INTELLIGENT SENSING FOR ROBOT MAPPING AND SIMULTANEOUS HUMAN LOCALIZATION AND ACTIVITY RECOGNITION

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS
ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Kerem Altun
July 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Billur Barshan (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Bülent Özgüler

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Ergin Atalar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Bülent E. Platin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assist. Prof. Dr. Çiğdem Gündüz Demir

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School of Engineering and Science

ABSTRACT

INTELLIGENT SENSING FOR ROBOT MAPPING AND SIMULTANEOUS HUMAN LOCALIZATION AND ACTIVITY RECOGNITION

Kerem Altun
Ph.D. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Billur Barshan
July 2011

We consider three different problems in two different sensing domains, namely ultrasonic sensing and inertial sensing. Since the applications considered in each domain are inherently different, this thesis is composed of two main parts. The approach common to the two parts is that raw data acquired from simple sensors is processed intelligently to extract useful information about the environment.

In the first part, we employ active snake contours and Kohonen's self-organizing feature maps (SOMs) for representing and evaluating discrete point maps of indoor environments efficiently and compactly. We develop a generic error criterion for comparing two different sets of points based on the Euclidean distance measure. The point sets can be chosen as (i) two different sets of map points acquired with different mapping techniques or different sensing modalities, (ii) two sets of fitted curve points to maps extracted by different mapping techniques or sensing modalities, or (iii) a set of extracted map points and a set of fitted curve points. The error criterion makes it possible to compare the accuracy of maps obtained with different techniques among themselves, as well as with an absolute reference. We optimize the parameters of active snake contours and SOMs using uniform sampling of the parameter space and particle swarm optimization. A demonstrative example from ultrasonic mapping is given based on experimental data and compared with a very accurate laser map, considered an absolute reference. Both techniques can fill the erroneous gaps in discrete point maps. Snake curve fitting results in more accurate maps than SOMs because it is more robust to outliers. The two methods and the error criterion are sufficiently general that they can also be applied to discrete point maps acquired with other mapping techniques and other sensing modalities.

In the second part, we use body-worn inertial/magnetic sensor units for recognition of daily and sports activities, as well as for human localization in GPS-denied environments. Each sensor unit comprises a tri-axial gyroscope, a tri-axial accelerometer, and a tri-axial magnetometer. The error characteristics of the sensors are modeled using the Allan variance technique, and the parameters of low- and high-frequency error components are estimated.

Then, we provide a comparative study on the different techniques of classifying human activities that are performed using body-worn miniature inertial and magnetic sensors. Human activities are classified using five sensor units worn on the chest, the arms, and the legs. We compute a large number of features extracted from the sensor data, and reduce these features using both Principal Components Analysis (PCA) and sequential forward feature selection (SFFS). We consider eight different pattern recognition techniques and provide a comparison in terms of the correct classification rates, computational costs, and their training and storage requirements. Results with sensors mounted on various locations on the body are also provided. The results indicate that if the system is trained by the data of an individual person, it is possible to obtain over 99% correct classification rates with a simple quadratic classifier such as the Bayesian decision method. However, if the training data of that person are not available beforehand, one has to resort to more complex classifiers with an expected correct classification rate of about 85%.

We also consider the human localization problem using body-worn inertial/magnetic sensors. Inertial sensors are characterized by drift error caused by the integration of their rate output to get position information. Because of this drift, the position and orientation data obtained from inertial sensor signals are reliable over only short periods of time. Therefore, position updates from externally referenced sensors are essential. However, if the map of the environment is known, the activity context of the user provides information about position. In particular, the switches in the activity context correspond to discrete locations on the map. By performing activity recognition simultaneously with localization, one can detect the activity context switches and use the corresponding position information as position updates in the localization filter. The localization filter also involves a smoother, which combines the two estimates obtained by running the zero-velocity update (ZUPT) algorithm both forward and backward in time.

We performed experiments with eight subjects in an indoor and an outdoor environment involving "walking," "turning," and "standing" activities. Using the error criterion in the first part of the thesis, we show that the position errors can be decreased by about 85% on the average. We also present the results of a 3-D experiment performed in a realistic indoor environment and demonstrate that it is possible to achieve over 90% error reduction in position by performing activity recognition simultaneously with localization.

Keywords: Intelligent sensing; ultrasonic sensing; inertial sensing; robot mapping; sensor error modeling; pattern recognition; wearable computing; human localization; human activity recognition; simultaneous human localization and activity recognition.

ÖZET

ROBOT HARİTALAMA VE İNSANLARDA EŞZAMANLI KONUM BELİRLEME VE AKTİVİTE AYIRDETME İÇİN AKILLI ALGILAMA

Kerem Altun
Elektrik ve Elektronik Mühendisliği, Doktora
Tez Yöneticisi: Prof. Dr. Billur Barshan
Temmuz 2011

Algılama alanının iki ayrı kolu olan ultrasonik algılayıcılar ve eylemsizlik algılayıcıları konularında üç farklı problem ele alınmıştır. Bu iki alandaki uygulamalar temelde birbirinden farklı olduğundan, bu tez iki ana kısımdan oluşmaktadır. Her iki alanda da ortak olarak basit algılayıcılardan alınan ham veriler akıllı yöntemlerle işlenerek dış ortam hakkında anlamlı bilgi edinilmiştir.

İlk kısımda, ayrık noktalardan oluşan iç ortam haritalarını düzenlemek ve değerlendirmek amacıyla yılan eğrileri ve Kohonen ağları yöntemleri uygulanmıştır. İki farklı nokta kümesini karşılaştırmak amacıyla Öklid uzaklığına dayanan genel bir hata ölçütü geliştirilmiştir. Bu nokta kümeleri, (i) farklı haritalama teknikleri veya algılama yöntemleriyle oluşturulmuş iki farklı haritanın, (ii) farklı haritalama teknikleri ile elde edilmiş haritalara uyarlanan iki eğrinin, ya da (iii) birisi bir haritanın, diğeri uyarlanmış bir eğrinin üzerindeki noktalardan oluşacak şekilde seçilebilir. Bu hata ölçütünü kullanarak farklı yöntemlerle elde edilmiş haritaları birbirleriyle ya da bir referans haritasıyla karşılaştırmak mümkündür. Yılan eğrileri ve Kohonen ağlarının parametreleri, parametre uzayını eşit parçalara bölme ve parçacık sürü optimizasyonu yöntemleri kullanılarak eniyilenmiştir. Ultrasonik haritalama alanından deneysel verilere dayalı açıklayıcı bir örnek verilerek mutlak referans olarak kabul edilen bir lazer haritası ile karşılaştırılmıştır. Her iki yöntem de ayrık noktalardan oluşan haritalardaki boşlukları doldurabilmektedir. Yılan eğrileri aykırı değerlere daha duyarsız olduğundan Kohonen ağlarından daha iyi sonuç vermektedir. Tanımlanan hata ölçütü ve kullanılan iki yöntem, başka haritalama yöntemleri veya algılayıcılar ile elde edilmiş haritalara da uygulanabilecek kadar geneldir.

İkinci kısımda, vücuda takılan eylemsizlik algılayıcıları ve manyetik

algılayıcılar yardımıyla günlük aktiviteler ve spor aktivitelerini ayırdetme ve GPS olmayan ortamlarda konum belirleme çalışmaları yapılmıştır. Kullanılan algılayıcı üniteleri üç-eksenli jiroskop, üç-eksenli ivmeölçer ve üç-eksenli manyetometrelerden oluşmaktadır. Allan varyans yöntemiyle bu algılayıcıların hata karakteristikleri modellenmiş, düşük ve yüksek frekanslı hata bileşenlerinin parametreleri kestirilmiştir.

Daha sonra, vücuda takılan eylemsizlik algılayıcıları ve manyetik algılayıcılar ile farklı aktivite ayırdetme yöntemleri arasında karşılaştırmalı bir çalışma yapılmıştır. Aktiviteler, göğüse, kollara ve bacaklara takılan beş adet algılayıcı ünitesi ile ayırdedilmiştir. Algılayıcı verilerinden çok sayıda öznitelik çıkarılmış, daha sonra bu öznitelikler asal bileşenler analizi ve ardışık öznitelik seçme yöntemleri ile sayıca azaltılmıştır. Sekiz farklı örüntü tanıma tekniği ele alınmış, ve bu teknikler doğru ayırdetme yüzdesi, hesaplama maliyeti, eğitme ve veri depolama gereksinimi açılarından karşılaştırılmıştır. Vücudun farklı yerlerine takılan algılayıcılar ile elde edilen sonuçlar sunulmuştur. Sonuçlara göre, belli bir kişinin verisi eğitme için kullanıldığında, Bayesçi karar verme gibi basit bir ikinci derece sınıflandırıcı ile %99'un üzerinde doğru ayırdetme oranlarına ulaşmak mümkündür. Ancak ilgili kişinin verileri eğitme için kullanılmıyorsa daha karmaşık sınıflandırıcılara başvurulmalıdır. Bu durumda beklenen doğru ayırdetme oranı %85 civarındadır.

Son olarak vücuda takılan algılayıcılar ile insanlarda konum belirleme problemi de ele alınmıştır. Eylemsizlik algılayıcılarında karakteristik olarak, konum belirleme için hız verilerinin tümlevi alınması nedeniyle bir sapma hatası bulunmaktadır. Bu sapma hatası sebebiyle eylemsizlik algılayıcılarından elde edilen konum ve yönelim verisi ancak kısa süreler için güvenilir olmaktadır. Dolayısıyla dış ortamdan referans alan algılayıcılar ile konum güncellemesi yapmak gereklidir. Ancak eğer ortamın haritası biliniyorsa, kullanıcının aktivite verisi, konumu hakkında da bilgi vermektedir. Özellikle aktiviteler arası geçişler haritada ayrık konumlara karşılık gelmektedir. Konum belirleme ile eşzamanlı olarak aktivite ayırdetme işlemi de yapıldığında, aktiviteler arası geçişleri belirlemek ve karşılık gelen konum bilgisini konum belirleme süzgecinde güncelleme olarak kullanmak mümkündür. Konum belirleme süzgeci aynı zamanda düzleştirici işlevi görmekte, sıfır hız güncellemesi yoluyla elde edilmiş ileriye ve geriye yönelik kestirimleri birleştirmektedir. Sekiz denek ile iç ve dış ortamda "yürüme," "dönme" ve "ayakta durma" aktivitelerini içeren deneyler yapılmıştır. İlk kısımdaki hata

ölçütü kullanılarak konum hatalarının %85 oranında azaldığı görülmüştür. Bunun dışında bir iç ortamda gerçeğe uygun üç-boyutlu bir deneyin sonuçları verilmektedir. Bu ortamda eşzamanlı konum belirleme ve aktivite ayırdetme yöntemiyle konum hatalarının %90'ın üzerinde düzeltilebildiği gösterilmiştir.

Anahtar sözcükler: Akıllı algılama; ultrasonik algılayıcılar; eylemsizlik algılayıcıları; robot haritalama; algılayıcı hata modellemesi; örüntü tanıma; giyilebilir bilgisayarlar; insanlarda konum belirleme; insanlarda aktivite ayırdetme; insanlarda eşzamanlı konum belirleme ve aktivite ayırdetme.

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Dr. Billur Barshan for her constant guidance throughout my Ph.D. study and her valuable comments and corrections on the manuscripts. Without her expertise and her continuous patience and understanding, completing this study would never have been possible.

I would like to thank my thesis monitoring committee members Prof. Dr. Orhan Arıkan and Asst. Prof. Dr. Selim Aksoy for their supervision and guidance not only at the regular meetings but also any time when I consulted their expertise.

Next in line are my collaborators, Orkun Tunçel and Murat Cihan Yüksek, to whom I am grateful for helping me with the many difficulties I encountered in this study.

I am also grateful to Aslı Ünlügedik, Ayça Arınan, Ceren Vardar, Deniz Kerimoğlu, Esragül Katırcıoğlu, Gözde Özcan, Hande Özcan, Harun Altun, Mahmut Kamil Aslan, Murat Cihan Yüksek, Orkun Tunçel, Özge Topuz, Pınar Tekir, and Temmuz Ceyhan, who took time out of their lives and participated in the experiments in this thesis. The experiments also would not have been realized without the support of Bilkent University Physical Education Unit and Bilkent University Sports Hall staff.

My roommates Tayfun Küçükyılmaz and Yahya Saleh possibly deserve the most gratitude for their patience and support during the course of my Ph.D. study.

I also owe my gratitude to Çiğdem Cengiz, who always stood by my side and put up with me during the Ph.D. years and many more. Her unique friendship is one of the things that made this thesis possible.

I would like to thank Ayça Arınan as well, not because she jokingly kept bugging me for a dedicated paragraph on this page, but for her ever warm friendship,

which is most valuable to me.

I would like to take this opportunity to thank my friends who have been with me throughout my study: Ayşe Küçükyılmaz, Başar Dalkılıç, Cem Gürcan, Dinç Yıldırım, Duygu Can, Elvin Özcan, Emre Kızılırmak, Gökçer Özgür, Hivren Özkol-Kırali, İbrahim Çalışır, İlker Ünsal, Orhan Koyun, Ömer Talat Ayhan, Özgür Orhangazi, Selma Şenozan, Turgut Esencan, Uğraş Dalkılıç, Uğur Susuz, ... Thanks for the fun times shared together. I am sure this list is incomplete; I also apologize for the people I possibly forgot to mention here.

And the last but definitely not the least, I would like to thank my family, who made all this possible by bringing me up to become the person that I am today. I dedicate this thesis to them.

The financial support I received throughout the study from TÜBİTAK-BİDEB for four and a half years, TÜBİTAK project EEEAG-109E059 for eight months, and Bilkent University for two months are also gratefully acknowledged.

To my family

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Networks
ATM	Arc-Transversal Median
BDM	Bayesian Decision Making
BU	Bayesian Update
CCPDF	Class-Conditional Probability Density Function
DFT	Discrete Fourier Transform
DM	Directional Maximum
DOF	Degrees-of-Freedom
DOI	Direction-of-Interest
DTW	Dynamic Time Warping
FPR	False Positive Rate
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
k-NN	k-Nearest Neighbor
L1O	Leave-One-Out
LOS	Line of Sight
LSM	Least Squares Method
MEMS	Micro Electro-Mechanical Systems
MP	Morphological Processing
PCA	Principal Components Analysis
PDR	Pedestrian Dead Reckoning
PM	Point Marking
PSD	Power Spectral Density
PSO	Particle Swarm Optimization
RBA	Rule-Based Algorithm

RCD	Regions of Constant Depth
RFID	Radio-Frequency Identification
ROC	Receiver Operating Characteristics
RRSS	Repeated Random Sub-Sampling
SBFS	Sequential Backward Feature Selection
SFFS	Sequential Forward Feature Selection
SOM	Kohonen's Self-Organizing Map
SVM	Support Vector Machines
TBF	Triangulation-Based Fusion
TOF	Time-of-Flight
TPR	True Positive Rate
UAM	Ultrasonic Arc Map
VT	Voting and Thresholding
WiFi	Wireless Fidelity
WS	Walking-to-Standing
WT	Walking-to-Turning
WZ	Walking-to-Stairs
ZUPT	Zero Velocity Update

List of Symbols

$\mathbf{a}_G$	acceleration vector in the local navigation frame
$\mathbf{a}_S$	acceleration vector in the sensor frame
$\mathbf{A}$	snake evolution matrix
$b(t)$	slow-varying bias model
$\mathbf{b}$	manufacturer-supplied bias vector
B	bias instability coefficient
c_1, c_2	bias model coefficients
$d(\cdot, \cdot)$	Euclidean distance between two points
d	distance traveled on the xy-plane
d_z	distance traveled on the z-direction
$D_Q(\cdot)$	Euclidean distance map of point set Q
E_{ext}	external energy of the snake
E_{int}	internal energy of the snake
E_{snake}	total energy of the snake
$\mathcal{E}_{(P-Q)}$	average distance between point sets P and Q
f_0	sinusoidal noise frequency
$\mathbf{G}$	gain matrix
g	gravitational acceleration of the Earth
$gBest$	global best value in PSO
$\mathbf{g}_n$	global best position in PSO
I_{step}	indicator variable of stance/swing phase
$\mathbf{I}$	identity matrix
$J(\mathbf{x})$	objective function
k_λ	learning rate decay coefficient
k_σ	standard deviation decay coefficient
$K(\cdot, \cdot)$	SVM kernel function
$\mathbf{K}_T$	conversion matrix
M_k	set of points on the kth ultrasonic map

$\mathbf{M}$	misalignment matrix
n	iteration number
N_k	number of points on the kth snake of SOM curve
N_s	inertial/magnetic sensor signal length
o	output value of neurons in ANN
$\mathbf{O}$	matrix representing higher-order modeling
$p_x(s)$, $p_y(s)$	x and y coordinates of snake or SOM curve
$pBest$	previous best value in PSO
$\mathbf{p}_n$	previous best position in PSO
P	finite set of discrete points
$\mathbf{P}_{ws,i}$	covariance matrix for the ith WS switch location
$\mathbf{P}_{wt,j}$	covariance matrix for the jth WT switch location
$\mathbf{P}_{\boldsymbol{\xi}}$	covariance matrix of the initial condition
q_c	correlated noise coefficient
q_{GS}	orientation of the sensor frame with respect to the local navigation frame
Q	finite set of discrete points
$\mathbf{Q}$	process noise covariance
$\mathbf{r}$	raw A/D converter reading
$R_{ss}(\Delta)$	autocorrelation of signal $\mathbf{s}$
$\hat{R}_{ss}(\Delta)$	autocorrelation estimate (unbiased)
$\tilde{R}_{ss}(\Delta)$	autocorrelation estimate (biased)
$\mathbf{R}_\theta$	rotation matrix on the plane
s	arc length parameter
$s[i]$, s_i	ith sample of signal $\mathbf{s}$
$\mathbf{s}$	measured inertial/magnetic signal
S_k	kth snake or SOM curve fitted
$S_s(f)$	power spectral density
$S_{DFT}(k)$	discrete Fourier transform
t	target output value in ANN
T	bias model time constant
T_c	correlated noise correlation time
$\mathbf{u}$	input to the localization state equation
$U(\cdot)$	potential function
$\mathbf{v}(s)$	parametric snake curve

w	inertia weight in PSO
$\mathbf{w}$	ANN weight
x, y, z	coordinates of position
$\mathbf{x}$	position of a particle in PSO
$\dot{\mathbf{x}}$	velocity of a particle in PSO
α	elasticity parameter
β	rigidity parameter
γ	Euler step size
Δ	first-order difference operator
ζ	SVM kernel parameter
κ	external force weight
λ	learning rate
$\mu_{\mathbf{s}}$	mean of signal $\mathbf{s}$
$\boldsymbol{\mu}_{ws,i}$	ith WS switch location
$\boldsymbol{\mu}_{wt,j}$	jth WT switch location
$\boldsymbol{\mu}_{\boldsymbol{\xi}}$	mean value of the initial condition
$\boldsymbol{\xi}$	state vector (position)
$\hat{\boldsymbol{\xi}}_f$	state vector forward estimate
$\hat{\boldsymbol{\xi}}_b$	state vector backward estimate
$\hat{\boldsymbol{\xi}}$	combined estimate
σ	standard deviation
$\sigma(\tau)$	Allan standard deviation
σ^2	variance
$\sigma^2(\tau)$	Allan variance
σ_K	rate random walk coefficient
σ_N	white noise power
σ_Q	quantization noise coefficient
$\boldsymbol{\Sigma}_f$	covariance matrix of the forward estimate
$\boldsymbol{\Sigma}_b$	covariance matrix of the backward estimate
$\boldsymbol{\Sigma}$	covariance matrix of the combined estimate
τ	averaging time
ψ	yaw angle
ω	angular velocity
Ω_T	angular speed threshold
Ω_0	sinusoidal noise amplitude

Chapter 1

Introduction

Sensing and perception are two key research areas in robotics as well as in human-computer interaction. These two terms are linked through intelligent processing, where raw sensor data should be processed intelligently to gain information about the environment. In this context, sensing should be regarded as a low-level process, where various physical quantities are measured using an electronic device. Correct processing and interpretation of the measured quantities by an intelligent agent such as a robot or a computer leads to perception, which stands for a higher-level understanding of the environment by the agent. The perceived information about the environment can then be used by the agent for many purposes; for example, navigation or mapping if the agent is a mobile robot, or if the agent is a wearable system worn by a human (who is part of the environment), the perceived information can be used in human-computer interaction.

The concepts in this thesis are situated at the boundary between sensing and perception. We use simple low-cost sensors to begin with, and apply intelligent processing methods to extract meaningful information from sensor data. We consider three different applications in two different sensing domains, namely ultrasonic sensing and inertial sensing. Since the applications considered in each domain are inherently different, this thesis is composed of two main parts. We motivate these applications in the rest of this chapter, and refer the reader to the corresponding chapter and our related publications for detailed descriptions

of the applied methods.

Part I includes Chapter 2, where we use active snake contours and Kohonen's self-organizing feature maps to represent the ultrasonic map data more compactly and efficiently. This part is based on the publications [1, 2, 3, 4]. In this chapter, we fit curves to the processed ultrasonic maps using these two methods, and eliminate the undesired artifacts on the map caused by problems inherent to ultrasonic sensors such as multiple and higher-order reflections and cross-talk. The concepts in this part can be applied to a discrete point map obtained with other sensing modalities as well, such as infrared or laser, in order to eliminate undesired outlier points while keeping the structure of the map.

Part II includes Chapters 3, 4, and 5. In this part, we address two problems involving inertial sensors in wearable systems: human activity recognition and localization. We also combine these problems in Chapter 5, using activity recognition information to aid in localization. In Chapter 3, we first use the Allan variance method to model the low- and high-frequency components of the noise in inertial sensor data. The transients are removed using a curve fitting procedure before the Allan variance method is applied. These error models can be used in navigation applications where the gyroscope and accelerometer outputs are integrated in a strap-down navigation system. Chapter 4 is based on the publications [5, 6]. In this chapter, we use body-worn inertial/magnetic sensor units and apply pattern recognition techniques to determine the activity context of the user. For our preliminary studies on activity recognition, the reader is referred to [7, 8], which is not included in this thesis. In [7, 8], we use two uniaxial gyroscopes worn on the leg to distinguish between eight leg motions. We report our classification results by considering various pattern recognition and feature selection techniques, which forms a basis for our studies on the recognition of daily and sports activities. Human activity recognition has many applications, some of them being biomechanics, home-based rehabilitation, remote monitoring of the elderly people and children, detecting and classifying falls, and motion capture and animation. With the advancement of mobile technology, inertial sensing equipment are standardized in new generation mobile phones. Activity recognition methods implemented in these phones enable the phone to have information

about the user, which can help in several human-computer interaction scenarios. Another increasingly popular application in mobile technology is location-based services [9], where the phone user is provided with different services depending on his/her location estimated with GPS or other absolute sensing mechanisms integrated in the mobile phone. Inertial sensors can be used for aiding localization as well, which is the subject of Chapter 5.

In Chapter 5, we consider the problem of localization using inertial sensors only. Due to the integration operations involved in estimating position and orientation from inertial sensor data, the drift errors in the position and orientation are unavoidable. Since these errors are cumulative, the overall error grows without bound over long periods of time. Therefore, position data from other absolute sensing modalities are usually utilized in order to correct the position estimates from inertial sensors. However, in this chapter, we use activity recognition information simultaneously with the localization algorithm and a given map of the environment to provide position updates. If the map is known, activity recognition information and especially switches between the activity context give important information about the location. For example, in an indoor setting, switching from walking to ascending stairs means that the person is at one of the stair areas of the building, just having started ascending stairs. This usually corresponds to a few discrete locations on the given map. By simultaneously performing activity recognition and localization, position updates provided by activity context switch detection can be used in the localization process, and accurate localization can be achieved without resorting to externally referenced sensing methods. Our method can especially be used in underground mines, where usually no absolute position sensing infrastructure is present. It can also aid the localization performance in outdoor environments to reduce the GPS positioning accuracy (usually in the order of several meters) or in urban indoor environments to reduce the GSM positioning accuracy (usually in the order of 100 meters). Simultaneous activity recognition and localization have not been done before as it is handled here. We were able to find only two papers in the previous work, one of which also uses data from externally referenced sensors (GPS) and the other uses activity recognition cues not for position updates, but to detect human gait phases.

The references are provided in the chapter text. Therefore, to our knowledge, the work in this chapter is the first in literature to perform activity recognition and localization simultaneously, using activity recognition cues for position updates, with only using body-worn inertial and magnetic sensors.

In Chapter 6, we summarize our results and provide future research directions for the applications considered in this thesis.

Part I

Ultrasonic Sensing

Chapter 2

Representing and Evaluating Ultrasonic Maps Using Active Snake Contours and Kohonen's Self-Organizing Feature Maps

2.1 Introduction and Previous Work

Autonomous robots must be aware of their environment and interact with it through sensory feedback. The potential of simple and inexpensive sensors should be fully exploited for this purpose before more expensive alternatives with higher resolutions and resource requirements are considered. Ultrasonic sensors have been widely employed because of their accurate range measurements, robustness, low cost, and simple hardware interface. We explore the limits of these sensors in mapping through intelligent processing and representation of ultrasonic range measurements. When coupled with intelligent processing, ultrasonic sensors are a useful alternative to more complex laser and camera systems. Furthermore, it may not be possible to use the latter in some environments due to surface characteristics or insufficient ambient light. Despite their advantages, the frequency

range at which air-borne ultrasonic transducers operate is associated with a large beamwidth that results in low angular resolution and uncertainty at the location of the echo-producing feature. Furthermore, ultrasonic range maps are characterized by echo returns resulting from multiple and higher-order reflections, cross-talk between transducers, and noise. These maps are extremely inefficient and unintuitive representations of even the simplest environmental structures that generate them. Thus, having an intrinsic uncertainty of the actual angular direction of the range measurement and being prone to the various phenomena mentioned above, a considerable amount of processing, interpretation, and modeling of ultrasonic data is necessary.

In this study, we propose two approaches for efficiently representing and evaluating discrete point maps of an environment obtained with different ultrasonic arc map (UAM) processing techniques. The first approach involves fitting active snake contours [10] to the processed UAMs. Snakes are inherently closed curves suitable for representing the features of an environment on a given map. The second approach involves fitting Kohonen's self-organizing feature maps (SOMs) [11] (which can be implemented as either closed or open curves, using the positions of map points as input features) to an artificial neural network. In ultrasonic maps, gaps frequently occur where a number of contiguous points are marked as empty despite the fact that they are occupied. Both approaches generate parametric curves that fill the erroneous gaps between map points and allow the map to be represented and stored more compactly and smoothly, with fewer points and using a limited number of parameters. These methods also make it possible to compare the accuracy of maps acquired with different techniques or sensing modalities among themselves, as well as to an absolute reference.

2.1.1 Ultrasonic Sensor Signal Processing

In earlier work, there have been two basic approaches to representing ultrasonic data: feature based and grid based. Grid-based approaches do not attempt to make difficult geometric decisions early in the interpretation process, unlike feature-based approaches that extract the geometry of the sensor data as the first

step. As a first attempt to feature-based mapping, several researchers have fitted line segments to ultrasonic data as features that crudely approximate the room geometry [12, 13, 14]. This approach proves to be difficult and brittle because straight lines fitted to time-of-flight (TOF) data do not necessarily match or align with the world model, and may yield many erroneous line segments. Improving the algorithms for detecting line segments and including heuristics do not really solve the problem. A more physically meaningful representation is to use *regions of constant depth* (RCDs) as features, which are circular arcs from specularly reflecting surfaces that are natural features of the raw ultrasonic TOF data. These arcs were first reported in [15] and further elaborated on in [16]. They are obtained by placing a small mark along the line of sight (LOS) at the range corresponding to the measured TOF value. In specularly reflecting environments, an accumulation of such marks usually produces arc-like features. As a more general approach that is not limited to specularly reflecting surfaces, the angular uncertainty in the range measurements has been represented by UAMs [17] that preserve more information (see Figure 2.3(a) for a sample UAM). Note that the arcs in the UAM are uncertainty arcs and different than the arcs corresponding to RCDs. The UAMs are obtained by drawing arcs spanning the beamwidth of the sensor at the measured range, representing the angular uncertainty of the object location and indicating that the echo-producing object can lie anywhere on the arc. The probability of the reflection point being in the middle of the arc is the largest, symmetrically decreasing towards the sides. In the literature, the probability of occupancy along the arc has been modeled by Elfes heuristically [18, 19].

Thus, when the same transducer transmits and receives, all that is known is that the reflection point lies on a circular arc of radius r, with a larger probability in the middle of the arc. More generally, when one transducer transmits and another receives, it is known that the reflection point lies on the arc of an ellipse whose focal points are the transmitting and receiving elements. The arcs are tangent to the reflecting surface at the actual point(s) of reflection.

Completely specular and completely diffuse (Lambertian) reflection are both idealizations corresponding to the two extreme cases of the actual spectrum of real reflection characteristics. These two extrema are unreachable in practice

and many real environments are, in fact, compositions of both specularly and diffusely reflecting elements. Compared to RCDs, constructing UAMs is more generally applicable in that they can be generated for environments comprised of both specularly and diffusely reflecting surfaces as is the case for many typical indoor environments. On the other hand, RCDs are structures that occur in environments where specular reflections are dominant. In earlier work, techniques based on the Hough transform have been applied to detect line and point features from arcs for both airborne and underwater ultrasonic data [20, 21].

2.1.2 UAM Processing Methods

Since the UAM is generated by drawing an arc for each received echo, the resulting map has many redundant points, as well as artifacts caused by cross-talk, and multiple and higher-order reflections (Figure 2.3(a)). Several techniques exist in the literature that can be used to process the UAMs. These techniques, listed in Table 2.1, are described in this section.

Table 2.1: UAM processing techniques used in this study and their indices.

k	UAM processing technique
1	point marking (PM) [15]
2	voting and thresholding (VT) [22]
3	directional maximum (DM) [19]
4	morphological processing (MP) [17]
5	Bayesian update (BU) [18]
6	arc-transversal median (ATM-org) [23]
7	modified ATM (ATM-mod) [19]
8	triangulation-based fusion (TBF) [24]

2.1.2.1 Point Marking (PM)

This is the simplest approach, where a mark is placed along the LOS at the measured range [15]. This method produces reasonable estimates for the locations of objects if the arc of the cone is small. This can be the case at higher frequencies

of operation where the corresponding sensor beamwidth is small or at nearby ranges. Since every arc is reduced to a single point, this technique cannot eliminate any of the outlying TOF readings. The resulting map is usually inaccurate with large angular errors and artifacts.

2.1.2.2 Voting and Thresholding (VT)

In this technique, each pixel stores the number of arcs crossing that pixel, resulting in a 2-D array of occupancy counts for the pixels [22]. By simply thresholding this array and zeroing the pixels lower than the threshold, artifacts can be eliminated and the map is extracted.

2.1.2.3 Directional Maximum (DM)

This technique is based on the idea that in processing the acquired range data, there is a direction-of-interest (DOI) associated with each detected echo. Ideally, the DOI corresponds to the direction of a perpendicular line drawn from the sensor to the nearest surface from which an echo is detected. However, in practice, due to the angular uncertainty of the object position, the DOI can be approximated as the LOS of the sensor when an echo is detected. Since prior information on the environment is usually unavailable, the DOI needs to be updated while sensory data are being collected and processed on-line [19].

In the implementation, the number of arcs crossing each pixel of the UAM is counted and stored, and a suitable threshold value is chosen, exactly the same way as in the VT method. The novelty of the DM method is the processing done along the DOI. Once the DOI for a measurement is determined using a suitable procedure, the UAM is processed along this DOI as follows: The array of pixels along the DOI is inspected and the pixel(s) exceeding the threshold with the maximum count is kept, while the remaining pixels along the DOI are zeroed out. If there exist more than one maxima, the algorithm takes their median (if the number of maxima is odd, the maxima in the middle is taken; if the number

is even, one of the two middle maxima is randomly selected). This way, most of the artifacts of the UAM can be removed.

2.1.2.4 Morphological Processing (MP)

The processing of UAMs using morphological operators was first proposed in [17]. This approach exploits neighboring relationships and provides an easy to implement yet effective solution to ultrasonic map building. By applying binary morphological operators, one can eliminate the artifacts of the UAM and extract the surface profile.

2.1.2.5 Bayesian Update Scheme for Occupancy Grids (BU)

Occupancy grids were first introduced by Elfes, and a Bayesian scheme for updating their probabilities of occupancy and emptiness was proposed in [18] and verified by ultrasonic data. Starting with a blank or completely uncertain occupancy grid, each range measurement updates the probabilities of emptiness and occupancy in a Bayesian manner. The reader is referred to [18] for a detailed description of the method and [19] for its implementation in this work.

2.1.2.6 Triangulation-Based Fusion (TBF)

The TBF method is primarily developed for accurately detecting the edge-like features in the environment based on triangulation [24]. The triangulation equations involved are not suitable for accurately localizing planar walls.

Unlike the previously introduced grid-based techniques, the TBF method extracts the features of the environment by using a geometric model suitable for edge-like features. In addition, TBF considers a sliding window of ultrasonic scans where the number of rows of the sliding window corresponds to the number of ultrasonic sensors fired, and the number of columns corresponds to the number of most recent ultrasonic scans to be processed by the algorithm. TBF is

focused on detection of edge-like features located at ≤ 5 m. The other methods consider all of the arcs in the UAM corresponding to all ranges, and are suitable for detecting all types of features.

2.1.2.7 Arc-Transversal Median (ATM)

The ATM algorithm requires both extensive bookkeeping and considerable amount of processing [23]. For each arc in the UAM, the positions of the intersection(s) with other arcs, if they exist, are recorded. For arcs without any intersections, the mid-point of the arc is taken to represent the actual point of reflection (as in PM). If the arc has a single intersection, the algorithm uses the intersection point as the location of the reflecting object. For arcs with more intersections, the median of the positions of the intersection points with other arcs is chosen to represent the actual point of reflection. In [23], the median operation is applied when an arc has *three or more* intersection points. If there is an even number of intersections, the algorithm uses the mean of the two middle values (except that arcs with two intersections are ignored). It can be considered as a much improved version of the PM approach.

We have also implemented a modified version of the algorithm (ATM-mod) where we ignored arcs with no intersections. Furthermore, since we could not see any reason why arcs with two intersections should not be considered, we took the mean of the two intersection points.

2.2 Methodology

One of the purposes of this study is to evaluate and compare the performances of different UAM processing techniques, by fitting active snake contours and SOMs to the map data. To establish a quantitative measure of the performance, we first define a generic error criterion between two sets of points, based on the Euclidean distance.

2.2.1 The Error Criterion

Let $P \subset \mathbb{R}^3$ and $Q \subset \mathbb{R}^3$ be two finite sets of arbitrary points with N_1 points in set P and N_2 points in set Q. We do not require the correspondence between the two sets of points to be known. Each point set could correspond to either (i) a set of map points acquired by different mapping techniques or different sensing modalities (e.g., laser, ultrasonic, or infrared map points), (ii) discrete points corresponding to an absolute reference (the true map), or (iii) some curve (in 2-D) or shape (in 3-D) fit to the map points (e.g., polynomials, snake curves, or spherical caps). The absolute reference (or ground truth) could be an available true map or plan of the environment or could be acquired by making range or time-of-flight measurements through a very accurate sensing system.

The well-known Euclidean distance $d(\mathbf{p}_i, \mathbf{q}_j) : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^{\geq 0}$ of the ith point in set P with position vector $\mathbf{p}_i = (p_{xi}, p_{yi}, p_{zi})^T$ to the jth point $\mathbf{q}_j = (q_{xj}, q_{yj}, q_{zj})^T$ in set Q is given by:

$$d(\mathbf{p}_i, \mathbf{q}_j) = \sqrt{(p_{xi} - q_{xj})^2 + (p_{yi} - q_{yj})^2 + (p_{zi} - q_{zj})^2} \tag{2.1}$$

where $i \in \{1, \ldots, N_1\}$ and $j \in \{1, \ldots, N_2\}$.

In [25], three different metrics to measure the similarity between two sets of points are considered and compared, each with certain advantages and disadvantages. In this work, we use the most favorable of them to measure the closeness or similarity between sets P and Q:

$$\mathcal{E}_{(P-Q)} = \frac{1}{2}\left(\frac{1}{N_1}\sum_{i=1}^{N_1}\min_{\mathbf{q}_j \in Q}\{d(\mathbf{p}_i, \mathbf{q}_j)\} + \frac{1}{N_2}\sum_{j=1}^{N_2}\min_{\mathbf{p}_i \in P}\{d(\mathbf{p}_i, \mathbf{q}_j)\}\right) \tag{2.2}$$

According to this criterion, we take into account all points in the two sets and find the distance of every point in set P to the nearest point in set Q and average them, and vice versa. The two terms in Equation (2.2) are also averaged, so that the criterion is symmetric with respect to P and Q. If the two sets of points are completely coincident, the average distance between them will be zero. If one set is a subset of the other, there will be some error. Had an asymmetric criterion been employed, say, including only the first (or the second) term in

Equation (2.2), the error would have been zero when $P \subset Q$ (or $Q \subset P$). Gaps occurring in the maps and sparsity are penalized by this error criterion, resulting in larger errors on average.

The error criterion we propose is sufficiently general that it can be used to compare any two arbitrary sets of points. This makes it possible to compare the accuracy of discrete point maps acquired with different techniques or sensing modalities with an absolute reference, as well as among themselves, both in 2-D and 3-D. When curves or shapes (e.g., lines, polynomials, snakes, spherical or elliptical caps) are fitted to the map points, the criterion proposed here also enables us to assess the goodness of fit of the curve or shape to the map points. In other words, a fitted curve or shape comprised of a finite number of points can be treated in exactly the same way.

2.2.2 Active Snake Contours

A snake, or an active contour, first introduced by Kass *et al.* [10], can be described as a continuous deformable closed curve. Active snake contours have been commonly used in image processing for edge detection and segmentation [10, 26, 27, 28], and have been mostly classified into three categories [29]:

- point-based snakes, where the curve is represented as a collection of discrete points,
- parametric snakes, where the curve is described using combinations of basis functions, and
- geometric snakes, where the curve is represented as a level set of a higher-dimensional surface.

This classification is not universal and many authors categorize snakes merely into two, namely parametric and geometric, with the first category stated above considered a special kind of parametric snake. The snake used in this study

belongs to the first category. The variants of snakes are analyzed in a unified manner in [30].

We define a snake as a parameterized closed curve $\mathbf{v}(s) = [p_x(s), p_y(s)]^T$, $s \in [0, 1]$, where $p_x(s)$ and $p_y(s)$ are functions representing the Cartesian coordinates of the snake in 2-D and s is the normalized arc length parameter of the snake curve. This parameterization is dimensionless. The deformation of the snake is controlled by internal and external forces. Internal forces impose elasticity and rigidity constraints on the curve, whereas external forces stretch or shrink the curve to fit to the image data. The total energy of the snake curve is given by the functional

$$E_{\text{snake}} = \int_0^1 \left[E_{\text{int}}(\mathbf{v}(s)) + E_{\text{ext}}(\mathbf{v}(s))\right] ds \tag{2.3}$$

The internal energy component is given by

$$E_{\text{int}}(\mathbf{v}(s)) = \frac{1}{2}\left(\alpha \left\| \frac{d(\mathbf{v}(s))}{ds} \right\|^2 + \beta \left\| \frac{d^2(\mathbf{v}(s))}{ds^2} \right\|^2\right) \tag{2.4}$$

where α is the elasticity parameter and β is the rigidity parameter, and $\|\cdot\|$ denotes the 2-norm. The first derivative term in Equation (2.4) penalizes long curves, whereas the second derivative term penalizes sharp curvatures. This internal energy definition is used in many applications, possibly with varying $\alpha(s)$ and $\beta(s)$. The use of other internal energy expressions is not very common.

The external energy component is denoted by $E_{\text{ext}}(\mathbf{v}(s)) = U(\mathbf{v}(s))$, where U is a potential function that depends on the image data. In general, the potential function can be selected in different ways, depending on the application. However, it must be at minimum on the edges of the image if the snake is to be used for edge detection, segmentation, or finding the boundaries of an environment as in our application. Kass *et al.* suggest using the negative of the image gradient magnitude as a potential function [10]. However, this is only feasible if the snake is initialized close to the image boundaries, otherwise the snake curve would be stuck in local minima or a flat region of the potential function. Filtering the image with a Gaussian low-pass filter is also suggested in the same paper to increase the capture range of the snake, but this causes the edges to become

blurry, thus reducing map accuracy. In black-on-white and gray-level images, the image intensity can be used as the potential function, either in binary form or convolved with a Gaussian blur [27]. Obviously, this method also suffers from the drawbacks stated above. Another solution, proposed in [28], is using a *distance map* as a potential function to increase the capture range of the contour, which is the approach used in this study.

In this work, we chose to use a potential function for the external energy term based on the Euclidean distance map, as suggested in [28]. Although our problem is in 2-D, let us first make a more general definition of a distance map in 3-D.

2.2.2.1 Euclidean Distance Map

Let $Q \subset \mathbb{R}^3$ be a finite set of arbitrary points. For all points $\mathbf{p}$ of the mapped region, we define a distance map $D_Q(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}^{\geq 0}$ between a point $\mathbf{p}$ and set Q as the minimum of the Euclidean distances of that point to all the points in the set Q. That is,

$$D_Q(\mathbf{p}) = \min_{\mathbf{q}_j \in Q} \{d(\mathbf{p}, \mathbf{q}_j)\} \quad j \in \{1, \ldots, N_2\} \tag{2.5}$$

where $\mathbf{q}_j$ is the position vector of the jth point in the set Q. According to this definition, the two summands in Equation (2.2) are, in fact, nothing but distance functions and the error criterion can be rewritten as:

$$\mathcal{E}_{(P-Q)} = \frac{1}{2}\left(\frac{1}{N_1}\sum_{i=1}^{N_1} D_Q(\mathbf{p}_i) + \frac{1}{N_2}\sum_{j=1}^{N_2} D_P(\mathbf{q}_j)\right) \tag{2.6}$$

Computing the Euclidean distance map is costly, and a number of algorithms and other distance functions have been proposed in the literature to approximate it [31, 32]. In this study, the Euclidean distance map is implemented in its original form and the potential function is chosen as:

$$U(\mathbf{p}) = D_Q(\mathbf{p}) = \min_{\mathbf{q}_j \in Q} \{d(\mathbf{p}, \mathbf{q}_j)\} \quad j \in \{1, \ldots, N_2\} \tag{2.7}$$

for all points $\mathbf{p}$ of the mapped region and a point set Q.

Approaches that do not use a potential function as the external energy term also exist in the literature [33]. Such approaches relax the constraint that the external forces pulling the snake towards the edges should be conservative, i.e., derived from a potential field. For example, Xu *et al.* define a non-conservative force field representing the external forces and use force-balance equations rather than an energy-based approach to solve the problem [33].

2.2.2.2 Snake Curve Evolution

With the above definitions of external and internal energy, calculus of variations can be used to find the curve that minimizes the energy functional in Equation (2.3). The minimizing curve should satisfy the following Euler-Lagrange equation [10]:

$$\alpha \frac{d^2\mathbf{v}(s)}{ds^2} - \beta \frac{d^4\mathbf{v}(s)}{ds^4} - \nabla U(\mathbf{v}(s)) = 0 \tag{2.8}$$

Although it may be possible to solve this equation analytically for some special cases, a general analytical solution does not exist. The common practice is to initialize an arbitrary time-dependent snake curve $\mathbf{v}(s,t)$. Equation (2.8) is then set equal to the time derivative of the snake, where a solution is found when the time derivative vanishes. That is,

$$\alpha \frac{\partial^2\mathbf{v}(s,t)}{\partial s^2} - \beta \frac{\partial^4\mathbf{v}(s,t)}{\partial s^4} - \nabla U(\mathbf{v}(s,t)) = \frac{\partial \mathbf{v}(s,t)}{\partial t} \tag{2.9}$$

These equations are then discretized to find a numerical solution. The snake is treated as a collection of discrete points joined by straight lines and is initialized on the image. Approximating the derivatives by finite differences, the evolution equations of the snake reduce to the following:

$$\mathbf{p}_x(n+1) = (\mathbf{A} + \gamma\mathbf{I})^{-1} \left(\gamma\, \mathbf{p}_x(n) - \kappa \left. \frac{\partial U}{\partial p_x} \right|_{[\mathbf{p}_x(n), \mathbf{p}_y(n)]} \right) \tag{2.10}$$

$$\mathbf{p}_y(n+1) = (\mathbf{A} + \gamma\,\mathbf{I})^{-1} \left(\gamma\, \mathbf{p}_y(n) - \kappa \left. \frac{\partial U}{\partial p_y} \right|_{[\mathbf{p}_x(n), \mathbf{p}_y(n)]} \right) \tag{2.11}$$

Here, n is the current time (or iteration) step, $\mathbf{p}_x(n)$ and $\mathbf{p}_y(n)$ are vectors representing the positions of the collection of discrete points on the snake at time n,

γ is the Euler step size, and κ is a weight factor for the external force. **I** is the identity matrix of the appropriate size and **A** is a penta-diagonal banded matrix that depends on α and β. The sizes of the matrices **A** and **I** are determined by the number of points on the snake, which may change as the algorithm is executed.

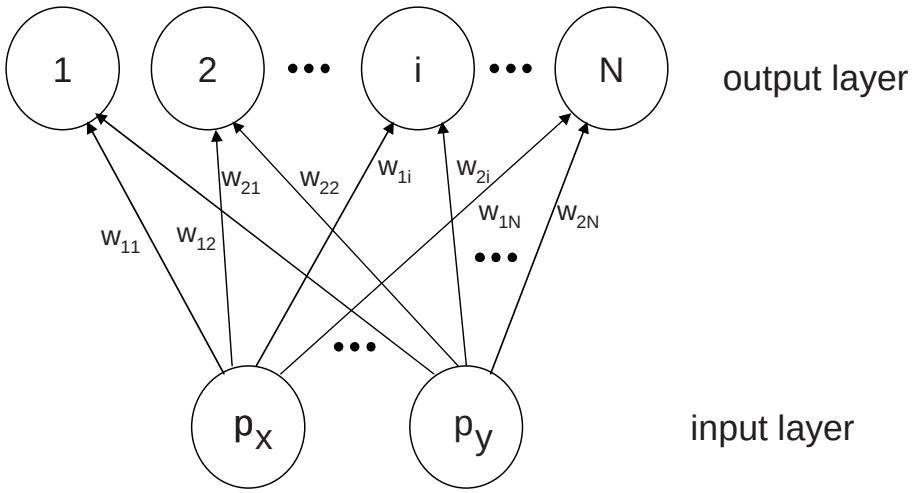


Figure 2.1: Structure of the 1-D SOM.

2.2.3 Kohonen's Self-Organizing Maps

Another method used in map representation and in evaluating the different techniques is the self-organizing map introduced by Kohonen [11], which is basically an artificial neural network that uses a form of unsupervised learning, and is suitable for applications where the topology of the data is to be learned. Robots that learn the environment structure using artificial neural networks are reported on in [34, 35]. SOMs for curve and surface reconstruction have been used in applications such as computer-aided design (CAD) modeling of objects having irregular shapes [36, 37]. In this study, we use the SOM for fitting curves to the ultrasonic map points obtained with the different UAM processing techniques.

An SOM is an artificial neural network with two layers. We use a 1-D SOM,

whose structure is illustrated in Figure 2.1. The two neurons at the input layer are used to input the p_x and p_y coordinates of a map point. Each neuron at the output layer represents a point on the curve to be fitted, and the associated connection weights are the p_x and p_y coordinates of this point. The output neurons are arranged as a chain-like structure, where each neuron, except those at the two ends, has two neighbors. This neighborhood affects the weight updates described below. For each input map point, the winning neuron is determined to be the closest point on the curve to that input. Thus, for the input map point $\mathbf{p} = (p_x, p_y)^T$, output neuron weights $\mathbf{w}_i = (w_{1i}, w_{2i})^T$, and a total of N points on the curve, the index i^* of the winning neuron is given by:

$$i^* = \arg \min_{i=1,\ldots,N} \sqrt{(p_x - w_{1i})^2 + (p_y - w_{2i})^2} \tag{2.12}$$

Through the use of a Gaussian function $g_{0,\sigma(n)}(\cdot)$, updating the weights is done such that the weight update of one neuron also affects the neighboring neurons. Then, for all neurons $i = 1, \ldots, N$, the weight update rule is

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \lambda(n)\, g_{0,\sigma(n)}(|i - i^*|)\, [\mathbf{p}(n) - \mathbf{w}_i(n)] \tag{2.13}$$

where n is the iteration step, $\mathbf{w}_i(n)$ is the 2×1 weight vector of neuron i, $\mathbf{p}(n)$ is the position vector of the input map point, $\lambda(n)$ is the time-dependent learning rate, and $g_{0,\sigma(n)}(\cdot)$ is a 1-D Gaussian function with zero mean and standard deviation $\sigma(n)$. Note that $\lambda(n)$ and $\sigma(n)$ are functions of the iteration number n and should be decreased at the end of each epoch as the iterations are performed, with respective decay coefficients k_λ and k_σ. An epoch is completed when all map points are given as input to the SOM once.

2.2.4 Parameter Selection and Optimization Methods

Active snake contours and SOMs have a number of parameters that affect the convergence characteristics and performance of curve fitting. Since each problem or situation requires a different set of parameter values, universally accepted values for these parameters do not exist in the literature. For the map points extracted from an unknown environment, it is difficult to guess or estimate these

parameter values beforehand. The parameters depend on the shape and the nature of the environment as well as on the mapping technique used. For example, a structured environment mostly composed of specularly reflecting surfaces may require a different set of parameters than an unstructured environment with some combination of specularly and diffusely reflecting surfaces. Optimization methods can be applied to select the best parameter values specific to the problem at hand. Below, we provide guidelines for selecting these parameters and suggest two alternative approaches that can be employed for this purpose.

For the active contour method, the parameters that must be selected are α, β (Equation (2.4)), γ, and κ (Equations (2.10) and (2.11)). As stated above, α penalizes elongation and β penalizes bending or sharpness of the snake curve. For example, selecting a small β value enforces the second derivative in the energy term to have smaller weight, thus allowing sharp corners in the snake. The parameter γ is the Euler step size of the discretization and κ is a weight factor for the external force. Some authors use $\kappa = 1$, as in the original definition, where it is also stated that the step size γ should be reduced if the external force becomes large [10]. Including a κ parameter (different than one) provides more user control of the problem. Determining these parameters depends on the initialization and the required curve features, such as length and curvature.

For the SOM method, the parameters that affect convergence and the performance of curve fitting are the learning rate $\lambda(n)$ and the standard deviation $\sigma(n)$. These are functions of the iteration number n and should be decreased as the iterations are performed. In our implementation, we multiply each with decay coefficients $0 < k_\lambda < 1$ and $0 < k_\sigma < 1$ at the end of each epoch. Thus, the parameters to be chosen are $\lambda(0), \sigma(0), k_\lambda$, and k_σ. The initial learning rate $\lambda(0)$ is usually chosen to be less than one in order not to "overshoot" the map points in fitting. If the curve is initialized close to the boundaries of the mapped region, $\lambda(0)$ and $\sigma(0)$ should be chosen smaller. The Gaussian function has a value of one at its peak point, where $i = i^*$, and decreases symmetrically and gradually. Using a Gaussian function in Equation (2.13) to update the weights allows all neurons to be updated at the same time, resulting in a smoother fitting curve to the input data at each iteration. If we visualize the 1-D SOM as a chain-like

structure, $\sigma(n)$ determines the tightness/looseness of the chain. The larger it is, the tighter the chain, and vice versa. Its initial value depends on the length and position of the initial curve and the total number of points on that curve.

Since finding the best-fitting parameters depends on many factors, it is not possible to determine and fix them for every purpose beforehand. In this work, we followed two different approaches to estimate the best parameter values for a given set of map points.

In the first approach, we sample the parameter space uniformly and search for parameter sets that minimize the error. For example, in snake fitting, since there are four parameters to be estimated, the parameter space is a 4-D hypercube. We divide this hypercube evenly into a sufficient number of smaller-sized hypercubes and use the parameter values that correspond to the center of each hypercube. In other words, each parameter is uniformly incremented and all possible combinations of the selected parameter values are considered.

We used particle swarm optimization (PSO) as the second approach [38]. This method is inspired by the idea of swarms in nature in order to find the minimum (or, without loss of generality, maximum) value of an objective function. Here, we outline the method as implemented in this study. For a more detailed overview of the original form of the algorithm and its many variants, we refer the reader to [39].

The swarm is composed of a number of particles, each of which is a candidate for a minimum. At an arbitrary iteration n of the algorithm, each particle in the parameter space has a position $\mathbf{x}_n$, a velocity $\dot{\mathbf{x}}_n$, a "previous best" value $pBest_n$, and the corresponding previous best position $\mathbf{p}_n$. The best value would be the lowest (highest) value of the objective function achieved by the particle so far, for a minimum (maximum). We denote the value of the objective function at a position $\mathbf{x}$ by $J(\mathbf{x})$. Furthermore, the minimum (maximum) of $pBest_n$ among all particles is denoted as the global best, $gBest_n$, and the corresponding position as $\mathbf{g}_n$. The velocity and position of each particle is updated according to the

equations:

$$\dot{\mathbf{x}}_{n+1} = w_n \dot{\mathbf{x}}_n + m_1 \varphi_1 (\mathbf{p}_n - \mathbf{x}_n) + m_2 \varphi_2 (\mathbf{g}_n - \mathbf{x}_n) \quad (2.14)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \dot{\mathbf{x}}_{n+1} \quad (2.15)$$

where w_n is called the inertia weight, m_1 and m_2 are constants, and φ_1 and φ_2 are independent uniform random variables in the interval $[0, 1]$. In this study, we take $m_1 = m_2 = 2$, as suggested in [38]. The inertia weight w_n serves as a memory coefficient and prevents the particles from taking "sharp turns" during the iterations, resulting in a smoother movement of the particles.

At each step of the algorithm, the values $pBest_n$ and $gBest_n$ are checked to see if they should be updated. In searching for a minimum, if the current value $J(\mathbf{x}_n)$ of a particle is less than its $pBest_n$ value, $pBest_n$ is set to the current value and $\mathbf{p}_n$ is set to its current position $\mathbf{x}_n$. Similarly, if the current value $J(\mathbf{x}_n)$ of any particle is less than the $gBest_n$ value, $gBest_n$ and $\mathbf{g}_n$ are updated accordingly.

2.3 Experiments and Results

The experimental data had been acquired before this thesis work started, using the front three ultrasonic sensors and structured-light system of the Nomad 200 robot. A simple rule-based wall-following algorithm was used for the indoor environment shown in Figures 2.2 and 2.3(a). The environment is a small room—approximately 2.75 m square—with four corners and an edge feature. There is a cater-corner opening in the lower left corner of the room from which no ultrasonic or structured-light data were received. Referring to Figure 2.3, the environment is comprised of smooth wooden (top and left) and painted (right) walls, and a window shade with vertical slats of 15 cm width (bottom). Some of the corners of the room are not perfect (e.g., where the shade and the right wall make a corner).

(a)

(b)

Figure 2.2: Views of the environment in Figure 2.3(a): (a) looking towards the right, showing the top, right, and bottom walls; (b) looking towards the lower right corner, showing the right and bottom walls in Figure 2.3(a). The cylinder is an additional feature.

To demonstrate our methodology, we used the mapping results of the different UAM processing techniques listed in Table 2.1 and described in detail in [19]. Each of these techniques results in a different set of map points, to which both a snake curve and an SOM are fitted in 2-D.

Table 2.2: Best parameter values found by (i) uniform sampling of the parameter space, and (ii) initializing PSO with the best parameters of (i).

	snake curve fitting				SOM			
method	α	β	γ	κ	$\lambda(0)$	$\sigma(0)$	k_λ	k_σ
(i) uniform sampling	4.20	0.60	0.60	1.80	0.03	2.59	0.83	0.76
(ii) PSO (initialized by (i))	4.58	1.23	0.98	1.72	0.02	2.85	0.90	0.76

The parameters of snake curve and SOM fitting are determined using two methods described in Section 2.2.4. After uniform sampling of the parameter space, the parameter values that resulted in the minimum error for both snake curve and SOM approaches are presented in the first line of Table 2.2. The objective function to be minimized is selected as the average error between the fitted curves and the absolute reference for the map, obtained by a very accurate laser system. For the PSO algorithm, we initialized the parameter values in two different ways. First, random initialization is considered: For the active contour method, all four parameters of each of the 20 particles of PSO are initialized randomly in the interval $[0, 12]$. For the SOM method, $\lambda(0)$ parameter is initialized randomly in $[0, 0.5]$, k_λ and k_σ are initialized randomly in $[0, 1]$, and $\sigma(0)$ is initialized randomly in $[0, 20]$. However, random initialization did not result in the smallest error values. As another alternative, we used the best parameter values found by uniform sampling and added white Gaussian noise to them to initialize each particle of PSO. For the active contour method, all parameters were added white Gaussian noise with standard deviation 0.2. For SOM, the standard deviations of the Gaussian noise added to the four parameters were 0.01, 0.4, 0.02, and 0.02, respectively. This approach reduced the errors considerably, as well as improving the convergence speed of PSO. The parameter values that resulted in the minimum error for active contours and SOM are presented in the second line of Table 2.2.

In the following, let each set of processed UAM data points be denoted as M_k, where k corresponds to one of the UAM processing techniques indexed in Table 2.1. For compatibility, let the set of original laser data points be denoted as M_0. Then, for the laser map ($k = 0$) and for the kth ultrasonic map ($k = 1, \ldots, 8$), the potential function used in fitting the kth snake is selected as

$$U_k(\mathbf{p}) = D_{M_k}(\mathbf{p}) \qquad k = 0, 1, \ldots, 8 \tag{2.16}$$

for all points $\mathbf{p}$. Here, $D_{M_0}(\mathbf{p})$ denotes an element of the Euclidean distance map with respect to the original laser data and $D_{M_k}(\mathbf{p})$ denotes an element of the distance map with respect to the kth processed UAM. Note that the value of the potential function is zero for those points on the image corresponding to the extracted map, and increases gradually with increasing distance of the point $\mathbf{p}$ from the map points.

The set of curve points fitted to the map points resulting from the kth UAM processing technique is referred to as S_k where $k = 1, \ldots, 8$. That is, the kth curve is represented as a collection of points $\mathbf{p}_{ik}, i = 1, \ldots, N_k$, where N_k is the total number of points on curve S_k. The set of points of the curve fitted to the original laser data will be referred to as S_0.

The map of the environment acquired with a structured-light laser system is shown in Figure 2.3(b). This is the original laser data, which is quite accurate, and it is used as the absolute reference. The corresponding Euclidean distance map is shown in Figure 2.3(c), and is drawn by rescaling the values of the potential function to be between zero and 255. In the distance map, the darkest points of value zero correspond to a distance of zero and, after normalization, the lightest points correspond to the value 255. The snake fitted to this laser data is shown in Figure 2.3(d), superimposed on the data points. Figure 2.3(e) shows the SOM fitted to the same laser data. These will be used for comparison in Sections 2.3.1 and 2.3.2. Values of some quantities used in the experiments are provided in Table 2.3.

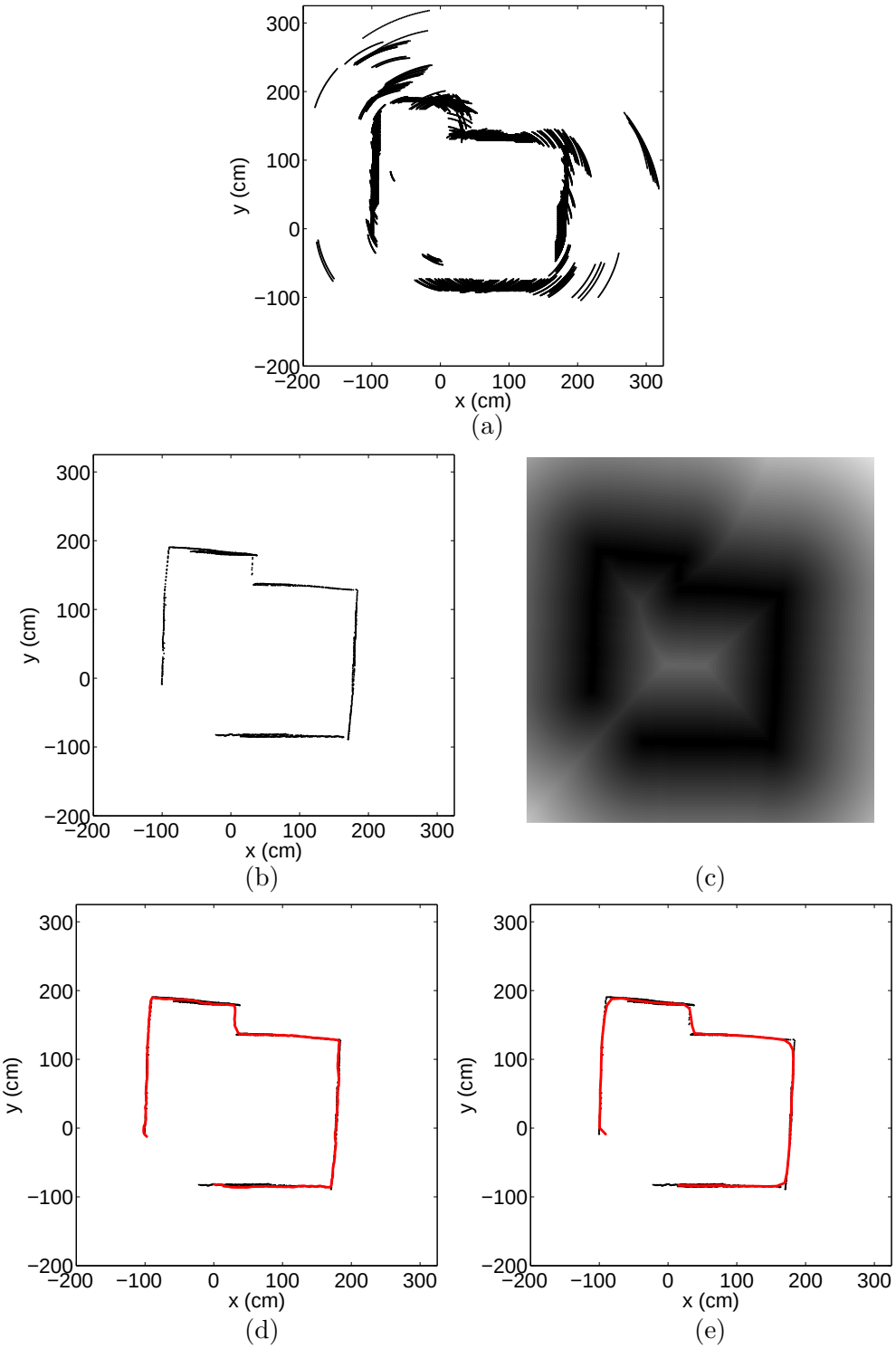


Figure 2.3: (a) The raw UAM, (b) original laser map, (c) distance map with respect to the laser data, (d) snake fitted to the laser data, (e) SOM fitted to the laser data.

Table 2.3: Values of some experimental quantities.

map size	525×525
initial curve center	$(30, 55)$
initial curve radius	185
no. of snake points	400–500
no. of snake iterations	250
intensity range of distance map	0–255
no. of SOM epochs	20
no. of SOM input neurons	2
no. of SOM output neurons	160
no. of PSO particles	20
no. of PSO iterations	20
no. of samples of parameter space	1,296

2.3.1 Active Snake Contours

The snake is initialized as a circle whose center is at $(30, 55)$ with a radius of 185 units to encompass the room boundary (Figure 2.3). Then, the snake is evolved for a fixed number of iterations (250). After each iteration, the points on the snake are checked for uniformity. The distance between any two neighboring points is maintained between two and four units, determined experimentally. That is, after each iteration, the points are deleted or created as required by this constraint. We allow the snake to converge to outlier points caused by cross-talk, multiple, and higher-order reflections to provide a fair evaluation of the different techniques.

Using the two-sided error criterion defined in Equation (2.2), we define the error of the fit at iteration n as follows:

$$\mathcal{E}_{(S_k(n)-M_k)}(n) = \frac{1}{2}\left(\frac{1}{N_k(n)}\sum_{i=1}^{N_k(n)} D_{M_k}\left(\mathbf{p}_{ik}(n)\right) + \frac{1}{L_k}\sum_{j=1}^{L_k} D_{S_k(n)}\left(\mathbf{q}_{jk}\right)\right) \quad (2.17)$$

for $k = 0, \ldots, 8$. Here, $n = 1, \ldots, 250$ is the iteration step, $\mathbf{p}_{ik}(n)$ is the position vector of the ith point on snake k at iteration n, $\mathbf{q}_{jk}$ is the position vector of the jth point of M_k, $N_k(n)$ is the number of points on snake k at step n, and L_k is the number of points of M_k. Note that if the snake fits perfectly to the extracted map points at any iteration (in other words, if every snake point corresponds to

a map point and vice versa), this error becomes zero, since each of the summed distance values in Equation (2.17) would be zero. We calculate and store this error for each iteration. Then, the snake curve that results in the minimum error is determined and selected as the snake that best represents the corresponding map points.

In [4], we used a fixed number of iterations and selected the snake that results in the minimum error. Our observations reveal that the error decreases to a certain value between iterations 100 and 150 and then oscillates around that value, which does not affect the results significantly. In a practical application, it is also possible to take the error at the end of a fixed number of iterations or to set an error threshold to stop the iterations when the error goes below the threshold [2].

To evaluate map accuracy based on the generic error criterion given in Equation (2.2), we chose the point sets P and Q in three different ways:

In the first, we compare the snake fitted to processed UAM points with the originally acquired laser data points so that the two point sets are S_k and M_0. Using the notation at the beginning of this section, the minimum distance of point i on snake S_k to the set M_0 is given by $D_{M_0}(\mathbf{p}_{ik})$. Then, the error is given as:

$$\mathcal{E}_{(S_k - M_0)} = \frac{1}{2}\left(\frac{1}{N_k}\sum_{i=1}^{N_k} D_{M_0}\left(\mathbf{p}_{ik}\right) + \frac{1}{L_0}\sum_{j=1}^{L_0} D_{S_k}\left(\mathbf{q}_{jk}\right)\right) \qquad k = 0, \ldots, 8 \quad (2.18)$$

For our example, when $k = 0$, $\mathcal{E}_{(S_0 - M_0)} = 1.07$, indicating that the average error of the laser snake fit to the original laser data is about one pixel. The errors for the other k values are tabulated in Tables 2.4 and 2.5.

In the second criterion, we compare the processed UAM snake S_k with the snake curve S_0 fitted to the original laser data (Figure 2.3(d)). We calculate the distance of every point on the snake S_k to the nearest point on the snake S_0 and

Table 2.4: Error values (in pixels) for snake curve fitting and SOM using parameters found by uniform sampling of the parameter space.

		snake curve fitting			SOM		
k	method	$\mathcal{E}_{(S_k-M_0)}$	$\mathcal{E}_{(S_k-S_0)}$	$\mathcal{E}_{(S_k-M_k)}$	$\mathcal{E}_{(S_k-M_0)}$	$\mathcal{E}_{(S_k-S_0)}$	$\mathcal{E}_{(S_k-M_k)}$
1	PM	2.71	2.29	3.77	6.06	7.12	4.67
2	VT	2.81	2.51	1.87	2.91	4.13	2.25
3	DM	2.69	2.63	1.98	2.86	3.77	2.54
4	MP	4.82	5.14	2.95	7.09	8.17	2.69
5	BU	5.89	5.35	4.28	9.32	10.38	2.89
6	ATM-org	2.97	2.58	3.07	5.81	6.69	3.73
7	ATM-mod	3.11	3.02	2.70	4.21	5.56	2.79
8	TBF	4.00	4.63	4.62	5.22	6.59	4.60

Table 2.5: Error values (in pixels) for snake curve fitting and SOM using PSO parameters.

		snake curve fitting			SOM		
k	method	$\mathcal{E}_{(S_k-M_0)}$	$\mathcal{E}_{(S_k-S_0)}$	$\mathcal{E}_{(S_k-M_k)}$	$\mathcal{E}_{(S_k-M_0)}$	$\mathcal{E}_{(S_k-S_0)}$	$\mathcal{E}_{(S_k-M_k)}$
1	PM	2.58	2.13	4.01	5.46	6.02	4.02
2	VT	2.78	2.45	1.91	2.85	3.65	2.18
3	DM	2.74	2.44	1.57	2.65	3.11	2.24
4	MP	5.02	5.39	2.90	6.88	7.46	2.63
5	BU	5.92	5.38	4.34	9.10	9.50	2.53
6	ATM-org	2.79	2.36	2.97	5.37	6.07	3.23
7	ATM-mod	2.97	2.91	2.39	3.99	4.79	2.54
8	TBF	4.28	4.93	4.68	4.81	5.67	4.21

average these distances, and vice versa:

$$\mathcal{E}_{(S_k-S_0)} = \frac{1}{2}\left(\frac{1}{N_k}\sum_{i=1}^{N_k} D_{S_0}(\mathbf{p}_{ik}) + \frac{1}{N_0}\sum_{j=1}^{N_0} D_{S_k}(\mathbf{q}_{jk})\right) \tag{2.19}$$

$$\approx \frac{1}{2}\left(\frac{\int_{S_k} D_{S_0}(\mathbf{p})\,ds}{\int_{S_k} ds} + \frac{\int_{S_0} D_{S_k}(\mathbf{q})\,ds}{\int_{S_0} ds}\right) \qquad k = 0,\ldots,8 \tag{2.20}$$

Note that $\mathcal{E}_{(S_0-S_0)} = 0$ by definition. The summations in Equation (2.19) are, in fact, discrete approximations of the line integrals of the distance map functions given in Equation (2.20).

The third way measures how well the kth snake fits to the ultrasonic map

points of the kth technique, and is not related to the reference laser data. It corresponds to the minimum error that is obtained during the snake iterations using Equation (2.17), and is given by:

$$\mathcal{E}_{(S_k-M_k)} = \frac{1}{2}\left(\frac{1}{N_k}\sum_{i=1}^{N_k} D_{M_k}\left(\mathbf{p}_{ik}\right) + \frac{1}{L_k}\sum_{j=1}^{L_k} D_{S_k}\left(\mathbf{q}_{jk}\right)\right) \qquad k = 1, \ldots, 8 \quad (2.21)$$

The errors of snake fitting for the different UAM processing techniques are tabulated in the third, fourth, and fifth columns of Tables 2.4 and 2.5. The results in Table 2.4 are obtained by using the best-fitting parameter values obtained by uniform sampling of the parameter space (first line of Table 2.2). Similarly, the parameter values found by initializing PSO with the uniform sampling parameters (second line of Table 2.2) are used to obtain the results in Table 2.5. In both tables, it can be observed that $\mathcal{E}_{(S_k-M_0)}$ and $\mathcal{E}_{(S_k-S_0)}$ values are mostly comparable with each other since both take the laser data as reference, in original and snake-fitted forms, respectively. According to the results, PM, DM, and VT techniques have the smallest errors, and MP and BU perform the worst because these latter processing techniques result in more spurious points on the extracted map. The remaining techniques are comparable with each other. It can also be observed that PM and ATM-org are the only methods for which the $\mathcal{E}_{(S_k-M_k)}$ value is larger than $\mathcal{E}_{(S_k-M_0)}$ and $\mathcal{E}_{(S_k-S_0)}$, indicating that the snake curves for these methods fit better to the laser data than the processed UAMs. This is because unlike most of the other methods, many outlying points remain after processing the UAM with these methods; the snake fits poorly to these outliers and this causes the error $\mathcal{E}_{(S_k-M_k)}$ to increase.

Because smaller errors are obtained by uniform sampling of the parameter space followed by PSO, the best-fitting parameter values found with this approach are used for the given illustrations. The snake curves fitted to the processed UAMs and the laser data are illustrated in Figure 2.4. In the different parts of the figure, the black features correspond to map points obtained with a particular UAM processing technique. The blue (thick) curves are the snakes fitted to these map points. The red (thin) curve is the snake fitted to the laser data, which is the same in each part of the figure and is included as a reference for visual comparison. This is also the same curve as in Figure 2.3(d).

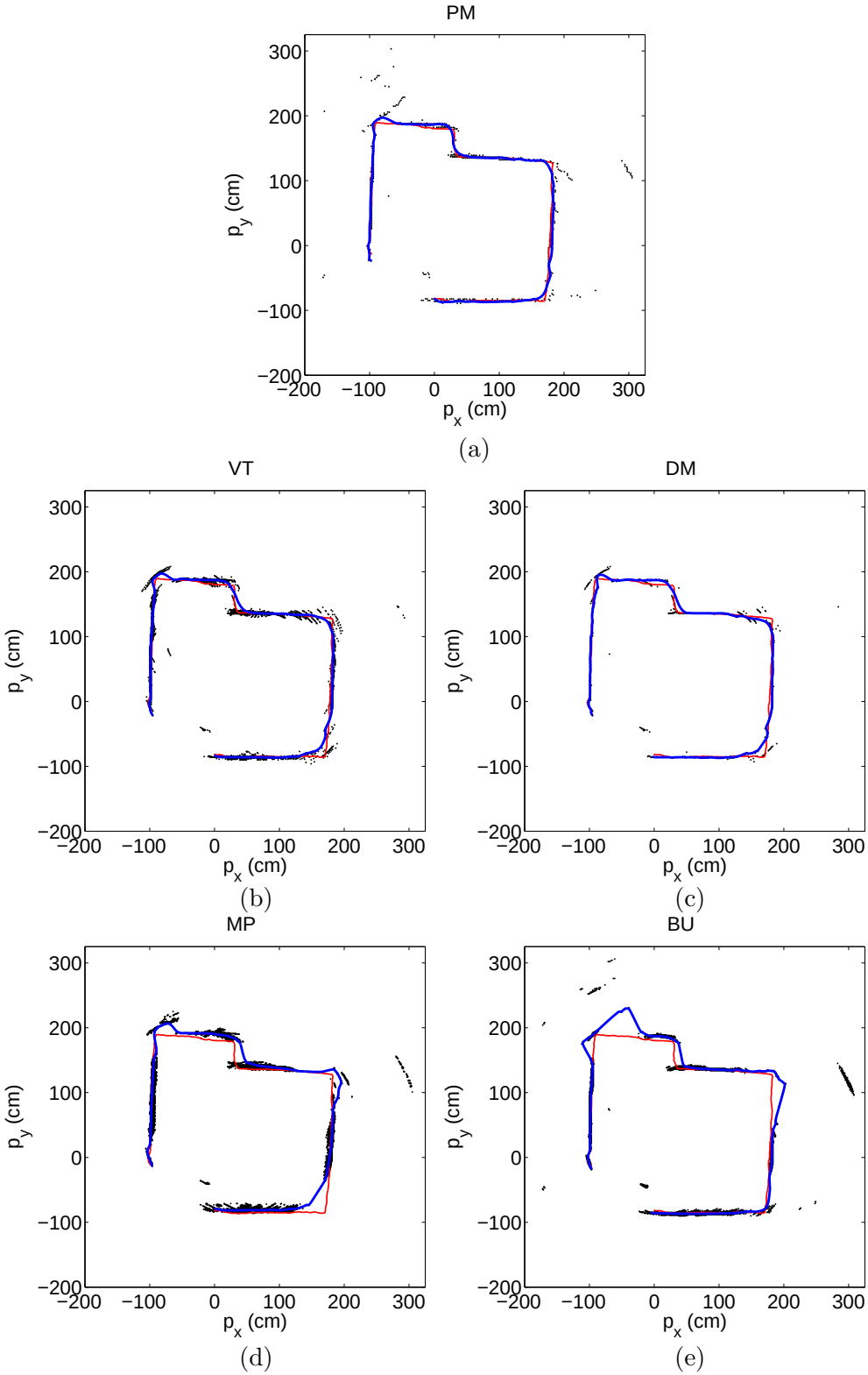
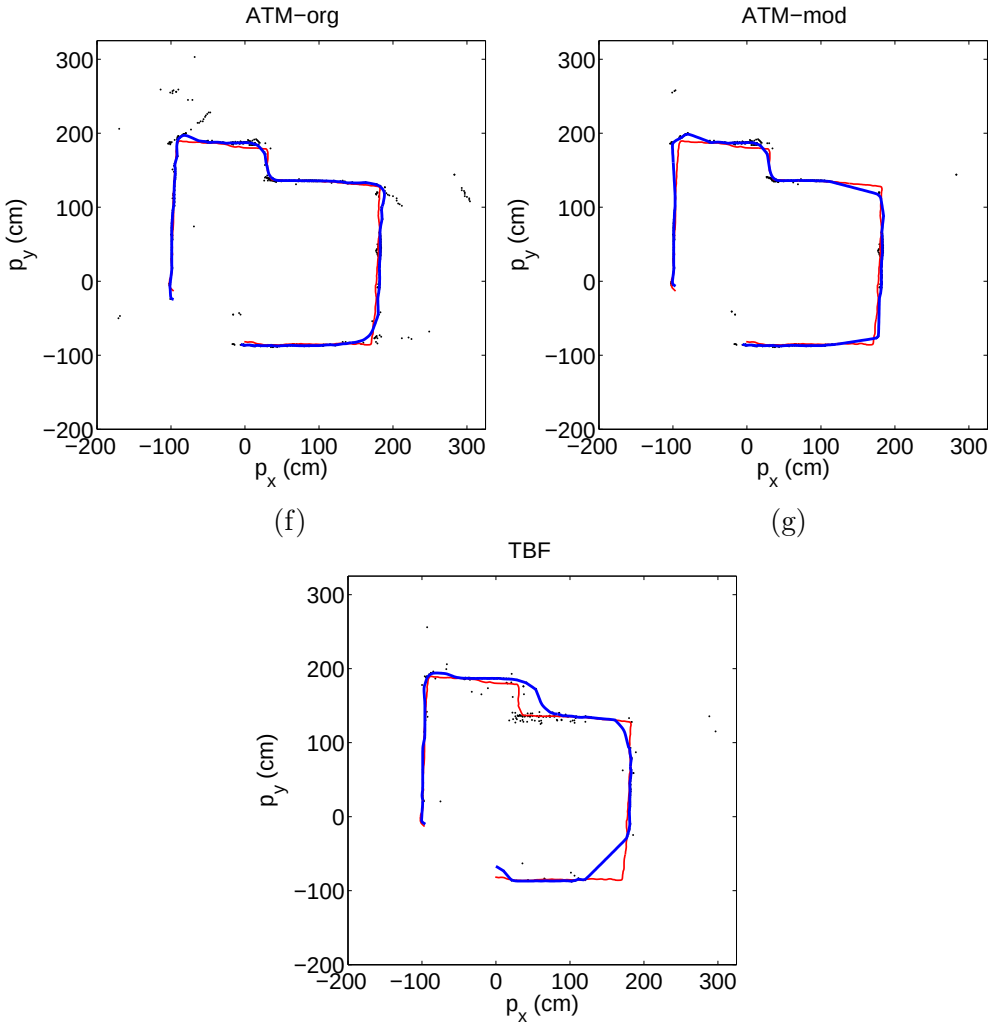


Figure 2.4: Results of snake fittings for (a) PM, (b) VT, (c) DM, (d) MP, (e) BU (continued).



(f) (g)

(h)

Figure 2.4: (continued) (f) ATM-org, (g) ATM-mod, and (h) TBF.

Note that in this study, the snake curves are initialized outside the boundaries of the room because the curve tends to shrink rather than expand due to the first derivative term in the energy expression. This fact should be taken into account in determining the initial location of the snake curve. Initializing the snake within the boundaries of the environment is also a possibility, as the spurious points outside the boundaries would not affect the snake curve as much, allowing it to follow the boundaries of the room more closely. However, inside initialization would not result in a fair comparison between the techniques in terms of the amount of spurious points left after UAM processing. In addition, in some mapping applications one may not be completely free to choose the initial location. For example, in a room with many obstacles close to the room boundaries,

it would be essential to initialize the snake curve outside in order to represent the boundaries of the room correctly. However, if detecting obstacle boundaries is more important, one would initialize the snake inside the boundaries. In fact, some applications may require both. The choice for the initial location should depend on the configuration of obstacles and the free space.

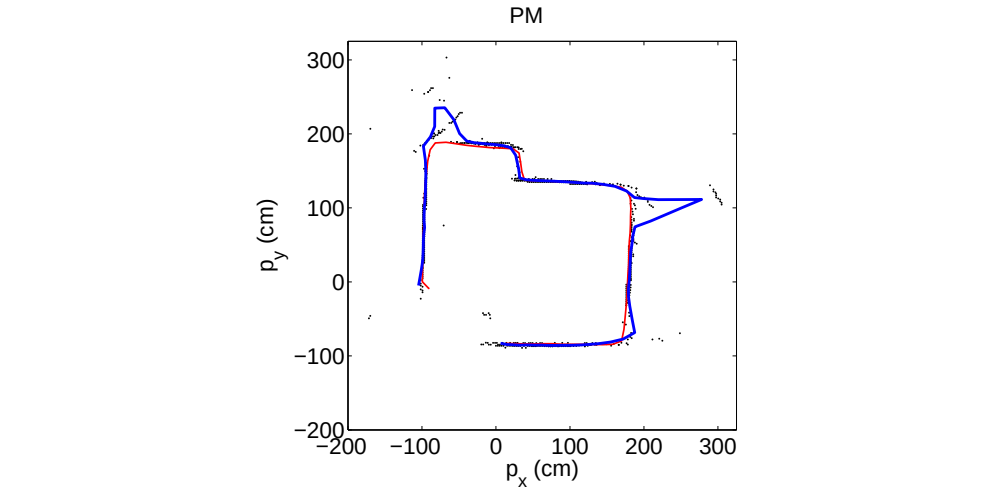
2.3.2 Kohonen's Self-Organizing Maps

We initialize an SOM with 160 neurons as a circle outside the boundaries of the room, with the same center and the same radius as in the snake curve fitting procedure. The iterations are stopped after 20 epochs. Similar to the case with the snake curve, the distance between neighboring points on the curve is maintained between 10 and 16 units. An SOM fitted to the laser data using the parameters of uniform sampling can be seen in Figure 2.3(e), superimposed on the data points. The curves fitted to the processed UAMs can be seen in Figure 2.5.

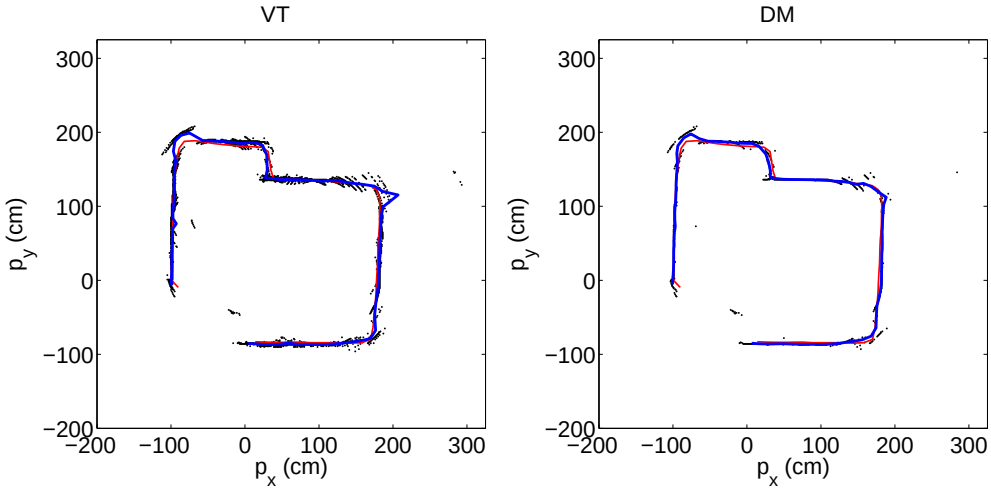
The previously defined error criterion is employed for this method as well, using the curves fitted by the SOM instead of snakes. The results of the SOM method are given in the sixth, seventh, and eighth columns of Tables 2.4 and 2.5. With a few exceptions, SOM errors are larger than snake errors. The DM and VT methods demonstrate the best performance. In Figure 2.5(f), it can also be observed that the SOM curve fitted to the map obtained by the ATM-org technique is highly affected by the outlier points, unlike the snake curve fitted for that technique (Figure 2.4(f)). Curves generated by fitting an SOM are not constrained by length or curvature as the snake curves are, thus they are more likely to fit to the outlier points. This results in larger error values in general.

2.4 Discussion

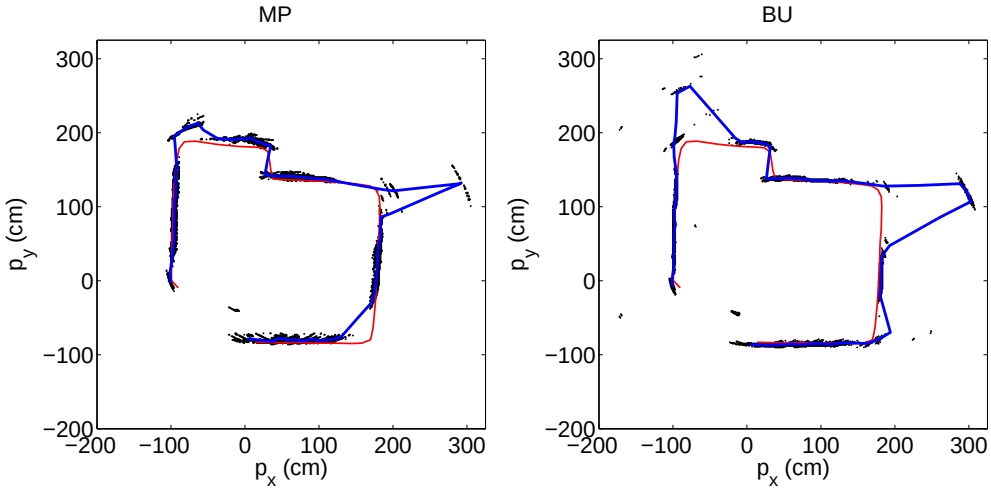
Looking at the error values in Tables 2.4 and 2.5 and observing Figures 2.4 and 2.5, it can be concluded that DM, VT, and ATM-mod methods eliminate



(a)

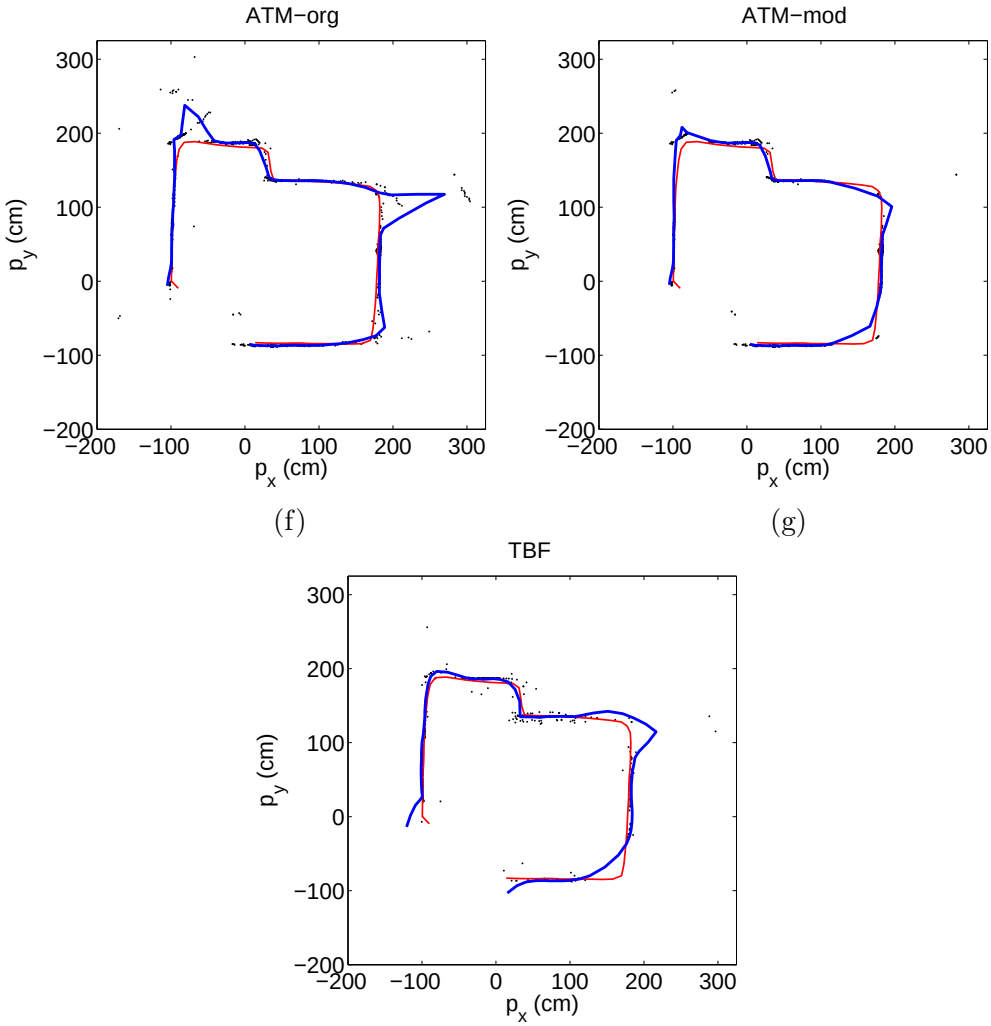


(b) (c)



(d) (e)

Figure 2.5: Results of SOMs for (a) PM, (b) VT, (c) DM, (d) MP, (e) BU (continued).



(f) (g) (h)

Figure 2.5: (continued) (f) ATM-org, (g) ATM-mod, and (h) TBF.

most of the artifacts in the ultrasonic data resulting from multiple and higher-order reflections, cross-talk, and erroneous measurements. The PM, MP, BU, and ATM-org methods cannot eliminate those artifacts as much, resulting in larger errors. This can be observed more clearly in Figure 2.5.

In general, if a UAM processing technique cannot eliminate artifacts well, the resulting errors are larger. DM, VT, and ATM-mod are superior to the other techniques in eliminating artifacts, therefore they result in smaller errors. In fact, DM can be considered an improved version of VT, where directional processing of the map points is incorporated in the algorithm [19]. The modified ATM is also quite accurate and eliminates the artifacts better than the original ATM. In the

UAM, if there are arcs with no intersections, these are removed with the modified ATM but not with the original ATM. Generally speaking, the ATM technique creates accurate yet sparsely filled maps. ATM is found to require a denser UAM to begin with in order to produce a map with approximately the same number of points as the other techniques.

The PM technique reduces each arc to a single point mark in the middle of the arc. ATM-org places a more accurate point mark on arcs with transversal intersections (except those with two and none), reducing many of the arcs to single points also. It treats arcs with no intersections in the same way as PM. For this reason, the number of extracted map points from these two techniques is quite similar.

The errors with TBF are usually larger than ATM-org and ATM-mod errors. The TBF obtains the fewest number of map points among all the techniques compared and results in more gaps in the resulting map. This result is, however, expected because apart from the fact that fewer arcs are used at a given time to begin with (due to the sliding window), TBF eliminates arcs with no meaningful and accurate correspondence. In addition, planar wall locations found with this method are not very accurate. This is also observed in Figures 15 and 16 of [40] as many outlying points extracted by the algorithm. A major advantage of TBF is that it is very fast and takes about the same time as the simplest PM method [19] because it does not divide the environment into grids but processes the information geometrically.

Among the eight approaches considered, DM produces relatively low errors and the associated computation time is small (Table 2.6). Considering that DM also has high range accuracy and is superior in eliminating the artifacts and outliers of the UAM, it can be considered as one of the best methods in terms of the overall performance.

In [41, 42, 43], VT and MP were investigated in detail based on simulations and experimental studies for different transducer configurations (linear, circular, random), different beamwidths (5° to 105°), different surface curvatures, roughness, distance, and different noise levels on time-of-flight measurements. The best

Table 2.6: Computation times for fitting snake curves and SOM for a given parameter set.

method	number of map points	computation time (s)			
		distance map	snake fitting	overall	SOM
PM	697	5.6	69.0	75	44
VT	2634	20.3	74.6	95	168
DM	863	7.1	78.2	85	55
MP	4692	35.1	110.6	146	299
BU	2994	21.9	105.9	128	191
ATM-org	920	5.8	66.6	72	59
ATM-mod	788	5.0	78.9	84	50
TBF	387	3.5	57.1	61	25

results were obtained with a random configuration of transducers, followed by circular and linear ones. For both methods, the errors were shown to increase with increasing beamwidth, increasing surface distance, curvature, and roughness. Although such detailed studies for the other methods have not been performed, we expect similar results for the remaining techniques because varying these parameters primarily affects the quality of the information inherent in the ultrasonic arc map. This also leads us to expect that for a given choice of these parameters, the results of comparing the methods will not be altered significantly.

The active contour and SOM methods used for map representation differ in many aspects. Snake curves minimize an energy function that is the sum of their internal energy (basically length and curvature) and a potential function that is at minimum on the map points. As illustrated in our example, snake curves are more robust to outliers than SOMs (compare Figures 2.4(f) and 2.5(f)). SOMs, on the other hand, are not constrained by length or curvature and try to adjust the curve to encompass all data points. For this reason, SOMs are more likely to fit to outlying map points and may not converge to the actual borders of the environment, resulting in larger errors in general. This is the case when larger values of k_λ and k_σ are used with the effect that the decay is slower. Experimenting with different k_λ and k_σ values, we have observed that decreasing $\lambda(n)$ and $\sigma(n)$ more rapidly causes the SOM to be more robust to the outlier

points of the ultrasonic map. When smaller values of k_λ and k_σ are used, it is more likely that the SOM will fit to the topology of the data rather than the outliers and the resulting errors will be smaller. However, even in this case, active snake contours are still superior in terms of the resulting errors.

The evolution equations (2.10) and (2.11) are used to update the position of the snake. The parameters involved in these equations control the final shape of the snake such as determining its length and curvature. Selecting the best parameters is less critical for the convergence of snakes than for SOMs. For snakes, there may be more than one parameter set that fit the given data points well. However, the convergence of snakes is more sensitive to the initialization of the curve than are SOMs, whereas the convergence of SOMs is sensitive to the order in which the map points are input to the neural network. In our implementation of the SOM, the points were input randomly.

The running times of active contours and SOMs are tabulated in Table 2.6 for each of the UAM processing techniques, together with the number of data points in the corresponding processed UAM. These running times are obtained using MATLAB on a computer with a 1.80 GHz dual core processor. It can be observed that the running times of both methods increase with increasing data size, however, the running time of the SOM is more dependent on data size. The computation time of the SOM increases roughly linearly with the number of data points, with an average processing time of 64 ms per map data point. In the active snake contour method, the data size mainly affects the time for calculating the distance map. On average, forming the distance map takes 7.6 ms per map data point. Once the distance map is formed, the running time of the snake fitting algorithm is mainly determined by the number of points comprising the snake curve. For our example, the average running time is about 80 sec. The overall average running time is about 93 sec. The slight increase in the snake fitting time for the BU and MP techniques is because the corresponding snake curves are longer, resulting in more points to process at each iteration (Figure 2.4). Furthermore, the overall running time and accuracy of the snake fitting technique change depending on the resolution of the image that represents the map. More

generally, the computation times for both methods depend on the selected parameters because these parameters control the convergence characteristics of the curves. Even though a fixed number of iterations is used in this study, parameters can be tuned for a specific application and the computation times can be reduced by using an application-specific number of iterations. However, regardless of a fixed or variable number of iterations, the results given in Table 2.6 can be viewed as an example of what to expect for processing times in an application in terms of the map size and the curve-fitting method used.

Part II

Inertial Sensing

Chapter 3

Characterization and Modeling of Inertial and Magnetic Sensor Errors

Inertial sensors are self-contained, nonradiating, nonjammable, dead-reckoning devices that provide dynamic motion information through direct measurements. Gyroscopes provide angular rate information around an axis of sensitivity, whereas accelerometers provide linear or angular velocity rate information.

For several decades, inertial sensors have been used for navigation of aircraft [44, 45], ships, land vehicles, and robots [46, 47, 48], for state estimation and dynamic modeling of legged robots [49, 50], for shock and vibration analysis in the automotive industry, and in telesurgery [51, 52]. Recently, the size, weight, and cost of commercially available inertial sensors have decreased considerably with the rapid development of micro electro-mechanical systems (MEMS) [53].

In this section, we analyze the error characteristics of the MEMS inertial and magnetic sensor units used in our experiments. We use MTx sensors (Figure 3.1) by Xsens Technologies B. V. [54].

3.1 Overview and Description of MTx Sensors

These sensors are referred to as miniature inertial three degrees-of-freedom (3-DOF) orientation tracker by the manufacturer, and their primary purpose is providing drift-free 3-D orientation data especially for human body segments. The sensor unit includes a tri-axial accelerometer, a tri-axial gyroscope, and a tri-axial magnetometer. In order to estimate the orientation, gyroscope outputs are fused with magnetometer and acceleration measurements in a Kalman filter. Furthermore, the sensors also provide kinematic data by outputting the 3-D acceleration, 3-D rate of turn, and 3-D Earth-magnetic field measurements.

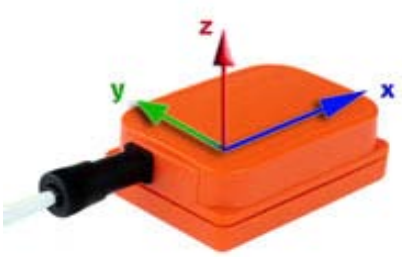


Figure 3.1: MTx 3-DOF orientation tracker
(reprinted from http://www.xsens.com/en/general/mtx).

The manufacturer provides standard sensor units as well as customized units. In the standard unit, the gyroscopes measure angular velocities in the range $\pm 1200^{\circ}/s$, the accelerometers measure acceleration in the range $\pm 5g$, and the magnetometers measure magnetic field in the range $\pm 75\mu$T. In our studies, in addition to the standard units, we also use customized units whose accelerometer ranges are modified as $\pm 18g$. Each unit also includes a temperature sensor that measures the internal temperature of the sensor unit. The A/D resolution is 16 bits for accelerometers, gyroscopes, and magnetometers, and 12 bits for the temperature sensor, both for the standard unit and the customized unit. Therefore, the acceleration accuracy is better in the standard unit than in the customized unit. However, accelerations more than $\pm 5g$ can be encountered in some body motions, which necessitates the use of a sensor unit that can measure large accelerations.

Usually, inertial sensors have an error on the output. This error can be characterized as a combination of various error components, the most dominant of which is a bias that varies according to the internal temperature of the sensor. To compensate for these errors, the manufacturer provides two output modes for the kinematic data output. In the uncalibrated mode, the raw data are directly provided by the internal sensors. In the calibrated mode, the sensor model developed by the manufacturer is also integrated in the output. In the technical manual [54], the basic model for the calibrated mode is given as

$$\mathbf{s} = \mathbf{K}_T^{-1}(\mathbf{r} - \mathbf{b}) \tag{3.1}$$

where $\mathbf{s}$ is the sensor output, $\mathbf{r}$ is the 16-bit A/D converter reading, $\mathbf{b}$ is the bias vector, and $\mathbf{K}_T$ is a matrix given by $\mathbf{K}_T = \mathbf{GM} + \mathbf{O}$. Here, $\mathbf{G}$ is the gain matrix and provides the conversion from the A/D output to the physically measured quantity, $\mathbf{M}$ is the misalignment matrix and compensates for possible misalignments while the sensors are mounted in the housing, and $\mathbf{O}$ represents higher-order models. The manufacturer does not provide information on how the higher-order models are handled.

In this chapter, we provide our error characterization procedure and results on one of the standard units (henceforth referred to as MTx-49A53G25) and one of the customized units (henceforth referred to as MTx-49A83G25). The matrices and the bias vector as given by the manufacturer are presented in Table 3.1 for both units. However, in our experiments, we observed that this calibration by the company is insufficient and further error modeling is necessary. The experiments that we perform are explained in the next section.

As stated before, the primary application area of MTx sensors is orientation measurement. These sensors provide drift-free orientation data in three different user-selectable modes: Euler angles, direction cosine matrix, and quaternions. The built-in Kalman filter fuses gyroscope, acceleration, and magnetic field data to estimate the orientation. The orientation output gives the orientation of the sensor coordinate frame with respect to the local navigation frame, whose x, y, and z axes coincide with the local North, West, and Up directions, respectively.

Table 3.1: Alignment matrices, gain matrices, and bias vectors for the (a) MTx-49A53G25 and (b) MTx-49A83G25 units.

tri-axial gyroscope	tri-axial accelerometer	tri-axial magnetometer
$\mathbf{G} = \begin{pmatrix} 1117 & 0 & 0 \\ 0 & 1132 & 0 \\ 0 & 0 & 1125 \end{pmatrix}$	$\mathbf{G} = \begin{pmatrix} 408.5 & 0 & 0 \\ 0 & 412.4 & 0 \\ 0 & 0 & 411.4 \end{pmatrix}$	$\mathbf{G} = \begin{pmatrix} 7617 & 0 & 0 \\ 0 & 7606 & 0 \\ 0 & 0 & 7604 \end{pmatrix}$
$\mathbf{M} = \begin{pmatrix} 1.00 & -0.01 & 0.01 \\ 0.00 & 1.00 & -0.01 \\ 0.00 & -0.01 & 1.00 \end{pmatrix}$	$\mathbf{M} = \begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ -0.01 & 0.00 & 1.00 \end{pmatrix}$	$\mathbf{M} = \begin{pmatrix} 1.00 & 0.01 & -0.05 \\ -0.01 & 1.00 & 0.03 \\ -0.06 & 0.02 & 1.00 \end{pmatrix}$
$\mathbf{b} = \begin{pmatrix} 32670 \\ 32597 \\ 32835 \end{pmatrix}$	$\mathbf{b} = \begin{pmatrix} 33138 \\ 33085 \\ 32480 \end{pmatrix}$	$\mathbf{b} = \begin{pmatrix} 33860 \\ 32070 \\ 31611 \end{pmatrix}$

(a)

tri-axial gyroscope	tri-axial accelerometer	tri-axial magnetometer
$\mathbf{G} = \begin{pmatrix} 1104 & 0 & 0 \\ 0 & 1116 & 0 \\ 0 & 0 & 1097 \end{pmatrix}$	$\mathbf{G} = \begin{pmatrix} 134.2 & 0 & 0 \\ 0 & 135 & 0 \\ 0 & 0 & 134 \end{pmatrix}$	$\mathbf{G} = \begin{pmatrix} 7417 & 0 & 0 \\ 0 & 7649 & 0 \\ 0 & 0 & 7381 \end{pmatrix}$
$\mathbf{M} = \begin{pmatrix} 1.00 & -0.01 & 0.00 \\ 0.00 & 1.00 & -0.01 \\ 0.00 & 0.00 & 1.00 \end{pmatrix}$	$\mathbf{M} = \begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ -0.01 & -0.01 & 1.00 \end{pmatrix}$	$\mathbf{M} = \begin{pmatrix} 1.00 & 0.00 & -0.05 \\ 0.01 & 1.00 & 0.03 \\ -0.04 & 0.01 & 1.00 \end{pmatrix}$
$\mathbf{b} = \begin{pmatrix} 32573 \\ 33169 \\ 33437 \end{pmatrix}$	$\mathbf{b} = \begin{pmatrix} 33100 \\ 33452 \\ 32632 \end{pmatrix}$	$\mathbf{b} = \begin{pmatrix} 33856 \\ 31961 \\ 31727 \end{pmatrix}$

(b)

3.2 Experiments

In our experiments, the sensors are left on a table for 12 hours and the data are recorded. Ideally, gyroscopes and x- and y-axis accelerometers should give zero output, z-axis accelerometer should output the gravitational acceleration g, and the magnetometers should give out a constant value representing the magnetic field of the Earth in the corresponding direction. However, this is not the case because of many factors. For example, the table may not be perfectly horizontal, i.e., not exactly perpendicular to the local gravity direction. Because of this fact, the effect of gravity is sensed by all three accelerometers. The second most important cause of noise is the temperature. The bias values of accelerometers and gyroscopes change according to the temperature. The MTx sensor unit has a built-in temperature sensor, and temperature data are also recorded for 12

hours. Quantization, stochastic noise, and round-off errors are other factors that affect the output. The data are recorded in the "calibrated output mode" of the MTx unit, for which the manufacturing company reports that errors in alignment and errors due to temperature change are compensated. The data without any processing are presented in Figures 3.2 and 3.3 for the standard and customized units, respectively. Looking at the figures, it can be observed that the built-in temperature compensation is insufficient for the accelerometers (Figure 3.3(g)). That is, there is a time-varying bias that depends on the internal temperature. Although invisible to the naked eye, this is also true for gyroscopes. It can be observed when the gyroscope signal is passed through a moving average filter. The gyroscope signal at the output of a 5-min long moving average filter is plotted in Figure 3.4.

Our experiments reveal that the biases of accelerometers and gyroscopes are time varying. To obtain orientation and position information, gyroscope signals have to be integrated once and the accelerometer signals have to be integrated twice after being transformed to the navigation coordinate frame. Thus, even the smallest bias in the sensor reading will cause unbounded drift in orientation and position. It is a well-known fact that a constant bias in accelerometers causes parabolic error growth in the position. Similarly, a constant bias in the gyroscope signals causes linear error growth in the orientation, and cubic error growth in the position due to integration after coordinate transformations [55]. Thus, the error characteristics of the sensors should be modeled for navigation applications.

3.3 Eliminating Transients

In order to determine the error characteristics, the usual practice is to remove the trends in the sensor data first [56]. In the literature, it is quite common to fit an exponential model given by $b(t) = c_1 e^{-t/T} + c_2$ to remove the transients (trends) in inertial sensor data [46]. For accelerometers, other models have been suggested, such as square root functions ($b(t) = c_1\sqrt{t} + c_2$) and logarithmic functions ($b(t) = c_1 \log(c_2 + c_3 t)$) [58]. In our studies, we fit the exponential model

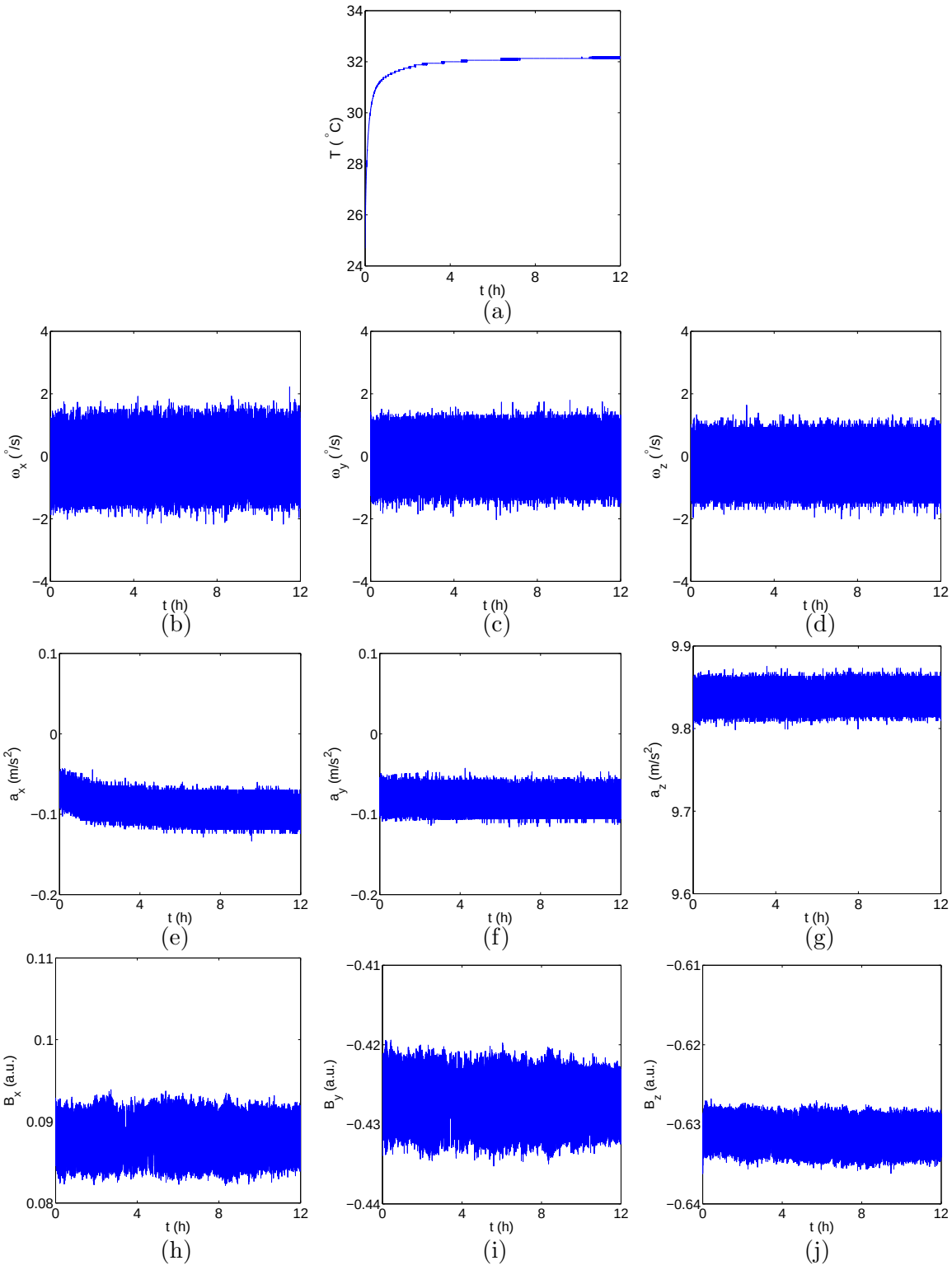


Figure 3.2: Calibrated mode data obtained from the MTx-49A53G25 unit, recorded for 12 hours. (a) Temperature, (b)-(d): x, y, z gyroscopes, (e)-(g): x, y, z accelerometers, (h)-(j): x, y, z magnetometers.

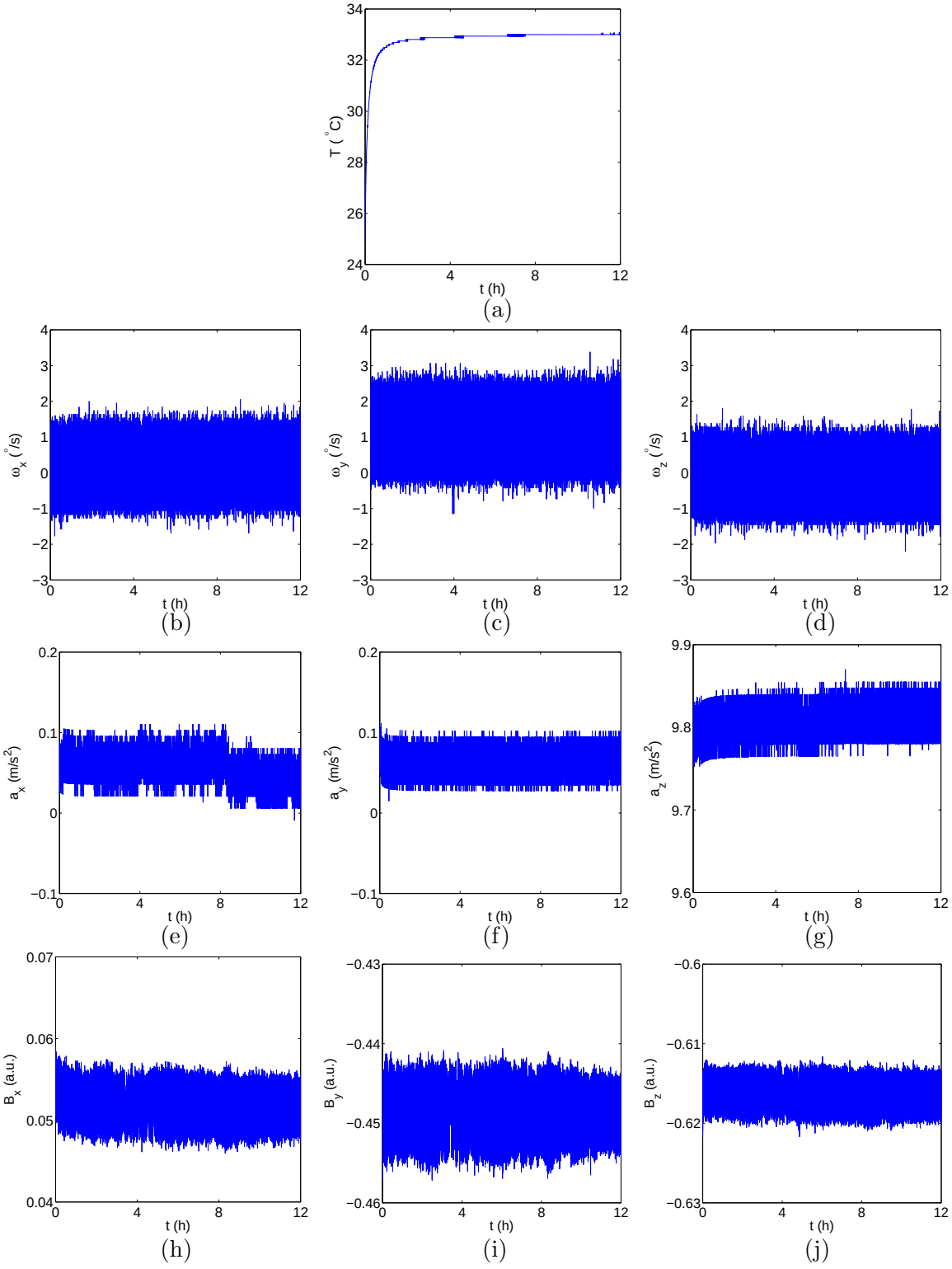


Figure 3.3: Calibrated mode data obtained from MTx-49A83G25 unit, recorded for 12 hours. (a) Temperature, (b)-(d): x, y, z gyroscopes, (e)-(g): x, y, z accelerometers, (h)-(j): x, y, z magnetometers.

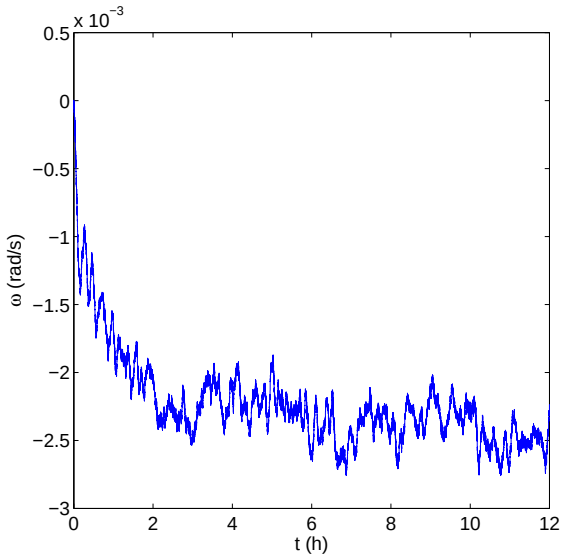


Figure 3.4: Gyroscope signal after applying the moving average filter.

to the gyroscope data. For the accelerometers, we fit the three models mentioned above and observed that the exponential function gives better results than the square root and logarithmic functions. We use the Levenberg-Marquardt algorithm, which is developed for nonlinear curve fitting [59]. Then, the residual signals are tested with the following whiteness test.

We estimate the autocorrelation function using the following unbiased and biased estimators:

$$\hat{R}_{ss}(\Delta) = \frac{1}{N_s - |\Delta|} \sum_{n=0}^{N_s-|\Delta|-1} s[n]s[n+\Delta] \tag{3.2}$$

$$\tilde{R}_{ss}(\Delta) = \frac{1}{N_s} \sum_{n=0}^{N_s-|\Delta|-1} s[n]s[n+\Delta] \tag{3.3}$$

For ideal white noise, these functions should be zero at every point except for $\Delta = 0$. The method in [60] states that the distribution of these estimates around the true value can be approximated by a Gaussian distribution with zero mean

Table 3.2: Fitted parameter values for the MTx-49A53G25 unit.

	c_1	c_2	T
x-gyro	–0.001673	–0.00122	7.957
y-gyro	–0.001008	–0.00141	0.5752
z-gyro	–0.000444	–0.00474	0.3638
x-acc	0.02556	–0.09575	2.637
y-acc	0.003407	–0.0825	6.352
z-acc	–0.7081	10.54	3917

and the following standard error:

$$\hat{\sigma}_{\hat{R}_{ss}}(\Delta) = \frac{\sqrt{N_s}}{N_s - |\Delta|}\hat{R}_{ss}(0), \quad \Delta \neq 0 \tag{3.4}$$

$$\tilde{\sigma}_{\tilde{R}_{ss}}(\Delta) = \frac{1}{\sqrt{N_s}}\tilde{R}_{ss}(0), \quad \Delta \neq 0 \tag{3.5}$$

for unbiased and biased estimates, respectively. Therefore, we can calculate the unbiased and biased estimates and determine whether at least 95.5% of the samples are inside the $\pm 2\sigma$ bounds for both cases.

The exponential model parameters are given in Table 3.2 for the MTx-49A53G25 unit and in Table 3.3 for the MTx-49A83G25 unit. In the tables, c_1 and c_2 are given in rad/s for gyroscopes, and m/s^2 for accelerometers. The time constant T is given in hours.

Table 3.3: Fitted parameter values for the MTx-49A83G25 unit.

	c_1	c_2	T
x-gyro	–0.001774	0.00355	2.154
y-gyro	–0.000712	0.02039	0.4944
z-gyro	0.001413	–0.00238	1.191
x-acc	–	–	–
y-acc	0.01838	0.06364	0.064
z-acc	–0.0224	9.819	7.674

Note that the time constant T for the z-axis accelerometer in Table 3.2 is very large. This practically means that there is no exponential dependence, and only

Table 3.4: Percentages of the samples inside $\pm 2\sigma_{R_{xx}}$ bounds for both MTx units.

	MTx-49A53G25 (%)	MTx-49A83G25 (%)
x-gyro	97.30	95.19
y-gyro	98.42	97.74
z-gyro	98.35	97.44
x-acc	54.02	5.84
y-acc	92.47	66.90
z-acc	74.68	50.84

the mean is subtracted. Also, the solver could not fit an exponential model to the x-axis accelerometer in the MTx-49A83G25 unit. This is also confirmed in Figure 3.3(e), where it can clearly be observed that an exponential model is not suitable for the data. However, this instability of the bias is observed when the operating duration is high, which is considered unlikely to occur in our practical applications. Therefore, we only subtract the mean from this signal in order to find the residual. The results of the residual whiteness test are presented in Table 3.4 for both units. As shown in the table, the residuals are white for the gyroscopes, but correlated noise components exist for the accelerometers.

3.4 Allan Variance Analysis

After the removal of the transients with the exponential model, Allan variances of the residual signals are calculated to identify the error components in the signals. Although originally intended for analyzing oscillators [61], this method is widely used in inertial sensor error modeling [56, 62, 63], in addition to being suggested as a standard modeling procedure by IEEE [57, 58]. Another option is to use approaches based on the power spectral density (PSD) [57]. In the literature, autoregressive process models [64] and expectation-maximization methods [65] have also been used to model inertial sensor errors . It is possible to analyze the error in two separate components, especially for gyroscopes [66]. The first component is a constant or slowly varying bias that is usually called the *bias instability*. The second component is due to high-frequency noise, and is observed as a random

walk in the integrated gyroscope signals. This component is called the *angle random walk* for gyroscopes, and *velocity random walk* for accelerometers. Most studies in the literature are focused on estimating the parameters of these two error components. There are other sources of error such as quantization noise, random walk, and rate ramp [57], whose corresponding parameters are more difficult to estimate [56]. In this section, we define the Allan variance, following a notation similar to [56], and present our results. Basically, in the Allan variance method, the signal is first partitioned into segments of equal length in time. This partitioning is done using various segment lengths. Then, each segment is averaged and the variance is calculated.

Consider the sampled sensor data $\mathbf{s} = \{s_k, k = 0, \ldots, N_s - 1\}$ with a sampling frequency of τ_0. Let $t_k = k\tau_0$ and define an averaging time $\tau = m\tau_0$. Then, filter $\mathbf{s}$ with a moving average filter of width m:

$$\bar{s}_k = \frac{1}{m} \sum_{i=k}^{k+m-1} s_i \tag{3.6}$$

Then, apply a first-order difference filter of width m to get

$$D_k = \frac{\bar{s}_{k+m} - \bar{s}_k}{\tau} \tag{3.7}$$

The Allan variance of $\mathbf{s}$ is defined as half the mean square of D_k:

$$\sigma^2(\tau) = \frac{1}{2\tau^2(N_s - 2m + 1)} \sum_{k=0}^{N_s-2m} (\bar{s}_{k+m} - \bar{s}_k)^2 \tag{3.8}$$

Another estimator can be obtained by decimating D_k by m:

$$\sigma^2(\tau) = \frac{1}{2\tau^2(M - 1)} \sum_{k=0}^{M-1} (\bar{s}_{(k+1)m} - \bar{s}_{km})^2 \tag{3.9}$$

where $M = \lceil \frac{N_s}{m-1} \rceil$. The ceiling function $\lceil \cdot \rceil$ maps its argument to the smallest integer not less than the argument. Both of these estimators are unbiased.

There is a relation between the PSD $S_s(f)$ of a random process and its Allan variance $\sigma^2(\tau)$:

$$\sigma^2(\tau) = 4 \int_0^\infty S_s(f) \frac{\sin^4(\pi f \tau)}{(\pi f \tau)^2} \, df \tag{3.10}$$

The relation between the PSD and the Allan variance is not one-to-one, and, in general, not invertible. Thus, the knowledge of the Allan variance of a random process does not uniquely identify the spectrum. A necessary and sufficient condition for two different spectra to have the same Allan variance is given in [67]. However, as discussed in the same paper, this non-uniqueness of the Allan variance does not cause a problem for most physical systems.

In the literature, the term "Allan variance" frequently refers to the Allan standard deviation, $\sigma(\tau)$, which is the square root of the Allan variance defined above. This is because the Allan variance analysis is usually conducted by plotting $\sigma(\tau)$ versus τ on a log-log plot. Different sources of error appear with different slopes in the Allan variance plot, usually in different regions of τ. Thus, various sources can be easily identified. For example, white noise is dominant for τ values less than one, whereas random walk is dominant for τ values greater than one. A sample Allan variance plot is given in Figure 3.5 [57]. The Allan variance and PSD domain expressions for various contributors to the error are tabulated in Table 3.5 [57].

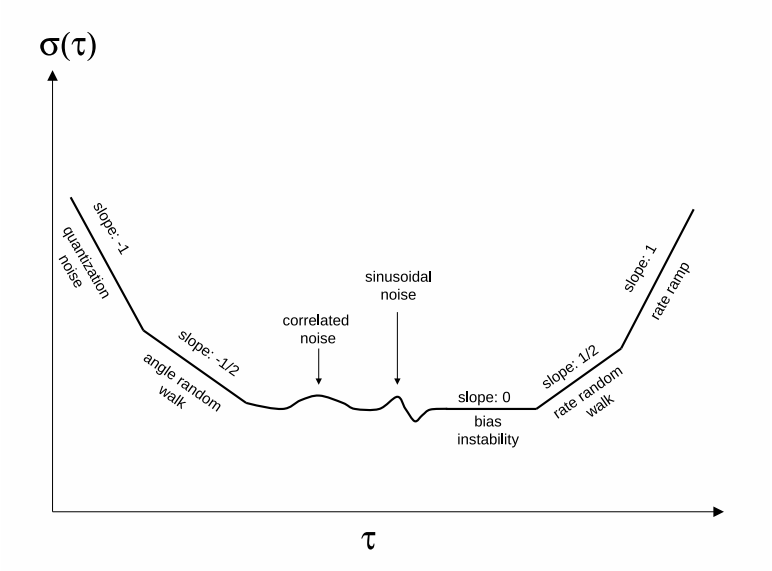


Figure 3.5: Sample Allan variance plot (adopted from [57]).

The Allan variance plots for the MTx-49A53G25 unit and MTx-49A83G25

Table 3.5: PSD and Allan variance representations of various noise types.

noise type	**PSD** ($S_s(f)$)	**Allan variance** ($\sigma^2(\tau)$)
quantization	$(2\pi f)^2 \sigma_Q^2$	$\frac{3\sigma_Q^2}{\tau}$
rate random walk	$\left(\frac{\sigma_K}{2\pi f}\right)^2$	$\frac{\sigma_K^2 \tau}{3}$
bias instability	$\frac{B^2}{2\pi f}$	$\frac{2B^2 \ln 2}{\pi}$
white	σ_N^2	$\frac{\sigma_N^2}{\tau}$
sinusoidal	$\frac{\Omega_0^2}{2}\delta(f - f_0)$	$\Omega_0^2 \left(\frac{\sin^2 \pi f_0 \tau}{\pi f_0 \tau}\right)^2$
correlated	$\frac{q_c T_c}{1+(2\pi f T_c)^2}$	$\frac{q_c^2 T_c^2}{\tau}\left[1 - \frac{T_c}{2\tau}\left(3 - 4e^{-\frac{\tau}{T_c}} + e^{-\frac{2\tau}{T_c}}\right)\right]$

unit are given in Figure 3.6((a), (c)) and Figure 3.6((b), (d)), respectively. As observed in the figures, the curves have a slope of $-1/2$ for small values of τ. This corresponds to white noise in the gyroscope or accelerometer data and is the angle random walk (for gyroscopes) and velocity random walk (for accelerometers) mentioned above. In the PSD domain, this noise is represented by

$$S_s(f) = \sigma_N^2 \tag{3.11}$$

where σ_N is the associated random walk coefficient. The corresponding Allan variance is

$$\sigma^2(\tau) = \frac{\sigma_N^2}{\tau} \tag{3.12}$$

which has a slope of $-1/2$ in the log-log Allan standard deviation plot. The corresponding coefficient σ_N can be found from the plots as the value corresponding to $\tau = 1$ [57]. The random walk coefficients for the sensors are given in Table 3.6.

The units given in the table are the units usually used by inertial sensor manufacturers. With proper conversion, this corresponds to the power of the white noise in the PSD plot. The conversion formulas are:

$$1\,^\circ/\sqrt{\text{h}} = \frac{\pi}{10800}\,\text{rad/s}/\sqrt{\text{Hz}} = 0.00029\,\text{rad/s}/\sqrt{\text{Hz}}$$

$$1\,\text{m/s}/\sqrt{\text{h}} = \frac{1}{60}\,\text{m/s}^2/\sqrt{\text{Hz}} = 0.01667\,\text{m/s}^2/\sqrt{\text{Hz}}$$

In the Allan variance plot, white noise in the signal dominates the other errors for small τ values. As the length τ of the averaging interval increases, the variance

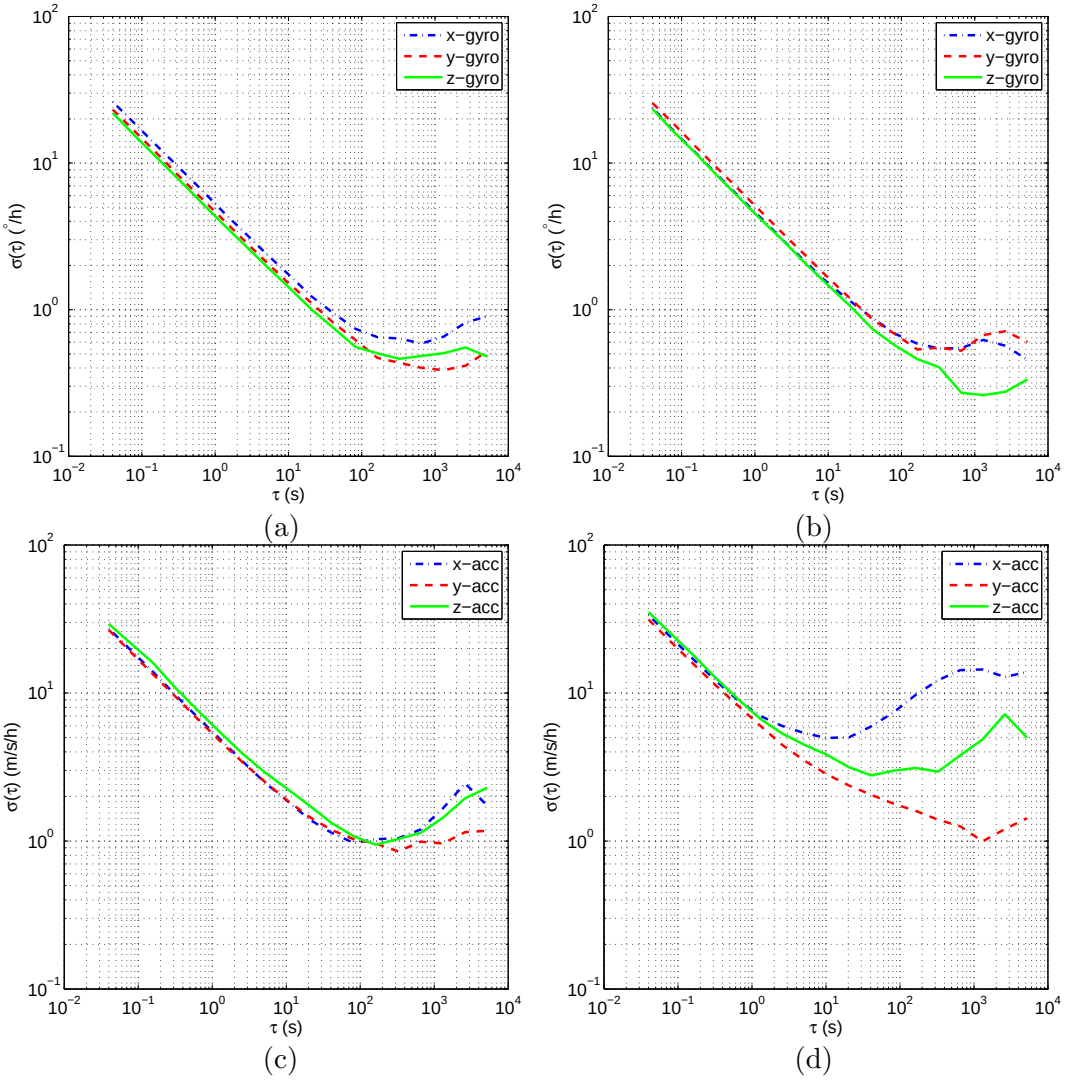


Figure 3.6: Allan variance plots for x, y, z gyroscopes of the (a) MTx-49A53G25 unit, (b) MTx-49A83G25 unit, and x, y, z accelerometers of (c) MTx-49A53G25 unit, (b) MTx-49A83G25 unit.

decreases. However, after a certain τ value, the variance begins to increase due to the random walk in the sensor output. This is referred to as *rate random walk* for gyroscopes and *acceleration random walk* for accelerometers. However, at larger τ values, other sources of error are also observed, and also the reliability of the Allan variance estimate decreases because fewer terms are included in the summations, in Equation (3.8) or (3.9) . Because of these facts, it is difficult to estimate the noise parameters in the large τ region [56].

The minimum value on the Allan variance plot is defined as *bias instability*.

Table 3.6: Velocity/angle random walk coefficients obtained from Allan variance plots. The units are $(^\circ/\sqrt{\text{h}})$ for gyroscopes and $(\text{m/s}/\sqrt{\text{h}})$ for accelerometers.

	MTx-49A53G25	MTx-49A83G25
x-gyro	5.2770	4.6306
y-gyro	4.6530	5.1360
z-gyro	4.3411	4.5241
x-acc	0.0916	0.1277
y-acc	0.0888	0.1126
z-acc	0.1019	0.1249

This is the best stability that can be achieved with a fully modeled sensor and active bias estimation [63]. In [68], the bias instability is estimated using the same way for MTx sensors. The PSD associated with this noise is

$$S_s(f) = \begin{cases} \frac{B^2}{2\pi f} & f \leq f_0 \\ 0 & f > f_0 \end{cases} \tag{3.13}$$

and the corresponding Allan variance can be approximated as

$$\sigma^2(\tau) = \frac{2B^2 \ln 2}{\pi} \tag{3.14}$$

where B is the bias instability coefficient. Table 3.7 gives the values of the bias instability coefficient B for the sensors considered in this study.

Table 3.7: Bias instability values obtained from Allan variance plots. The units are (mg) for accelerometers and ($^\circ$/h) for gyroscopes.

	MTx-49A53G25	MTx-49A83G25
x-gyro	0.8854 (655 s)	0.8173 (327 s)
y-gyro	0.5822 (1311 s)	0.7902 (655 s)
z-gyro	0.6956 (327 s)	0.3935 (1311 s)
x-acc	0.0414 (81 s)	0.2122 (10 s)
y-acc	0.0363 (327 s)	0.0425 (1311 s)
z-acc	0.0402 (163 s)	0.1182 (40 s)

Note that the bias instability of the x-axis accelerometer in the MTx-49A83G25 unit is significantly larger than the bias instability values of other

accelerometers. This can also be observed in Figure 3.3(e), where the bias of the accelerometer changes at about 8 hours. For the gyroscopes, the signals are composed of uncorrelated noise, which means that biases are successfully removed. Thus, the bias instability values for the gyroscopes in Table 3.7 are quite small.

3.5 Magnetometer Characterization

Unlike gyroscopes and accelerometers, the temperature-dependent bias is not observed for the magnetometers in our experiments. Therefore, the curve fitting procedure above is not applied to the magnetometer signals. When placed on a stationary table, the magnetometer outputs are not expected to be zero; they are expected to output the Earth magnetic field strength in the corresponding direction. Thus, we only subtract the mean value in order to obtain the residuals of magnetometer signals. The Allan variance plots of these residuals are given in Figure 3.7 for MTx-49A53G25 and MTx-49A83G25 units, respectively.

As observed in Figure 3.7, there is high correlation in the magnetometer residuals and it is not possible to identify distinct regions in the Allan variance plot. We also present the autocorrelation and corresponding PSD plots in Figure 3.7. The PSD of the signals is estimated using the Welch method [69], which is used for signals of long duration.

Observing the figures, it can be concluded that white noise term is dominant in the high-frequency region. However, low-frequency correlated noise also exists in the residuals. In addition to these components, it is clear from the figures that there is a periodic component at 5 Hz and a less dominant component at 10 Hz. This may be caused by a magnetic disturbance in the laboratory environment, either with a fundamental frequency of 5 Hz, or with a frequency higher than the sampling rate that appears as 5 Hz due to aliasing. In any case, the power related to the noise components in the magnetometer signal is rather low, as compared to the inertial sensors. Furthermore, the magnetometer signals are not integrated in the localization process, but used directly. The errors caused by

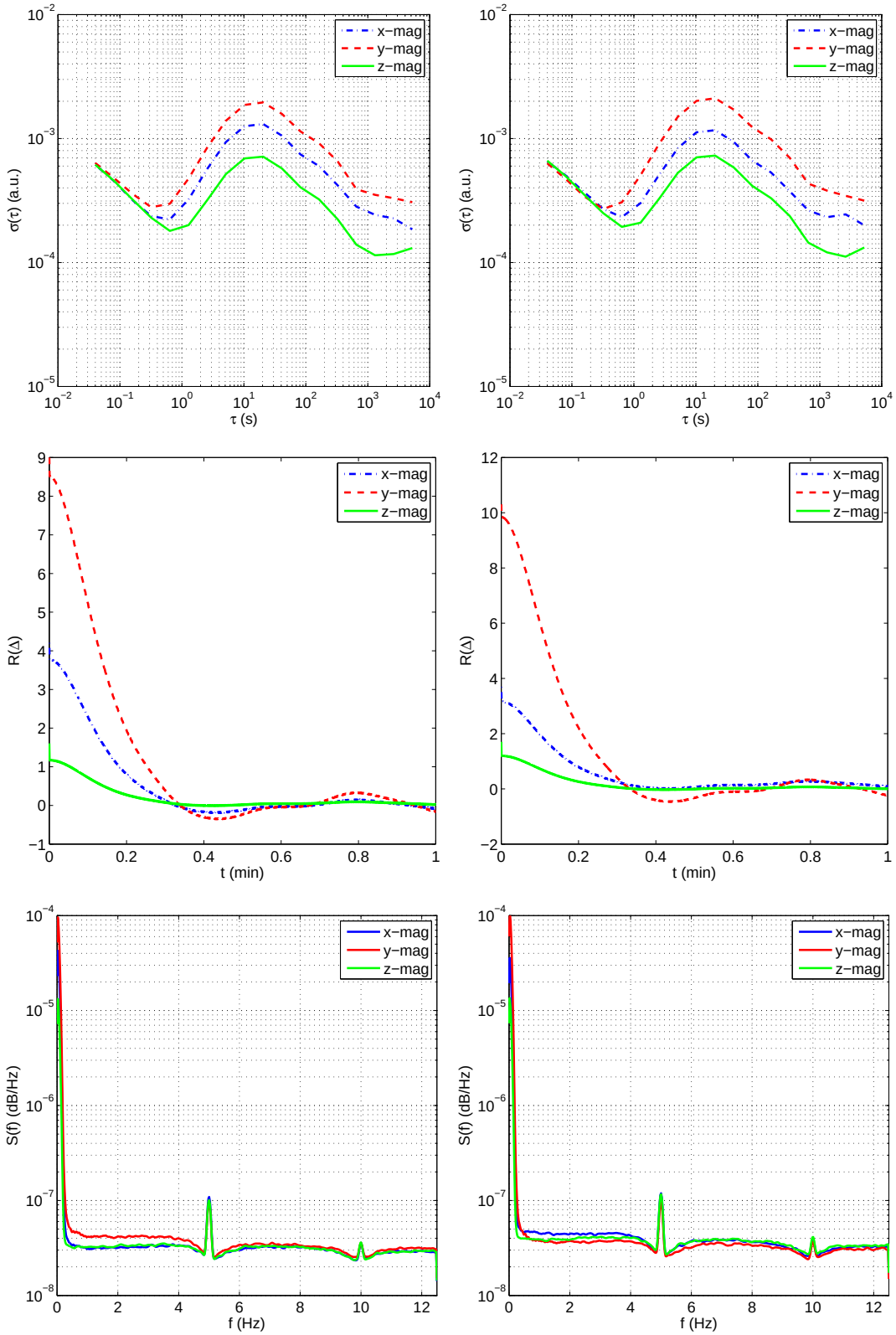


Figure 3.7: Allan variance, autocorrelation, and PSD plots of magnetometer residuals for the MTx-49A53G25 (left column), and MTx-49A83G25 (right column) units.

the magnetometer noise are expected to be low compared to the inertial sensor errors that grow with time. Therefore, we do not believe that it is necessary to go on further in modeling the correlated noise components.

Chapter 4

Human Activity Recognition Using Body-Worn Inertial/Magnetic Sensors

4.1 Introduction

In this chapter, we explore the application of inertial sensing to human activity monitoring, recognition, and classification through body-worn sensors [5, 8, 70, 71, 72, 73]. This has a broad range of potential applications in biomechanics [73, 74], ergonomics [75], remote monitoring of the physically or mentally disabled, the elderly, and children [76], detecting and classifying falls [77, 78, 79], medical diagnosis and treatment [80], home-based rehabilitation and physical therapy [81], sports science [82], ballet and other forms of dance [83], animation and film making, computer games [84, 85], professional simulators, virtual reality, and stabilization of equipment through motion compensation.

Early studies in activity recognition employed vision-based systems with single or multiple video cameras, and this remains to be the most common approach to date [86, 87, 88, 89]. For example, although the gesture recognition problem has been well studied in computer vision [90], much less research has been done in

this area with body-worn inertial sensors [91, 92]. The use of camera systems may be acceptable and practical when activities are confined to a limited area such as certain parts of a house or office environment and when the environment is well lit. However, when the activity involves going from place to place, camera systems are much less convenient. Furthermore, camera systems interfere considerably with privacy, may supply additional, unneeded information, and cause the subjects to act unnaturally.

Miniature inertial sensors can be flexibly used inside or behind objects without occlusion effects. This is a major advantage over visual motion-capture systems that require a free line of sight. When a single camera is used, the 3-D scene is projected onto a 2-D one, with significant information loss. Points of interest are frequently pre-identified by placing special, visible markers such as light-emitting diodes (LEDs) on the human body. Occlusion or shadowing of points of interest (by human body parts or objects in the surroundings) is circumvented by positioning multiple camera systems in the environment and using several 2-D projections to reconstruct the 3-D scene. This requires each camera to be separately calibrated. Another major disadvantage of using camera systems is that the cost of processing and storing images and video recordings is much higher than those of 1-D signals. 1-D signals acquired from multiple axes of inertial sensors can directly provide the required information in 3-D. Unlike high-end commercial inertial sensors that are calibrated by the manufacturer, in low-cost applications that utilize these devices, calibration is still a necessary procedure. Accelerometer-based systems are more commonly adopted than gyros because accelerometers are easily calibrated by gravity, whereas gyro calibration requires an accurate variable-speed turntable and is more complicated.

The use of camera systems and inertial sensors are two inherently different approaches that are by no means exclusive and can be used in a complementary fashion in many situations. In a number of studies, video cameras are used only as a reference for comparison with inertial sensor data [93, 94, 95, 96, 97, 98]. In other studies, data from these two sensing modalities are integrated or fused [99, 100]. The fusion of visual and inertial data has attracted considerable attention recently because of its robust performance and potentially wide applications [101,

102]. Fusing the data of inertial sensors and magnetometers is also reported in the literature [96, 103, 104].

Previous work on activity recognition based on body-worn inertial sensors is fragmented, of limited scope, and mostly unsystematic in nature. Due to the lack of a common ground among different researchers, results published so far are difficult to compare, synthesize, and build upon in a manner that allows broad conclusions to be reached. A unified and systematic treatment of the subject is desirable; theoretical models need to be developed that will enable studies designed such that the obtained results can be synthesized into a larger whole.

Most previous studies distinguish between sitting, lying, and standing [76, 93, 94, 95, 98, 105, 106, 107, 108], as these postures are relatively easy to detect using the static component of acceleration. Distinguishing between walking, and ascending and descending stairs has also been accomplished [105, 106, 108], although not as successfully as detecting postures. The signal processing and motion detection techniques employed, and the configuration, number, and type of sensors differ widely among the studies, from using a single accelerometer [76, 109, 110] to as many as 12 [111] on different parts of the body. Although gyroscopes can provide valuable rotational information in 3-D, in most studies, accelerometers are preferred to gyroscopes because of their ease of calibration. To the best of our knowledge, guidance on finding a suitable configuration, number, and type of sensors does not exist [105]. Usually, some configuration and some modality of sensors is chosen without strong justification, and empirical results are presented. Processing the acquired signals is also often done ad hoc and with relatively unsophisticated techniques.

In this work, we use miniature inertial sensors and magnetometers positioned on different parts of the body to classify human activities. The motivation behind investigating activity classification is its potential applications in the many different areas mentioned above. The main contribution of this study is that unlike previous studies, we use many redundant sensors to begin with and extract a variety of features from the sensor signals. Then, we use an unsupervised feature transformation technique that allows considerable feature reduction through

automatic selection of the most informative features. As another approach, we perform feature selection among the large number of features using a greedy search technique. We provide an extensive and systematic comparison between various classification techniques used for human activity recognition based on the same data set. We compare the successful differentiation rates, confusion matrices, and computational requirements of the techniques.

4.2 Classified Activities and Experimental Methodology

The 19 activities that are classified using body-worn miniature inertial sensor units are: sitting (A1), standing (A2), lying on back and on right side (A3 and A4), ascending and descending stairs (A5 and A6), standing in an elevator still (A7) and moving around (A8), walking in a parking lot (A9), walking on a treadmill with a speed of 4 km/h (in flat and 15° inclined positions) (A10 and A11), running on a treadmill with a speed of 8 km/h (A12), exercising on a stepper (A13), exercising on a cross trainer (A14), cycling on an exercise bike in horizontal and vertical positions (A15 and A16), rowing (A17), jumping (A18), and playing basketball (A19).

Five MTx 3-DOF orientation trackers (Figure 3.1) are used, manufactured by Xsens Technologies [54]. We use two standard units (MTx-49A53G25) worn on the arms (wrists) and three customized units (MTx-49A83G25) worn on the legs and the chest, as depicted in Figure 4.1. Since leg motions in general may produce larger accelerations, we use customized units with higher accelerometer ranges on the legs.

Each activity listed above is performed by eight different subjects (4 female, 4 male, between the ages 20 and 30) for 5 min. The profiles of the subjects are given in Table 4.1. The subjects are asked to perform the activities in their own style and were not restricted on how the activities should be performed. For this reason, there are inter-subject variations in the speeds and amplitudes of some

Figure 4.1: Positioning of Xsens sensor modules on the body.

Table 4.1: Subjects that performed the experiments and their profiles.

subject no.	gender	age	height (cm)	weight (kg)
S1	f	25	170	63
S2	f	20	162	54
S3	m	30	185	78
S4	m	25	182	78
S5	m	26	183	77
S6	f	23	165	50
S7	f	21	167	57
S8	m	24	175	75

activities. The activities are performed at the Bilkent University Sports Hall, in the Electrical and Electronics Engineering Building, and in a flat outdoor area on campus. Sensor units are calibrated to acquire data at 25 Hz sampling frequency. The 5-min signals are divided into 5-s segments, from which certain features are extracted. In this way, 480 (= 60 × 8) signal segments are obtained for each activity.

4.3 Feature Extraction

After acquiring the signals as described above, we obtain a discrete-time sequence of N_s elements that can be represented as an $N_s \times 1$ vector $\mathbf{s} = [s_1 \; s_2 \; \ldots \; s_{N_s}]^T$. For the 5-s time windows and the 25-Hz sampling rate, $N_s = 125$. The initial set of features we use before feature reduction are the minimum and maximum values, the mean value, variance, skewness, kurtosis, autocorrelation sequence, and the peaks of the discrete Fourier transform (DFT) of $\mathbf{s}$ with the corresponding frequencies. These are calculated as follows:

$$\begin{aligned}
\text{mean}(\mathbf{s}) &= \mu_{\mathbf{s}} = E\{\mathbf{s}\} = \frac{1}{N_s}\sum_{i=1}^{N_s} s_i \\
\text{variance}(\mathbf{s}) &= \sigma^2 = E\{(\mathbf{s}-\mu_{\mathbf{s}})^2\} = \frac{1}{N_s}\sum_{i=1}^{N_s}(s_i-\mu_{\mathbf{s}})^2 \\
\text{skewness}(\mathbf{s}) &= \frac{E\{(\mathbf{s}-\mu_{\mathbf{s}})^3\}}{\sigma^3} = \frac{1}{N_s\sigma^3}\sum_{i=1}^{N_s}(s_i-\mu_{\mathbf{s}})^3 \\
\text{kurtosis}(\mathbf{s}) &= \frac{E\{(\mathbf{s}-\mu_{\mathbf{s}})^4\}}{\sigma^4} = \frac{1}{N_s\sigma^4}\sum_{i=1}^{N_s}(s_i-\mu_{\mathbf{s}})^4 \\
\text{autocorrelation}: \quad R_{\mathbf{ss}}(\Delta) &= \frac{1}{N_s-\Delta}\sum_{i=0}^{N_s-\Delta-1}(s_i-\mu_{\mathbf{s}})\,(s_{i-\Delta}-\mu_{\mathbf{s}}) \\
\text{DFT}: \quad S_{\text{DFT}}(k) &= \sum_{i=0}^{N_s-1} s_i\, e^{-\frac{j2\pi ki}{N_s}}
\end{aligned} \tag{4.1}$$

In these equations, s_i is the ith element of the discrete-time sequence $\mathbf{s}$, $E\{\cdot\}$ denotes the expectation operator, $\mu_{\mathbf{s}}$ and σ are the mean and the standard deviation of $\mathbf{s}$, $R_{\mathbf{ss}}(\Delta)$ is the unbiased autocorrelation sequence of $\mathbf{s}$, and $S_{\text{DFT}}(k)$

is the kth element of the 1-D N_s-point DFT. In calculating the first five features above, it is assumed that the signal segments are the realizations of an ergodic process so that ensemble averages are replaced with time averages. Apart from those listed above, we have also considered using features such as the total energy of the signal, cross-correlation coefficients of two signals, and the discrete cosine transform coefficients of the signal.

Since there are five sensor units (MTx), each with three tri-axial devices, a total of nine signals are recorded from every sensor unit. Different signal representations, such as the time-domain signal, its autocorrelation function, and its DFT for two selected activities are given in Figure 4.2. In parts (a) and (c) of the figure, the quasi-periodic nature of the walking signal can be observed.

When a feature such as the mean value of a signal is calculated, 45 (= 9 axes $\times$ 5 units) different values are available. These values from the five sensor units are placed in the feature vectors in the order of right arm, left arm, right leg, torso, and left leg. For each one of these sensor locations, nine values for each feature are calculated and recorded in the following order: the x, y, z axes' acceleration, the x, y, z axes' rate of turn, and the x, y, z axes' Earth's magnetic field. In constructing the feature vectors, the above procedure is followed for the minimum and maximum values, the mean, skewness, and kurtosis. Thus, 225 (= 45 axes $\times$ 5 features) elements of the feature vectors are obtained by using the above procedure.

After taking the DFT of each 5-s signal, the maximum five Fourier peaks are selected so that a total of 225 (= 9 axes $\times$ 5 units $\times$ 5 peaks) Fourier peaks are obtained for each segment. Each group of 45 peaks is placed in the order of right arm, left arm, right leg, torso, and left leg, as above. The 225 frequency values that correspond to these Fourier peaks are placed after the Fourier peaks in the same order.

Eleven autocorrelation samples are placed in the feature vectors for each axis of each sensor, following the order given above. Since there are 45 distinct sensor signals, 495 (= 45 axes $\times$ 11 samples) autocorrelation samples are placed in each feature vector. The first sample of the autocorrelation function (the variance)

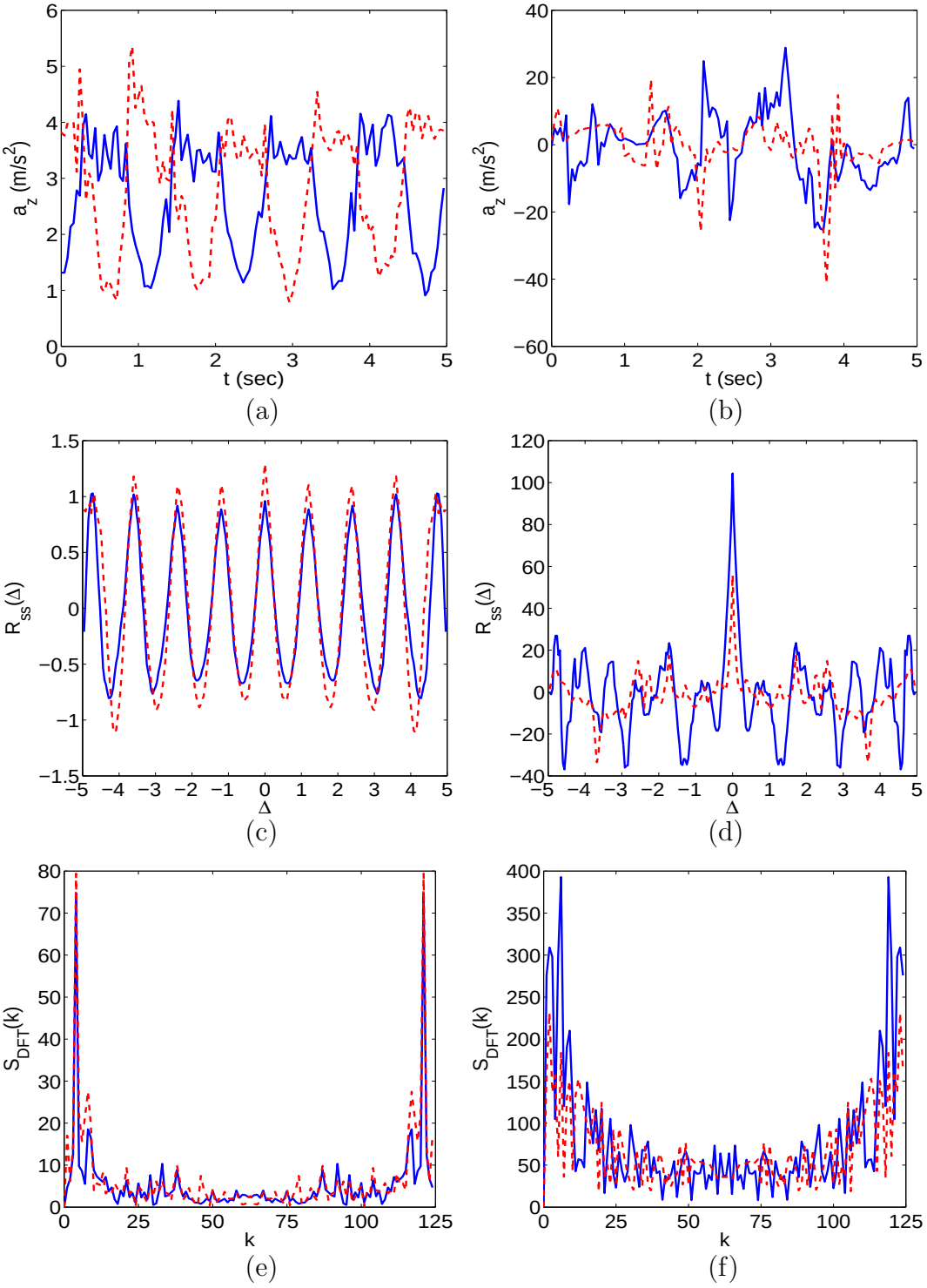


Figure 4.2: (a) and (b): Time-domain signals for walking and basketball, respectively; z axis acceleration of the right (solid lines) and left arm (dashed lines) are given; (c) and (d): autocorrelation functions of the signals in (a) and (b); (e) and (f): 125-point DFT of the signals in (a) and (b), respectively.

and every fifth sample up to the fiftieth are placed in the feature vectors for each signal.

As a result of the above feature extraction process, a total of $1,170\,(= 225 + 225 + 225 + 495)$ features are obtained for each of the 5-s signal segments so that the dimensions of the resulting feature vectors are $1,170 \times 1$. All features are normalized to the interval $[0, 1]$ so as to be used for classification.

4.4 Pattern Recognition Methodology

4.4.1 Feature Reduction and Selection

When the number of features used for classification is large, it is often required to use a reduced feature set. This is because of the well-known "curse of dimensionality," which states that the number of samples required for an accurate estimation of the probability density function of a class grows exponentially with the dimension of the feature space [112]. Thus, various methods are developed to reduce the number of features in the samples. One class of these methods reduces the features by using combinations of the existing features, and they are referred to as feature reduction methods. Another class selects individual features from the existing feature set, and they are referred to as feature selection methods. Here, we explain Principal Components Analysis (PCA) as a feature reduction method and sequential feature selection methods.

4.4.1.1 Principal Components Analysis

The PCA method projects the feature space to a lower-dimensional subspace, by considering the variance of the data. The eigenvalues and eigenvectors of the covariance matrix of the data are calculated. Since the covariance matrix is symmetric and positive semi-definite, the eigenvalues are nonnegative and the eigenvectors are orthogonal. Sorting the eigenvectors in the descending order of

corresponding eigenvalues, the sorted eigenvectors represent the principal directions in the feature space in which the variance of the data is the largest. The eigenvectors with comparably small eigenvalues are discarded, and the data is projected to the subspace spanned by the first few eigenvectors.

4.4.1.2 Feature Selection

As opposed to most feature reduction methods, feature selection methods usually make use of physically motivated features rather than using combinations of features. We consider two greedy search algorithms, namely sequential forward and backward feature selection methods.

Sequential forward feature selection (SFFS) method starts with the empty feature set, then adds features one at a time to the current feature set such that the classification performance is maximized. A more detailed description of the method can be found in [113].

Sequential backward feature selection (SBFS) method starts with the full set of features, and removes features one at a time from the current feature set such that the classification performance is maximized [113].

4.4.2 Pattern Classifiers

In this section, we describe the classification techniques used in this study.

4.4.2.1 Bayesian Decision Making (BDM)

In BDM, class conditional probability density functions (CCPDFs) are estimated for each class. In this study, the CCPDFs are assumed to have a multi-variate Gaussian parametric form, and the mean vector and the covariance matrix of the CCPDF for each class are estimated using maximum likelihood estimators on the training vectors. It also assumed that the prior probabilities are equal. For a

given test vector $\mathbf{x}$, the maximum a posteriori (MAP) decision rule is used for classification [113].

4.4.2.2 Rule-Based Algorithm (RBA)

A rule-based algorithm or a decision tree can be considered as a sequential procedure that classifies given inputs. An RBA follows predefined rules at each node of the tree and makes binary decisions based on these rules. Rules correspond to conditions such as "is feature $f_i \leq \eta_i$?," where η is the threshold value for a given feature and $i = 1, 2, \ldots, M$, with M being the total number of features used [114].

As the information necessary to differentiate between the activities is completely embodied in the decision rules, the RBA has the advantage of not requiring the storage of any reference feature vectors. The main difficulty is in designing the rules and making them independent of absolute quantities so that they will be more robust and generally applicable.

In this study, we automatically generate a binary decision tree based on the training data using the CART algorithm [115]. Given a set of training vectors along with their class labels, a binary tree, and a decision rule for each node of the tree, each node corresponds to a particular subset of the training vectors where each element of that subset satisfies the conditions imposed by the ancestors of that node. Thus, a decision at a node splits the corresponding subset into two: those that satisfy the condition and those that do not. Naturally, the ideal split is expected to isolate a class from others at each decision node. Since this is not the case in practice, a decision rule is found by searching among all possible decisions that minimize the *impurity* of that node. We use entropy as a measure of impurity, and the class frequencies at each node to estimate the entropy [115]. Test vectors are then used to evaluate the classification performance of the decision tree.

4.4.2.3 Least Squares Method (LSM)

In LSM, the average training vector for each class is calculated as a representative for that particular class. Each test vector is compared with the average training vector (instead of each individual reference vector) as follows:

$$\mathcal{D}_i^2 = \sum_{n=1}^{N_f}(f_n - e_{in})^2 = (f_1 - e_{i1})^2 + \ldots + (f_{N_f} - e_{iN_f})^2 \qquad i = 1, \ldots, c \quad (4.2)$$

The test vector is assigned to the same class as the nearest average reference vector. In this equation, $\mathbf{f} = [f_1 \ f_2 \ \ldots \ f_{N_f}]^T$ represents a test feature vector, $\mathbf{e} = [e_{i1} \ e_{i2} \ \ldots \ e_{iN_f}]^T$ represents the average of the reference feature vectors for each distinct class, and $\mathcal{D}_i^2$ is the square of the distance between these two vectors.

4.4.2.4 k-Nearest Neighbor (k-NN)

In k-NN, the k nearest neighbors of the vector $\mathbf{f}$ in the training set are considered and the vector $\mathbf{f}$ is classified into the same class as the majority of its k nearest neighbors [113]. In this study, the Euclidean distance measure is used. The k-NN algorithm is sensitive to the local structure of the data. The selection of the parameter k, the number of neighbors considered, is a very important issue that can affect the decision made by the k-NN classifier. Unfortunately, a pre-defined rule for the selection of the value of k does not exist. In this study, the number of nearest neighbors k is determined experimentally by maximizing the correct classification rate over different k values.

4.4.2.5 Dynamic Time Warping (DTW)

Dynamic time warping is an algorithm for measuring the similarity between two sequences that may vary in time or speed. An optimal match between two given sequences (e.g., a time series) is found under certain restrictions. The sequences are "warped" nonlinearly in the time dimension to determine a measure of their similarity independent of certain nonlinear variations in the time dimension. In

DTW, the aim is to find the least-cost warping path for the tested feature vector among the stored reference feature vectors [116] where the cost measure is typically taken as the Euclidean distance between the elements of the feature vectors. DTW is used mostly in automatic speech recognition to handle different speaking speeds [116, 117]. Besides speech recognition, DTW has been used in signature and gait recognition, for ECG signal classification, for fingerprint verification, for word spotting in handwritten historical documents on electronic media and machine-printed documents, and for face localization in color images [118, 119]. In this study, DTW is used for classifying feature vectors of different activities extracted from the signals of miniature inertial sensors.

4.4.2.6 Support Vector Machines (SVM)

The support vector machine classifier is a machine learning technique proposed early in the 1980s [112, 120, 121]. It has been mostly used in applications such as object, voice, and handwritten character recognition, and in text classification.

If the feature vectors in the original feature space are not linearly separable, SVMs pre-process and represent them in a higher-dimensional space where they can become linearly separable. The dimension of the transformed space may sometimes be much higher than the original feature space. With a suitable nonlinear mapping $\phi(.)$ to a sufficiently high dimension, data from two different classes can always be made linearly separable, and separated by a hyperplane. The choice of the nonlinear mapping method depends on the prior information available to the designer. If information is not available, one might choose to use polynomials, Gaussians, or other types of basis functions. The dimensionality of the mapped space can be arbitrarily high, however, in practice, it may be limited by computational resources. The complexity of SVMs is related to the number of resulting support vectors rather than the high dimensionality of the transformed space.

In this study, the SVM method is applied to differentiate feature vectors that belong to more than two classes (19 classes). Following the one-versus-the-rest

method, c different binary classifiers are trained, where each classifier recognizes one of c activity types. A nonlinear classifier with a radial basis function kernel $K(\mathbf{f}, \mathbf{f}_i) = e^{-\zeta|\mathbf{f}-\mathbf{f}_i|^2}$ is used with $\zeta = 4$. A library for SVMs (LIBSVM toolbox) is used in the MATLAB environment [122].

4.4.2.7 Artificial Neural Networks (ANN)

Multi-layer ANNs consist of an input layer, one or more hidden layers to extract progressively more meaningful features, and a single output layer, each composed of a number of units called *neurons*. The model of each neuron includes a smooth nonlinearity, called the *activation function*. Due to the presence of distributed nonlinearity and a high degree of connectivity, theoretical analysis of ANNs is difficult. These networks are trained to compute the boundaries of decision regions in the form of connection weights and biases by using training algorithms. The performance of ANNs is affected by the choice of parameters related to the network structure, training algorithm, and input signals, as well as by parameter initialization [123, 124].

In this work, a three-layer ANN is used for classifying human activities. The input layer has N neurons, equal to the dimension of the feature vectors (30). The hidden layer has 12 neurons, and the output layer has c neurons, equal to the number of classes. The number of hidden neurons is determined experimentally. In the input and hidden layers each, there is an additional neuron with a bias value of 1. For an input feature vector $\mathbf{x} \in \mathbb{R}^N$, the target output is 1 for the class that the vector belongs to, and 0 for all other output neurons. The sigmoid function used as the activation function in the hidden and output layers is given by $g(x) = (1 + e^{-x})^{-1}$.

The output neurons can take continuous values between 0 and 1. Fully connected ANNs are trained with the back-propagation algorithm [123] by presenting a set of *training patterns* to the network. The aim is to minimize the average of

the sum of squared errors over all training vectors:

$$\mathcal{E}_{\text{av}}(\mathbf{w}) = \frac{1}{2I} \sum_{i=1}^{I} \sum_{k=1}^{c} [t_{ik} - o_{ik}(\mathbf{w})]^2 \tag{4.3}$$

Here, $\mathbf{w}$ is the weight vector, t_{ik} and o_{ik} are the desired and actual output values for the ith training pattern and the kth output neuron, and I is the total number of training patterns. When the entire training set is covered, an *epoch* is completed. The error between the desired and actual outputs is computed at the end of each iteration and these errors are averaged at the end of each epoch (Equation (4.3)). The training process is terminated when a certain precision goal on the average error is reached or if the specified maximum number of epochs (5,000) is exceeded, whichever occurs earlier. The latter case occurs very rarely. The acceptable average error level is set to a value of 0.03. The weights are initialized randomly with a uniform distribution in the interval $[0, 0.2]$, and the learning rate is chosen as 0.2.

In the test phase, the test feature vectors are fed forward to the network, the outputs are compared with the desired outputs, and the error between them is calculated. The test vector is said to be correctly classified if this error is below a threshold value of 0.25, determined experimentally.

4.4.3 Statistical Cross-Validation Methods

In many applications of pattern recognition, it is needed to evaluate the performances of the designed classifiers. Usual practice is forming separate training and test sets. The training set is used to design the classifiers, i.e., to estimate the classifier parameters. Then the test set is used to evaluate the classification performance.

One approach to this problem is acquiring training and test data separately. In our case of human activity recognition, this would correspond to acquiring two sets of data from all subjects, unavoidably under slightly different conditions, in order to be used in separate training and test sets.

The approach followed in this study is using cross-validation methods. In this approach, all available data are partitioned into training and test sets, and each sample is expected to be used both for training and for testing. Below, we explain the cross-validation methods used in this study.

4.4.3.1 Repeated Random Sub-Sampling (RRSS)

In this method, the data are randomly divided into two sets, one to be used for training and one to be used for testing. Then, the classifiers are trained with the training set and validated with the test set. Usually this division is repeated multiple times and the classification results are averaged. Due to the randomness of the division, some samples may not at all be used for validation in this method.

4.4.3.2 P-fold Cross Validation

In P-fold cross validation, the data are randomly divided into P disjoint sets. One of these sets is retained for validation, while the remaining $P-1$ are used to train the classifiers. This process is repeated P times (the folds), each time using a different set for validation. The results are then averaged. In this method, each sample is used exactly once for validation. As a rule of thumb, P is chosen around 10.

4.4.3.3 Leave-one-out (L1O) Cross Validation

In most pattern recognition applications, L1O cross validation is the same as P-fold cross validation, with P being equal to the number of samples in the whole data set. Thus, at each fold, only one sample is used for validation and the remaining samples are used for training.

In this study, our experiments are conducted with multiple subjects. Thus, we modify this method as subject-based L1O cross validation. At each fold, the data obtained from a single subject are retained for validation, and the data from

remaining subjects are used for training. Therefore, a particular subject's data are exactly used once for validation. This method can also be considered as a variant of P-fold cross validation, with P being equal to the number of subjects, and the partitioning is not performed randomly but according to the subject from which the data are acquired.

4.5 Experimental Results

4.5.1 Feature Selection and Reduction

Because the initial set of features is quite large (1,170) and not all features are equally useful in discriminating between the activities, we investigate different feature selection and reduction methods [125]. As the first approach, we reduce the number of features from 1,170 to 30 through PCA. The reduced dimension of the feature vectors is determined by observing the eigenvalues of the covariance matrix of the $1,170 \times 1$ feature vectors, sorted in Figure 4.3(a) in descending order. The 30 eigenvectors corresponding to the largest 30 eigenvalues (Figure 4.3(b)) are used to form the transformation matrix, resulting in 30×1 feature vectors. Although the initial set of 1,170 features do have physical meaning, because of the matrix transformation involved, the transformed feature vectors cannot be assigned any physical meaning. Scatter plots of the first five transformed features are given in Figure 4.4 pairwise. As expected, in the first two plots or so (parts (a) and (b) of the figure), the features for different classes are better clustered and more distinct.

4.5.2 Selection of Validation Sets

The classification techniques described in Section 4.4.2 are employed to classify the 19 different activities using the 30 features selected by PCA. A total of $9,120$ ($=$ 60 feature vectors $\times$ 19 activities $\times$ 8 subjects) feature vectors are available, each containing the 30 reduced features of the 5-s signal segments. In

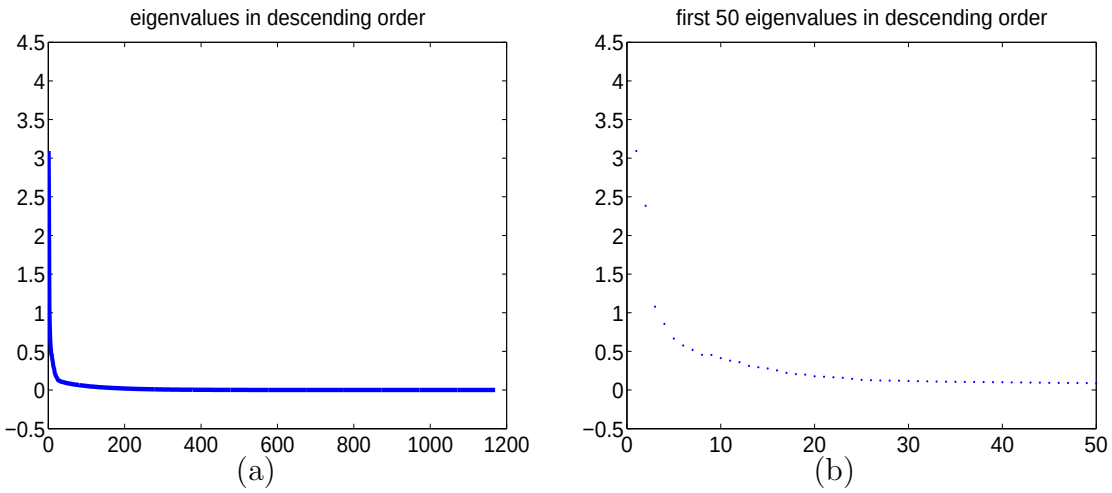


Figure 4.3: (a) All eigenvalues (1,170) and (b) the first 50 eigenvalues of the covariance matrix sorted in descending order.

the training and testing phases of the classification methods, we use the RRSS, P-fold, and L1O cross-validation techniques. In RRSS, we divide the 480 feature vectors from each activity type randomly into two sets so that the first set contains 320 feature vectors (40 from each subject) and the second set contains 160 (20 from each subject). Therefore, two-thirds (6,080) of the 9,120 feature vectors are used for training and one third (3,040) for testing. This is repeated 10 times and the resulting correct differentiation percentages are averaged. The disadvantage of this method is that some observations may never be selected in the testing or the validation phase, whereas others may be selected more than once. In other words, validation subsets may overlap.

In P-fold cross validation, the 9,120 feature vectors are divided into $P = 10$ partitions, where the 912 feature vectors in each partition are selected completely randomly, regardless of the subject or the class they belong to. One of the P partitions is retained as the validation set for testing, and the remaining $P - 1$ partitions are used for training. The cross-validation process is then repeated P times (the folds), where each of the P partitions is used exactly once for validation. The P results from the folds are then averaged to produce a single estimate. The random partitioning is repeated 10 times and the average correct differentiation percentage is reported. The advantage of this validation method over RRSS is that all feature vectors are used for both training and testing, and each feature

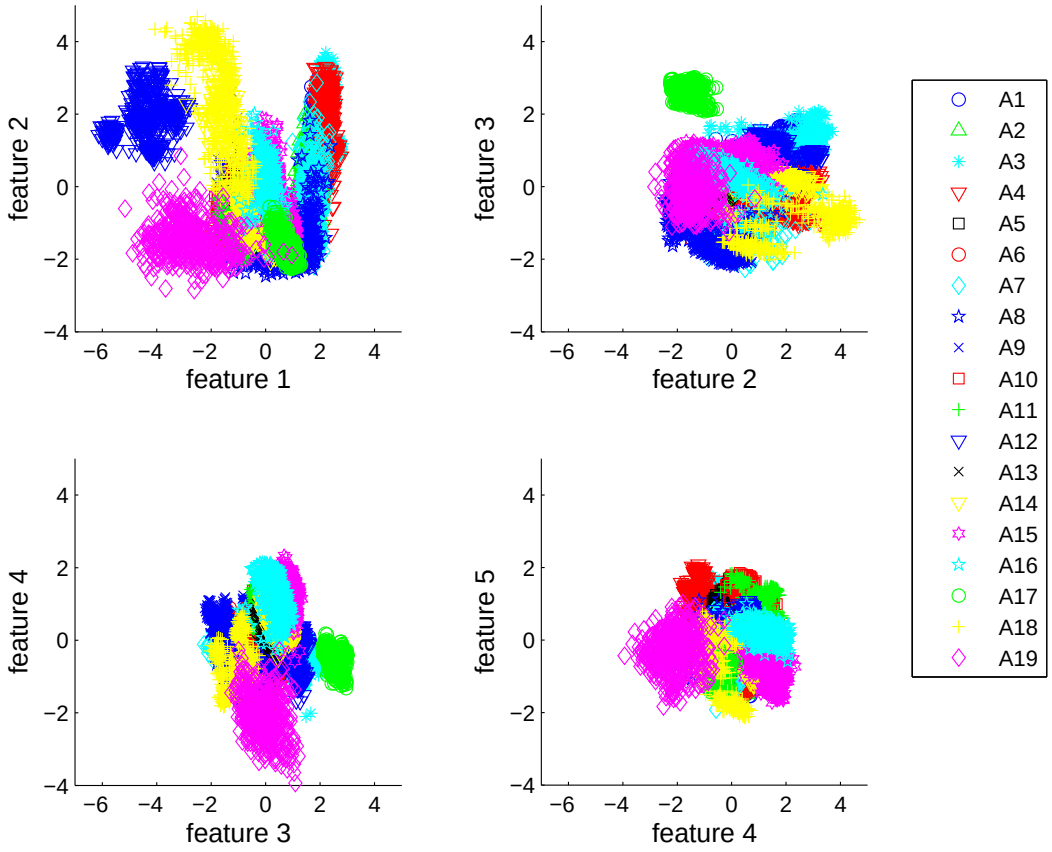


Figure 4.4: Scatter plots of the first five features selected by PCA.

vector is used for testing exactly once in each of the 10 runs.

Finally, we also used subject-based L1O cross validation, where the $7,980$ ($=$ 60 vectors $\times$ 19 activities $\times$ 7 subjects) feature vectors of seven of the subjects are used for training and the 1,140 feature vectors of the remaining subject are used in turn for validation. This is repeated eight times such that the feature vector set of each subject is used once as the validation data. The eight correct classification rates are averaged to produce a single estimate. This is similar to P-fold cross validation with P being equal to the number of subjects ($P = 8$), and where all the feature vectors in the same partition are associated with the same subject.

4.5.3 Results

Among the classification techniques we considered and implemented, when RRSS and P-fold cross validation are used, BDM gives the highest classification rate, followed by SVM and k-NN. RBA and DTW_1 perform the worst in general. In subject-based L1O cross validation, SVM is the best, followed by k-NN. The correct classification rates reported for L1O cross validation can be interpreted as the expected correct classification rates when data from a new subject are acquired and given as input to the classifiers. The most significant difference in the performances of the different validation methods is observed for the BDM method (Table 4.2). The RRSS and P-fold cross validation result in 99% correct classification rate, suggesting that the data are well represented by a multi-variate Gaussian distribution. However, the 76% correct classification rate of L1O cross validation implies that the parameters of the Gaussian, when calculated by excluding one of the subjects, cannot represent the data of the excluded subject sufficiently well. Thus, if one is to classify the activities of a new test subject whose training data are not available to the classifiers, SVM, k-NN, or LSM methods could be used.

Table 4.2: Correct differentiation rates for all classification methods and three cross-validation techniques. The results of the RRSS and P-fold cross-validation techniques are calculated over 10 runs, whereas those of L1O are over a single run.

	correct differentiation rate (%) ± one standard deviation		
method	RRSS	P-fold	L1O
BDM	99.1 ±0.12	99.2 ±0.02	75.8
RBA	81.0 ±1.52	84.5 ±0.44	53.6
LSM	89.4 ±0.75	89.6 ±0.10	85.3
k-NN ($k = 7$)	98.2 ±0.12	98.7 ±0.07	86.9
DTW_1	82.6 ±1.36	83.2 ±0.26	80.4
DTW_2	98.5 ±0.18	98.5 ±0.08	85.2
SVM	98.6 ±0.12	98.8 ±0.03	87.6
ANN	86.9 ±3.31	96.2 ±0.19	74.3

We chose to employ the P-fold cross-validation technique in reporting the results presented in Tables 4.3–4.9. Looking at the confusion matrices of the different techniques, it can be observed that A7 and A8 are the activities most confused with each other. This is because both of these activities are performed in the elevator and the signals recorded from these activities have similar segments. Therefore, confusion at the classification stage becomes inevitable. A2 and A7, A13 and A14, as well as A9, A10, A11, are also confused from time to time for similar reasons. Two activities that are almost never confused are A12 and A17.

The confusion matrices for BDM and RBA are provided in Tables 4.3 and 4.4. With these methods, correct differentiation rates of 99.2% and 84.5% are, respectively, achieved. The features used in the RBA correspond to the 30 features selected by PCA and the rules change at every training cycle.

In the LSM approach, test vectors are compared with the average of the reference vectors calculated for each of the 19 activities. The confusion matrix for this method is provided in Table 4.5. The overall successful differentiation rate of LSM is 89.6%.

Performance of the k-NN method changes for different values of k. A value of $k = 7$ gives the best results, therefore the confusion matrix of the k-NN algorithm is provided for $k = 7$ in Table 4.6, and a successful differentiation rate of 98.7% is achieved.

We have implemented the DTW algorithm in two different ways: In the first (DTW_1), the average reference feature vector of each activity is used for distance comparison. The confusion matrix for DTW_1 is presented in Table 4.7, and a correct differentiation rate of 83.2% is achieved. As a second approach (DTW_2), DTW distances are calculated between the test vector and each of the $8,208\,(= 9,120-912)$ reference vectors from other classes. The class of the nearest reference vector is assigned as the class of the test vector. The success rate of DTW_2 is 98.5% and the corresponding confusion matrix is given in Table 4.8.

In SVM, following the one-versus-the-rest method, each type of activity is assumed as the first class and the remaining 18 activity types are grouped into the

second class. With P-fold cross validation, 19 different SVM models are created for classifying the vectors in each partition, resulting in a total of 190 SVM models. The number of correctly and incorrectly classified feature vectors for each activity type is tabulated in Table 4.9(a). The overall correct classification rate of the SVM method is calculated as 98.8%.

For ANN, since the network classifies some samples as belonging to none of the classes and output neurons take continuous values between 0 and 1, it is not possible to form a confusion matrix. The number of correctly and incorrectly classified feature vectors with P-fold cross validation is given in Table 4.9(b). The overall correct classification rate of this method is 96.2%. On average, the network converges in about 400 epochs when P-fold cross validation is used.

Table 4.3: Confusion matrix for BDM (P-fold cross validation, 99.2%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t	A1	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A2	0	478	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
	A3	0	0	478	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
	A4	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A5	0	0	0	0	478	0	0	2	0	0	0	0	0	0	0	0	0	0	0
	A6	0	0	0	0	0	477	0	3	0	0	0	0	0	0	0	0	0	0	0
	A7	0	0	0	0	0	0	467	13	0	0	0	0	0	0	0	0	0	0	0
r	A8	0	0	0	0	0	0	44	435	0	0	0	0	0	0	0	0	0	0	1
	A9	0	0	0	0	0	0	0	1	479	0	0	0	0	0	0	0	0	0	0
u	A10	0	0	0	0	0	0	0	0	0	478	2	0	0	0	0	0	0	0	0
e	A11	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0	0
	A12	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
	A13	0	0	0	0	0	0	0	0	0	0	0	0	479	1	0	0	0	0	0
	A14	0	0	0	0	0	0	0	2	0	0	0	0	0	478	0	0	0	0	0
	A15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0
	A16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0
	A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0
	A18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0
	A19	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	476

Table 4.4: Confusion matrix for RBA (P-fold cross validation, 84.5%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t r u e	A1	418	6	22	4	0	0	8	2	0	0	1	1	0	5	7	2	2	0	2
	A2	9	404	0	1	2	5	37	9	1	0	2	2	1	1	3	2	0	1	0
	A3	12	1	439	13	0	0	2	1	1	1	1	2	3	0	2	1	1	0	0
	A4	7	3	11	446	4	0	4	0	0	0	0	0	0	0	0	2	1	2	0
	A5	0	2	1	1	421	1	4	11	10	4	5	1	1	4	0	6	3	3	2
	A6	4	2	1	0	4	409	10	19	7	0	0	0	9	1	7	0	1	5	1
	A7	8	33	2	3	1	16	360	48	0	0	0	0	1	0	2	0	1	2	3
	A8	2	17	1	2	21	31	60	266	10	4	4	1	13	7	7	3	8	7	16
	A9	0	3	1	1	6	5	1	4	397	20	11	4	7	8	0	9	0	2	1
	A10	0	1	0	1	2	1	0	2	14	416	27	0	2	7	1	2	0	2	2
	A11	0	1	1	2	2	0	0	2	13	38	404	3	1	9	0	1	0	2	1
	A12	1	0	2	2	0	0	0	0	0	3	2	456	2	2	2	3	0	1	4
	A13	1	1	0	0	1	0	1	4	8	1	3	4	404	35	6	6	0	1	4
	A14	0	1	1	1	1	0	0	8	5	3	6	5	20	411	5	4	0	3	6
	A15	3	0	2	0	1	8	0	4	2	0	0	3	4	1	432	9	9	0	2
	A16	2	3	1	1	9	2	1	3	3	2	3	2	9	8	7	420	2	0	2
	A17	1	0	3	0	2	7	0	1	1	0	0	0	2	0	5	1	455	2	0
	A18	0	1	1	1	2	7	1	9	8	1	2	4	0	1	1	2	5	430	4
	A19	1	1	1	3	1	6	1	17	2	5	2	11	12	10	0	1	7	5	394

Table 4.5: Confusion matrix for LSM (P-fold cross validation, 89.6%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t	A1	415	0	60	0	0	0	3	2	0	0	0	0	0	0	0	0	0	0	0
	A2	4	398	0	0	0	1	72	5	0	0	0	0	0	0	0	0	0	0	0
	A3	3	0	471	0	0	0	1	0	0	0	0	0	0	0	0	0	5	0	0
	A4	0	0	0	478	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	A5	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A6	0	0	0	0	0	448	0	32	0	0	0	0	0	0	0	0	0	0	0
	A7	16	55	0	1	6	1	350	51	0	0	0	0	0	0	0	0	0	0	0
r	A8	1	11	0	0	9	5	57	384	9	0	0	0	0	0	0	0	0	0	4
	A9	0	0	0	0	19	7	0	0	361	52	35	0	6	0	0	0	0	0	0
u	A10	0	0	0	0	0	0	0	0	0	414	66	0	0	0	0	0	0	0	0
e	A11	0	0	0	0	1	0	0	0	0	78	401	0	0	0	0	0	0	0	0
	A12	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
	A13	0	0	0	0	0	0	0	4	0	0	0	0	466	9	0	0	0	1	0
	A14	0	0	0	0	0	0	0	1	0	0	0	0	123	347	0	0	0	9	0
	A15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	476	1	2	0	0
	A16	0	0	0	0	16	0	0	0	0	0	0	0	0	1	0	462	0	1	0
	A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0
	A18	0	0	0	0	1	65	0	0	15	0	0	0	0	0	0	0	0	399	0
	A19	0	0	0	0	0	1	0	11	0	0	0	1	2	0	0	0	0	0	465

Table 4.6: Confusion matrix for the k-NN algorithm for $k = 7$ (P-fold cross validation, 98.7%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t r u e	A1	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A2	0	479	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	A3	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A4	0	0	0	479	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	A5	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A6	0	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0
	A7	1	11	0	0	0	2	446	20	0	0	0	0	0	0	0	0	0	0	0
	A8	0	5	0	0	4	10	38	422	1	0	0	0	0	0	0	0	0	0	0
	A9	0	0	0	0	0	1	0	0	477	1	1	0	0	0	0	0	0	0	0
	A10	0	0	0	0	0	0	0	0	0	474	6	0	0	0	0	0	0	0	0
	A11	0	0	0	0	0	0	0	0	0	1	479	0	0	0	0	0	0	0	0
	A12	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
	A13	0	0	0	0	0	0	0	0	0	0	0	0	476	4	0	0	0	0	0
	A14	0	0	0	0	0	0	0	0	0	0	0	0	1	479	0	0	0	0	0
	A15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	479	1	0	0	0
	A16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	479	0	0	0
	A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0
	A18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0
	A19	0	0	0	0	0	0	0	7	0	0	0	0	1	0	0	0	0	0	472

Table 4.7: Confusion matrix for DTW_1 (P-fold cross validation, 83.2%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t r u e	A1	413	2	44	15	0	0	5	0	0	0	0	0	1	0	0	0	0	0	0
	A2	14	436	1	0	0	11	16	2	0	0	0	0	0	0	0	0	0	0	0
	A3	38	2	430	2	0	0	4	0	0	0	0	0	0	0	1	0	3	0	0
	A4	2	0	33	442	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
	A5	0	0	0	0	476	1	0	2	0	1	0	0	0	0	0	0	0	0	0
	A6	0	0	0	0	0	433	2	45	0	0	0	0	0	0	0	0	0	0	0
	A7	24	85	2	0	10	3	305	51	0	0	0	0	0	0	0	0	0	0	0
	A8	3	13	0	0	19	4	68	355	8	3	0	0	2	1	1	0	0	1	2
	A9	0	0	0	0	36	11	0	0	338	42	41	0	6	4	0	2	0	0	0
	A10	0	0	0	0	4	0	0	0	2	312	136	0	5	3	0	18	0	0	0
	A11	0	0	0	0	5	0	0	1	1	134	307	0	4	7	1	20	0	0	0
	A12	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
	A13	0	0	0	0	1	0	0	3	1	1	0	0	433	36	0	5	0	0	0
	A14	0	0	0	0	0	2	0	0	1	0	3	0	167	296	0	8	0	3	0
	A15	1	0	0	0	0	0	0	7	1	0	0	0	9	0	448	14	0	0	0
	A16	0	1	0	0	38	0	0	1	3	0	0	0	8	2	6	421	0	0	0
	A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0
	A18	0	3	0	0	4	68	12	3	18	0	0	0	5	13	0	0	7	346	1
	A19	0	0	0	0	4	2	0	12	0	2	3	3	7	5	0	0	0	3	439

Table 4.8: Confusion matrix for DTW_2 (P-fold cross validation, 98.5%).

		c l a s s i f i e d																		
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
t r u e	A1	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A2	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A3	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A4	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A5	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	A6	0	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0
	A7	1	7	0	0	0	1	430	41	0	0	0	0	0	0	0	0	0	0	0
	A8	0	1	0	0	2	9	47	418	0	0	0	0	0	0	0	0	0	0	3
	A9	0	0	0	0	0	1	0	0	478	0	1	0	0	0	0	0	0	0	0
	A10	0	0	0	0	0	0	0	0	0	469	11	0	0	0	0	0	0	0	0
	A11	0	0	0	0	0	0	0	0	0	5	475	0	0	0	0	0	0	0	0
	A12	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
	A13	0	0	0	0	0	0	0	0	0	0	0	0	479	1	0	0	0	0	0
	A14	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0	0
	A15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0	0
	A16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0	0
	A17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0	0
	A18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480	0
	A19	0	0	0	0	0	0	0	5	0	0	0	0	1	0	0	0	0	0	474

Table 4.9: (a) Number of correctly and incorrectly classified motions out of 480 for SVMs (P-fold cross validation, 98.8%); (b) same for ANN (P-fold cross validation, 96.2%).

		classified	
		correct	incorrect
true	A1	480	0
	A2	479	1
	A3	478	2
	A4	477	3
	A5	480	0
	A6	478	2
	A7	445	35
	A8	430	50
	A9	476	4
	A10	479	1
	A11	479	1
	A12	480	0
	A13	477	3
	A14	480	0
	A15	480	0
	A16	479	1
	A17	480	0
	A18	480	0
	A19	473	7

(a)

		classified	
		correct	incorrect
true	A1	471	9
	A2	454	26
	A3	475	5
	A4	478	2
	A5	473	7
	A6	463	17
	A7	421	59
	A8	388	92
	A9	457	23
	A10	471	9
	A11	464	16
	A12	479	1
	A13	467	13
	A14	470	10
	A15	475	5
	A16	472	8
	A17	479	1
	A18	478	2
	A19	461	19

(b)

As an alternative to PCA, we considered using SFFS and SBFS algorithms [113] that use the extracted features themselves instead of linear combinations of features. Since SFFS performed better than SBFS in general, here we report the results of SFFS that adds features one at a time to the selected feature set such that the classification performance is maximized. This method is a greedy algorithm for finding the most discriminative features, and is computationally costly. For this reason, we employ this method only for BDM, LSM, and k-NN classifiers. The selected features and the corresponding correct classification rates are presented in order in Table 4.10. The algorithm is run several times and the run with the most frequently selected features is shown in the table. As an example, the scatter plots of the first three selected features are shown

pairwise in Figure 4.5 for the BDM method.

Table 4.10: First five features selected by SFFS using BDM, LSM, and k-NN (T: torso, RA: right arm, LA: left arm, RL: right leg, LL: left leg).

BDM			
feature	loc.	sensor	%
mean	LL	x-acc	33.1
DFT pk 5	RL	y-mag	57.5
max	LL	y-mag	74.8
max	T	x-acc	86.0
mean	RL	y-acc	92.0

LSM			
feature	loc.	sensor	%
min	RL	x-acc	40.0
DFT pk 3	T	x-gyro	59.0
min	RA	x-acc	70.4
max	RL	x-acc	76.0
max	LL	z-acc	79.6

k-NN			
feature	loc.	sensor	%
max	LL	x-mag	47.2
mean	RL	z-mag	84.9
mean	RL	y-mag	92.4
max	T	x-mag	94.7
min	RL	x-mag	96.0

Based on Table 4.10, it can be concluded that features of magnetometer and accelerometer signals recorded on the legs are more discriminative in general. Furthermore, time-domain features are selected more often than frequency-domain features, as also confirmed in a previous study [125]. For the first five features, the classification rates of the k-NN method are higher than BDM and LSM. However, when about 10 features are selected, both the BDM and k-NN methods achieve above 95% correct classification rate. In fact, in most runs, the correct classification rate is around 99%. We note that since feature selection is performed

Table 4.11: Correct classification percentages using the first five features obtained by PCA using BDM, LSM, and k-NN.

no. of features	BDM	LSM	k-NN
1	38.4	36.2	34.9
2	52.7	47.1	56.8
3	75.8	67.0	84.3
4	84.1	73.9	90.5
5	90.0	78.0	94.9

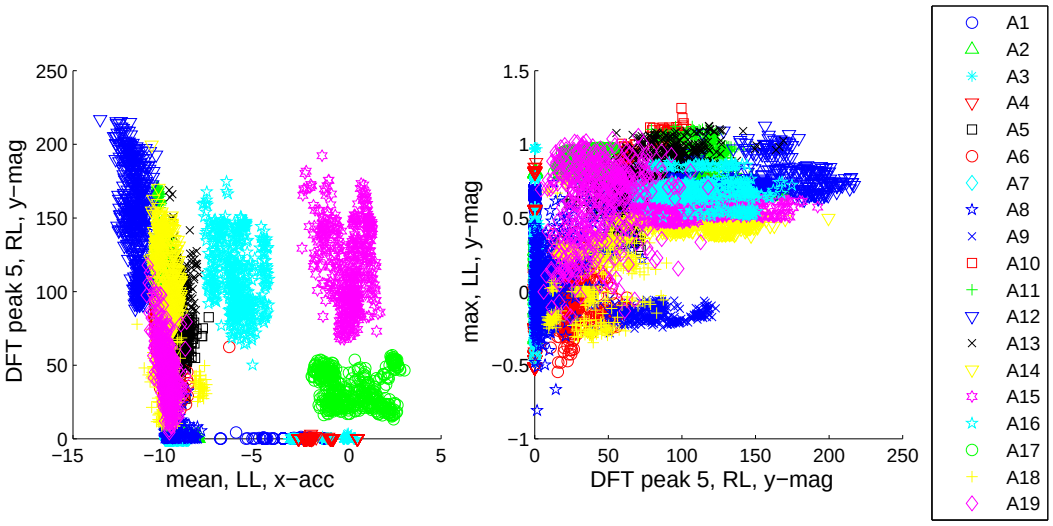


Figure 4.5: Scatter plots of the first three features selected using BDM and SFFS.

sequentially in SFFS, these features may not be the optimal subsets of all features considered together. One should consider all subsets of the total number of features to determine the optimal subsets with a certain number of features. Obviously, this is a very time-consuming process.

Table 4.11 gives the results of BDM, LSM, and k-NN classifiers when up to first five features selected by PCA are used. Comparing with Table 4.10, it can be observed that SFFS gives better results, especially for the first few selected features. While the SFFS algorithm tries to maximize the correct classification rate, PCA captures the features with largest variances in the data by making

a transformation into principal directions. The difference in performance of the two feature reduction techniques becomes smaller as more features are added to the set.

4.5.3.1 Receiver Operating Characteristics

To determine which activities can be distinguished easily, we employ the receiver operating characteristic (ROC) curves of some of the classifiers [113]. For a specific activity, we consider the instances belonging to that activity as positive instances, and all other instances as negative instances. Then, by setting a decision threshold or criterion for a classifier, the *true positive rate* (TPR) (the ratio of the true positives to the total positives) and the *false positive rate* (FPR) (the ratio of the false positives to the total negatives) can be calculated. Varying the decision threshold over an interval, a set of TPRs and the corresponding FPRs are obtained and plotted as a ROC curve.

Figure 4.6 depicts the ROC curves for BDM, LSM, k-NN, and ANN classifiers as examples. In BDM and k-NN, the decision threshold is chosen as the posterior probability. For BDM, the posterior probability is calculated using the Bayes' rule. For k-NN, it is estimated by the ratio $\frac{k_i+1}{k+c}$, where $k = 7$ for our case, $c = 19$ is the total number of classes, and k_i is the number of training vectors that belong to class i, out of the k nearest neighbors. This gives smoother estimates than using binary probabilities. In LSM, the decision threshold is chosen as the distance between a test vector and the average reference vector of each class; and in ANN, the norm of the difference between the desired and actual outputs. Since there are 19 activities, the number of positive instances of each class is much less than the number of negative instances. Consequently, the FPRs are expected to be low and therefore, we plot the FPR in the logarithmic scale for better visualization. It can be observed in Figure 4.6 that the sensitivity of BDM classifier is the highest. A test vector from classes A2, A7, or A8 is less likely to be correctly classified than a test vector belonging to one of the other classes. It is also confirmed by the confusion matrices that these are the most confused activities. For the LSM classifier, the same can be said for A13 and A14, as

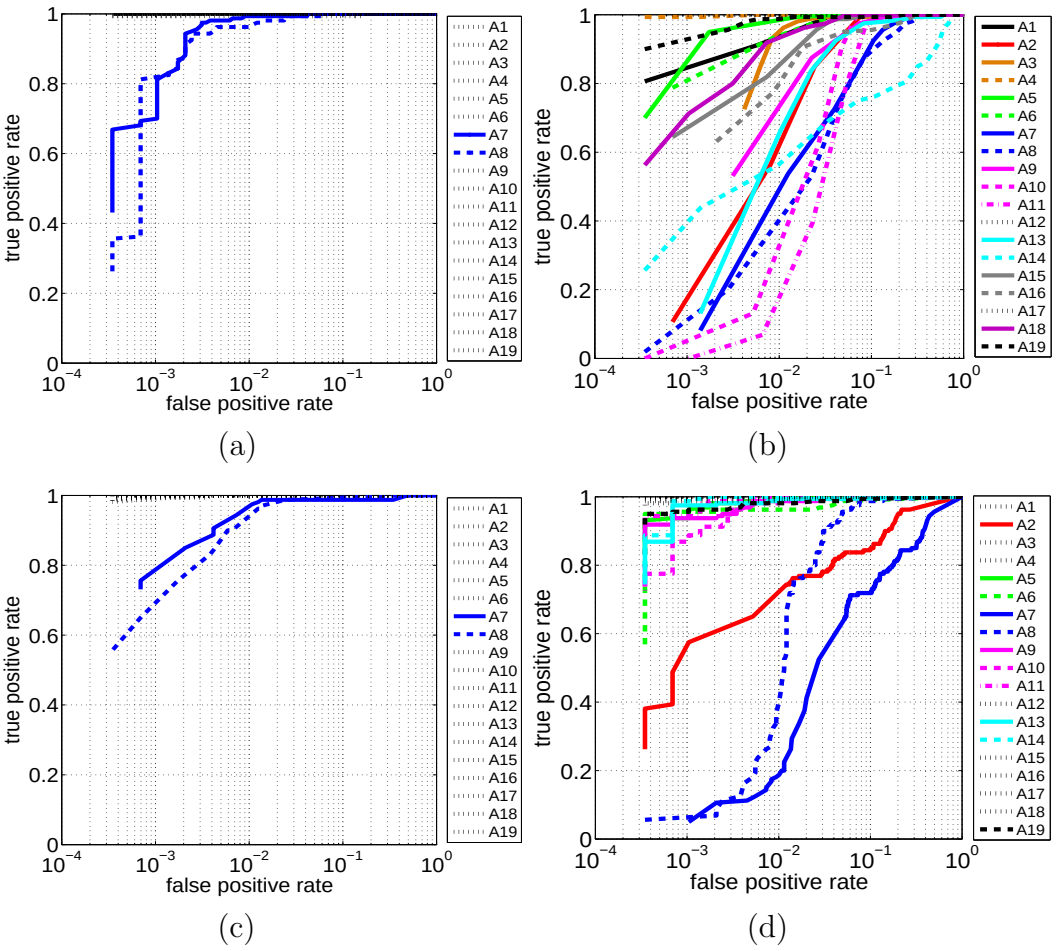


Figure 4.6: ROC curves for (a) BDM, (b) LSM, (c) k-NN, and (d) ANN using RRSS cross validation. In parts (a) and (c), activities other than A7 and A8 are all represented by dotted horizontal lines at the top where the TPR equals one.

well as for A9, A10, and A11 where the FPRs for a given TPR are rather high. Despite this, for a tolerable FPR such as, say, 0.1, the TPR for LSM and ANN still remains above 0.75.

4.5.3.2 Processing Times

We also compared the classification techniques given above based on their computational costs. Pre-processing and classification times are calculated with MATLAB version 7.0.4, on a desktop computer with AMD Athlon 64 X2 dual core

processor at 2.2 GHz and 2.00 GB of RAM, running Microsoft Windows XP Professional operating system. Pre-processing/training, storage requirements, and processing times of the different techniques are tabulated in Table 4.12. The pre-processing time of BDM is used for estimating the mean vector, covariance matrix, and the CCPDFs that need to be stored for the test stage. In RBA, the pre-processing phase involves extracting the rules based on the training data. Once the rules are available, the vectors need not be stored and any test vector can be classified using the RBA. In LSM and DTW_1, the averages of the training vectors for each class need to be stored for the test phase. Note that the pre-processing times of these two methods are exactly equal. For k-NN and DTW_2, all training vectors need to be stored. For SVM, the SVM models constructed in the training phase need to be stored for the test phase. For ANN, the structure of the trained network and the connection weights need to be saved for testing. ANN and SVM require the longest training time and SVM also has considerable storage requirements. These are followed by RBA, BDM, and LSM (same as DTW_1). The k-NN and DTW_2 methods do not require any pre-processing.

The processing times for classifying a single feature vector are given in the same table. The classification time for ANN is the smallest, followed by LSM, RBA, BDM, SVM, DTW_1, and DTW_2 or k-NN methods. The latter two take the longest amount of classification time because of the nature of the classifiers and also because a comparison should be made with every training vector.

Table 4.12: Pre-processing and training times, storage requirements, and processing times of the classification methods. The processing times are given for classifying a single feature vector.

method	pre-processing/training time (ms)			storage requirements	processing time (ms)		
	RRSS	P-fold	L1O		RRSS	P-fold	L1O
BDM	28.98	28.62	24.70	mean, covariance, CCPDF	4.56	5.70	5.33
RBA	2,514.21	3,874.78	3,400.14	rules	0.64	0.95	0.84
LSM	6.77	9.92	5.42	average of training vectors for each class	0.25	0.24	0.21
k-NN	–	–	–	all training vectors	101.32	351.22	187.32
DTW_1	6.77	9.92	5.42	average of training vectors for each class	86.26	86.22	85.57
DTW_2	–	–	–	all training vectors	116.57	155.81	153.25
SVM	7,368.17	13,287.85	10,098.61	SVM models	19.49	7.24	8.02
ANN	290,815	228,278	214,267	network structure and connection weights	0.06	0.06	0.06

4.5.3.3 Sensor Combinations

The performances of the classifiers do not change considerably if one or more of the sensors fail because of power cut or any other malfunction. In Table 4.13, we present classification results with a reduced number of sensors. For example, using only the sensor on the torso, correct classification rate of 96.6% can be achieved with BDM. It can also be observed that the sensors on the legs are more discriminative than the sensors on the arms, with the left leg being more discriminative for most of the classification methods. Most of the activities performed in this study involve quasi-symmetric movement of the body with respect to the sagittal plane. That is, left and right sides of the body follow basically the same movement patterns that are either stationary (sitting, standing), in phase (jumping, rowing), or out of phase (walking, running, cycling). Exceptions are basketball and lying on the right side activities. The cycling activities involve symmetric out-of-phase movement of the legs, but not the arms. The sensor locations are symmetric as well, thus one can expect redundancy in the information acquired by the sensors. However, this redundancy can be exploited in case of the failure of one or more sensors. This can also be observed in Table 4.13. The correct classification rates using the left and right side sensors are close to each other, which means that if a sensor on either side fails, its symmetric counterpart on the other side will compensate for that sensor. The torso sensor does not have a symmetric counterpart; however, its failure would result in only a slight decrease in the correct classification rate as can be seen in the table.

Table 4.13: All possible sensor combinations and the corresponding correct classification rates for some of the methods using P-fold cross validation (T: torso, RA: right arm, LA: left arm, RL: right leg, LL: left leg).

sensors used	BDM	RBA	LSM	k-NN	DTW_1	DTW_2	SVM		BDM	RBA	LSM	k-NN	DTW_1	DTW_2	SVM
–	–	–	–	–	–	–	–	+T	96.6	67.5	79.0	92.8	62.5	92.9	93.4
RA	94.5	56.8	72.5	88.4	57.1	87.5	90.6	+T	98.1	74.9	84.0	95.8	65.0	95.7	97.7
LA	93.7	59.6	75.3	87.8	47.8	84.4	91.0	+T	98.1	69.5	87.2	95.6	67.2	94.6	97.4
RL	97.3	66.5	82.3	91.0	70.2	87.3	93.8	+T	98.4	80.8	85.4	97.2	76.3	97.6	98.0
LL	94.8	79.8	79.1	96.7	74.6	97.5	96.0	+T	98.4	80.7	85.4	97.3	75.9	97.5	97.9
RA+LA	97.6	68.8	83.4	95.5	61.6	94.8	97.0	+T	98.5	76.8	86.8	97.5	74.3	97.4	98.3
RL+LL	98.8	78.2	84.0	96.6	75.8	95.6	97.8	+T	99.0	83.5	86.3	97.7	79.5	97.7	98.4
RA+RL	98.0	75.6	84.3	96.9	73.2	95.9	97.5	+T	98.8	79.4	87.6	98.0	77.2	97.7	98.5
LA+LL	98.4	76.1	85.6	95.9	72.4	94.9	97.9	+T	98.8	80.0	88.1	97.2	76.4	97.0	98.2
RA+LL	98.5	77.0	83.6	96.3	73.9	96.4	98.0	+T	98.9	80.6	87.2	97.6	80.2	97.8	98.5
LA+RL	97.8	72.5	86.1	95.9	72.7	94.5	97.1	+T	98.7	77.4	88.9	97.5	76.3	97.2	98.4
RA+LA+RL	98.7	77.0	87.1	97.7	76.4	97.3	98.4	+T	98.9	79.2	89.0	98.3	79.4	98.3	98.7
RA+LA+LL	98.7	79.0	86.7	97.8	77.3	97.3	98.5	+T	99.0	81.6	88.9	98.4	80.7	98.2	98.6
RA+RL+LL	99.0	81.5	86.1	98.1	78.7	97.7	98.7	+T	99.1	82.4	88.3	98.4	82.3	98.6	98.8
LA+RL+LL	98.9	80.7	87.2	97.6	75.9	97.4	98.4	+T	99.0	83.7	89.3	98.2	78.5	98.2	98.5
RA+LA+RL+LL	99.0	82.5	88.0	98.3	79.0	98.4	98.8	+T	99.2	84.5	89.6	98.7	83.2	98.5	98.8

4.5.3.4 Subject Combinations

Among the classification techniques we considered and implemented, when RRSS and P-fold cross-validation techniques are used, BDM gives the highest classification rate, followed by SVM and k-NN. SVM and k-NN methods give the highest classification rates also with subject-based L1O cross validation, but the performance of BDM is not as good. To further compare these three methods, we calculated the correct classification rates using data from subsets of the subjects. All possible subject combinations are considered exhaustively, and those that result in the highest correct classification rates are reported in Tables 4.14 and 4.15, using P-fold and subject-based L1O cross validation, respectively. Note that for L1O cross validation (Table 4.15), the results of a single subject cannot be provided. This is because partitioning in this method is subject-based and requires the availability of data from at least two subjects.

Table 4.14: Best combinations of the subjects and correct classification rates using P-fold cross validation.

BDM	
subject no.	%
5	99.0
2,5	99.6
2,5,6	99.5
1,2,4,6	99.5
2,4,5,6,7	99.4
1,2,3,5,6,7	99.4
1,2,3,4,5,6,7	99.2

k-NN	
subject no.	%
1	98.9
1,2	99.4
1,2,5	99.3
1,2,5,6	99.1
1,2,3,5,6	99.0
1,2,3,4,5,6	98.9
1,2,3,4,5,6,8	98.8

SVM	
subject no.	%
5	98.5
1,2	99.4
1,2,5	99.4
1,2,5,6	99.3
1,2,5,6,7	99.1
1,2,3,4,5,6	99.0
1,2,3,4,5,6,7	98.9

When P-fold cross validation is used, the performances of all three methods are comparable (Table 4.14). Using data from more than two subjects causes a slight decrease in performance which is expected. When L1O cross validation is used (Table 4.15), the classification rates are lower than those in Table 4.14 and it can be also observed that k-NN and SVM are superior to BDM, regardless of the number of subjects used. This means that although data from multiple

Table 4.15: Best combinations of the subjects and correct classification rates using subject-based L1O.

BDM	
subject no.	%
1,7	64.5
1,2,7	73.2
1,2,6,7	75.9
1,2,3,6,7	75.6
1,2,3,5,6,7	76.4
2,3,4,5,6,7,8	76.8

k-NN	
subject no.	%
2,6	87.0
2,4,6	90.2
2,4,6,7	89.8
1,2,4,6,7	89.3
1,2,4,6,7,8	88.6
1,2,4,5,6,7,8	88.1

SVM	
subject no.	%
2,6	65.7
2,6,7	76.6
1,2,6,7	80.0
1,2,5,6,7	82.0
1,2,4,5,6,7	85.0
1,2,4,5,6,7,8	86.9

subjects can be well-approximated by a multi-variate Gaussian distribution, the parameters of the distribution, when calculated by excluding one of the subjects, cannot represent the data of the excluded subject sufficiently well. The performance of BDM and SVM tend to increase with increasing number of subjects (Table 4.15), indicating that these classifiers generalize better as data from more subjects are included. In the case of BDM, the data may be slowly converging to a multi-variate Gaussian distribution as the number of subjects is increased. In k-NN, there is a slight decrease in performance after the addition of the fourth subject.

4.6 Discussion

Given its very high correct classification rate and relatively small pre-processing and classification times and storage requirements, it can be concluded that BDM is superior to the other classification techniques we considered for the given classification problem. This result supports the idea that the distribution of the activities in the feature space can be well approximated by multi-variate Gaussian distributions. The low processing and storage requirements of the BDM method make it a strong candidate for similar classification problems.

SVM, although very accurate, requires a considerable amount of training time to construct the SVM models. Its storage requirements and processing time fall in the middle. The k-NN method is also very accurate, with zero pre-processing time but its processing time is one of the two largest. For real-time applications, LSM could also be a suitable choice because it is faster than BDM at the expense of a 10% lower correct classification rate. The ANN requires considerable training time but once it is trained and the connection weights are stored, classification is done very rapidly.

This work can serve as a guideline in designing context-aware wearable systems that involve recognition of daily activities of an individual. Many context-aware wearable systems are designed to be used by a single person. This work shows that for such applications, a simple quadratic classifier such as BDM is sufficient with almost perfect performance. If such a system is to be used by more than one person, providing training data from all the users is expected to result in above 95% performance. The values in the P-fold column in Table 4.2 can be interpreted as the expected correct classification rates of the classifiers for this case. However, it is evident that if, for some reason, training data from an individual are not available, the correct classification rates are expected to drop to the values given in the L1O column in Table 4.2. In this case, one must resort to more complex classifiers such as k-NN and SVM that require more computational resources.

Chapter 5

Self-Contained Pedestrian Dead Reckoning Using Body-Worn Inertial/Magnetic Sensors

5.1 Introduction

Dead reckoning is the process of estimating the current position of a moving entity using the position estimate (or fix) calculated at previous time instants and the velocity (or speed) estimate at the current time instant. It can also be used to predict the future position by projecting the current known position and speed to a future instant [126]. Since the past position estimates are projected through time to obtain new estimates in dead reckoning, position errors accumulate over time. Because of this cumulative error propagation, dead-reckoning estimates are unreliable if calculated over long periods of time. Hence, dead reckoning is seldom used alone in practice, and is often combined with other types of position sensing to improve position accuracy.

Historically, dead reckoning has been used in ship navigation for centuries. Reference [126] explains its use in ship navigation in detail. It has been used in air navigation since the beginning of 1900s; a thorough survey appears in

[127] and [128]. A survey on the positioning and navigation methods for vehicles appears in [129]. Dead reckoning is employed in mobile robotics through the use of odometry [130] and/or inertial navigation systems.

Inertial navigation systems (INS) [53] can be used for both indoor and outdoor positioning and navigation. Fundamentally, gyroscopes provide angular rate information, and accelerometers provide velocity rate information. Although the rate information is reliable over long periods of time, it must be integrated to provide position, orientation, and velocity estimates. Thus, even very small errors in the rate information provided by inertial sensors cause an unbounded growth in the error of the integrated measurements. As a consequence, an INS by itself is characterized by position errors that grow with time and distance, usually referred to as the "drift error." One way of overcoming this problem is to periodically reset inertial sensors with external absolute sensing mechanisms and to eliminate this accumulated error. Thus, in most cases, data from an INS must be integrated with absolute location-sensing mechanisms to provide useful information about position.

An inertial measurement unit (IMU) consists of orthogonally-mounted accelerometers and gyroscopes in three spatial directions. If the IMU is directly mounted on the moving object, the system is called a strap-down INS [53]. The IMU gives three acceleration and three angular velocity (or angular rate) outputs in the object coordinate frame. A basic block diagram of a strap-down INS is given in Figure 5.1. In order to estimate the orientation (or attitude) of the moving object, the gyroscope outputs should be integrated. Then, using the estimated orientation, accelerometer outputs should be transformed to the Earth coordinate frame. The acceleration values in the Earth coordinate frame are integrated twice to get the position. Because of the integration operations involved in the position calculation, any error in the sensor outputs accumulates in the position output, causing a rapid drift in both the gyroscope and accelerometer outputs. Thus, the reliability of position estimates decreases with time. For example, a constant bias in the gyroscope will cause an error in the position that grows proportional to the cube of time, and a constant bias in the accelerometer will cause an error that grows proportional to the square of time [55]. For this

reason, inertial sensors are usually used in conjunction with other sensing systems that provide absolute external reference information.

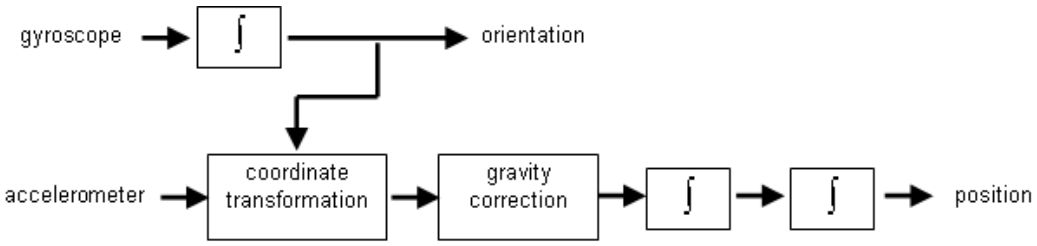


Figure 5.1: Strap-down INS integration.

One application of INS is in pedestrian dead reckoning (PDR). PDR systems are generally used in GPS-denied environments such as inside buildings, tunnels, underground, or dense forests and around tall buildings in urban areas where GPS data are not accurate or available. References [131] and [132] provide brief surveys on PDR systems. Such systems are usually developed for security personnel and emergency responders [133]. Unlike land vehicles and robots, a method called "zero velocity update" (ZUPT) enables the stand-alone usage of INSs on pedestrians, without any external reference sensor. The ZUPT method exploits the fact that during walking, the velocity of the foot is zero at some time interval during the stance phase (see Section 5.3.1). If this time interval is correctly detected, the drift in the velocities calculated in strap-down integration can be reset to zero and the drift in one step will not be carried over to the next step. As an alternative, instead of directly resetting the velocities to zero, this information can be used as a measurement in a Kalman filter [134, 135]. In [133], the ZUPT method is used to estimate the distance traveled and a high-grade gyroscope is employed to estimate the orientation. Alternative methods for orientation estimation also exist in the literature. In [136], a Kalman filter is used to estimate the orientation. Accelerometers and magnetometers can also be used interchangeably with gyroscopes depending on whether the body is in motion or not [137]. Another approach is to use the orientation output of a commercially available sensor module that integrates accelerometer, gyroscope, and

magnetometer measurements [104]. An extensive survey on orientation estimation methods using body-worn sensors appears in [138]. Heuristic methods that exploit the usual walking patterns of people can also be applied for drift reduction [66] and elimination [139] in gyroscopes.

In order to apply the ZUPT method, correct detection of gait events such as the stepping instants and correct estimation of gait parameters such as stride length are crucial for many PDR systems. This detection can be performed using only inertial sensors as in [140]. Zero-velocity detection algorithms using inertial sensors are compared in [141, 142]. In [143], an external pressure sensor is used to detect the steps. It is also possible to perform activity recognition with inertial sensors to detect the stepping instants and estimate the stride length [144, 145].

Integrating external reference sensors with PDR systems is also common in the literature. In [136, 146], a shoe-mounted inertial/magnetic system is used together with a quaternion-based Extended Kalman filter (EKF) to estimate the 3-D path traveled by a walking person. Magnetic sensors are used in the initialization of the EKF. Reference [147] combines dead reckoning with GPS in outdoor environments. For indoor environments, WiFi fingerprinting method is used for localization. Reference [148] uses the GPS data for error correction. The pedestrian trajectory is estimated using a PDR system and a wireless sensor network in [131].

Another alternative for integrating external references is map matching. If a map of the environment is available, this information can be used to provide drift error correction. In [149], this idea is applied in an outdoor environment, combined with a heuristic drift elimination procedure described in [139]. In indoor environments, activity-based map matching can be used [145]. This idea exploits the fact that the activity context of the pedestrian gives information about the location. For example, if the pedestrian is ascending stairs, most locations on an indoor map can be ruled out, improving the position estimate. Here, we follow a similar approach.

In this study, we perform pedestrian localization using five inertial and magnetic sensor units worn on the body. Localization is performed simultaneously

with activity recognition, where activity recognition cues are used as position updates in order to correct the drift errors of inertial sensors. Apart from being inherent in inertial sensors, drift errors and offsets in body-worn systems can also arise from initial misplacement, occasional slips from the initial position and orientation during operation, or loose mounting on the body. Even though the initial errors are expected to be small, they are accumulated and result in larger errors over long periods of time.

To our knowledge, these issues have not been addressed before in the literature. We demonstrate that using activity recognition cues and a given map of the environment, these errors can be reduced considerably and accurate localization can be achieved without using any external reference sensor. In practice, the proposed method can be used in applications where a map is available and GPS data is not reliable or not available at all (e.g., underground mines, indoor areas, and urban outdoor areas with tall buildings). We would like to note that although here we use activity recognition information to improve localization performance, the converse is also possible, i.e., localization information and a map can improve the activity recognition performance. However, in our previous studies, we observed that activity recognition with high accuracy can already be achieved using proper signal processing and pattern recognition techniques [5].

We consider three different activities in our experiments; namely walking, standing, and turning. We also consider ascending/descending stairs activity in a 3-D localization example. We assume that a map of the environment is available, such that the switches between these activities usually correspond to multiple locations on the map. For example, in an indoor environment, switching from walking to turning activity might correspond to the end of a corridor or to the front of a room, whereas switching from walking to standing activity might correspond to a location in front of an elevator. Therefore, the switches between activities correspond to several discrete locations in the environment. If one can detect the switches between activities correctly, it is possible to use the corresponding position information in order to correct the drift in the position.

5.2 Experiments

A total of 11 experiments are performed in this study, in two different environments. The first set of experiments is performed outdoors on a straight line of 66 m length. The line is divided into four segments of equal length, and the endpoints of each segment are marked with a + or a × sign. The path is illustrated in Figure 5.2.

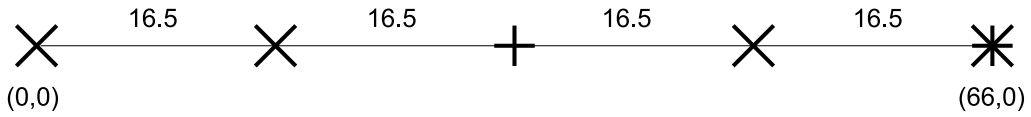


Figure 5.2: The path followed in first four experiments (all dimensions in m).

A coordinate system is assigned in this environment such that the line coincides with the x-axis. The origin of the coordinate system is at the leftmost point of the line. The × marks indicate possible locations to perform the "walking-to-standing" (WS) activity switch, and the + marks indicate the locations to perform the "walking-to-turning" (WT) activity switch. Note that the point (66,0) is marked with both symbols, which means that it is possible to perform both WS and WT activity switches at this location. In this outdoor environment, four experiments are performed:

1. start from point (0,0), stop at (16.5,0), stop at (49.5,0), stop at (66,0),
2. start from point (0,0), stop at (16.5,0), turn back at (33,0), stop at (16.5,0), stop at (0,0),
3. start from point (0,0), stop at (16.5,0), stop at (49.5,0), turn back at (66,0), stop at (49.5,0), stop at (16.5,0), stop at (0,0),
4. start from point (0,0), stop at (49.5,0), turn back at (66,0), stop at (16.5,0), stop at (0,0).

Note that it is not required to stop at every × mark, or turn back at every + mark, but these marks indicate that there is some nonzero probability that these events will occur at that location.

The second environment is set up at the sports hall of Bilkent University. The subjects are required to walk on lines drawn on the floor. The map of this indoor environment is shown in Figure 5.3. Similar to the first setup, the × marks indicate possible locations to perform the standing activity. Each corner in the figure indicates a possible location to perform the turning activity.

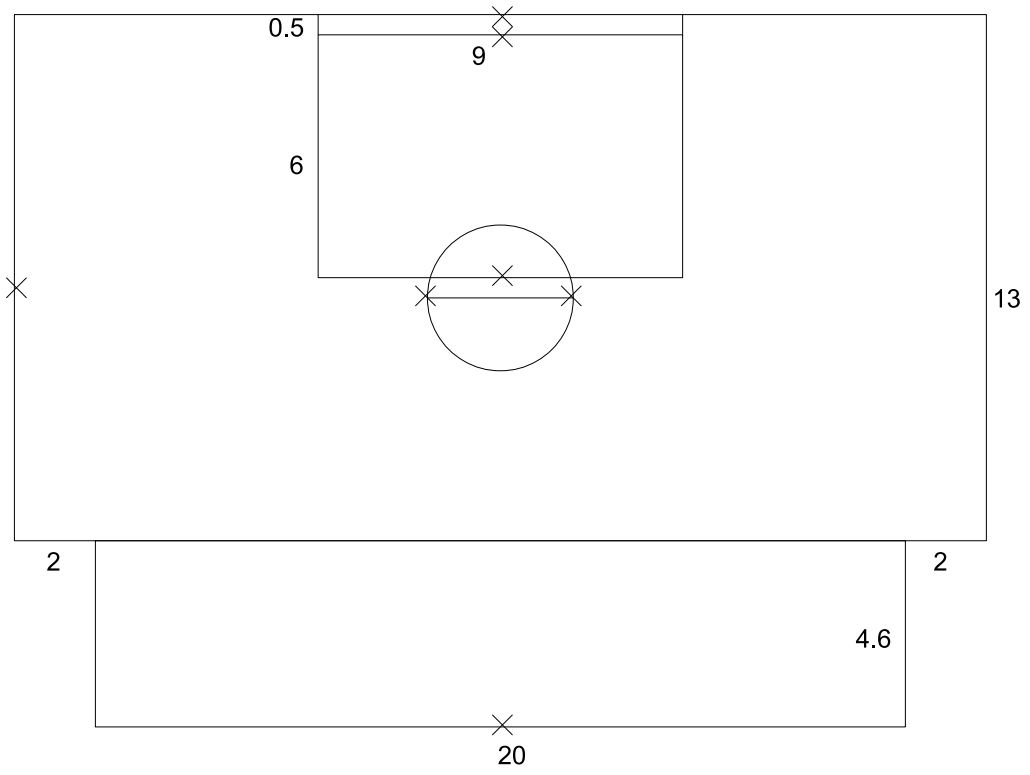


Figure 5.3: The path followed in the second set of experiments (all dimensions in m).

The seven experiments performed in this environment are as follows:

5. walk for three laps on a rectangle of size 24 m × 13 m,

6. walk for three laps on a rectangle of size 24 m × 13 m, stopping at the midpoint of the longer side,
7. walk for three laps on a rectangle of size 9 m × 6 m,
8. walk for three laps on a rectangle of size 9 m × 6 m, stopping at the midpoint of the longer side,
9. walk for three laps on a circle of diameter 3.6 m, stopping each time at the endpoints of the diameter,
10. walk for one lap on a rectilinear polygon,
11. walk for one lap on a rectilinear polygon, stopping at three different points.

The total path lengths of these experiments are tabulated in Table 5.1.

Table 5.1: Total path lengths of the experiments.

experiment no.	path length (m)
1	66
2	66
3	132
4	132
5	222
6	222
7	90
8	90
9	33.9
10	96.2
11	96.2

These 11 experiments are performed by four male and four female subjects, whose ages, heights, and weights are presented in Table 5.2. The subjects wear five MTx 3-DOF orientation trackers (Figure 3.1), manufactured by Xsens Technologies [54].

In addition to the kinematic outputs described in detail in Chapter 3, each sensor unit also has a built-in Kalman filter that outputs the orientation of the

Table 5.2: Profiles of the eight subjects.

subject no.	gender	age	height (cm)	weight (kg)
S1	f	32	158	45
S2	f	34	161	51
S3	m	25	180	79
S4	f	22	166	47
S5	f	24	178	60
S6	m	33	175	95
S7	m	22	187	75
S8	m	25	182	75

sensor with respect to the global coordinate frame. In the default configuration, the global coordinate frame has its z-axis pointing upward along the vertical ($-g$ direction) and x-axis pointing towards the magnetic north. The y-axis completes the right-handed coordinate system (Figure 5.4). Three orientation output modes can be used for the output: direction cosine matrix, quaternion, and Euler angles. In this study, we use the quaternion output mode.

The sensors are placed on five different positions on the subject's body. Two of the customized ($\pm 18g$ accelerometer range) sensor units are placed on the feet, the remaining customized unit is placed on the subject's chest, and the two standard ($\pm 5g$ accelerometer range) units are placed on the sides of the knees (right side of the right knee and left side of the left knee). The customized units are used on the feet in order to avoid any saturation in the sensor outputs, because feet accelerations are expected to be larger than knee accelerations (up to $\pm 9g$ in our experiments).

Before starting the experiments, an "alignment reset" is performed on the sensors in order to reset the coordinate frames such that the initial orientation transformation is the unit operator (that is, the initial orientation output is $\mathbf{I}_{3\times 3}$ in the direction cosine matrix mode, $q = 1$ in the quaternion output mode, or zero Euler angles in the Euler angle output mode). In this case, all orientation outputs during the experiments are obtained with respect to the initial orientation of the sensor. The top views of the global and the sensor-fixed coordinate frames before

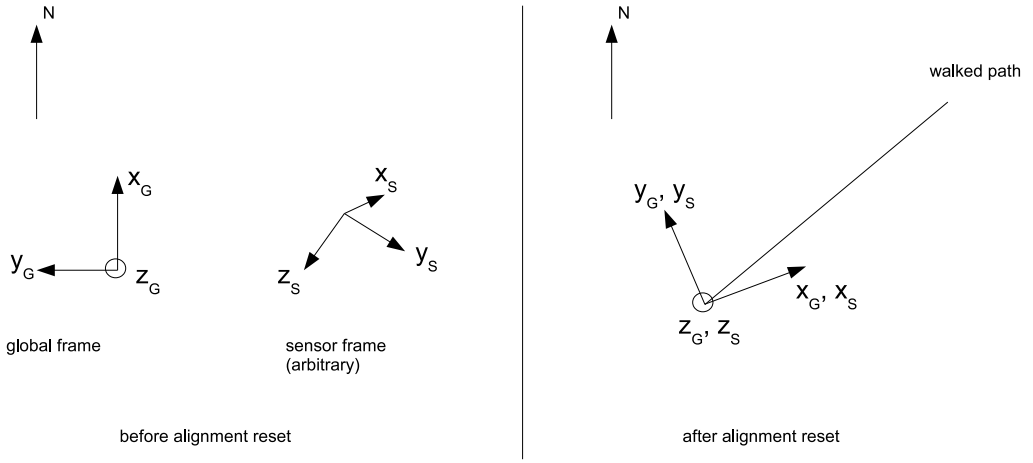


Figure 5.4: Top views of global and the sensor-fixed coordinate frames, before and after the alignment reset operation is performed.

and after the alignment reset are shown in Figure 5.4. Note that before the reset, the global frame is in the default configuration. After the reset, the global and the sensor-fixed coordinate frames coincide, and the z-axes are perpendicular to the horizontal plane.

5.3 Methodology

The MTx units provide raw acceleration, angular velocity, and magnetic field data, in addition to the orientation data that are calculated by the built-in Kalman filter. In this section, the steps used for processing these data are explained. The processing is done in two separate tracks, one of which is for activity recognition and the other is for localization. The processing for localization is done in two main steps. In the first step, the trajectories are found using well-known methods such as ZUPT, mentioned in Section 5.1. In the second step, a Kalman-filter-like state estimation procedure is employed in order to utilize the activity recognition cues and improve the results.

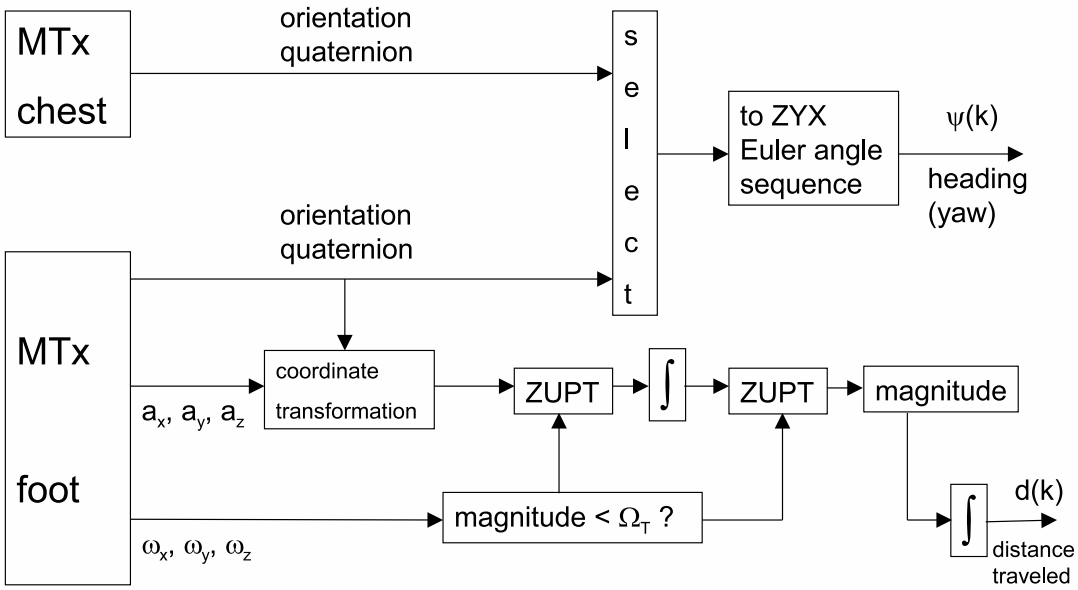


Figure 5.5: Block diagram for the first processing step.

5.3.1 Localization

In this step, we perform the regular strap-down integration procedure, using the orientation data output from the MTx sensor and zero-velocity updates. A block diagram that summarizes this step is depicted in Figure 5.5.

As shown in the diagram, calculations for the distance traveled and the heading are performed separately. To find the heading, it is possible to use the orientation output of the MTx unit either on the chest or on the feet. We use the chest sensor output directly. This unit is selected because during walking, the chest is a relatively stable reference to measure the person's heading as opposed to the feet. That is, the signals on the chest are less oscillatory than the signals acquired from other locations. The orientation output mode is set as the quaternion output mode to avoid the occurrence of any singularities possible in the Euler angle mode, even though this is unlikely for the chest. The chest sensor coordinate frame is reset to another coordinate frame at the beginning of the experiments. This reset operation is a facility of the sensor software, and after the reset, the z-axis points upwards, the x-axis points to the initial walking direction and the y

axis is assigned such that the coordinate frame is right-handed. In other words, the reset ensures that the yaw angle is initially set to zero, and is measured with respect to the vertical axis during the motion. Then the orientation data are converted to Euler angles (see Appendices A.1 and A.2). In the Euler angle domain, the yaw angle (ψ) represents the instantaneous heading. Here, it is assumed that the left and right turns performed by the subject are along the vertical axis.

In order to determine the distance traveled, the sensor signals on either foot can be used. First, using the orientation output of the sensor unit, the accelerations are transformed from the sensor coordinate frame to the local navigation coordinate frame. The local navigation coordinate frame is a translated version of the global frame, and therefore has its z-axis pointing upwards along the vertical, x-axis pointing in the magnetic north direction, and y-axis pointing along the west direction in order to end up in a right-handed coordinate frame. The transformation can simply be performed as (see Appendix A.1):

$$\mathbf{a}_G = q_{GS}\,\mathbf{a}_S\,q_{GS}^* = q_{GS}\,\mathbf{a}_G\,q_{SG} \tag{5.1}$$

where $\mathbf{a}_G$ is the acceleration vector in the local navigation frame, $\mathbf{a}_S$ is the acceleration vector in the sensor coordinate frame, and q_{GS} is the quaternion representing the orientation of the sensor coordinate frame with respect to the local navigation frame.

To determine the position from the acceleration signal, the acceleration must be integrated twice. However, because of this integration procedure, the errors in the sensor readings are accumulated, causing unbounded drift in the position.

We use the ZUPT method [133] to reduce the drift in position. When a human is walking, the motion of the leg is quasiperiodic. The collection of these motions within one period is called the gait cycle. The human gait cycle is roughly divided into two phases called the stance phase and the swing phase. The stance phase is defined as the time interval during which the foot is in contact with the ground, and the swing phase is the time interval during which the foot does not touch the ground. Stance phase takes approximately 60% of the gait cycle, as shown in Figure 5.6. During a sub-interval ΔT of the stance phase, the foot velocity and acceleration are expected to be zero. Thus, the true values of the velocity

and acceleration are known. If one can successfully detect this sub-interval, the sensor signals can be reset to zero and the error in one step will not be carried over the next step.

Figure 5.6: The human gait cycle (figure from http://www.sms.mavt.ethz.ch/research/projects/prostheses/GaitCycle).

The problem is now converted to successfully detecting the ΔT interval where the foot velocity is exactly zero. There are a number of detectors used in the literature for this purpose: acceleration moving variance detector, acceleration magnitude detector, and angular rate magnitude detector [141]. In most of the studies, the angular rate magnitude detector outperforms the others. In a recent study, another detector was proposed that gives slightly better results than the angular rate magnitude detector [141]. We use the angular velocity magnitude detector in this study because of its performance and simplicity of implementation. Using the magnitude of the angular velocity, the following binary signal is constructed:

$$I_{step}(k) = \begin{cases} 1, & |\boldsymbol{\omega}(k)| \leq \Omega_T \\ 0, & |\boldsymbol{\omega}(k)| > \Omega_T \end{cases} \tag{5.2}$$

where $|\boldsymbol{\omega}(k)| = \sqrt{\omega_x(k)^2 + \omega_y(k)^2 + \omega_z(k)^2}$ and Ω_T is a pre-set threshold value. This signal is constructed separately for the left foot and the right foot sensors. When this signal is 1, the foot is assumed to be in the stance phase, otherwise it is assumed to be in the swing phase. To eliminate possible instantaneous 0-1-0 or 1-0-1 switches in this signal, a median filter is applied to the signal. Then, the velocities and accelerations are set to zero when this signal is 1, and the

integrations in the block diagram in Figure 5.5 are performed. The resulting signal represents the distance traveled, and is denoted as $d(k)$.

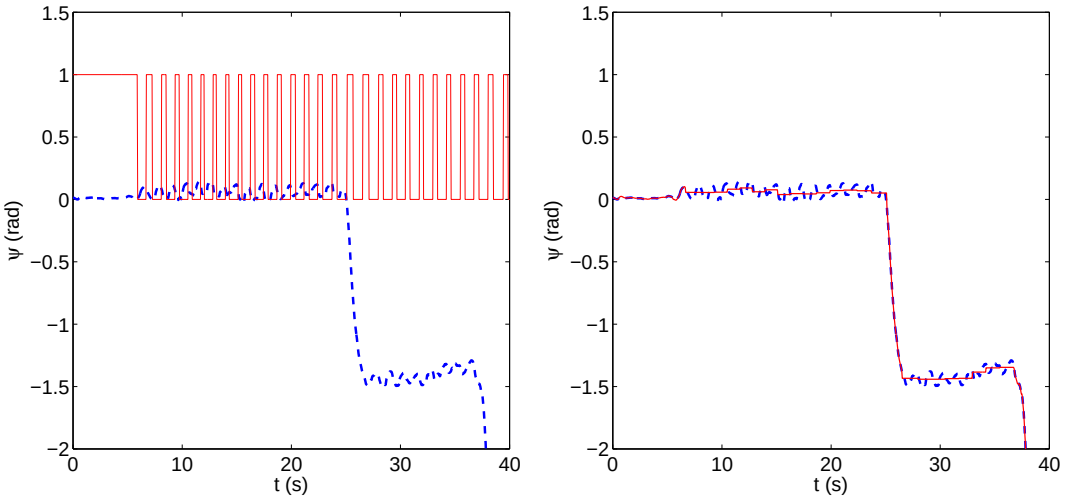


Figure 5.7: (a) Original heading signal (dashed blue line) and swing-stance phase indicator variable (solid red line) superimposed; (b) original heading signal (dashed blue line) and corrected heading signal (solid red line).

Because of the slight movement of the chest during walking, the heading signal contains ripples, as shown in blue (dashed line) in Figure 5.7(a). This signal can be smoothed using the gait phase data obtained using the aforementioned method. The I_{step} signal of the right foot is superimposed on this plot in red (solid line) in Figure 5.7(a). These data are obtained in an experiment where the subject stands for 5 s, then starts walking along a straight line, then turns 90° to the right at about $t = 25$ s and continues walking. As it can be observed in the figure, when a right step is taken ($I_{step} = 0$), the chest angle swings slightly to the left, and vice versa. To remove the ripples, the mean of the heading data between rising edges of the I_{step} signal can be calculated and replaced as a corrected heading signal. This is shown in Figure 5.7(b). In this figure, the original heading data are shown in blue (dashed line) and the corrected heading is shown in red (solid line). Obviously, this correction should be made separately for either feet depending on which foot's data are used in evaluating $d(k)$, using the I_{step} indicator for that foot. In this case, the correction is made using the right foot data. The initial ripple is not corrected (Figure 5.7(b)) because in this experiment, the subject

started walking with his/her left foot. The corrected heading data are denoted as $\psi(k)$ in the rest of this text.

After determining $d(k)$ and $\psi(k)$, the path can be reconstructed using the simple state model given below:

$$\begin{aligned} x(k) &= x(k-1) + \Delta d(k-1)\cos\left(\psi(k-1)\right) \\ y(k) &= y(k-1) + \Delta d(k-1)\sin\left(\psi(k-1)\right) \end{aligned} \tag{5.3}$$

with initial conditions $x(0)$ and $y(0)$. Here, $\Delta d(k-1) = d(k) - d(k-1)$ represents the distance traveled during the kth time step.

By defining a state vector $\boldsymbol{\xi}(k) = [x(k) \quad y(k)]^T$ and an input vector $\mathbf{u}(k) = [\Delta d(k)\cos\psi(k) \quad \Delta d(k)\sin\psi(k)]^T$, the equation becomes

$$\boldsymbol{\xi}(k) = \boldsymbol{\xi}(k-1) + \mathbf{u}(k-1) \tag{5.4}$$

with initial condition $\boldsymbol{\xi}(0) = [x(0) \quad y(0)]^T$.

The performance of the above model depends on the performances of distance and heading determination methods. In our experiments, we observed that both have errors, which causes the reconstructed path to drift over time. This drift is naturally amplified as the walking path gets longer. The most dominant cause of error is the dislocation of the mounted sensors during the experiments, especially the heading sensor. For example, a slight dislocation of the chest sensor causes a slight measurement error in the heading, which causes the path to drift drastically over long periods of walking. This could be caused by attaching the sensors to loose rather than tight clothing. Magnetic disturbance caused by the ferromagnetic materials in the environment is another source of error for the magnetometers, which directly affects the heading. Accelerometer data can be used to estimate the inclination angle, but the only external reference available for determining the heading is the magnetic field data. Furthermore, the thresholds that we use are constants, i.e., they are not selected specifically according to the subject in question. Considering the age, height, and weight variations among the subjects (Table 5.2), such errors are unavoidable. Therefore, we use cues obtained from activity recognition and perform position updates when such cues are available, in order to improve the results.

5.3.2 Activity Recognition

In our earlier work [5], we demonstrated that it is possible to distinguish between various activities using body-worn inertial and magnetic sensors and provided an extensive comparison between various classifiers. Simple Bayes classifiers with Gaussian probability density functions are sufficient to obtain over 95% correct classification rates if training data from that specific person are available. However, if such training data are not available to the classifiers, more complex classifiers such as support vector machines (SVM) can be utilized that have expected correct classification rates of about 85%. The reader is referred to [5, 71, 73, 150] for surveys of literature on activity recognition using body-worn sensors.

In this work, we consider a reduced activity set, composed of only walking, standing, and turning activities. Since these three activities are quite different from each other, using complex classifiers is not necessary. We use a rule-based classifier for these three activities, in which the following rules are applied in this particular order:

1. if the filtered heading value is above a certain threshold, the activity is classified as turning,
2. if both feet are stationary, then the activity is classified as standing,
3. if the above conditions do not hold, then the activity is classified as walking.

For the first rule, the heading signal is passed through a first-order difference filter of length 1 s and thresholded. The second rule is realized by performing an AND operation to the I_{step} indicator variables (Equation (5.2)) for the left and the right feet. Then, the time values corresponding to activity switches are determined and used for position updates, as explained in the next section.

5.3.3 Simultaneous Localization and Activity Recognition

In this section, we combine the localization results with position updates simultaneously obtained from activity recognition cues. We assume that a map of the environment is available, and some of the switches between recognized activities correspond to, in general, multiple locations on the map. That is, knowledge of an activity switch gives information about the possible positions on the map.

We consider two switches of activity (walking-to-standing (WS) and walking-to-turning (WT)). WS switch corresponds to locations marked with $\times$ on the maps given in Figures 5.2 and 5.3. WT switch corresponds to locations marked with + in Figure 5.2 and to all the corners in Figure 5.3. However, these locations are not deterministic; for example, subjects are not expected to turn at the exact corner location. Hence, locations are represented by Gaussian random vectors with means $\boldsymbol{\mu}_{ws,i}$ and $\boldsymbol{\mu}_{wt,j}$ and covariances $\mathbf{P}_{ws,i}$ and $\mathbf{P}_{wt,j}$, corresponding to WS and WT locations, respectively. Here, $i = 1, \ldots, N_{WS}$ and $j = 1, \ldots, N_{WT}$ are the indices of WS and WT switch locations.

In the previous section, we use the state Equation (5.4) in order to predict the position. To model the uncertainty in the position, consider the state equation

$$\boldsymbol{\xi}(k) = \boldsymbol{\xi}(k-1) + \mathbf{u}(k-1) + \mathbf{R}_{\psi(k)}\mathbf{w}(k) \tag{5.5}$$

with initial condition $\boldsymbol{\xi}(0)$ modeled as a Gaussian random vector with mean $\boldsymbol{\mu}_{\boldsymbol{\xi}}(0)$ and covariance matrix $\mathbf{P}_{\boldsymbol{\xi}}(0)$. Note that here $\boldsymbol{\xi}(k)$ is a random process and is different from the deterministic state equation in Equation (5.4). However, we use the same notation for simplicity. The input $\mathbf{u}(k)$ is the same as in Equation (5.4). In Equation (5.5), $\mathbf{R}_\theta$ represents rotation on the plane by an arbitrary angle θ:

$$\mathbf{R}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{5.6}$$

and $\mathbf{w}(k)$ is the process noise modeled as a white Gaussian noise with a diagonal covariance matrix $\mathbf{Q}$. In Equation (5.5), the noise vector is rotated by $\psi(k)$ at each time step k. This way, the noise introduced to the system is modeled such that it is uncorrelated (and independent, since it is Gaussian) in the current

heading direction and in the perpendicular direction to the heading. If there were no rotation, the noise would be uncorrelated in the global x and y directions, as long as the covariance matrix $\mathbf{Q}$ is diagonal. We believe that introducing this rotation matrix is a more realistic assumption for our model than considering noise in the x and y directions as being uncorrelated.

Suppose that a position update is performed at time $k = k_1$. Until the next position update, Equation (5.5) can be used to model the position. The prediction equations using this forward model are given as:

$$\hat{\boldsymbol{\xi}}_f(k|k_1) = \hat{\boldsymbol{\xi}}_f(k-1|k_1) + \mathbf{u}(k-1) \tag{5.7}$$

$$\boldsymbol{\Sigma}_f(k|k_1) = \mathbf{R}_{\Delta\psi(k)}\boldsymbol{\Sigma}_f(k-1|k_1)\mathbf{R}^T_{\Delta\psi(k)} + \mathbf{R}_{\psi(k)}\mathbf{Q}\mathbf{R}^T_{\psi(k)} \tag{5.8}$$

for $k > k_1$, where the subscript f stands for the forward model and $\Delta\psi(k) = \psi(k) - \psi(k-1)$. The initial conditions for these prediction equations depend on whether the activity switch at $k = k_1$ was a WS or a WT switch. They are given as $\hat{\boldsymbol{\xi}}_f(k_1|k_1) = \boldsymbol{\mu}_{ws,i}$ and $\boldsymbol{\Sigma}_f(k_1|k_1) = \mathbf{P}_{ws,i}$ if it was a WS switch at the ith location, or $\hat{\boldsymbol{\xi}}_f(k_1|k_1) = \boldsymbol{\mu}_{wt,j}$ and $\boldsymbol{\Sigma}_f(k_1|k_1) = \mathbf{P}_{wt,j}$ if it was a WT switch at the jth location. If no position update was performed up to time k, then $k_1 = 0$ and $\hat{\boldsymbol{\xi}}_f(0|0) = \boldsymbol{\mu}_{\boldsymbol{\xi}}(0)$ and $\boldsymbol{\Sigma}_f(0|0) = \mathbf{P}_{\boldsymbol{\xi}}(0)$.

When an activity switch is detected at $k = k_2$, we run the same system backwards in time, up to the previous activity switch and position update at $k = k_1$. The backward filter equations are:

$$\hat{\boldsymbol{\xi}}_b(k-1|k_2) = \hat{\boldsymbol{\xi}}_b(k|k_2) - \mathbf{u}(k-1) \tag{5.9}$$

$$\boldsymbol{\Sigma}_b(k-1|k_2) = \mathbf{R}_{\Delta\psi(k-1)}\boldsymbol{\Sigma}_b(k|k_2)\mathbf{R}^T_{\Delta\psi(k-1)} + \mathbf{R}_{\psi(k-1)}\mathbf{Q}\mathbf{R}^T_{\psi(k-1)} \tag{5.10}$$

for $k_1 < k \leq k_2$, where the subscript b stands for the backward model and $\Delta\psi(k-1) = \psi(k-1) - \psi(k)$. The initial conditions for these prediction equations again depend on whether the current activity switch at $k = k_2$ is a WS or a WT switch. The initial conditions are $\hat{\boldsymbol{\xi}}_b(k_2|k_2) = \boldsymbol{\mu}_{ws,i^*}$ and $\boldsymbol{\Sigma}_b(k_2|k_2) = \mathbf{P}_{ws,i^*}$ if it is a WS switch or $\hat{\boldsymbol{\xi}}_b(k_2|k_2) = \boldsymbol{\mu}_{wt,j^*}$ and $\boldsymbol{\Sigma}_b(k_2|k_2) = \mathbf{P}_{wt,j^*}$ if it is a WT switch. The subscripts i^* and j^* indicate the closest predefined WS or WT location to the forward state estimate just before the position update. More precisely,

$$i^* = \arg\min_i ||\hat{\boldsymbol{\xi}}_f(k_2|k_1) - \boldsymbol{\mu}_{ws,i}|| \tag{5.11}$$

for a WS switch, and

$$j^* = \arg\min_j ||\hat{\boldsymbol{\xi}}_f(k_2|k_1) - \boldsymbol{\mu}_{wt,j}|| \tag{5.12}$$

for a WT switch.

At this point, for each $k = k_1 + 1, \ldots, k_2 - 1$, we have two estimates available for the position. It can be proved that (see Appendix B) the linear combination of these two estimates with the minimum covariance is

$$\hat{\boldsymbol{\xi}}(k|k_1, k_2) = \boldsymbol{\Sigma}(k|k_1, k_2) \left[\boldsymbol{\Sigma}_f(k|k_1)^{-1}\hat{\boldsymbol{\xi}}_f(k|k_1) + \boldsymbol{\Sigma}_b(k|k_2)^{-1}\hat{\boldsymbol{\xi}}_b(k|k_2)\right] \tag{5.13}$$

where

$$\boldsymbol{\Sigma}(k|k_1, k_2) = \left[\boldsymbol{\Sigma}_f(k|k_1)^{-1} + \boldsymbol{\Sigma}_b(k|k_2)^{-1}\right]^{-1} \tag{5.14}$$

is the covariance of the combined estimate.

In practice, we run the forward filter in a causal manner until an activity switch is detected. When an activity switch is detected at $k = k_2$, the backward filter is run all the way back to the previous position update at $k = k_1$, and the position estimates for $k = k_1 + 1, \ldots, k_2 - 1$ are calculated. If there is no previous position update, then $k_1 = 0$. After the update, the new k_1 value is assigned as k_2.

This is illustrated in Figure 5.8, which corresponds to part of experiment 3. In the experiment, the subject starts from point (0,0) and walks in the $+x$ direction, which is shown by red (thin, solid) line and represents the ground truth. The ground truth data are obtained by measuring the distances with a standard tape measure. The green (dash-dot) line shows the reconstructed path until an activity switch is detected, which is a WS switch at point (16.5,0). The reconstructed path is in error, as shown in the figure. The average heading error is about 18°. Such high heading errors are not frequently observed in our experiments, however, this experiment is chosen in order to demonstrate the performance of combining activity recognition cues. After the WS activity switch, the backward filter should be run all the way back to the previous activity switch. Since there is no previous activity switch, the backward filter is run to the beginning, $k = 0$. This path is shown in magenta (dashed) line. Then, these estimates are combined

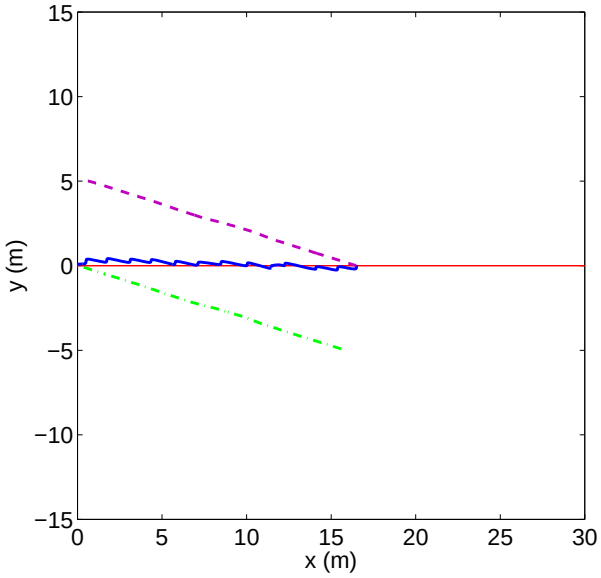


Figure 5.8: Optimal combination (solid blue line) of the forward (dash-dot green line) and backward (dashed magenta line) estimates. The thin solid red line shows the true path.

to find the improved estimate, which is shown by the blue (solid) line in the figure. The reconstruction almost coincides with the ground truth after the update, as confirmed by the figure.

The procedure stated above is applied to 11 experiments performed by eight subjects. The results are presented in the next section.

5.4 Results

In this section, we present and compare the results of the reconstruction with and without using any activity recognition cues. We calculate the error between the reconstructed path and the true path by discretizing the true path with equally spaced points on the path, and consider either path as a finite set of points. We use the symmetric error criterion between two point sets P and Q, proposed in [25] and used in Chapter 2 of this thesis (Equation (2.2)).

The parameters selected for the experiments are tabulated in Table 5.3. Each

Table 5.3: Parameter values used in the experiments.

parameter	value
Ω_T	1 rad/s
$\mathbf{P}_{\boldsymbol{\xi}}(0)$	$0.01\mathbf{I}_{2\times 2}$
$\mathbf{Q}$	$\begin{pmatrix} 0.01 & 0 \\ 0 & 0.1 \end{pmatrix}$
$\mathbf{P}_{ws,i}$	$0.01\mathbf{I}_{2\times 2}, \quad \forall i$
$\mathbf{P}_{wt,j}$	$0.04\mathbf{I}_{2\times 2}, \quad \forall j$

of the first set of experiments (1–4) is performed on the map given in Figure 5.2. For the second set of experiments (5–11), we first consider each experiment separately. That is, the possible activity switch locations are not defined for the whole map, but only for the activity switch points on the walked path. Examples of reconstructed paths are presented in Figure 5.9. In this figure, reconstructed paths without (with) activity recognition cues are shown in green and dashed (blue and solid) line. In other words, the green and dashed line shows the result of using ZUPT only. It can be observed that the reconstruction improves considerably when activity recognition cues are utilized.

Table 5.4: Error values without activity recognition updates (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
1	1.21	0.31	4.33	0.71	2.56	5.30	1.02	2.71	**2.27**
2	3.76	4.32	1.04	1.93	1.32	0.69	0.74	0.59	**1.80**
3	3.70	6.26	1.17	4.32	0.67	0.46	0.21	0.54	**2.17**
4	1.77	1.76	1.39	3.14	0.92	0.81	1.08	0.72	**1.45**
5	0.45	0.87	0.21	0.67	1.31	1.00	0.77	0.53	**0.73**
6	0.94	1.20	0.50	0.68	0.74	0.33	1.13	0.52	**0.76**
7	0.56	1.92	1.00	0.64	0.30	0.30	0.75	1.16	**0.83**
8	0.73	0.51	0.24	1.47	0.53	0.60	1.30	0.42	**0.73**
9	0.84	1.04	0.83	0.65	0.95	0.49	1.29	1.12	**0.90**
10	1.47	1.76	1.18	1.64	1.77	0.64	0.78	1.74	**1.37**
11	1.35	1.31	1.17	2.09	2.40	1.26	0.77	0.78	**1.39**
							overall average:		**1.31**

The errors between the true path and the reconstructed path without and

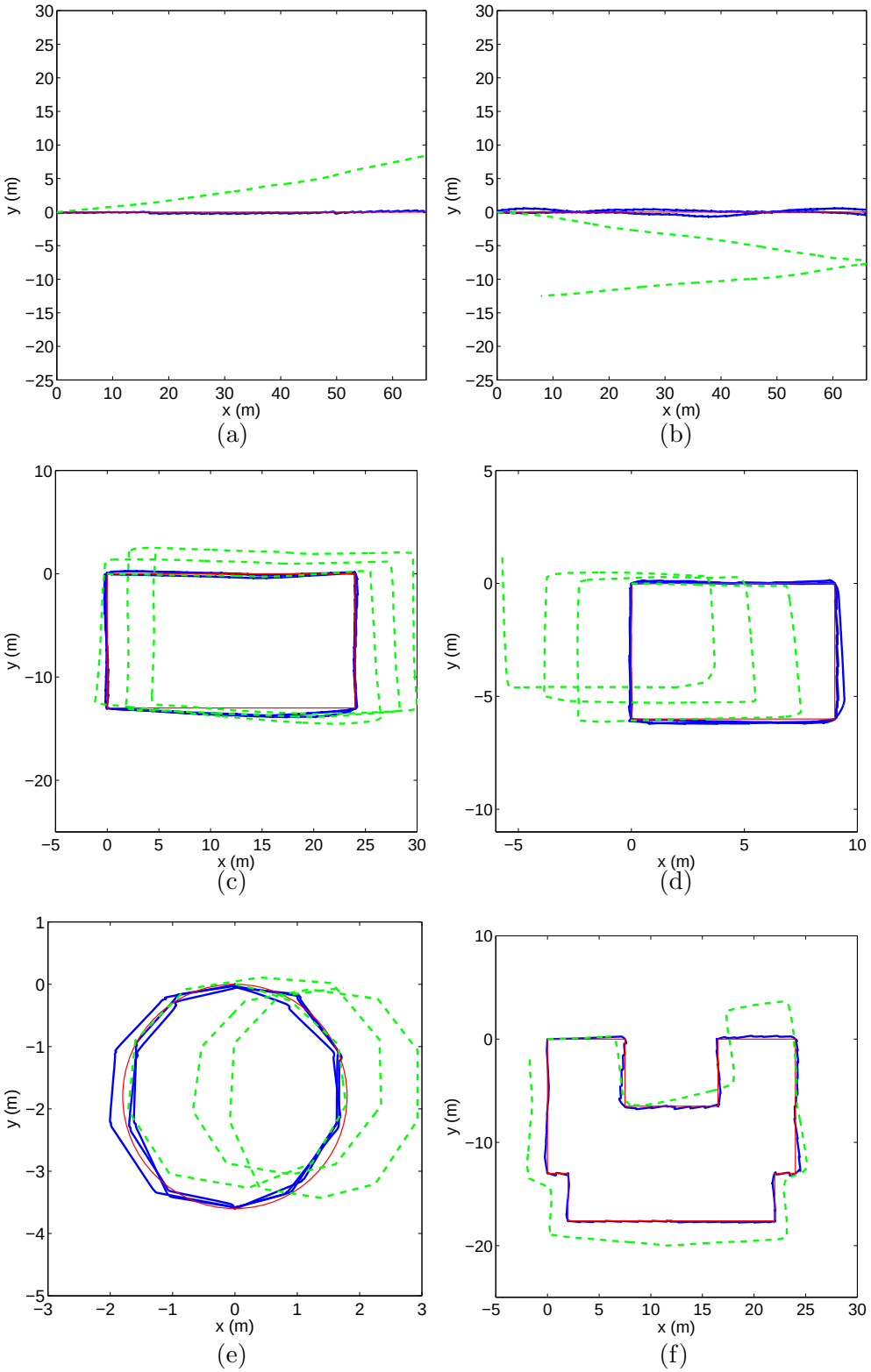


Figure 5.9: Sample reconstructed paths for experiments (a) 1, (b) 3, (c) 5, (d) 8, (e) 9, (f) 11, with (solid blue line) and without (dashed green line) activity recognition cues. The true path is indicated with the thin red line.

Table 5.5: Error values with activity recognition updates (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
1	0.10	0.11	0.11	0.12	0.11	0.18	0.10	0.14	**0.12**
2	0.16	0.50	0.08	0.34	0.23	0.18	0.11	0.08	**0.21**
3	0.08	0.16	0.04	0.13	0.09	0.06	0.08	0.05	**0.09**
4	0.17	0.09	0.14	0.23	0.19	0.09	0.12	0.13	**0.15**
5	0.10	0.15	0.09	0.12	0.07	0.11	0.09	0.09	**0.10**
6	0.09	0.13	0.04	0.10	0.11	0.08	0.09	0.04	**0.08**
7	0.06	0.20	0.08	0.11	0.12	0.14	0.15	0.09	**0.12**
8	0.10	0.18	0.09	0.15	0.08	0.14	0.08	0.06	**0.11**
9	0.21	0.29	0.62	0.15	0.22	0.55	0.55	0.19	**0.35**
10	0.16	0.30	0.20	0.45	0.22	0.23	0.44	0.48	**0.31**
11	0.64	0.13	0.12	0.30	0.14	0.13	1.02	0.10	**0.32**
							overall average:		**0.18**

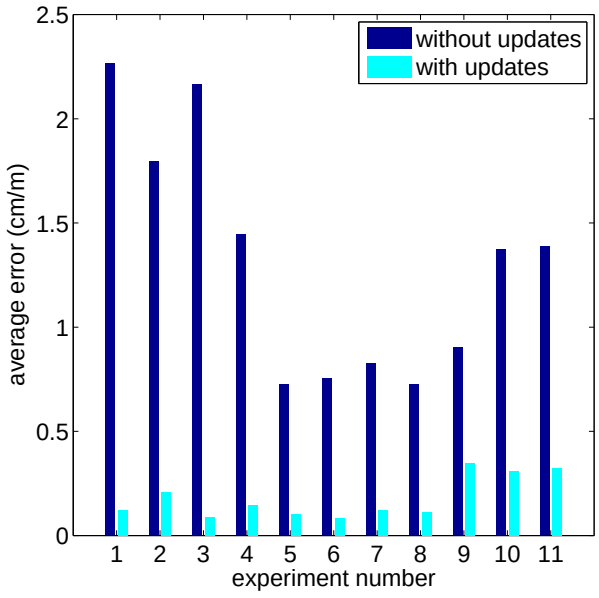


Figure 5.10: Average error values for all experiments with and without applying activity recognition position updates.

with activity recognition updates are presented in Tables 5.4 and 5.5, respectively. In the tables, the calculated errors using Equation (2.2) are divided to the path length for each experiment (Table 5.1) and then multiplied by 100 in order to convert to centimeters. Therefore, the values are in terms of (cm/m), interpreted as centimeter error per unit meter of path length. The last columns in both tables are the averages of the other columns, which represent the resulting average errors in a given experiment. The reduction in the average error values by introducing activity recognition position updates is illustrated in Figure 5.10, in which the percentage decrease in the errors can be visualized. For experiments 1–4 performed outdoors along a straight line, the average error without the updates is 1.92 cm/m. With the updates, this error is reduced to 0.14 cm/m, for which the percentage decrease in the average error can be calculated as $\frac{1.92-0.14}{1.92} \times 100 = 92.7\%$. For indoor experiments 5–11, the average error without the updates is 0.96 cm/m, which is reduced to 0.20 cm/m after the updates. Similarly, the average percentage decrease can be calculated as $\frac{0.96-0.20}{0.96} \times 100 = 79.1\%$. On the average, the error is reduced by 86%. We also calculate the error values at the activity switch locations. That is, when a position update is performed, the corresponding error is calculated. Then, these errors are averaged, yielding the values presented in Table 5.6. However, in a few cases, the positions are not updated to the correct location, as explained below.

Table 5.6: Averaged position errors at the position update locations (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
1	1.48	0.48	3.00	4.21	3.29	4.34	2.65	3.36	**2.85**
2	2.76	5.63	2.04	4.62	1.86	1.51	2.59	2.06	**2.88**
3	1.61	4.62	1.25	3.29	0.91	0.98	0.65	1.22	**1.82**
4	2.04	2.32	1.22	6.29	1.21	1.38	2.93	2.19	**2.45**
5	1.41	0.99	0.41	1.78	0.73	1.63	1.14	1.02	**1.14**
6	2.50	0.94	0.51	1.04	0.69	0.64	0.57	0.79	**0.96**
7	0.45	5.53	0.74	1.42	0.63	1.22	0.81	0.96	**1.47**
8	0.59	0.87	0.95	1.40	0.69	0.92	0.83	0.70	**0.87**
9	1.39	1.05	4.55	1.33	1.27	3.27	2.40	1.19	**2.06**
10	1.19	1.43	1.01	1.90	1.00	0.75	1.44	1.54	**1.28**
11	1.79	1.29	0.73	2.04	1.03	0.80	5.32	0.89	**1.73**
						overall average:			**1.77**

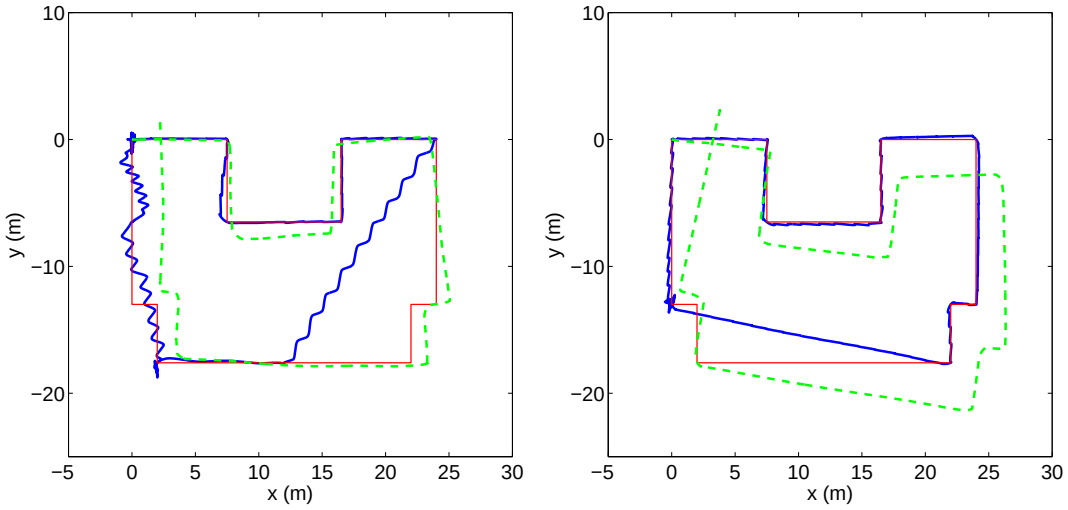


Figure 5.11: Incorrectly reconstructed paths caused by (a) incorrect activity recognition, and (b) offsets in sensor data.

The activity recognition performance is perfect for the WS switches, i.e., all WS switches are correctly recognized for all subjects in all experiments. Some instantaneous false alarms (type I errors[1]) are observed but they have been eliminated by applying a simple median filter to the activity "signal." For the WT switches, no false alarms are observed. However, some of the WT activity switches are not correctly recognized (type II errors[2]), since the thresholds are not set individually for each subject. These type II errors in WT switches sometimes cause the subsequent updates to be made in incorrect locations, such as the example shown in Figure 5.11(a). Here, the two WT switches while walking on the lower-right corner in the figure are not correctly detected. Over the $8 \times 11 = 88$ experiments performed in this study, this problem occurs only once. Even if there is no incorrect detection of activity, the same problem can still occur, as shown in Figure 5.11(b). Here, the offset in the angle measurement causes the forward filter to diverge from the actual path, and when a WT switch is detected, the calculated closest WT switch point (see Equation (5.12)) is not the actual turning point. This phenomenon is observed five times in all 88 experiments.

[1]In the context of this work, a type I error means that an activity switch has not actually occurred, but the recognition algorithm falsely detects that it has occurred.

[2]Conversely, a type II error means that an activity switch has actually occurred, but the

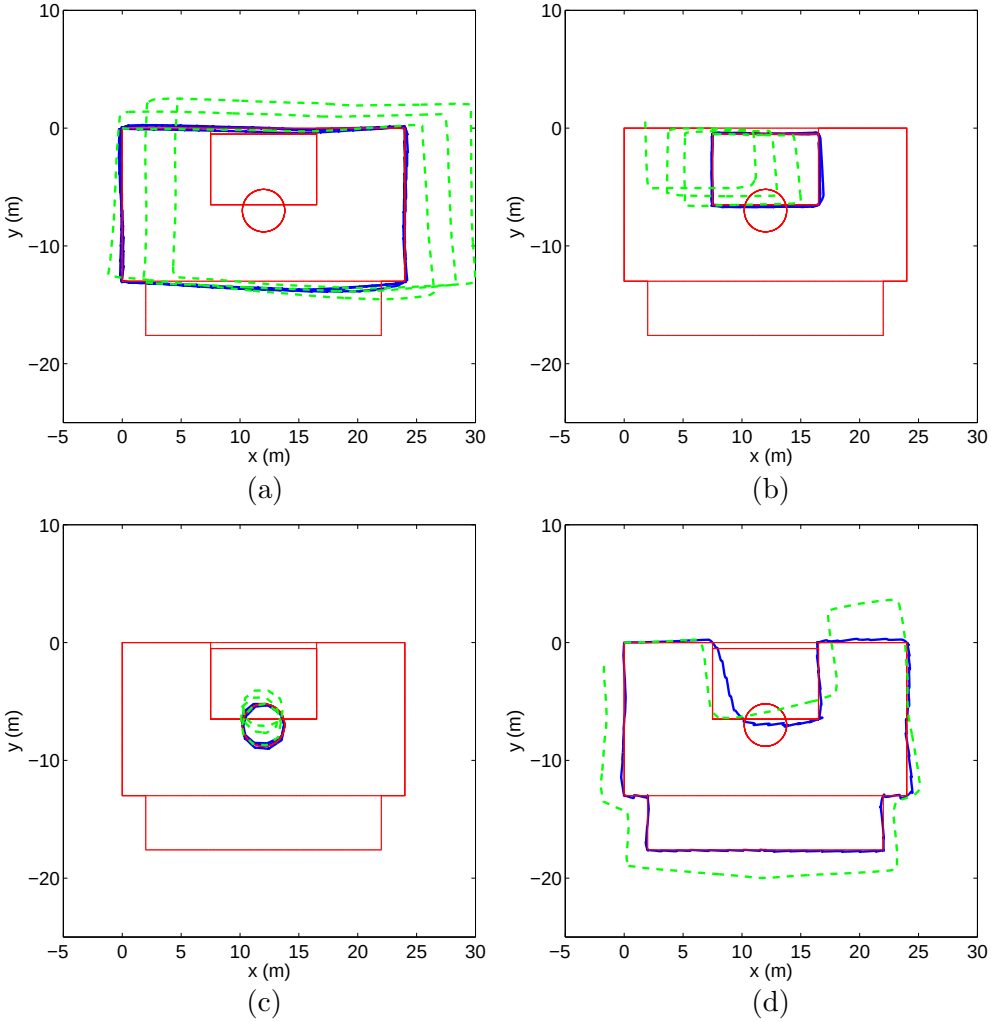


Figure 5.12: Sample reconstructed paths for experiments (a) 5, (b) 8, (c) 9, (d) 11, with (solid blue line) and without (dashed green line) activity recognition cues on the whole map. The true path is indicated with thin red line.

For experiments 5–11, we also reconstruct the paths using the whole of the map in Figure 5.3. That is, we define all corners on the map as WT switch points, and the points marked with × as WS switch points. The error values without activity recognition updates are the same as in Table 5.4. The results with activity recognition updates are given in Table 5.7, and the changes in the average error are given as a bar chart in Figure 5.13. The average errors for most of the experiments are reduced for this case as well, with the exception of

recognition algorithm fails to detect the activity switch. These terms are borrowed from the statistics terminology.

the experiment involving walking on a circle (experiment 9). In Table 5.7, it can be observed that the errors have increased for only three of the subjects. In these cases, the paths are not correctly reconstructed. This is caused by the fact that the circle experiment involves continuous turning activity, although not as sharp as turning at the corners. In fact, the thresholds for detecting the turning activity should be chosen such that the slow turning motion on the circular path is not detected as an activity switch, but the sharp turning motion at the corners is detected. This will, of course, depend on the radius of curvature of the circle, and the smaller it is, the larger will be the error. Looking at the experimental results, we observe that it is not possible to choose a single threshold that performs perfectly for all subjects, because every subject performs the walking motion uniquely in their own style. This problem can easily be solved by introducing uniformly spaced WT switch points on the circle. By defining 36 additional WT switch points on the circle that are 10° apart, we reduced the average error to 0.32 cm/m. However, since the radius of curvature of the circle in this experiment is too small and such sharp turns would very rarely be encountered on locations other than corners in a realistic situation, such a procedure would not be necessary in most cases. Sample reconstructions for this method are shown in Figure 5.12.

Table 5.7: Error values with activity recognition updates using the whole map (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
5	0.13	0.15	0.09	0.12	0.65	0.44	0.09	0.09	**0.22**
6	0.39	0.20	0.04	0.25	0.20	0.08	0.17	0.04	**0.17**
7	0.07	0.80	0.16	0.16	0.18	0.21	0.17	0.13	**0.23**
8	0.21	0.28	0.18	0.20	0.13	0.25	0.08	0.09	**0.18**
9	0.77	0.29	16.21	0.15	7.99	2.32	0.71	0.20	**3.58**
10	0.16	0.29	0.18	0.45	0.20	0.23	0.44	0.49	**0.31**
11	0.79	0.13	0.13	0.31	0.14	0.12	2.55	0.10	**0.53**
						overall average:			**0.75**

After introducing these additional WT switch positions, the errors between the true and reconstructed paths are given in Table 5.8, and the average position errors at the update locations are given in Table 5.9. In this case, the average error without the updates is again 0.96 cm/m, which is reduced to 0.28 cm/m using

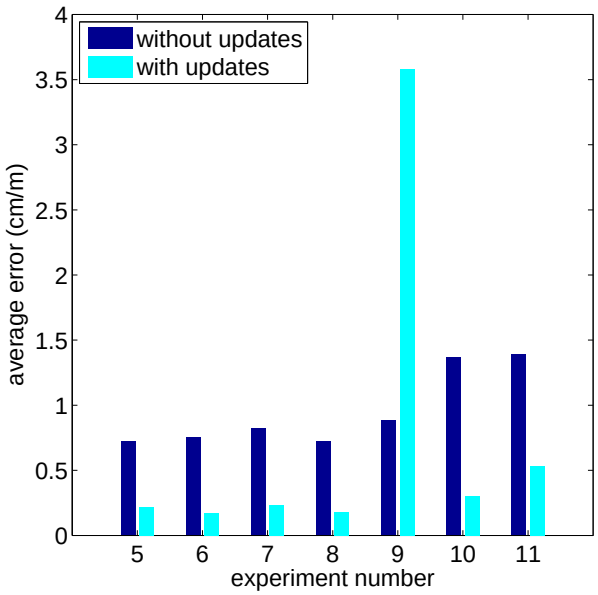


Figure 5.13: Average error values for the experiments 5–11 with and without activity recognition position updates when the whole map is used.

the activity updates and defining new WT switch points on the circle as stated in the previous section. In this case, the percentage reduction in the average error is $\frac{0.96-0.28}{0.96} \times 100 = 70.8\%$.

Note that the errors of experiments 10 and 11 increased slightly after the addition of more WT switch locations. This is illustrated in the reconstruction in Figure 5.12(d), which belongs to the same experiment as in Figure 5.9(f). Here, it can be observed that the performance of the latter is better. The degradation in the performance of the former is because of the addition of more WT switch points on the circle in order to improve the incorrect reconstructions of the circular path. This causes the closest WT switch point in Equation (5.12) to differ from the actual turning point in Figure 5.12(d). This means that, the addition of more switch points may cause degradations in the performances of other path reconstructions, and may affect the overall error negatively. Therefore, using more activity switch points on a map does not necessarily improve the overall performance.

Table 5.8: Error values with activity recognition updates using the whole map, after defining more WT switch locations (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
5	0.13	0.15	0.09	0.12	0.65	0.44	0.09	0.09	**0.22**
6	0.39	0.20	0.04	0.25	0.20	0.08	0.17	0.04	**0.17**
7	0.07	0.79	0.16	0.16	0.18	0.21	0.17	0.13	**0.23**
8	0.21	0.28	0.18	0.20	0.13	0.25	0.08	0.09	**0.18**
9	0.22	0.29	0.34	0.15	0.19	0.46	0.71	0.20	**0.32**
10	0.16	0.37	0.18	0.45	0.20	0.23	0.44	0.49	**0.32**
11	0.88	0.21	0.13	0.31	0.14	0.12	2.54	0.10	**0.55**
							overall average:		**0.28**

Table 5.9: Averaged position errors at the position update locations (in cm/m).

experiment no.	S1	S2	S3	S4	S5	S6	S7	S8	average
5	1.41	0.99	0.41	1.78	2.19	1.63	1.14	1.02	**1.32**
6	2.34	0.89	0.51	1.04	0.67	0.64	0.65	0.79	**0.94**
7	0.45	4.00	0.70	1.37	0.63	1.14	0.79	0.89	**1.25**
8	0.50	0.76	0.89	1.30	0.70	0.88	0.83	0.62	**0.81**
9	1.15	1.05	2.12	1.33	0.92	2.19	1.78	1.19	**1.47**
10	1.19	1.43	0.92	1.90	1.02	0.75	1.44	1.54	**1.27**
11	1.88	1.40	0.73	2.02	1.03	0.81	4.04	0.89	**1.60**
							overall average:		**1.24**

5.4.1 A Demonstrative 3-D Path Example

To demonstrate the applicability of our method in a realistic setting, we performed experiments in two floors of an indoor environment. The experiments are conducted in the Electrical and Electronics Engineering building in Bilkent University campus. Since this experiment is in 3-D, z-axis motion is also involved. Similar to the model above, the state equations now become

$$\begin{aligned} x(k) &= x(k-1) + \Delta d(k-1)\cos\left(\psi(k-1)\right) \\ y(k) &= y(k-1) + \Delta d(k-1)\sin\left(\psi(k-1)\right) \\ z(k) &= z(k-1) + \Delta d_z(k-1) \end{aligned} \tag{5.15}$$

where $d_z(k)$ is the distance traveled in the z direction, and $\Delta d_z(k-1) = d_z(k) - d_z(k-1)$ is the first-order difference. The z-direction distance $d_z(k)$ is calculated using ZUPT, in exactly the same manner as $d(k)$, after subtracting the gravitational acceleration g. We estimate g by taking the mean of the z-axis acceleration signal, while the subject stands still in the beginning of the experiment, after resetting the coordinate frames.

Additionally, we introduce the "stairs" activity to the activity set, which represents the activity state of the subject while ascending or descending stairs. We denote the walking-to-stairs and stairs-to-walking activity switches as a WZ switch (because we already used WS for a walking-to-standing switch), which emphasizes that the stairs activity involves motion in the z direction. Note that walking-to-stairs and stairs-to-walking switches are denoted by a single label, because at each walking-to-stairs switch location, a stairs-to-walking switch can also occur, and vice versa. In other words, walking-to-stairs and stairs-to-walking activity switch locations can not be separated from each other in any given map.

Distinguishing between walking and stairs activities is not straightforward, and a simple rule-based method like the one applied above can not be used for this case. Therefore, we use the k-nearest neighbor (k-NN) classifier described in the preceding chapter of this thesis. We use the data acquired in that work as training data for the classifier. From the previous chapter, we combine the activities walking in a parking lot (A9) and walking on a treadmill (A10) data

for the "walking" activity, and ascending stairs (A5) and descending stairs (A6) data for the "stairs" activity. To perform activity recognition between these two activities, we use the sensors on the right leg and the left leg, since they are mounted at the same position as in the previous chapter, and no coordinate frame reset is performed on the leg sensors. Therefore, the data are expected to be similar. We calculate the running mean and running variance values from the test data as features, using a sliding window of length 5 s. This length was chosen since the same length was also used in the training data for feature extraction. We do not use magnetometer data, since the accuracy of magnetometers are known to degrade in indoor environments [138]. The activities standing and turning are recognized using the same rule-based method as in the previous section.

In [5], we demonstrated that including training data from an individual improves the classification performance considerably. This is also confirmed in this study. The subject S8 in this study was also one of our test subjects in the activity recognition section, and the best classification performance in this experiment is achieved with subject S8. The results of the reconstruction before and after activity recognition updates are presented in Figure 5.14 using the data from subject S8. In the figure, the red (thin) line represents the true path, the green (light gray) line represents the reconstructed path without activity recognition updates, and the blue line (dark gray) represents the reconstructed path after applying the activity recognition updates.

We set the initial position of the subject as the origin, and the initial walking direction as the x-direction. In this setting, the only WS switch point is (0,0,0). We did not introduce any artificial WS switch locations in addition, since this experiment is performed in a realistic environment. The WT and WZ switch points are presented in Table 5.10 in matrix form for compactness, whose rows correspond to the coordinates of activity switch locations. These locations were determined considering the walked path and the construction plans of the building. We also used a tape measure to determine the coordinates of some of the waypoints on the path.

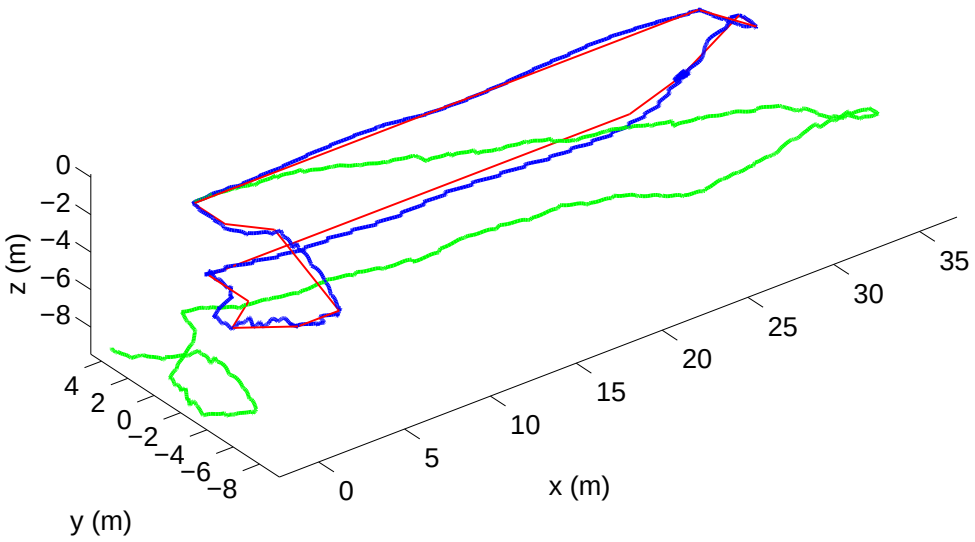


Figure 5.14: Sample reconstructed path for the 3-D experiment.

Table 5.10: Walking-to-turning (WT) and walking-to-stairs (WZ) activity switch locations.

walking-to-turning (WT)	walking-to-stairs (WZ)
$\begin{pmatrix} 0 & 0 & 0 \\ 32.78 & 0 & -2.08 \\ 32.78 & 1.30 & -2.08 \\ 0.90 & 0 & -4.16 \\ 0.90 & -3.00 & -4.16 \\ -1.20 & -4.50 & -4.16 \\ -0.90 & -9.10 & -2.08 \\ 1.50 & -9.10 & -2.08 \\ 1.50 & -4.20 & 0 \\ 0 & -2.40 & 0 \end{pmatrix}$	$\begin{pmatrix} 29.40 & 0 & 0 \\ 29.40 & 1.30 & -4.16 \\ -1.20 & -4.50 & -4.16 \\ 1.50 & -4.20 & 0 \\ -0.90 & -9.10 & -2.08 \\ 1.50 & -9.10 & -2.08 \end{pmatrix}$

Several heuristics are used in the simultaneous localization and activity recognition process. We observed that there are some instantaneous WZ switches while the subject is ascending or descending stairs. That is, occasionally the activity classifier instantaneously decides that the subject is walking even when he is actually not. Very rarely, the converse also occurs, i.e., the classifier detects the "stairs" activity while the subject is walking on the level floor. To avoid an incorrect position update at these instants, we introduce a condition on the WZ switches such that the switched activity (in this case, walking) must go on for at least three seconds for a position update to be applied. Another heuristic is that if the current activity is detected as walking, we do not modify the position in the z-direction in the prediction equation. This a fair heuristic because on the given map, walking activities only take place on the horizontal plane. If a map was given with possible uphill or downhill walking platforms (which is very unlikely in an indoor building environment), this heuristic would lead to incorrect results and should not be used.

As shown in Figure 5.14, the path reconstruction is almost perfect after introducing the updates. Using the error measure (2.2), the error between the true path and the reconstructed path using ZUPT only is calculated as 4.44 cm/m. With the activity recognition updates, this error is reduced to 0.35 cm/m, which corresponds to a percentage decrease in the average error by $\frac{4.44-0.35}{4.44} \times 100 = 92.1\%$. We also performed this experiment with the remaining subjects, whose data are not included in training the k-NN classifier. Since these subjects' data are not used in training, the activity recognition performance is not as good for some of the subjects. This is because each person has a different style of walking on the stairs as well as on a straight path. Distinguishing between these two motions is not possible with high accuracy if the classifiers are trained with the data of other subjects. Therefore, in a practical application, the classifier must be trained with the data of the user, which is an operation to be performed only once. Then, our method of simultaneous localization and activity recognition can be used, which improves the localization performance by reducing positioning errors about 90%.

5.4.2 Experiments on Spiral Stairs

To test the performance of the 3-D algorithm with continuous turning activity, we also performed experiments on spiral stairs. The subject ascends stairs on a fire escape for eight stories. We detect the turning activity using the rule-based algorithm in our 2-D experiments. Even though there is constant turning activity, the threshold defined in the rule-based algorithm is exceeded only occasionally, which results in a stairs-to-turning (ST) activity switch. Therefore, 80 equally-spaced ST activity switch locations are defined on the spiral stairs. The results are presented in Figure 5.15. Similarly, the green and blue lines represent the reconstructed path without and with activity recognition updates, respectively. The thin red line represents the actual path. For this experiment, the error is decreased from 2.08 cm/m to 0.24 cm/m with the activity recognition updates, resulting in 88% reduction.

5.5 Discussion

Our results demonstrate that path reconstruction improves significantly when activity recognition is performed simultaneously with localization, i.e., the activity switch cues are used for position updates. Considering the whole of the maps for both sets of 2-D experiments, the average percentage decrease in the error is 79%. The errors at the final point of the experiments are zero for all experiments, because the subjects stop at the end of the experiment on a WS switch point, and a final position update is performed.

The above errors are calculated using Equation (2.2), which represents the average distance between the true path curve and the reconstructed path curves. This is a spatial error measure between two sets of points that construct the curves. If the true position of the subject as a function of time was available, a more reliable error criterion would be to calculate the error between the true position and the estimated position at all time values, and then to take the time average. However, in our experiments, the true position of the subjects is not

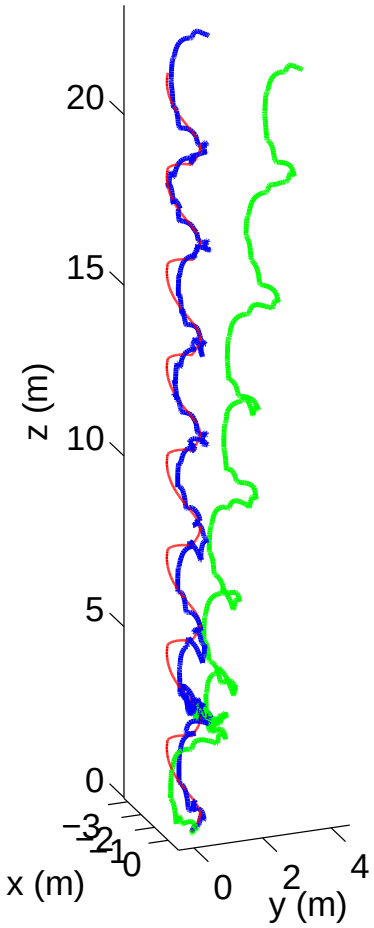


Figure 5.15: Sample reconstructed path for the spiral stairs.

available. Obtaining accurate true position data as a function of time is a difficult task outdoors because low-cost handheld GPS equipment have accuracies in the order of several meters. In indoor environments, accurate WiFi- or RFID-based positioning system setups might be necessary.

In the 2-D experiments, the WS and WT activity switch points on the map are assigned manually; all corners are defined as WT switch points, and WS switch points are assigned arbitrarily. In a practical setting, more activities may be added to the activity set and the map can be enhanced. For example, in an indoor office flat setting, we added the walking-to-stairs activity. In addition,

opening-door and closing-door activities might be added that correspond to locations in front of office doors. Determination of such activity switch locations can be automated as well; e.g., if cameras watching the corridors of a building are available, the activity switches can be determined by analyzing the activity patterns of individuals. Similar approaches can be used in urban outdoor settings, by observing locations such as street crossings or pedestrian traffic lights. However, this study shows that increasing the activity switch locations does not necessarily lead to more accurate position estimates. The drifts in inertial sensor data may lead to incorrect position updates as shown in Figure 5.11(b), which might not have occurred if there were fewer activity switch points. Even though the number of correct position updates increases with increasing number of activity switch locations, the number of such incorrect updates is expected to increase as well. Automatic or manual determination of activity switch locations in order to minimize the number of incorrect position updates on a given map remains an issue to be solved.

Chapter 6

Summary and Conclusion

In this thesis, we considered three different problems in the intelligent sensing area, specifically in ultrasonic sensing and inertial sensing. Here, we summarize the results and provide the reader with future research directions.

In Chapter 2, we have presented two approaches to compactly and efficiently represent the maps obtained by processing ultrasonic arc maps (UAMs) with different techniques. Representing the map points with snake curves or Kohonen's self-organizing maps (SOMs) makes it possible to compare maps obtained with different techniques among themselves, as well as with an absolute reference. For this purpose, we have defined and employed an error criterion that can be used to compare two different sets of points based on the Euclidean distance measure. The two sets of points can be chosen as (i) two different sets of map points acquired with different mapping techniques or different sensing modalities (e.g., ultrasonic data and laser data), (ii) two sets of curve points (e.g., two snake curves) fitted to maps extracted by different mapping techniques or different sensing modalities, or (iii) a set of extracted map points and a set of curve points fitted to those points. Our purpose here was to represent ultrasonic maps efficiently and to evaluate the performance of UAM processing techniques through the use of a demonstrative example. Among the eight UAM processing techniques considered, the directional maximum method can be considered as one of the best in terms of the overall performance.

Although both active contours and SOMs can be employed to fill the erroneous gaps in discrete point maps, active contours are superior in eliminating outliers in the data, resulting in smaller errors. Active contours are more capable of fitting accurately to environmental features with high curvature, such as edges and corners, as well as smoother features, such as planar walls, in typical indoor environments.

A physical interpretation of the parameters and guidelines on parameter selection are provided. The best-fitting parameter values are found by uniform sampling of the parameter space and by particle swarm optimization (PSO). For snake curve fitting, the error values are comparable for the two approaches, whereas for SOM, PSO consistently results in smaller errors. Another alternative is to use non-parametric snakes that utilize a kernel density estimation approach [151]. However, this formulation does not completely eliminate the parameter optimization procedure, as a number of parameters must be determined for both fixed- and variable-bandwidth kernels, especially for noisy images. Furthermore, an additional heuristic procedure needs to be used to assure convergence to concavities in the map.

The two methods are sufficiently general that they can also be applied to map data points acquired with other mapping techniques and sensing modalities. The results can be extended to 3-D data by fitting 3-D shapes. Another possible extension of this work would be the automatic determination of the appropriate number of curves or shapes to be fitted to a given set of extracted map points using clustering techniques. Determining whether the curves should be open or closed, or the shapes convex or concave, and initializing the multiple curve parameters are other challenging issues.

In Chapter 3, we applied the well-known Allan variance procedure in order to model the error characteristics in the inertial/magnetic sensor units used. The analysis was performed using data acquired when the sensors are stationary on a table for 12 hours. For the gyroscopes, eliminating the transients by a curve fitting procedure resulted in white noise. However, correlated noise components exist for accelerometers and magnetometers. Further research on this area can

be conducted using data when the sensors are in motion, e.g., on a rotary table, in order to analyze the dynamic error characteristics of the sensors.

In Chapter 4, we have presented the results of a comparative study where features extracted from miniature inertial and magnetometer signals are used for classifying human activities. We compared a number of classification techniques based on the same data set in terms of their correct differentiation rates, confusion matrices, computational costs, and training and storage requirements. Bayesian decision making (BDM) achieves higher correct classification rates in general compared to the other classification techniques, and has relatively small computational time and storage requirements. This parametric method can be employed in similar classification problems, where it is appropriate to model the feature space with multi-variate Gaussian distributions. The support vector machine (SVM) and k-nearest neighbor (k-NN) methods are the second-best choices in terms of classification accuracy but SVM requires a considerable amount of training time. For real-time applications, least squares method (LSM) could also be considered a suitable choice because it is faster than BDM at the expense of a lower correct classification rate.

We implemented and compared a number of different cross-validation techniques in this chapter. The correct classification rates obtained by subject-based leave-one-out cross validation (L1O) are usually lower. This is because in L1O, the data of a particular subject are not used for training the classifiers; it is only used for testing. However, in P-fold cross-validation and repeated random sub-sampling (RRSS) methods, data from a particular subject are used for both training and testing. We observed that BDM gives the best results in P-fold and RRSS methods, but not in L1O. In L1O, k-NN and SVM methods overperform the BDM method. This means that, in a practical application, if training data of an individual are available, a simple Gaussian classifier like BDM can be used with confidence. However, if the training data are not available for some reason, one should resort to more complex classifiers such as k-NN and SVM for better performance, in exchange for additional computational cost.

There are several possible future research directions that can be explored in

this area:

An aspect of activity recognition and classification that has not been much investigated is the normalization between the way different individuals perform the same activity. Each person does a particular activity differently due to differences in body size, style, and timing. Although some approaches may be more prone to highlighting personal differences, new techniques need to be developed that involve time-warping and projections of signals and comparing their differentials.

To the best of our knowledge, optimizing the positioning, number, and type of sensors has not been much studied. Typically, some configuration, number, and modality of sensors is chosen and used without strong justification.

Detecting and classifying falls using inertial sensors is another important problem that has not been sufficiently well investigated [79], due to the difficulty of designing and performing fair and realistic experiments in this area [71]. Therefore, standard definitions of falls and systematic techniques for detecting and classifying falls still do not exist. In our ever-aging population, it seems imperative to develop such definitions and techniques as soon as possible [77, 78].

Fusing information from inertial sensors and cameras can be further explored to provide robust solutions in human activity monitoring, recognition, and classification. Joint use of these two sensing modalities increases the capabilities of intelligent systems and enlarges the application potential of inertial and vision systems.

In Chapter 5, we presented the results of introducing activity recognition cues in order to perform position updates in a pedestrian dead-reckoning system, thus performing localization and activity recognition simultaneously. We use the well-known zero velocity update (ZUPT) method to estimate the distance traveled, and an orientation sensor mounted on the chest to estimate the heading. Position errors occur because of drifts and offsets in these inertial sensors caused by initial misplacement, slips on the body during operation because of loose mounting, in addition to the characteristic drift present in the inertial sensing equipment.

These errors are corrected using a map of the environment with predefined activity switch locations at which position updates are performed. When a current position update is performed using the activity recognition information, optimal estimation techniques are also applied to the past estimates and they are also corrected. It is concluded that activity recognition cues considerably improve the performance of the pedestrian dead reckoning (PDR) system, reducing the position errors drastically. However, distinguishing between similar activities such as ascending/descending stairs and walking is not an easy task that can be accomplished by using a simple rule-based classifier. In this case, using more complex classifiers would be necessary. To improve the performance, training data of the test subject should be available to the classifiers, otherwise similar activities can be confused. If such training data are available, it is possible to reduce the localization error by about 10-fold when activity recognition cues are utilized simultaneously in localization. Even though in this chapter we demonstrate that it is possible to perform pedestrian localization without any external aid other than a map, there are several unresolved issues related to this approach. We list possible future research directions in this track below:

We demonstrate that increasing the possible activity switch locations does not necessarily improve the localization results. Therefore, methods of automated or manual determination of activity switch locations and the number of such locations can be developed. Such procedures should take into account possible misclassifications of activity for a given map, as well as drift characteristics of the sensors used.

More activities can be defined, depending on the setting involved. For example, underground mines are one of the most suitable settings to which the methods developed in this study can be applied, because no external positioning method is available. By defining activities specific to that setting and implementing specific activity recognition methods, accurate positioning can be obtained.

Similar to how activity recognition cues aid localization in our method, location cues can be utilized in order to perform accurate activity recognition. Even though activity recognition is quite accurate when training and test data from

the same user are available, the accuracy degrades when training data from the same user are not available. However, for a given position on a map, the possible set of activities that can be performed is limited. By utilizing the location information in the activity recognition algorithm, this problem can be eliminated. In this case, activity recognition and localization can be viewed as a loop, very much like simultaneous localization and mapping (SLAM) methods in mobile robotics, where activity recognition aids localization and vice versa.

Appendix A

Quaternions and Rotations in 3-D Space

A.1 Quaternion Algebra

Here, we give basic definitions and properties related to quaternions. Most of the nomenclature and properties are adopted from [152].

A quaternion can be viewed as a four dimensional vector given by

$$q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{A.1}$$

Here, q_0 is the scalar part, and $\mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ is the vector part of the quaternion q. $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are defined such that the following property is satisfied:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{A.2}$$

From Equations (A.2), the following can be deduced:

$$\begin{aligned} \mathbf{ij} &= \mathbf{k} = -\mathbf{ji} \\ \mathbf{jk} &= \mathbf{i} = -\mathbf{kj} \\ \mathbf{ki} &= \mathbf{j} = -\mathbf{ik} \end{aligned} \tag{A.3}$$

Using these properties, two quaternions p and q given by

$$\begin{aligned} p &= p_0 + \mathbf{p} = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k} \\ q &= q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \end{aligned} \tag{A.4}$$

can be multiplied to give the resulting quaternion as

$$pq = p_0q_0 - \mathbf{p}\cdot\mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q} \tag{A.5}$$

The complex conjugate of a quaternion is defined as

$$q^* = q_0 - \mathbf{q} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \tag{A.6}$$

and the norm of a quaternion is defined as $||q|| = \sqrt{q^*q}$.

Unit quaternions ($||q|| = 1$) can be used to represent rotations in 3-D space. A unit quaternion can be written as

$$q = \cos(\theta) + \mathbf{u}\sin(\theta) \tag{A.7}$$

where $\mathbf{u}$ is a unit vector. For a vector $\mathbf{v} \in \mathbb{R}^3$ and the unit quaternion q given in Equation (A.7), the operation

$$L_q(\mathbf{v}) = q\mathbf{v}q^* \tag{A.8}$$

corresponds to the rotation of the vector $\mathbf{v}$ by 2θ about the axis defined by $\mathbf{u}$. Note that here $\mathbf{v}$ should be regarded as a quaternion with zero scalar part.

A.2 Euler Angles

Another method of representing rotations in 3-D space is using Euler angles. Euler angles represent the rotation required to transform a coordinate system XYZ to another coordinate system xyz in terms of three different angles. Twelve different rotation sequences are possible for this representation, however, here only the ZYX sequence (also known as the aerospace sequence) will be described.

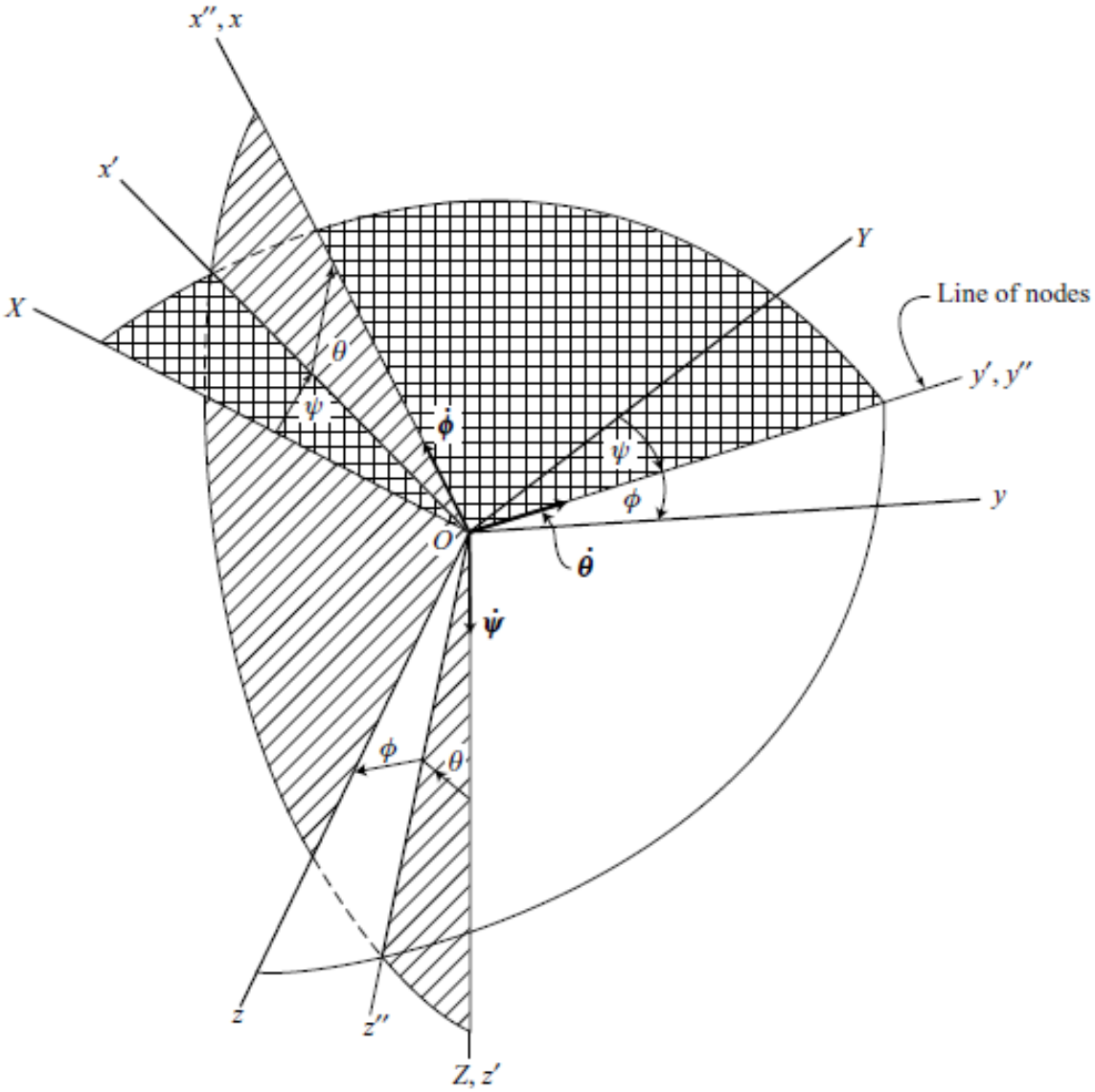


Figure A.1: Euler angles ZYX sequence (reprinted from [154]).

In Figure A.1, three different angles that represent the rotation between XYZ and xyz are shown [154]. Suppose that initially these two coordinate systems coincide. In order to obtain xyz from XYZ, the following rotations should be performed in this particular order:

1. rotate about Z-axis by ψ (obtain $x'y'z'$ coordinate system),
2. rotate about y'-axis by θ (obtain $x''y''z''$ coordinate system),
3. rotate about x''-axis by ϕ (obtain xyz coordinate system)

This convention is mostly used in aerospace applications. In aerospace terminology, ϕ is known as roll angle, θ is known as pitch angle, and ψ is known as the yaw angle. The Euler angles for this sequence satisfy the following inequalities:

$$\begin{aligned} 0 < \phi < 2\pi \\ -\pi/2 < \theta < \pi/2 \\ 0 < \psi < 2\pi \end{aligned} \tag{A.9}$$

In this sequence, if the pitch angle is exactly $\pm\frac{\pi}{2}$, the configuration becomes singular and it becomes impossible to distinguish between the roll and the yaw angles. Thus, this sequence should be used in applications where the pitch angle does not take values close to $\pm\frac{\pi}{2}$. In fact, it has been shown that any three-parameter representation of 3-D rotations must involve singularities [153].

It is possible to convert a unit quaternion to Euler angles (ZYX sequence) using the following formulas:

$$\begin{aligned} \phi &= \tan^{-1}\left(\frac{2q_2q_3 + 2q_0q_1}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right) \\ \theta &= \sin^{-1}(2q_0q_2 - 2q_1q_3) \\ \psi &= \tan^{-1}\left(\frac{2q_1q_2 + 2q_0q_3}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right) \end{aligned} \tag{A.10}$$

where the inverse tangent function should be evaluated by taking all four quadrants into account.

Appendix B

Combining Unbiased Estimators

Suppose $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are two estimators of a random vector $\mathbf{X}$, with error covariances $\mathbf{\Sigma}_1 = E[(\hat{\mathbf{x}}_1 - \mathbf{X})(\hat{\mathbf{x}}_1 - \mathbf{X})^T]$ and $\mathbf{\Sigma}_2 = E[(\hat{\mathbf{x}}_2 - \mathbf{X})(\hat{\mathbf{x}}_2 - \mathbf{X})^T]$, respectively. We assume that the estimators are unbiased, i.e.,

$$E(\hat{\mathbf{x}}_1 - \mathbf{X}) = E(\hat{\mathbf{x}}_2 - \mathbf{X}) = 0 \tag{B.1}$$

and the estimation errors are uncorrelated, i.e.,

$$E[(\hat{\mathbf{x}}_1 - \mathbf{X})(\hat{\mathbf{x}}_2 - \mathbf{X})^T] = \mathbf{0} \tag{B.2}$$

A linear combination of these two estimators is given as

$$\hat{\mathbf{x}} = \mathbf{W}_1\hat{\mathbf{x}}_1 + \mathbf{W}_2\hat{\mathbf{x}}_2 \tag{B.3}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are weighting matrices.

We would like to find the optimal weighting matrices such that the covariance matrix of the combined estimate given by

$$\mathbf{\Sigma} = E[(\hat{\mathbf{x}} - \mathbf{X})(\hat{\mathbf{x}} - \mathbf{X})^T] \tag{B.4}$$

is "minimum," which should be interpreted in the sense of the following definition [155].

Definition. Let $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ be symmetric non-negative definite matrices such that the difference $\mathbf{\Sigma}_2 - \mathbf{\Sigma}_1$ is non-zero and non-negative definite. Then, $\mathbf{\Sigma}_1$ is said to be less than $\mathbf{\Sigma}_2$.

If the combined estimate is to be unbiased, the following condition should hold:

$$E(\hat{\mathbf{x}}) = \mathbf{W}_1 E(\hat{\mathbf{x}}_1) + \mathbf{W}_2 E(\hat{\mathbf{x}}_2) = E(\mathbf{X}) \tag{B.5}$$

Since $E(\hat{\mathbf{x}}_1) = E(\hat{\mathbf{x}}_2) = E(\mathbf{X})$, the condition reduces to $\mathbf{W}_1 + \mathbf{W}_2 = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of appropriate size. The combined estimate can be expressed as

$$\hat{\mathbf{x}} = \mathbf{W}_1 \hat{\mathbf{x}}_1 + (\mathbf{I} - \mathbf{W}_1)\hat{\mathbf{x}}_2 \tag{B.6}$$

and the optimization criterion is given as

$$\begin{aligned} \mathbf{W}_1^* &= \arg\min_{\mathbf{W}_1} \mathbf{\Sigma} \\ &= \arg\min_{\mathbf{W}_1} \mathbf{W}_1(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2)\mathbf{W}_1^T - \mathbf{W}_1\mathbf{\Sigma}_2 - \mathbf{\Sigma}_2\mathbf{W}_1^T + \mathbf{\Sigma}_2 \end{aligned} \tag{B.7}$$

In this case, the minimizing $\mathbf{W}_1$ can be found as [155]

$$\mathbf{W}_1^* = \mathbf{\Sigma}_2(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2)^{-1} = \left(\mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}\right)^{-1} \mathbf{\Sigma}_1^{-1} \tag{B.8}$$

and $\mathbf{W}_2^*$ is given as

$$\mathbf{W}_2^* = \mathbf{I} - \mathbf{W}_1^* = \left(\mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}\right)^{-1} \mathbf{\Sigma}_2^{-1} \tag{B.9}$$

Therefore, the optimal combination of estimators $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ is

$$\hat{\mathbf{x}} = \left(\mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}\right)^{-1} \left(\mathbf{\Sigma}_1^{-1}\hat{\mathbf{x}}_1 + \mathbf{\Sigma}_2^{-1}\hat{\mathbf{x}}_2\right) \tag{B.10}$$

and the error covariance can be calculated as

$$\mathbf{\Sigma} = \left(\mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}\right)^{-1} \tag{B.11}$$

Note that in Equation (B.10), individual estimates are weighted by inverses of the corresponding covariance matrices. This result also agrees with physical intuition. For example, if one estimate is more reliable than the other (i.e., its covariance matrix is less than the other), its weight will be larger compared to the other weight, and vice versa.

Bibliography

[1] B. Barshan and K. Altun, "Directional processing of ultrasonic arc maps and its comparison with existing techniques (Doğrultulu maksimum yöntemiyle ultrasonik ark haritalarının işlenmesi ve varolan yöntemlerle karşılaştırılması)," in *Proceedings of the IEEE 15th Conference on Signal Processing, Communications and Applications (CD-ROM)*, Eskişehir, Turkey, 11–13 June 2007.

[2] K. Altun and B. Barshan, "Performance evaluation of ultrasonic arc map processing techniques by active snake contours," *Springer Tracts in Advanced Robotics (STAR) Series*, vol. 44, pp. 185–194, Berlin, Heidelberg, Germany: Springer-Verlag, February 2008, in *European Robotics Symposium 2008*.

[3] K. Altun and B. Barshan, "Employing active contours and artificial neural networks in representing ultrasonic range data," in *Proceedings of the 16th European Signal Processing Conference*, (Lausanne, Switzerland), 25–28 August 2008.

[4] K. Altun and B. Barshan, "Representing and evaluating ultrasonic maps using active snake contours and Kohonen's self-organizing feature maps," *Autonomous Robots*, 29(2):151–168, 2010.

[5] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, 43(10):3605–3620, 2010.

[6] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," *Lecture Notes in Computer Science (LNCS)*, vol.6219, pp. 38–51, Berlin, Heidelberg, Germany: Springer-Verlag, August 2010, in *Human Behavior Understanding*.

[7] O. Tunçel, K. Altun, and B. Barshan, "Classification of leg motions by processing gyroscope signals (Jiroskop sinyallerinin işlenmesiyle bacak hareketlerinin sınıflandırılması)," in *Proceedings of the IEEE 17th Conference on Signal Processing, Communications and Applications (CD-ROM)*, Side, Antalya, Turkey, 9–11 April 2009.

[8] O. Tunçel, K. Altun, and B. Barshan, "Classifying human leg motions with uniaxial piezoelectric gyroscopes," *Sensors*, 9(11):8508–8546, 2009.

[9] A. Küpper, *Location-Based Services—Fundamentals and Operation*, U.K.: John Wiley & Sons, Ltd., 2005.

[10] M. Kass, A. Witkin, and D. Tersopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1(4):321–331, 1987.

[11] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, 43(1):59–69, 1982.

[12] J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE Transactions on Robotics and Automation*, RA-1(1):31–41, 1985.

[13] W. Gex and N. Campbell, "Local free space mapping and path guidance," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 424–431, Raleigh, NC, U.S.A., 30 March–2 April 1987.

[14] M. Drumheller, "Mobile robot localization using sonar," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):325–332, 1987.

[15] R. Kuc and M. W. Siegel, "Physically-based simulation model for acoustic sensor robot navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(6):766–778, 1987.

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

[17] D. Başkent and B. Barshan, "Surface profile determination from multiple sonar data using morphological processing," *International Journal of Robotics Research*, 18(8):788–808, 1999.

[18] A. Elfes, "Sonar based real-world mapping and navigation," *IEEE Transactions on Robotics and Automation*, RA-3(3):249–265, 1987.

[19] B. Barshan, "Directional processing of ultrasonic arc maps and its comparison with existing techniques," *International Journal of Robotics Research*, 26(8):797–820, 2007.

[20] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *International Journal of Robotics Research*, 21(4):311–330, 2002.

[21] D. Ribas, P. Ridao, J. Neira, and J. D. Tardós, "Line extraction from mechanically scanned imaging sonar," *Pattern Recognition and Image Analysis, Lecture Notes in Computer Science*, vol. 4477, pp. 322–329, July 2007.

[22] B. Barshan, "Ultrasonic surface profile determination by spatial voting," *Electronics Letters*, 35(25):2232–2234, 1999.

[23] D. Silver, D. Morales, I. Rekleitis, B. Lisien, and H. Choset, "Arc carving: obtaining accurate, low latency maps from ultrasonic range sensors," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1554–1561, New Orleans, LA, U.S.A., 26 April–1 May 2004.

[24] O. Wijk and H. I. Christensen, "Triangulation-based fusion of sonar data with application in robot pose tracking," *IEEE Transactions on Robotics and Automation*, 16(6):740–752, 2000.

[25] B. Barshan, "Objective error criterion for evaluation of mapping accuracy based on sensor time-of-flight measurements," *Sensors*, 8(12):8248–8261, 2008.

[26] S. Menet, P. Saint-Marc, and G. Medioni, "Active contour models: Overview, implementation and applications," in *Proceedings of the IEEE*

International Conference on Systems, Man and Cybernetics, pp. 194–199, Los Angeles, CA, U.S.A., 4–7 November 1990.

[27] L. D. Cohen, "On active contour models and balloons," *Computer Vision, Graphics, and Image Processing (CVGIP): Image Understanding*, 53(2):211–218, 1991.

[28] L. D. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1993.

[29] M. Jacob, T. Blu, and M. Unser, "Efficient energies and algorithms for parametric snakes," *IEEE Transactions on Image Processing*, 13(9):1231–1244, 2004.

[30] T. M. J. Liang and D. Terzopoulos, "United snakes," *Medical Image Analysis*, 10(2):215–233, 2006.

[31] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[32] A. Rosenfeld and J. L. Pfaltz, "Distance functions on digital pictures," *Pattern Recognition*, 1(1):33–61, 1968.

[33] C. Xu and J. L. Prince, "Snakes, shapes and gradient vector flow," *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

[34] A. Kurz, "Constructing maps for mobile robot navigation based on ultrasonic range data," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(2):233–242, 1996.

[35] S. Yamada, "Recognizing environments from action sequences using self-organizing maps," *Applied Soft Computing*, 4(1):35–47, 2004.

[36] G. S. Kumar, P. K. Kalra, and S. G. Dhande, "Curve and surface reconstruction from points: an approach based on self-organizing maps," *Applied Soft Computing*, 5(1):55–66, 2004.

[37] G. K. Knopf and A. Sangole, "Interpolating scattered data using 2D self-organizing feature maps," *Graphical Models*, 66(1):50–69, 2004.

[38] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 26 November–2 December 1995.

[39] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization—an overview," *Swarm Intelligence*, 1(1):33–57, 2007.

[40] O. Wijk and H. I. Christensen, "Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data," *Robotics and Autonomous Systems*, 31(1–2):31–42, 2000.

[41] B. Barshan and D. Başkent, "Morphological surface profile extraction with multiple range sensors," *Pattern Recognition*, 34(7):1459–1467, 2001.

[42] B. Barshan and D. Başkent, "Map building from range data using mathematical morphology," *World Scientific Series in Robotics and Intelligent Systems*, vol. 26, ch. 7, pp. 111–135, New Jersey: World Scientific, 2001. in *Active Sensors for Local Planning in Mobile Robotics*, P. Probert Smith (ed.).

[43] B. Barshan and D. Başkent, "Comparison of two methods of surface profile extraction from multiple ultrasonic range measurements," *Measurement Science and Technology*, 11(6):833–844, 2000.

[44] I. J. Cox and G. T. Wilfong, ed., *Autonomous Robot Vehicles*, New York: Springer-Verlag, 1990. Section on Inertial Navigation edited by M. M. Kuritsky, and M. S. Goldstein.

[45] D. A. Mackenzie, *Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance*. Cambridge, MA, U.S.A.: MIT Press, 1990.

[46] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[47] B. Barshan and H. F. Durrant-Whyte, "Evaluation of a solid-state gyroscope for robotics applications," *IEEE Transactions on Instrumentation and Measurement*, 44(1):61–67, 1995.

[48] C.-W. Tan and S. Park, "Design of accelerometer-based inertial navigation systems," *IEEE Transactions on Instrumentation and Measurement*, 54(6):2520–2530, 2005.

[49] J. G. Nichol, S. P. N. Singh, K. J. Waldron, L. R. Palmer, III, and D. E. Orin, "System design of a quadrupedal galloping machine," *International Journal of Robotics Research*, 23(10–11):1013–1027, 2004.

[50] P.-C. Lin, H. Komsuoglu, and D. E. Koditschek, "Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits," *IEEE Transactions on Robotics*, 22(5):932–943, 2006.

[51] W. T. Ang, P. K. Khosla, and C. N. Riviere, "Design of all-accelerometer inertial measurement unit for tremor sensing in hand-held microsurgical instrument," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1781–1786, Taipei, Taiwan, September 2003.

[52] W. T. Ang, P. K. Pradeep, and C. N. Riviere, "Active tremor compensation in microsurgery," in *26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 1, pp. 2738–2741, San Francisco, CA, U.S.A., 1–5 September 2004.

[53] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, U.K.: IEE, 2nd ed., 2004.

[54] Xsens Technologies B.V., Enschede, Holland, *MTi and MTx User Manual and Technical Documentation*, 2009. `http://www.xsens.com`.

[55] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, "A high integrity IMU/GPS navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics and Automation*, 15(3):572–578, 1999.

[56] A. M. Sabatini, "A wavelet-based bootstrap method applied to inertial sensor stochastic error modeling using the Allan variance," *Measurement Science and Technology*, 17(11):2980–2988, 2006.

[57] IEEE Std 952-1997, *IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros*, IEEE, 1997.

[58] IEEE Std 1293-1998, *IEEE Standard Specification Format Guide and Test Procedure for Linear, Single-Axis, Nongyroscopic Accelerometers*, IEEE, 1998.

[59] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, pp. 681–688, Cambridge, U.K.: Cambridge University Press, 2nd ed., 1992.

[60] R. L. Anderson, "Distribution of the serial correlation coefficient," *Annals of Mathematical Statistics*, 13(1):1–13, 1942.

[61] D. W. Allan, "Statistics of atomic frequency standards," *Proceedings of the IEEE*, 54(2):221–230, 1966.

[62] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Transactions on Instrumentation and Measurement*, 57(1):140–149, 2008.

[63] W. Stockwell, "Bias stability measurement: Allan variance," Tech. Rep., Crossbow Technology Inc., CA, U.S.A. `http://www.xbow.com/pdf/Bias_Stability_Measurement.pdf`.

[64] M. Park and Y. Gao, "Error analysis and stochasic modeling of low-cost MEMS accelerometer," *Journal of Intelligent and Robotic Systems*, 46(1):27–41, 2006.

[65] Y. Stebler, S. Guerrier, J. Skaloud, and M. Victoria-Feser, "Constrained expectation-maximization algorithm for stochastic inertial error modeling: study of feasibility," *Measurement Science and Technology*, 22(8):085204, 2011.

[66] J. Borenstein, L. Ojeda, and S. Kwanmuang, "Heuristic reduction of gyro drift for personnel tracking systems," *Journal of Navigation*, 62(1):41–58, 2009.

[67] C. A. Greenhall, "Spectral ambiguity of Allan variance," *IEEE Transactions on Instrumentation and Measurement*, 47(3):623–627, 1998.

[68] O. J. Woodman, "An introduction to inertial navigation," Tech. Rep. UCAM-CL-TR-696, University of Cambridge, Cambridge, U.K., 2007.

[69] S. M. Kay, *Modern Spectal Estimation: Theory and Application*, New Jersey: Prentice Hall, 1998.

[70] W. Zijlstra and K. Aminian, "Mobility assessment in older people: new possibilities and challenges," *European Journal of Ageing*, 4(1):3–12, 2007.

[71] M. J. Mathie, A. C. F. Coster, N. H. Lovell, and B. G. Celler, "Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement," *Physiological Measurement*, 25(2):R1–R20, 2004.

[72] W. Y. Wong, M. S. Wong, and K. H. Lo, "Clinical applications of sensors for human posture and movement analysis: a review," *Prosthetics and Orthotics International*, 31(1):62–75, 2007.

[73] A. M. Sabatini, "Inertial sensing in biomechanics: a survey of computational techniques bridging motion analysis and personal navigation," pp. 70–100, *Computational Intelligence for Movement Sciences: Neural Networks and Other Emerging Techniques*, Idea Group Publishing, Hershey, PA, U.S.A., 2006.

[74] F. Audigié, P. Pourcelot, C. Degueurce, D. Geiger, and J. M. Denoix, "Fourier analysis of trunk displacements: a method to identify the lame limb in trotting horses," *Journal of Biomechanics*, 35(9):1173–1182, 2002.

[75] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors,"

IEEE Transactions on Information Technology in Biomedicine, 10(1):119–128, 2006.

[76] M. J. Mathie, B. G. Celler, N. H. Lovell, and A. C. F. Coster, "Classification of basic daily movements using a triaxial accelerometer," *Medical and Biological Engineering and Computing*, 42(5):679–687, 2004.

[77] K. Hauer, S. E. Lamb, E. C. Jorstad, C. Todd, and C. Becker, "Systematic review of definitions and methods of measuring falls in randomised controlled fall prevention trials," *Age and Ageing*, 35(1):5–10, 2006.

[78] N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, "Fall detection—principles and methods," in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1663–1666, Lyon, France, 22–26 August 2007.

[79] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait & Posture*, 28(2):285–291, 2008.

[80] W. H. Wu, A. A. T. Bui, M. A. Batalin, D. Liu, and W. J. Kaiser, "Incremental diagnosis method for intelligent wearable sensor system," *IEEE Transactions on Information Technology in Biomedicine*, 11(5):553–562, 2007.

[81] E. Jovanov, A. Milenkovic, C. Otto, and P. C de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, article no. 6, 2005.

[82] M. Ermes, J. Pärkkä, J. Mäntyjärvi, and I. Korhonen, "Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions," *IEEE Transactions on Information Technology in Biomedicine*, 12(1):20–26, 2008.

[83] R. Aylward and J. A. Paradiso, "Sensemble: a wireless, compact, multi-user sensor system for interactive dance," in *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 134–139, Paris, France, 4–8 June 2006.

[84] J. Lee and I. Ha, "Real-time motion capture for a human body using accelerometers," *Robotica*, 19(6):601–610, 2001.

[85] T. Shiratori and J. K. Hodgins, "Accelerometer-based user interfaces for the control of a physically simulated character," *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, 27(5):article no. 123, 2008.

[86] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[87] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, 104(2–3):90–126, 2006.

[88] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, 36(3):585–601, 2003.

[89] J. K. Aggarwal and Q. Cai, "Human motion analysis: a review," *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[90] L. Hyeon-Kyu and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, 1999.

[91] H. Junker, O. Amft, P. Lukowicz, and G. Troester, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, 41(6):2010–2024, 2008.

[92] J. C. Lementec and P. Bajcsy, "Recognition of arm gestures using multiple orientation sensors: gesture classification," in *Proceedings of the 7th International Conference on Intelligent Transportation Systems*, pp. 965–970, Washington DC, U.S.A., 3–6 October 2004.

[93] M. Uiterwaal, E. B. C. Glerum, H. J. Busser, and R. C. van Lummel, "Ambulatory monitoring of physical activity in working situations, a validation study," *Journal of Medical Engineering and Technology*, 22(4):168–172, 1998.

[94] J. B. Bussmann, P. J. Reuvekamp, P. H. Veltink, W. L. Martens, and H. J. Stam, "Validity and reliability of measurements obtained with an 'activity monitor' in people with and without transtibial amputation," *Physical Therapy*, 78(9):989–998, 1998.

[95] K. Aminian, P. Robert, E. E. Buchser, B. Rutschmann, D. Hayoz, and M. Depairon, "Physical activity monitoring based on accelerometry: validation and comparison with video observation," *Medical and Biological Engineering and Computing*, 37(3):304–308, 1999.

[96] D. Roetenberg, P. J. Slycke, and P. H. Veltink, "Ambulatory position and orientation tracking fusing magnetic and inertial sensing," *IEEE Transactions on Biomedical Engineering*, 54(5):883–890, 2007.

[97] B. Najafi, K. Aminian, F. Loew, Y. Blanc, and P. Robert, "Measurement of stand-sit and sit-stand transitions using a miniature gyroscope and its application in fall risk evaluation in the elderly," *IEEE Transactions on Biomedical Engineering*, 49(8):843–851, 2002.

[98] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. J. Büla, and P. Robert, "Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly," *IEEE Transactions on Biomedical Engineering*, 50(6):711–723, 2003.

[99] Y. Tao, H. Hu, and H. Zhou, "Integration of vision and inertial sensors for 3D arm motion tracking in home-based rehabilitation," *International Journal of Robotics Research*, 26(6):607–624, 2007.

[100] T. Viéville and O. D. Faugeras, "Cooperation of the Inertial and Visual Systems," *NATO ASI Series*, vol. F63, pp. 339–350. Berlin, Heidelberg, Germany: Springer-Verlag, 59th ed., 1990. in *Traditional and Non-Traditional Robotic Sensors*.

[101] *Proceedings of the Workshop on Integration of Vision and Inertial Sensors (InerVis)*, Coimbra, Portugal, June 2003; Barcelona, Spain, April 2005.

[102] Special Issue on the 2nd Workshop on Integration of Vision and Inertial Sensors (InerVis05), *International Journal of Robotics Research*, vol. 26, June 2007.

[103] R. Zhu and Z. Zhou, "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2): 295–302, 2004.

[104] X. Yun, E. R. Bachmann, H. Moore, and J. Calusdian, "Self-contained position tracking of human movement using small inertial/magnetic sensor modules," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2526–2533, Rome, Italy, 10–14 April 2007.

[105] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Lecture Notes in Computer Science*, vol. 3001, pp. 1–17, 2004, in *Proceedings of Pervasive Computing*. `web.media.mit.edu/~intille/papers-files/BaoIntille04.pdf`.

[106] P. H. Veltink, H. B. J. Bussmann, W. de Vries, W. L. J. Martens, and R. C. Van Lummel, "Detection of static and dynamic activities using uniaxial accelerometers," *IEEE Transactions on Rehabilitation Engineering*, 4(4):375–385, 1996.

[107] K. Kiani, C. J. Snijders, and E. S. Gelsema, "Computerized analysis of daily life motor activity for ambulatory monitoring," *Technology and Health Care*, 5(4):307–318, 1997.

[108] F. Foerster, M. Smeja, and J. Fahrenberg, "Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring," *Computers in Human Behavior*, 15(5):571–583, 1999.

[109] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, 10(1):156–167, 2006.

[110] F. R. Allen, E. Ambikairajah, N. H. Lovell, and B. G. Celler, "Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models," *Physiological Measurement*, 27(10):935–951, 2006.

[111] N. Kern, B. Schiele, and A. Schmidt, "Multi-sensor activity context detection for wearable computing," *Lecture Notes in Computer Science*, vol. 2875, pp. 220–232, November 2003, in *Ambient Intelligence*. `citeseer.ist.psu.edu/kern03multisensor.html`.

[112] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed., New York: John Wiley & Sons, Inc., 2001.

[113] A. Webb, *Statistical Pattern Recognition*, U.K.: John Wiley & Sons, Ltd., 2002.

[114] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Orlando, FL, U.S.A.: Academic Press, Inc., 2006.

[115] L. Breiman, J. H. Friedman, R. A. Ohlsen, and C. J. Stone, *Classification and Regression Trees*. Boca Raton, FL, U.S.A.: Chapman & Hall/CRC, 1998.

[116] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals*, New York: IEEE Press, 2000.

[117] T. Parsons, *Voice and Speech Processing*, New York: McGraw-Hill Book Company, 1986.

[118] Z. M. Kovács-Vajna, "A fingerprint verification system based on triangular matching and dynamic time warping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

[119] M. F. Zanuy, "On-line signature recognition based on VQ-DTW," *Pattern Recognition*, 40(3):981–992, 2007.

[120] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Moscow, Russia: Nauka, 1979. (in Russian, English translation: Springer-Verlag, New York, 1982).

[121] C. W. Hsu, C. C. Chang, and C. J. Lin, *A Practical Guide to Support Vector Classification*, National Taiwan University, Taipei, Taiwan, 2008.

[122] C. C. Chang and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[123] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New York: Macmillan Publishing, 1994.

[124] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[125] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *IEEE Transactions on Biomedical Engineering*, 56(3):871–879, 2009.

[126] N. Bowditch, *The American Practical Navigator*, Bethesda, MD: National Imagery and Mapping Agency, 1995.

[127] A. Ayliffe, "The development of airborne dead reckoning. Part I: Before 1940—Finding the wind," *Journal of Navigation*, 54(2):223–233, 2001.

[128] A. Ayliffe, "The development of airborne dead reckoning. Part II: After 1940—Staying on track," *Journal of Navigation*, 54(3):463–476, 2001.

[129] I. Skog and P. Händel, "In-car positioning and navigation technologies—a survey," *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21, 2009.

[130] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on Robotics and Automation*, 12(6):869–880, 1996.

[131] L. Fang, L. A. Montestruque, M. B. McMickell, M. Lemmon, Y. Sun, I. Koutroulis, M. Haenggi, M. Xie, and X. Xie, "Design of a wireless assisted pedestrian dead reckoning system—the NavMote experience," *IEEE Transactions on Instrumentation and Measurement*, 54(6):2342–2358, 2005.

[132] C. Fischer and H. Gellersen, "Location and navigation support for emergency responders: a survey," *IEEE Pervasive Computing*, 9(1):38–47, 2010.

[133] L. Ojeda and J. Borenstein, "Non-GPS navigation for security personnel and first responders," *Journal of Navigation*, 60(3):391–407, 2007.

[134] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[135] S. Godha and G. Lachapelle, "Foot mounted inertial system for pedestrian navigation," *Measurement Science and Technology*, 19(7):075202, 2008.

[136] A. M. Sabatini, "Dead-reckoning method for personal navigation systems using Kalman filtering techniques to augment inertial/magnetic sensing," pp. 251–268, *Kalman Filter: Recent Advances and Applications*, I-Tech, Vienna, Austria, 2009.

[137] A. M. Sabatini, "Quaternion-based strap-down integration method for applications of inertial sensing to gait analysis," *Medical and Biological Engineering and Computing*, 43(1):94–101, 2005.

[138] A. M. Sabatini, "Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing," *Sensors*, 11(2):1489–1525, 2011.

[139] J. Borenstein and L. Ojeda, "Heuristic drift elimination for personnel tracking systems," *Journal of Navigation*, 63(4):41–58, 2010.

[140] A. M. Sabatini, C. Martelloni, S. Scapellato, and F. Cavallo, "Assessment of walking features from foot inertial sensing," *IEEE Transactions on Biomedical Engineering*, 52(3):486–494, 2005.

[141] I. Skog, P. Händel, J. Nilsson, and J. Rantakokko, "Zero-velocity detection—an algorithm evaluation," *IEEE Transactions on Biomedical Engineering*, 57(11):2657–2666, 2010.

[142] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU," in *IEEE International Symposium on Intelligent Signal Processing*, Budapest, Hungary, 26–28 August 2009.

[143] O. Bebek, M. A. Suster, S. Rajgopal, M. J. Fu, X. Huang, M. C. Çavuşoğlu, D. J. Young, M. Mehregany, A. J. van den Bogert, and C. H. Mastrangelo, "Personal navigation via high-resolution gait-corrected inertial measurement units," *IEEE Transactions on Instrumentation and Measurement*, 59(11):3018–3027, 2010.

[144] Z. Sun, X. Mao, W. Tian, and X. Zhang, "Activity classification and dead reckoning for pedestrian navigation with wearable sensors," *Measurement Science and Technology*, 20(1):015203, 2009.

[145] C. I. D. Gusenbauer and J. Krösche, "Self-contained indoor positioning on off-the-shelf mobile devices," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–9, Zurich, Switzerland, 15–17 September 2010.

[146] A. M. Sabatini, "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.

[147] G. Retscher, "Test and integration of location sensors for a multi-sensor personal navigator," *Journal of Navigation*, 60(1):107–117, 2007.

[148] R. Jirawimut, P. Ptasinski, F. Cecelja, and W. Balanchandran, "A method for dead reckoning parameter correction in pedestrian navigation system," *IEEE Transactions on Instrumentation and Measurement*, 52(1):209–215, 2003.

[149] P. Aggarwal, D. Thomas, L. Ojeda, and J. Borenstein, "Map matching and heuristic elimination of gyro drift for personal navigation systems in GPS-denied conditions," *Measurement Science and Technology*, 22(2):025205, 2011.

[150] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, D. Howard, K. Meijer, and R. Crompton, "Activity identification using body-mounted sensors—a review of classification techniques," *Physiological Measurement*, 30(4):R1–R33, 2009.

[151] U. Özertem and D. Erdoğmuş, "Nonparametric snakes," *IEEE Transactions on Image Processing*, 16(9):2361–2368, 2007.

[152] J. B. Kuipers, *Quaternions and Rotation Sequences*. NJ: Princeton University Press, 1999.

[153] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM Review*, 6(4):422–430, 1964.

[154] D. T. Greenwood, *Advanced Dynamics*, U.K.: Cambridge University Press, 2006.

[155] P. L. Odell, D. Dorsett, D. Young, and J. Igwe, "Estimator models for combining vector estimators," *Mathematical and Computer Modelling*, 12(12):1627–1642, 1989.