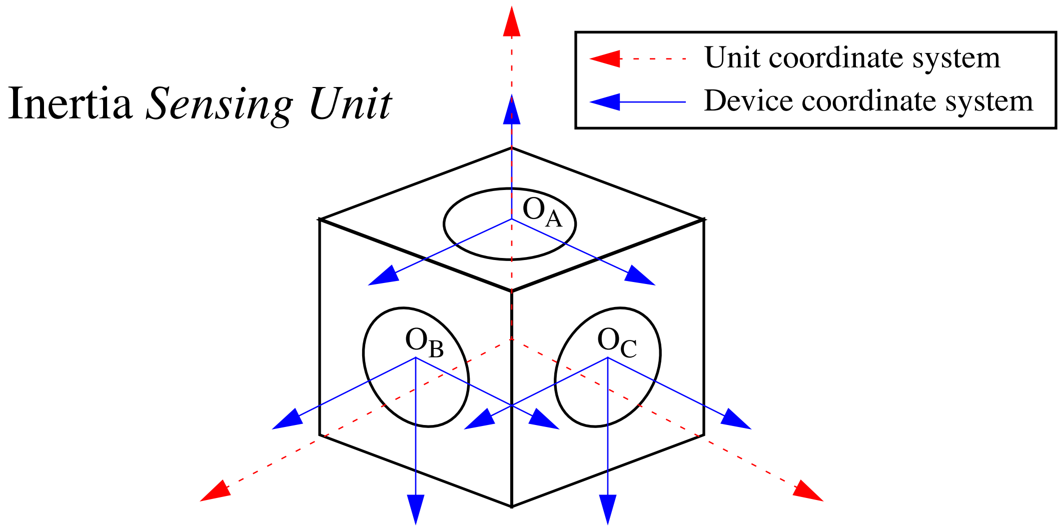


Figure 2.9: Intrinsic parameters. The precise relationship between each inertial sensor's coordinate system (O_A, O_B, and O_C) must be known.

operation. For example, if an end user chose to form a hybrid system by physically joining the sensor units from two commercially-available systems such as those depicted in figure 1.1, the parameters describing the precise geometric relationship between the two sensors would be considered extrinsic parameters (see figure 2.10). Similar geometric relationships must also be determined between all source or sensor units, especially if they are not physically joined. This is true for heterogeneous hybrid systems such as those discussed in section 2.4.1, for homogeneous multi-device systems such as the Ascension Flock of Birds, for small working volumes where precision is necessary, and for large working volumes (e.g. an entire building) where smooth and accurate transitions are desired. Yet even for single-device systems, the user often depends on knowing the precise position and orientation of a device (or multiple devices) in world coordinates. In particular, for applications that attempt registration of synthetic images with real-world objects, such parameters are critical in terms of overall registration error [Holloway95]. Yet the user does not usually have the ability to directly measure such parameters, because the necessary coordinate system is typically "buried" in the device. As such, the user must attempt some form of indirect calibration in the field.

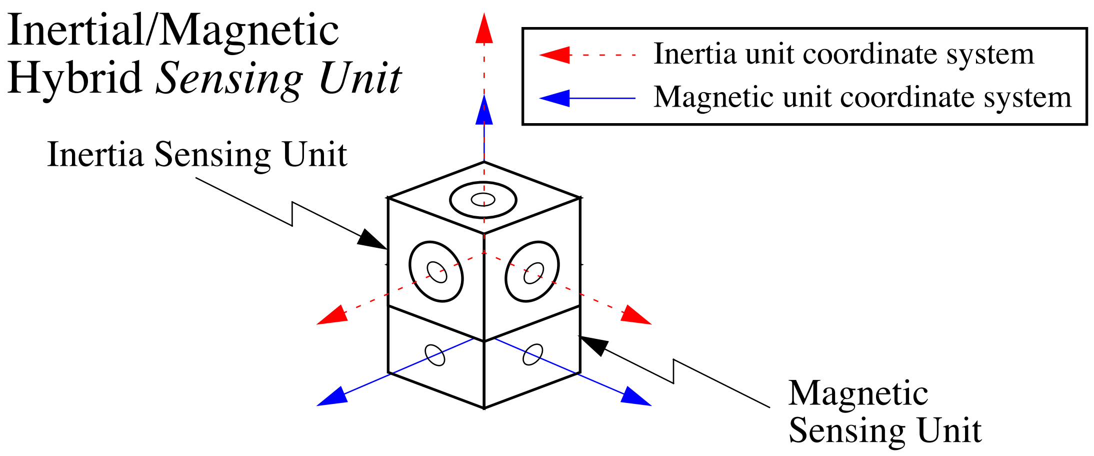

Inertial/Magnetic Hybrid *Sensing Unit*
Inertia unit coordinate system
Magnetic unit coordinate system
Inertia Sensing Unit
Magnetic Sensing Unit

Figure 2.10: Extrinsic parameters. A hypothetical inertial/magnetic hybrid where the two respective units are physically attached. The precise relationship between the units' coordinate systems must be known.

2.5.2 When and Where to Calibrate?

In general, static intrinsic parameters can be determined in the factory by the manufacturer prior to individual product shipment. However, not all manufacturers perform the calibrations necessary for their device to be used under the conditions required for tracking in virtual environments. For example, a camera manufacturer may only provide the intrinsic camera parameters as specified in their design, as opposed to the *actual* parameters for each product. If this is the case, or if a manufacturer's data is suspect, the user can usually perform such calibrations in the field under properly controlled conditions, one time prior to use.

On the other hand, extrinsic parameters are entirely dependent on the manner in which the devices or units are used in the field. As such they have to be determined by the end user. In cases where the parameters are few and known to remain constant, they can be calibrated under properly controlled conditions, one time prior to use. On the other hand, if there are many devices, it may be impractical to perform such calibrations in a thorough manner. Consider for example a wide-area system such as the optoelectronic tracker developed at the University of North Carolina [Ward92] where there are literally thousands of beacons whose precise position with respect to each other (or the world) must be known.

Even if the number of parameters is limited, it may be the case that some, typically extrinsic, will vary over time. For example, mechanical vibrations can slightly move devices from their original (possibly calibrated) position and orientation, electrical biases can drift over time, and changing temperatures can affect all sorts of devices both mechanically and electrically. In such cases it might be desirable or indeed necessary to continually refine estimates of certain parameters dynamically while the system is in operation.

2.5.3 Calibration vs. Autocalibration

While knowledge about intrinsic or extrinsic parameters is most commonly obtained via off-line calibration under controlled circumstances, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or *autocalibration* attractive. The idea is that a system should, in some sense, calibrate itself. At the very least, this means that it should be possible for a user to calibrate a system, using the system itself without additional apparatus. This idea is certainly not new (see for example [Wang90, Watanabe94, Gottschalk93, Azarbayejani95]), but autocalibration appears more difficult with current tracking schemes as discussed in the next section.

2.5.4 Why SCAAT?

For conventional systems, the practice of mathematically grouping individual measurements makes assessment of *individual* device characteristics difficult as depicted in figure 2.11. If in retrospect one can somehow determine that a particular estimate (computed with a particular group of measurements) was “bad”, it becomes difficult to determine the individual error contributions of the devices used in obtaining the group of measurements. It may well be the case that one bad apple spoils the entire bunch.

On the other hand, because the SCAAT method generates a new tracking estimate with each individual measurement, individual device imperfections are more readily identified. Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.3.2 are removed, and autocalibration can be performed while concurrently tracking a target under normal conditions. The specific autocalibration method is presented in chapter 4, with experimental results in chapter 6.

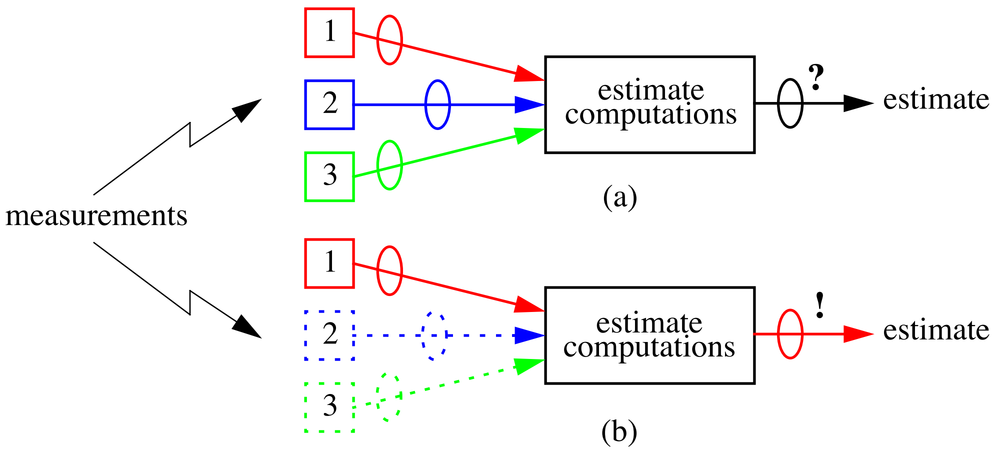


Figure 2.11: Autocalibration and attribution of measurement error. (a) Most algorithms operate on multiple measurements as a group, hence uncertainty or error (represented by the ellipses) in the final estimates is difficult to attribute to any individual sensor. (b) With the SCAAT method, uncertainty in final estimates can more easily be attributed to a particular individual sensor.

2.6 Broad Applicability

In terms of tracking or navigation, the SCAAT method can be implemented with many types and combinations of devices as shown in table 2.2. In each example, the measurement update of the SCAAT method would involve a minimal set of devices. In some cases, current manufacturers would have to provide new "hooks" for the SCAAT method. In particular, in addition to the conventional (complete) estimates they would have to provide access to the individual observations.

Beyond tracking or navigation, the SCAAT method can be applied to any situation where stochastic estimation is desired, and multiple measurements (simultaneous or not) are used to form a complete estimate. Some examples are real-time or off-line state estimation, parameter estimation [Lewis86], and generalized data fusion or assimilation (e.g. see [Watanabe94, VanPabst95, Grandjean89, Mahmoud94, Ikeda95]). As such I believe that this method may prove to be of interest to the larger scientific and engineering community in addressing a more general class of tracking and estimation problems.

Table 2.2: Some example technologies and SCAAT observations.

Technology	SCAAT Observation
GPS*	One satellite
Computer Vision	One camera, or one feature
Optoelectronic	One LED
Inertial	One accelerometer or rate gyro
Acoustic	One transducer
Mechanical	One joint (one potentiometer)
Magnetic	One excitation and sense

* See section 3.8 on page 71 for discussion of Kalman filters and GPS.

2.7 Putting Things In Perspective

Some people claim that for most VE systems insurmountable downstream latencies are so significant that tracker accuracy and timing concerns have become moot. On the contrary I believe that tracker timing, accuracy, and stability are critical concerns.

While other latencies certainly do exist in the typical VE system [Mine93, NRC94, Wloka95] tracker latency is unique in that it (with the estimation rate) determines how much time elapses before the *first possible opportunity* to respond to user motion. When the user moves, we want to know as soon as possible. For example, post-rendering *image deflection* techniques have been employed in an attempt to address the insurmountable latencies in the image generation pipeline [So92, Mazuryk95]. To perform best, these techniques need readily available, recent, and accurate tracking data.

Furthermore while accuracy and stability are important in their own right, they are particularly important with respect to prediction. Without highly accurate and timely estimates of where a user is and has been, one can hardly hope to predict where they will be in the near future. Holloway showed that prediction is necessary to address the otherwise insurmountable latency-induced errors [Holloway95], while Azuma and Bishop saw how prediction can introduce jitter into image sequences when the tracker data is not accurate and stable [Azuma94, Azuma95].

Chapter 3. Related Work

This chapter contains the primary discussion of previous work related to the topic of this dissertation. In chapter 2 and chapter 7 I also introduce some related work, but it is related specifically in terms of motivation and proposed future work respectively.

3.1 Bearings-Only Tracking

The SCAAT method involves the unusual use of a Kalman filter to estimate a globally observable system using only measurements from locally unobservable systems. The Kalman filter has been similarly used to solve the *bearings-only* target motion analysis problem [Lindgren78, Petridis81]. However the bearings-only circumstances differ in that the systems are *inherently* unobservable due to the availability of only one (moving) sensor.

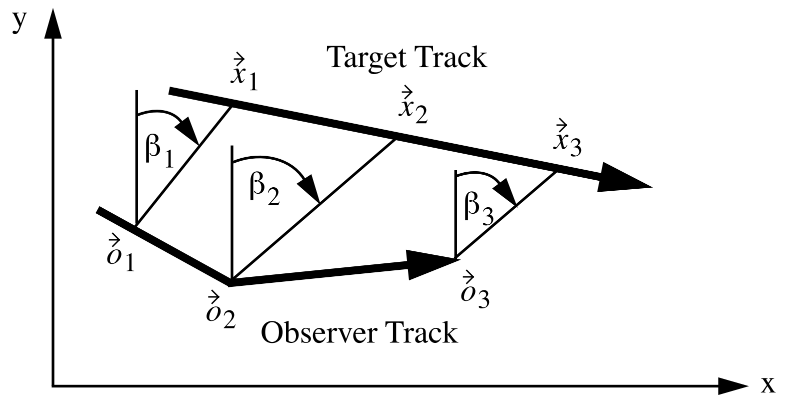


Figure 3.1: The circumstances surrounding the bearings-only tracking problem. The target is tracked by observing the bearing angles β_k of the target at positions $\vec{x}_k$ with respect to the corresponding observer positions $\vec{o}_k$. At each time t_k the system is inherently unobservable, so periodic observer maneuvers are necessary (e.g. leg $\overrightarrow{o_2 o_3}$).

As depicted in figure 3.1, the problem involves trying to estimate the two-dimensional position and velocity of a moving target, using only bearing measurements from a single moving observer that is equipped with a single passive sensor, e.g. a passive sonar device.

During any one leg of observer motion the corresponding system is unobservable. Thus periodic observer maneuvers, e.g. leg $\overrightarrow{o_2 o_3}$, are required to obtain complete target observability.

VE tracking systems, on the other hand, do not normally face a similar single-sensor problem, but rather make use of multiple sensors and sources that are used quasi-simultaneously to obtain complete observability of the target. Despite the availability of multiple sensors, I present motivation and a method for recasting such *multiple-sensor* problems in terms of *single-sensor* observations.

3.2 Sequential Updates

The Kalman filter operates in a predictor-corrector fashion, repeating a single *time update* and *measurement update* step whenever a new measurement vector becomes available. In the time update step the filter *predicts* what the state should be at the time of the measurements, based on the previous state estimate and a model of the process dynamics. In the measurement update step the filter uses the newly available measurement vector to correct the predicted state. In a normal implementation, depicted in figure 3.2, the time and measurement steps do not occur until *all* of the components of the measurement vector are available, i.e. until the state can be determined uniquely from the measurement vector. The measurement update step then processes the entire measurement vector in one batch, e.g. all three measurements in figure 3.2. This batch processing of the measurement data is relatively inflexible and can be computationally expensive if the measurement vector is large.

However if portions of the complete measurement vector are available at different times as depicted in figure 3.2, an equivalent method of *sequential processing of measurement data* [Brown92] is often suggested as a means to add flexibility while simultaneously reducing the computational complexity of the measurement update step. This “sequential updates” method, depicted in figure 3.3, involves performing the normal time update step, but then breaking-up the measurement update step into several smaller steps that each process only a portion of the complete measurement vector, e.g. one measurement in figure 3.3. The complete measurement vector is partitioned into sub-vectors that are believed to be independent. As individual measurements become

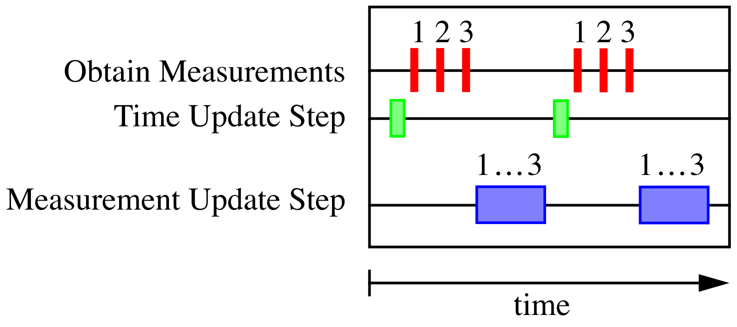


Figure 3.2: A timing diagram for a conventional Kalman filter. When all of the process measurements (3 in this example) are available, the filter proceeds with complete time and measurement update steps.

available, only their corresponding partition is processed. This is repeated until all of the partitions have been processed, at which time the measurement update step is complete and the filter is ready for another time update step. This sequential processing of the measurement data is more flexible and usually results in fewer computations.

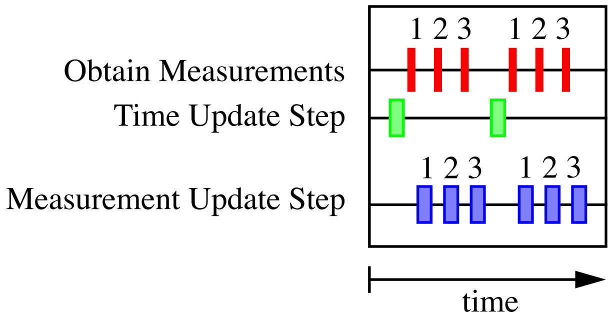


Figure 3.3: A timing diagram for a Kalman filter using the *sequential updates* method. When all of the process measurements (3 in this example) are available, the filter proceeds with a time update step, but then processes the measurements sequentially during the measurement update step.

As in the sequential updates method, the SCAAT method processes individual measurements sequentially. However, similar to the standard batch method, the SCAAT method performs only one set of computations during the measurement update. The difference is that a SCAAT implementation maintains no explicit notion of a complete measurement vector, and therefore does not need to construct one during the measurement update step. Instead each individual process measurement is *treated* as if it was individually complete. In particular a SCAAT implementation repeats a time update and measurement update step for every new *individual* measurement as shown in figure 3.4.

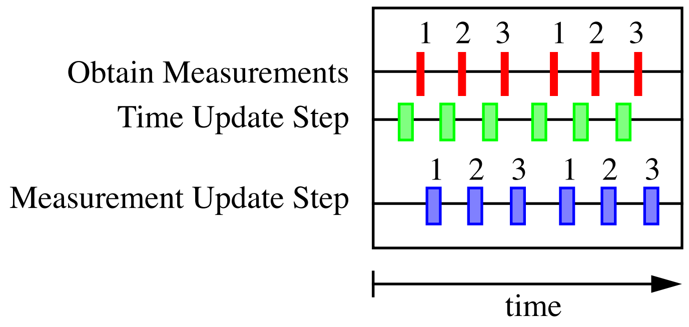


Figure 3.4: A timing diagram for a Kalman filter using the SCAAT method. When any individual process measurement is available, the filter proceeds with complete time and measurement update steps.

With the sequential updates method there is just one time update step for *all* measurement update steps associated with a complete measurement vector. Thus the filter dynamics remain static during the sequential processing of the measurements. On the other hand, with the SCAAT method there is one time update step for *each* measurement update step. Thus the filter dynamics continue to change during the sequential processing of the measurements, based on the current state and the process model.

3.3 The Iterated Kalman Filter

The extended Kalman filter is often employed for optimal estimation of a nonlinear process, i.e. a process where the relationship between the state vector and the measurement vector is nonlinear (e.g. see [Gelb74] or [Bell93]). The standard EKF implements this relationship in the form of a nonlinear *measurement function* which is used to predict a measurement vector given a particular state vector. The *iterated Kalman filter* or IKF (see [Bell93], p. 190) successively repeats the measurement update step, using a truncated Taylor series expansion of the measurement function in place of the standard (un-expanded) measurement function. In this way it seeks to improve an estimate by continually linearizing about the most recent estimate. (See figure 3.5.) Usually the iterations are repeated until little further improvement is noticed from additional repetitions.[1] Operating in this way, the IKF can significantly reduce EKF estimate errors caused by the nonlinearities of the measurement function [Wishner69].

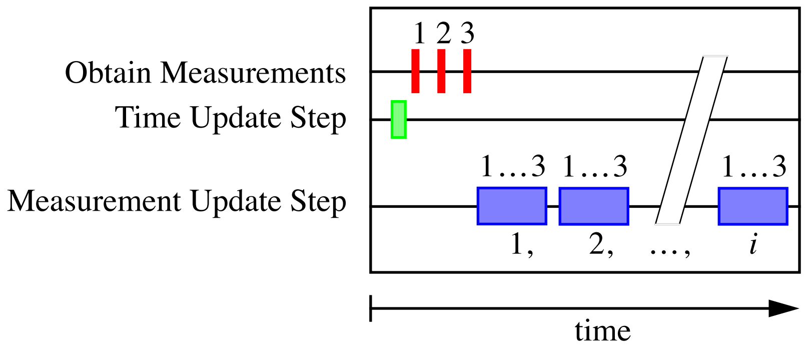


Figure 3.5: A timing diagram for an iterated Kalman filter. The filter iterates the measurement update step some number of times i using a truncated Taylor series expansion of the measurement function.

The IKF does not advance time, i.e. perform a time update step, between iterations of the measurement update step because it is iterating with a single measurement vector. While a SCAAT filter could be implemented to iterate the measurement update step as in an IKF, each iteration contributes to the computation time, thus likely confounding efforts to improve the estimate rate and latency as discussed in section 2.2. Certainly if there was nothing better to do while waiting for a new measurement it would make sense.

3.4 The Collinearity Method

The original UNC optoelectronic tracker (1991-1995) estimated the target position and orientation by setting up and attempting to solve for a set of *collinearity condition equations* [Ward92, Azuma91]. The equations represent some geometric relationships between various vectors in world and camera coordinates that must exist for the target to be in a particular state while observing a particular set of images. Therefore given a set of images and the imaging conditions, the target state can be estimated uniquely, but not necessarily in closed form. As such, a Collinearity implementation inherently bases

1. Bell et al. [Wishner69] show how the iterated measurement update step of the IKF is an application of the Gauss-Newton method for approximating a maximum likelihood estimate.

estimates on a *collection* of observations that when used together uniquely determine the unknowns.

In practice, because the observations may be noisy or the system may be ill-conditioned, the UNC implementation of the Collinearity method generally collected an over-abundance of observations prior to attempting an estimate, thus overconstraining the system of unknowns and rendering it completely observable. Besides being slow because of the need for multiple observations, the Collinearity method cannot directly estimate the target motion derivatives, e.g. linear and angular velocities. (See [Azuma91] for a complete description and mathematical derivation.) The newer UNC optoelectronic HiBall tracker (1997) on the other hand is expected to use the SCAAT method for tracking. As discussed in section 1.2 the SCAAT method bases estimates on individual observations so it is faster and more accurate as described in chapter 2.

Not only are the Collinearity and SCAAT methods related by function, but the UNC Collinearity implementation provides a real example of a batch-measurement strategy that can be compared with a corresponding SCAAT implementation. Indeed the primary experimental comparisons presented in chapter 6 are between Collinearity and SCAAT implementations.

3.5 Autocalibration (Gottschalk & Hughes)

In 1993 Gottschalk and Hughes presented a method for *autocalibrating* beacon positions in the original UNC optoelectronic tracking system [Gottschalk93]. (The original tracking system and its more recent version are briefly described in appendix D, beginning on page 191.) The presentation of a method for autocalibration was a significant event because it meant that the special ceiling panels (see figure D.1 on page 191) could be made less precise, hence less expensive and easier to install.

The SCAAT method for autocalibration, when applied to the same tracking system, differs from that of [Gottschalk93] not only in terms of the actual method but also in terms of the circumstances under which the respective methods can be used. The autocalibration method of [Gottschalk93] required no special equipment, but it needed to be performed off-line in a special autocalibration session. In contrast, the SCAAT method facilitates autocalibration *concurrently* with unrestricted (normal) ongoing tracking.

In chapter 6 (beginning on page 119) I present SCAAT simulations of the UNC HiBall tracking system, a more recent version of the system described in [Gottschalk93]. As part of the chapter 6 experiments I evaluate SCAAT autocalibration.

3.6 The SELSPOT System

As in the original UNC optoelectronic tracking system discussed in section 3.4, the SELSPOT optoelectronic tracking system [Woltring74] also employs cameras, optoelectronic beacons, and the collinearity condition equations. It differs in that (1) it directly estimates position only, (2) it does so off-line rather than in real time, and (3) while it employs a Kalman filter it does so in a normal (overconstrained or locally observable) fashion. The measurement update of the Kalman filter incorporates a pair of 2D images of a beacon from two distinct cameras. Thus the measurements obtained per estimate (2x2=4) overconstrain the system of unknowns (3 position parameters), and the system is completely observable.

3.7 MCAAT Estimators

For the purpose of comparison with a SCAAT Kalman filter, I categorize estimators that use $N > 1$ constraints per estimate as *multiple-constraint-at-a-time* or MCAAT estimators. In figure 1.3 on page 38 this represents all cases at or above $N_{\text{ind}} = 1$ (SCAAT) as long as $N > 1$. Furthermore, following the discussion in "A Single Constraint" on page 37, I categorize estimators that employ $N_{\text{ind}} \geq C$ constraints per estimate as *well-constrained* estimators. In figure 1.3 this represents all cases where $N \geq N_{\text{ind}} \geq C$. For example I would say that the SELSPOT optoelectronic tracking system discussed in section 3.6 employs a well-constrained Kalman filter. Also figure 3.2 depicts a well-constrained approach. Indeed one of the reasons the SCAAT method is so interesting is that the well-constrained implementation is the normal approach. As such the literature is full of examples of what I call well-constrained estimators. For some examples, see [Brown92] chapters 6 through 9.

Because the typical Kalman filter implementation operates in a well-constrained fashion, the experiments presented in chapter 6 compare a SCAAT implementation with both a Collinearity (section 3.4) and a multiple-constraint-at-a-time Kalman filter implementation.

3.8 Kalman Filters and GPS

The Kalman filter is often used in conjunction with GPS receivers to obtain filtered position estimates. In addition, because the Kalman filter can operate with incomplete information, it offers the advantage that one can continue to obtain position estimates for a short period of time while less than the requisite number of satellites are available. As pointed out in [Kaplan96] (p. 54) the ability to incorporate such incomplete sets of measurements can be advantageous for example when an aircraft executes a banked turn (and antennae are blocked) or a user is in a forest with an excessively thick canopy.

The SCAAT method takes this approach to the extreme by operating a Kalman filter with *only* partial measurement sets—a single constraint at a time. While the Kalman filter has been used to ride out exceptional cases of partial satellite data for short and infrequent periods of time, I am not aware of anybody suggesting, much less justifying, intentionally riding out partial data as the normal method of tracking as I do in presenting the SCAAT method. A SCAAT implementation for the purpose of GPS position estimation would operate *only* on *individual* satellite observations, i.e. one satellite at a time.

Chapter 4. The SCAAT Method

In this chapter I attempt to describe the SCAAT method in very general terms. In doing so I assume that the reader is somewhat familiar with the Kalman filter. As such I only provide with a very brief introduction to the Kalman filter in section 4.1; however a more extensive introduction with some relatively simple examples is provided in appendix B. For those who desire a more thorough treatment, I suggest the following external resources (in order):

- a very accessible introduction is given in Chapter 1 of [Maybeck70];
- a more complete introduction can be found in [Sorenson70], which also contains some interesting historical narrative;
- entire books on the Kalman filter include [Brown92] and [Grewal96];
- some estimation and control theory texts that also include extensive discussion of the Kalman filter include [Jacobs93, Gelb74, Maybeck70, Lewis86].

The SCAAT method is presented here using an *extended Kalman filter* because it accommodates both linear and nonlinear measurement models, as discussed in section 4.1 and the preceding references. If all dynamic and measurement models were known to be linear, a standard Kalman filter could just as well have been used.

In section 4.2 I describe the SCAAT Kalman filter components, in section 4.3 I describe the associated algorithm for tracking, and in section 4.4 I present modifications to facilitate concurrent device autocalibration.

4.1 The Kalman Filter in Brief

Ever since R. E. Kalman published his seminal paper in 1960 [Kalman60] the Kalman filter has been the subject of extensive research and application. It has been used successfully for over 30 years in applications ranging from weather and economic forecasting, to aircraft and spacecraft navigation.

The Kalman filter is a set of mathematical equations that allows one to recursively estimate the state of a process, given a model of the process dynamics, a model of the measurement system, and an ongoing sequence of discrete, noisy measurements. The process or dynamic model may be either continuous or discrete, but it is assumed to be driven by normally distributed (Gaussian) white noise called the *process noise*. The measurement model, which is discrete, deduces what a measurement should be given the current process state. The ongoing measurements are assumed to be corrupted by normally distributed (Gaussian) white noise with known variance.

The filter operates in a predictor-corrector cycle, predicting the process state using the dynamic model, and then correcting the prediction with a (noisy) measurement. In doing so it inherently maintains the first two statistical moments of the process state: the mean and the variance. The Kalman filter is optimal in the sense that if several conditions involving the process, the models, and the actual measurements are met, it minimizes the expected squared estimate error. While in practice these conditions are often not met, the Kalman filter is nevertheless very robust and quite often performs quite satisfactorily even in sub-optimal circumstances.

While the original discrete Kalman filter described in [Kalman60] assumes linear dynamic and measurement models, nonlinear models can be accommodated using a variation called the *extended Kalman filter*. At each filter update step the EKF linearizes its state and measurement predictions about the previous estimate using a truncated Taylor series. While this linearization means that the filter will be sub-optimal, the EKF has nonetheless proved very effective and has thus been historically very popular.

4.2 The SCAAT Kalman Filter

4.2.1 The Dynamic (Process) Model

Modeling the Target Motion

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* (PV) model to suffice for the dynamics of our sample data sets.[1] In fact we use this same PV model for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [Brown92] pp. 415-420. While we believe that a different dynamic model might improve performance under some circumstances, we consider such *model identification* to be future work. This is discussed in section 7.1 of chapter 7.

The continuous-time PV model is depicted in figure 4.1. This model assumes that the accelerations can be modeled as zero-mean white noise sources $\vec{\eta}[i]$. The velocities are modeled as random walks (integrated white noise), while the position/orientation states are modeled as integrated random walks. The relevant units for the noise sources are given in table 4.1.

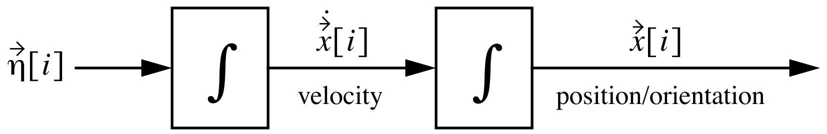


Figure 4.1: Continuous-time position-velocity dynamic model. The linear and angular velocities are modeled as a random walks. Each $\vec{\eta}[i]$ reflects the presumed white noise source that drives the model for the position and orientation state elements corresponding to $i \in \{x, y, z, \phi, \theta, \psi\}$.

Table 4.1: Output, correlation, and spectral density units for noise sources in figure 4.1

Property	Units for $i \in \{x, y, z\}$	Units for $i \in \{\phi, \theta, \psi\}$
Output of $\vec{\eta}[i]$	$[\text{meters}/\text{sec}^2]$	$[\text{radians}/\text{sec}^2]$
Correlation* $\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon) = E\{\vec{\eta}(t)\vec{\eta}^T(t+\varepsilon)\}$	$[\text{meters}/\text{sec}^2]^2$	$[\text{radians}/\text{sec}^2]^2$
Spectral density* $\bar{\Psi}_{\vec{\eta}\vec{\eta}}(\omega) = F\{\Psi_{\vec{\eta}\vec{\eta}}(\varepsilon)\}$	$\dfrac{[\text{meters}/\text{sec}^2]^2}{[\text{radians}/\text{sec}]}$	$\dfrac{[\text{radians}/\text{sec}^2]^2}{[\text{radians}/\text{sec}]}$

* [Maybeck70, pp. 137-140] $E\{\bullet\}$ and $F\{\bullet\}$ indicate expectation and Fourier transform.

1. For most tracking problems the precise target motion model is unknown. The resulting effect on filter stability is discussed in section 5.1.5.

The variances of the noise sources can be determined empirically and tuned for different dynamics, e.g. using the method described in section E.4 of appendix E (page 197).

4.2.2 The Model State and Discrete Transitions

Because our actual implementation is discrete with inter sample time δt we model the discrete state transitions with the standard Kalman filter linear difference equation

$$\vec{x}(t) = A(\delta t)\vec{x}(t-\delta t) + \vec{\omega}(t). \tag{4.1}$$

In the standard model corresponding to equation (4.1), the n dimensional Kalman filter *state vector* $\vec{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [Broida86, Azarbayejani95] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\vec{x}(t)$ we maintain the complete target position as the cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations $(\Delta\phi, \Delta\theta, \Delta\psi)$ about the (x, y, z) axis. Externally we maintain the complete target orientation as the *external quaternion* $\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [Hamilton1853, Chou92] for discussion of quaternions.) At each filter update step, the incremental orientations $(\Delta\phi, \Delta\theta, \Delta\psi)$ are factored into the external quaternion $\vec{\alpha}$, and then zeroed as shown below in section 4.3. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\vec{x}(t)$. We maintain the complete angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The complete target state is then represented by the $n = 12$ element internal state vector

$$\vec{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \Delta\phi & \Delta\theta & \Delta\psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{4.2}$$

and the four-element external orientation quaternion

$$\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \tag{4.3}$$

where the time designations (t) have been omitted for clarity.

The $n \times n$ *state transition matrix* $A(\delta t)$ in equation (4.1) projects the state forward over the discrete sample time δt. For this linear model, the matrix implements the relationships

$$\begin{aligned} x(t) &= x(t-\delta t) + \dot{x}(t-\delta t)\delta t \\ \dot{x}(t) &= \dot{x}(t-\delta t) \end{aligned} \tag{4.4}$$

for the x position element, and likewise for the remaining elements of equation (4.2). Thus the complete state transition matrix $A(\delta t)$ is given by

$$A(\delta t) = \begin{bmatrix} 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

4.2.3 The Process Noise

Uncertainty in the Target Motion

The n dimensional *process noise vector* $\vec{\omega}(t)$ in equation (4.1) models the uncertainty in the state elements since the previous sample at time $t - \delta t$. The elements of the process noise vector are normally-distributed zero-mean white sequences that model the driven responses of the respective states due to the presence of the white noise sources $\vec{\eta}$ over the sample time δt. Note that I associate the process noise vector in equation (4.1) with sample time t as opposed to time $t - \delta t$. We assume that the elements of $\vec{\omega}(t)$ are uncorrelated over time, but as can be seen in figure 4.1 where $\frac{d}{dt}x(t) = \dot{x}(t)$, those elements of $\vec{\omega}(t)$ associated with the same spatial coordinates will have non trivial cross correlations at any particular instant in time. These instantaneous mutual correlations are reflected in the corresponding $n \times n$ *process noise covariance matrix* Q of equation (4.6), where $E\{\bullet\}$ denotes the statistical expected value. Note that "covariance" is used here in the statistical sense, and should not be confused with "covariant" in the differential geometry sense.

$$E\{\vec{\omega}(t)\vec{\omega}^T(t+\varepsilon)\} = \begin{cases} Q, & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases} \tag{4.6}$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [Brown92] (pp. 221-222) to compute a *sampled* process noise covariance matrix which I denote $Q(\delta t)$. (Because the associated random processes are presumed to be time stationary, I present the process noise covariance matrix as a function of the inter-sample duration δt only.) For the model of figure 4.1, the non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i, i] &= \vec{\mu}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i, j] = Q(\delta t)[j, i] &= \vec{\mu}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j, j] &= \vec{\mu}[i](\delta t) \end{aligned} \tag{4.7}$$

for each pair

$$(i, j) \in \{(x, \dot{x}), (y, \dot{y}), (z, \dot{z}), (\phi, \dot{\phi}), (\theta, \dot{\theta}), (\psi, \dot{\psi})\}, \tag{4.8}$$

where

$$\vec{\mu}[i] \equiv \Psi_{\vec{\eta}\vec{\eta}}(0)[i, i]. \tag{4.9}$$

The remaining elements of $Q(\delta t)$ are 0. The matrix $\Psi_{\vec{\eta}\vec{\eta}}(0)$ is the correlation kernel of the noise sources $\vec{\eta}$ shown in figure 4.1. (See also table 4.1.) Note that when comparing the position and orientation elements of figure 4.1, table 4.1, and equation (4.8) with the state vector elements in equation (4.2), $\phi \Leftrightarrow \Delta\phi$, $\theta \Leftrightarrow \Delta\theta$, and $\psi \Leftrightarrow \Delta\psi$. Because the noise source elements $\vec{\eta}$ are independent and the off-diagonal elements of $\Psi_{\vec{\eta}\vec{\eta}}(0)$ are zero, I adopt the abbreviated diagonal-vector notation of equation (4.9). Determination of the autocorrelation elements $\vec{\eta}$ is discussed in section E.4 on page 197.

The elements $Q(\delta t)[i, i]$ from equation (4.7) are the variances of the noise seen at each state due to the presence of the zero-mean white noise sources $\vec{\eta}[i]$ during the sample interval δt. The variance units for the elements $Q(\delta t)[i, i]$ are given in table 4.2.

Table 4.2: Variance units for the elements .

$i \in \{x, y, z\}$	$i \in \{\phi, \theta, \psi\}$	$i \in \{\dot{x}, \dot{y}, \dot{z}\}$	$i \in \{\dot{\phi}, \dot{\theta}, \dot{\psi}\}$
$[\text{meters}]^2$	$[\text{radians}]^2$	$[\text{meters/second}]^2$	$[\text{radians/second}]^2$

The covariance matrix $Q(\delta t)$ describes a multivariate normal (Gaussian) probability density similar to that depicted in figure 4.3 on page 85 for a subset of the elements in equation (4.2). The density exists in the state space defined by equation (4.2), and reflects uncertainty in the predicted state $A(\delta t)\vec{x}(t - \delta t)$ in equation (4.1), given the estimated process noise source magnitudes and the dynamic model configuration. As the inter sample time δt grows, so does the process noise covariance $Q(\delta t)$, resulting in an increasing uncertainty in the value of $A(\delta t)\vec{x}(t - \delta t)$.

4.2.4 The Measurement Model

The use of a Kalman filter requires not only a dynamic model as described in section 4.2.1, but also a *measurement model*. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (4.2) and (4.3). The prediction is then compared with an actual measurement, and the results are used to generate a correction for the filter's current estimate of the target state. If the system employs multiple heterogeneous sensors, one must develop multiple corresponding heterogeneous measurement models.

> *It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes the SCAAT Kalman filter from a well-constrained Kalman filter.*

In the remainder of section 4.2.4 I discuss what to consider when deciding how and when to measure the available sensors, and how to incorporate the corresponding information. This includes discussion of how sensors, measurements, and constraints are related, and what to consider when choosing, designing, and implementing the measurement models for a SCAAT Kalman filter.

Single Scalar Measurements

In choosing the measurement elements to incorporate during a SCAAT Kalman filter measurement update one must consider the available sources and sensors as described in “Defining the Nomenclature” on page 35, and then identify the constraints and corresponding measurements that will be used to update the filter. Recall from “A Single Constraint” on page 37 that in the purest mathematical sense, a single constraint corresponds to one scalar equation describing a known relationship between the unknown state vector elements. Correspondingly a SCAAT Kalman filter should, in the purest sense, generate each new estimate with only a single scalar measurement from one source and sensor pair.

Multi-Dimensional Measurements

However, just as it sometimes makes sense to discuss estimation problems in terms of multi-dimensional constraints, e.g. geometric constraints involving points, lines, and planes, it sometimes makes sense to implement a SCAAT Kalman filter that incorporates one *multi-dimensional* constraint per estimate. Such might be the case if, for example, the available devices can be used in a fashion that yields m simultaneous scalar elements per measurement. Note that as long as $1 \leq m < C$, where C is as described in “A Single Constraint” on page 37, the m-dimensional measurement offers incomplete information. A Kalman filter that relies only on a stream of such multi-dimensional but incomplete measurements still embodies the incremental SCAAT idea.

Simultaneous Multi-Dimensional Measurements

As suggested at the end of section 2.3.2, which begins on page 54, even if the available devices can be used in a fashion that yields $m \geq C$ *simultaneous* or near-simultaneous scalar elements per measurement, it is sometimes appropriate to sequentially incorporate smaller groups of m_σ elements per estimate for some $m_\sigma < m$. Such might be the case if, for example, the m scalar elements are obtained simultaneously from multiple sources and/or sensors that one wants to calibrate individually. By sequentially incorporating the m simultaneously obtained scalar elements in smaller source/sensor-specific batches of m_σ elements, one can isolate sensor-specific errors and thus improve the estimates. Again, if $1 \leq m_\sigma < C$, the filter still embodies the incremental SCAAT idea.

In section 6.2.5 on page 144, I present experiments that demonstrate improved performance when a batch of simultaneously obtained scalar elements are processed sequentially, a single constraint at a time.

Criteria for Choosing the Measurements

So given the preceding discussion, what criteria should one use when choosing the measurement elements or constraints to incorporate during a SCAAT Kalman filter measurement update? Given the complete set of available sources and sensors for the system, I suggest the following guidelines:

a. begin by identifying the set of all sensor and corresponding source elements that can possibly be observed at any one instant in time;

b. within this set identify any single source and sensor pairs that should be isolated from the others;

c. for each such pair identify the m_σ scalar elements yielded from a measurement of the sensor with the corresponding source; and

d. apply these guidelines until all the available sources and sensors have been considered.

Consider, for example, tracking two rigidly mounted 2D cameras that can observe four fixed beacons or scene points, as depicted for one camera in figure 1.4 on page 39. The combined use of two sensors (the 2D cameras) and four sources (the beacons) yields a total of 16 scalar measurement elements for the complete set of sources and sensors: a (u, v) pair for each of four beacons as seen by each of two cameras. If the two cameras cannot be shuttered and scanned-out simultaneously then guideline (a) would reduce the original set to two new sets, each with one 2D camera and four beacons. If one is uncertain about the 3D locations of the beacons, and/or wishes to calibrate (estimate) the positions concurrently while tracking, then guideline (b) would break these two sets into eight sets, each with one 2D camera and one beacon. Finally, per guideline (c) one would note that the eight camera-beacon pairs each yield a (u, v) image coordinate, i.e. $m_\sigma = 2$ scalar elements. If there were more source and sensor types, one would repeat this process per guideline (d).

In light of the device isolation discussion in section 2.5.4 on page 61, the application of the above guidelines in the general case leads to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

> *During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.*

Thus for the two-camera, four-beacon example, we could have immediately determined that each SCAAT Kalman filter measurement update should incorporate the (u, v) image coordinate of one beacon as seen in one camera. Each such observation could in fact be considered a single geometric constraint: the intersection of a line, the line from the beacon to the principal point of the camera lens, and a plane, the image plane.

Predicting the Measurement from the State

Applying the previous heuristic, for each physical sensor of type σ we identify the m_σ-dimensional *measurement function* $\vec{h}_\sigma(\bullet)$ and the corresponding *measurement estimate vector* $\hat{z}_\sigma(t)$ as in equation (4.10).

$$\hat{z}_\sigma(t) = \vec{h}_\sigma(\vec{x}(t), \vec{\alpha}(t), \vec{b}(t), \vec{c}(t)) + \vec{v}_\sigma(t) \tag{4.10}$$

Each possibly nonlinear measurement function $\vec{h}_\sigma(\bullet)$ predicts the ideal noise-free response of a sensor of type σ given the complete target state $\vec{x}(t)$ and $\vec{\alpha}(t)$, along with the *device parameter vectors* $\vec{b}(t)$ and $\vec{c}(t)$ that describe the source and sensor respectively. One can think of the “b” and “c” in $\vec{b}(t)$ and $\vec{c}(t)$ as referring to *beacon* and *camera*, the example source and sensor pair discussed above, illustrated in figure 4.4 on page 86, and later presented in chapter 6.

4.2.5 The Measurement Noise

The m_σ-dimensional *measurement noise vector* $\vec{v}_\sigma(t)$ in equation (4.10) is a normally-distributed zero-mean sequence that reflects the actual measurement uncertainty due to random error such as electrical noise. Like the process noise vector, the elements of the measurement noise vector are considered to be uncorrelated over time but to have an instantaneous non trivial correlation. This instantaneous correlation is reflected in the corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* $R_\sigma(t)$ of equation (4.11), where again $E\{\bullet\}$ denotes the expected value. Again note that "covariance" is used here in the statistical sense.

$$E\{\vec{v}_\sigma(t)\vec{v}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{4.11}$$

Note that unlike the process noise, the measurement uncertainty is associated with a particular instant in time, i.e. the instant of measurement, rather than an interval over time. Thus while the process noise covariance matrix is a function of a time interval δt, the measurement noise covariance matrix is a function of a particular instant in time t. The individual variance elements of $R_\sigma(t)$ can be estimated from off-line measurements using the actual sources and sensors, or they can be estimated in real time, perhaps based on the current state and perhaps making use of knowledge from previous off-line measurements. The off-diagonal elements of $R_\sigma(t)$ may be non-zero as further discussed under "Added Measurement Uncertainty" on page 90. Similar to the matrix $Q(\delta t)$ in equation (4.6), $R_\sigma(t)$ describes a multivariate normal probability density. In this case it reflects the uncertainty of the *actual* measurement $\vec{z}_\sigma(t)$ relative to the predicted measurement $\hat{z}_\sigma(t)$ of equation (4.10). As such, the multivariate density associated with $R_\sigma(t)$ exists in the measurement space defined by the physical sensor. An actual density for the tracking system simulated in chapter 6 is given in figure C.4 on page 188.

4.2.6 The Measurement Jacobian

The Sensitivity of the Measurement

For each possible measurement function $\vec{h}_\sigma(\bullet)$ we identify a corresponding *measurement Jacobian*

$$H_\sigma(\vec{x}(t), \vec{\alpha}(t), \vec{b}(t), \vec{c}(t))[i, j] \equiv \frac{\partial}{\partial \vec{x}[j]} \vec{h}_\sigma(\vec{x}(t), \vec{\alpha}(t), \vec{b}(t), \vec{c}(t))[i], \tag{4.12}$$

where $1 \le i \le m_\sigma$, the size of the measurement vector in equation (4.10), and $1 \le j \le n$, the size of the state vector in equation (4.2). Figure 4.5 on page 87 illustrates the Jacobian for the example image-based tracking system used throughout this dissertation.

4.2.7 The State Error Covariance

The Filter's Estimate of its Uncertainty

Finally, I note the use of the standard Kalman filter $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state:

$$P(t) = E\{\hat{\xi}(t)\hat{\xi}^T(t)\}, \tag{4.13}$$

where $\hat{\xi}(t) = \vec{x}(t) - \hat{x}(t)$ is the error in the filter's estimate of the state; $\vec{x}(t)$ is the true filter state; and $\hat{x}(t)$ is the filter's estimate of the state. In the actual KF implementation as described below, $P(t)$ reflects the *filter's estimate* of its own uncertainty. In practice this may or may not be the same as the *actual* error. A multivariate density corresponding to the error covariance matrix is depicted in figure 4.3 on page 85 for a subset of the elements in equation (4.2).

4.3 The SCAAT Algorithm for Tracking

In this section I present the complete set of steps necessary to generate an updated SCAAT filter estimate. While doing so I also describe, from a *geometric* point of view, what is happening at each step. It is my hope that this geometric interpretation will leave the reader with a better feeling for what occurs when a new state estimate is formed from a locally unobservable or under-constrained measurement.

While this section is not meant as a tutorial for the Kalman filter in general, it should also shed some light on the operation of a conventional implementation, i.e. a well-constrained implementation. While the circumstances are different, the basic update steps are the same. This should be especially evident in the algorithm summary of section 4.3.7 for the reader who is familiar with the conventional Kalman filter.

4.3.1 Time Update

As with the standard Kalman filter, we assume the availability of previous estimates for $\hat{x}(t - \delta t)$, $\hat{\alpha}(t - \delta t)$, and $P(t - \delta t)$. Then for a new measurement $\vec{z}_\sigma(t)$ at time t. from some sensor of type σ and the corresponding source, we first compute the elapsed time δt

since the previous estimate, and then predict the new state and error covariance. This is often called the Kalman filter *time update step*.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t - \delta t) \\ P^- &= A(\delta t)P(t - \delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{4.14}$$

During the time update step both the state $\hat{x}$ and the error covariance P are extrapolated forward from the previous estimate to the current time. Because the results represent the best estimates of each just prior to incorporation of the new measurement in the subsequent steps, they are frequently called the *a priori* state and error covariance, and are distinguished from their *a posteriori* (post measurement correction) counterparts with a super minus.

Given the dynamic model described in section 4.2.1, the extrapolation makes use of the derivative information in the state to integrate both $\hat{x}$ and P forward in time as in equation (4.4). As such, $\hat{x}^-$ is a slightly moved and reoriented version of $\hat{x}$ as shown in figure 4.2. Recall that in terms of orientation, the state $\hat{x}$ maintains only the incremental rotation, thus the orientation elements are always zero for $\hat{x}$, and often non-zero for $\hat{x}^-$.

When P is extrapolated forward in time and augmented by $Q(\delta t)$, the error covariance does not move per se, but its uncertainty grows as depicted in figure 4.3. The error covariance can also be re-oriented in the process but this is not shown in figure 4.3. The addition of the process noise covariance matrix $Q(\delta t) \neq 0$ introduces a fixed amount of uncertainty, ensuring $P^- \neq 0$ which actually improves stability by preventing the filter from reaching a state where it ignores future measurements. Complete conditions for stability are discussed in section 5.1.3.

4.3.2 Measurement Prediction

After the time update step, we begin what is often called the *measurement update step* by predicting the measurement and computing the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \hat{\alpha}(t), \vec{b}(t), \vec{c}(t)) \\ H &= H_\sigma(\hat{x}^-, \hat{\alpha}(t), \vec{b}(t), \vec{c}(t)) \end{aligned} \tag{4.15}$$

To best illustrate the results of this step I have chosen to employ a concrete example of a particular measurement system. Specifically I have chosen to continue with the image-

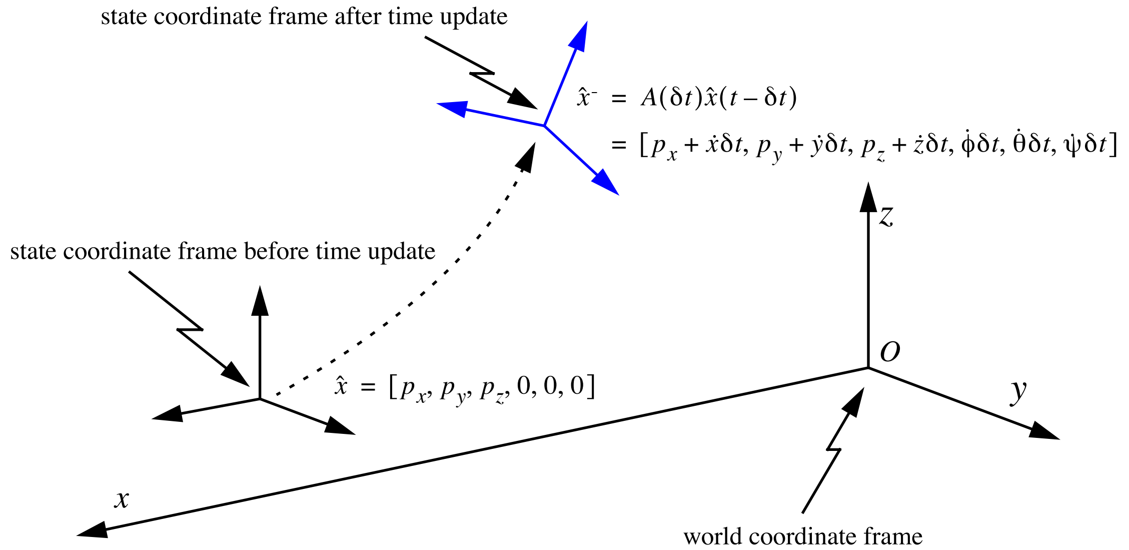

state coordinate frame after time update
$\hat{x}^{-} = A(\delta t)\hat{x}(t-\delta t)$
$= [p_x + \dot{x}\delta t, p_y + \dot{y}\delta t, p_z + \dot{z}\delta t, \dot{\phi}\delta t, \dot{\theta}\delta t, \dot{\psi}\delta t]$
z
state coordinate frame before time update
O
$\hat{x} = [p_x, p_y, p_z, 0, 0, 0]$
y
x
world coordinate frame

Figure 4.2: Geometric view of state change during time update step. Per the derivatives in the state, the target is predicted to move and reorient since the time of the last filter estimate. Note that the target rotation is maintained incrementally in the state so the state orientation is zero before the time update and non-zero afterwards. The derivative elements of the state have been omitted and the spacial relationship exaggerated for clarity.

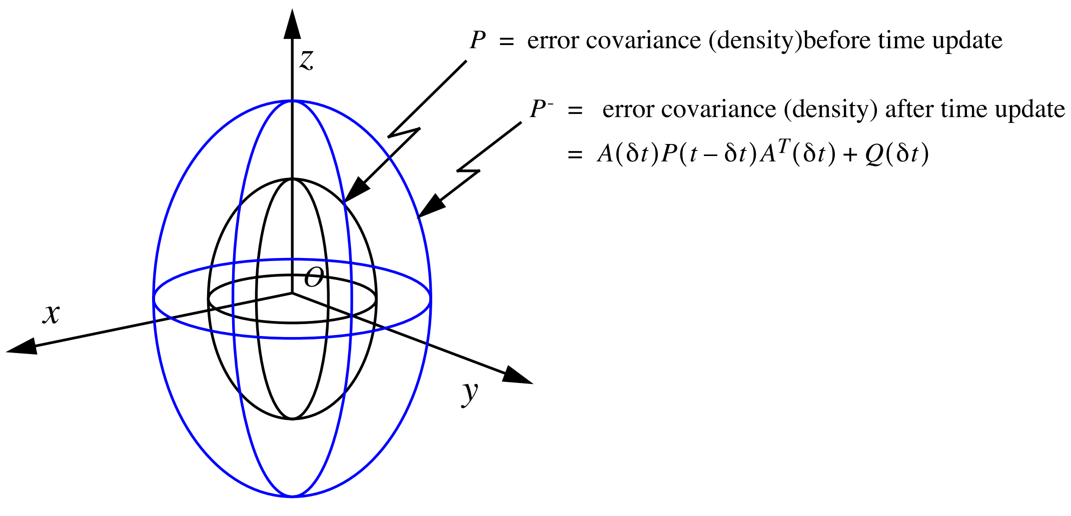

P = error covariance (density)before time update
z
P^{-} = error covariance (density) after time update
$= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t)$
O
x
y

Figure 4.3: Geometric view of error covariance change during time update step. The change is shown by a growing probability density for each of *x*, *y*, and *z*. Note that reorientation is possible in general, but not for the dynamic model given in section 4.2.1. To visualize with Euclidean dimensions, the density has been limited to 3D: *x*, *y*, and *z* only. The shape, magnitude, and orientation are illustrative only.

based example originally introduced in section 1.2. (See figures 1.4-1.6 beginning on page 39.) This choice has the added benefit that it is similar to the system presented in chapter 6.

Predicting the Measurement

It is important to emphasize that the Kalman filter measurement function $\vec{h}_\sigma(\bullet)$ computes what the system *would* measure given a certain state and the associated parameters. With respect to VE tracking, this notion was captured elegantly by my advisor, Dr. Gary Bishop, when observing that unlike conventional position recovery algorithms,

> *"Nowhere do you compute where you are from what you see, you only compute what you would see from where you [think you] are."*

This notion is closely related to the idea that rather than attempting to compute precise closed-form estimates of *points* in state space, the SCAAT method instead pushes the current state estimate along the *track* most consistent with the most recent incomplete observation. (See section 1.3, "An Unusual Approach to Tracking".) Figure 4.4 illustrates the relationship between the camera, the scene point, and the measurement $\hat{z}$ for the hypothetical image-based system.

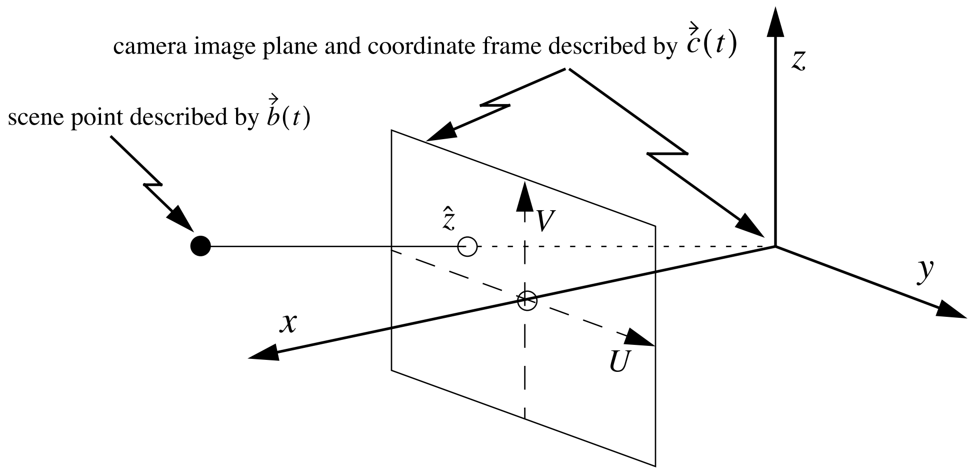

camera image plane and coordinate frame described by $\vec{c}(t)$
scene point described by $\vec{b}(t)$
z
y
x
$\hat{z}$
V
U

Figure 4.4: Measurement prediction from the state and associated parameters. The measurement function $h_\sigma(\bullet)$ is used to predict the measurement $\hat{z}$ of the sensor type σ, a 2D camera in this example.

For this system, the two scalar elements of $\hat{z}$ would be computed by using $\hat{x}^-$ and $\hat{\alpha}(t)$ to translate and rotate the scene point described by $\vec{b}(t)$ into the target coordinate system, then using $\vec{c}(t)$ to translate and rotate the point into the camera coordinate system, and finally using $\vec{c}(t)$ again to project the point onto the image plane.

Jacobian of the Measurement Function

Figure 4.5 illustrates the meaning of the Jacobian function $H_\sigma(\bullet) \equiv \partial \overrightarrow{h_\sigma}(\bullet)/\partial \hat{x}$. By relating small changes or error in the state to corresponding changes in the measurement (function), the Jacobian reflects the direction and magnitude of information or "correction" provided by the measurement. Notice in this case that the measurement only provides information along directions orthogonal to the line of sight to the scene point.

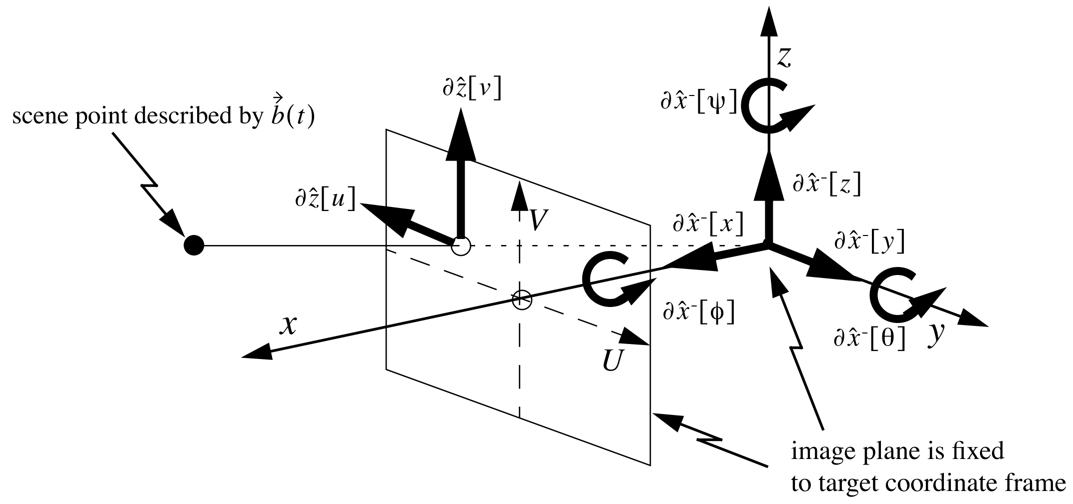


Figure 4.5: The measurement Jacobian $H = H_\sigma(\bullet)$. The Jacobian matrix H reflects the change in the measurement $\hat{z}$ with respect to the state $\hat{x}^-$.

4.3.3 The Kalman Gain

We next compute what is usually called the *Kalman gain*.

$$K = P^- H^T (H P^- H^T + R_\sigma(t))^{-1} \tag{4.16}$$

The Kalman gain *K* is used in equation (4.20) below to weight the difference or *residual* between the prediction and the actual noisy measurement when correcting the state and error covariance prediction. The formulation for the gain in equation (4.16) is a result of the derivation of the Kalman filter [Kalman60, Brown92]. Under the proper conditions, the Kalman gain provides optimal estimation in the sense that the expected squared state error is minimized.

The Kalman Gain as a Ratio

Through inspection of equation (4.16) we see that the gain is of the form of a ratio involving the *a priori* state error covariance P^- from equation (4.14) and the presumed known measurement error covariance R_σ. Let us represent this ratio informally as

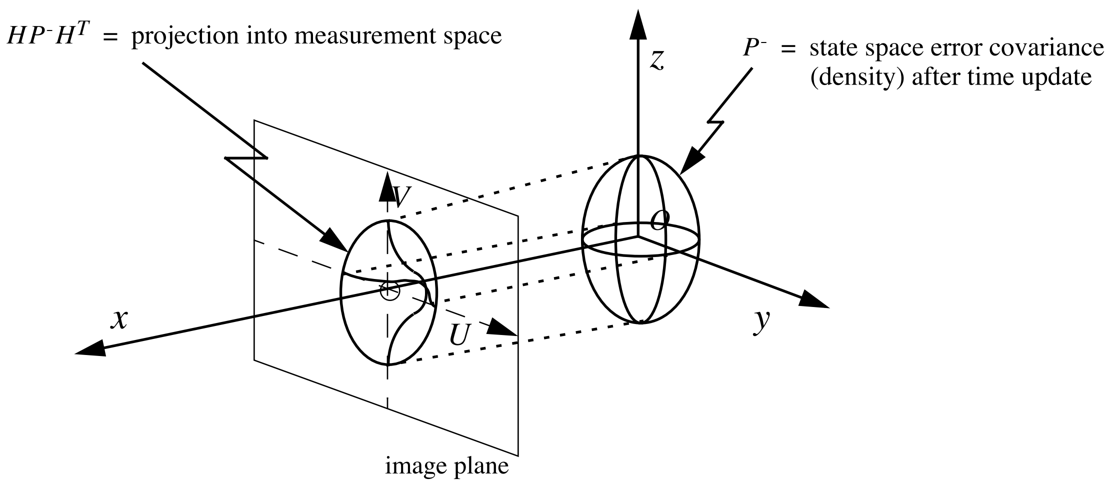
$$K \approx \frac{P^-}{P^- + R_\sigma}. \tag{4.17}$$

Note that equation (4.17) is not proper in that P and R_σ exist in different multi-dimensional spaces, and must be transformed by H. The ratio is used here only to illustrate the spirit of one aspect of the actual Kalman gain of equation (4.16).

From State Space to Measurement Space

The first step in actually forming the Kalman gain equation (4.16) is to compute the product HP^-H^T. While not reflected in the informal ratio of equation (4.17), the Jacobian H plays two crucial roles in computing this product (and in the overall Kalman gain computation). The first role is to transform the state error covariance P^- from state space into measurement space so that it can be combined with the measurement error covariance R_σ to form the denominator of the informal representation equation (4.17). Continuing with the image-based example, I illustrate this transformation for a limited set of state elements in figure 4.6. An actual projected density HP^-H^T from one of the simulations of chapter 6 is shown in figure C.4 of appendix C (page 188).

The second role of the Jacobian H is to approximate the nonlinear relationship between the state and the measurement. It can be illuminating to consider the product HP^-H^T in equation (4.16) in terms of the *error magnification* of the nonlinear function $h_\sigma(\bullet)$ in equation (4.10). Because $h_\sigma(\bullet)$ uses an *estimate* of the state to predict the measurement in equation (4.15), we would like to know how error in that state estimate propagates through it into the measurement prediction. If we let $\vec{\sigma}_{\vec{x}} = E\{\vec{x} - \hat{x}\}$ be the standard deviation of the state error, and $\vec{\sigma}_{\vec{z}} = E\{\vec{z} - \hat{z}\}$ be the standard deviation of the



HP^-H^T = projection into measurement space
z
P^- = state space error covariance (density) after time update
V
O
x
U
y
image plane

Figure 4.6: Project state-space uncertainty into measurement space. The Jacobian matrix H, which represents change in the measurement with respect to each state, is used to determine the filter uncertainty in measurement space (e.g. image plane) given P^-, the uncertainty in state space. As was the case with figure 4.3, the density has been limited to 3D, the shape, magnitude, and orientation are illustrative only.

measurement error, we can work backwards to arrive at the measurement error covariance corresponding to the state error covariance:

$$\begin{aligned}
(\vec{\sigma}_{\vec{z}})(\vec{\sigma}_{\vec{z}})^T &= \left(\frac{\partial}{\partial \vec{x}}\vec{h}_\sigma(\bullet) \cdot \vec{\sigma}_{\vec{x}}\right)\left(\frac{\partial}{\partial \vec{x}}\vec{h}_\sigma(\bullet) \cdot \vec{\sigma}_{\vec{x}}\right)^T \\
&= (H \cdot \vec{\sigma}_{\vec{x}})(H \cdot \vec{\sigma}_{\vec{x}})^T \\
&= (H \cdot \vec{\sigma}_{\vec{x}})(\vec{\sigma}_{\vec{x}}^T \cdot H^T) \\
&= HPH^T.
\end{aligned}$$

While in theory the nonlinear approximation using H is not optimal, in practice the EKF often provides satisfactory results. (See the Kalman filter references presented at the beginning of chapter 4 for more explanation and justification.) Recently Julier and Uhlmann [Julier95] presented a new and very interesting method for approximating nonlinear transformations of probability distributions. They claim that the method is more accurate, more stable, and easier to implement than an EKF. The potential application of their work is discussed in chapter 7.

Added Measurement Uncertainty

The next step in computing the Kalman gain equation (4.16) is to add the transformed state error covariance HP^-H^T to the measurement error covariance R_σ. This augmenting of the transformed state error covariance is depicted in figure 4.7. Figure C.4 of appendix C (page 188) shows actual simulated densities for both HP^-H^T and $HP^-H^T + R_\sigma$. The inversion of the sum $HP^-H^T + R_\sigma$ in equation (4.16) completes the denominator of the informal gain ratio of equation (4.17).

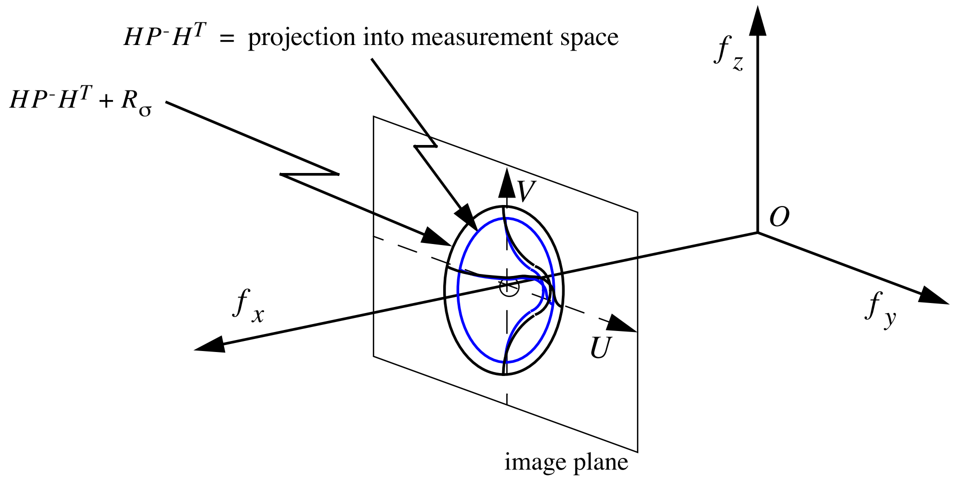


Figure 4.7: Completing the denominator of the Kalman gain. The measurement error covariance R_σ is added to the transformed state error covariance HP^-H^T.

Note that if R_σ is of dimension greater than one, as in this example where it is two, it is not necessarily the case that the off-diagonal elements would be zero. As discussed under "A Single Constraint" on page 37, there may exist some known correlation between the elements of the measurement vector. If this is the case, then both the shape and the orientation of the final density in figure 4.7 can also change.

Back to State Space

The next-to-final step in computing the Kalman gain equation (4.16) is to transform the $m_\sigma \times m_\sigma$ augmented and inverted covariance $(HP^-H^T + R_\sigma(t))^{-1}$ back into state space by pre-multiplying it with the $n \times m_\sigma$ transposed Jacobian H^T as in equation (4.18).

$$H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{4.18}$$

The result of equation (4.18) is effectively the completed denominator of equation (4.17).

The error magnification viewpoint discussed above is applicable here: H^T reflects how the augmented and inverted measurement-space uncertainty $(HP^-H^T + R_\sigma(t))^{-1}$ propagates back into state space. And while the $n \times m_\sigma$ result of equation (4.18) is not square (much less symmetric, assuming $n \neq m_\sigma$ as can be expected for the SCAAT approach) the overall operation can be considered somewhat the reverse of that depicted in figure 4.6.

The Final Gain

The final step in computing the Kalman gain equation (4.16) is to complete the ratio in equation (4.17) by multiplying P^- by the denominator equation (4.18). The final result of the expression equation (4.16) is the $n \times m_\sigma$ Kalman gain matrix K which is used to optimally weight the measurement residual $\overrightarrow{\Delta z}$ in equation (4.19) discussed in the next section.[2] In fact, the Kalman gain can also be viewed from the error magnification perspective. From this viewpoint, the error is the measurement residual $\overrightarrow{\Delta z}$, and the Kalman gain controls how the error (residual) propagates into state space.

It is important to note that even if the Jacobian H indicates a complete lack of information along a particular direction in state space, the final Kalman gain may actually indicate a non-zero gain along that same direction! Mathematically, this is because the final operation in the gain computation is the product of P^- and the denominator equation (4.18). In particular, each element of the gain matrix is a linear combination of a row of P^- and a column of equation (4.18). Therefore if P^- has non-zero off-diagonal elements, then an otherwise zero gain element can become non-zero. Intuitively, P^- having non-zero off-diagonal elements implies a correlation between two or more states. Thus even if the current measurement provides no information along a particular direction, a necessary correction may be implied by a correlation indicated in P^-.

2. Optimality is as discussed in section 2.1 beginning on page 44.

Finally, because the measurement residual $\overrightarrow{\Delta z}$ (described next in section 4.3.4) has units [measurement units] and the state has units [state units], we would expect the final Kalman gain K to have units

$$\left[\frac{\text{state units}}{\text{measurement units}}\right],$$

since the purpose of the product $K\overrightarrow{\Delta z}$ in equation (4.20), presented in the next section, is to correct the state with the measurement information. Indeed the final result of the product $K\overrightarrow{\Delta z}$ in equation (4.20) has the appropriate units [state units]. With respect to correcting (updating) the error covariance in equation (4.20), because the Jacobian H has units

$$\left[\frac{\text{measurement units}}{\text{state units}}\right],$$

the product KH in equation (4.20) is unitless, and the error covariance units remain unchanged.

4.3.4 The Measurement Residual

The purpose for computing the preceding Kalman gain is to determine the proper weighting for the measurement innovation or *residual* equation (4.19).

$$\overrightarrow{\Delta z} = \vec{z}_\sigma(t) - \hat{z} \tag{4.19}$$

This m_σ-dimensional residual $\overrightarrow{\Delta z}$ reflects the error in the measurement prediction $\hat{z}$ of equation (4.15) with respect to the actual (noisy) sensor measurement $\vec{z}_\sigma(t)$. As such the residual consists of both prediction and measurement error, which is why the Kalman gain of equation (4.16) is a function of both the estimate and measurement covariance.

Continuing with the image-based measurement example, the situation is depicted in figure 4.8. The filter has some notion of where the target is located and how it is oriented, which determines its prediction of the measurement $\overrightarrow{\Delta z}$ (the image coordinates of the known scene point). But the actual measurement $\vec{z}_\sigma(t)$ indicates what the camera really “saw”. The difference between the two is the residual—the error in the measurement prediction.

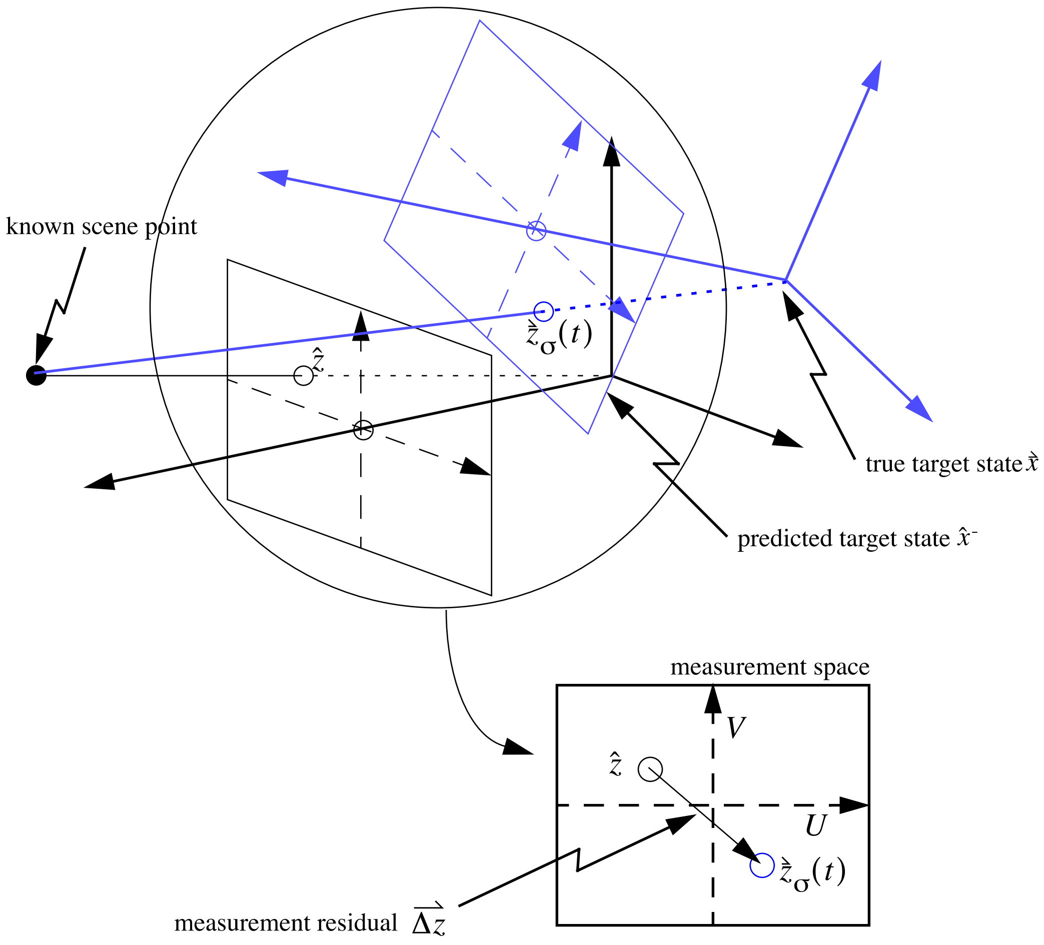

known scene point
$\hat{z}$
$\vec{z}_\sigma(t)$
true target state $\vec{x}$
predicted target state $\hat{x}^-$
measurement space
V
U
$\hat{z}$
$\vec{z}_\sigma(t)$
measurement residual $\overrightarrow{\Delta z}$

Figure 4.8: The measurement residual. Continuing with the image-based measurement example, the residual is the m_σ-dimensional vector from the predicted measurement (image-plane coordinates) $\hat{z}$ to the actual measurement $\vec{z}_\sigma(t)$. The measurement prediction is what the filter thinks it will see given the predicted state, the actual measurement comes directly from the camera. (The relationship between the true and predicted state has been exaggerated for the purpose of illustration.)

Note that the Kalman filter gain equation equation (4.16) is based on the assumption that the sequence of residuals will be “white” in nature and normally distributed over some period of time. If the residuals are not white or normally distributed, this could be an indication that model for the target dynamics is incorrect (see section 4.2.1) or that some component of the system has experienced a failure. Thus ad hoc residual monitoring schemes are sometimes employed to detect events such as devices failures or model transitions. (See for example [Maybeck70], and also section 7.3.)

4.3.5 Correct the State and Update the Error Covariance

The final step in the Kalman filter update is to obtain the *a posteriori* state and error covariance estimates $\hat{x}(t)$ and $P(t)$, i.e. to "correct" the *a priori* state estimate per the most recent measurement and to update the error covariance appropriately. The necessary operations are given in equation (4.20).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\overrightarrow{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{4.20}$$

These operations are illustrated in figures 4.9 and 4.10, which can be compared with their *a priori* counterparts depicted in figures 4.2 and 4.3 on page 85.

Both operations of equation (4.20) make the use of the Kalman filter gain equation (4.16) to effect their change. And while the operation to form the *a posteriori* state estimate $\hat{x}(t)$ should seem somewhat intuitive given the preceding explanations, the operation to form the *a posteriori* error covariance $P(t)$ is probably less so. Its origin is in the Kalman gain derivation which can be found in the various Kalman filter books, e.g. [Brown92]. Informally, its purpose is to reflect the increased knowledge provided by the measurement. As such, one would expect the corresponding densities to become smaller and smaller over time as more and more measurements are incorporated into the estimates. For a process driven by no noise, i.e. $Q(\delta t) = [0]$ in equation (4.6) and equation (4.14), this is indeed the case: $P(t)$ will tend to zero (the null matrix) as more measurements are incorporated. However for purposes of stability as discussed in section 5.1 beginning on page 107, we employ a model that is driven by non-zero noise. As such, in a sense $P(t)$ approaches $Q(\delta t)$ in magnitude over time, and is continually being reoriented to reflect the information (or lack thereof) provided by the most recent measurement. If all of the measurements provide approximately the same magnitude of information, one can visualize the density as "spinning" about the origin in response to the stream of incomplete measurements. A real example of this reorientation can be seen in appendix C beginning on page 184.

Referring back to figure 4.8 on page 93, two points warrant repetition. The first point is that in a SCAAT implementation the information provided by the residual $\overrightarrow{\Delta z}$ (which is based on the measurement) is insufficient to completely correct the filter's *a*

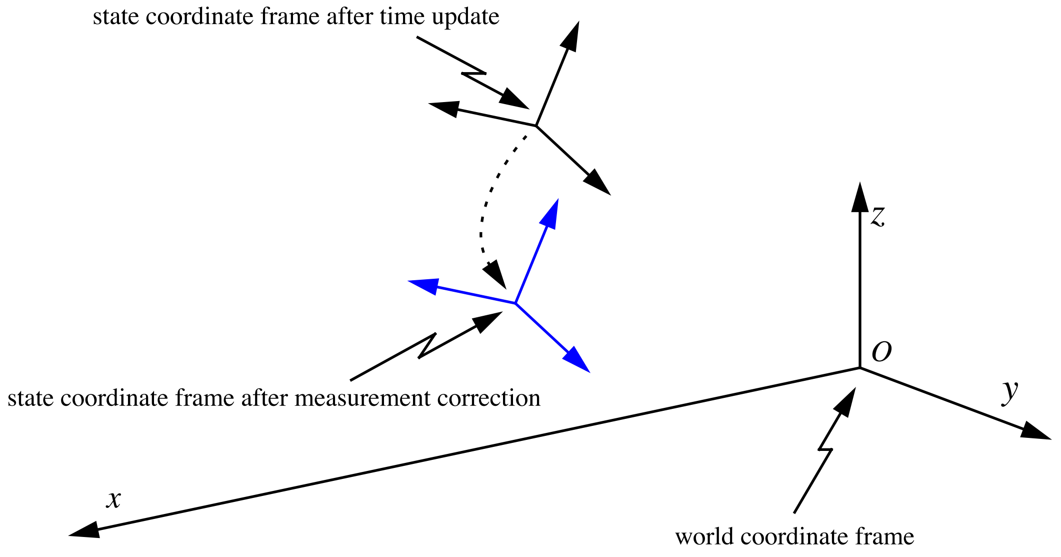

state coordinate frame after time update
z
O
y
state coordinate frame after measurement correction
x
world coordinate frame

Figure 4.9: Geometric view of state change after measurement correction. Per the information in the measurement residual, the *a posteriori* state estimate is formed, i.e. the estimated target coordinate frame is moved and reoriented. Compare this with figure 4.2 on page 85. (Again the derivative elements of the state have been omitted and the spacial relationship exaggerated for clarity.)

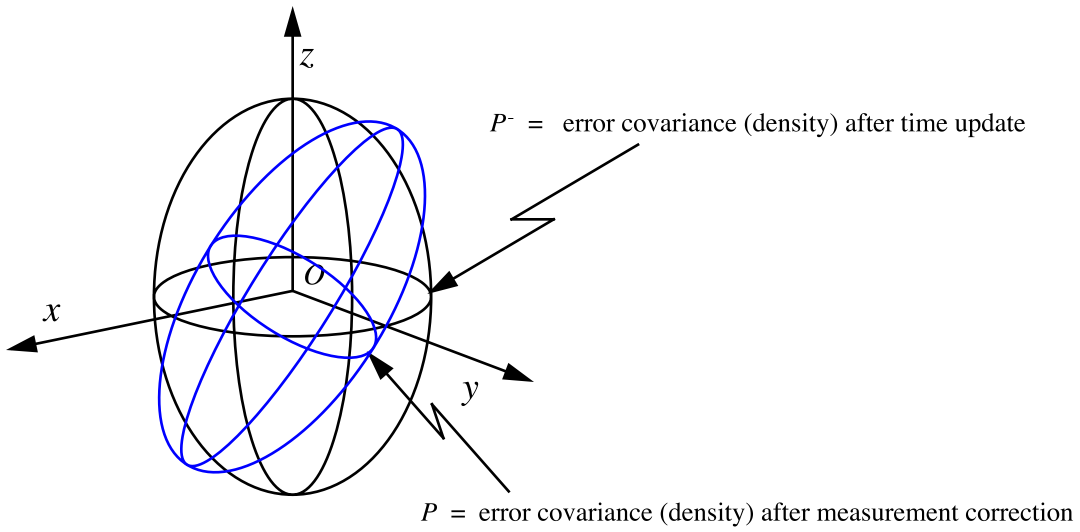

z
P^{-} = error covariance (density) after time update
O
x
y
P = error covariance (density) after measurement correction

Figure 4.10: Geometric view of error covariance change after measurement update. The change is shown by a refined probability density for each of x, y, and z. Note that in general the density is reoriented to reflect the constraint provided by the measurement. Compare this with figure 4.3 on page 85. (Again to visualize with Euclidean dimensions, the density has been limited to 3D: x, y, and z only. The absolute shape, magnitude, and orientation are illustrative only.)

priori state estimate. For this image-based example, the constraint provided by the measurement is only two-dimensional, but the target position and orientation is six-dimensional. As stated in section 1.3 on page 42, rather than attempting to compute precise closed-form estimates of *points* in state space, a SCAAT filter instead "pushes" the current state estimate along the *track* indicated by the most recent (incomplete) observation as indicated by the dashed arc in figure 4.9. A second related point worth repeating (with respect to the Kalman filter in general) is that the filter does not attempt to compute where the target is located and how it is oriented, it only computes what the sensors would "see" given the filter's estimate of where the target is located and how it is oriented.

It is interesting to point out that the *a posteriori* state and error covariance in equation (4.20) are both based on their *a priori* counterparts from equation (4.14), which are based on the *a posteriori* results from the previous filter update step. This recursive nature of the Kalman filter means that every estimate is conditioned by *all* of the previous measurements, and the initial conditions, but in a very computationally efficient manner.

4.3.6 Update External Orientation

One final bit of geometric "housekeeping" that must be performed is to update the external orientation and to (subsequently) zero the incremental orientation elements of the state.

$$\begin{aligned} \hat{\Delta\alpha} &= \text{quaternion}(\hat{x}[\Delta\phi], \hat{x}[\Delta\theta], \hat{x}[\Delta\psi]) \\ \hat{\alpha} &= \hat{\alpha} \otimes \hat{\Delta\alpha} \end{aligned} \tag{4.21}$$

$$\hat{x}[\Delta\phi] = \hat{x}[\Delta\theta] = \hat{x}[\Delta\psi] = 0 \tag{4.22}$$

Figure 4.11 depicts the complete sequence of internal filter state coordinate frame transitions throughout a single update. Recall that the incremental orientation elements in $\hat{x}(t)$ are zero *prior* to equation (4.14) as a result of equation (4.22) from the *previous* filter update. After equation (4.14) they reflect the velocity-based extrapolated incremental rotation, after equation (4.20) they reflect the measurement-corrected incremental rotation, and after equation (4.22) they are returned to zero, i.e. the target state estimate is returned to an "upright" orientation as depicted in figure 4.11. The entire process is then ready to begin again when the next measurement is available.

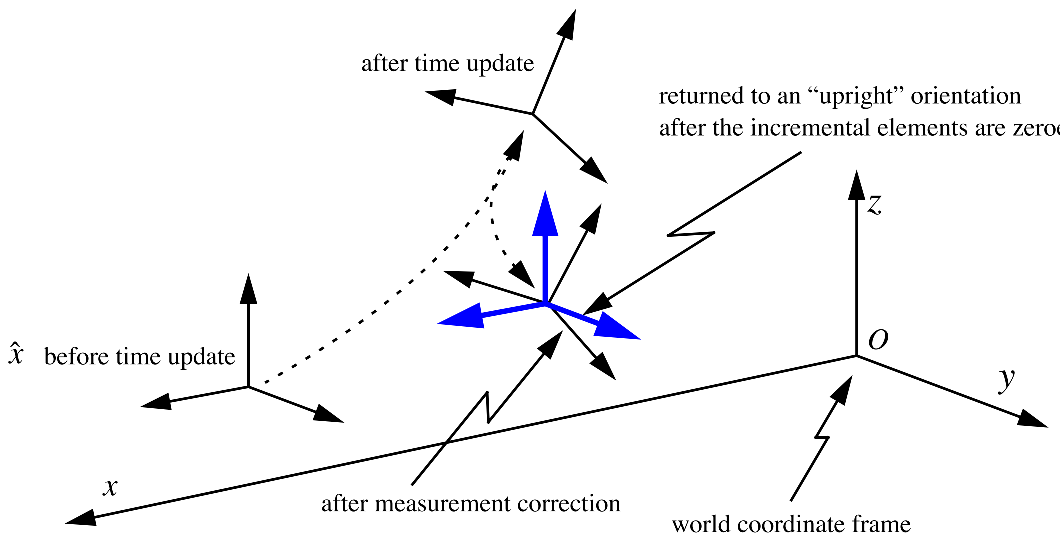

after time update
returned to an "upright" orientation
after the incremental elements are zeroed
z
$\hat{x}$ before time update
O
y
x
after measurement correction
world coordinate frame

Figure 4.11: Complete sequence of filter state coordinate frame transitions. This figure repeats the depictions in figures 4.2 and 4.9, and adds the final handling of the incremental rotations. After the measurement update, the incremental rotation is factored into the external orientation, and the incremental elements are zeroed in preparation for the next filter

4.3.7 The SCAAT Algorithm Summary

The following is a summary of the steps outlined in sections 4.3.1-4.3.6. Each step is identified with an alphabetic step identifier for later reference, and the original equation number from sections 4.3.1-4.3.6.

Given an initial state estimate $\hat{x}(0)$, external orientation estimate $\hat{\alpha}(0)$, and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF. For each measurement $\vec{z}_{\sigma}(t)$ at time t. from some sensor of type σ and corresponding source, we cycle through the following update steps:

a. Compute the elapsed time δt since the previous estimate, predict the state and error covariance (the Kalman filter *time update step*).

$$\begin{aligned} \hat{x}^{-} &= A(\delta t)\hat{x}(t-\delta t) \\ P^{-} &= A(\delta t)P(t-\delta t)A^{T}(\delta t) + Q(\delta t) \end{aligned} \tag{4.14}$$

b. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \hat{\alpha}(t), \vec{b}(t), \vec{c}(t)) \\ H &= H_\sigma(\hat{x}^-, \hat{\alpha}(t), \vec{b}(t), \vec{c}(t)) \end{aligned} \quad (4.15)$$

c. Compute the *Kalman gain*.

$$K = P^- H^T (H P^- H^T + R_\sigma(t))^{-1} \quad (4.16)$$

d. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,\, t}$ and the predicted measurement from equation (4.15).

$$\overrightarrow{\Delta z} = \vec{z}_\sigma(t) - \hat{z} \quad (4.19)$$

e. Correct the predicted tracker state estimate and error covariance from equation (4.14).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\overrightarrow{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \quad (4.20)$$

f. Update the external orientation equation (4.3) per the change indicated by the $(\Delta\phi, \Delta\theta, \Delta\psi)$ elements of the state vector.[3]

$$\begin{aligned} \hat{\Delta\alpha} &= \text{quaternion}(\hat{x}[\Delta\phi], \hat{x}[\Delta\theta], \hat{x}[\Delta\psi]) \\ \hat{\alpha} &= \hat{\alpha} \otimes \hat{\Delta\alpha} \end{aligned} \quad (4.21)$$

g. Zero the orientation elements of the state vector.

$$\hat{x}[\Delta\phi] = \hat{x}[\Delta\theta] = \hat{x}[\Delta\psi] = 0 \quad (4.22)$$

4.3.8 Discussion

Following up on the discussion of section 1.2 on page 36, the key to the SCAAT method is the set of constraints provided by the measurement vector and measurement function in equation (4.10). In the general case for the problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because

3. The operation $\alpha \otimes \Delta\alpha$ indicates a quaternion multiply [Chou92].

individual sensor measurements typically provide less than six constraints, conventional implementations generally construct a complete measurement vector (or its equivalent)

$$\vec{z}(t) = \left[\vec{z}^T_{\sigma_1}(t_1)\ \dots\ \vec{z}^T_{\sigma_N}(t_N)\right]^T \tag{4.23}$$

from some set of $N \geq C$ sequential sensor measurements over time $t_1 \dots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. (See section E.3.3 of appendix E, page 196.) In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual incomplete observations over time to provide a globally observable linear system. It does this by computing an updated estimate using the steps in section 4.3 for each new underconstrained measurement as it becomes available. The EKF inherently provides the means for this blending, independent of the completeness of the information content of each individual measurement $\vec{z}_{\sigma_k}(t_k)$ through use of the Kalman gain K in equation (4.16). Moreover, using the measurement function Jacobian of equation (4.12), the Kalman gain determines a weighting for the residual $\overrightarrow{\Delta z}$ in equation (4.19) based on the rate of change of each measurement with respect to the current state—an indicator of the information content of the individual measurement.

4.4 SCAAT Autocalibration

4.4.1 An Overview

The autocalibration approach I describe here involves augmenting the main tracker filter presented in section 4.2 to effectively implement a distinct *device filter* (Kalman filter) for each source or sensor to be calibrated.

In general, any constant device-related parameters used by a measurement function $\vec{h}_\sigma(\bullet)$ from equation (4.10) are candidates for this autocalibration method. As such I assume that the parameters to be estimated are contained in the device parameter vectors $\vec{b}(t)$ and $\vec{c}(t)$. I also present autocalibration of both the source and sensor—once it is understood how to include both, omission is straight-forward. Finally I add the following new notation.

$$\widehat{x} = \text{augmented matrix/vector (wide hat)}$$

4.4.2 Device Filters

As needed for each device (source or sensor), we create a distinct device filter as follows: Let $\vec{x}_\pi[i]$, $i = 0 \ldots n_\pi$, represent the corresponding device parameters.

a. Allocate an n_π dimensional state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.[4]

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix P_π, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

4.4.3 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 4.2, with the following exceptions. After step (a) in the original algorithm summary of section 4.3.7, we form augmented versions of the state vector

$$\widehat{x}(t-\delta t) = \begin{bmatrix} \hat{x}^T(t-\delta t) & \hat{x}_b^T(t-\delta t) & \hat{x}_c^T(t-\delta t) \end{bmatrix}^T, \tag{4.24}$$

the error covariance matrix

$$\widehat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_b(t-\delta t) & 0 \\ 0 & 0 & P_c(t-\delta t) \end{bmatrix}, \tag{4.25}$$

4. Theoretically there is no process noise in a system that estimates a *constant*. However in practice, a small amount of uncertainty prevents settling on incorrect values and allows ongoing correction. See the stability discussion in section 5.1.4 on page 111.

the state transition matrix[5]

$$\widehat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{4.26}$$

and the process noise matrix

$$\widehat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_b(\delta t) & 0 \\ 0 & 0 & Q_c(\delta t) \end{bmatrix}. \tag{4.27}$$

The method then follows steps (b)-(g) from the original algorithm summary of section 4.3.7, substituting the augmented variables, equations (4.24)-(4.27), and noting that the measurement and Jacobian functions used in equation (4.15) have become $\vec{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\vec{b}(t)$ and $\vec{c}(t)$ ($\hat{x}_b$ and $\hat{x}_c$) are now contained in the augmented state vector $\widehat{x}$ per equation (4.24). After step (g) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_b(t) &= \widehat{x}(t)[i\ldots j] \\ P_b(t) &= \widehat{P}(t)[i\ldots j, i\ldots j] \\ \hat{x}_c(t) &= \widehat{x}(t)[k\ldots l] \\ P_c(t) &= \widehat{P}(t)[k\ldots l, k\ldots l] \end{aligned} \tag{4.28}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter respectively.

5. Remember that the device parameters are modeled as constants, hence the identity matrices in equation (4.26).

To summarize, we leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while swapping the appropriate device filter components in and out with each estimate as shown in figure 4.12. The result is that individual device filters are updated less frequently than the main tracker filter as depicted in figure 4.13. The more often a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

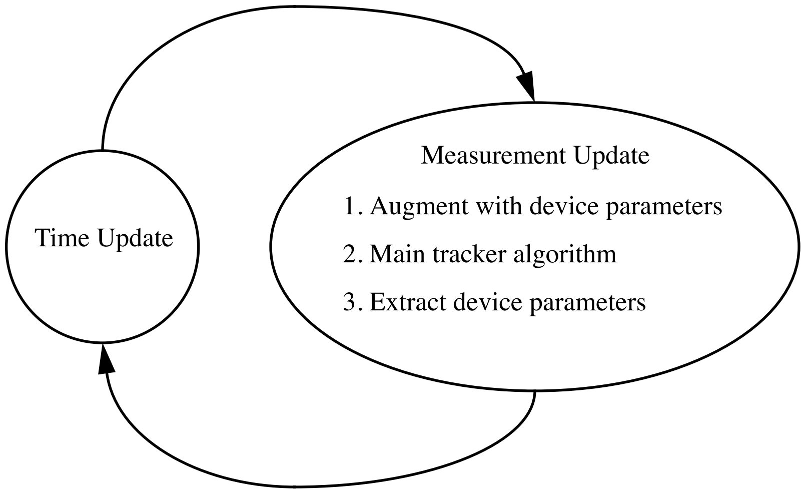

Time Update
Measurement Update
1. Augment with device parameters
2. Main tracker algorithm
3. Extract device parameters

Figure 4.12: The revised tracking algorithm for autocalibration. The time update consists of equation (4.14). The measurement update consists of equations (4.24)-(4.27), then (4.15)-(4.22), and finally equation (4.28).

4.4.4 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some time-stationary random acceleration, reflected in the noise parameters $\eta[i]$ of $Q(\delta t)$ in equation (4.7). Conversely, we assume the device parameters are constant, and so the elements of $Q_{\pi}(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or relatively small when compared to the elements of $Q(\delta t)$. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state equation (4.2). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous

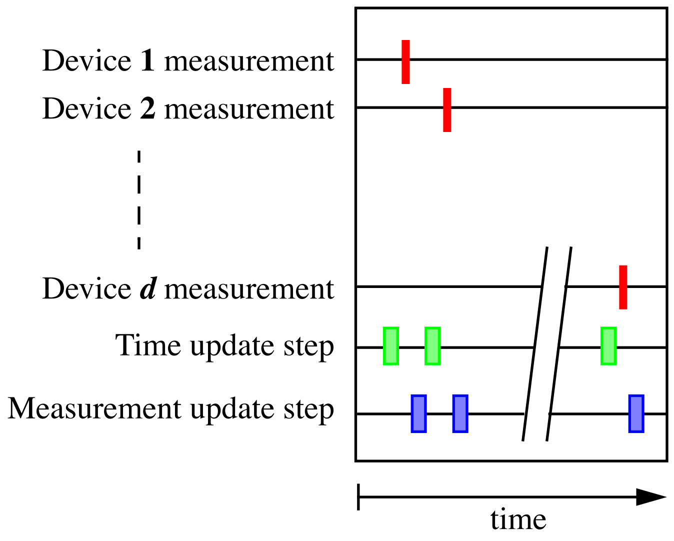


Figure 4.13: A timing diagram for autocalibration with ***d*** devices. Each device filter is updated only when it is used. The more often a device is used, the more it is calibrated. If a device is never used, it is never calibrated. The tracker filter is updated on *every* step.

target motion. The increased rate of the one-step method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

4.5 Code Optimization Strategies

Several optimizations are possible to speed the computations described earlier in this chapter, including modifications made for autocalibration as discussed in section 4.4. All of the following optimizations were implemented in the simulations described in chapter 6. Additional optimizations are suggested by [Maybeck70], p. 356.

4.5.1 Jacobian Evaluation

In step (b) of the algorithm (see section 4.3.7) the measurement Jacobian $H_\sigma(\bullet)$ of equation (4.15) must be computed in addition to the actual measurement prediction $\vec{h}_\sigma(\bullet)$. While the two sets of computations could certainly be carried out independently, it will likely make sense to compute the two together concurrently. It is likely that between the two sets of computations there are common sub-expressions that can be computed once and then used repeatedly as needed. While modern compilers generally perform common sub-expression optimizations automatically, these expressions should be examined manually because any common sub-expressions are not otherwise likely to be nearby in the code and thus not likely to be optimized automatically.

4.5.2 Sparse Matrices

The computations of equations (4.14), (4.16) and (4.20) can be optimized to eliminate unnecessary operations on matrices and vectors that are known to be sparse. In particular, certain elements of the Jacobian H may be zero because the corresponding state elements are not used in computing $\hat{z}$ in equation (4.15). Consider the simple case where a measurement arrives from a rate gyro, e.g. consider $\vec{z}_{\dot{\theta}}(t)$. In this case equation (4.15) becomes

$$\begin{aligned} \hat{z} &= \hat{x}[\dot{\theta}] \\ H &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

A more complicated but certainly relevant example is that of the UNC HiBall tracking system (simulated in chapter 6) in which the available measurements are from 2D cameras observing active beacons in the environment. In this case the velocity components of the state vector are not needed to compute the corresponding 2D measurement vector $\hat{z}$ in equation (4.15)—it is a purely geometric computation involving only the state elements $\hat{x}[x, y, z, \Delta\phi, \Delta\theta, \Delta\psi]$ and the external orientation $\vec{\alpha}$. Thus the Jacobian H is

$$H = \begin{bmatrix} \frac{\partial}{\partial x}\hat{z}[u] & \frac{\partial}{\partial y}\hat{z}u & \frac{\partial}{\partial z}\hat{z}[u] & 0 & 0 & 0 & \frac{\partial}{\partial \Delta\phi}\hat{z}[u] & \frac{\partial}{\partial \Delta\theta}\hat{z}[u] & \frac{\partial}{\partial \Delta\psi}\hat{z}[u] & 0 & 0 & 0 \\ \frac{\partial}{\partial x}\hat{z}[v] & \frac{\partial}{\partial y}\hat{z}[v] & \frac{\partial}{\partial z}\hat{z}[v] & 0 & 0 & 0 & \frac{\partial}{\partial \Delta\phi}\hat{z}[v] & \frac{\partial}{\partial \Delta\theta}\hat{z}[v] & \frac{\partial}{\partial \Delta\psi}\hat{z}[v] & 0 & 0 & 0 \end{bmatrix}$$

where $\hat{z}[u, v]$ represents the coordinates of the beacon as imaged by the 2D camera. (The $\hat{x}[\bullet]$ notation was omitted in the denominator of the partial derivatives for clarity.)

Clearly $A(\delta t)$ in equation (4.5) is sparse. Thus for example the matrix multiplies in equation (4.14) simplify to computations not unlike equation (4.4) for both $\hat{x}$ and P. Likewise it quite possible that R_σ in equation (4.11) is diagonal in which case computations in equation (4.16) can be simplified.

4.5.3 Symmetry

The matrices $Q(\delta t)$, R_σ and P are all required to be symmetric in the original derivation of the Kalman filter as indicated by equations (4.6), (4.11) and (4.13) respectively. Implementing equations (4.14), (4.16) and (4.20) to take advantage of the symmetry by copying instead of computing the upper (or lower) diagonal elements has the dual effect of reducing computation and ensuring exact symmetry.

4.5.4 Matrix Inversion

As discussed in chapter 2, one of the basic reasons for employing the SCAAT approach is to reduce the number of measurements used in computing a new estimate. In particular, the reduced size of the measurement vector, as compared to a well-constrained implementation, may allow the matrix inversion in equation (4.16) to be greatly simplified into a few simple closed-form computations. In the extreme, a one-dimensional measurement vector means that the inversion in equation (4.16) becomes a single scalar reciprocal.

4.5.5 Small Angles

The increased data rate of the SCAAT method means it's likely the $(\Delta\phi, \Delta\theta, \Delta\psi)$ elements of the state vector $\vec{x}(t)$ will be relatively small, thus allowing the use of the small angle approximations $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in the computations of $\hat{z}$ and H in equation (4.15), and $\Delta\hat{\alpha}$ in equation (4.21). While this optimization can significantly reduce the constants associated with the asymptotic time complexity by eliminating invocations of the respective trigonometric routines, it will in general introduce some non-white non-Gaussian noise into the system.

As it turns out, we implemented this optimization in the simulations of chapter 6 and have not perceived any problems. Perhaps this is because the noise in the residual of equation (4.19) is associated with a particular source/sensor combination and we are continually changing source/sensor pairs. Or perhaps other (white) system noise is more significant. In any case, under normal circumstances where there are sufficient constraints over time, the Kalman filter residuals from the experiments of chapter 6 appear to be unbiased (zero-mean) and white.

4.6 Source & Sensor Ordering Schemes

Beyond a simple round-robin approach, one might envision a measurement scheduling algorithm that makes better use of the available resources. In doing so one would like to be able to monitor and control uncertainty in the state vector. By periodically observing the eigenvalues and eigenvectors of the error covariance matrix $P(t)$, one can determine the directions in state-space along which more or less information is needed [Ham83]. This approach can be used to monitor the stability of the tracker, and to guide the source/sensor ordering.

The SCAAT method is uniquely positioned to make use of such information, since conventional (observable) Kalman filters don't have as much flexibility in picking and choosing what to measure. As such this approach is discussed as possible future work in section 7.5 on page 158.

Chapter 5. Mathematical Analysis

5.1 Filter Stability

5.1.1 General Conditions for Kalman Filter Stability

Possibly the most significant (and obvious) concern with respect to a SCAAT filter is its stability. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [Jacobs93]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [Gelb74], p. 132):

a. The filter must be uniformly completely observable,

b. the dynamic and measurement noise matrices in equations (4.6) and (4.11) must be bounded from above and below, and

c. the dynamic behavior represented by $A(\delta t)$ in equation (4.1) must be bounded from above.

As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation as presented in this section. Throughout this section I continue the notation from chapter 4.

5.1.2 Locally Unobservable Measurement Systems

A basic description of observability and its relationship to a sufficiently constrained linear system is presented in section 1.2 on page 36. More in-depth explanations can be found on pp. 178-188 of [Jacobs93] and pp. 43-48 of [Maybeck70].

The linear system associated with any *single* SCAAT observation can be viewed alone as a distinct *time-invariant* system. In section 1.2 I stated that each such system in a SCAAT implementation is locally unobservable. From linear system theory (see

[Maybeck70] p. 48) we know that a general necessary and sufficient condition for observability of a time-invariant linear system is that the observability matrix of equation (5.1) must have rank n.

$$M \equiv \left[H^T \mid A^T H^T \mid \ldots \mid (A^T)^{n-1} H^T \right] \tag{5.1}$$

where n is the dimension of the state. For example, consider a SCAAT system such as that simulated in chapter 6 where the state vector has dimension $n = 12$ as in equation (4.2) and the individual measurement vector has dimension $m = 2$ (a 2D camera observing a single scene point). For any single filter update step we can demonstrate the unobservability by forming the observability matrix of equation (5.1) and determining its rank. In particular if we let the 12×12 matrix $A = A(\delta t)$ as in equation (4.14) and the 2×12 matrix H be as in equation (4.15), the resulting 12×24 matrix M has a rank of 2, i.e. the system is unobservable.

So in general there is some number $N \geq n/m$ of measurements that must be taken before the model is completely observable [Maybeck70]. Referring back to "A Single Constraint" on page 37, N is the actual number of measurements, N_{ind} is the number of independent constraints that can be formed from the N measurements, and $C = n/m$. In other words it must be the case that $N \geq N_{\text{ind}} \geq n/m$. A SCAAT implementation does not explicitly collect the necessary measurements at any one point in time, but instead implicitly and continually collects them over time by employing a different measurement model as in equation (4.10) at each filter update step. Because the resulting overall system is not time invariant, the global observability must be examined through some means other than equation (5.1).

5.1.3 Complete Conditions for SCAAT Filter Stability

Fortunately there exist some broad conditions which (if satisfied) imply global observability and stability as outlined in section 5.1.1. Per the work of [Deyst68] (see also [Gelb74] pp. 131-132) if there are real numbers $\alpha_1, \beta_1 > 0$ and $\alpha_2, \beta_2 < \infty$ such that the conditions

$$\alpha_1 I \leq \sum_{i=k-N}^{k-1} A(t_k - t_{i+1}) Q(t_k - t_{i+1}) A^T(t_k - t_{i+1}) \leq \alpha_2 I \tag{5.2}$$

and

$$\beta_1 I \le \sum_{i=k-N}^{k} \Gamma(i,k) \le \beta_2 I \tag{5.3}$$

where

$$\Gamma(i,k) = A^T(t_i - t_k) H^T(t_i) R_\sigma^{-1}(t_i) H(t_i) A(t_i - t_k)$$
$$H(t_i) = H_\sigma(\vec{x}(t_i), \vec{b}(t_i), \vec{c}(t_i))$$

are satisfied for all $k \ge N$ for some number of estimates $N \ge n/m$, then the global system formed by equation (4.1) and the various measurement models from equation (4.10) is uniformly asymptotically stable (and observable) in the large.

Bounded Dynamics

Given $A(\delta t)$ from equation (4.4) the requirement of equation (5.2) says that the dynamic change over the time between estimates $\delta t = t_k - t_{i+1}$, and the corresponding uncertainty, must be sufficiently bounded. In other words one must continue to produce estimates, and one must assume some finite non-zero uncertainty in the form of $Q(\delta t)$ in equation (4.6) in-between each estimate to prevent the filter from reaching a state where it ignores future measurements.

Bounded Measurement Noise

The requirement of equation (5.3) says that over some number of estimates $N \ge n/m$, the measurement model noise $R_\sigma(t)$ of equation (4.11) must always be bounded. (It is reasonable to expect that both $Q(\delta t)$ and $R_\sigma(t)$ are bounded in most problems of physical interest.)

Sufficient Constraints

The requirement of equation (5.3) also says that the estimates must always make use of a sufficient set of sources and sensors such that the Jacobians $H_\sigma(\vec{x}(t), \vec{b}(t), \vec{c}(t))$ are also bounded. Bounding the Jacobians from below is achieved collectively over time. Bounding from above is expected for any implementation, lest the system be ill-conditioned. In other words, one must incorporate a sufficient set of non-degenerate source/sensor measurements, i.e. in general $N \geq N_{\text{ind}} \geq C$ as discussed in “A Single Constraint” on page 37, and in particular $N \geq N_{\text{ind}} \geq n/m$ as discussed above in section 5.1.2. This is a requirement for any implementation however equation (5.3) allows the bounding to occur over the time interval $[t_{k-N}, t_k]$.

It is interesting to note that while the requirement in equation (5.3) for sufficient non-degenerate measurements must be ensured by the source/sensor ordering scheme, to some extent the SCAAT method inherently addresses its own requirements for bounded dynamics in equation (5.2). Because each estimate is based on an individual measurement, the per-estimate measurement time is reduced; the reduced measurement time reduces $\delta t = t_k - t_{i+1}$; and this in turn can improve the stability of the filter by reducing the intra estimate dynamics.

Sufficient Sampling Rate

Recall from section 2.2.2 on page 50 that in general the measurement or sampling frequency should be at least twice the true target motion bandwidth or an estimator may track an alias of the true motion. The SCAAT method is in principle no different in this respect, although the requirement has a somewhat unique implication. If the maximum target motion bandwidth is $\Omega\ [\text{cycles/second}]$, then a SCAAT implementation must—in theory—incorporate constraints at the rate of

$$\Omega\left[\frac{\text{cycles}}{\text{second}}\right] \times 2\left[\frac{\text{WC estimates}}{\text{cycle}}\right] \times C\left[\frac{\text{constraints}}{\text{WC estimate}}\right] = 2\Omega C\left[\frac{\text{constraints}}{\text{second}}\right],$$

where C, the number of non-degenerate constraints per well-constrained (WC) estimate, is as in “A Single Constraint” on page 37. Thus if each measurement provides a single constraint, measurements must incorporated at the rate of $2\Omega C\ [\text{measurements/second}]$.

While at first this requirement might appear to dispel some of the temporal advantages claimed in section 2.2 on page 48, there are three points to keep in mind. First, in practice it is unreasonable to expect that the maximum bandwidth will always be needed. When it is, the system will need $2\Omega C\ [\text{measurements/second}]$ to track the target, when it is not, any excess estimation bandwidth will provide additional filtering (smoothing). Second, even when the bandwidth is needed, the target cannot move in all directions of the state space at once. Thus one could use a measurement ordering scheme such as that presented in section 4.6 (page 106) to sample along the most fruitful directions. Finally, in any case the SCAAT method affords all of the other unique benefits described in chapter 2, e.g. the ability to perform multisensor data fusion and concurrent autocalibration.

5.1.4 SCAAT Filter With Calibration

With appropriate substitutions, the conditions given in section 5.1.3 on page 108 also apply to the augmented tracker filter described in section 4.4. Although one is estimating two sets of dependent unknowns, the conditions can still be satisfied under most circumstances, for reasonable finite tracking sessions, by tuning the filter parameters.

There is however an added concern about long-term stability of such a system. If the process noise $Q_\pi(\delta t)$, see step (b) in section 4.4.2, for all device filters is non-zero, the individual device filters may collectively drift over long periods of time as the augmented filter tries to minimize error with general disregard for the "truth". This is particularly a problem if a beacon is only observed from one general direction. I claim that this concern can be addressed by either precisely calibrating a small subset of devices or parameters through some other (off-line) means, by governing their motion at a fixed threshold, or by simply *declaring* a small subset to be perfect. One can effectively lock parameters in place by initializing the respective device's process noise matrix $Q_\pi(\delta t)$ to zero. When the parameters are subsequently used in an estimate, the measurement residual $\Delta\vec{z}$ will be attributed to all *other* elements of the state. The system would thus in a sense be autocalibrated with respect to the locked components (devices or parameters).

5.1.5 Inaccuracy of the Dynamic Model

The stability discussion in the preceding sections 5.1.1-5.1.4 assumes that equation (4.1) is a precise model of the actual target dynamics. In practice, the precise target motion model is not known. For example, with VE tracking systems the target is a human for which we have no precise motion model. This situation can be represented by rewriting equation (4.1) as

$$\vec{x}(t) = A(\delta t)\vec{x}(t-\delta t) + \vec{f}(t) \tag{5.4}$$

where $f(t)$, representing the actual (unknown) target dynamics at time t, replaces the uncertainty represented by $\vec{\omega}(\delta t)$ in equation (4.1).

Charles Price [Price68] presents an elegant analysis of Kalman filter stability under circumstances where one *can* determine $\vec{f}(t)$ for a particular problem. However in concluding remarks he points out that "The problem remains of determining $\vec{f}(t)$ in a particular application so that $\Delta\vec{f}(t)$ is actually bounded." (The term $\Delta\vec{f}(t)$ represents the difference between the actual unknown signal $\vec{f}(t)$ and an estimate of it $\hat{f}(t)$.)

For VE tracking systems we cannot determine or impose structure on $\vec{f}(t)$ as Price suggests—this would amount to determining or imposing structure on the user motion. As such the conditions outlined in section 5.1.3 cannot be fulfilled in practice. While this is generally the case for any Kalman filter implementation, it might seem of particular significance for a SCAAT implementation. On the contrary, the SCAAT method inherently mitigates the lack of knowledge about $\vec{f}(t)$ by producing estimates at a faster rate than a corresponding multiple-constraint-at-a-time implementation. The increased rate means that no matter what the true nature of $\vec{f}(t)$, the change $\vec{f}(t) - \vec{f}(t - \delta t)$ is minimized by reducing δt. Thus the uncertainty introduced by the lack of knowledge of $\vec{f}(t)$ can be better subsumed by $\vec{\omega}(\delta t)$ in equation (4.1).

The experiments presented in chapter 6 all made use of the simple dynamic model given in section 4.2.1. In other words it was assumed that the model was precise or that $\vec{f}(t) = 0$. We are certain that this is not a valid assumption, but because our experiments have demonstrated both stability and improved accuracy (as compared to the other methods) we are not immediately concerned by the assumption. It remains to be seen whether an improved model noticeably improves the accuracy. (See section 7.1, page 155.)

5.2 Computational Complexity

In chapter 2 I argued for the use of a Kalman filter in general, and a SCAAT implementation in particular. In section 2.2.4 I introduced the notion that the amount of estimate computation is a function of the measurement information. In this section I provide specific numbers to support the asymptotic expression equation (2.6), and a comparison with the previous UNC optoelectronic tracker algorithm, the Collinearity algorithm discussed in section 3.4 on page 68. Note that because all of the tracking data is floating-point, I count only floating-point operations to assess the complexities.

5.2.1 EKF Methods (SCAAT and MCAAT)

Tracking (Without Autocalibration)

Table 5.1 provides floating-point instruction counts for the EKF tracking algorithm presented in section 4.3 on page 83 for a varying number of measurement elements. The specific implementation was that employed in the simulations of chapter 6. As discussed in section 2.2.4 on page 52, the smaller m, the less computation $(m \Leftrightarrow N)$.

Table 5.1: Upper bound on EKF floating-point operations. The bounds, which are based on the algorithm presented in section 4.3 on page 83, are presented as a function of the state and measurement vector dimensions n and m. This allows the comparison of a SCAAT implementation with various MCAAT implementations. No attempt was made to account for data accesses.

Step from pp. 97-98	As a function of n state and m measurement elements	PV model $(n = 12)$	Optimized per section ~~4.3.8~~ 4.5
a	$4n^3 + n^2 - n$	7044	12
b[*]	$162nm + \frac{97}{2}m$	$\frac{3985}{2}m$	$\frac{3985}{2}m$
c	$4n^2m + 4nm^2 + m^3 - 3nm$	$m^3 + 48m^2 + 540m$	$m^3 + 24m^2 + 126m$
d	m	m	m
e	$2n^3 + 2n^2m - n^2 + 2nm$	$312m + 3312$	$204m + 3312$
f	71	71	71
g	3	3	3
Total	$6n^3 + 6n^2m + 4nm^2 + m^3 + 161nm - n + \frac{99}{2}m + 474$	$m^3 + 48m^2 + \frac{5691}{2}m + 10830$	$m^3 + 24m^2 + \frac{4647}{2}m + 3398$

* The expression shown corresponds to the image-based example presented above in section 5.1, and originally in section 1.2.

With the exception of step (b), the expressions given in table 5.1 are valid for any similarly-implemented 6 DOF EKF-based tracking system. The problem with step (b) is that different types of devices will likely require different measurement function equations (4.10). At one extreme, a sensor may offer a *direct* measurement of one of the state elements in equation (4.2). In this case, the measurement function performs a trivial data copy, i.e. no floating-point operations. For example rate gyros could be used to directly measure each of $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$. At the other extreme, the measurement function may have to perform relatively complicated operations (such as the case with the image-based example used previously). For the sake of comparison, I have chosen to offer an expression that reflects the latter. Not only is this more interesting but it also facilitates comparison with another existing approach, the one previously employed with the UNC optoelectronic tracking system.

For the example image-based system from section 5.1 there are $m = 2N$ measurement elements (camera coordinates) for N scene points. Using the expressions for the PV model EKF from table 5.1 ($n = 12$) the total number of floating-point operations $\mathrm{FP}(N)$ for an optimized EKF implementation is given in equation (5.5). Compare equation (5.5) to equation (2.6) on page 52.

$$\mathrm{FP}_{\mathrm{EKF}}(N) = 8N^3 + 96N^2 + 4647N + 3398, \quad N \geq 1 \tag{5.5}$$

Adding Autocalibration

Table 5.2 shows the additional floating-point operations necessary to add device autocalibration (see section 4.4 beginning on page 99) to the optimized EKF reflected in table 5.1. Note that calibrating $n_\pi = n_b + n_c$ device parameters affects the size n of the state vector, but not the size m of the measurement vector. Thus for example while several matrix multiplies will grow, the time for the matrix inversion in step (c) will not change. (The expressions in table 5.2 were obtained by evaluating the expressions in table 5.1 with $n = n + n_\pi$, and then subtracting the original expressions in table 5.1.) Note that the augmentation that occurs in step (a) involves only moving data, hence there are no additional floating-point operations. Also, the increase in operations for step (b) is due only to the additional Jacobian elements in equation (4.12), otherwise the autocalibration will not affect the complexity of the measurement function.

Table 5.2: Added floating-point operations with autocalibration. Expressions reflect the increase in the fourth column of table 5.1 to accommodate the autocalibration of n_b source and n_c sensor parameters ($n_\pi = n_b + n_c$) in an optimized EKF.

Step from pp. 97-98	*Additional* floating-point operations
a	0
b*	$162n_\pi m$
c	$4n_\pi m^2 + (4n_\pi^2 + 45n_\pi)m$
d	n_π
e	$2n_\pi^3 + \left(\frac{3}{2}m + 71\right)n_\pi^2 + (39m + 840)n_\pi + \frac{95}{2}m$
f	0
g	0
Total	$2n_\pi^3 + \left(\frac{11}{2}m + 71\right)n_\pi^2 + (4m^2 + 246m + 841)n_\pi + \frac{95}{2}m$

* The expression shown corresponds to the image-based example presented above in section 5.1, and originally in section 1.2.

While the total additional floating-point operations presented in table 5.2 may appear daunting, keep in mind that for a SCAAT implementation (in particular) m is likely to be relatively small, e.g. $m = 2$ for the previous image-based examples. In addition, n_π is likely to be relatively small for most devices, e.g. $n_\pi = n_b = 3$ when trying to autocalibrate (better estimate) scene point locations for the same example system.

Consider again the image-based system from section 5.1 where $m = 2N$ for N scene points. For an EKF-based system that attempts to concurrently autocalibrate (estimate) the 3D scene point locations, the additional necessary floating-point operations for an optimized implementation becomes

$$\Delta\mathrm{FP}_{\mathrm{EKF}}(N) = 48N^2 + 1670N + 3216,\ N \geq 1\ . \tag{5.6}$$

By combining equation (5.5) and equation (5.6) one arrives at equation (5.7), an expression for the total number of EKF floating-point operations with autocalibration.

$$\mathrm{FP}_{\mathrm{EKF}}(N) = 8N^3 + 144N^2 + 6317N + 6614, \; N \geq 1 \tag{5.7}$$

5.2.2 The Collinearity Method

For the example image-based system from section 5.1 there are $m = 2N$ measurement elements (camera coordinates) for N scene points. Using the Collinearity method (section 3.4 on page 68) implemented as described in [Azuma91], an upper bound on the number of floating-point operations is given in equation (5.8).[1]

$$\mathrm{FP}_{\mathrm{Collinearity}}(N) \in O(((1597 + 504i)N + 1524i - 123)j), \; N \geq 3 \tag{5.8}$$

In equation (5.8) the parameters i and j reflect the number of run-time iterations in the Collinearity algorithm for the SVD (singular value decomposition) and Δ-convergence respectively.[2] For the experiments of chapter 6, $E\{i\} = 33.0$ and $E\{j\} = 1.8$.

Note that for an EKF method, equations (5.5)-(5.7) offer precise floating-point instruction counts. This is possible because the EKF computations are closed-form. On the other hand, for the Collinearity method equation (5.8) provides only an upper bound on the floating-point instruction count. This is because the Collinearity computations are iterative. If the expected values for the parameters i and j are substituted into the upper bound of equation (5.8), the result is

$$\mathrm{FP}_{\mathrm{Collinearity}}(N)\Big|_{i = 33.0,\, j = 1.8} \leq 32813N + 90305, \; N \geq 3 \tag{5.9}$$

In practice, the number of Collinearity floating-point instructions is typically fewer than indicated by the upper bound of equation (5.9) because many loops tend to terminate early, etc. For the experiments of chapter 6,

$$\mathrm{FP}_{\mathrm{Collinearity}}(N) \approx 4630N, \; N \geq 3\,. \tag{5.10}$$

5.2.3 Collinearity vs. the EKF Methods

To compare the computational complexities of the EKF and Collinearity methods using any of equations (5.5)-(5.10) one must consider the practical range of scene point

1. Note that $n = 6$ here because the algorithm does not estimate derivatives.

2. See [Azuma91] for more explanation about the Collinearity algorithm.

observations N. For a Collinearity implementation, $N \geq 3$ known scene points can in theory be used per estimate, while $10 \leq N \leq 15$ are more typically necessary in practice for the system described by [Azuma91]. On the other hand, for an EKF implementation $N \geq 1$ known scene points can be used per estimate. The number determines the observability of the filter, i.e. its classification in the diagram of figure 1.3 on page 38. Under the circumstances described in section 1.2 on page 36, the use of $N \geq 4$ known coplanar scene points per estimate would offer a well-constrained (unique) solution, i.e. a completely observable EKF implementation. On the other hand, the use of only one scene point per estimate, i.e. $N = 1$, reflects a SCAAT EKF implementation.

Figure 5.1 (page 118) offers a graphic comparison between Collinearity and EKF implementations for varying numbers of scene point observations N, with and without autocalibration, optimized and unoptimized. From a per-estimate standpoint, a SCAAT EKF implementation ($N = 1$) is computationally more attractive than either a multiple-constraint implementation with $N > 1$ or a Collinearity implementation where it must be that $N \geq 3$. In addition to being more computationally complex per estimate, the Collinearity algorithm does not directly estimate the target velocities as can a SCAAT implementation. If these derivatives were deemed necessary, they would have to (somehow) be computed in addition to the six parameters inherently computed by the Collinearity algorithm.

More Estimates per Computation

It is interesting to note from equations (5.5)-(5.10) and figure 5.1 that the *total* computation involved in processing a series of twelve (for example) scene points is *less* for a Collinearity implementation than for a SCAAT implementation. In fact, twelve times equation (5.7) with $N = 1$ indicates almost three times the floating-point operations of equation (5.10) with $N = 12$. So if one was concerned only with the computation involved in simply "processing twelve observations", the Collinearity approach might seem more attractive. But while the SCAAT implementation was "processing" the twelve observations it would have provided twelve corresponding estimates. The Collinearity algorithm on the other hand would have provided only *one*. The SCAAT approach provides more estimates per computation.

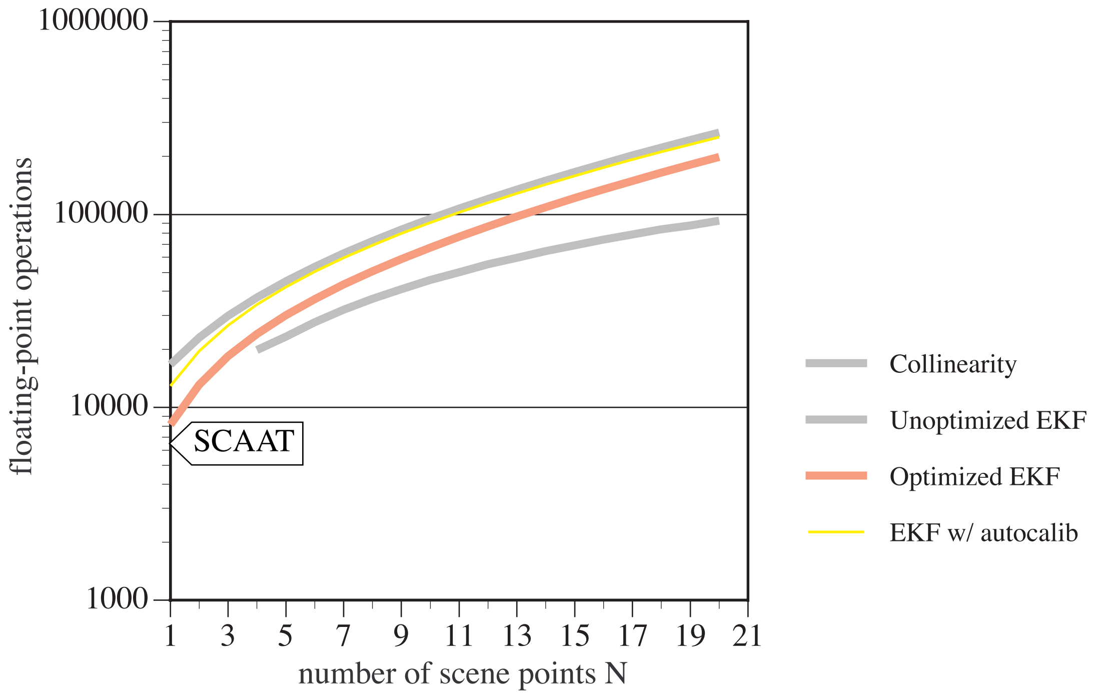


Figure 5.1: Floating-point operation comparison: Collinearity vs. EKF. The EKF data is based on table 5.1 and table 5.2. The EKF w/ Autocalib was optimized. The Collinearity data is based on simulated runs of the implementation given by [Azuma91]. (The Collinearity simulations would only converge with $N \geq 4$ points.)

Chapter 6. Experiments

In this chapter I describe experiments using the SCAAT approach to tracking with the UNC HiBall tracking system. See appendix D for a description of the HiBall tracking system. As part of the development of the HiBall system software, we have built an extensive simulation environment that makes use of detailed mechanical and electrical models that are based on real components. In addition, we have executed the SCAAT computations on the actual target platform to confirm the execution times, etc. See appendix E for a description of the simulation environment. We (Bishop, Chi, Fuchs, Welch et al. at UNC) hope to demonstrate a working HiBall shortly after the publication of this dissertation.

6.1 SCAAT Filter Configuration

In this section, I describe the complete configuration of our SCAAT filter implementation as it applies to both my simulations and our actual implementation of the HiBall tracking system. Because I only simulated multiple-constraint-at-a-time and Collinearity methods for the sake of comparison (we have no intent to actually implement these methods) I include brief discussions of them in section E.3 of appendix E (page 195).

For the initial implementation of our HiBall system, and thus these simulations, we have decided to implement autocalibration only for the sources, the beacon positions, and not the sensors, the HiBall cameras. Thus the SCAAT implementation is configured to employ both a main HiBall filter as described below in section 6.1.1, and many individual beacon filters—one filter per LED—as described below in section 6.1.2.

The reasons for this decision coincide with those given in section 2.5.1 beginning on page 58. The individual HiBall sensors can be readily calibrated off-line and because they are rigidly mounted within the sensor unit the respective parameters should remain constant. This is not the case with the ceiling-mounted beacons. The quantity and distributed nature of the beacons renders their off-line calibration impractical, and the beacon positions may change over time as ceiling panels shift from building vibrations.

6.1.1 The HiBall Filter

We have implemented the HiBall filter exactly as presented in section 4.2 beginning on page 73, including the simple PV dynamic model and the incremental rotations. At any point in time, the complete filter state is described by the state vector equation (4.2) and the global quaternion equation (4.3), thus these elements together provide the best estimate of the HiBall state. The state vector is augmented to implement the beacon position autocalibration as described in general in section 4.4, and in particular in the next section. The final augmented state vector is then

$$\widehat{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \Delta\phi & \Delta\theta & \Delta\psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & x_b & y_b & z_b \end{bmatrix}^T \qquad (6.1)$$

where $x \ldots \psi$ are as in equation (4.2), and $\begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^T$ describes the position of the beacon currently being used in the filter update.

6.1.2 The Beacon Filters

Following the autocalibration method of section 4.4, we maintain a distinct Kalman filter for each individual beacon (LED). Because as described earlier we are primarily concerned with the position of each beacon in world coordinates, this is the per-beacon parameter we have chosen to autocalibrate.

With more than 3,000 beacons in the current implementation, this may seem like a daunting task. Referring to section 4.4.2, for each device we need to maintain a state vector, a noise covariance matrix, and an error covariance matrix. However, because we assume that all of the beacons are equally likely to move, very little at that, all of the beacon filters can share a *single* beacon process noise covariance matrix. Therefore for each beacon filter we only maintain a three-dimensional state vector and error covariance matrix, for a total of 12 elements per beacon. The total memory needed for beacon autocalibration is on the order of 36,000 double words—a relatively small amount by today’s standards.

At each filter update step, after the observation of a beacon, the appropriate beacon filter state vector and error covariance matrix are swapped in and out of their respective HiBall filter counterparts as described in section 4.4.3. The single (common) beacon process noise covariance matrix was determined as described below.

6.1.3 Process Noise

Because we chose to autocalibrate the beacon positions there are actually two processes being estimated at each filter update step: the HiBall process and the beacon process. As a result, we need two distinct process noise matrices: $Q(\delta t)$ in equation (4.6) for the HiBall filter, and $Q_b(\delta t)$ in the augmentation equation (4.27) that is shared by all of the beacon filters as discussed above in section 6.1.2.

For $Q(\delta t)$ we needed to determine noise autocorrelation elements $\vec{\mu}[i]$ in equation (4.7) for $i \in \{x, y, z, \phi, \theta, \psi\}$. Using the definition of equation (4.9) on page 77 the units for these can be determined from table 4.1 on page 74. Because we could see no reason to expect a user to favor lateral translation in any particular direction, we defined a single lateral autocorrelation $\vec{\mu}[\lambda] \equiv (\vec{\mu}[x] = \vec{\mu}[y])$ to reduce the necessary parameters from six to five.

For the common beacon noise covariance $Q_b(\delta t)$ we needed to determine the matrix elements that would reflect a very small amount of uncertainty in the "constant" parameters, each beacon's position in world coordinates in this case. (See footnote 4 on page 100.) We chose to model the uncertainty in each 3D position parameter as a random walk, i.e. integrated white noise, and we empirically determined the correlation elements directly as opposed to using transfer function methods. Again, to reduce the number of parameters, we chose to define one beacon position noise source autocorrelation $\vec{\mu}[b] \equiv (\vec{\mu}[x_b] = \vec{\mu}[y_b] = \vec{\mu}[z_b])$ for all of the beacon position parameters x_b, y_b, and z_b. In addition we assumed that there was no correlation between these elements, i.e. $Q_b(\delta t)$ should be diagonal:

$$Q_b(\delta t)[i, j] = \begin{cases} \vec{\mu}[b] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$. The units for $\vec{\mu}[b]$ are $[\text{meters}]^2$.

In total then we needed to determine six SCAAT autocorrelation parameters: $\vec{\mu}[i]$ for $i \in \{\lambda, z, \phi, \theta, \psi, b\}$. Our method for empirically choosing these parameters is discussed in section E.4 of appendix E.

6.1.4 Initial Error Covariance

Because we are estimating multiple processes there are multiple error covariance matrices: the Hiball error covariance matrix and an error covariance matrix for each beacon. As a result there will be multiple distinct error covariance matrices: $P(t)$ in equation (4.13) for the HiBall filter, and a $P_{b_i}(t)$ in the augmentation equation (4.25) for each of the beacon filters as discussed above in section 6.1.2.

For the various beacon filter's $P_{b_i}(t)$, where i is the beacon number, we needed to determine the matrix elements that would reflect the initial uncertainty (variance) in the constant parameters, each beacon's position in world coordinates in this case. Because we believed our initial beacon position estimates to all be equally good (or bad) we chose to use a common initial error covariance matrix $P_b(0)$. In other words, we let $P_{b_i}(0) = P_b(0)$ for all beacons i. To reduce the number of parameters, we chose to use a single initial variance ξ_b for all of the beacon position parameters x_b, y_b, and z_b. In addition, we assumed that there was no correlation between these elements, i.e. $P_b(0)$ should be diagonal. In other words,

$$P_b(0)[i, j] = \begin{cases} \xi_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$. The non-zero elements ξ_b of the error covariance matrix have units of $[\text{meters}]^2$. Our method for empirically choosing ξ_b is discussed in section E.4 of appendix E.

For the initial HiBall filter error covariance $P(0)$, I used two strategies as follows. For all of the comparison simulations that I present in section 6.2.3 below, I initialized the state vector (and the global orientation) with the correct position and orientation values, and zero velocity components. This approach is justified because in practice we could use a conventional method such as Collinearity to determine the initial state and then transition to the SCAAT method once the filter was initialized. For these simulations I set $P(0)$ to the null matrix to reflect the initial "certainty" in the state vector.

However when attempting a “cold start” (start-up with a null state vector) as discussed below in section 6.2.7, I needed to set the elements of $P(0)$ to reflect the true lack of confidence in the initial state. The non-zero elements of $P(0)$ for this case are given in that section.

6.1.5 Measurement Function

With respect to the SCAAT method, we have chosen to define a single constraint for the HiBall tracking system as one observation of one beacon with one HiBall lens and sensor pair. Thus the measurement vector given in equation (4.10) and the respective components in equations (4.11) and (4.12) are all two-dimensional, i.e. $m_\sigma = 2$. The situation is similar to that depicted in figure 1.6 on page 41.

For the initial UNC HiBall tracking system implementation, and hence for these simulations, there is only one sensor type σ: a HiBall camera (photodiode). Therefore there is only a need for a single measurement function equation (4.10) to predict the measurement in equation (4.15). This function needs to predict the $\begin{bmatrix} u & v \end{bmatrix}^T$ coordinates that would reflect the position of the image of the respective beacon in the HiBall camera as discussed in section D.2 of appendix D. In particular for equation (4.15),

$$\begin{aligned} \hat{z} &= \begin{bmatrix} u & v \end{bmatrix}^T \\ &= \vec{h}_\sigma(\hat{x}^-(t), \hat{\alpha}(t), \vec{c}(t)). \end{aligned} \tag{6.2}$$

The augmented state vector $\hat{x}(t)$ contains estimates for the HiBall position, the incremental rotation, and the beacon position as in equation (6.1). Note that because the source (beacon) information is contained in the augmented state vector there is no need for a source parameter vector $\vec{b}(t)$ as in equation (4.10). The sensor parameter vector $\vec{c}(t)$ on the other hand contains the camera geometry, the camera position and orientation in HiBall coordinates and the focal length of the lens/sensor pair. Therefore given the augmented state vector *prediction* $\hat{x}^-(t)$, the global orientation estimate $\hat{\alpha}(t)$, and the camera parameter vector $\vec{c}(t)$, the measurement function $\vec{h}_\sigma(\bullet)$ in equation (6.2) predicts the measurement as follows:

a. transform the beacon from world space into HiBall space using the HiBall and beacon position estimates in the predicted state $\hat{x}^-(t)$, the global orientation $\hat{\alpha}(t)$, and the incremental rotation in $\hat{x}^-(t)$;

b. transform the beacon from HiBall space into camera space by using the results of step (a) and the camera position and orientation in $\vec{c}(t)$; and

c. project the results from step (b) onto the camera's image plane using the focal length in $\vec{c}(t)$.

We have implemented the Jacobian matrix function $H_\sigma(\hat{x}^-(t), \hat{\alpha}(t), \vec{c}(t))$ in equation (4.12) as a set of functions that collectively reflect the derivative of the predicted measurement $\hat{z}$ in equation (6.2) with respect to the augmented state vector $\hat{x}^-(t)$. We determined the set of functions in an off-line examination of the expressions corresponding to the preceding steps. The resulting 2×15 Jacobian indicated in equation (4.15) is therefore

$$\begin{aligned} H[i, j] &= \frac{\partial}{\partial \hat{x}^-(t)[j]} \hat{z}[i] \\ &= \frac{\partial}{\partial \hat{x}^-(t)[j]} \vec{h}_\sigma(\hat{x}^-(t), \vec{\alpha}(t), \vec{c}(t))[i] \\ &= H_\sigma(\hat{x}^-(t), \hat{\alpha}(t), \vec{c}(t)) \end{aligned}$$

for all $i \in \hat{z}$ and $j \in \hat{x}$. In the interest of brevity, I have omitted the mathematical details for both the preceding steps and the Jacobians, as they are relatively straightforward. Keep in mind that throughout we have implemented the optimizations discussed in section 4.5 on page 103.

6.1.6 Measurement Noise

To compute the Kalman gain in equation (4.16) we need an estimate of the uncertainty in each actual camera measurement $\vec{z} = \begin{bmatrix} u & v \end{bmatrix}^T$. This uncertainty takes the form of a measurement error covariance matrix $R_\sigma(t)$ as in equation (4.11), where the elements reflect the expected variances and covariances of the measurement vector elements. While not necessary, I chose for these simulations to represent the uncertainty in both measurement elements with a single variance ξ_c. In addition, I assumed that there was no correlation between the elements, i.e. I chose to make $R_\sigma(t)$ diagonal. In other words,

$$R_\sigma(t)[i, j] = \begin{cases} \xi_c & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

for $i, j \in \{u, v\}$. We might achieve improved accuracy with a more selective $R_\sigma(t)$. Note that like the error covariance matrix discussed in section 6.1.4 above, the non-zero elements ξ_c of the measurement error covariance matrix have units of $[\text{meters}]^2$.

For the beacon and camera approach of the HiBall, the variance ξ_c in the measured camera coordinates depends primarily on the beacon signal strength as seen by the camera. For a particular (constant) beacon energy, this signal strength depends primarily on two factors: the angle α_b between the primary axis of the beacon and the primary axis of the camera, and the distance d_b from the camera to the beacon as shown below in figure 6.1. So while not reflected by equation (6.3), the uncertainty (variance) ξ_c is actually a function of this angle and distance, which in turn depend on the state (the position and orientation of the HiBall), which of course varies with time.

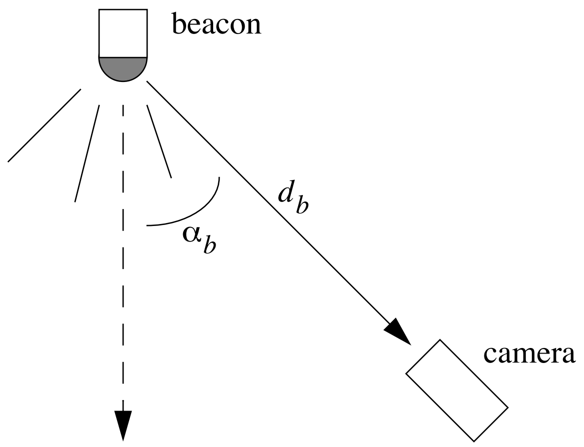


Figure 6.1: Measurement signal strength. The strength of the beacon signal as seen by the camera depends on the angle α_b and the distance d_b. As the angle and the distance increase, the signal strength decreases.

In 1995 Vernon Chi at UNC published a technical report that developed a very complete noise model for outward-looking optical tracking systems—such as the UNC HiBall system—along with MACSYMA simulation code and suggested parameters [Chi95].

Using Chi's model, Gary Bishop formulated an approximation for ξ_c as a function of the angle and the distance:

$$\sqrt{\xi_c} = \frac{\sqrt{\xi_0} d_b^2}{a\alpha_b^3 + b\alpha_b^2 + c\alpha_b + 1}, \tag{6.4}$$

where

$$\begin{aligned} \xi_0 &= 2.89\text{e-}14, \\ a &= -1.074\text{e-}6 \\ b &= -2.331\text{e-}6 \\ c &= -2.394\text{e-}3 \end{aligned}$$

The quantity ξ_0 is the variance of the measurement noise for a sighting with $\alpha_b = 0$ degrees and $d_b = 1$ meter. The coefficients a, b, and c were determined by Bishop using the MACSYMA simulation environment along with the noise model, the MACSYMA code, and the parameters provided in [Chi95]. An important thing to note about equation (6.4) is that the noise increases with the square of the distance, and because $a, b, c < 1$, it increases with the cube of the angle.

Just prior to computing the Kalman gain in equation (4.16), we use the HiBall and beacon information contained in the augmented state vector $\hat{x}^-(t)$ to compute the camera-to-beacon angle α_b and distance d_b. We then compute ξ_c using equation (6.4), and form $R_\sigma(t)$ as in equation (6.3).

6.2 Simulation Results

6.2.1 EKF Parameters

High Sampling Rate

Choosing a representative portion of the "typical" data set discussed in section E.1 of appendix E and the method discussed in section E.4, I determined the parameter sets $\Pi[N]$, $N = 1, 3, 10$, for the three sets of circumstances simulated throughout this section: no beacon error; 1.7 millimeter RMS beacon error without autocalibration, and 1.7 millimeter RMS beacon error with autocalibration.[1] (Gottschalk and Hughes reported approximately one millimeter of beacon error after use of their calibration scheme

[Gottschalk93].) The results of the parameter searches for each of these circumstances, with each of $N = 1, 3, 10$, are presented below in tables 6.4-6.6 for an observation rate of 1000 Hz. Note that an EKF with $N = 1$ is a SCAAT implementation.

Table 6.4: EKF parameter sets for *perfect* beacons. Parameters $\vec{\mu}[b]$ and ξ_b do not apply here because beacon autocalibration was not employed. The units are discussed in sections 4.2.1 and 6.1.3.

	$\Pi[N]$						
units	$[\text{meters}/\text{sec}^2]^2$		$[\text{radians}/\text{sec}^2]^2$			$[\text{meters}]^2$	
N	$\vec{\mu}[\lambda]$	$\vec{\mu}[z]$	$\vec{\mu}[\phi]$	$\vec{\mu}[\theta]$	$\vec{\mu}[\psi]$	$\vec{\mu}[b]$	ξ_b
1	5.05e-1	1.58e-1	6.08e-1	3.66e-1	2.04e-1	—	—
3	4.48e-2	3.47e-2	2.68e-1	1.48e-1	4.43e-2	—	—
10	1.28e-2	3.44e-3	3.70e-2	2.16e-2	2.94e-2	—	—

Table 6.5: EKF parameter sets for 1.7 mm beacon error *without* autocalibration. Again as with table 6.4, parameters $\vec{\mu}[b]$ and ξ_b do not apply. The units are discussed in sections 4.2.1 and 6.1.3.

	$\Pi[N]$						
units	$[\text{meters}/\text{sec}^2]^2$		$[\text{radians}/\text{sec}^2]^2$			$[\text{meters}]^2$	
N	$\vec{\mu}[\lambda]$	$\vec{\mu}[z]$	$\vec{\mu}[\phi]$	$\vec{\mu}[\theta]$	$\vec{\mu}[\psi]$	$\vec{\mu}[b]$	ξ_b
1	1.48e-1	5.81e-2	1.05e+1	1.19e+1	1.34e+1	—	—
3	3.50e-2	1.83e+0	1.40e+0	9.34e-1	6.71e-1	—	—
10	5.64e-3	3.79e-3	4.07e+1	1.17e-2	3.93e-1	—	—

1 1.7 millimeter RMS error was measured after perturbing the beacon positions with zero-mean normally-distributed position error with a one millimeter standard deviation.

Table 6.6: EKF parameter sets for 1.7 mm beacon error *with* autocalibration. The units are discussed in sections 4.2.1 and 6.1.3.

	$\Pi[N]$						
units	$[\text{meters}/\text{sec}^2]^2$		$[\text{radians}/\text{sec}^2]^2$			$[\text{meters}]^2$	
N	$\vec{\mu}[\lambda]$	$\vec{\mu}[z]$	$\vec{\mu}[\phi]$	$\vec{\mu}[\theta]$	$\vec{\mu}[\psi]$	$\vec{\mu}[b]$	ξ_b
1	3.95e-1	1.07e-1	5.18e-1	1.14e+0	5.89e-1	3.29e-4	2.41e-6
3	4.88e-2	1.42e+0	8.25e-1	1.24e+0	4.49e-1	6.24e-5	5.83e-9
10	4.85e-3	3.29e-3	8.41e+1	8.59e-3	2.14e-2	1.91e-10	9.09e-7

Notice in table 6.4 that as the number of constraints N grows, the noise autocorrelation values $\vec{\mu}[\lambda]\ldots\vec{\mu}[\psi]$ shrink. This same inverse relationship is shown in table 6.6 for the beacon filter parameters $\vec{\mu}[b]$ and ξ_b. This is likely due to the corresponding relationship between the filter update rate and the number of constraints N—the smaller N, the higher the update rate, and the higher the allowable target motion cutoff frequency.

On the other hand, in table 6.5 and table 6.6 there appears to be no clear relationship between the HiBall filter parameters and the number of constraints N. This appears to be a result of the noise introduced by the beacon error: along some directions in parameter space the search algorithm was able to take large strides with no significant improvement in the cost function.

Low Sampling Rates

One question we were interested in was how high (in practice) the filter update rate must be to ensure the stability of a SCAAT implementation, in particular when beacon autocalibration was being used. I attempted to address this question by simulating HiBall systems with lower measurement rates. Because it is reasonable to expect any given HiBall sampling rate to remain relatively constant, we would likely determine a unique set of SCAAT filter parameters for the given rate. As such, I repeated the parameter search for both 100 and 10 Hz sampling rates. The resulting SCAAT parameters are given in table 6.7. The corresponding tracking simulation results are presented below in section 6.2.9.

Table 6.7: SCAAT parameter sets for low sampling rates. The filter parameters were optimized for 1.7 mm beacon error with autocalibration. The process noise parameters $\vec{\eta}[i]$ have units as discussed in sections 4.2.1 and 6.1.3, ξ_b has units $[\text{meters}]^2$.

	$\Pi[1]$						
units	$[\text{meters}/\text{sec}^2]^2$		$[\text{radians}/\text{sec}^2]^2$			$[\text{meters}]^2$	
rate (Hz)	$\vec{\mu}[\lambda]$	$\vec{\mu}[z]$	$\vec{\mu}[\phi]$	$\vec{\mu}[\theta]$	$\vec{\mu}[\psi]$	$\vec{\mu}[b]$	ξ_b
100	3.07e-1	1.65e-1	5.10e-1	4.20e-1	3.43e-1	1.27e-3	5.94e-6
10	2.74e-1	1.46e-1	7.64e-1	6.89e-1	8.29e-1	1.53e-10	9.42e-10

Notice that the HiBall filter noise autocorrelation values $\vec{\mu}[\lambda]...\vec{\mu}[\psi]$ do not appear to change significantly with the sampling rate. On the other hand, the beacon filter parameters $\vec{\mu}[b]$ and ξ_b clearly decrease in magnitude with the sampling rate. This makes sense because when the sampling rate approaches the target motion cutoff frequency it becomes more difficult to distinguish between filter residuals due to target motion and those due to beacon position error.

Low Dynamics

The PV model employed in these simulations (section 4.2.1 beginning on page 73) is based on the assumption that the user's acceleration can be modeled as normally distributed white noise. While this may or may not be a valid characterization for most user motion of interest, the *level* of user dynamics is directly related to the *magnitude* of the EKF noise source parameters.

In particular, one would suspect that the optimal set of parameters for low target dynamics, e.g. a user attempting to remain still, would be very different from those for high user dynamics. Indeed this turns out to be the case. Using the "still" data set described in section E.1 of appendix E, I repeated the parameter search (optimization) for the SCAAT EKF, with 1.7 mm beacon error and autocalibration. The results are presented below in table 6.8.

Table 6.8: SCAAT parameters for "still" target dynamics. Compared with the SCAAT row ($N = 1$) of table 6.6 these parameters are small. The parameters were optimized for 1.7 mm beacon error *with* autocalibration. The units, provided in the second row, are discussed in sections 4.2.1 and 6.1.3.

$\Pi[1]$						
$[\text{meters}/\text{sec}^2]^2$		$[\text{radians}/\text{sec}^2]^2$			$[\text{meters}]^2$	
$\vec{\mu}[\lambda]$	$\vec{\mu}[z]$	$\vec{\mu}[\phi]$	$\vec{\mu}[\theta]$	$\vec{\mu}[\psi]$	$\vec{\mu}[b]$	ξ_b
4.14e-6	9.30e-4	5.95e-4	1.63e-1	7.27e-3	6.99e-3	1.91e-7

Notice that the parameters given in table 6.6 are relatively small when compared to the parameters in the SCAAT row ($N = 1$) of table 6.8. In section 6.2.10 below, I present, for the sake of comparison, simulation results using both the parameters of table 6.8 and those of the SCAAT row from table 6.6. As one would expect, the tracker accuracy is best when the parameters and the structure of the dynamic model match the target dynamics. Depending on the expected dynamics, one might therefore consider implementing a multiple-model implementation, where the filter automatically switches or continuously varies between multiple dynamic models and (or) parameters. In section 7.3 beginning on page 156, I discuss such a multiple-model filter as future work.

6.2.2 Error Nomenclature

Throughout sections 6.2.3-6.2.10 I use three main phrases to indicate error quantities: *overall RMS error*, *overall peak error*, and *per-estimate RMS error*. All three phrases refer to measures of error in the positions of each of three points in a group located 1 meter in front of the HiBall as described in section E.4 on page 197 with respect to the *cost function* used for parameter optimization (searches). Whenever I refer to *point error vector length*, I mean the length of the vector between one of the points' estimated position and its true position.

Overall RMS Error

When I use the phrase *overall RMS error*, I mean the root-mean-square point error vector length over all three points and all of the tracker estimates for a particular run. Because there are three points examined per estimate, this is the RMS length of $3E$ point error vectors, where E is the overall number of estimates for the simulation run. This provides a measure of the point error over an entire simulation run.

Overall Peak Error

When I use the phrase *overall peak error*, I mean the largest (peak) point error vector length over all three points and all of the tracker estimates for a particular run. This provides a measure of the worst-case point error over an entire simulation run.

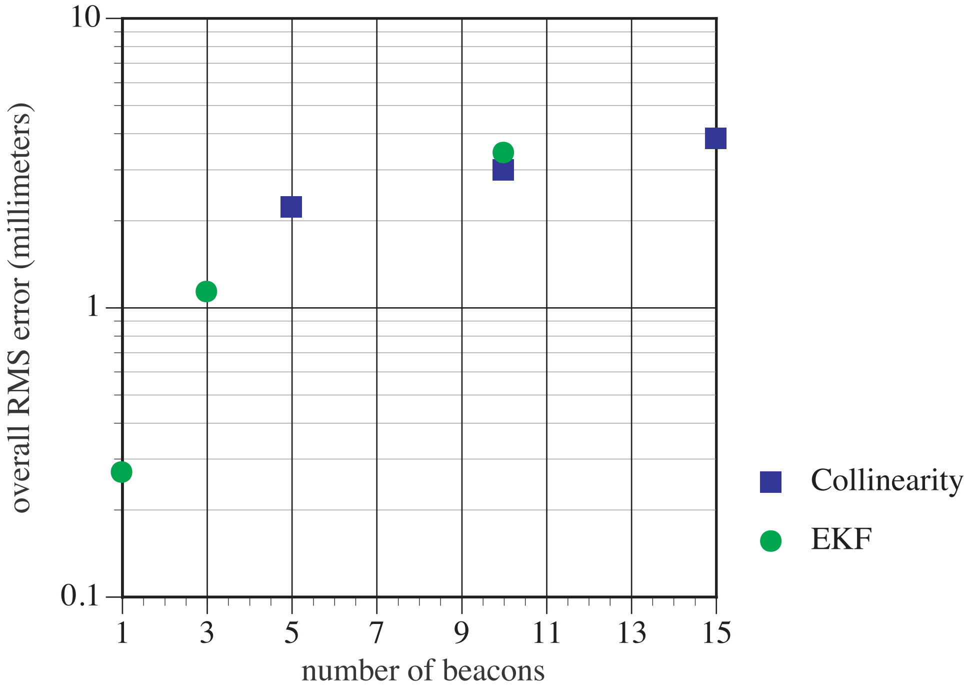
Per-Estimate RMS Error

When I use the phrase per-estimate RMS error, I mean the root-mean-square point error vector length *within* the group of all three points for a single tracker estimate. This provides a measure of the *per estimate* point error.

6.2.3 EKF and Collinearity Tracking

Accuracy

Figures 6.2-6.6 (pages 132-135) are plots that offer comparisons between the simulated accuracy of the Collinearity and EKF methods (including SCAAT) under various conditions. Figure 6.2 presents Collinearity and EKF comparisons for the "typical" motion data set (described in section E.1 of appendix E) with no beacon error and thus no autocalibration. Figure 6.3 presents Collinearity and EKF comparisons for the same data set with 1.7 mm beacon error *without* autocalibration. Figure 6.4 presents a similar comparison, but only for EKF implementations *with* autocalibration. No Collinearity results are included in figure 6.4 because Collinearity cannot directly perform autocalibration. Indeed, part of the motivation for using a Kalman filter is the ability to perform autocalibration (see section 2.1.4 on page 48).



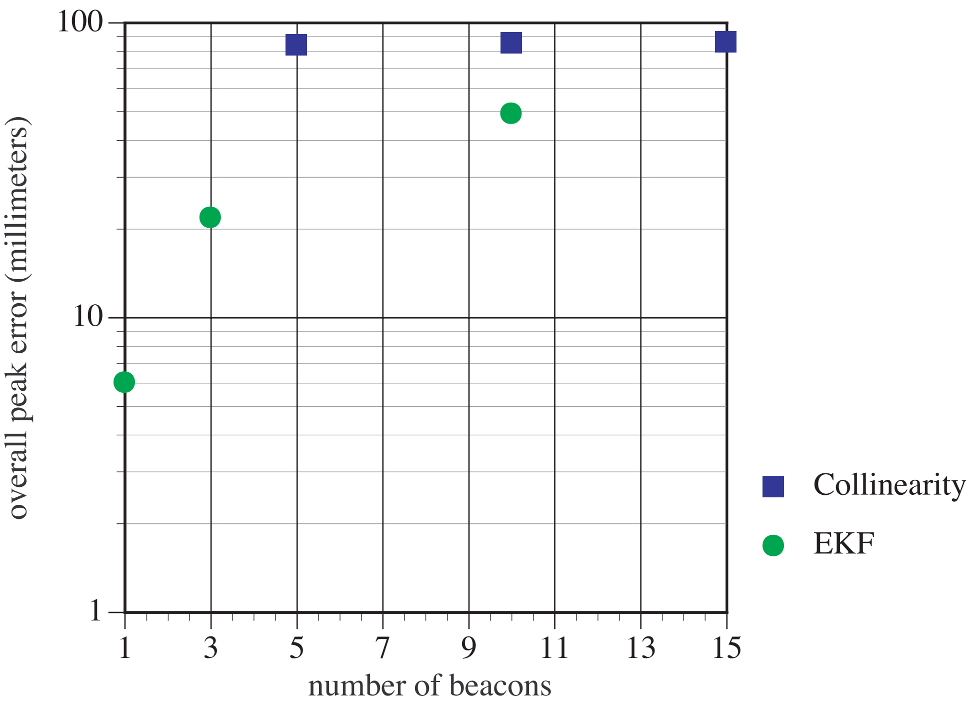
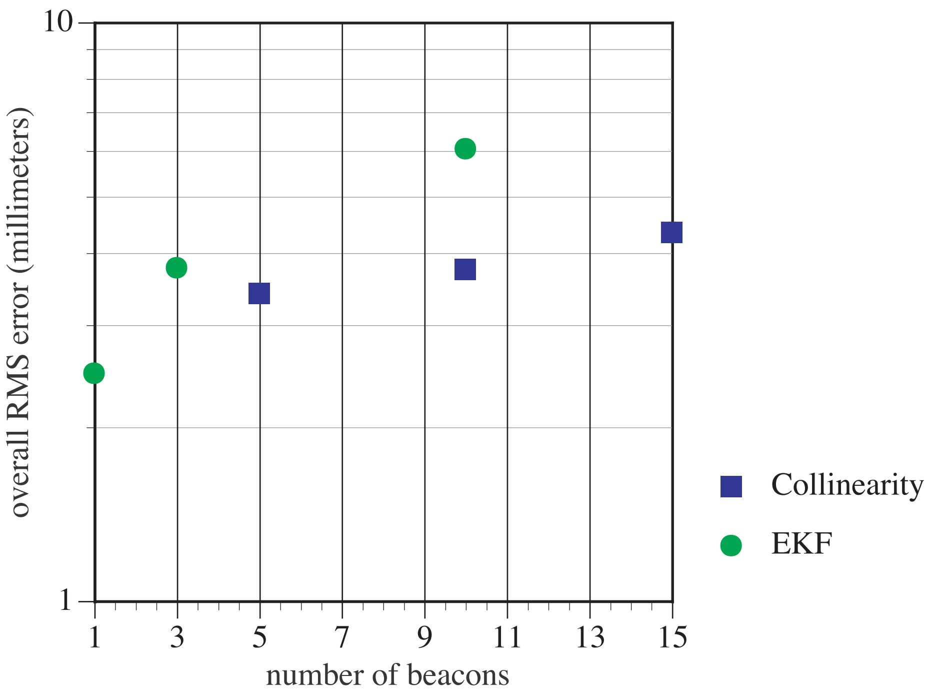


Figure 6.2: Collinearity vs. EKF for perfect beacons ("typical" data set as described in section E.1 of appendix E). An EKF with $N = 1$ beacons is a SCAAT implementation. The EKF parameters were taken from table 6.4.



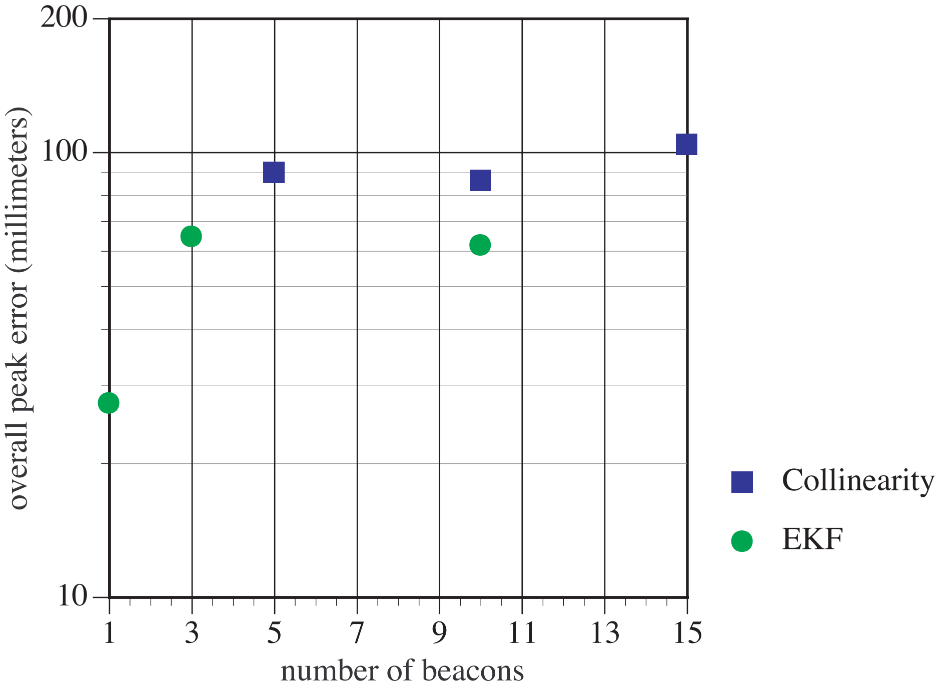
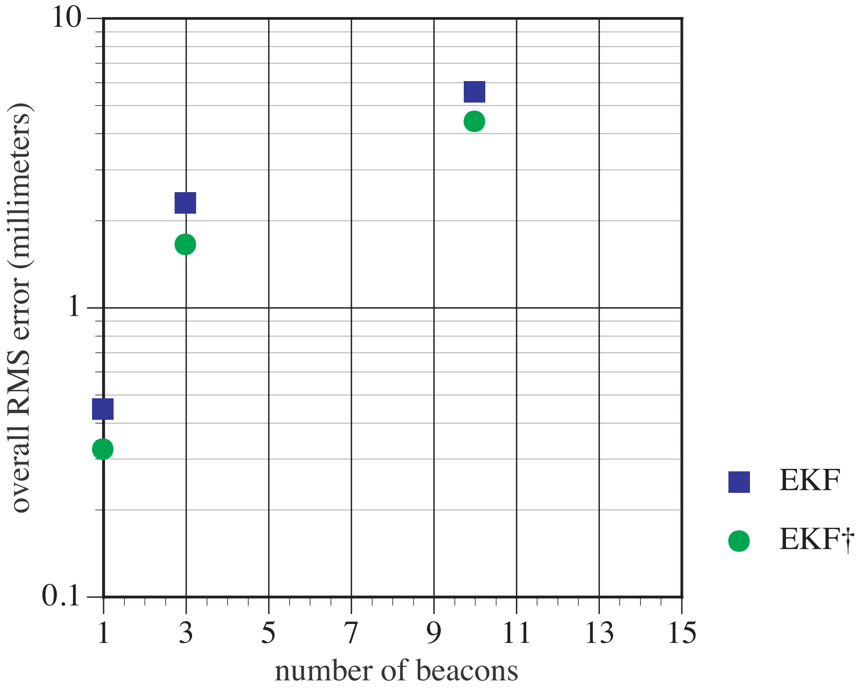


Figure 6.3: Collinearity vs. EKF *without* autocalibration ("typical" data set, 1.7 mm RMS beacon position error). The EKF parameters were taken from table 6.5.



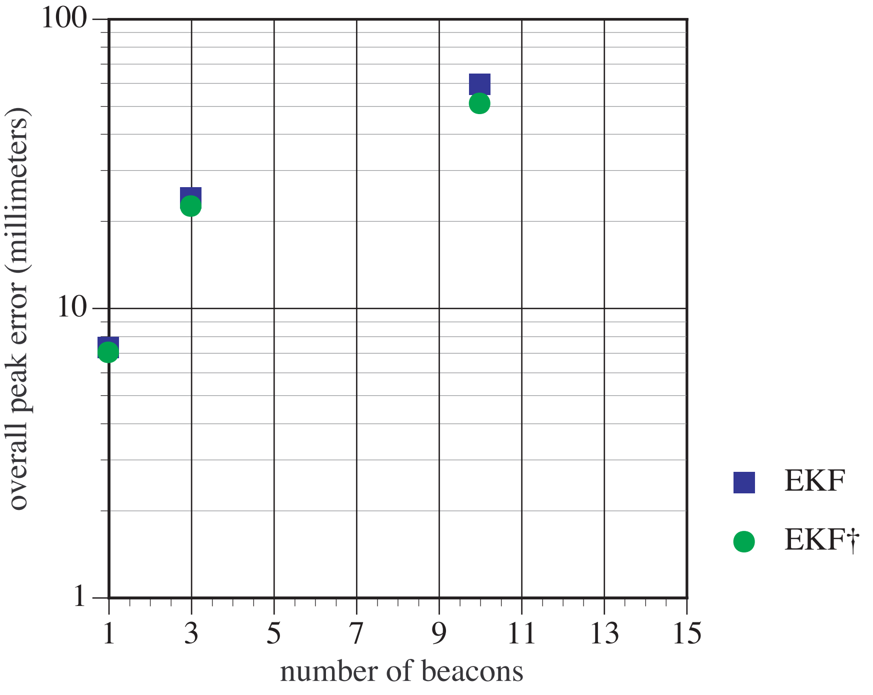


Figure 6.4: EKF *with* autocalibration ("typical" data set, 1.7 mm RMS beacon position error). Collinearity results are not included here because Collinearity cannot (directly) perform autocalibration. The † symbol indicates a *second* EKF run through the same data set, inheriting the autocalibrated beacon position estimates from the *first* run. The EKF parameters were taken from table 6.6.

As a sanity check for the results in figure 6.4, figure 6.5 (page 136) provides a comparison of Collinearity and SCAAT (with autocalibration) implementations for various other individual motion data sets (see section E.1 of appendix E). For both methods, all of the runs began with 1.7 mm beacon error.

Additionally, a SCAAT run through the data sets was repeated a second time to see how SCAAT would perform if given already (partially) autocalibrated beacons. This is meant to simulate a user walking under a particular area of the ceiling more than once, or possibly the systematic inheritance of calibration values from a previous user's session. The results for this second run are denoted with the † symbol in each of figures 6.4-6.6.

Finally, figure 6.6 (below) summarizes the individual results of figure 6.5 with high, low, and average RMS and peak error.

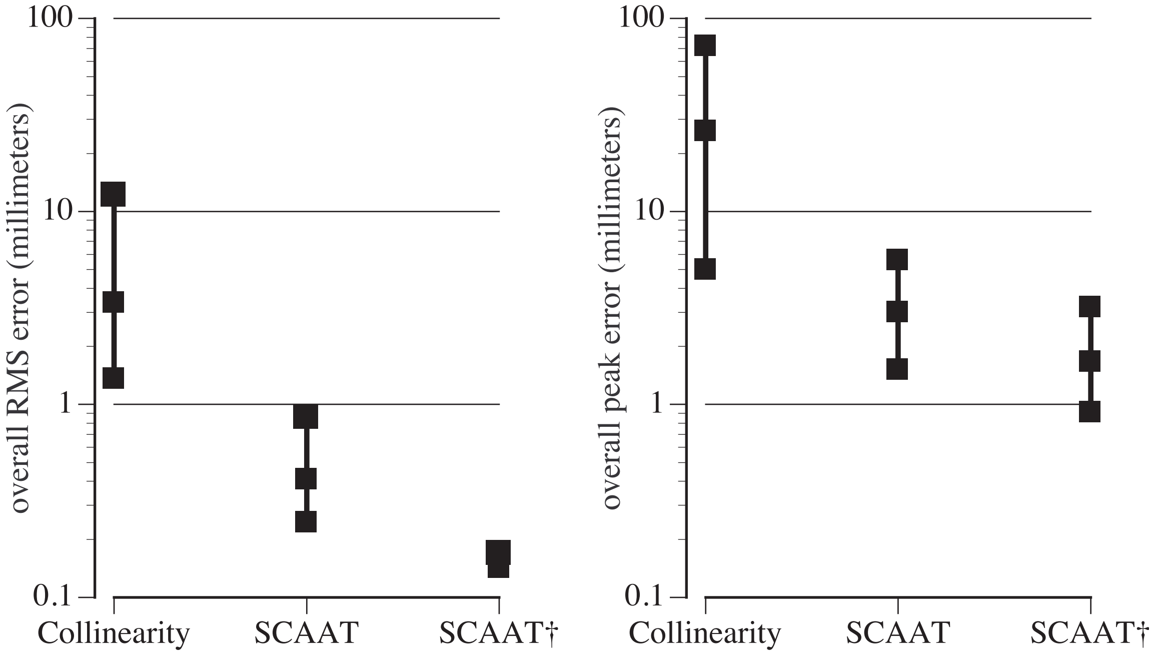


Figure 6.6: Summary—Collinearity vs. SCAAT *with* autocalibration. This chart summarizes the data from figure 6.5, providing the high, low, and average for each method.

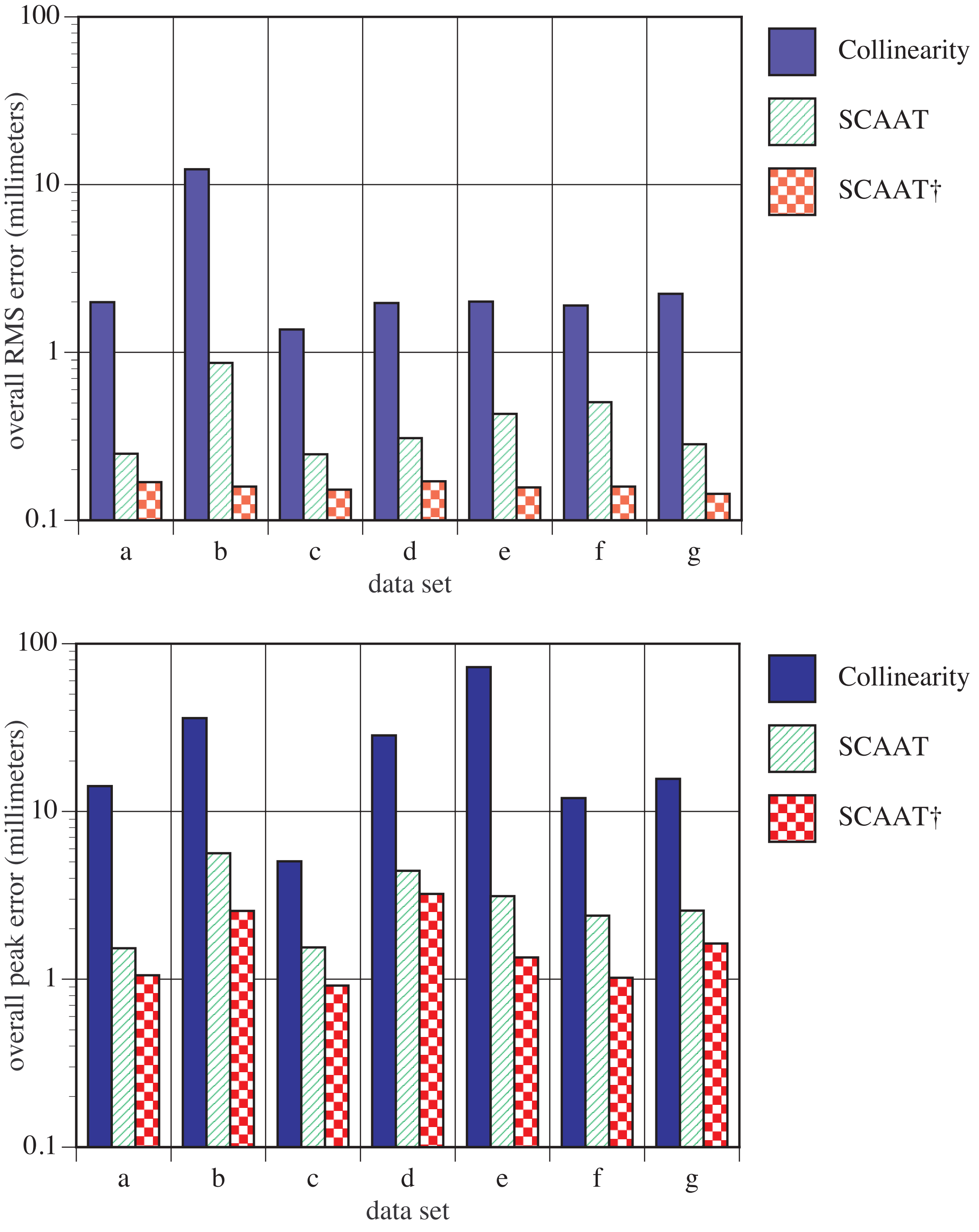


Figure 6.5: Collinearity vs. SCAAT *with* autocalibration (various data sets, 1.7 mm RMS *initial* beacon position error). The data sets are a subset of those described in section E.1 of appendix E. The † symbol indicates a *second* run through the same data set, inheriting the autocalibrated beacon position estimates from the *first* run. (The "typical" data set is not included in sets a-g.)

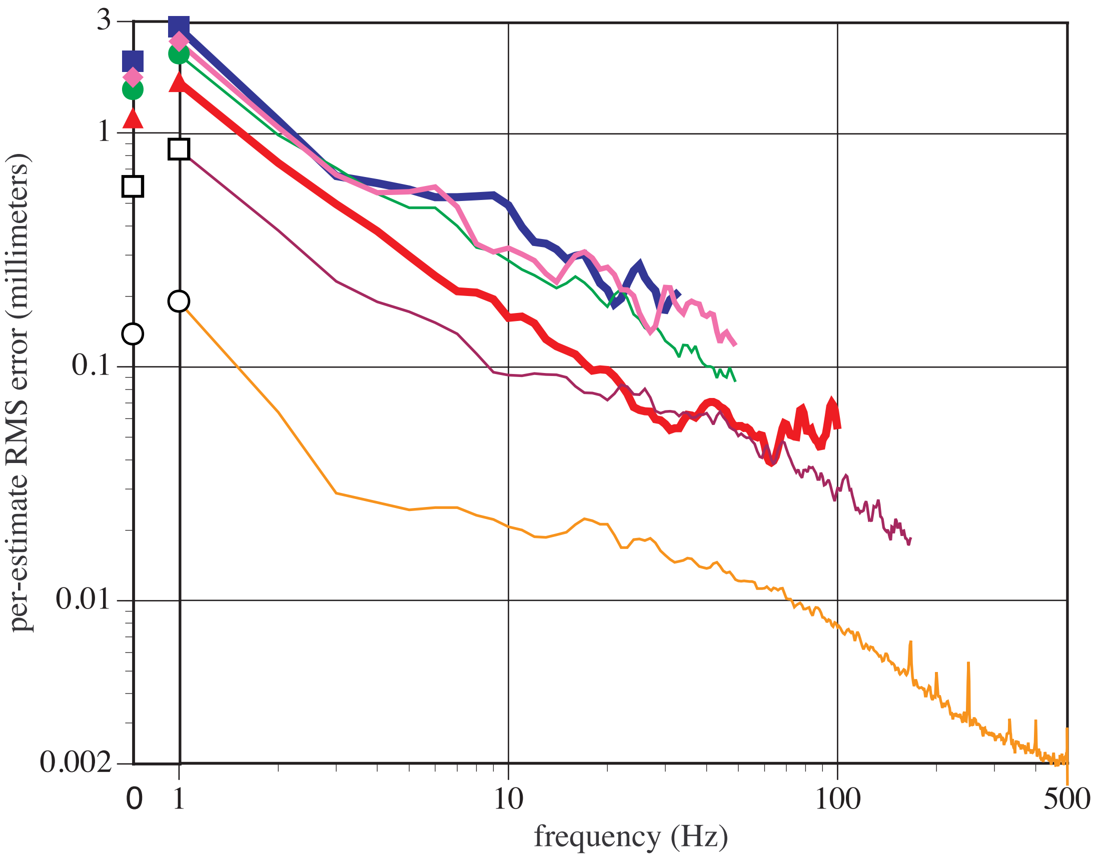
Error Spectra

Figures 6.7-6.9 (pages 138-140) are plots that offer comparisons between the simulated per-estimate RMS error spectra of the Collinearity and EKF methods, including SCAAT, under various conditions. Figure 6.7 presents a comparison between Collinearity and EKF methods with varying numbers of *perfectly* calibrated beacons. Figure 6.8 presents a similar comparison but for the case where the beacons have 1.7 millimeter RMS position error, and autocalibration is turned *off*. Compared with the case of perfect beacons, the Collinearity noise increased only slightly across the spectrum, while the EKF noise was “lifted” to approximately that of the Collinearity results. Compare figure 6.8 with figure 6.7.

Figure 6.9 presents a comparison of the different SCAAT results with and without 1.7 mm beacon error, with and without autocalibration. No Collinearity results are presented in figure 6.9 because Collinearity cannot directly perform autocalibration. As discussed with figure 6.5 (above), results for a repeat SCAAT run—with inherited calibrated beacons—are denoted with the † symbol in figure 6.9. Both the first and second-run SCAAT results with calibration appear to be near perfect in terms of per-estimate RMS error, i.e. the SCAAT results of figure 6.9 very closely resemble the SCAAT results of figure 6.7.

The filtering offered by the EKF methods can be seen in each of figures 6.7-6.9 as the MCAAT and SCAAT error tends to roll-off with higher frequencies. There is no reason to expect the Collinearity method to do the same because it does not perform any intentional filtering (by design). Like autocalibration, the ability to filter or *smooth* estimates is one of the motivations for using a Kalman filter (see section 2.1.1 on page 48).

Notice in each of figures 6.7-6.9 that there are peaks in the 100-500 Hz range of the SCAAT data. We believe these result from a correlation between the direction of user motion and the directions of beacon sightings. Because our camera view selection algorithm is simply round-robin, we may regularly visit views that provide little information. For example, if one-third of all sightings are along or near the direction of motion, then there will likely be a peak at $1000/3 = 333.3$ Hz. This hypothesis is reinforced by the results presented in figure 6.10; when the view selection is random, the peaks disappear. A more intelligent view selection algorithm would likely do the same.



■	Collinearity (15)	◆	MCAAT (10)
●	Collinearity (10)	□	MCAAT (3)
▲	Collinearity (5)	○	SCAAT

Figure 6.7: Error spectra for *perfect* beacons ("typical" data set). The numbers in parenthesis in the legend indicated the number of beacons used. Note the interesting SCAAT peaks above 100 Hz. These are discussed on page 137 and with figure 6.10. The SCAAT approach clearly has the best noise characteristics under the perfect-beacon conditions, certainly the best conditions any autocalibration scheme could hope to achieve.

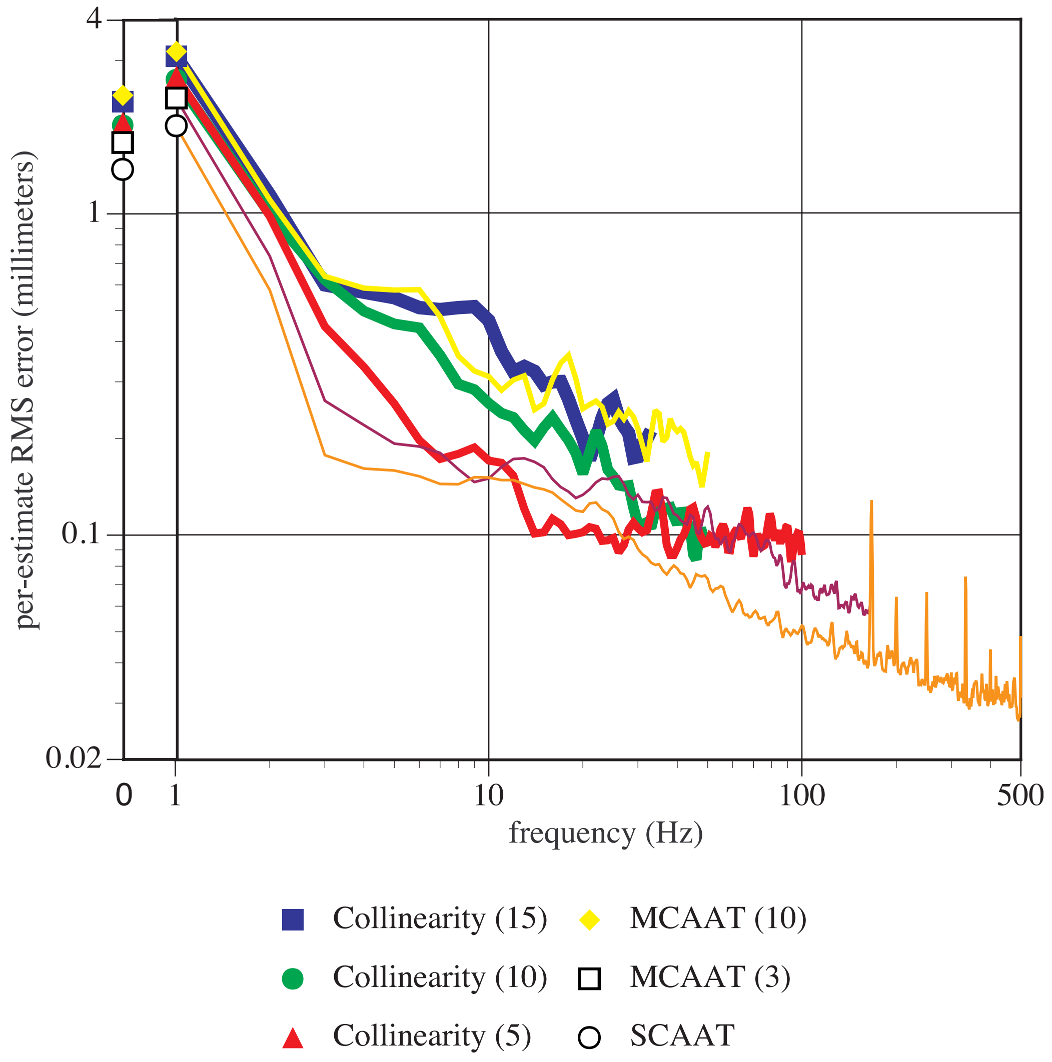


Figure 6.8: Error spectra *without* autocalibration ("typical" data set, 1.7 mm RMS beacon position error). Collinearity appears to be relatively flat above 15 Hz, while the EKF methods continue to roll-off the error. Note the interesting SCAAT peaks above 100 Hz. These are discussed on page 137 and with figure 6.10.

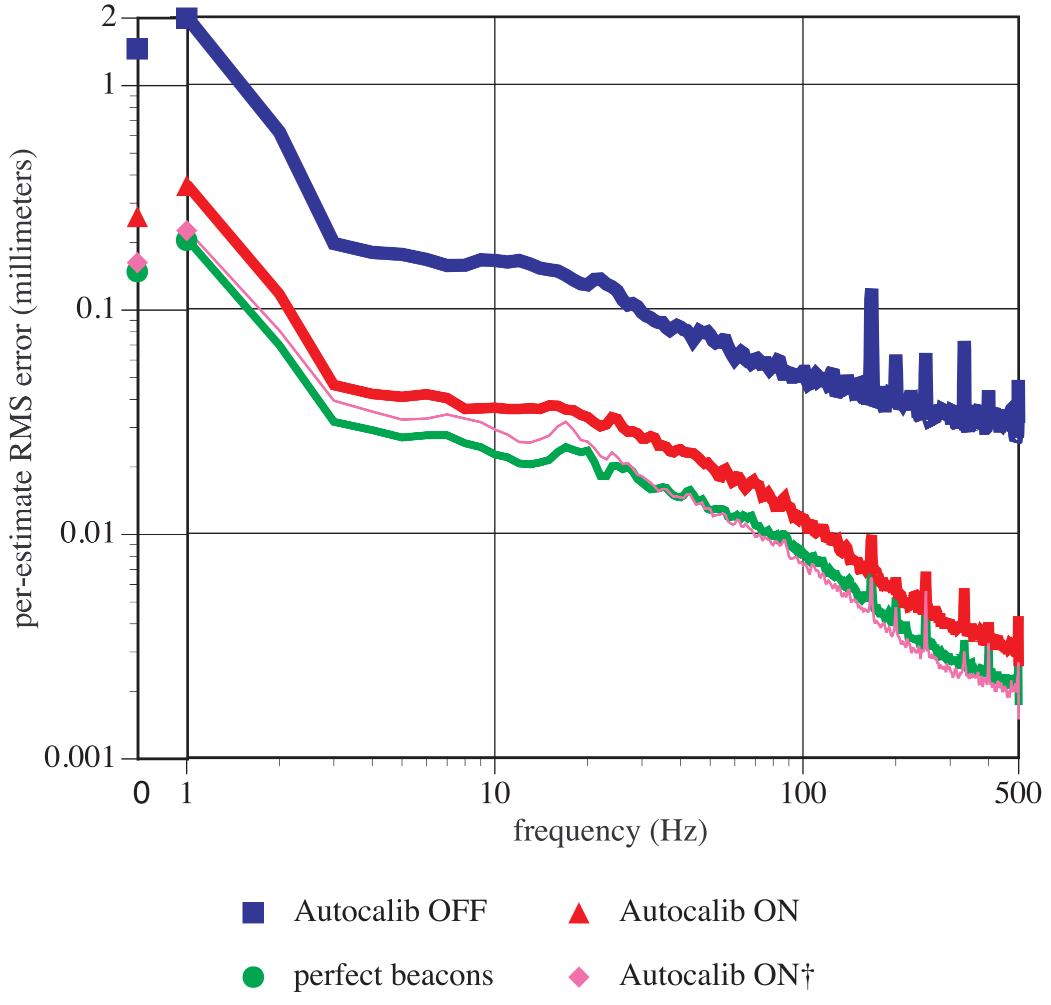


Figure 6.9: SCAAT error spectra ("typical" data set, with and without 1.7 mm beacon position error, with and without autocalibration). Note the interesting SCAAT peaks above 100 Hz. These are discussed on page 137 and with figure 6.10.

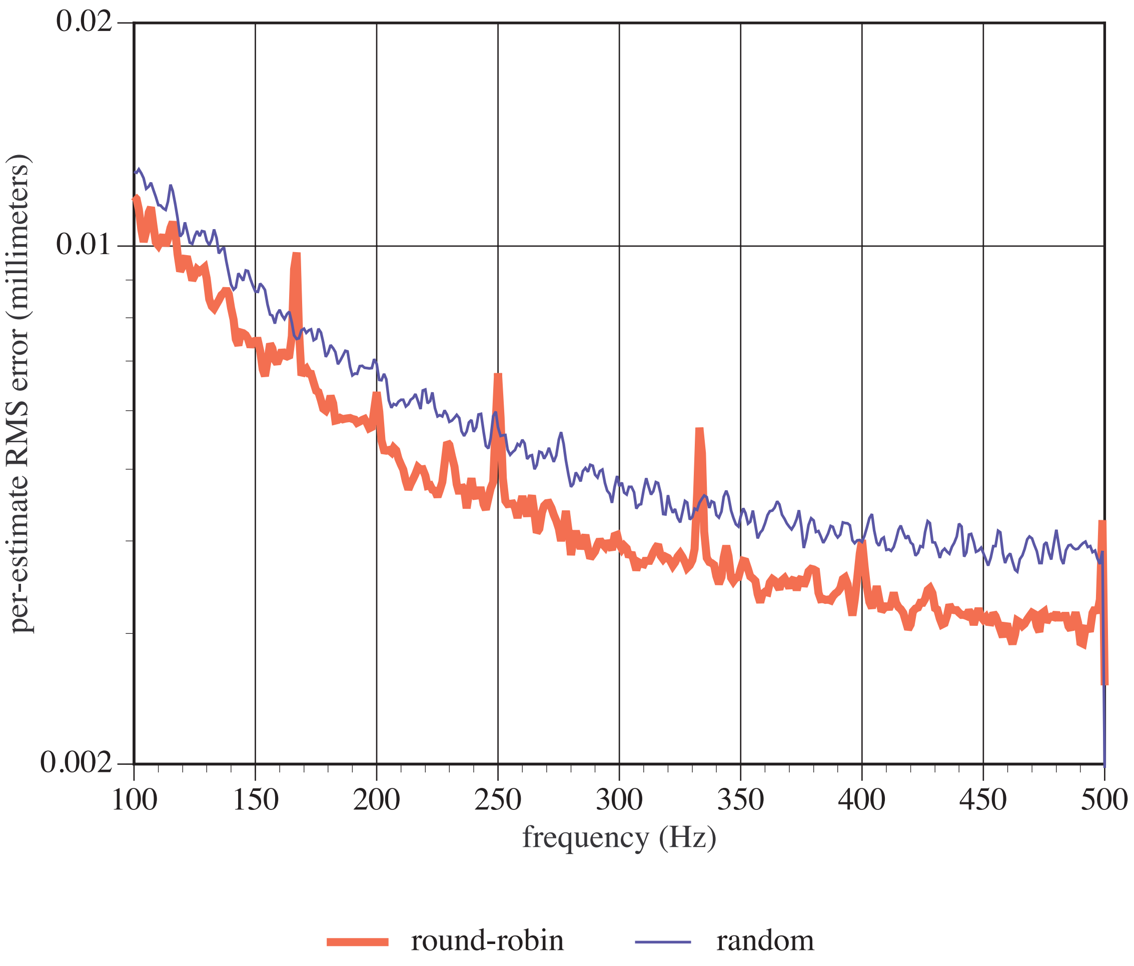


Figure 6.10: Peaks in the error spectrum. In all of the error spectrum plots of figures 6.7-6.9 there are interesting SCAAT peaks above 100 Hz. We believe that these result from a correlation between the direction of user motion and the direction of regular beacon sightings. If the camera selection is random rather than round-robin, the peaks go away. See “Error Spectra” on page 137 for more information.

6.2.4 EKF Beacon Autocalibration

Figures 6.11 and 6.12 (page 143) are charts that show the effects of MCAAT and SCAAT (EKF) autocalibration on the RMS beacon position error. In all of the autocalibration experiments of this chapter the beacons were initially perturbed with normally distributed random position error. (See footnote 1, page 127.) I seeded the random number generator in all simulations to ensure a consistent distribution.

Figure 6.11 compares the results for EKF implementations with the "typical" data set and varying numbers of beacons: MCAAT implementations with 10 and 3 beacon observations, and a SCAAT (single beacon at a time) implementation. Figure 6.12 compares the SCAAT autocalibration results for a variety of data sets.

Close inspection of the final position errors for *individual beacons* in several simulations revealed that some beacon position estimates actually got worse. This is to be expected for the reasons discussed in section 4.4.4 (page 102) and section 5.1.4 (page 111). I actually implemented a method for "locking" particular individual beacons in place as suggested in section 5.1.4, and this seemed to prove successful for these cases. In any case, the *overall* beacon estimates always improved with the SCAAT autocalibration implementation.

While I did observe occasional individual beacon drift (worsening position) with autocalibration, I did not observe any collective drift as mentioned in section 5.1.4 of chapter 5 (page 111). Perhaps this is because my simulations were not long enough—only four minutes at the longest. However it is my belief that as beacons are seen more and more, and seen from different perspectives, the beacon estimates should converge to the true positions. Indeed the occasional drift that I observed was with beacons that were only briefly observed, and from only one general direction. Like the main tracker Kalman filter, the individual beacon filters must have observable measurement systems to remain stable, i.e. to avoid divergence. If a particular beacon is observed only along one direction, the corresponding measurement system is unobservable and divergence would be expected. See section 5.1.4 for more discussion on addressing potential drift.

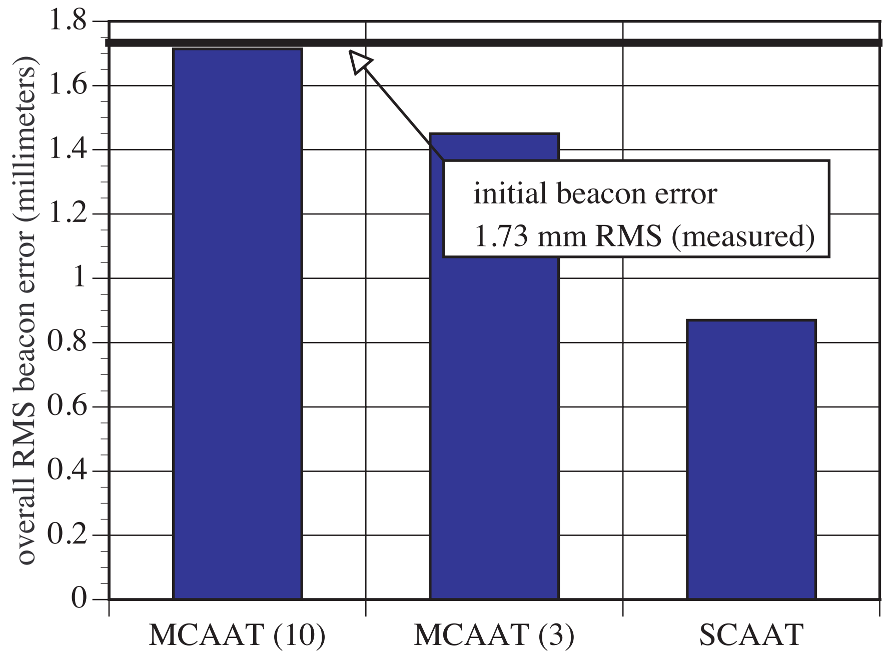


Figure 6.11: Final beacon error for EKF runs *with* autocalibration ("typical" data set).

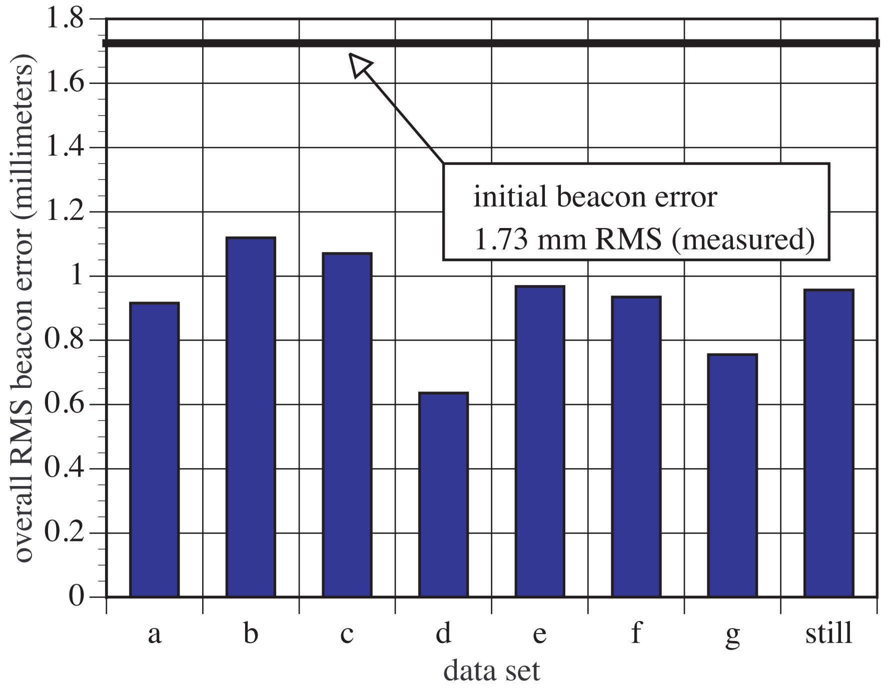


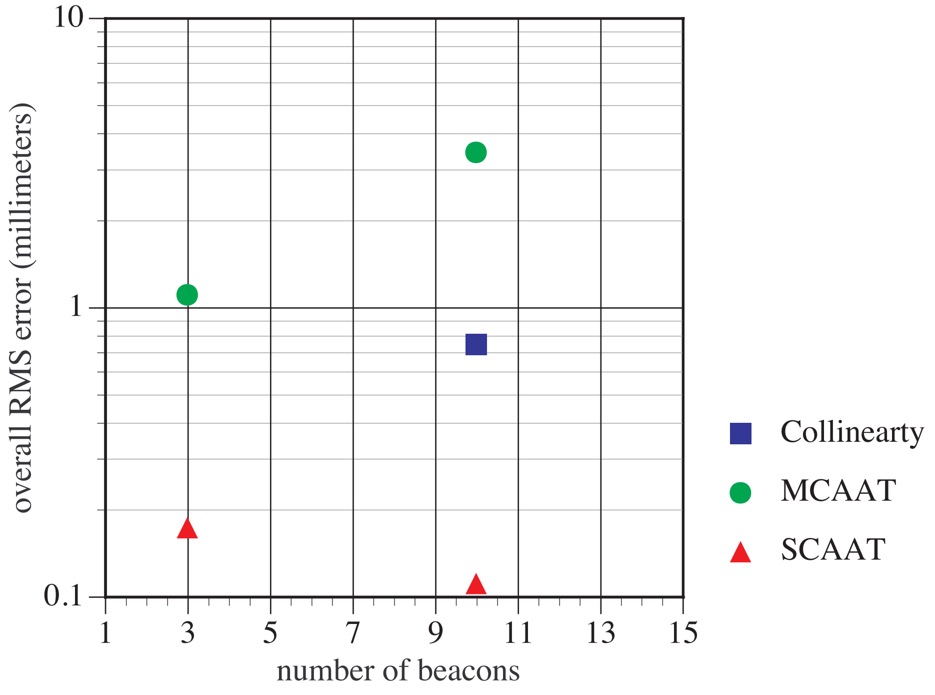
Figure 6.12: Final beacon error for SCAAT *with* autocalibration (various data sets).

6.2.5 Simultaneous Observations

At the end of section 2.3.2, which begins on page 54, I stated that the SCAAT approach had proved valuable even in cases where the observations were *truly simultaneous*, i.e. cases where the simultaneity assumption was *not* inherently violated. In this section I present some simulation results to support this claim.

For the Collinearity and MCAAT methods, the use of simultaneous observations is relatively easy to understand as the methods inherently operate on groups of $N > 1$ observations anyway. But what does it mean for a SCAAT implementation to operate on such a group? It means that even though the SCAAT filter has access to a relatively "large" simultaneously obtained measurement vector with $N > 1$ constraints, it still operates on the individual constraints one at a time. Each time a measurement vector with N simultaneous constraints becomes available, the SCAAT implementation updates the entire state N distinct times, resulting in N times the computation. That is, it offers N complete estimates for the *same* point in time—each estimate better than the previous, the N^{th} being the best for that particular point in time. In this way the SCAAT filter looks a little bit like an iterated Kalman filter (section 3.3, page 67) in that it offers multiple, N in this case, subsequently improving estimates for a particular point in time. And while it is similar, it is *not* the same as sequential updates (see section 3.2, page 65) because each of the N SCAAT measurement updates, with its filter residual and associated Kalman gain, still involves only a single constraint.

Figures 6.13 and 6.14 below are plots that offer comparisons between the simulated accuracy of the Collinearity and EKF methods, including SCAAT, in systems where the N beacons can be observed simultaneously. Figure 6.13 presents the case where the beacon positions are known perfectly and no autocalibration was used. Figure 6.14 presents the case where the beacon estimates are initially perturbed with 1.7 mm RMS error, and autocalibration is used (for the EKF implementations only). Finally, in a manner similar to that of section 6.2.4, figure 6.15 depicts the effects of simultaneous observations on beacon position autocalibration. In each of figures 6.13-6.15 the SCAAT implementation actually performs better with ten beacons than it does with three as the ten-beacons heavily overconstrain the solution at each point in time—each of the ten constraints simply improves the particular estimate more and more.



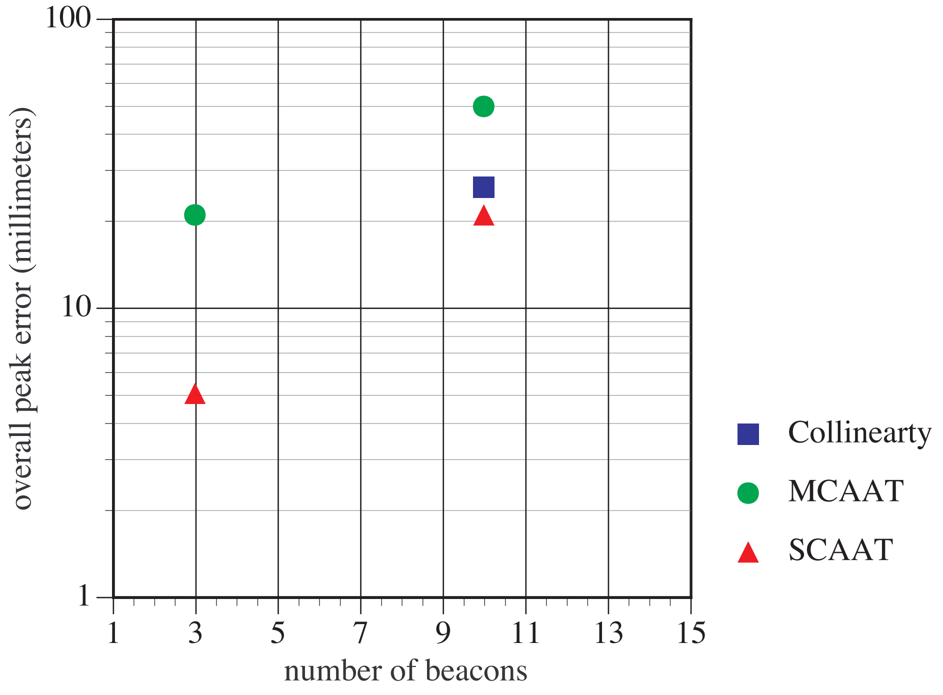
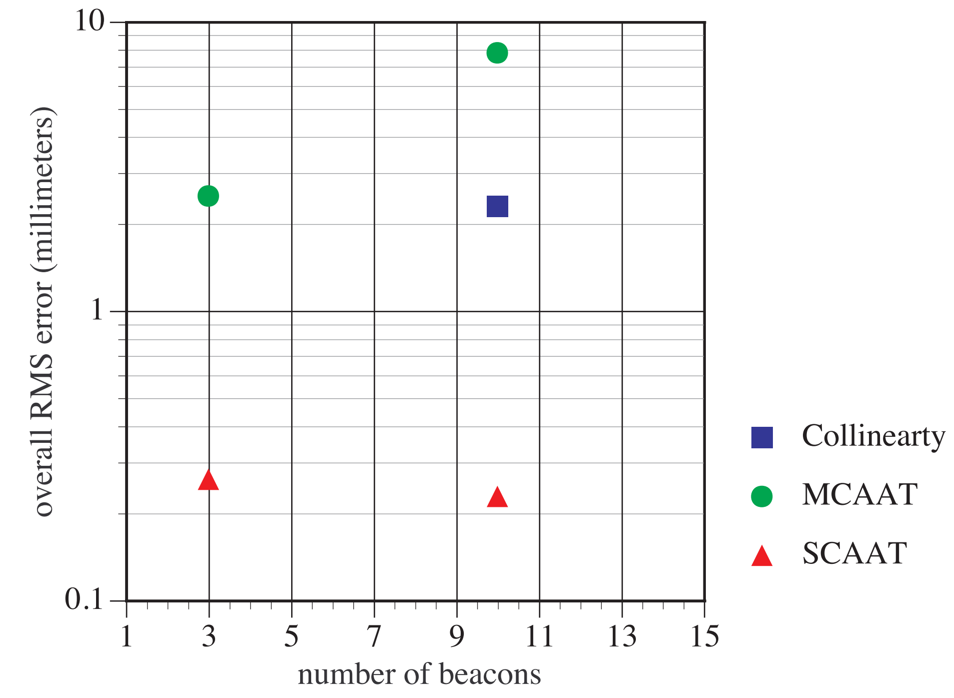


Figure 6.13: Simultaneous observations, *perfect* beacons. For the SCAAT implementation, the *N* simultaneous beacon observations were still processed one at a time. Note that there are no $N = 1$ SCAAT results because "*one* simultaneously observed beacon" does not make sense.



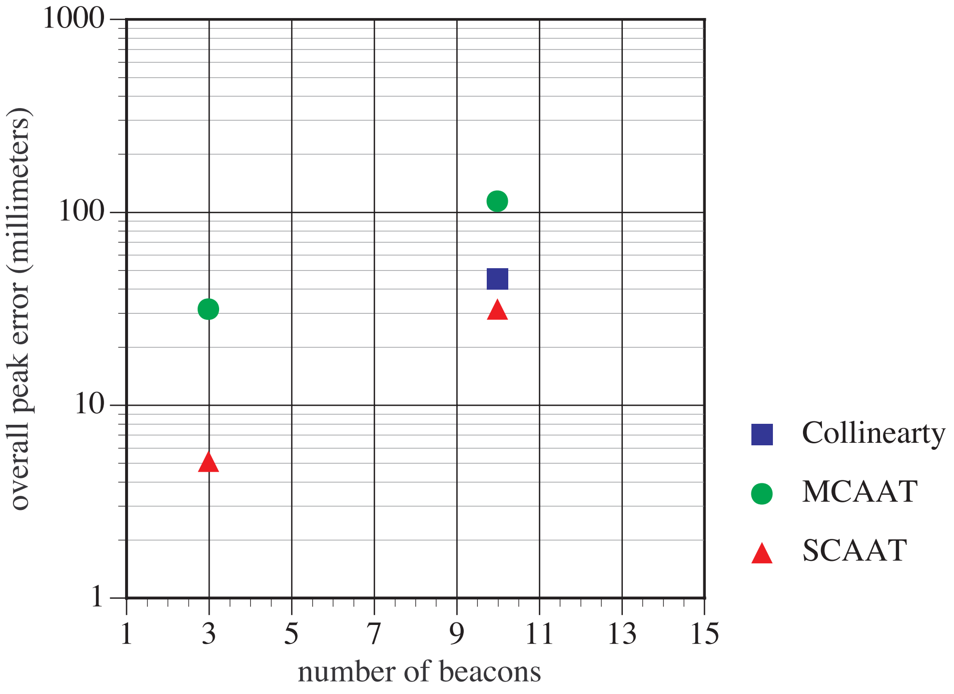


Figure 6.14: Simultaneous observations *with* autocalibration (1.7 mm RMS beacon position error).

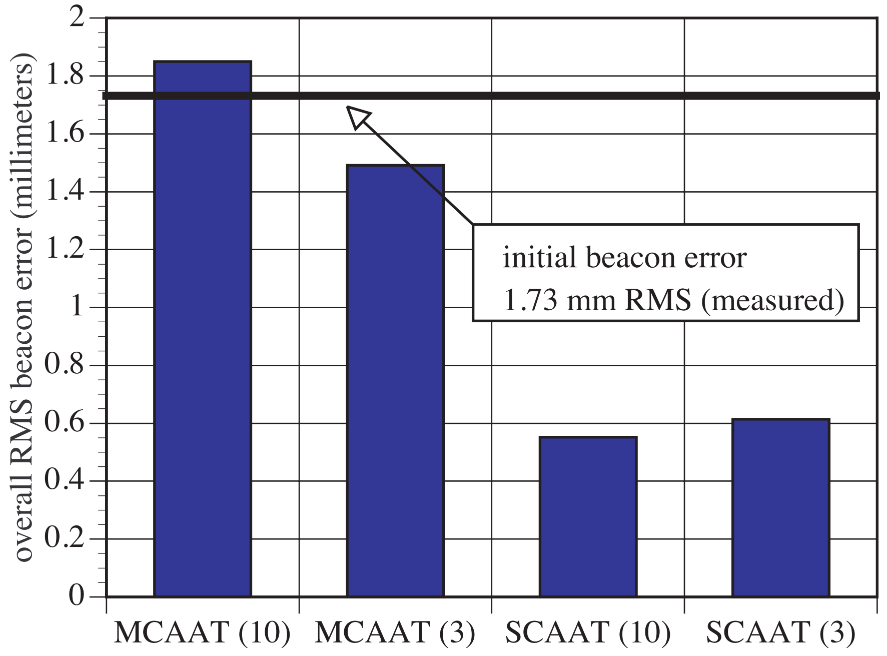


Figure 6.15: Autocalibration with *truly simultaneous* observations ("typical" data set, 1.7 mm RMS beacon position error). Even if the simultaneity assumption is not violated, a SCAAT EKF implementation offers improved performance in terms of autocalibration.

6.2.6 Moving-Window Observations

I performed several simulations of SCAAT, MCAAT, and Collinearity implementations using moving-window observations as described in section E.3.3 of appendix E, page 196. The results were disappointing in that while the data rates were better, the respective accuracies were not, indeed the accuracy was usually slightly worse than for the fixed-window counterparts in the presence of beacon position error. I attribute these results to the fact that a moving-window version "holds on to" a bad measurement, i.e. a measurement of a poorly located beacon, longer than a fixed-window version. The resulting error is certainly not white, but is biased along a particular direction as long as the beacon is "held". In the interest of time and space I have omitted the quantitative results of these experiments.

6.2.7 Cold Start

It is possible for a SCAAT implementation to converge even when starting with a completely erroneous initial state vector. In other words, it is possible for it to perform a “cold start”. Similarly it is possible for a SCAAT implementation to recover from a randomly destroyed state. This behavior under proper conditions is a testament to the stability of the SCAAT method.

Whether or not such cold-start convergence can occur depends on several factors including the observables of the particular system and the degree of difference between the actual state and the “cold” state estimate. It is possible to encounter circumstances where the measurement prediction of equation (4.15) on page 84 has no physical interpretation. For example, in the case of the UNC HiBall tracking system, one can encounter circumstances where a beacon is actually in front of a camera, but the filter thinks it is behind. The problem is that for this particular system, the measurement function “doesn’t care”, i.e. it will happily predict a measurement, but one that is *backwards*.

While there are problems with SCAAT cold starts in general, we have observed successful cold-start and recovery convergence in various simulations. For the cases where convergence will not occur, we currently see two possibilities. First, we feel that we should be able to come up with an elegant method for addressing the “backwards camera” problem completely within the framework of the SCAAT filter. This seems like the most attractive option. As a somewhat less attractive, but nonetheless reasonable option, we could implement a conventional method, e.g. Collinearity, to be used only in times of cold start or recovery. The SCAAT method would otherwise always be used in the steady-state operation of the system.

Figure 6.16 (page 150) presents the results of a cold start simulation that successfully converges. The chart shows the actual per-estimate RMS error (right axis, thicker line) and the filter error covariance elements (left axis, thinner lines) that correspond to the given state elements. I show the error standard deviations because they are more intuitive than covariances. Given 1000 Hz observations it takes a fraction of a second for the error covariances to converge at some steady-state values and the per-estimate RMS error to converge to near zero from over four meters initially.

For this simulation I used $\hat{x}(0) = 0$, $\hat{\alpha}(0) = 0$, $P(0)[x \ldots z] = 2[\text{ meters}]^2$, and $P(0)[\phi \ldots \psi] = 180[\text{ degrees}]^2$. The initial error covariance values need to be large (qualitatively) so that the filter will rely heavily on the measurements at first. The position values were chosen based on the dimensions of the simulated ceiling.)

6.2.8 Blocked Cameras

Similar to the case of cold start or recovery described in section 6.2.7, it is possible for the filter to weather a loss of measurements for some period of time, and then to recover when measurements are again available. While measurements are unavailable the Kalman filter offers predictions based on the last state update and the dynamic model.

For the UNC HiBall tracker, we envision these circumstances occurring when the HiBall cameras are not pointed at the ceiling, e.g. because the user's head is tilted to one side, or equivalently when the camera views are physically blocked, e.g. because the user places their hands over the lenses.

In a manner similar to that of figure 6.16, figure 6.17 (page 150) presents the results of a simulation where for two seconds the cameras are effectively blocked. While the cameras are blocked, the KF error covariance diagonals continue to grow as shown by the respective standard deviations (left axis) as do the KF error estimates as reflected in the growing per-estimate RMS error (right axis). When the cameras are unblocked, the filter quickly recovers in terms of the estimated and actual error.

6.2.9 Low Data Rates

The effect of the measurement rate on SCAAT stability is addressed in theory by the discussions of general stability in section 5.1.1 (page 107). However we wanted to have some idea of how the method is affected by low data rates in practice. In particular, we wanted to know "How slow can we go?" before the filter becomes unstable.

Our simulations seem to indicate that the overriding factor is the Nyquist rate associated with the bandwidth of the user motion. Given the 5 Hz low-pass filtering that I performed on the data sets (see "Motion Bandwidth" on page 199) I observed stability, albeit increased per-estimate RMS error, with measurement rates as low as 10 Hz. On the other hand, a simulation with a 7 Hz measurement rate quickly diverged.

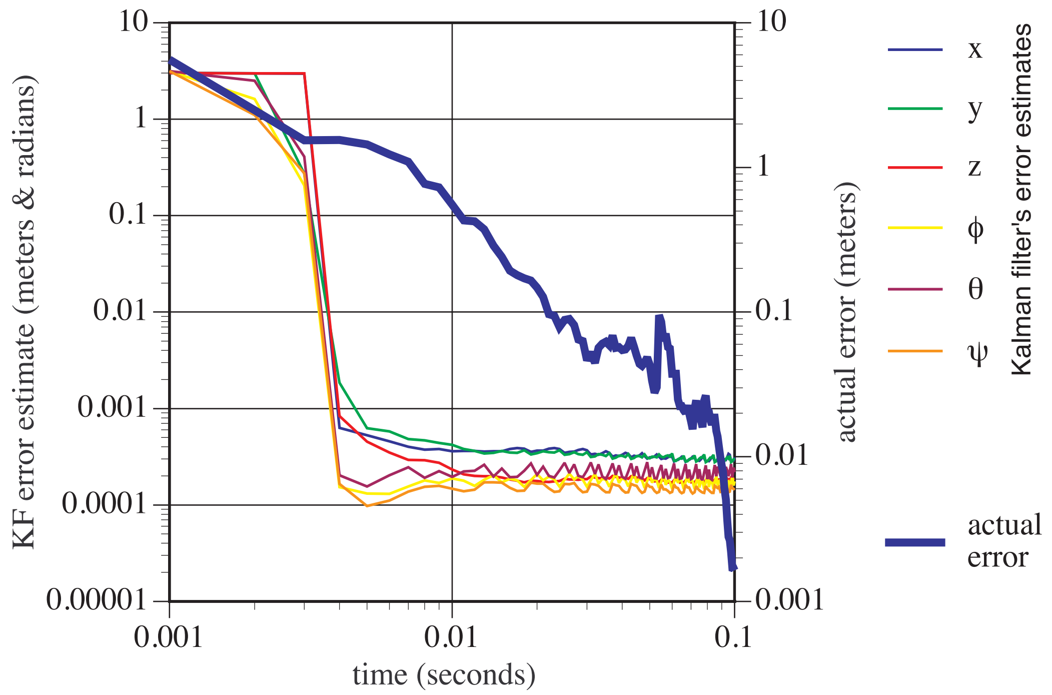


Figure 6.16: Cold start ("typical" data set). For this experiment I initialized the SCAAT filter state to the null vector and the error covariance elements to relatively large values: one-half the dimension of the ceiling in translation, 180 degrees in orientation.

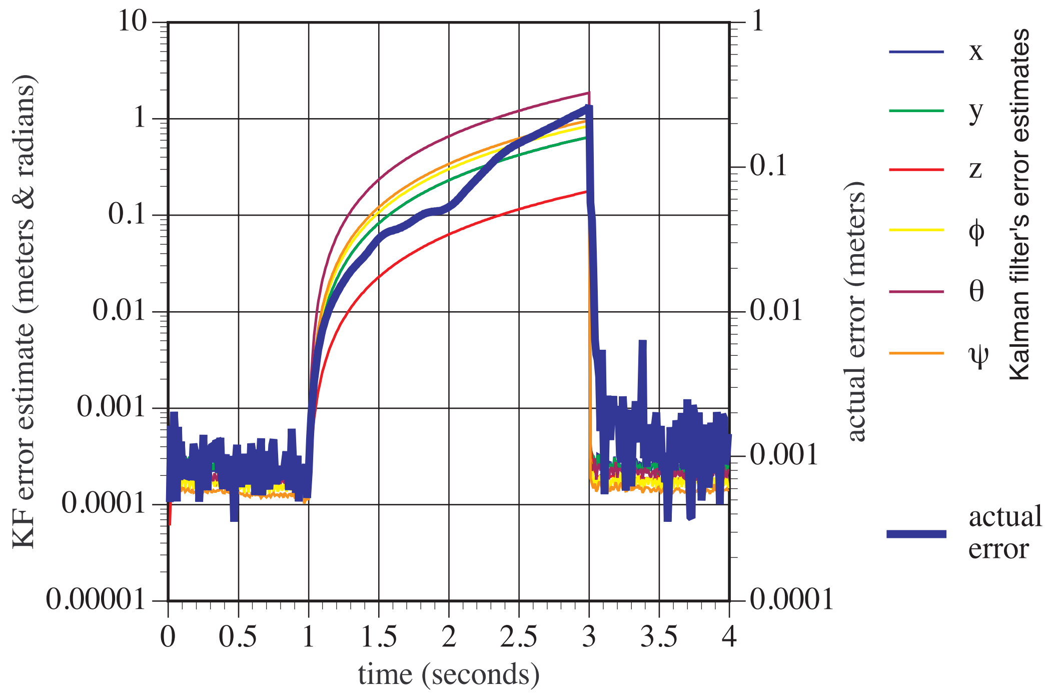


Figure 6.17: Blocked HiBall cameras ("typical" data set). At one second into the data set *all* of the cameras are blocked, and two seconds later they are all unblocked.

While I opted to omit quantitative information about the accuracy at low data rates, figure 6.18 presents quantitative error spectrum data. Clearly the systems with lower data rates suffer more in general in terms of noise—compare figure 6.18 with figure 6.9 on page 140. In figure 6.18 the 10 Hz simulations, both first and second pass through the data set, demonstrate an order of magnitude greater noise in the common area of the spectrum. It makes perfect sense that as the sampling rate approaches the user motion bandwidth the estimator becomes less effective.

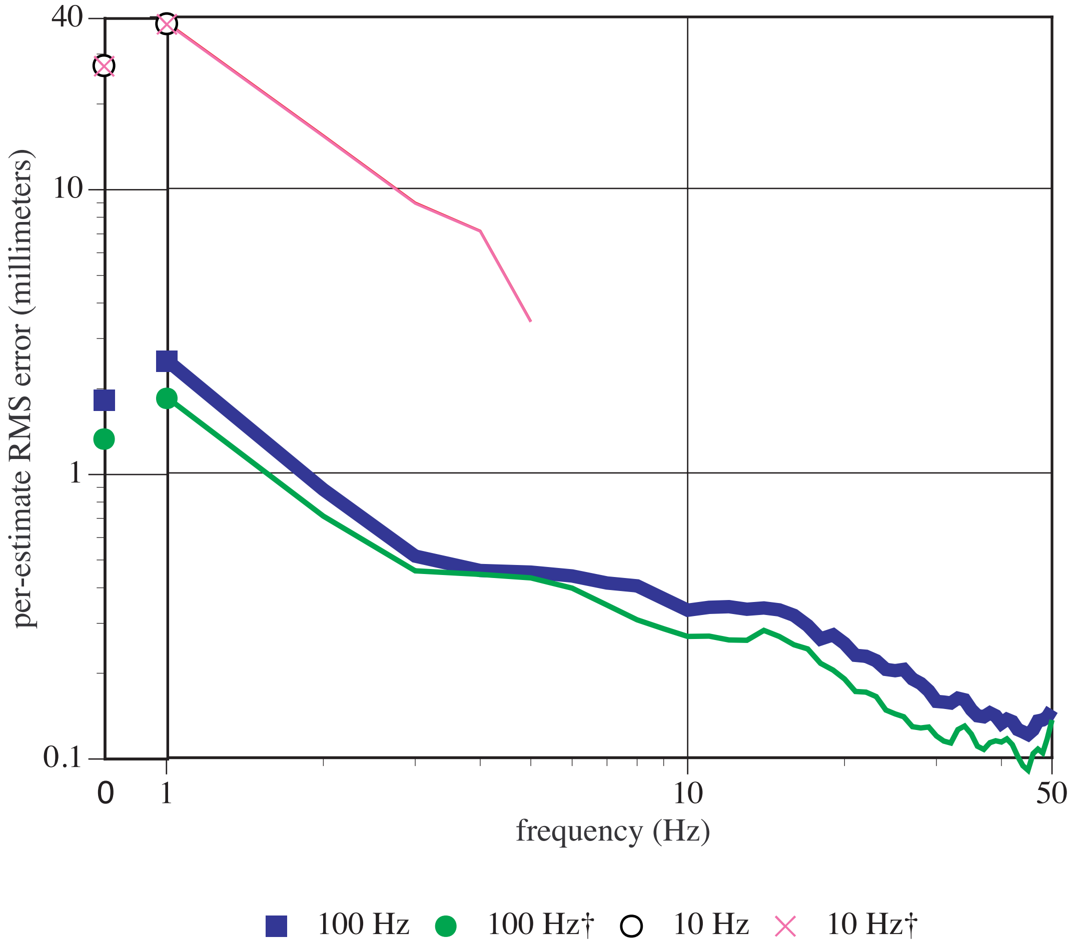


Figure 6.18: Error spectra for SCAAT with low sampling rates ("typical" data set, 1.7 mm initial beacon position error, *with* autocalibration). Note that the 10 Hz and 10 Hz† signals cannot be distinguished.

6.2.10 Low Dynamics

In choosing EKF parameters as described in section E.4 of appendix E (page 197) it was clear that the results would not be optimal for all user motion (optimal in terms of minimized error for a given dynamic model) but only for the particular motion data set used during the parameter search. Furthermore, it is likely that the dynamic model we present in section 4.2.1 and figure 4.1 (page 74) will also be appropriate for only certain types of VE motion.

This is not a problem that is unique to the subject of this dissertation—it spans a large area of research usually referred to as *system identification*. (See for example [Jacobs93] and [Maybeck70].) Because it is not an inherent SCAAT problem I only touch on it briefly here, leaving further investigations as suggested future work (see sections 7.1 and 7.3 of chapter 7, page 155).

To demonstrate the effectiveness of multiple parameter sets, I simulated a SCAAT run of the "still" data set discussed in section E.1 of appendix E (page 194) with and without specially optimized parameters. The results are given in figure 6.19.

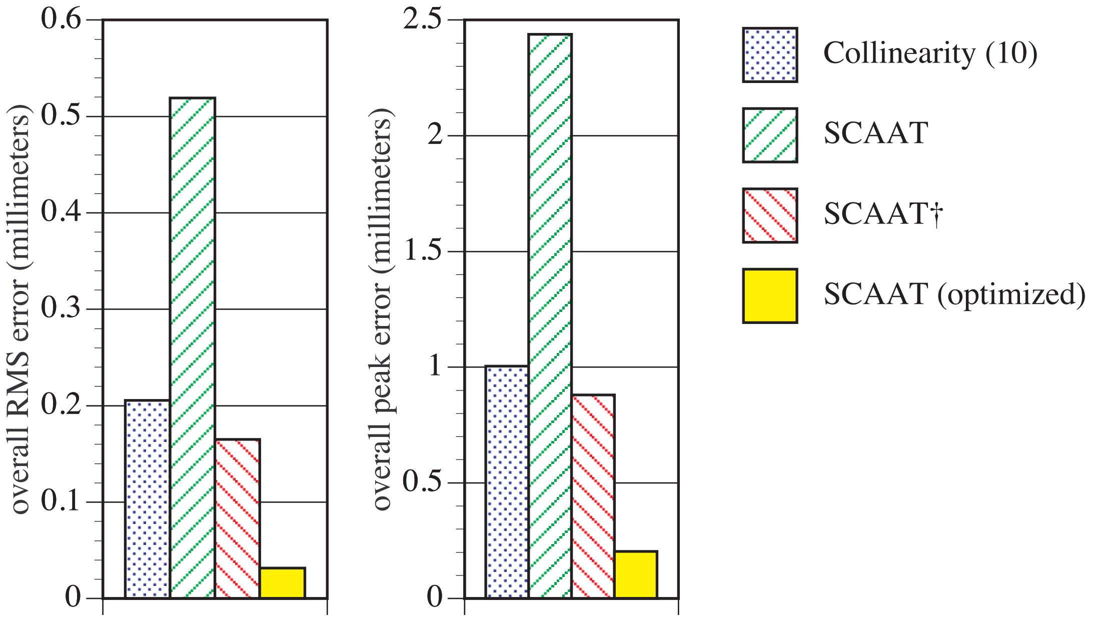


Figure 6.19: Collinearity vs. SCAAT *with* autocalibration ("still" data set). The third SCAAT run uses the specially optimized "still" parameters from table 6.8. The other two SCAAT runs, plain SCAAT and SCAAT†, used parameters given in table 6.6.

Notice in figure 6.19 that the initial SCAAT results are worse than the Collinearity results. We believe that this is a result of the averaging effect of the over-constrained Collinearity approach as the simultaneity assumption (see section 2.7 on page 63). In the case of low or no dynamics, the simultaneity assumption is reasonable, but in the face of dynamics it is not. Having said this, notice that the second SCAAT results are approximately the same as the Collinearity results. This demonstrates the improvement due to the autocalibration of the SCAAT approach. Finally notice the further (possibly unnoticeable) SCAAT improvements with a specially optimized parameter set.

Figure 6.20 illustrates the filtering by the SCAAT implementation under the conditions of very low user dynamics, with and without specialized parameters. Again the filter performs better when the parameters are optimized to match the actual dynamic conditions.

The results presented in figure 6.19 and figure 6.20 are not meant to be comprehensive, but to give some indication of the potential improvements from a multiple-model implementation with, for example, a specialized low user dynamics mode. It remains to be seen whether or not a multiple-model approach is warranted in practice for specific circumstances.

6.2.11 Single Scalar Measurements

Recall from "Single Scalar Measurements" on page 79 that a SCAAT Kalman filter should, in the purest sense, generate each new estimate with only a single scalar measurement from one source and sensor pair. Yet for my experiments I follow the "Criteria for Choosing the Measurements" on page 80 and present results for measurements and the corresponding models where $m_\sigma = 2$ scalar measurement elements, a (u, v) camera image coordinate.

I note here, only for reassurance, that I did indeed repeat the simulations of this chapter incorporating only a single scalar element at each measurement update, i.e. I designed the filter with $m_\sigma = 1$. The results, as might be expected, were better in terms of error and noise, but only to a *very* minor extent. Given the additional computational cost, I continue to recommend the heuristic presented on page 81.

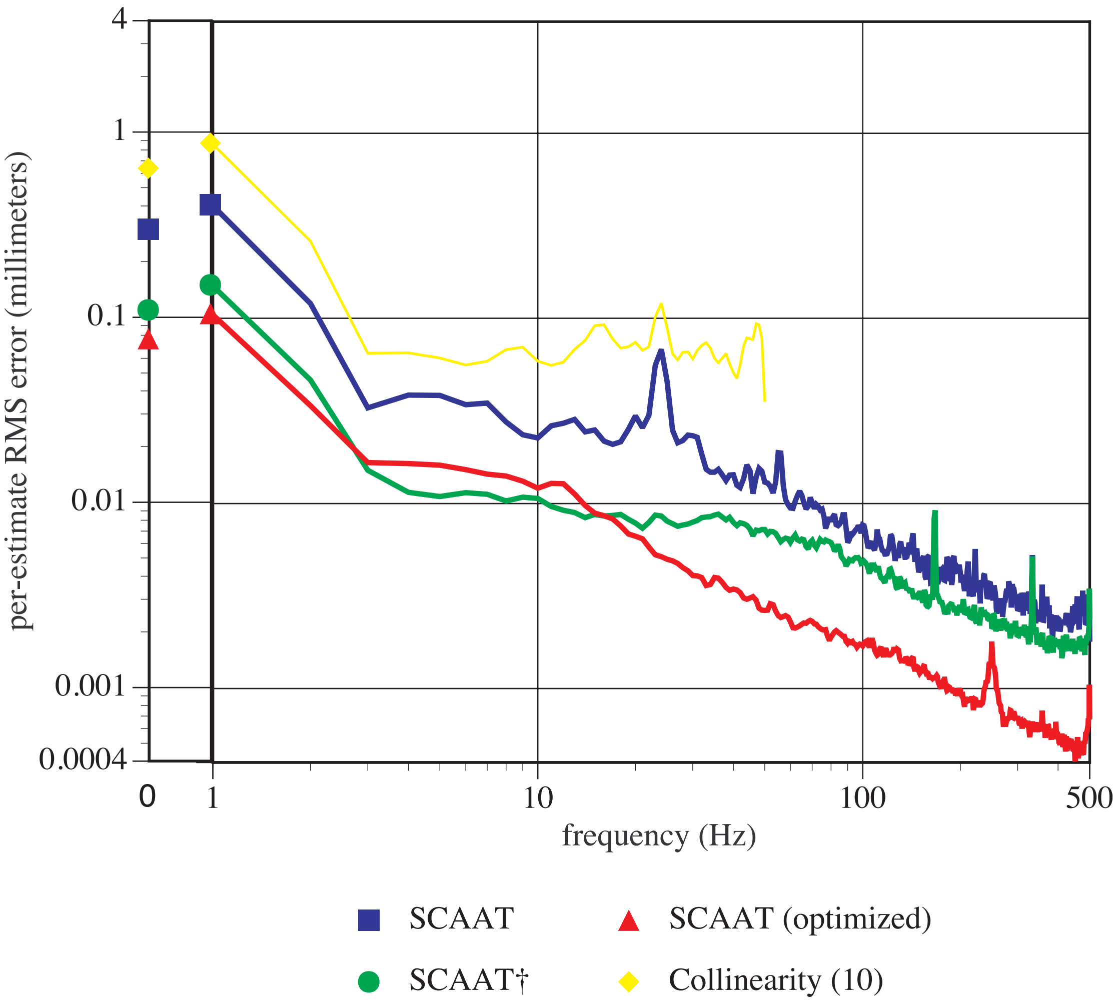


Figure 6.20: Error spectra for the "still" data set. The parameters for the SCAAT and SCAAT† runs were taken from table 6.6, i.e. they were *not* optimized for still motion. The remaining SCAAT run used specially optimized parameters. As with earlier experiments, the unusual peaks appear to result from regular sightings along directions with little information. The peaks disappear with random camera selection.

Chapter 7. Future Work

7.1 An Improved Dynamic Model

It is our opinion that the dynamic model presented in section 4.2.1 of chapter 4 (page 73) is possibly the simplest model one could reasonably imagine implementing. Our reasons for implementing such a simple model were twofold. First, our simulations (with the simple model) indicate that error is dominated by other phenomena (e.g. noise). Second, the simple model was convenient. Since the form of the dynamic model is not a key point for this dissertation I chose to implement the simplest one that we could get away with. (We sometimes joke that it's a testament to the Kalman filter that our simple model works at all!)

As such we wonder what model is the *right* model to use. Consider that the PV dynamic model presented in chapter 4 (see figure 4.1) assumes that the user's acceleration can be modeled as normally distributed white noise. Clearly if the acceleration is zero the assumption is invalid. (See the experiments of section 6.2.10 on page 152.) Likewise if the acceleration is constant but non-zero the assumption is invalid.

Some valid attempts have been made to choose dynamic models, sometimes based on reasonable assumptions about or observations of human motion, e.g. see [Azuma95, Liang91, So92], but we feel that there may be significant performance improvement hidden in an as-yet unknown better model for human motion.

7.2 Model Parameter Sensitivity

In section E.4 of appendix E (beginning on page 197) I presented a method for determining the SCAAT filter parameters used in the experiments of chapter 6, and in tables 6.4-6.6 (page 127) I presented some specific results. However, based on our experiences using Powell's method to determine the dynamic model parameters, we remain curious about how slight changes in those parameters would affect the filter performance. While monitoring the progress of the parameter search, we noticed large improvements (error reductions) with some parameter changes, and almost no change with others.

As such, no matter what the configuration of the dynamic models (section 7.1) it would be interesting to evaluate the sensitivity of a particular system to changes in the dynamic model parameters. Such an assessment might lead to a different configuration, or even a change in the tracking system itself.

7.3 An Adaptive or Multiple-Model Filter

Because the activities relating to interactive computer graphics range from sitting essentially still, e.g. looking at a computer monitor, to relatively fast swinging of arms or even jogging, e.g. a simulator for soldiers, I feel that there may be tremendous advantage to a *multiple-model* filter approach [Bar-Shalom93]. Such a multiple-model filter could be implemented statically where one particular model is chosen one-time just prior to a tracking session, or dynamically where one or more filters monitor the user dynamics and switch models as necessary. The switching could involve changes in model *form* or simply model *parameters*, and it could be discreet or piece-wise continuous in nature. (Again, see [Bar-Shalom93] for a thorough discussion of this topic.)

One source of information that is often used to make switching decisions is the filter's residual sequence $\overrightarrow{\Delta z}$ from equation (4.19) on page 98. Often, the characteristics of the residual stream as observed over some finite window of time can indicate the validity of the current dynamic model or parameter set. If several models or sets are used in parallel, a separate process can monitor all of the residuals and at every step choose the estimate from the one filter that seems most likely, based on the size of the residual as compared to the filter's current error covariance.

A somewhat unusual or tricky aspect of the SCAAT method with respect to such residual monitoring is that the ongoing sequence of residuals is based on a constantly changing set of sources and sensors. In other words, the filter employs a family of measurement systems as opposed to a single measurement system. Because a constant overall filter bias, e.g. that resulting from an improper dynamic model, will affect the individual measurement systems in different ways, biases that are clearly evident for each individual measurement system can begin to appear normal (Gaussian) when examined collectively!

This problem is related to the *central limit theorem* of probability theory [Kelly94, Maybeck70]. Informally the theorem says that the distribution of a sum of random variables will approach a normal (Gaussian) distribution as the number of individual random variables grows. Somewhat surprisingly, this phenomenon actually begins to occur with even a small number of individual random variables, and is true no matter what their individual distributions.

Therefore residual monitoring schemes applied to a SCAAT filter must take care to monitor a separate residual sequence for each measurement system, i.e. source/sensor pair, as opposed to the single residual sequence resulting from equation (4.19) as would be the case for a standard Kalman filter. Otherwise, it is possible that an overall filter bias may go undetected as the very act of switching between measurement systems tends to "whiten" the collective residual sequence of equation (4.19).

7.4 A Better Estimator

Although I have not taken the opportunity to more thoroughly examine their recent publications, I am very interested in recent work by Julier and Uhlmann at Oxford University. In [Julier95] the authors convincingly argue that the linearization approach of the EKF can actually lead to significant problems for a vehicle (target) undergoing certain dynamics. They then present a new approach to non-linear filtering, an approach that they claim is "...more accurate, more stable, and far easier to implement than an extended Kalman filter."

Their approach, if I understand it correctly, is to essentially pass around, through the estimator, complete "miniature" sample sets for the distribution in question. By transforming the actual values in the sets, the non-linearities are preserved. The necessary statistics are then extracted from the sample sets as needed. In contrast, these non-linearities are lost in the EKF as it is implemented using a truncated Taylor series to approximate the non-linear function.

I think that this work warrants further investigation, as it may offer noticeable improvements resulting from a better and possibly even faster (computationally) estimator. The potential improvements discussed above in sections 7.1 and 7.3 still apply to this method also.

7.5 Source and Sensor Ordering

As discussed in section 4.6 on page 106, it might be possible to improve performance and flexibility by dynamically altering the individual source/sensor selection strategy. For example, in the HiBall simulations of chapter 6 we observed peaks in the error spectra that we believe are induced by repeated visitation of source/sensor pairs that do not provide useful information. See figure 6.10 on page 141 and the corresponding discussion in "Error Spectra" on page 137. This strategy could also prove useful in other systems, e.g. hybrid tracking or estimation systems.

I believe that relatively fast eigenvector and eigenvalue determination strategies exist and could be applied to this problem as discussed in section 4.6. In any case, the computations may not be necessary at each step but might be performed periodically to choose a new selection strategy. Furthermore it may be possible to compute the eigenvectors and eigenvalues one time at initialization and then propagate them along at the same time as the Kalman filter error covariance matrix, using the same or similar mathematics.

7.6 Solving a System of Simultaneous Equations

Given a set of n unknowns, and a set of m equations (linear or nonlinear) that simultaneously describe a known relationship between the unknowns, several methods exist for solving for the unknowns. For several examples, see [Press90]. The Kalman filter can also be used to estimate the unknowns by mapping the given equations and constraints into the Kalman filter framework.Furthermore, I believe that it would be interesting to investigate the application of the SCAAT method to this problem.

For example, consider a linear system of equations

$$A\vec{x} = \vec{b}.$$

where $\vec{x}$, $\vec{b}$, and A have dimensions n, n, and $n \times n$ respectively. In this case one could use the vector $\vec{x}$ as the state vector, the vector $\vec{b}$ as the measurement vector, and the matrix A as the measurement matrix. Then as with equation (4.10), one could define the measurement models as

$$\hat{b}[j] = A[j, \bullet]\hat{x} + \vec{v}[j] \tag{7.1}$$

for all $j = 1, 2, \ldots, n$. (The notation $A[j, \bullet]$ is used to indicate row j of matrix A.) In this case, each corresponding noise element $\vec{v}[j]$ could be used to represent the uncertainty in the *actual* constraint $\vec{b}[j]$. Because the unknowns are (presumably) constant, the state transition matrix would be the identity matrix, and the process noise would likely be very small and constant.

Note that for this simple example one could indeed perform the estimation in batch mode, that is by using equation (7.1) to estimate the n elements of $\vec{b}$, then computing the n-dimensional residual $\overrightarrow{\Delta b} = \vec{b} - \hat{b}$, and using that to update the filter. However, I believe that there might be case where one wants to estimate using fewer than n elements at a time. For example, one might want to trade-off computation time for convergence time in order to reach some relatively quick approximation of $\vec{x}$.

Furthermore if the elements of $\vec{b}$ are changing over time, one could implement a "smart" ordering scheme in the spirit of section 7.5. One could for example maintain a sorted list of the residuals (the individual elements) from $\overrightarrow{\Delta b} = \vec{b} - \hat{b}$, and at each update incorporate only those constraints (equations) with the largest corresponding residuals, perhaps even just a single constraint.

7.7 Other Tracking Implementations

Finally, I would like to see the SCAAT algorithm implemented in some other tracking systems beyond the UNC HiBall tracking system, for example the systems listed in table 2.2 on page 63. In particular, I would be interested in seeing the method applied to (1) magnetic tracking systems, e.g. the Polhemus Fastrak and the Ascension Bird, (2) a GPS navigation system, and (3) an computer-vision-based system. With respect to the magnetic systems, the implementation could be difficult because it's not clear what level of measurement control or insight the general user has. A GPS implementation shows more promise as many modern receivers provide the user with access to relatively low-level measurement information [Kaplan96]. Finally, a computer-vision based system may be in the future as we (Bishop, Chi and Welch) are currently in the process of proposing a new inertial/optical hybrid tracker.

APPENDIX A. THE SIMULTANEITY ASSUMPTION

A.1 An Example 2D Tracker

Let us evaluate a hypothetical two-dimensional inside-looking-out optoelectronic tracker as shown below in figure A.1. In this case, the target to be tracked is a platform with two one-dimensional pinhole cameras (sensors) mounted on it, side by side, both looking out at a beacon that is fixed in the environment at point $\vec{b}$. Let's assume that we are interested in estimating the platform's true position $\vec{x} = [x, y]$ as it moves about in two dimensions.

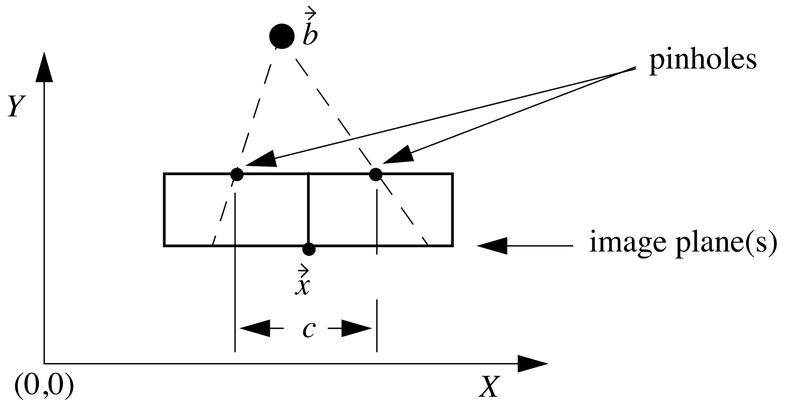


Figure A.1: A simple 2D inside-looking-out optoelectronic tracker. The diagram represents a 2D tracking application where a platform with origin $\vec{x} = [x, y]$ moves about in X and Y. On the platform are two 1D pinhole cameras, side by side with baseline c, and a fixed beacon at $\vec{b} = [x_b, y_b]$.

Each of the two pinhole cameras mounted on the platform has an origin that is distance $c/2$ from the platform origin $\vec{x} = [x, y]$ as shown in figure A.2. An observation of either camera consists of measuring the point along the respective camera's 1D image axis at which the image of the beacon at $\vec{b}$ impinges. We will refer to this point as u_1 for camera 1, and u_2 for camera 2 as shown in figure A.2. Each of these measurements is taken with respect to corresponding camera's origin.

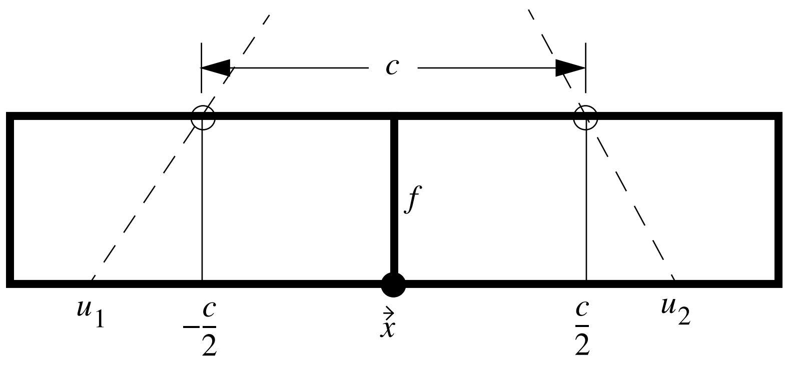


Figure A.2: A close up of the 2D target platform and the two 1D pinhole cameras. The focal length f is the same for both cameras, the origins of the cameras with respect to the platform origin $\vec{x} = [x, y]$ are $-c/2$ and $c/2$, and the camera coordinate axis extend right and up. The image of the beacon $\vec{b}$ along each 1D camera's image plane is at u_1 and u_2 respectively.

Assuming that we know the fixed position $\vec{b}$ of the beacon, the common focal length f of the cameras, and the baseline c between the cameras, we can measure the values u_1 and u_2, and then compute the position $\vec{x} = [x, y]$ of our platform:

$$x = x_b - \left(\frac{c}{2}\right)\left(\frac{u_1 + u_2}{u_1 - u_2}\right), \tag{A.1}$$

$$y = y_b - f + \frac{f}{u_2}\left(x_b - x - \frac{c}{2}\right). \tag{A.2}$$

A.2 The Target in Motion

Referring back to figure A.1 and figure A.2, given a true platform position $\vec{x} = [x, y]$, the fixed position $\vec{b} = [x_b, y_b]$ of the beacon, the common focal length f of the cameras, and the baseline c between the cameras, we can arrive at closed-form equations for the observed camera measurements u_1 and u_2:

$$u_1 = \left(\frac{f}{2}\right)\left(\frac{2x - 2x_b - c}{y_b - y - f}\right), \tag{A.3}$$

$$u_2 = \left(\frac{f}{2}\right)\left(\frac{2x - 2x_b + c}{y_b - y - f}\right). \tag{A.4}$$

Using these equations we can illustrate the effect of the simultaneity assumption on our 2D tracker by simulating a position estimate while making the assumption. If we assert that u_1 is measured first, then equation (A.3) can remain as is. If it takes τ_m seconds to measure u_1, then we would be measuring u_2 not at position $\vec{x}(t)$ but at position $\vec{x}(t+\tau_m)$ as shown in figure A.3.

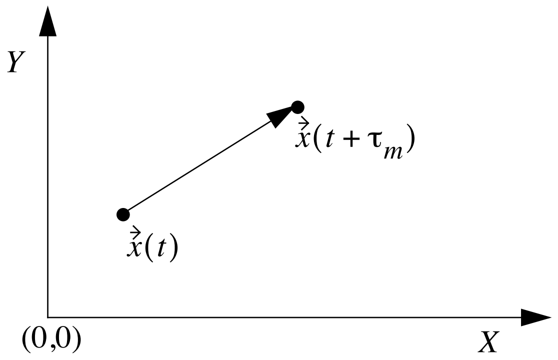


Figure A.3: Sample track. Example motion undergone during time τ_m by our simple 2D tracker. The scale is exaggerated for illustration of the point.

If during time τ_m we are moving with linear velocities $\dot{x}$ and $\dot{y}$, then u_2 measured at time $t+\tau_m$ becomes

$$u_2 = \left(\frac{f}{2}\right)\left(\frac{2(x+\dot{x}\tau_m)-2x_b+c}{y_b-(y+\dot{y}\tau_m)-f}\right). \tag{A.5}$$

For the purpose of our example let us fix $f = 0.035$ [meters], $c = 0.2$ [meters], and $\vec{b} = [1, 2]$ [meters]. Then let's assume that the object being tracked is moving diagonally with $\dot{x} = \dot{y} = 0.5$ [meters/second], and that the measurement time is $\tau_m = 10$ ms. At the time of the first beacon observation, u_1, let's say that the target is located at

$$\vec{x}(t) = [1, 1] \text{ [meters]}. \tag{A.6}$$

Then at the time of the second beacon observation, u_2, the target has moved to the position

$$\vec{x}(t+\tau_m) = [1.005, 1.005] \text{ [meters]}. \tag{A.7}$$

In other words, the target has moved 5 mm in *x* and *y* between observations u_1 and u_2. So what would be the actual measured values of u_1 and u_2 given the preceding conditions? Using equations (A.3) and (A.5) we see that the tracker would have measured

$$u_1 = -3.627e-3 \text{ [meters], and}$$

$$u_2 = 3.828e-3 \text{ [meters].}$$

These numbers represent the observations that would have occurred under the given circumstances. Given these observations we can then use equations (A.1) and (A.2) to see what our estimate would have been:

$$x = 1.003 \text{ [meters], and} \tag{A.8}$$

$$y = 1.026 \text{ [meters],} \tag{A.9}$$

in other words,

$$\vec{x}(t + \tau_m) = [1.003, 1.026] \text{ [meters].} \tag{A.10}$$

Comparing equation (A.10) with equations (A.6) and (A.7) we see significant differences. Our estimate in y was off by almost three centimeters when compared to the true position at either time t or $t + \tau_m$. I like to think of this error as a "distortion" introduced by the simultaneity assumption.

To further observe the effect of this distortion we can substitute equations (A.3) and (A.5) into equations (A.1) and (A.2) to obtain closed-form solutions for the distorted x and y position estimates. In figure A.4 I have plotted the distorted x and y estimates versus τ_m from 0 to 100 milliseconds for the given conditions, and in figure A.5 a parametric version showing the distorted x versus the distorted y (i.e. the distorted position) for the same range of τ_m.

A.3 Further Illustrations

For this particular simulation, the problems caused by the simultaneity assumption are even more evident when the platform undergoes acceleration. figure A.6 below contains a family of curves that show how the x position estimates become skewed as the platform undergoes one second of motion with $\dot{x} = \dot{y} = \sin(2\pi(time/1000))$ [meters/second]. The "truth" curve with $\tau_m = 0$ resembles $x = -\cos(2\pi(time/1000)) + C$ for some constant C as would be expected when integrating a sinusoidal velocity. Note the skewing of x with measurement delay $\tau_m = \{0, 10, 40, 70, 100\}$ milliseconds.

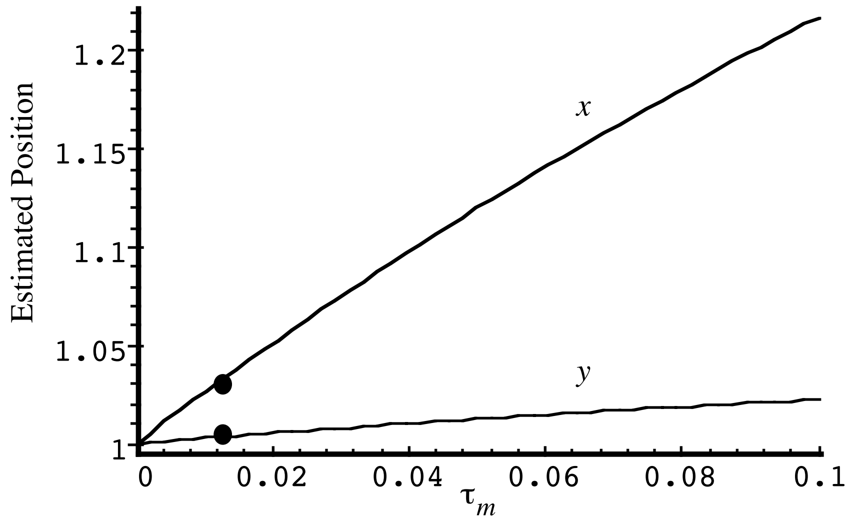


Figure A.4: Estimated position versus measurement delay. The position estimate (in meters) versus the time τ_m between observations u_1 and u_2. Note that if τ_m is 0, then the simultaneity assumption is valid and the estimated position is the true position [1,1]. However as the measurement time increases, the final estimates for x and y become distorted as shown for delays from 0 to 100 milliseconds. The solid dots show the distorted positions reported by equations (A.8) and (A.9).

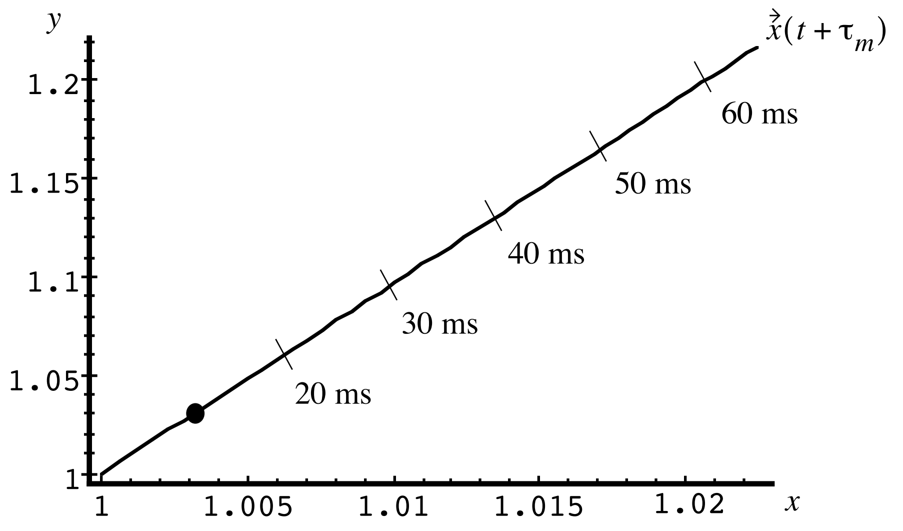


Figure A.5: A parametric version of figure A.4. The plot depicts the 2D position estimate $\vec{x}(t+\tau_m)$ parametric in τ_m from 0 to 65 milliseconds. The solid dot marks the distorted position estimate given by equation (A.10). Again if τ_m is 0 (lower left on curve) then the simultaneity assumption is correct and the estimate position is the true position [1,1]. It is not shown, but with a measurement time of 100 milliseconds the estimate is off by approximately two centimeters in x, and two *decimeters* in y.

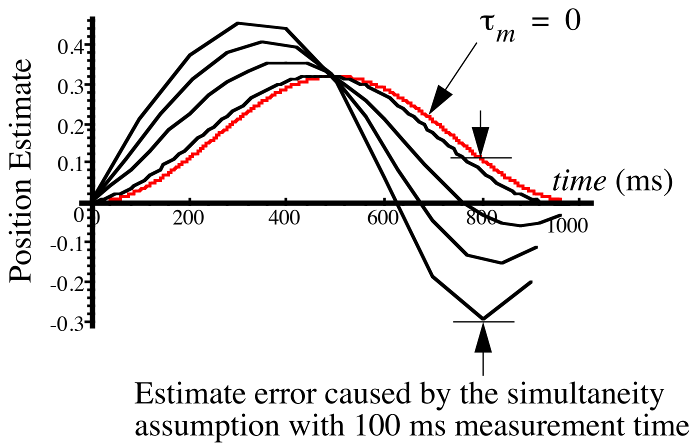


Figure A.6: Error caused by the simultaneity assumption. The family of curves shows how simulated position estimates become skewed by the simultaneity assumption as a target undergoes one second of motion with sinusoidal velocity. Note the skewing of the estimate with sensor measurement times of $\tau_m \in \{0, 10, 40, 70, 100\}$ milliseconds.

This example was meant to highlight the distortion caused by the simultaneity assumption. It should be apparent that as measurement times increase, the SCAAT method appears more and more attractive in terms of avoiding the distortion caused by the simultaneity assumption.

APPENDIX B. THE KALMAN FILTER

This appendix is a copy of a UNC technical report written by Welch and Bishop in 1995 [Welch95]. It is included to provide a ready and accessible introduction to both the discrete Kalman filter and the extended Kalman filter.

B.1 The Discrete Kalman Filter

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [Kalman60]. Since that time, due in large part to advances in digital computing, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. A very "friendly" introduction to the general idea of the Kalman filter can be found in Chapter 1 of [Maybeck70], while a more complete introductory discussion can be found in [Sorenson70], which also contains some interesting historical narrative. More extensive references include [Jacobs93, Gelb74, Maybeck70, Lewis86, and Brown92]

B.1.1 The Process to be Estimated

The Kalman filter addresses the general problem of trying to estimate the state $\vec{x} \in \Re^n$ of a first-order, discrete-time controlled process that is governed by the linear difference equation

$$\vec{x}_{k+1} = A_k \vec{x}_k + B\vec{u}_k + \vec{\omega}_k , \tag{B.1}$$

with a measurement $\vec{z} \in \Re^m$ that is

$$\vec{z}_k = H_k \vec{x}_k + \vec{v}_k . \tag{B.2}$$

The random variables $\vec{\omega}_k$ and $\vec{v}_k$ represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(\vec{\omega}) \sim N(0, Q), \tag{B.3}$$

$$p(\vec{v}) \sim N(0, R). \tag{B.4}$$

The $n \times n$ matrix A in the difference equation (B.1) relates the state at time step k to the state at step k+1, in the absence of either a driving function or process noise. The $n \times l$ matrix B relates the control input $u \in \Re^l$ to the state $\vec{x}$. The $m \times n$ matrix H in the measurement equation (B.2) relates the state to the measurement z_k.

B.1.2 The Computational Origins of the Filter

We define $\hat{x}_k^- \in \Re^n$ (note the "super minus") to be our *a priori* state estimate at step k given knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be our *a posteriori* state estimate at step k given measurement $\vec{z}_k$. We can then define *a priori* and *a posteriori* estimate errors as

$$\begin{aligned} \vec{\xi}_k^- &\equiv \vec{x}_k - \hat{x}_k^-, \text{ and} \\ \vec{\xi}_k &\equiv \vec{x}_k - \hat{x}_k. \end{aligned} \tag{B.5}$$

The *a priori* estimate error covariance is then

$$P_k^- = E\left\{\vec{\xi}_k^- \vec{\xi}_k^{-T}\right\}, \tag{B.6}$$

where $E\{\bullet\}$ denotes mathematical expectation, and the *a posteriori* estimate error covariance is

$$P_k = E\{\vec{\xi}_k \vec{\xi}_k^T\}. \tag{B.7}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement $\vec{z}_k$ and a measurement prediction $H_k\hat{x}_k^-$ as shown in equation (B.8). Some justification for equation (B.8) is given in section B.1.3.

$$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - H_k\hat{x}_k^-) \tag{B.8}$$

The difference vector $(\vec{z}_k - H_k\hat{x}_k^-)$ in equation (B.8) is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $H_k\hat{x}_k^-$ and the actual measurement $\vec{z}_k$. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K in equation (B.8) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance of equation (B.7). This minimization can be accomplished by first substituting equation (B.8) into the above definition for $\vec{e}_k$, substituting that into equation (B.7), performing the indicated expectations, taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for K. For more details see [Jacobs93, Maybeck70, and Brown92]. One form of the resulting K that minimizes equation (B.7) is given by[*]

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ &= \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \end{aligned} \quad . \tag{B.9}$$

Looking at equation (B.9) we see that as the measurement error covariance R_k approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H_k^{-1} .$$

On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K_k = 0 .$$

Another way of thinking about the weighting by K is that as the measurement error covariance R_k approaches zero, the actual measurement $\vec{z}_k$ is "trusted" more and more,

[*] All of the Kalman filter equations can be algebraically manipulated into to several forms. Equation (B.9) represents the Kalman gain in one popular form.

while the predicted measurement $H_k\hat{x}_k^-$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero the actual measurement $\vec{z}_k$ is trusted less and less, while the predicted measurement $H_k\hat{x}_k^-$ is trusted more and more.

B.1.3 The Probabilistic Origins of the Filter

The justification for equation (B.8) is rooted in the probability of the *a priori* estimate $\hat{x}_k^-$ conditioned on all prior measurements $\vec{z}_k$ (Bayes' rule). For now let it suffice to point out that the Kalman filter maintains the first two moments of the state distribution,

$$\begin{aligned} E\{\vec{x}_k\} &= \hat{x}_k \\ E\{(\vec{x}_k - \hat{x}_k)(\vec{x}_k - \hat{x}_k)^T\} &= P_k. \end{aligned}$$

The *a posteriori* state estimate equation (B.8) reflects the mean (the first moment) of the state distribution— it is normally distributed if the conditions of equations (B.3) and (B.4) are met. The *a posteriori* estimate error covariance equation (B.7) reflects the variance of the state distribution (the second non-central moment). In other words,

$$\begin{aligned} p(\vec{x}_k | \vec{z}_k) &\sim N(E\{\vec{x}_k\}, E\{(\vec{x}_k - \hat{x}_k)(\vec{x}_k - \hat{x}_k)^T\}) \\ &= N(\hat{x}_k, P_k). \end{aligned}.$$

For more details on the probabilistic origins of the Kalman filter, see [Jacobs93, Maybeck70, and Brown92].

B.1.4 The Discrete Kalman Filter Algorithm

We will begin this section with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter (see the previous footnote). After presenting this high-level view, we will narrow the focus to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown in figure B.1.

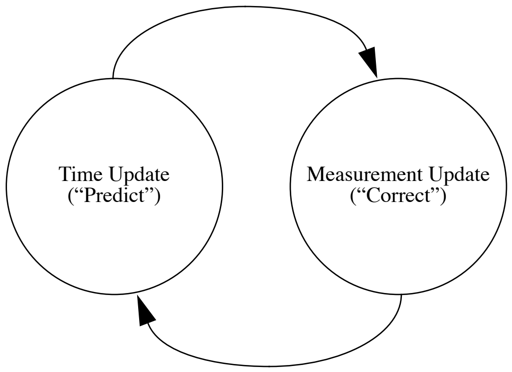


Figure B.1: The ongoing discrete Kalman filter cycle. The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time. Notice the resemblance to a *predictor-corrector* algorithm

The specific equations for the time and measurement updates are presented in table B.1 and table B.2. Again notice how the time update equations in table B.1 project the state and covariance estimates from time step k to step k+1. A_k and B are from equation (B.1), while Q_k is from equation (B.3). Initial conditions for the filter are discussed in the earlier references.

Table B.1: Discrete Kalman filter *time* update equations.

$$\hat{x}^-_{k+1} = A_k \hat{x}_k + B\vec{u}_k \qquad \text{(B.10)}$$

$$P^-_{k+1} = A_k P_k A_k^T + Q_k \qquad \text{(B.11)}$$

Table B.2: Discrete Kalman filter *measurement* update equations.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{B.12}$$

$$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - H_k \hat{x}_k^-) \tag{B.13}$$

$$P_k = (I - K_k H_k) P_k^- \tag{B.14}$$

The first task during the measurement update is to compute the Kalman gain, K_k. Notice that the equation given here as equation (B.12) is the same as equation (B.9). The next step is to actually measure the process to obtain $\vec{z}_k$, and then to generate an *a posteriori* state estimate by incorporating the measurement as in equation (B.13). Again equation (B.13) is simply equation (B.8) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via equation (B.14).

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible than (for example) an implementation of a Weiner filter which is designed to operate on *all* of the data *directly* for each estimate [Brown92]. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure B.2 offers a complete picture of the operation of the filter, combining the high-level diagram of figure B.1 with the equations from table B.1 and table B.2.

B.1.5 Filter Parameters and Tuning

In the actual implementation of the filter, each of the measurement error covariance matrix R_k and the process noise Q_k, given by equations (B.4) and (B.3) respectively, might be measured prior to operation of the filter. In the case of the measurement error covariance R_k in particular this makes sense—because we need to be able to measure the process (while operating the filter) we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement error.

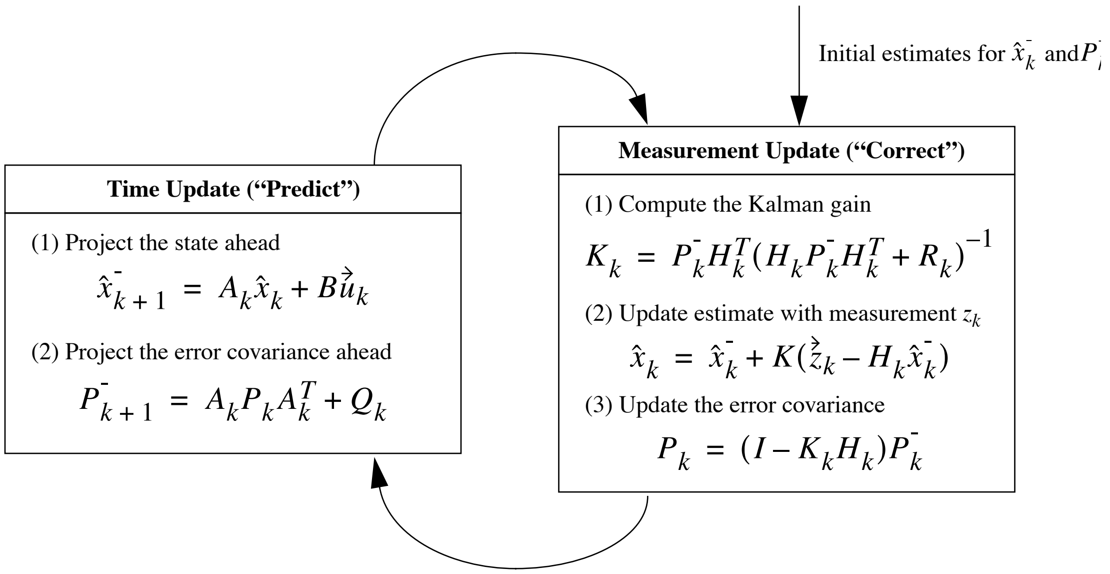

Initial estimates for $\hat{x}_k^-$ and P
Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_{k+1}^- = A_k\hat{x}_k + B\vec{u}_k$
(2) Project the error covariance ahead
$P_{k+1}^- = A_kP_kA_k^T + Q_k$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^-H_k^T(H_kP_k^-H_k^T + R_k)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - H_k\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_kH_k)P_k^-$

Figure B.2: The complete Kalman filter algorithm. This figure combines the high-level diagram of figure B.1 with the equations from table B.1 and table B.2.

In the case of Q_k, often times the choice is less deterministic. For example, this noise source is often used to represent the uncertainty in the process model equation (B.1). Sometimes a very poor model can be used simply by "injecting" enough uncertainty via the selection of Q_k. Certainly in this case one would hope that the measurements of the process would be reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by "tuning" the filter parameters Q_k and R_k. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter.

In closing we note that under conditions where Q_k and R_k .are constant, both the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant (see the filter update equations in figure B.2). If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by solving equation (B.11) for the steady-state value of P_k by defining $P_k^- \equiv P_k$ and solving for P_k.

It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting beacons in our optoelectronic tracker ceiling panels, the noise in measurements of nearby beacons will be smaller than that in far-away beacons. Also, the process noise Q_k is sometimes changed dynamically during filter operation in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of Q_k if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such a case Q_k can be used to model not only the uncertainty in the model, but also the uncertainty of the user's intentions.

B.2 The Extended Kalman Filter (EKF)

B.2.1 The Process to be Estimated

As described above in section B, the Kalman filter addresses the general problem of trying to estimate the state $\vec{x} \in \Re^n$ of a first-order, discrete-time controlled process that is governed by a *linear* difference equation. But what happens if the process to be estimated and (or) the measurement relationship to the process is non-linear? Some of the most interesting and successful applications of Kalman filtering have been such situations. A Kalman filter that linearizes about the current mean and covariance is referred to as an *extended Kalman filter* or EKF.†

In something akin to a Taylor series, we can linearize the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships. To do so, we must begin by

† A fundamental "flaw" of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply an ad hoc state estimator that only approximates the optimality of Bayes' rule by linearization. Some very interesting work has been done by Julier et al. in developing a variation to the EKF, using methods that preserve the normal distributions throughout the non-linear transformations [Julier95].

modifying some of the material presented in section B. Let us assume that our process again has a state vector $\vec{x} \in \Re^n$, but that the process is now governed by the *non-linear* difference equation

$$\tilde{x}_{k+1} = f(\vec{x}_k, \vec{u}_k, \vec{\omega}_k), \tag{B.15}$$

with a measurement $\vec{z} \in \Re^m$ that is

$$\tilde{z}_k = h(\vec{x}_k, \vec{v}_k). \tag{B.16}$$

Again the random variables $\vec{\omega}_k$ and $\vec{v}_k$ represent the process and measurement noise as in equations (B.3) and (B.4).

In this case the *non-linear* function *f*(•) in the difference equation equation (B.15) relates the state at time step *k* to the state at step *k*+1. It includes as parameters any driving function $\vec{u}_k$ and the process noise $\vec{\omega}_k$. The *non-linear* function *h*(•) in the measurement equation (B.16) now relates the state to the measurement $\vec{z}_k$.

B.2.2 The Computational Origins of the Filter

To estimate a process with non-linear difference and measurement relations, we begin by writing new governing equations that linearize about equations (B.15) and (B.16),

$$\vec{x}_{k+1} = \tilde{x}_{k+1} + \Xi(\vec{x}_k - \hat{x}_k) + W\vec{\omega}_k, \tag{B.17}$$

$$\vec{z}_k = \tilde{z}_k + H(\vec{x}_k - \hat{x}_k) + V\vec{v}_k. \tag{B.18}$$

where

- $\tilde{x}_{k+1}$ and $\tilde{z}_k$ are the projected state and measurement vectors from equations (B.15) and (B.16),
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step *k*,
- the random variables $\vec{\omega}_k$ and $\vec{v}_k$ represent the process and measurement noise as in equations (B.3) and (B.4).
- Ξ is the Jacobian matrix of partial derivatives of *f*(•) with respect to $\vec{x}_k$, that is

$$\Xi[i, j] = \frac{\partial}{\partial \vec{x}_k[j]} f[i](\vec{x}_k, \vec{u}_k, 0), \tag{B.19}$$

- W is the Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to $\vec{\omega}_k$,

$$W[i, j] = \frac{\partial}{\partial \vec{\omega}_k[j]} f[i](\vec{x}_k, \vec{u}_k, 0), \tag{B.20}$$

- H is the Jacobian matrix of partial derivatives of $h(\bullet)$ with respect to $\vec{x}_k$,

$$H[i, j] = \frac{\partial}{\partial \vec{x}_k[j]} h[i](\vec{x}_k, 0), \tag{B.21}$$

- V is the Jacobian matrix of partial derivatives of $h(\bullet)$ with respect to $\vec{v}_k$,

$$V[i, j] = \frac{\partial}{\partial \vec{v}_k[j]} h[i](\vec{x}_k, 0). \tag{B.22}$$

Now we define a new notation for the prediction error,

$$\tilde{\xi}_{x_k} \equiv \vec{x}_k - \tilde{x}_k, \tag{B.23}$$

and the measurement residual,

$$\tilde{\xi}_{z_k} \equiv \vec{z}_k - \tilde{z}_k. \tag{B.24}$$

Using equations (B.23) and (B.24) we can rewrite equations (B.17) and (B.18) as follows,

$$\tilde{\xi}_{x_k} \approx \Xi(\vec{x}_k - \hat{x}_k) + \vec{\varepsilon}_k, \tag{B.25}$$

$$\tilde{\xi}_{z_k} \approx H\tilde{\xi}_{x_k} + \vec{\eta}_k, \tag{B.26}$$

where $\vec{\varepsilon}_k$ and $\vec{\eta}_k$ are sets (ensembles) of independent random variables having zero mean and covariance matrices WQW^T and VRV^T. Note that $\vec{\varepsilon}_k$ and $\vec{\eta}_k$ have very different meanings here than from previous chapters.

Notice that the equations (B.25) and (B.26) are linear, and that they closely resemble the difference and measurement equations (B.1) and (B.2) from the discrete Kalman filter. This motivates us to use the measured residual $\tilde{\xi}_{z_k}$ in equation (B.24) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{\xi}_{x_k}$ given by equation (B.25). This estimate, call it $\hat{\xi}_k$, could then be used along with equation (B.23) to obtain the *a posteriori* state estimates for the original non-linear process as

$$\hat{x}_k = \tilde{x}_k + \hat{\xi}_k. \tag{B.27}$$

The random variables of equations (B.25) and (B.26) have *approximately* the following probability distributions (see the previous footnote):

$$p(\tilde{\xi}_{x_k}) \sim N(0, E\{\tilde{\xi}_{x_k}\tilde{\xi}_{x_k}^T\})$$
$$p(\vec{\varepsilon}_k) \sim N(0, WQ_kW^T)$$
$$p(\vec{\eta}_k) \sim N(0, VR_kV^T).$$

Given these approximations and letting the predicted value of $\hat{\xi}_k$ be zero, the Kalman filter equation used to estimate $\hat{\xi}_k$ is

$$\hat{\xi}_k = K_k\tilde{\xi}_{z_k}. \tag{B.28}$$

By substituting equation (B.28) into equation (B.27) and making use of equation (B.24) we see that we do not actually need the second (hypothetical) Kalman filter:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k\tilde{\xi}_{z_k} \\ &= \tilde{x}_k + K_k(\vec{z}_k - \tilde{z}_k) \end{aligned} \tag{B.29}$$

Equation (B.29) can now be used for the measurement update in the extended Kalman filter, with $\tilde{x}_k$ and $\tilde{z}_k$ coming from equations (B.15) and (B.16), and the Kalman gain K_k coming from equation (B.12) with the appropriate substitution for the measurement error covariance. The complete set of EKF equations is shown in table B.3 and table B.4.

Table B.3: Extended Kalman filter *time* update equations.

$$\hat{x}_{k+1}^- = f(\vec{x}_k, \vec{u}_k, 0) \tag{B.30}$$

$$P_{k+1}^- = \Xi_kP_k\Xi_k^T + WQ_kW^T \tag{B.31}$$

Table B.4: Extended Kalman filter *measurement* update equations.

$$K_k = P_k^-H_k^T(H_kP_k^-H_k^T + VR_kV^T)^{-1} \tag{B.32}$$

$$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - h(\hat{x}_k^-, 0)) \tag{B.33}$$

$$P_k = (I - K_kH_k)P_k^- \tag{B.34}$$

As with the basic discrete Kalman filter, the time update equations in table B.3 project the state and covariance estimates from time step k to step k+1. Again $f(\bullet)$ in equation (B.30) comes from equation (B.15), Ξ_k and W are the Jacobians of the respective equations (B.19) and (B.20) at step k, and Q_k is the process noise covariance matrix of equation (B.3) at step k.

As with the basic discrete Kalman filter, the measurement update equations in table B.4 correct the state and covariance estimates with the measurement $\vec{z}_k$. Again $h(\bullet)$ in equation (B.33) comes from equation (B.16), H_k and V are the respective Jacobians of equations (B.21) and (B.22) at step k, and R_k is the measurement noise covariance matrix of equation (B.4) at step k.

The basic operation of the EKF is the same as the linear discrete Kalman filter. Figure B.3 offers a complete picture of the operation of the EKF, combining the high-level diagram of figure B.1 with the equations from table B.3 and table B.4.

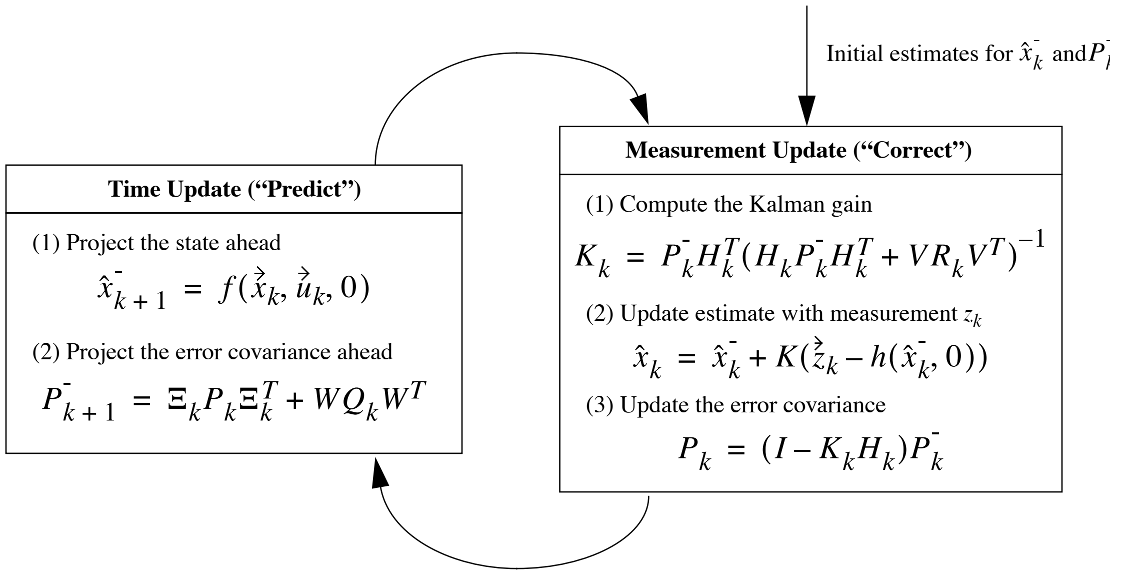

Initial estimates for $\hat{x}_k^-$ and P_k^-
Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_{k+1}^- = f(\vec{x}_k, \vec{u}_k, 0)$
(2) Project the error covariance ahead
$P_{k+1}^- = \Xi_k P_k \Xi_k^T + W Q_k W^T$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V R_k V^T)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - h(\hat{x}_k^-, 0))$
(3) Update the error covariance
$P_k = (I - K_k H_k) P_k^-$

Figure B.3: The complete *extended* Kalman filter operation. This figure combines the high-level diagram of figure B.1 with the equations from table B.3 and table B.4.

An important feature of the EKF is that the Jacobian H_k in the equation for the Kalman gain K_k serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement $\tilde{z}_k$ and the state via $h(\bullet)$, the Jacobian H_k affects the Kalman gain so that it only magnifies the portion of the residual $\tilde{z}_k - h(\hat{x}_k^-, 0)$ that does affect the state. Of course if for *all* measurements there is *not* a one-to-one mapping between the measurement $\tilde{z}_k$ and the state via $h(\bullet)$, then as you might expect the filter will quickly diverge. The control theory term to describe this situation is *unobservable*.

B.3 Example: Estimating a Random Constant

In the previous two sections we presented the basic form for the discrete Kalman filter, and the extended Kalman filter. To help in developing a better feel for the operation and capability of the filter, we present a very simple example here.

B.3.1 The Process Model

In this simple example let us attempt to estimate a scalar random constant, a voltage for example. Let's assume that we have the ability to take measurements of the constant, but that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise. In this example, our process is governed by the linear difference equation

$$\begin{aligned} x_{k+1} &= A_k x_k + B u_k + \omega_k \\ &= x_k + \omega_k \end{aligned},$$

with a measurement $z \in \Re^1$ that is

$$\begin{aligned} z_k &= H_k x_k + \nu_k \\ &= x_k + \nu_k \end{aligned}.$$

The state does not change from step to step so $A = 1$. There is no control input so $u = 0$. Our noisy measurement is of the state directly so $H = 1$. (Notice that we dropped the subscript k in several places because the respective parameters remain constant in our simple model.)

B.3.2 The Filter Equations and Parameters

Our time update equations are simply

$$\hat{x}^-_{k+1} = \hat{x}_k ,$$

$$P^-_{k+1} = P_k + Q ,$$

and our measurement update equations are

$$\begin{aligned} K_k &= P^-_k (P^-_k + R)^{-1} \\ &= \frac{P^-_k}{P^-_k + R}, \end{aligned} \tag{B.35}$$

$$\hat{x}_k = \hat{x}^-_k + K(z_k - \hat{x}^-_k) ,$$

$$P_k = (1 - K_k) P^-_k .$$

Presuming a very small process variance, we let $Q = 1e-5$. (We could certainly let $Q = 0$ but assuming a small but non-zero value gives us more flexibility in "tuning" the filter as we will demonstrate below.) Let's assume that from experience we know that the true value of the random constant has a standard normal probability distribution, so we will "seed" our filter with the guess that the constant is 0. In other words, before starting we let $\hat{x}^-_k = 0$.

Similarly we need to choose an initial value for P_k, i.e. P^-_k. If we were absolutely certain that our initial state estimate $\hat{x}^-_k = 0$ was correct, we would let $P^-_k = 0$. However given the uncertainty in our initial estimate $\hat{x}^-_k$, choosing $P^-_k = 0$ would cause the filter to initially and always believe $\hat{x}_k = 0$. As it turns out, the alternative choice is not critical. We could choose almost any $P^-_k \neq 0$ and the filter would eventually converge. We'll start our filter with $P^-_k = 1$.

B.3.3 The Simulations

To begin with, we randomly chose a scalar constant $z = -0.37727$ (there is no "hat" on the z because it represents the "truth"). We then simulated 50 distinct measurements z_k that had error normally distributed around zero with a standard deviation of 0.1 (remember we presumed that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise). We could have generated the individual measurements within the filter loop, but pre-generating the set of 50 measurements allowed me to run several simulations with the same exact measurements (i.e. same measurement noise) so that comparisons between simulations with different parameters would be more meaningful.

In the first simulation we fixed the measurement variance at $R = (0.1)^2 = 0.01$. Because this is the "true" measurement error variance, we would expect the "best" performance in terms of balancing responsiveness and estimate variance. This will become more evident in the second and third simulation. Figure B.4 depicts the results of this first simulation. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

When considering the choice for P_k^- above, we mentioned that the choice was not critical as long as $P_k^- \neq 0$ because the filter would eventually converge. In figure B.5 we have plotted the value of P_k^- versus the iteration. By the 50th iteration, it has settled from the initial (rough) choice of 1 to approximately 0.0002 (Volts^2).

In section B.1.5 we briefly discussed changing or "tuning" the parameters Q and R to obtain different filter performance. In figure B.6 and figure B.7 we can see what happens when R is increased or decreased by a factor of 100 respectively. In figure B.6 the filter was told that the measurement variance was 100 times greater (i.e. $R = 1$) so it was "slower" to believe the measurements. In figure B.7 the filter was told that the measurement variance was 100 times smaller (i.e. $R = 0.0001$) so it was very "quick" to believe the noisy measurements.

While the estimation of a constant is relatively straight-forward, it clearly demonstrates the workings of the Kalman filter. In figure B.6 in particular the Kalman "filtering" is evident as the estimate appears considerably smoother than the noisy measurements.

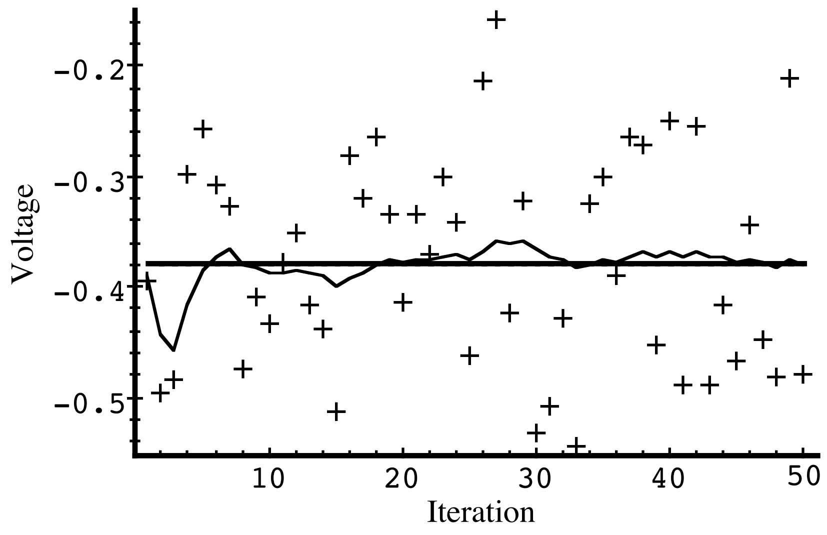


Figure B.4: The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

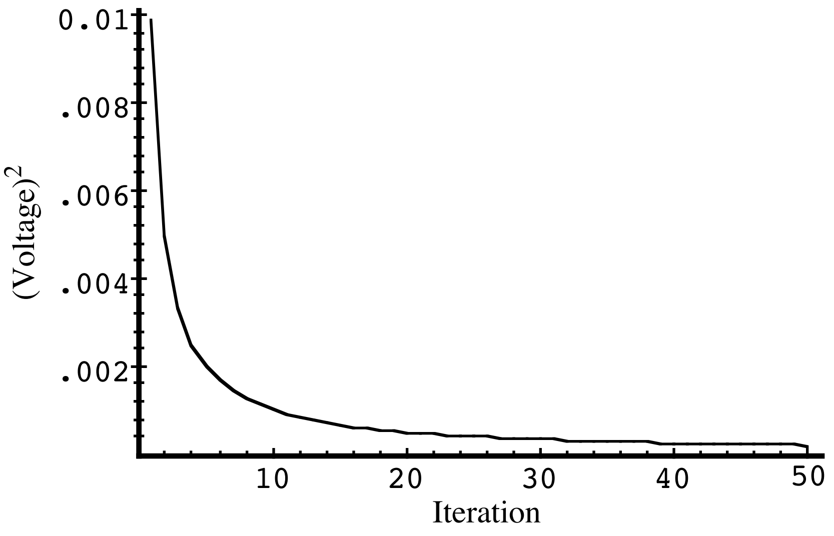


Figure B.5: Error covariance, first simulation. After 50 iterations, our initial (rough) error covariance P_k^- choice of 1 has settled to about 0.0002 [Volts^2].

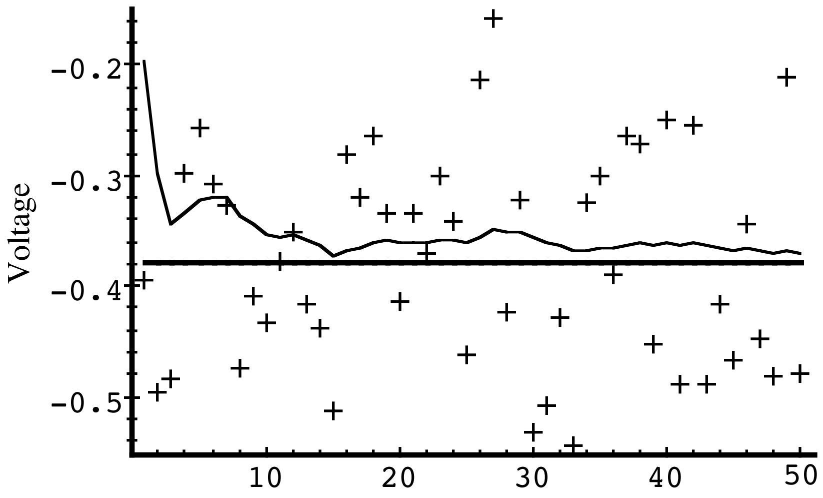


Figure B.6: Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

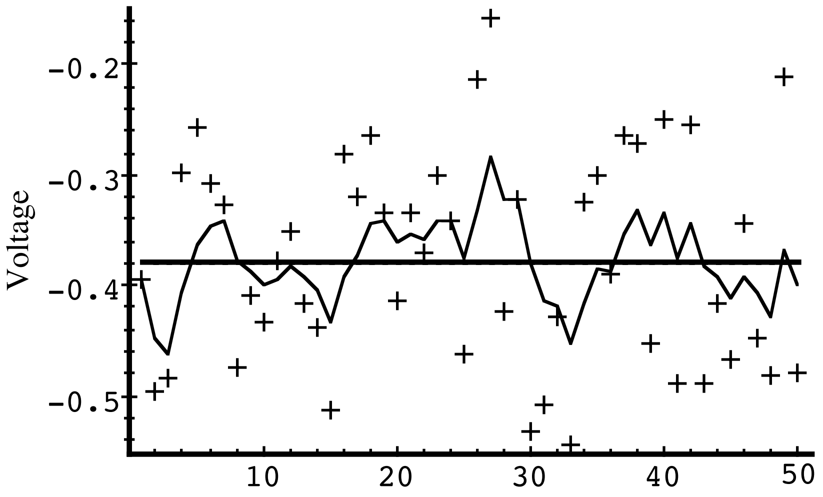
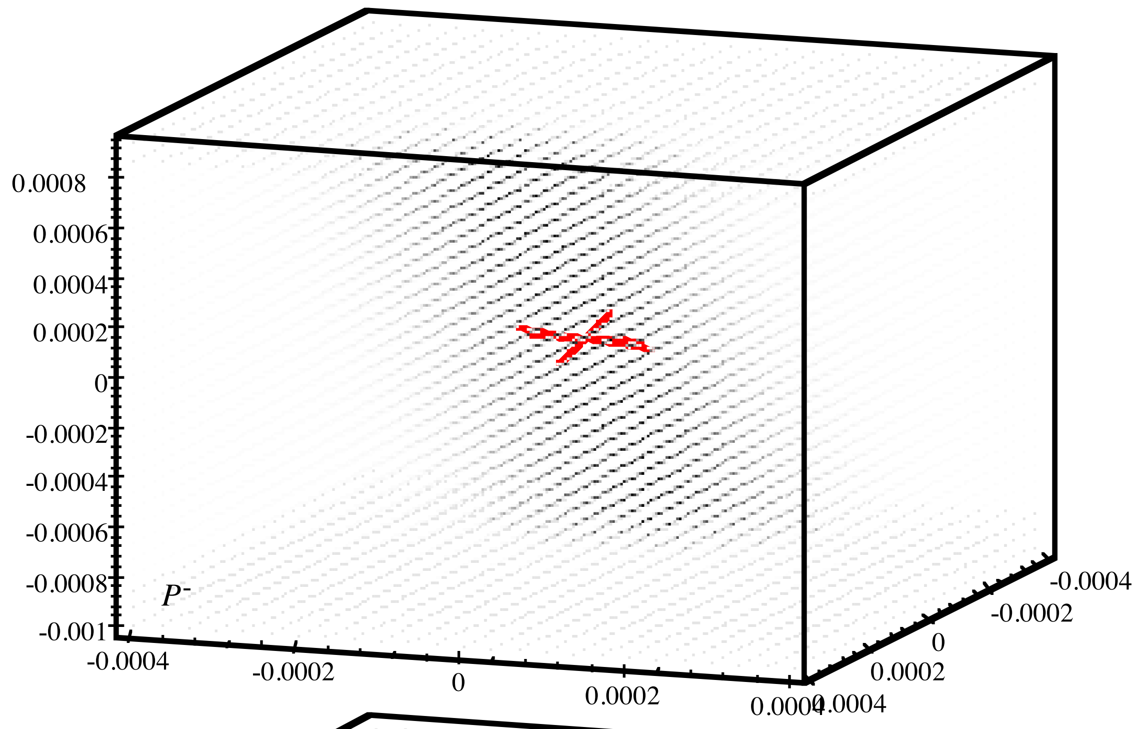


Figure B.7: Third simulation: $R = 0.0001$. The filter responds to measurements quickly, increasing the estimate variance.

APPENDIX C. ACTUAL PROBABILITY DENSITIES

This appendix presents plots of both actual (from simulation) error probability densities in various forms, and an actual Jacobian (measurement) matrix. These plots should be compared with the sketches in section 5.1 of chapter 5 (page 107). As such I have included (when appropriate) with each figure caption below references to the appropriate counterpart in chapter 5.

Aside from the figure captions, this appendix is presented entirely without commentary, other than to remind the reader that all of the densities are normal (normally distributed).



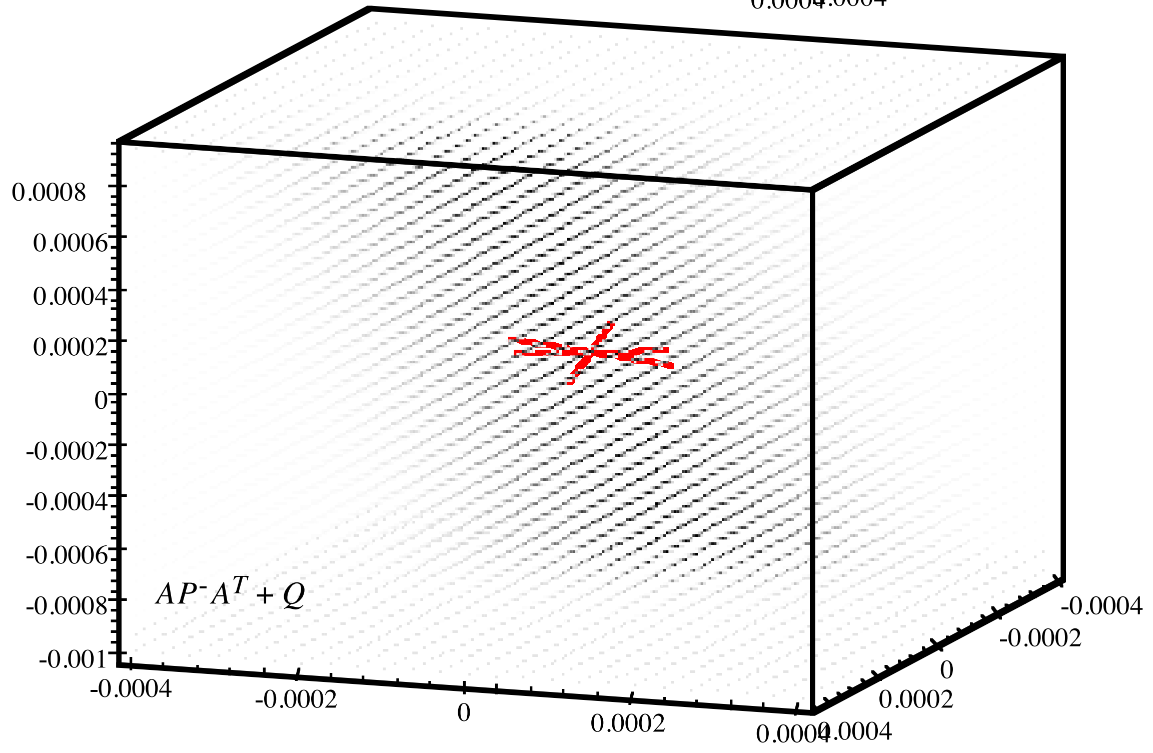


Figure C.1: Actual state error density before and after time update step. To visualize with Euclidean dimensions, the densities have been limited to the three position elements of the state. All axes are in meters. Darker points reflect greater densities. Notice that the region of darker points in the lower plot $APA^T + Q$ is "fatter" (and slightly reoriented) as the likelihood of error has *increased*. The given Eigenvectors further indicate the direction of uncertainty. (c.f. figure 4.3, page 85, and figure C.2 below.)

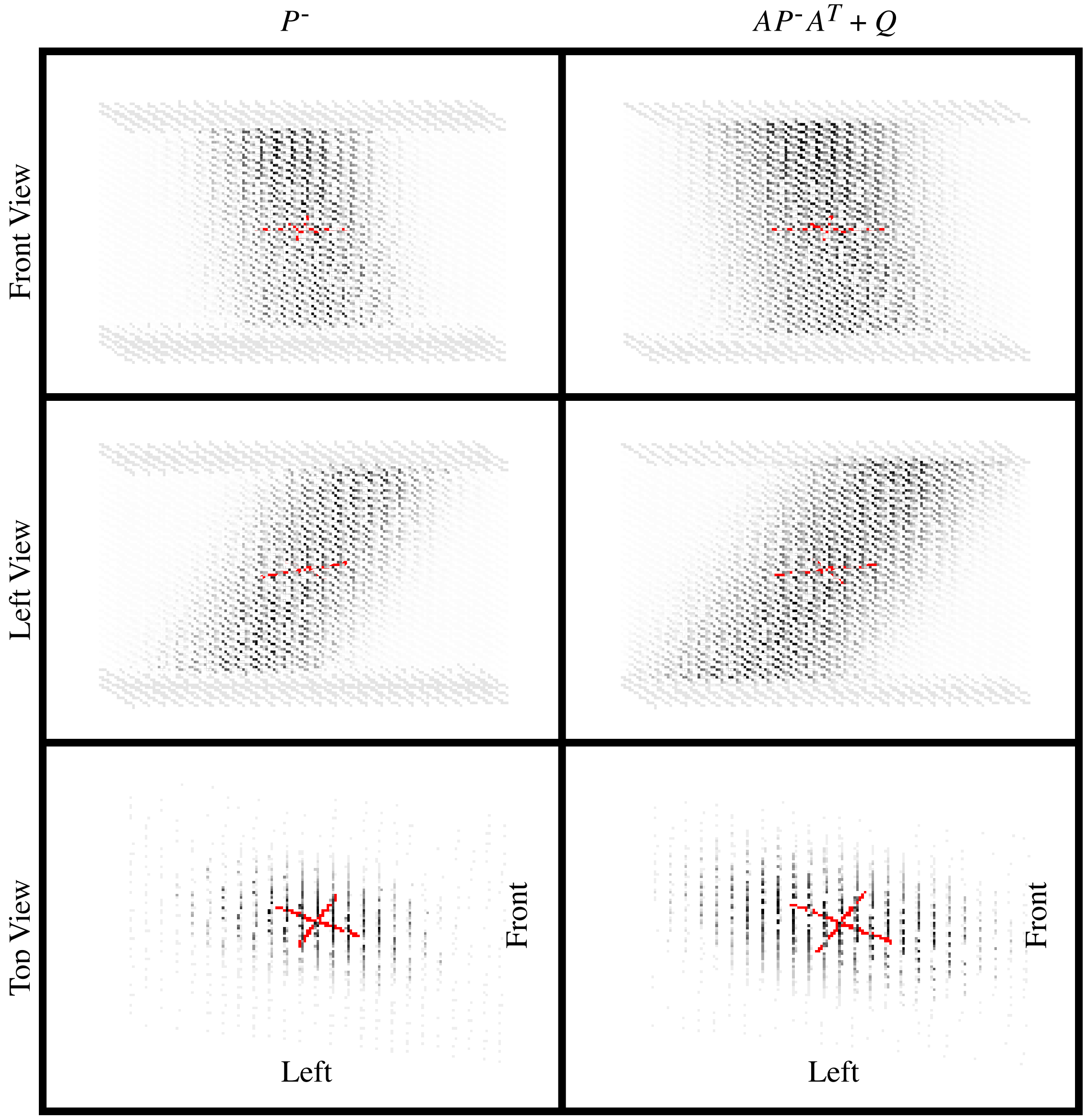


Figure C.2: Multiple viewpoints of densities from figure C.1. Any slight horizontal grey regions above and below some of the views are artifacts of the visualization process.

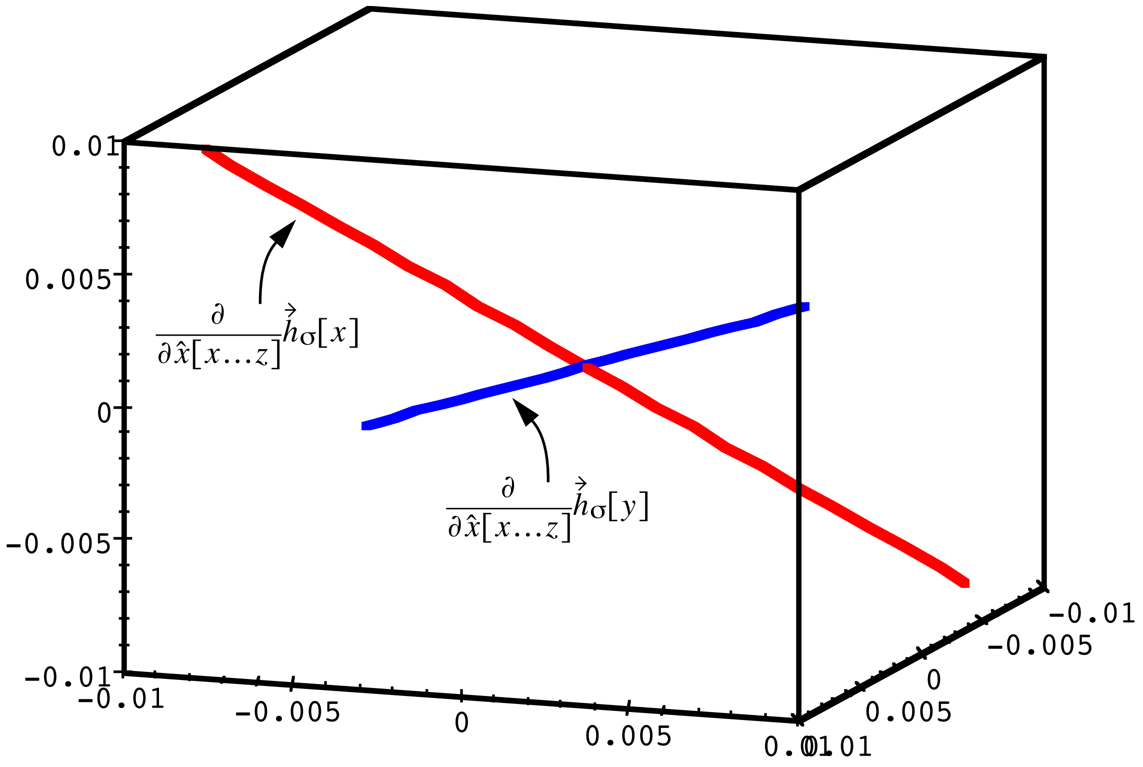


Figure C.3: Actual measurement Jacobian H. In order to simplify visualization, the Jacobian data has been limited to the three position elements of the state. All axes are in meters/meters. Note that for this measurement, the rate of change of the *x* camera coordinate with respect to the state position parameters is greater than that for the *y* camera

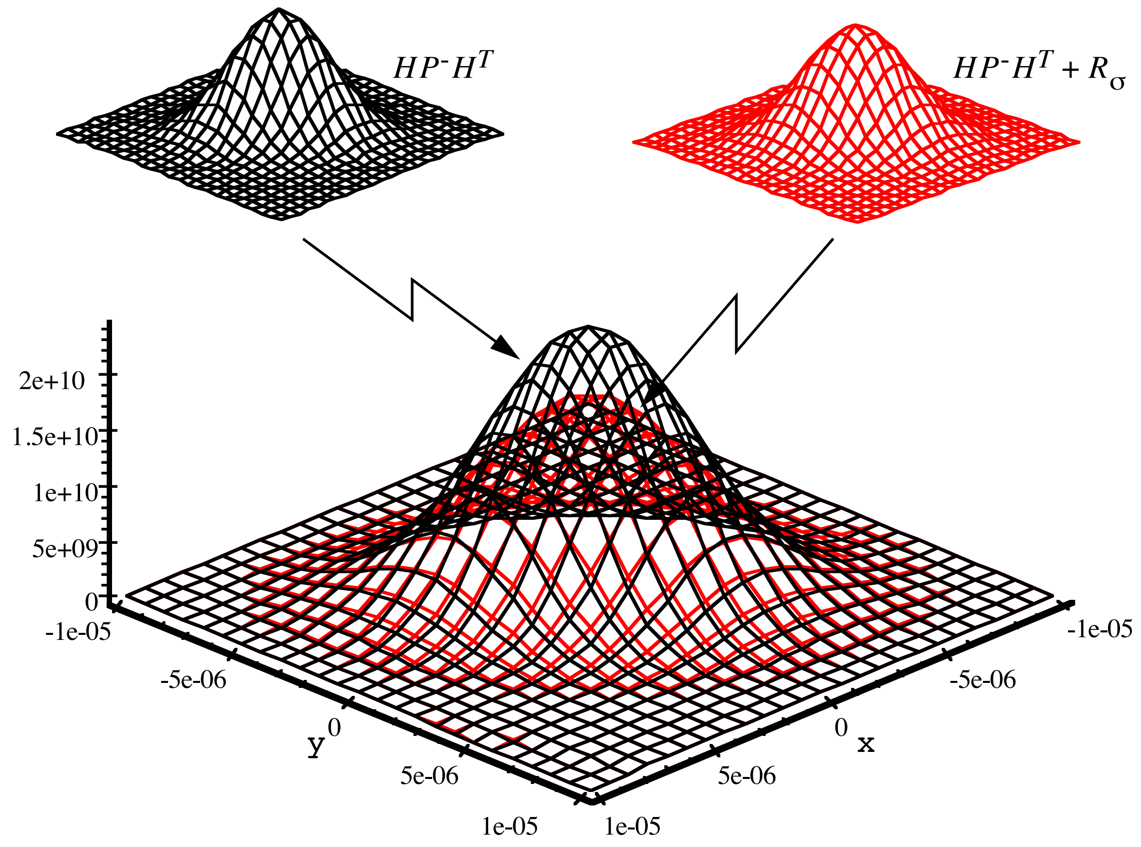


Figure C.4: Actual measurement-space error densities. The upper-left density reflects an actual 2D projection of the state error covariance P^{-} as depicted in figure 4.6 on page 89. The upper-right density is the same augmented by the measurement error covariance R_{σ} as depicted in figure 4.7 on page 90. The two are superimposed in the lower image for comparison. Note the augmented density is “shorter” but slightly “fatter” as would be expected for an increasing variance.

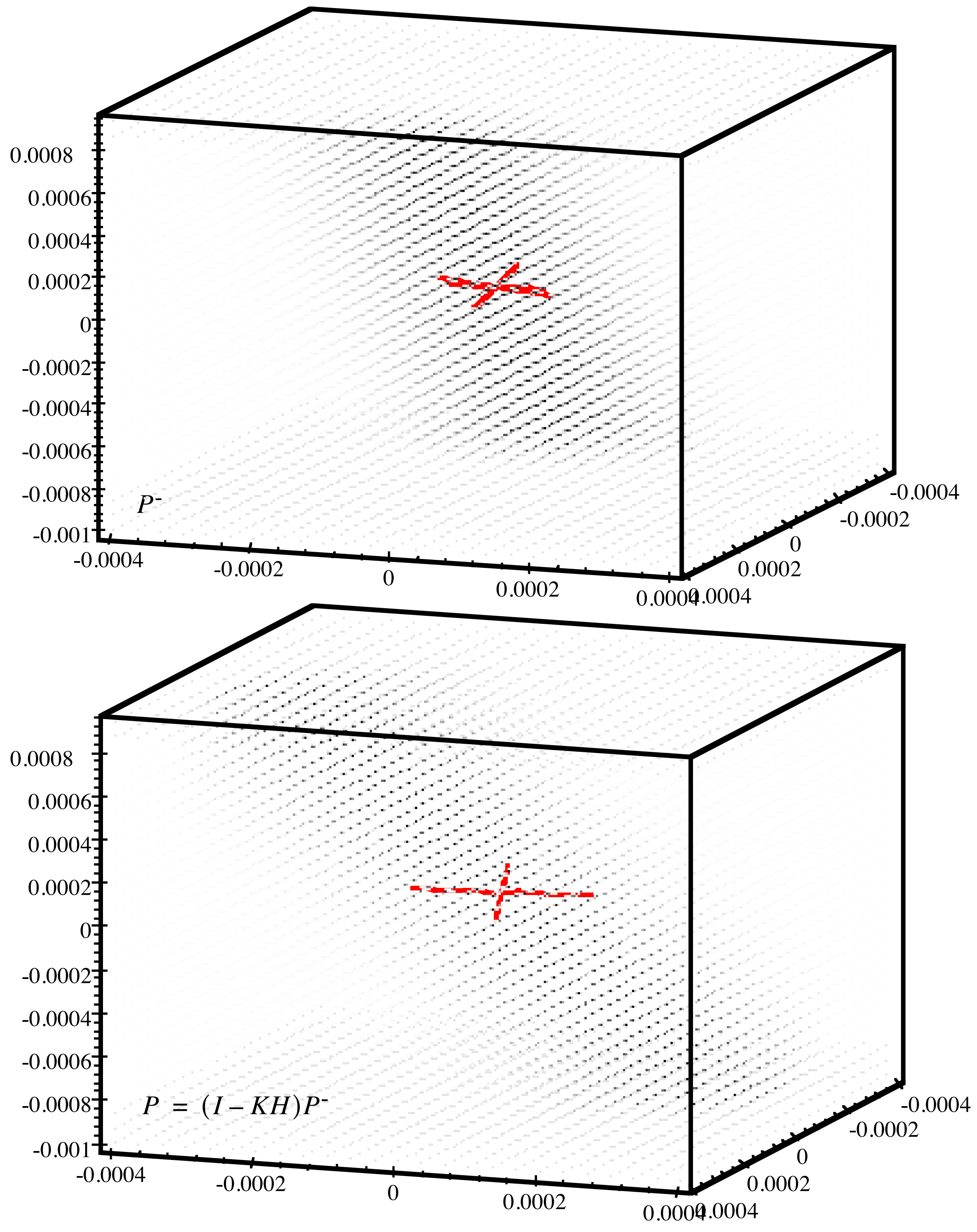


Figure C.5: Actual state error density before and after complete filter update. The density in the upper plot is the same as that in figure C.1. Notice that the density in the lower plot $P = (I - KH)P^-$ is “skinnier” (and slightly reoriented) as the likelihood of error has *decreased* per the actual measurement and the Jacobian shown in figure C.3. (c.f. figure 4.10, page 95, and figure C.6 below.)

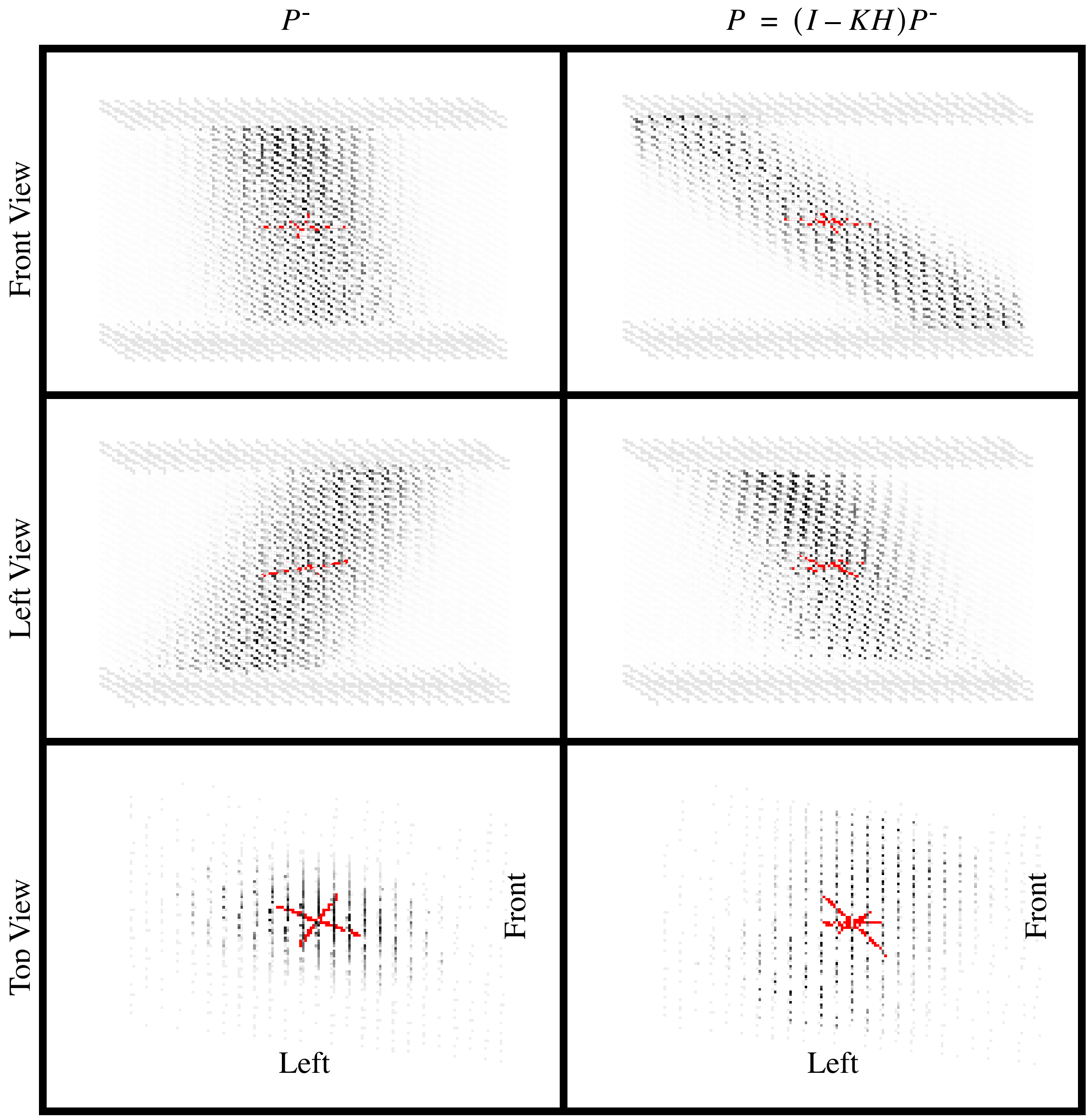


Figure C.6: Multiple viewpoints of densities from figure C.5. The left set of panes is the same as in figure C.2 (left panes).

APPENDIX D. THE UNC HIBALL TRACKER

The tracking system that I employ in my experiments is an “inside-looking-out” optoelectronic system designed for interactive computer graphics or *virtual environments*. The system, which we call the “HiBall tracker”, is an improved version of the wide-area system described in [Ward92]. In these systems, user-mounted optical sensors observe infrared light-emitting diodes (LEDs) or *beacons* mounted in the ceiling above the user. The locations of the beacons are known (to some degree) so observations of them can be used to estimate the user’s position and orientation. The overall system is depicted in figure D.1.

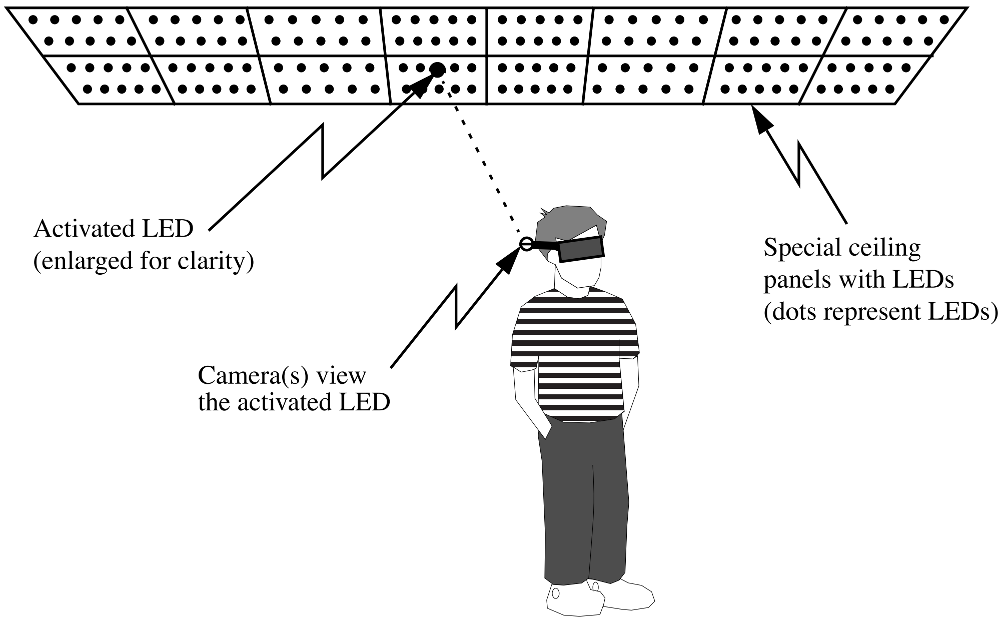


Figure D.1: An outward-looking optoelectronic tracking system. User-mounted cameras look outward (generally upward) at active beacons in the environment.

D.1 The Ceiling

One goal of the HiBall system is to provide a user with *wide-area* tracking. To this end, the LEDs are installed in special ceiling panels that replace standard-size acoustic ceiling tiles. (See figure D.1.) These tiles can be installed over a large area, thus providing a large working area. In fact, the current UNC installation consists of enough ceiling tiles to cover an area of approximately 5x5 meters, with a corresponding total of over 3,000 LEDs. The LED positions in world coordinates are initially estimated based on the ceiling panel design, which has them installed in a regular two-dimensional grid.

The LEDs are connected to special circuit boards that are mounted on (above) the ceiling tiles, and these circuit boards are then connected to a computer. Thus LEDs can be individually activated under computer control as needed for the SCAAT observations. (The ceiling circuitry allows beacon activation at over 5000 LEDs per second.)

D.2 The Hiball

In the original system described by [Ward92] the user wore a relatively large head-mounted mechanical fixture that supported several individually self-contained optical sensors or "cameras" and a backpack that contained the necessary signal processing and A/D conversion circuitry as shown in figure D.2.

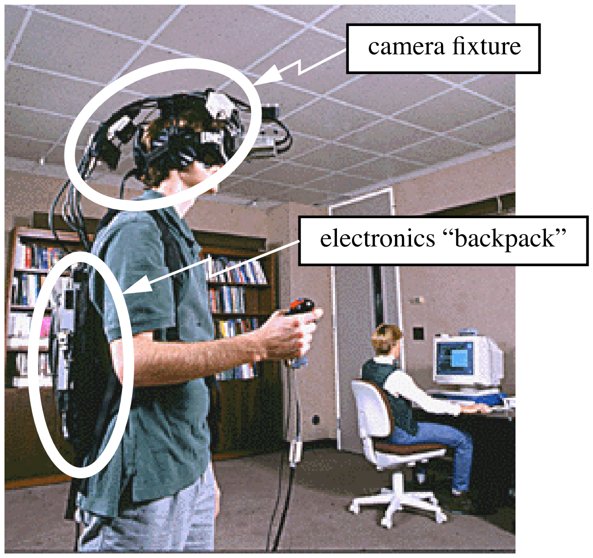


Figure D.2: The original UNC optoelectronic ceiling tracker. Graduate student Stefan Gottschalk is shown wearing the cumbersome camera fixture (on his head) and the electronics backpack.

In the new system the camera fixture and backpack in figure D.2 are together replaced by a relatively small sensor cluster called the *HiBall*. Figure D.3 is a picture of an unpopulated HiBall with a golf ball to convey a notion of size.

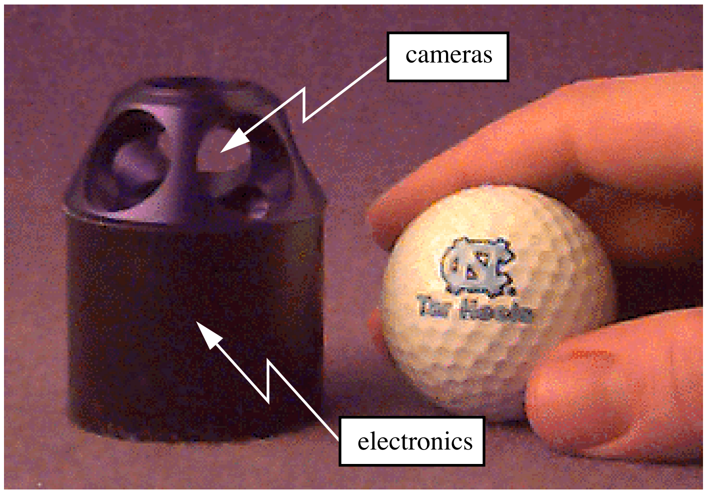


Figure D.3: The new HiBall camera cluster. The HiBall is shown next to a golf ball to convey a notion of size.

A populated HiBall contains six lenses, six photodiodes with infrared filters, and all of the necessary circuitry for signal processing, A/D conversion, control, and high-speed serial communication. It is designed so that each photodiode can view infrared beacons (LEDs in this case) through each of several adjacent lenses, thus implementing up to 26 distinct infrared cameras.

The photodiodes provide four signals that together indicate the position of the centroid of light (infrared in this case) as it appears on the two-dimensional photodiode surface. At any point in time, the internal A/D conversion circuitry can sample these signals for any one photodiode. These samples would then be sent to an external computer via a high-speed serial link where they would be converted to the measurement $\vec{z} = \begin{bmatrix} u & v \end{bmatrix}^T$ that reflect the position of the image of the infrared beacon on the two-dimensional photodiode.

If the photodiode measurements are viewed as images of the ceiling beacons (known scene points), the HiBall tracker can be viewed as an implementation of the abstract image-based example initially introduced in section 1.2 on page 39 (see figures 1.4-1.6) and later used in chapter 5. Hence my frequent references to a HiBall lens and sensor pair as a camera.

APPENDIX E. THE SIMULATION ENVIRONMENT

E.1 User Motion Data Sets

To observe the performance of the various methods, we chose to employ real user motion data in our simulations. Specifically we used several motion data sets that were collected by Ron Azuma during various "demo days" at UNC. On a regular basis we at UNC offer public demonstrations of the systems we have devised in our ongoing virtual environments research. Among other things, the original UNC wide area optoelectronic tracking system was demonstrated with a head-mounted display driven by UNC's Pixel-Planes 5 graphics engine. On several occasions, Ron Azuma configured the ceiling tracker to transparently dump the position and orientation estimates to a file in real-time so that they could be examined off-line at a later time. The lengths of these data sets range from 15 seconds to 4 minutes. These are the data sets that I used for the simulations described in chapter 6, except for section 6.2.10 as described in the next paragraph. From all of the motion data sets I chose one particularly representative or "typical" 3 minute data set to use for comparisons where one such data set would suffice to demonstrate a phenomenon of interest. The characteristics of this data set are given in table E.1 (page 195).

In addition, I created a special "still" data set that simulates holding the HiBall perfectly still for several minutes, as if resting on a sturdy table, with the state

$$(x, y, z, \phi, \theta, \psi) = (2.74, 4.27, 1.60, 0, 0, 0)$$

in meters and degrees. I used this data set to investigate SCAAT performance under low dynamics in section 6.2.10 on page 152.

E.2 Beacon Observations

For all of the data sets discussed in section E.1, we low-pass filtered (see section E.5 below) and then resampled the data to obtain position and orientation sequences that we defined as the "truth". These sequences were used to generate a sequence of beacon

Table E.1: Characteristics of the "typical" data set. Note that linear and angular velocities are given as magnitudes (absolute values).

units	dimension	minimum	maximum	range	average
[meters]	x	1.599658	3.892787	2.293129	2.524411
	y	3.031233	5.502644	2.471410	4.223665
	z	0.774936	1.584778	0.809842	1.454678
$\left[\frac{\text{meters}}{\text{sec}}\right]$	$\lvert\dot{x}\rvert$	0.000000	0.731384	0.731383	0.080369
	$\lvert\dot{y}\rvert$	0.000001	0.662029	0.662029	0.079128
	$\lvert\dot{z}\rvert$	0.000000	0.466307	0.466307	0.030943
[degrees]	ϕ	-58.057716	9.848511	67.906227	-21.270809
	θ	-17.455811	5.933789	23.389600	-6.233829
	ψ	-179.985365	179.979526	359.964891	-14.975613
$\left[\frac{\text{degrees}}{\text{sec}}\right]$	$\lvert\dot{\phi}\rvert$	0.000043	95.192619	95.192576	14.003885
	$\lvert\dot{\theta}\rvert$	0.000176	88.479740	88.479565	6.324057
	$\lvert\dot{\psi}\rvert$	0.000031	88.886032	88.886001	6.409921

observations as they would have occurred under the "true" circumstances, given a round-robin selection strategy (between HiBall camera views, and then between beacons within the individual camera views). Specifically, the truth sequences were used to project beacons onto camera "image planes", and the corresponding measurements were then passed along to the appropriate implementation, e.g. SCAAT or Collinearity. (See section E.5 below for a related assumption about "Primary HiBall Views".)

The observations (image plane measurements) were corrupted with normally-distributed random error as needed to simulate measurement error. All random number generators could be seeded to maintain consistency in comparisons between methods.

E.3 EKF and Collinearity Implementations

In chapter 6 I present simulations of a SCAAT EKF implementation, an multiple-constraint-at-a-time EKF implementation (see section 3.7 on page 70), and a Collinearity implementation (see section 3.4 on page 68). In this section I present a brief explanation for the implementation of each.

E.3.1 SCAAT and MCAAT

To facilitate both SCAAT and MCAAT simulations I actually implemented a generic EKF HiBall simulator where the number of observations N was made a command line parameter. A call to the simulator with $N = 1$ would result in a SCAAT simulation, a call with $N > 1$ would result in an MCAAT simulation. The simulator simply allocated a measurement vector for the N observations and then invoked the measurement function N times, once for each of the N observations as they arrived. When the N observations were collected and the measurement vector was "full", the filter would compute the residual and update the state. Hence a new estimate was generated every N observations.

E.3.2 Collinearity

I implemented a Collinearity version of the HiBall (for the purpose of simulation) exactly as described in [Azuma91]. In particular, I implemented the *singular value decomposition* approach described as the current "method of choice". As it turns out, this does not appear to be the exact method used in the most recent implementation of UNC wide-area optoelectronic tracking system, however it is very similar in terms of computational complexity.

The Collinearity position-recovery algorithm was encapsulated within the same framework as the generic EKF implementation described above in section E.3.1. It was also designed to use the number of observations indicated by the command line parameter N, although in this case it was ensured that $N \geq 3$ as required by the algorithm.

E.3.3 Moving-Window Observations

Both the EKF implementation of section E.3.1 and the Collinearity implementation of section E.3.2 were designed to allow *moving-window* observations as depicted in figure E.1 The normal (fixed-window) version simply discards the most recent N observations after they are used, while the moving-window version preserves the most recent observations in a circular buffer of size N, using the N most recent observations at each step. Though not implemented directly as such, the moving-window version can be

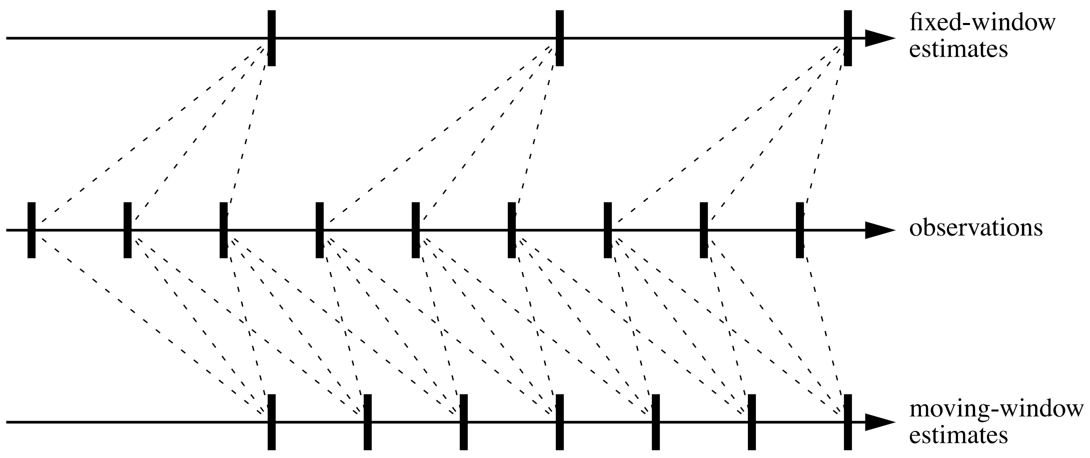


Figure E.1: Moving vs. fixed-window observations. This example shows a system that uses $N = 3$ observations per estimate. The estimate rate for the moving-window version is clearly higher.

thought of as a *finite impulse response* (FIR) filter. In this case, the influence of any one observation extends over N steps in time. Almost all of the experiments in chapter 6 process observations in a normal normal (fixed-window) fashion, with the exception being the experiments described in section 6.2.6 on page 147.

E.4 EKF Parameters

In chapter 6 I present simulations of both SCAAT and multiple-constraint-at-a-time extended Kalman filters. Because these implementations differ only in N, the number of constraints incorporated during the filter update, they can (indeed should) employ the same dynamic models. As such they share a need for the determination of the process noise and initial error covariance parameters as described in section 6.1.3 and section 6.1.4 (respectively) for our HiBall SCAAT implementation.

In particular, for any extended Kalman filter implementation that employed the process models presented in section 6.1 and employed $N \geq 1$ constraints, I needed to be able to determine the complete parameter set

$$\Pi[N] \equiv \{\vec{\eta}[\lambda], \vec{\eta}[z], \vec{\eta}[\phi], \vec{\eta}[\theta], \vec{\eta}[\psi], \vec{\eta}[b], \xi_b\}\ .$$

To find a reasonable parameter set for any particular N could prove difficult. To find such a set for *any* $N \geq 1$ is even more difficult. In order to make the task more manageable I chose to follow the approach taken by Azuma and Bishop in [Azuma94] and to employ Powell's method as given in [Press90] to search the parameter space represented by $\Pi[N]$ for any $N \geq 1$.

To use Powell's method, I needed to define a *cost function* that returned a scalar indication of the "goodness" of a particular set of parameters. In simulations I had access to both the estimated filter state and the "true" state (see "Motion Bandwidth" below) for several motion data sets as described in section E.1. Again employing a scheme similar to that of Azuma and Bishop, I designed a special simulation framework for the purpose of parameter optimization. At every filter update step I compute the locations (in world coordinates) of three points arranged in a triangle that is oriented upright and faces the HiBall approximately one meter in front of the HiBall. I compute these three points for both the estimated filter state and the true state, and then I compute the average distance between the respective estimated and true *point groups*. This average distance provides a per-estimate scalar cost, which I then average for an entire simulation run (for a particular data set) to obtain the necessary scalar cost for a particular parameter set $\Pi[N]$. This approach nicely combines position and orientation error into a single cost. The parameters found using this method for various test cases are given in section 6.2.1 of chapter 6.

E.5 Assumptions

Computation Time

We assumed that the computation time $\tau_c(N)$ was negligible, i.e. zero. This is a reasonable assumption because for any method the measurement and computation stages could be pipelined, adding only a fixed amount of latency to the corresponding estimates. In any case, if the actual computation times *were* accounted for, the SCAAT performance would be affected the least because its computation time is the smallest. (See section 5.2 on page 112.)

Motion Bandwidth

To create "truth" position and orientation sequences, we low-pass filtered and resampled actual data sets as described above in section E.2. For all of the experiments described in chapter 6 we used a low-pass cutoff frequency of 5 Hz. While in general one would desire a motion bandwidth of 20 Hz (see section 2.2 of chapter 2), we examined the power spectra for the various data sets described above in section E.2 *prior* to filtering and determined that the majority of the power was below 5 Hz. This makes sense under the circumstances of the original data collection—inexperienced users and a relatively bulky headset (see figure D.2 of appendix D, page 192). Other filter parameters could easily be obtained using the method of section E.4 above as necessary for other dynamics.

Primary HiBall Views

Because the HiBall camera cluster contains sensors that can image beacons through multiple lenses it is possible to see both *primary* and *secondary* views of a beacon as shown below in figure E.2. (See also figure D.3 of appendix D, page 193.)

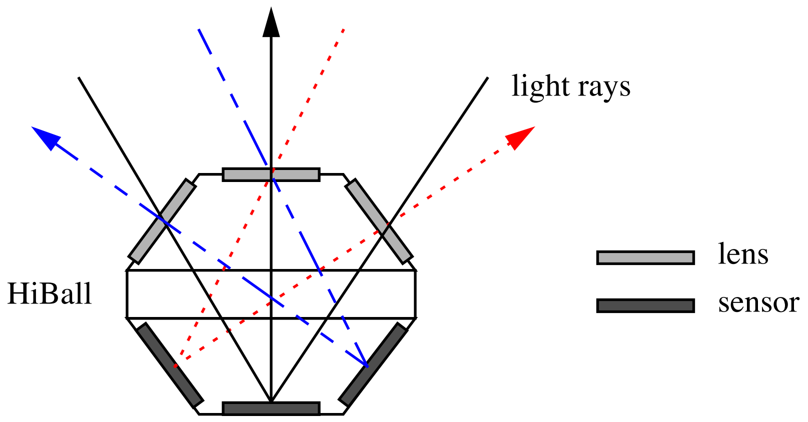


Figure E.2: Primary and secondary HiBall views. Each sensor can "see" light from multiple lenses. A sensor's view through the lens directly opposite is called a *primary* view (rays with arrows). There are only six such views corresponding to the six lens and sensor opposite pairs. The other rays shown correspond to *secondary* views. *Tertiary* views are also possible but none are shown. (This cut-away figure shows only three each of the six lenses and sensors.)

For the purpose of these simulations we assumed the various implementations, e.g. SCAAT or Collinearity, received only observations from primary views, thus they were limited to only six cameras. (That is to say that we only generated observations through primary views.) This approach is reasonable because primary vs. secondary is an issue that is not directly related to the subject of this dissertation—while it is an issue that would be of concern for any implementation. In addition, we feel that our actual SCAAT implementation will be able to (under most circumstances) distinguish between primary and secondary sightings of beacons. As such we believe that the incorporation of secondary views will improve performance by facilitating a larger number of beacon observations.

REFERENCES

American81	The American Heritage Dictionary, Houghton Mifflin Company, 1981, p. 980.
Atkeson85	Atkeson, C.G., and J.M. Hollerbach. 1985. “Kinematic features of unrestrained vertical arm movements,” Journal of Neuroscience, 5:2318-2330.
Azarbayejani95	Azarbayejani Ali, and Alex Pentland. 1995. “Recursive Estimation of Motion, Structure, and Focal Length,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, June 1995, 17(6).
Azuma91	Azuma, Ronald and Mark Ward. 1991. “Space-Resection by Collinearity: Mathematics Behind the Optical Ceiling Head-Tracker,” UNC Chapel Hill Department of Computer Science technical report TR 91-048 (November 1991).
Azuma94	Azuma, Ronald and Gary Bishop. 1994. “Improving Static and Dynamic Registration in an Optical See-Through HMD,” Proceedings of ACM SIGGRAPH '94 (Orlando, FL, July 1994), Computer Graphics, Annual Conference Series.
Azuma95	Azuma, Ronald. 1995. “Predictive Tracking for Augmented Reality,” Ph.D. dissertation, University of North Carolina at Chapel Hill, TR95-007.
Bar-Shalom93	Bar-Shalom, Yaakov, and Xiao-Rong Li. 1993. *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Inc., ISBN 0-89006-643-4.
Bell93	Bell, Bradley M., and Fredrick W. Cathey. 1993. “The Iterated Kalman Filter Update as a Gauss-Newton Method,” *IEEE Transactions on Automatic Control*, Vol. 38, No. 2, February, 1993.
Broida86	Broida, Ted J. and Rama Chellappa. 1986. “Estimation of object motion parameters from noisy images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, January 1986, 8(1), pp. 90-99.
Brown92	Brown, R. G. and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, 2nd Edition*, John Wiley & Sons, Inc.
Burton73	Burton, Robert Preece. June 1973. “Real-time measurements of multiple three-dimensional positions,” Ph.D. dissertation, Computer Science, University of Utah. UTEC-CSc-72-122.

Chi95	Chi, Vernon L. 1995. "Noise Model and Performance Analysis of Outward-looking Optical Trackers Using Lateral Effect Photo Diodes," University of North Carolina, Department of Computer Science, TR 95-012 (April 3, 1995).
Chou92	Chou, Jack C.K. 1992. "Quaternion Kinematic and Dynamic Differential Equations," IEEE Transactions on Robotics and Automation, Vol. 8, No. 1, pp. 53-64.
Crowley93	Crowley, J. L. and Y. Demazeau. 1993. "Principles and Techniques for Sensor Data Fusion," *Signal Processing (EURASIP)* Vol. 32. pp. 5-27.
Deyst68	Deyst J. J. and C. F. Price. 1968. "Conditions for Asymptotic Stability of the Discrete Minimum-Variance Linear Estimator," *IEEE Transactions on Automatic Control*, December, 1968.
Emura94	Emura, S. and S. Tachi. 1994. "Sensor Fusion based Measurement of Human Head Motion," *Proceedings 3rd IEEE International Workshop on Robot and Human Communication, RO-MAN'94 NAGOYA* (Nagoya University, Nagoya, Japan).
Fischer90	Fischer, P., R. Daniel and K. Siva. 1990. "Specification and Design of Input Devices for Teleoperation," *Proceedings of the IEEE Conference on Robotics and Automation* (Cincinnati, OH), pp. 540-545.
Foxlin93	Foxlin, Eric. 1993. "Inertial Head Tracking," Master's Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
Foxlin96	Foxlin, Eric. 1996. "Inertial Head-Tracker Sensor Fusion by a Complimentary Separate-Bias Kalman Filter," *VRAIS 96 Proceedings*, March 30-April 3, Santa Clara, California.
Friedman92	Friedman, M., T. Starner and A. Pentland. 1992. "Synchronization in Virtual Realities," *Presence: Teleoperators and Virtual Environments,* 1:139-144.
Ganapathy84	Ganapathy, S. November 1984. "Camera Location Determination Problem," AT&T Bell Laboratories Technical Memorandum, 11358-841102-20-TM.

Geier87	Geier, G. J., P. V. W. Loomis and A. Cabak. 1987. "Guidance Simulation and Test Support for Differential GPS (Global Positioning System) Flight Experiment," National Aeronautics and Space Administration (Washington, DC) NAS 1.26:177471.
Gelb74	Gelb, A. 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.
Gottschalk93	Gottschalk, Stefan and John F. Hughes. 1993. "Autocalibration for Virtual Environments Tracking Hardware," *Proceedings of ACM SIGGRAPH '93* (Anaheim, CA, 1993), Computer Graphics, Annual Conference Series.
Grandjean89	P. Grandjean, A Robert De Saint Vincent. 1989. "3-D Modeling of Indoor Scenes by Fusion of Noisy Range and Stereo Data," *IEEE International Conference on Robotics and Automation* (Scottsdale, AZ), 2:681-687.
Grewal96	Grewal M.S., and A.P. Andrews. 1996. *Kalman Filtering: Theory and Practice*, Prentice-Hall, Inc., 3rd Printing.
Ham83	Ham F. C. and R. G. Brown. 1983. "Observability, Eigenvalues, and Kalman Filtering," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-19, No. 2, pp. 269-273.
Hamilton1853	Hamilton, W. R. 1853. "Lectures on Quaternions," Dublin: Hodges and Smith.
Held87	Held, R. and N. Durlach. 1987. *Telepresence, Time Delay, and Adaptation*. NASA Conference Publication 10023.
Holloway95	Holloway, Richard L. 1995. "Registration Errors in Augmented Reality Systems," Ph.D. dissertation, The University of North Carolina at Chapel Hill, TR95-016.
Ikeda95	Ikeda, M., G. Evensen and L. Cong. 1995. "Comparison of sequential updating, Kalman filter and variational methods for assimilating Rossby waves in the simulated Geosat altimeter data into a quasi-geostrophic model," *Journal of Marine Systems (Special Issue: Data Assimilation in Marine Science)*, 6:15-30.
Jacobs93	Jacobs, O. L. R. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.

Julier95 Julier, Simon and Jeffrey Uhlman. “A General Method of Approximating Nonlinear Transformations of Probability Distributions,” Robotics Research Group, Department of Engineering Science, University of Oxford [cited 14 November 1995]. Available from http://phoebe.robots.ox.ac.uk/reports/New_Fltr.zip (88K).

Kalman60 Kalman, Rudolph Emil. 1960. “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

Kaplan96 Kaplan, Elliot D. (Editor) 1996. *Understanding GPS Principles and Applications*, Artech House.

Kelly94 Kelly, Douglas. 1994. *Introduction to Probability*, Macmillan Publishing Company, New York.

Kuipers80 Kuipers, J. B. 1980. “SPASYN—An Electromagnetic Relative Position and Orientation Tracking System,” *IEEE Transactions on Instrumentation and Measurement*, Vol. IM-29, No. 4, pp. 462-466.

Lewis86 Lewis, Richard. 1986. *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.

Liang91 Liang, J., C. Shaw and M. Green. 1991. “On Temporal-spatial Realism in the Virtual Reality Environment,” *Fourth Annual Symposium on User Interface Software and Technology*, pp. 19-25.

Lindgren78 Lindgren, Allen G., and Kai F. Gong. July 1978. "Position and Velocity Estimation Via Bearing Observations," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-14, No. 4.

Mahmoud94 Mahmoud, R., O. Loffeld and K. Hartmann. 1994. “Multisensor Data Fusion for Automated Guided Vehicles,” *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 2247, pp. 85-96.

Maybeck70 Maybeck, Peter S. 1979. *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.

Mazuryk95 Mazuryk, Thomas, and Michael Gervautz. 1995. “Two-Step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments,” *EUROGRAPHICS ‘95*, Vol. 14, No. 3, pp. 30-41.

Meditch69 Meditch, S. 1969. *Stochastic Optimal Linear Estimation and Control*, New York: McGraw-Hill.

Meyer92 Meyer, K., H. Applewhite and F. Biocca. 1992. A Survey of Position Trackers. *Presence,* a publication of the *Center for Research in Journalism and Mass Communication,* The University of North Carolina at Chapel Hill.

Mine93 Mine, Mark. 1993. “Characterization of End-to-End Delays in Head-Mounted Display Systems,” The University of North Carolina at Chapel Hill, TR93-001.

Nalwa93 Nalwa, Vishvjit S. 1993. *A Guided Tour of Computer Vision*, Addison-Wesley Publishing Company, ISBN 0-201-54853-4

Neilson72 Neilson, P.D. 1972. “Speed of Response or Bandwidth of Voluntary System Controlling Elbow Position in Intact Man,” *Medical and Biological Engineering*, 10:450-459.

NRC94 National Research Council. 1994. “Virtual Reality, Scientific and Technological Challenges,” National Academy Press (Washington, DC).

Petridis81 Petridis, Vassilios. April 1981. "A Method for Bearings-Only Position and Velocity Estimation," IEEE Transactions on Automatic Control, Vol. AC-26, No. 2.

Press90 Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. 1990. *Numerical Recipes in C, The Art of Scientific Computing*, Cambridge University Press.

Price68 Price, Charles F. 1968. “An Analysis of the Divergence Problem in the Kalman Filter,” IEEE Transactions on Automatic Control, Vol. AC-13, No. 6, pp. 699-701.

Raab79 Raab, F. H., E. B. Blood, T. O. Steiner, and H. R. Jones. 1979. “Magnetic Position and Orientation Tracking System,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-15, 709-718.

Rauch63 Rauch, H.E. 1963. “Solutions to the Linear Smoothing Problem,” *IEEE Transactions on Auto. Control, AC-8*

Rauch65 Rauch, H.E., F. Tung, and C.T. Striebel. 1965. “Maximum Liklihood Estimates of Linear Dynamic Systems,” *AIAA J., 3*

Rebo88 Rebo, Robert. 1988. “A Helmet-Mounted Virtual Environment Display System,” M.S. Thesis, Air Force Institute of Technology.

Selspot	Selspot Technical Specifications, Selcom Laser Measurements, obtained from Innovision Systems, Inc. (Warren, MI).
So92	So, Richard H. Y., and Michael J. Griffin. July-August 1992. "Compensating Lags in Head-Coupled Displays Using Head Position Prediction and Image Deflection," *AIAA Journal of Aircraft*, Vol. 29, No. 6, pp. 1064-1068.
Sorenson70	Sorenson, H. W. 1970. "Least-Squares estimation: from Gauss to Kalman," *IEEE Spectrum*, Vol. 7, pp. 63-68, July 1970.
State96	State, Andrei, Gentaro Hirota, David T. Chen, Bill Garrett, Mark Livingston. 1996. "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking," Proceedings of ACM SIGGRAPH '96 (New Orleans, LA, August 1996), Computer Graphics, Annual Conference Series.
VanPabst95	Van Pabst J. V. L. and Paul F. C. Krekel. "Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space," TNO Physics and Electronics Laboratory, The Hague, The Netherlands. [cited 19 November 1995]. Available from http://www.bart.nl/~lawick/index.html.
Wang90	Wang, J., R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. 1990. "Tracking a head-mounted display in a room-sized environment with head-mounted cameras," *Proceeding: SPIE'90 Technical Symposium on Optical Engineering & Photonics in Aerospace Sensing* (Orlando, FL).
Ward92	Ward, Mark, Ronald Azuma, Robert Bennett, Stefan Gottschalk, and Henry Fuchs. 1992. "A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems," *Proceedings of 1992 Symposium on Interactive 3D Graphics* (Cambridge, MA, 29 March - 1 April 1992), pp. 43-52.
Watanabe94	Watanabe, Kajiro, Kazuyuki Kobayashi, and Fumio Munekata. 1994. "Multiple Sensor Fusion for Navigation Systems," 1994 Vehicle Navigation and Information Systems Conference Proceedings, pp. 575-578.
Wefald84	Wefald, K.M., and McClary, C.R. "Autocalibration of a laser gyro strapdown inertial reference/navigation system," *IEEE PLANS '84*. Position Location and Navigation Symposium Record, pp. 66-74.

Welch95	Welch, Gregory and Gary Bishop. 1995. "An Introduction to the Kalman Filter," University of North Carolina, Department of Computer Science, TR 95-041.
Wishner69	Wishner, R. P., J. A. Tabaczynski, and M. Athans. 1969. "A Comparison of Three Non-Linear Filters," *Automatica*, Vol. 5, pp. 487-496.
Wloka95	Wloka, Mathhhias M. 1995. "Lag in Multiprocessor Virtual Reality," *PRESENCE: Teleoperators and Virtual Environments 4.1*, Winter 1995, pp. 50-63.
Woltring74	Woltring, H. J. 1974. "*New possibilities for human motion studies by real-time light spot position measurement*," Biotelemetry, Vol. 1.