Francisco Monteiro Santos Brás Couceiro

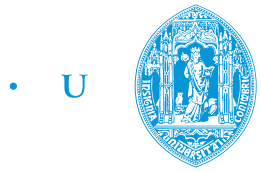
SmartLocator - Indoor Human Localization using a Smartphone

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Computer Engineering

July, 2017

· U C ·

UNIVERSIDADE DE COIMBRA

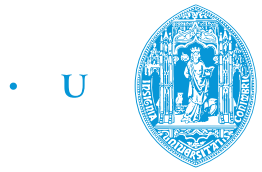


FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

SmartLocator - Indoor Human Localization using a Smartphone

Francisco Monteiro Santos Brás Couceiro

Coimbra, July 2017



SmartLocator - Indoor Human Localization using a Smartphone

Supervisor:

Prof. Doutor Rui Paulo Pinto da Rocha

Jury:

Prof. Doutora Lúcia Maria dos Reis Albuquerque Martins

Prof. Doutor Marco Alexandre Cravo Gomes

Prof. Doutor Rui Paulo Pinto da Rocha

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, July 2017

Acknowledgements

I would first like to thank my thesis supervisor Prof. Doutor Rui Paulo Pinto da Rocha, he was always available whenever I had questions about my research or writing and consistently allowed this work to be my own, while steering me in the right direction.

I would also like to acknowledge the remaining members of the AP4ISR team for their advices during my stay at the Institute of Systems and Robotics.

I will be forever grateful to my parents Vitor Couceiro and Dulce Santos, to my brother André Couceiro and my girlfriend Catarina Coimbra for supporting me throughout my life, my years of study and through the process of writing this thesis. I am also grateful to my other family members, in special to Sara Couceiro, for their support and to my four dogs for their companionship.

Finally, a very special gratitude goes out to my friends Diogo Justo, Filipe Rainho, Ricardo Mendes, Tiago Chaves and in special to João Pedro Paulo for your outstanding help and the great times we had during the writing of this thesis. Your friendship means a lot to me.

Thank you!

Abstract

Context awareness is very important for services provided by digital platforms of ubiquitous computing, being localization one of the fundamental dimensions in defining the context. There are currently more than 2.3 billion smartphone users. With the growing trend of social sharing, there are more internet users using mobile devices than personal computers. The massification of smartphone adoption has created the emerging need to acquire their users localization information. Whether it is to deliver cured content to users or to create surveillance systems, there is a new market to explore.

Being able to estimate the position of a device was the main motivation behind the development of GPS - Global Positioning System - which is the mainstream localization technology in outdoor environments used for navigation, of both cars and airplanes. However, in indoor environments, this method lack of accuracy has led to the development of alternative localization solutions. There is already a vast infrastructure of Wi-Fi network points available across the globe, which coupled with recent developments in the quality of embedded sensors on mobile devices, make these platforms a focus of pervasive computing.

The SmartLocator system offers a way to localize humans using a smartphone, in environments where a Wi-Fi infrastructure already exists. An algorithm based on the fusion of data provided by the Wi-Fi network and by the inertial sensors available on the mobile device is proposed. It is possible to estimate the position of a smartphone using a pre-populated database of received signal strength from the existing access points. This technique is called Fingerprinting. However, the variability of the Wi-Fi infrastructure and in received signals strength, due to common propagation issues, influence the precision of this method. By using data obtained from the embedded inertial sensors it is possible to continuously track displacements from a known initial position, possibly with high cumulative errors. This technique is known as Dead Reckoning. The fusion of these two sources of data improves the localization accuracy.

Keywords: Indoor localization, Inertial sensors, Wi-Fi based localization, Fingerprinting,

Dead reckoning, Smartphone localization, Data fusion.

Resumo

A sensibilidade ao contexto é de elevada importância para serviços prestados por plataformas digitais de computação ubíqua, onde a localização é uma das dimensões fundamentais desse contexto. Há mais de 2.3 mil milhões de utilizadores de smartphones. Juntamente com a crescente onda de partilha social, há mais utilizadores da internet que usam dispositivos móveis do que computadores pessoais. A massificação do uso do smartphone criou a necessidade emergente de adquirir informação de localização nestes dispositivos. Seja para fornecer conteúdos curados aos utilizadores ou para sistemas de alerta/monitorização, há um novo mercado por explorar.

Conseguir estimar a localização de um dispositivo foi a grande motivação para a criação do GPS - Global Positioning System - que é a principal tecnologia de localização em ambientes exteriores utilizada para navegação, tanto de automóveis como de aviões. Porém, em locais interiores, a falta de precisão deste método tem favorecido o aparecimento de soluções de localização alternativas. Por todo o mundo há já uma enorme infraestrutura de pontos de rede Wi-Fi disponíveis, o que aliado aos recentes avanços na qualidade dos sensores embebidos em dispositivos móveis, faz desta plataforma o foco actual da computação pervasiva.

O SmartLocator procura localizar humanos através de um smartphone, em ambientes onde exista uma infraestrutura Wi-Fi. É proposto um algoritmo de localização *indoor* baseado na fusão de dados fornecidos pela rede Wi-Fi e pelos sensores inerciais disponíveis no dispositivo móvel. A criação *a priori* de um mapa de potência de sinais de rádio emitidos pelos *access points* existentes, permite numa fase *a posteriori* a estimação da posição geográfica de um smartphone. Esta técnica chama-se Fingerprinting. Contudo, a variabilidade da infraestrutura Wi-Fi e da potência dos sinais recebida, devido a problemas de propagação comuns, influenciam a precisão deste método. Usando dados provenientes dos sensores inerciais é possível estimar deslocações relativas a um ponto cuja localização seja conhecida, ainda que com erros cumulativos elevados. Esta técnica é conhecida por Dead Reckoning.

A fusão destas duas fontes de informação aumenta a precisão de localização.

Palavras-Chave: Localização indoor, Sensores inerciais, Localização baseada em Wi-Fi, Fingerprinting, Dead reckoning, Localização de smartphone, Fusão de dados

"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time."

— Thomas A. Edison

Contents

Acknowledgements **iii**

Abstract **v**

Resumo **vii**

List of Acronyms **xv**

List of Figures **xix**

List of Tables **xxi**

1 Introduction **1**

1.1 Context 1

1.2 Motivation 2

1.3 Objectives 2

1.4 Outline of the document 3

2 Related Work **4**

2.1 Fingerprinting 4

2.2 Range-based 6

2.2.1 Trilateration 6

2.2.2 Triangulation 8

2.3 Motion Sensing 8

2.4 Vision-based 10

2.5 Hybrid 11

3 SmartLocator system **13**

3.1 Design 15

3.2 Localization Estimation Architecture 16
3.3 Fingerprinting 17
3.3.1 Training phase 18
3.3.2 Runtime phase 20
3.4 Dead Reckoning 23
3.4.1 Step detection 26
3.4.2 Step length estimation 27
3.4.3 Heading direction estimation 28
3.5 Fusion 30

4 Experimental Results **35**
4.1 Setup and method 35
4.2 Fingerprinting 40
4.2.1 Weighting influence 40
4.2.2 Orientation influence 42
4.2.3 K influence 43
4.3 Dead reckoning 45
4.3.1 Step detection 45
4.3.2 Accuracy 45
4.4 Fusion 47
4.4.1 K influence 47
4.4.2 Training points density influence 49

5 Conclusion and Future Work **51**
5.1 Conclusion 51
5.2 Future work 52

References **53**

Appendices **59**

A ITU Recommendation factors **61**

B Raw results **62**
B.1 Fingerprinting results 62
B.2 Dead reckoning results 67

B.3 Fusion results 72

List of Acronyms

GPS	Global Positioning System
Wi-Fi	Wireless Fidelity
AP	Access Point
ITU	International Telecommunication Union
KNN	K Nearest Neighbors
WKNN	Weighted K Nearest Neighbors
FSM	Finite State Machine
GSM	Global System for Mobile Communications
RSS	Received Signal Strength
SVC	Support Vector Classification
SVR	Support Vector Regression
SVM	Support Vector Machine
BMIM	Bilinear Median Interpolation Method
DCA	Dynamic Channel Assignment
BLE	Bluetooth Low Energy
LLS	Linear Least Squares
MDS	Multi-Dimensional Scaling
TOA	Time of Arrival
TDOA	Time Difference of Arrival

AOA	Angle of Arrival
RTOF	Roundtrip Time of Flight
NLOS	Non Line of Sight
UWB	Ultra Wide Band
IMU	Inertial Measurement Unit
PNS	Pedestrian Navigation System
DoF	Degrees of Freedom
ZUPT	Zero Velocity Updates
MEMS	Micro-Electro-Mechanical Systems
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
DTW	Dynamic Time Warping
WSN	Wireless Sensor Network
2D	Two-Dimensional
3D	Three-Dimensional
SLAM	Simultaneous Localization and Mapping
SfM	Structure-from-Motion
VINS	Visual-Inertial Systems
FOV	Field of View
AP4ISR	Artificial Perception for Intelligent Systems and Robotics
UI	User Interface
API	Application Programming Interface
SDK	Software Development Kit
BSSID	Basic Service Set Identifier

ENU	East-North-Up
ISR	Institute of Systems and Robotics
DSLR	Digital Single-Lens Reflex camera
Std	Standard deviation

List of Figures

2.1 Estimating user's position based on trilateration. 7
2.2 Estimating user's position based on triangulation. 8

3.1 The SmartLocator application main screen displaying the area of interest overlaid by the user estimated position. 13
3.2 The SmartLocator application settings screen show casing some fine tunable variables. 14
3.3 SmartLocator two-tier design layout. 15
3.4 SmartLocator front-end layers and their relationship. 16
3.5 SmartLocator localization estimation architecture. 17
3.6 Fingerprinting architecture. 18
3.7 Fingerprinting layout example with three access points. 19
3.8 Fingerprinting heading categorization intervals. 19
3.9 Fingerprinting runtime regression example. 22
3.10 Fingerprinting runtime regression example using WKNN. 23
3.11 Dead reckoning tracking basis. 24
3.12 IMUs intertial frame. 24
3.13 User handling smartphone with arbitrary orientation. 25
3.14 East-North-Up frame explanation. 25
3.15 Dead reckoning architecture. 26
3.16 Vertical linear acceleration sensed on the Up axis while walking. 27
3.17 Hip vertical displacement in different walking phases. 28
3.18 Heading angle representation. 29
3.19 Complementary filter architecture. 29
3.20 Natural heading offset problematic representation. 30
3.21 Data fusion basic FSM. 31

3.22 Data fusion example using just the dead reckoning technique. 32
3.23 Absolute position rejection based on a distance threshold. 33
3.24 Data fusion drift correction example. 33
3.25 Data fusion extended FSM with confidence system. 34

4.1 Experiments common walking pattern. 36
4.2 Plot of an experiment estimated positions. 37
4.3 Matlab application developed to inspect experiments recorded videos. 38
4.4 Smartphone holding criteria for improved inspection accuracy. 39
4.5 Homography matrix estimation between the layout image and the physical room. 39
4.6 Localization of the train points used in fingerprinting experiments. 40
4.7 Fingerprinting non weighted experiment results. 41
4.8 Fingerprinting weighted experiment results. 42
4.9 Fingerprinting with orientation category matching experiment results. 43
4.10 Fingerprinting using K as 3 experiment results. 44
4.11 Dead reckoning results. 46
4.12 Fusion using K as 2 results. 47
4.13 Fusion using K as 3 results. 49
4.14 Radio maps with different densities. 50

B.1 Experiments room coordinates frame. 62

List of Tables

2.1 Fingerprinting techniques accuracy comparison. 6
2.2 Motion sensing techniques accuracy comparison. 10

4.1 KNN vs WKNN performance comparison summary. 41
4.2 Orientation matching vs Without orientation matching performance comparison summary. 43
4.3 K=2 vs K=3 performance comparison summary. 44
4.4 Step detection results. 45
4.5 Dead reckoning vs Best fingerprinting setup comparison. 46
4.6 Fusion using K as 2 vs Dead reckoning vs Best fingerprinting setup comparison. 48
4.7 Fusion using a lower density radio map vs Fusion using regular density radio map vs Fusion using higher density radio map comparison. 50

A.1 Power loss coefficients, N, for indoor transmission loss calculation. 61
A.2 Floor penetration loss factors, Lf (dB) with n being the number of floors penetrated, for indoor transmission loss calculation. 61

B.1 KNN error results (cm). 62
B.2 WKNN error results (cm). 63
B.3 Using orientation category while matching results (cm). 64
B.4 K = 3 results (cm). 65
B.5 Dead reckoning results (cm). 67
B.6 Fusion using K as 2 results (cm). 72
B.7 Fusion using K as 3 results (cm). 79
B.8 Fusion using low density radio map results (cm). 88
B.9 Fusion using high density radio map results (cm). 93

1 Introduction

1.1 Context

Localization solutions are extensively used every day in navigation systems, such as in planes, boats or vehicles. The most used technology to provide absolute localization is the Global Positioning System - GPS. Its capability of providing latitude, longitude, and altitude information has also enabled other types of applications to enhance their knowledge about their user or context. For instance, compasses that use true heading as opposed to the traditional magnetic field approach.

The Global Positioning System is able to localize a user with an accuracy of 5 meters when outside and line of sight to at least four GPS satellites is available. However, large objects such as tall buildings can impair its accuracy. Furthermore, GPS is not accurate in indoor environments. As a result, applications that rely on this type of localization will not perform accurately in these conditions.

As people tend to spend more time at home, malls, museums and airports it is of great interest to overcome this accuracy challenge. Also, the advent of ubiquitous computing applications demands for new and improved ways to obtain information about their user and the surrounding context, with localization being one of the fundamental dimensions in defining it.

Smartphones gained huge popularity in a world where communication is becoming more and more fundamental. Their massive adoption presents an opportunity to use them as the substitute candidate in indoor positioning systems. Using smartphones may help to localize elderly people in disaster situations, as a guidance while visiting a museum or even as direct replacement for GPS in situations where it fails . Therefore, researchers have put much effort in this topic, presenting increasingly accurate indoor localization algorithms.

1.2 Motivation

The mainstream localization method GPS is not capable of acceptable accuracy in indoor environments, as it requires line of sight to at least four GPS satellites. Using smartphones for accurate indoor localization creates a new topic in mobile services, allowing new opportunities to enhance the user experience in such environments.

There are currently more than 2.3 billion smartphone users, and the mobile applications market is continuously growing. Across the globe, there is already a vast infrastructure of Wi-Fi network points available that, coupled with recent developments in the quality of embedded sensors on mobile devices (smartphones and tablets), makes these platforms a focus of ubiquitous computing.

Available indoor localization solutions rely on different types of sensors, such as Wi-Fi network cards, GSM antennas or inertial sensors such as accelerometers and gyroscopes. Solutions based on Wi-Fi have proved great accuracy given its simple implementation. However, location estimation is not available instantaneously, taking a couple of seconds. The opposite happens with solutions based on inertial sensors. It is an interesting topic to explore the complementarity of Wi-Fi based algorithms with the instant availability of the ones based on inertial sensors.

1.3 Objectives

This work addresses the localization problem with a hybrid approach. Both Wi-Fi and inertial sensor data can be collected from the user's smartphone. Estimating the user's location using only Wi-Fi measurements or just inertial sensors readings, individually, does not achieve the desired accuracy.

The main goal is to develop an Android application capable of locating the user in environments where a Wi-Fi infrastructure exists. The core algorithm fuses both Wi-Fi and inertial sensor data in order to improve the overall indoor localization accuracy, using only the smartphone capabilities and its computation power. The resulting Android application should be able to display an indoor layout map representing the area of interest overlaid by the user estimated location. Also, there is one sub goal: the algorithm should be suitable for mobile use, such that the device's computation power and battery consumption limitations are taken into account.

1.4 Outline of the document

This document is divided into 5 chapters. In chapter 1, the context and motivation that led to this research work are introduced and the main objectives are presented. In chapter 2, a literature review regarding indoor localization using smartphones can be found. Chapter 3 begins by outlining the overall system architecture and further describes the proposed approach in detail. The results are presented and discussed in chapter 4. Finally, conclusions and future work are available in chapter 5.

2 Related Work

This chapter reviews the main techniques used to address the indoor localization problem throughout the literature: fingerprinting, range-based, motion sensing and vision-based. Hybrid approaches, where two or more single techniques are combined to achieve better results, are also reviewed, in the end.

2.1 Fingerprinting

Fingerprinting or scene analysis is a technique which uses the received signal strength (RSS) in order to estimate user's location. This solution is divided into two phases: training and runtime. During the training or off-line phase the site is scanned with a Wi-Fi enabled device, recording in each position the Wi-Fi signal strength from multiple access points (APs), possibly in different channels, as proposed in [1]. These readings are then stored in a radio map database. Usually, in each position several measurements are taken and averaged, per access point. Later, during the runtime or on-line phase, the current signal reception measurements are matched against all the readings available in the radio map. The most identical stored readings yield the user estimated position. Some approaches use the smartphone to store the radio map database and run localization algorithms locally, other use an external server. In [2] the localization algorithm is implemented using a best matching approach, where the provided RSS measurement is used to compute the distance to each one in the radio map using an Euclidean metric. The position associated with the stored measurement that is closest to the provided measurement is chosen. In [1] the Sparse Bayesian Learning paradigm is used during the runtime phase to enhance localization precision, instead. Support Vector Classification (SVC) of multiple classes and Support Vector Regression (SVR) have been used in [3] with success. Support Vector Machine (SVM) algorithms also have been used in [4].

The training phase plays an important role on this type of technique. It can be a labour-

intensive process and its thoroughness can have nearly direct impact on the final accuracy. Some authors have presented solutions to reduce the training time by applying completion algorithms to the collected data. This way, instead of scanning the whole interest area, one can scan it partially. In [4], using the bilinear median interpolation method (BMIM), the radio map is completed while retaining the accuracy of user localization. In [1] another map completion algorithm is proposed, using Matrix Completion.

Traditional APs use a single channel for communication. In recent years, AP designers introduced Dynamic Channel Assignment (DCA) strategies to improve network capacity [5]. These strategies introduce a new feature where APs change their operating channel over time. An AP operating channel affects the RSS measurements, as stated in [6]. Building a fingerprint radio map using only one channel would render the training data not useful, as a smartphone during the runtime phase could be assigned a different channel, and the localization accuracy would drop significantly. The influence of channel selection is taken into account in [1] and confirms that radio maps collected with only one channel communications are not as accurate.

Several other research groups have explored the use of RSS coupled with fingerprinting techniques: RADAR [7], which is one of the first systems to ever implement this approach, capable of locating the user with accuracies of 2 to 3 meters; Horus [8], an improvement of the RADAR system, making use of probabilistic analysis; Compass [9], which leverage object orientation to improve localization precision.

Besides Wi-Fi technologies, other radio-frequency technologies have been utilized with RSS fingerprinting techniques. In [10] Bluetooth LE is used. The author implemented a deterministic distance estimation approach based on the K-Nearest Neighbors (KNN) [11] method and K-means clustering algorithm, and a probabilistic approach using the Naive Bayes classifier. Also, using Zigbee technology [12], a fingerprinting technique merged with other algorithms such as gradient-based search, the linear least squares (LLS) approximation and multidimensional scaling (MDS) methods, was applied, claiming accuracies under 1.25m.

Table 2.1 compares all these techniques based on their accuracy.

Table 2.1: Fingerprinting techniques accuracy comparison.

Authors	Technique	Accuracy
Sofia Nikitaki et al. [1]	Multi-Channel fingerprinting, matrix completion and Bayesian Sparse Learning	2-3 m
Eladio Martin et al. [2]	Fingerprinting using only a smartphone	1.5 m
Mauro Brunato et al. [3]	Fingerprinting with SVM	3 m
Yu Feng et al. [4]	Fingerprinting with SVM	1.5 m
Paramvir Bahl et al. [7]	RADAR	2-3 m
Moustafa Youssef et al. [8]	Horus	1-3 m
Thomas King et al. [9]	Compass	1.65 m
Disha Adalja et al. [10]	Fingerprint with Bluetooth	1.5-2 m
Shih-Hau Fang et al. [12]	Fingerprint with ZigBee	1.25 m

Fingerprinting techniques, besides requiring a labour-intensive training phase, require a particular infrastructure. APs may have to be deployed in order to be used as RSS sources. However, nowadays almost all buildings have an existing Wi-Fi infrastructure. Hence the popularity of this type of localization methods.

2.2 Range-based

Range-based techniques use location metrics such as received signal strength (RSS), time of arrival (TOA), time difference of arrival (TDOA), and angle of arrival (AoA) to estimate the distance between two nodes. Different algorithms implement this technique to estimate the user location, such as the trilateration and triangulation methods.

2.2.1 Trilateration

Trilateration uses the RSS metric to estimate the user's position by computing the distance to at least three reference points. GPS implements this principle where the reference points are satellites, instead of terrestrial beacons. The distance is obtained by applying path-loss models which account for signal strength loss due to free-space loss, multi-path fading, reflection and refraction, such as the mainstream logarithm distance path-loss model or the International Telecommunication Union (ITU) model. The ITU's P.1238 recommendation [13] presents the model as shown in 2.1.

$$L = 20 \log f + N \log d + L_f(n) - 28 \tag{2.1}$$

L denotes the path loss, in decibel units (dB), f the frequency of transmission, in megahertz (MHz), N the distance power loss coefficient, d the distance, in meters, n the number of floors between the transmitter and receiver, and L_f the floor loss penetration factor. Both N and L_f are empirical values. N expresses the loss of signal power due to travel distance. L_f depends on the number of floors the signal has to penetrate, and models the loss of power signal due to floor penetration. Tables A.1 and A.2 express different values for these factors in various environments and for different transmission frequencies.

For example, using GSM-900 signals (900 MHz), in an office with 1 floor, $n = 1$, the power loss factor, N, is 33 and the floor penetration factor, L_f, is 9 dB, the distance can be obtained by: $d = 10^{\frac{L-40}{33}}$. The path loss, L, is the difference between the transmitted power and the received power.

Instead of using RSS measurements, time of arrival (TOA) or time difference of arrival (TDOA) measurements are often used, too. The distance is then obtained by multiplying the signal propagation velocity and the travel time. Some systems use the roundtrip time of flight (RTOF) or the received signal phase method metric.

By computing the distance to at least three reference points, circles can be drawn centred in each point with radius equal to the distance between that point and the user. The intersection point of the three circles is the user estimated position. Figure 2.1 represents this technique.

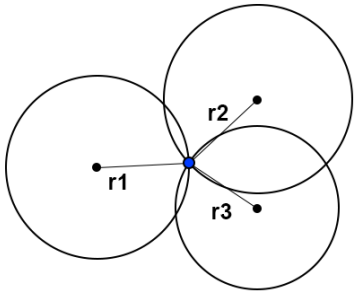


Figure 2.1: Estimating user's position based on trilateration.

The SmartLOCUS system created by HP [14] uses ultrasound RTOF measurements to estimate the relative location between several interconnected nodes. This system has accuracies from 2 to 15 centimeters, although not suitable for user localization.

2.2.2 Triangulation

Triangulation estimates user's position by computing angles relative to multiple reference points, as illustrated in Fig. 2.2.

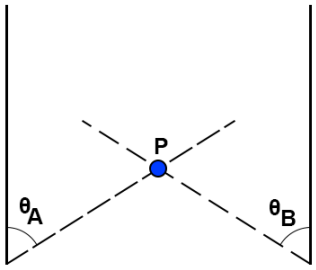


Figure 2.2: Estimating user's position based on triangulation.

Different antenna technologies exist to compute the angle between the receiving sensor and the emitter, θ_A and θ_B. Mechanically-agile directional antennas are able to adjust to the point of highest signal strength. Other technology uses antenna arrays, where TDOA measurements in each element of the array can be converted to an AOA measurement. This technique has the advantage of only needing two reference points to estimate 2D positions. However, it does not perform well on non line of sight (NLOS) situations, such as indoor environments. Another disadvantage is the use of complex hardware, making it not suitable for a mobile application.

The Ubisense system [15] considers the Ultra Wide Band (UWB) technology and the use of angle of arrival to estimate user's position. This system claims accuracies between 0.15-1 meters, even under complicated construction environments.

In [16] the teleoperation of Unmanned Ground Vehicles (UGVs) is aided by a graphical representation of the RSS gradient at the UGV location. The gradient estimation is computed by using multiple receivers with directional antennas. With this information the UGV operator is aware of low wireless connectivity areas and is able to adjust its trajectory accordingly.

2.3 Motion Sensing

Inertial measurement units (IMUs) such as motion sensors (accelerometers) and rotation sensors (gyroscopes) provide useful data capable of tracking a user by continuously estimating displacement from a known initial position. Some approaches based on inertial sensors are proposed in [17–20], using the so-called pedestrian navigation systems (PNS). These have the

added value of not relying on infrastructure assistance, as no access points or off-line/training phase are required.

These solutions do not generally achieve the desired meter-level accuracy, because they rely on temporal integration of linear acceleration and/or angular velocity which induce high cumulative errors. Drift errors can exceed 150 meters after just 60 seconds of operation [21]. However, in short time intervals, accelerometers and gyroscopes provide accurate measurements.

Typically, external accelerometers and gyroscopes are carried by the user, in these dead reckoning approaches. In [18], a 6 degrees of freedom (DoF) inertial sensor suite is equipped in user's foot. To reduce cumulative errors, this approach implements an indirect Kalman filter and a zero velocity updates (ZUPT) detection algorithm. Using the zero velocity updates, the known acceleration direction due to gravity is used to correct the previous accumulated drift errors. Another foot-mounted approach [20] also uses ZUPT to reduce drift errors. However, a particle filter is utilized to accommodate building map constraints. Moreover, this building map is a 2.5D representation, which accounts for walls and steps, where each object has its height recorded.

Nowadays smartphone sensors are capable of acceptable accuracy, since IMUs can now be packed into small circuits thanks to micro-electro-mechanical systems (MEMS) technology improvements. The fusion of accelerometer and magnetometer data highly reduces drift errors [21]. Magnetometers operate with almost no drift, however quick changes in orientation are not accurately sensed. In [22] a new approach to step detection and user heading inference is proposed, achieving meter-level positioning accuracy. Using the device's accelerometer, magnetometer and gyroscope, this system can detect steps and their length through a personalized step model derived from the last taken step, estimate the user heading, and compute the overall position with a particle filter algorithm.

The step detection algorithm is fundamental in these approaches. Accelerometers have been widely used for this purpose. However, due to gravity, accelerometer data has to be compensated. It is also common to filter the data using finite impulse response (FIR) or or infinite impulse response (IIR) filters. In [17] the acceleration signal vector magnitude is filtered using a FIR averaging filter with a history of 8 samples. After analyzing data from a few steps, the author presents an algorithm based on signal vector magnitude thresholds that is capable of detecting steps. Also, in [22], a low pass FIR filter with cut-off frequency set to 3 Hz is used to remove high frequency noise and spikes from raw accelerometer data. Using two thresholds, almost every false peak caused by acceleration jitters that are either

too small in magnitude or to short in time duration can be filtered out. This algorithm further improves the detection accuracy using heuristic constraints and dynamic time warping (DTW) validation. In addition to this, in [23] the building map constraints are fused into inertial measurements to filter out the impossible motion trends. Here, the fusion technique uses a particle filter.

Table 2.2 compares these motion sensing techniques based on their accuracy.

Table 2.2: Motion sensing techniques accuracy comparison.

Authors	Technique	Accuracy
Lasse Klingbeil et al. [17]	WSN with mobile nodes worn by the user	3.5 m
Oliver Woodman et al. [20]	6-DOF foot-mounted IMU, ZUPT	1 m
Fan Li et al. [22]	PSN using smartphone	2-4 m

2.4 Vision-based

Vision-based techniques make use of computer vision techniques to localize a user by using data acquired from cameras. There are several types of cameras and different algorithms to take advantage of each one. As mobile devices cameras are getting better and consumer level 3D cameras becoming cheaper, a majority of the existing techniques rely on 2D images or 3D RGB-D (color and depth) data.

In [24] a single 2D monocular camera is used to estimate the position and orientation for vehicle navigation. This system requires a training phase to pre-populate a database that correlates physical location with image holistic features and 3D landmarks along the course. To estimate the position, in runtime, the current frame holistic feature vector is computed and matched with every feature vector present on the database. The most suitable one determines the initial global position. Then, 3D landmarks are used to estimate the camera pose (position and orientation – 6 DoF) with high precision.

Several solutions rely on available or pre-populated databases that contain 3D landmark data or building models such as facade image features/layout [24] [25]. Another approach to this method is the Simultaneous Localization and Mapping (SLAM) technique aided by vision. In [26] the object recognition algorithm (SIFT) [27] is used to continuously maintain a database of unique visual landmarks. Using a particle filter to perform localization this system presents good accuracy and can recover from "kidnapping" scenarios (situations

where the robot position is changed by external mechanisms, for instance being lifted up and moved by a person).

Recent progress in Structure-from-Motion (SfM) algorithms allow impressive localization performance, but rely on powerful and memory intensive local image descriptors such as SIFT [27] to establish 2D-3D matches between image features and scene points. In [28], a solution to localize users carrying smartphones and take advantage of their camera is proposed. By using an in-device map (database) that is constantly updated using keyframe-based SLAM [29], the system tracks the camera locally. SfM algorithms running on external and powerful servers provide global position estimates for the local keyframes. Then, the local and global estimates are aligned to provide the best 6 DoF estimate possible.

2.5 Hybrid

Although single technique solutions already have accuracies of few meters and even in the centimeters range, researchers have explored fusion techniques that merge multiple sources of data to address the localization problem. The main goal is to improve the overall accuracy, but also to increase each system robustness by making them not so dependent on a certain type of localization method.

This technique is used in [23] to increase the algorithm convergence rate, using site fingerprints and motion sensing. Only few training readings are recorded: one per room/corridor. In an initial phase, Wi-Fi is used to achieve room-level localization, matching the nearest training location in signal space. Afterwards, inertial sensors data is used to detect steps and heading angle. Along with a particle filter, it is possible to estimate the user's location, merging accelerometer, compass and Wi-Fi RSS measurements. In some situations, where the user experiences a narrow turn, multi-clustered particles may appear. In this case, an on-line room-level localization algorithm can filter out undesired clusters.

In [30] another hybrid solution is presented, where robots know their position based on an off-line created map (fingerprint) and estimate user's position through acoustic relative ranging. The robots are equipped with Kinect vision sensors and are able to follow humans. Each user carries a smartphone that plays a pre-designed beep file in a known manner, and simultaneously records the received beep files. Then, the files are sent to the robot for relative position estimation by applying a ranging algorithm, such as time of arrival (TOA).

Also, visual-inertial systems (VINS) explore the fusion between SLAM algorithms and PNS based on motion sensors. In [31] the presented algorithm is capable of global localization

for an autonomous aerial robot. By using offline-generated 3D models of buildings, in runtime images from one fisheye camera can be used to match them. Using IMUs to stabilize the image allows the use of just one camera with a large field of view (FOV).

Another hybrid system [25] fuses data from vision, GPS and odometry to achieve better results. This is an expansion to the AVENUE [25] system by integrating vision in the pipeline. Vision is used to estimate the robot pose whenever the confidence in pose estimation based only on GPS and odometry is low. Using the Canny edge detector [32] and a RANSAC [33] approach the system is able to match a certain building model against a pre-populated database and query its current location.

A hybrid approach was followed in this work in order to take advantage of the Fingerprinting and Motion Sensing techniques. Fingerprinting is a popular technique and its low complexity absolute position estimation algorithm is ideal for low computationally power smartphones. The recent developments in the quality of embedded sensors of mobile devices also enable motion sensing techniques to be further explored. On the other hand, range-based techniques may require external hardware other than the handheld device to deliver high accuracy estimations. And vision-based techniques perform complex and computationally expensive algorithms that would fail in low cost smartphones.

3 SmartLocator system

The SmartLocator is a localization system for the Android platform and was developed using the Java language. It is able to estimate the user location in indoor environments by acquiring and fusing data provided by the Wi-Fi network, using the smartphone embedded network card, and by the IMUs, using the embedded accelerometer, magnetometer and gyroscope.

From the user perspective, the SmartLocator application only has two main interactive screens. The initial and main screen displays a layout of the area of interest (museum, school, airport, etc) and a virtual indicator representing the user estimated position. Fig. 3.1 shows the main screen of the system user interface (UI) while localizing its user in the AP4ISR laboratory.

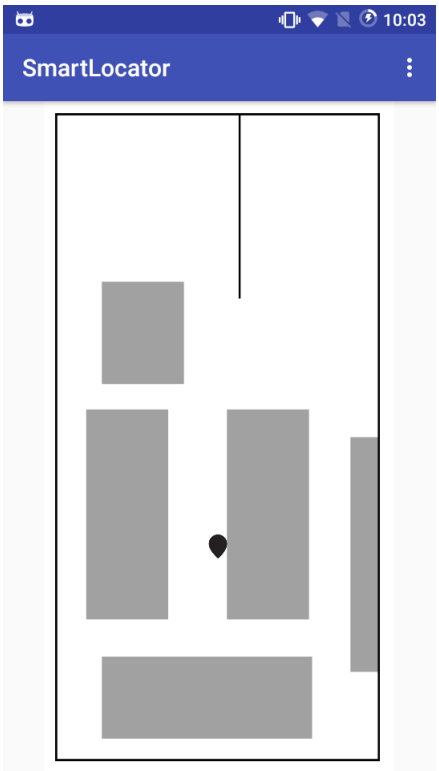


Figure 3.1: The SmartLocator application main screen displaying the area of interest overlaid by the user estimated position.

The other screen is the settings one, which is accessible through the common menu available on Android applications. In this screen the user is able to fine tune several aspects of the SmartLocator localization system layer, as can be seen in Fig. 3.2.

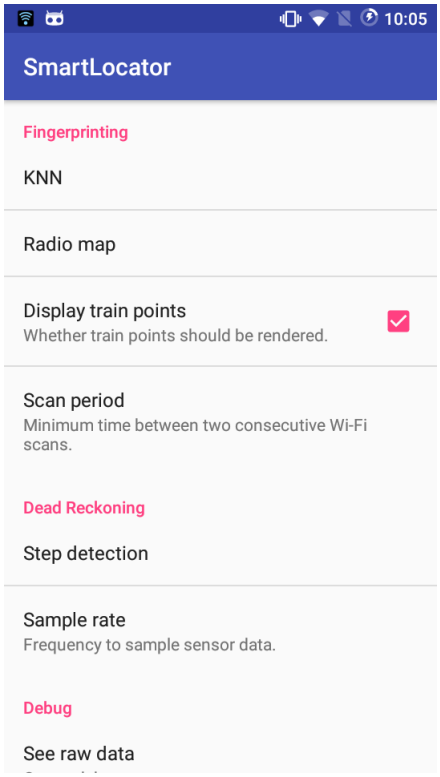


Figure 3.2: The SmartLocator application settings screen show casing some fine tunable variables.

By using the Android platform the development of the SmartLocator application is eased, as the SDK[1] is extremely well documented and offers already created APIs to access embedded sensors and extract data from them. This way, one can focus on the design and implementation of the inner algorithms, rather than complex data acquisition.

Also, the Android SDK offers a wide range of UI components, called Widgets. The SmartLocator application makes extensive use of these components in its UI layer, in order to display the area of interest layout overlaid by the user estimated position.

The remainder of this chapter describes the overall system design and architecture, using a top down approach. The several layers that compose the SmartLocator application are presented and discussed in section 3.1. In section 3.2 each component of the localization system layer is further detailed.

[1] `https://developer.android.com/sdk/`

3.1 Design

In order to achieve the desired goal of displaying the area of interest layout overlaid by the estimated position, as well as allowing different users (with different smartphones) to take advantage of the system, the interest areas layouts need to be available online. By centralizing this information, different users can navigate the same area using different smartphones. With that in mind the system was designed with two different tiers: the back-end tier and the front-end tier. Fig. 3.3 represents the top-level layout of the SmartLocator system. The back-end tier is powered by a Node.js[2] server and a MongoDB[3] database. Its main purpose is to serve the front-end tier with the different interest areas available and also to store relevant fingerprinting data. The front-end tier runs in smartphones and is where all the localization logic is implemented.

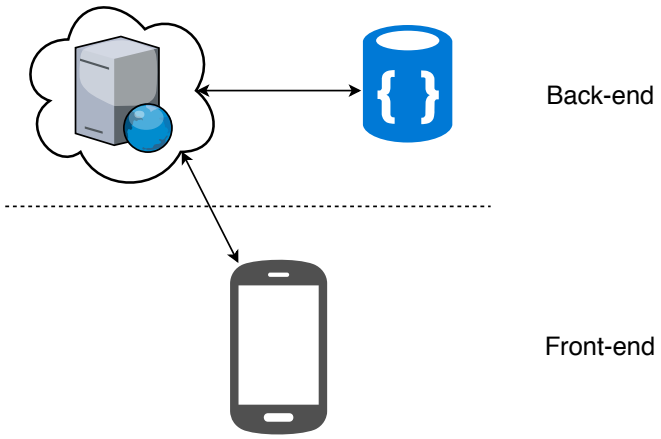


Figure 3.3: SmartLocator two-tier design layout.

Some solutions use cloud-computing power to perform complex and computationally expensive localization algorithms [1] [10]. The SmartLocator systems is able to estimate its user position using only the device capabilities. Therefore, the back-end tier only provides support for the front-end tier in terms of image assets and shared information among different devices/users.

[2] https://nodejs.org
[3] https://www.mongodb.com

The front-end tier is composed by four layers: data acquisition, localization, communication and UI. Each layer has a different purpose:

Data acquisition Is capable of extracting sensor data in a meaningful way.

Localization Is the core of the system and contains all algorithms responsible for the user localization.

Communication Is responsible for Wi-Fi data transactions between the front-end and back-end tiers.

UI Is the user interface of the system.

The four layers are carefully integrated with each other, as shown in Fig. 3.4.

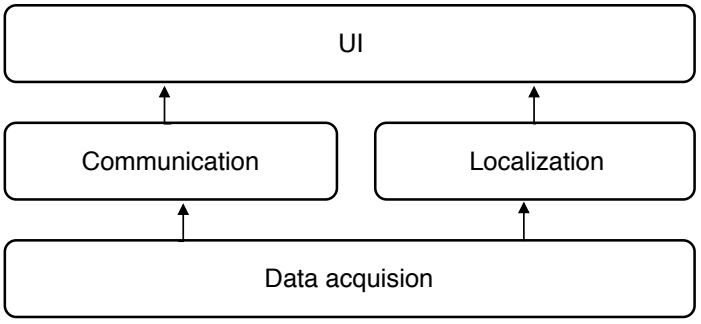


Figure 3.4: SmartLocator front-end layers and their relationship.

Data acquisition provides both IMUs and Wi-Fi readings to the upper level layers: localization and communication. The communication layer also uses data acquisition as it implements a socket connection which when active is used for debugging purposes. Other than that the communication layer is responsible for retrieving/storing data or assets from/to the back-end tier, which is tied to the UI layer. Position estimation is performed in the localization layer. As it only implements mathematical algorithms, its implementation does not depend on the Android SDK. It is a design option that allows this layer to be used in other devices other than Android powered smartphones, as long as they support Java.

3.2 Localization Estimation Architecture

The desired estimated position is obtained by fusing two types of localization techniques: fingerprinting and dead reckoning. As explained in chapter 2 these techniques differ a lot.

Fingerprinting solutions generally achieve better accuracy and deliver absolute positioning, but can take a few seconds to estimate a new position. On the other hand, dead reckoning systems are much more dynamic, but can drift over time.

The proposed localization solution is able to take advantage of the best features of each technique. Dead reckoning instant availability is preserved, as well as fingerprinting absolute positioning. So, after fusing, the SmartLocator application can localize its user in a dynamic drift-free manner. Fig. 3.5 represents the top level architecture of this solution.

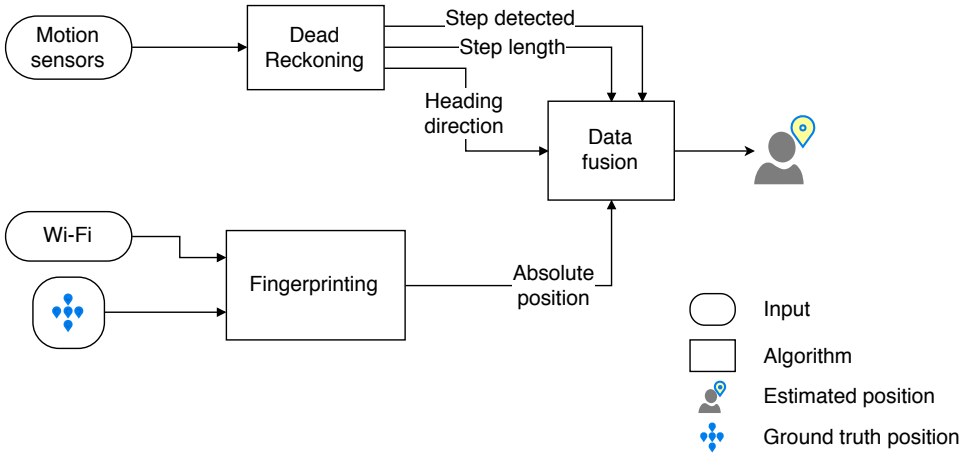


Figure 3.5: SmartLocator localization estimation architecture.

Dead reckoning techniques are based on IMUs, also known as motion sensors, namely accelerometers, magnetometers and gyroscopes. Those are also the sensors used by this solution. The dead reckoning component is capable of detecting steps, their length and the user heading direction (angle). In section 3.4 its implementation is further detailed.

Fingerprinting techniques operate in two distinct phases. An initial phase allows the population of a radio map (database) to be later used, in the runtime phase, to localize the user. In the training phase both Wi-Fi RSS values and ground truth user positions are stored in the database. Later, in the runtime phase, only Wi-Fi is used to estimate the user position. Both phases implementations are discussed in the subsections 3.3.1 and 3.3.2.

Lastly, in order to benefit from the two techniques, the data fusion component implements an opportunistic fusion algorithm further detailed in section 3.5.

3.3 Fingerprinting

Fingerprinting is a technique that correlates the RSS value from various available Wi-Fi access points with the physical position of the Wi-Fi receiver sensor. In this case, the sensor

is embedded in a smartphone and thus also corresponds to the user physical position.

In order for the smartphone to be able to estimate its position using this technique, there must be a radio map of the area of interest. The radio map is a database with only one table that in each entry stores a physical position and a vector of RSS values. Having a populated radio map, the estimated position can then be obtained by matching a query set of RSS values with the ones available. The query RSS values correspond to the Wi-Fi APs that are sensed in the user unknown position. Fig. 3.6 represents the architecture of this technique and how the training and runtime phases take place.

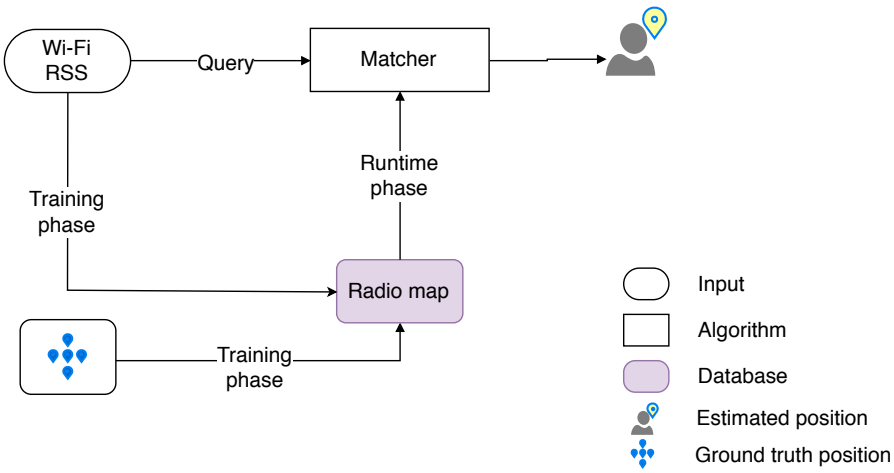


Figure 3.6: Fingerprinting architecture.

3.3.1 Training phase

The process in which the radio map database is populated is often called training phase. In this phase, the user carrying a smartphone with a Wi-Fi sensor can thoroughly survey the area of interest and in each position scan for the sensed Wi-Fi APs. Each scan, i, in each position, p_i, will produce a set of n Wi-Fi RSS values (one for each sensed AP), r_i, that are related to that physical position. The set of RSS readings, r_i, is called fingerprint and different physical positions have different fingerprints. For each scan a tuple in the form of (p_i, r_i) is recorded in the radio map.

Depending on the physical position its fingerprint vector may vary in length, as a result of defective signal propagation such as a wall attenuating the signal strength of a certain AP. Also, in the case of sensing only one or two different APs there may be locations within the area of interest that have similar fingerprints, potentially leading to incorrect position estimation. Fig. 3.7 shows an example room with three existing APs in which these situations may occur.

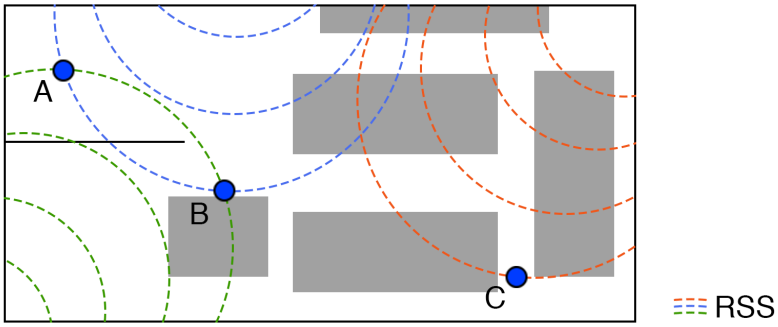


Figure 3.7: Fingerprinting layout example with three access points.

Points A and B may have similar fingerprints even though they do not share the same physical location. On the other hand, this area of interest has three access points, but point C only senses one of them. In order to overcome these issues each fingerprint data was extended to also include the user orientation, namely the smartphone orientation against the magnetic north. Subsection 3.4.3 of the Dead Reckoning chapter further details how the orientation is obtained. The heading angle between the magnetic north and the smartphone orientation is classified into four categories, it can either be: north, east, south or west. Fig. 3.8 summarizes how the heading angle categorization is performed. The area labeled by 1 corresponds to north, by 2 to east, by 3 to south and by 4 to west.

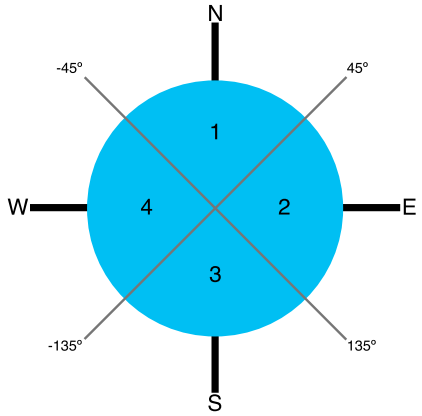


Figure 3.8: Fingerprinting heading categorization intervals.

Each fingerprint acts as a feature of that physical location, so the more information they have the more unique each might be. Having unique fingerprints is the main goal in fingerprinting techniques as they directly relate to their physical location counterpart. With the addition of smartphone orientation to the fingerprint data, each radio map entry, i, is now of the form (p_i, θ_i, r_i) where θ_i represents the orientation category in which that position was scanned. Another important information regarding each RSS value, r_{ij}, is what AP originated it. So, each RSS value is now extended to also include the basic service

set identifier (BSSID) [34] of its AP which is unique: $r_{ij} = \{RSS, BSSID\}$. In summary, each radio map entry, rm_i, includes the physical position in which it was scanned, p_i, the orientation category of the smartphone, θ_i, and a vector of n tupples with the RSS value and the BSSID of the AP that originated it, r_i, regarding the n sensed APs in that physical position:

$$rm_i = \{p_i, \theta_i, [\{RSS_1, BSSID_1\}, \{RSS_2, BSSID_2\}, \ldots, \{RSS_n, BSSID_n\}]\}$$

The training phase is crucial in the overall performance of this technique. Several solutions [1] [4] use completion algorithms to further populate the radio map without scanning the entire area of interest. These algorithms are useful in situations where fingerprinting is the only localization technique used and thus the radio map points density should be bigger. In this work, as it is not only based on fingerprinting, these algorithms were not implemented in order to explore the use of dead reckoning in between fingerprints.

3.3.2 Runtime phase

The runtime phase is where the actual position estimation is performed. After training the system, estimating the user position is a matter of matching the most up-to-date fingerprint and smartphone orientation with the entries available in the radio map database. The most up-to-date fingerprint and smartphone orientation are called the query reading. It is a set of RSS values and corresponding BSSID sensed from the available APs, r_q, in the current unknown location and the smartphone orientation category, θ_q. Given that each radio map entry, i, is of the form (p_i, θ_i, r_i), the estimated position can be obtained by matching the query reading (θ_q, r_q) with every radio map entry fingerprint data (θ_i, r_i). The physical position, p_i, associated with the fingerprint (θ_i, r_i) that is most identical with the query (θ_q, r_q) is the estimated position.

An important aspect regarding Wi-Fi RSS data acquisition is that the Android SDK is not instantaneous while delivering new RSS values from the Wi-Fi network card. Knowing that users will be potentially moving around in the area of interest and the collected fingerprints available on the radio map were recorded in a static manner, the RSS values are first processed before matching. The query RSS vector is the result of a three time arithmetic average. Given that a new set of RSS values is only available after a few milliseconds, the resulting averaged query vector is only available after a second or two, depending on the Android operating system.

Several fingerprinting solutions rely on classification algorithms such as KNN or SVM to effectively match a query against the whole radio map. In this work the KNN classification algorithm is used. The reason behind this choice is that KNN is computationally lightweight and yet it has proven high accuracy [2] [35]. The KNN algorithm further enhances position estimation by regression fitting the K most suitable positions instead of selecting just the most identical one. K is a constant that controls how many possibilities from the radio map will be used to estimate the final position.

In order to match the query reading with the ones available in the radio map, it is first computed the distance in signal space between the query RSS values, $r_{q_{RSS}}$, and each one in the radio map $r_{i_{RSS}}$. The distance, d_i, is obtained by using the $L2$ (Euclidean) metric:

$$d_i = \sqrt{(r_{q_{RSS}} - r_{i_{RSS}}) \cdot (r_{q_{RSS}} - r_{i_{RSS}})} \tag{3.1}$$

Special care is needed to handle the subtraction of the two vectors, $r_{q_{RSS}}$ and $r_{i_{RSS}}$, because first they need to be sorted by AP BSSID, so that only RSS values corresponding to the same AP are subtracted from each other.

After having the distances between the query and radio map values, the estimated position is obtained by averaging the K most suited positions. To select the K most appropriate positions, the distances are sorted in ascending order. Let $D = [d_2, d_4, d_7, d_1, \ldots, d_3]_n$ be the vector containing the n ordered distances and $D_K = [d_3, \ldots, d_5]_K$ the subset of D corresponding to the first K values: the K most appropriate positions are the ones corresponding to those distances, $P_K = [p_3, \ldots, p_5]_K$. Finally, the estimated position, p_e, is obtained by computing the centroid of the positions in P_K:

$$p_e = \Big(\frac{\sum_{i=1}^{K} P_{K_{ix}}}{K}, \frac{\sum_{i=1}^{K} P_{K_{iy}}}{K}\Big) \tag{3.2}$$

Different and distinct points can have identical fingerprints, depending on the Wi-Fi infrastructure. By selecting K as 1 the algorithm works as a lookup table, returning the most identical position in signal space, meaning that estimated positions coincide with available training points. Small variations in RSS values can change the estimated position to other distant training point, depending on the radio map density. In order to take advantage of the regression feature, it is popular to either select K as 2 or 3. This way estimated positions are still close to the user real position, but mispredictions have less impact in estimation accuracy.

To better understand the geometry involved, Fig. 3.9 shows an example with four train

points available in the radio map.

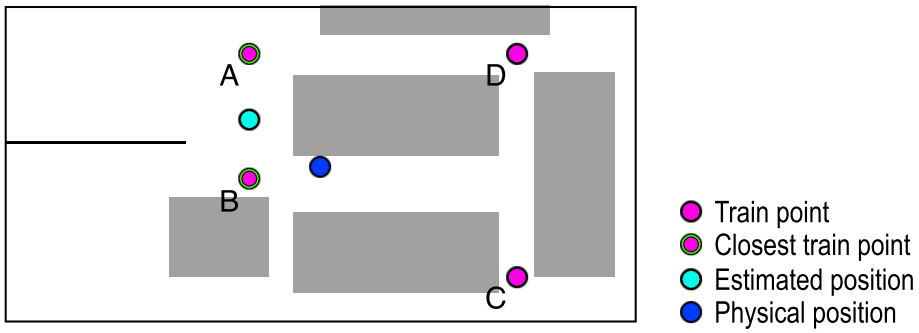


Figure 3.9: Fingerprinting runtime regression example.

Points A, B, C and D represent the trained positions available on the radio map. By selecting K as 2, from those 4 train points the KNN algorithm will estimate the position based on 2. Points A and B are the ones closest to the user real position, so the estimated position is the centroid point between A and B.

This is how the KNN algorithm is able to estimate the position, however it is clear that the centroid of the closest training points is not that close to the user real position. An improvement to this algorithm is the Weighted K Nearest Neighbors (WKNN) algorithm. The WKNN differs from the KNN in the regression phase: instead of computing the centroid of the K closest points, the regression is weighted. By using the inverse of the distance, in signal space, between the query RSS values and the K closest train points RSS values as the weight, the estimated position will be closer to the train point that has the most identical fingerprint. Let W_K be the vector of the inverse distances present in D_K, $W_K = [\frac{1}{d_3}, \ldots, \frac{1}{d_5}]_K$, the estimated position, p_{ew}, can be computed as the following weighted average:

$$p_{ew} = \left(\frac{\sum_{i=1}^{K} W_{K_i} P_{K_{i_x}}}{\tau}, \frac{\sum_{i=1}^{K} W_{K_i} P_{K_{i_y}}}{\tau} \right) \tag{3.3}$$

where τ is the sum of all distances, $\tau = \sum_{j=1}^{K} W_{K_j}$.

Geometrically, Fig. 3.10 represents the new estimated position using the WKNN algorithm.

By comparing the previous estimated position (the one with lower opacity) with the new one, it is clear that the estimated position using the WKNN algorithm is closer to point B which is closer to the real position.

The position estimation can be even improved by taking advantage of the orientation category also present on each radio map entry. This addition is simple yet it allows faster position estimation and with better accuracy. While using KNN or WKNN the query RSS

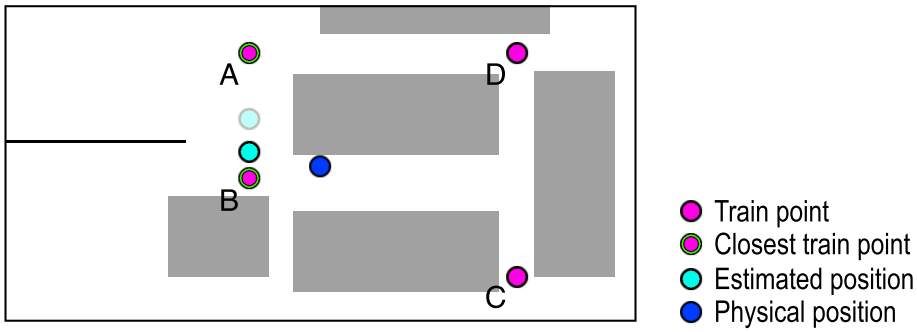


Figure 3.10: Fingerprinting runtime regression example using WKNN.

values are compared to all the existing ones in the radio map, using the orientation category it is possible to reduce the number of computations. By filtering out the radio map entries that do not match in orientation category, it is possible to only estimate the position using train points that are oriented in the same direction as the user, reducing the amount of distance computations performed.

In summary, the fingerprinting technique is able to estimate the position based on information previously recorded in a radio map. In this implementation, there are 3 variables that can be used to tweak the system behavior: either use KNN or WKNN matching algorithm, select the K number of possibilities to estimate the position and either use the orientation category while matching or not. These variables allow for finer control of the SmartLocator fingerprinting component and can be defined in the application settings.

3.4 Dead Reckoning

Dead reckoning is a localization technique that uses IMUs such as accelerometers, gyroscopes and magnetometers to infer meaningful information regarding the user motion. Unlike fingerprinting, this technique is not capable of estimating an absolute position. Its type of localization is relative, which means that an initial position must be known before hand.

By continuously analyzing inertial sensors data, it is possible to track a user from an initial position. Some existing solutions [18] [36] rely on numerical integration to perform the tracking. Using linear acceleration obtained from accelerometers, the relative position can be obtained by double integrating it. However, poor data acquisition may lead to accumulated errors that degrade the estimation quality.

In this work, instead, another approach is followed. Being able to detect the user steps, their length and the user heading direction also allows for tracking. If each time a step is detected, its length, r, and heading direction, θ, can be computed, the new user position can

be obtained by summing the step displacement vector to the last known position. Fig. 3.11 represents the basis of this technique.

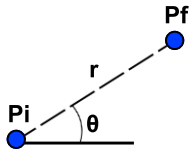


Figure 3.11: Dead reckoning tracking basis.

The displacement vector, $\vec{\delta}$, can be represented in polar coordinates as (r, θ). Therefore the new user position, Pf, can be easily computed by summing the displacement vector to the initial known position, Pi, as $Pf = Pi + \vec{\delta}$.

To successfully use inertial sensors data, there is the need to understand the working principle of each sensor. Accelerometers measure acceleration along the x, y and z axes in its own inertial frame. Gyroscopes measure the rate of rotation or angular velocity along each axis: roll, pitch and yaw. And magnetometers measure the ambient geomagnetic field also for all the three axes. Fig. 3.12 represents the inertial frame of these 3 types of sensors.

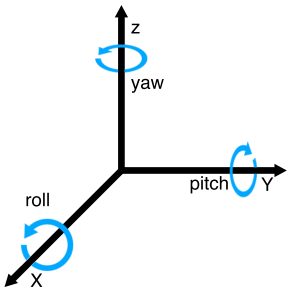


Figure 3.12: IMUs intertial frame.

Thanks to MEMS technology, in most modern smartphones, both accelerometer, gyroscope and magnetometer are packed as a single integrated circuit. And, as it is soldered flat in the smartphone motherboard, the device inertial frame is the same as the sensors inertial frame. However, the user may not align the mobile device with its own inertial frame while using it. Smartphones are most likely to be handled with arbitrary orientation, as shown in Fig. 3.13.

This impacts the way steps are detected and most importantly the overall accuracy of the system. To overcome this issue, the inertial frames should be aligned. As the user can be positioned and oriented according to any world frame, it was made the decision to use the East-North-Up (ENU) frame as the reference frame. The ENU frame, as the name suggests,

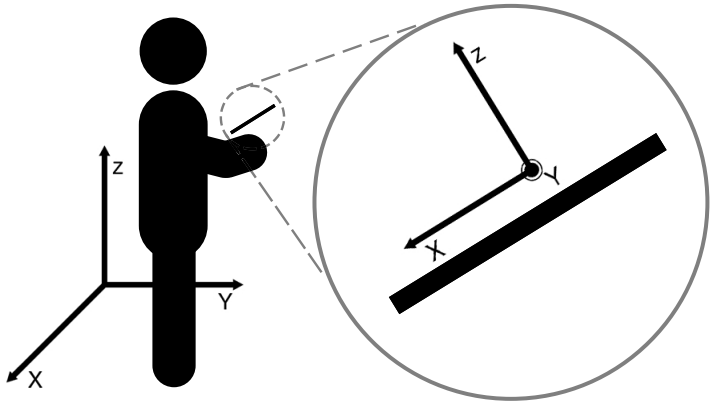


Figure 3.13: User handling smartphone with arbitrary orientation.

has an axis pointing east, another pointing north, defining a plane parallel to the ground, and has an axis pointing outwards the center of the earth (up), as represented in Fig. 3.14.

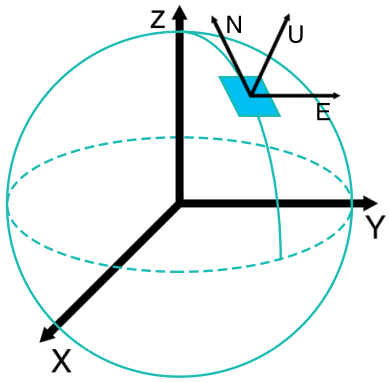


Figure 3.14: East-North-Up frame explanation.

In order to align the smartphone frame with the ENU one, a rotation matrix can be estimated and further used to project accelerometer and gyroscope readings coordinates into world space coordinates. Fig. 3.15 represents the architecture of the SmartLocator dead reckoning component and where this projection takes place.

Acceleration obtained from the accelerometer also contains a term corresponding to gravity. The linear acceleration, on the other hand, is the acceleration with gravity removed and is the input required for both step detection and step length estimation algorithms. The relationship between linear acceleration, a_l, raw acceleration obtained from the accelerometer, a_r, and gravity, g, is thus the following: $\vec{a_l} = \vec{a_r} - \vec{g}$. This way, gravity and magnetic field vectors can be used to project the linear acceleration in the ENU world space frame, as gravity points the opposite direction of the Up axis and the magnetic field vector points the North direction.

Instead of feeding the projected linear acceleration to the step detection and step length

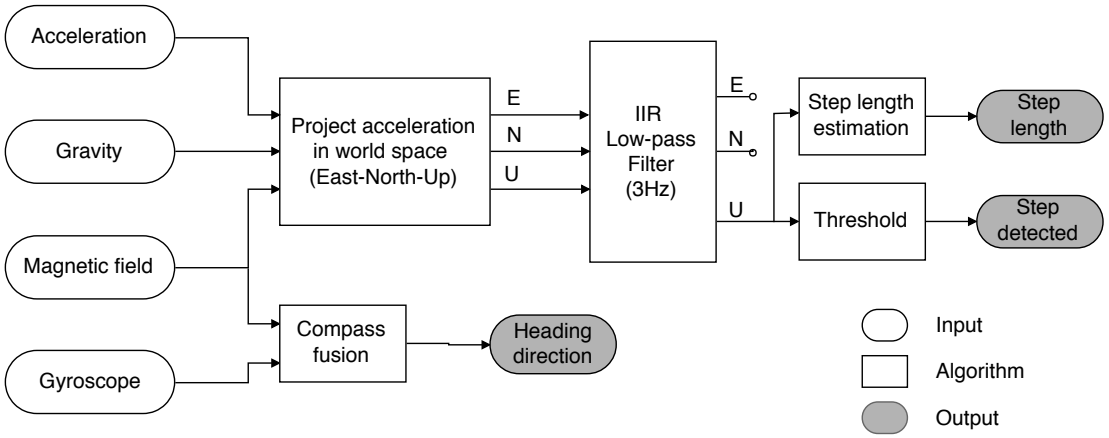


Figure 3.15: Dead reckoning architecture.

estimation algorithms, it is first filtered. The raw acceleration has a lot of noise. Given the nature of user walking motion, it was selected experimentally 3 Hz as the cut off frequency of the low-pass filter implemented to reduce the undesired peaks. Also, several other solutions employ cut off frequencies similar to this.

The next subsections further detail how the step detection is performed and its length estimated, based on filtered acceleration present on the Up axis and also how the user heading direction is estimated.

3.4.1 Step detection

For the step detection, a simple relative threshold algorithm is implemented. The walking motion follows a certain pattern and so does the vertical acceleration sensed in the smartphone accelerometer. By using the projected linear acceleration in the ENU frame, the vertical component (relative to the Up axis) becomes independent of the mobile device attitude. So, regardless of the smartphone orientation the algorithm is able to detect steps.

While walking, before taking a new step, our heel strikes the ground. When that happens, a negative peak can be sensed in the vertical linear acceleration. Furthermore, in between steps, during the swing phase, a positive peak can be detected. Knowing these two facts that describe the two top-level phases of our walking mechanics (stance and swing phases), a step can be considered as detected when two consecutive negative-positive peaks are detected. In Fig. 3.16 the linear acceleration along the Up axis from 5 steps is shown.

During the swing phase the peaks are identified as red, and during the stance phase as green. In order to computationally consider a step as detected, two thresholds are taken into account. First, only sufficiently close consecutive peaks in time are allowed. As the algorithm

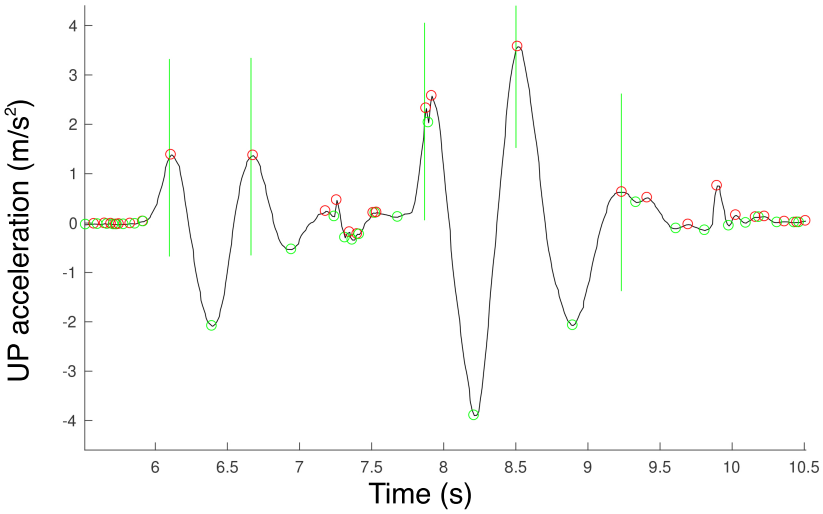


Figure 3.16: Vertical linear acceleration sensed on the Up axis while walking.

looks for consecutive peaks in the specific order of first a negative one and in second a positive one, if a negative peak is detected as a result of high magnitude noise or smartphone erroneous handling, after a short period of time, Δ_{max}, it is able to detect a negative peak again. If this was not account for, some steps would fail to be detected. The other threshold determines the minimum magnitude difference between the consecutive peaks, A_{min}, for the step to be detected. This second threshold can be regulated by performing a set of experimental calibrations. Two consecutive steps, taken at times t_{-1} and t_0, with magnitudes of A_{-1} and A_0, respectively, are detected if the following conditions are met:

$$\begin{cases} t_0 - t_{-1} <= \Delta_{max} \\ A_0 - A_{-1} >= A_{min} \end{cases} \tag{3.4}$$

In Fig. 3.16, the steps that are considered as detected according to conditions 3.4 are marked with a vertical green line in the positive peak. The A_{min} threshold is selected as 1 m/s^2 and Δ_{max} is 0.8 seconds. These values were selected after performing a set of calibrations and highly depends on the person using the system.

3.4.2 Step length estimation

When humans take steps, their hip experiences a displacement along the Up axis. In a stationary phase the hip is in its most higher position. Contrary, while in between steps, during the swing phase, the hip reaches its lower position. Harvey Weinberg demonstrated that the step length is related to the distance between the hip positions, d, during the step [37]. Fig. 3.17 represents the hip displacement experienced while walking:

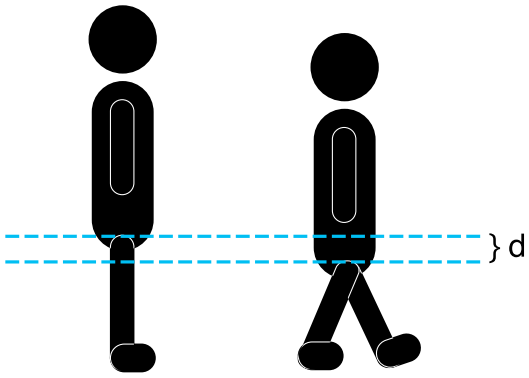


Figure 3.17: Hip vertical displacement in different walking phases.

Furthermore, Weinberg demonstrated that the step length, S_l, can be computed using the maximum and minimum acceleration sensed during the step, A_{max} and A_{min}, respectively, as following:

$$S_l = K \cdot \sqrt[4]{A_{max} - A_{min}} \tag{3.5}$$

Where K is a constant used for unit conversion.

3.4.3 Heading direction estimation

Some approaches [38] [39] infer motion heading by analyzing acceleration over the horizontal east-north, EN, plane. However, this specific technique requires high sampling rates and IMUs available on mobile devices are only capable of outputting at 20-25 Hz.

In order to estimate the user heading angle, it is assumed that the user walks with the smartphone aligned with his movement. More precisely, that the yaw of the smartphone is the same as the yaw of the user. Figures 3.12 and 3.13 help in understanding how the yaw axes are assumed to be aligned. This way the smartphone magnetometer can be used as a compass to effectively estimate the user motion heading angle, but at the same time the other axes remain free.

Magnetometers output a vector with the sensed ambient magnetic field strength in each axis. It is possible to compute the heading angle between that vector and the magnetic north. The magnetic north vector can be seen as parallel to the EN plane and with a yaw of 0^{o}. The heading angle, θ, is thus the angle along the z-axis between the magnetic north vector and the vector obtained from the magnetometer, as shown in Fig. 3.18.

Where the magnetic north vector is represented in blue and an arbitrary vector obtained from the magnetometer in orange.

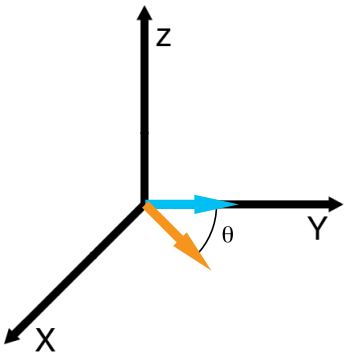


Figure 3.18: Heading angle representation.

However, magnetometers accuracy is directly influenced by the presence of other magnetic fields other than the earth's. To compensate these external influences, the gyroscope is used. Gyroscopes can be used to obtain the rate of rotation along a certain axis. By numerical integrating the angular velocity, rotation deltas can be computed and fused with the magnetometer heading to improve the estimation. The fusion is implemented using a complementary filter, as proposed by Shane Colton [40]. This filter fuses two sources of data by filtering both inputs and summing them to obtain the final result. The first input is low-pass filtered and the second high-pass filtered, which in this case corresponds to the magnetometer heading angle and the orientation obtained from the gyroscope, respectively. Fig. 3.19 represents the fusion using the complementary filter.

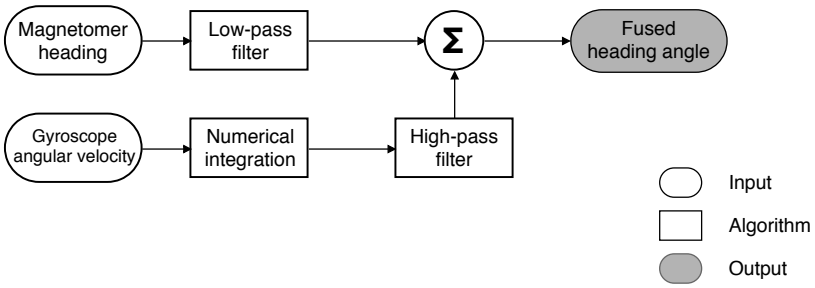


Figure 3.19: Complementary filter architecture.

By filtering the inputs this way, the high frequencies of the orientation from the gyroscope are preserved, resulting in a dynamic fused orientation and the low frequencies of the magnetometer preserve the long-term changes.

In the scope of this work the complementary filter implementation is based on the library created by Paul Lawitzki [41].

Besides the actual heading estimation, this component of the SmartLocator system can also account for a heading offset. Given that the heading angle is relative to the mag-

netic north, there might be cases where the natural heading of the area of interest is not aligned with the magnetic north. However, the representation of the area of interest in the smartphone is always along the natural heading. Fig. 3.20 helps understanding this issue.

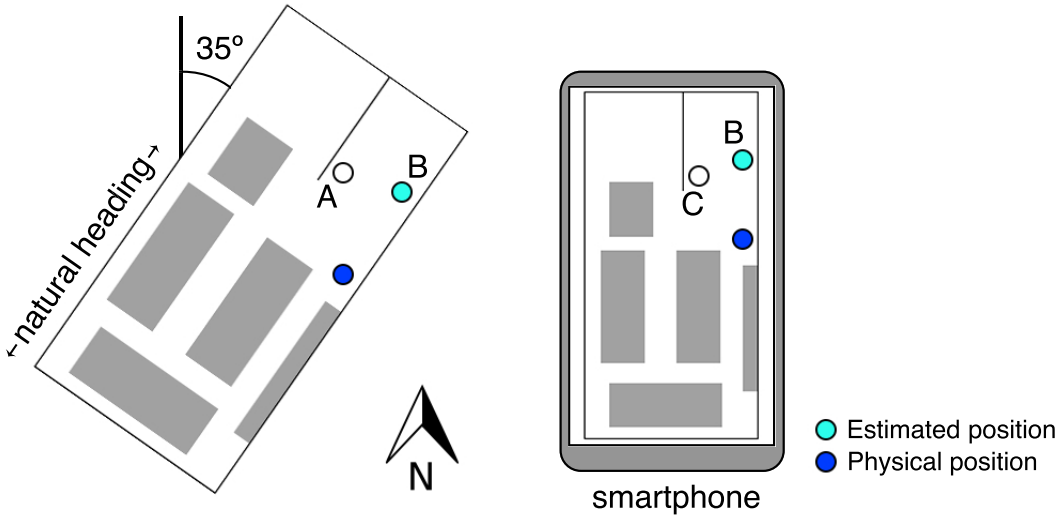


Figure 3.20: Natural heading offset problematic representation.

As shown, if the user walked along the north direction it would reach point A in the area of interest, as the heading angle is reported as being 0^{o}, the SmartLocator would estimate the position as being in point B. To accommodate this, the SmartLocator application allows for the offset to be defined in the settings. This offset will be subtracted from the fused orientation every time a new step is detected. In this case, the natural heading of the area of interest is about 35^{o} from the magnetic north. This way, if the user walked north it would reach point A, but the reported heading angle would be -35^{o}, allowing the SmartLocator to correctly estimate the user position as being in point C.

3.5 Fusion

The last, yet most important component of the SmartLocator localization estimation system is data fusion. As shown in Fig. 3.5 both fingerprinting and dead reckoning components produce output that is further processed in the data fusion component. By combining the absolute position estimation based on the fingerprinting technique with the instantaneously available step displacement vectors from the dead reckoning technique, the data fusion component is able to achieve better estimation accuracy.

As presented in detail in section 3.3, the fingerprinting component is able to output a new absolute position at every one or two seconds, depending on the Android operating system.

On the other hand, the dead reckoning component works in a reactive manner. As soon as a step is detected, it outputs the information regarding the new step: its length and heading angle.

Some approaches [22] [23] use particle filters to perform the fusion between these two sources of data. In this work, an opportunistic approach is implemented, instead. Given that the two types of localization differ a lot, it is interesting to analyze how they might complement one another. Estimated fingerprinting positions are absolute in respect to the area of interest frame. In contrast, estimated dead reckoning positions are relative to a known initial position. This is the main difference between the two. To take advantage of these individual features, the fingerprinting component will be responsible for the long term changes, as to reduce the drift imposed by the dead reckoning. The dead reckoning component can deliver fast updates and so it can provide dynamic position estimation in between fingerprints, being responsible for the high frequency variations in short time intervals. The philosophy behind this approach is based on the complementary filter [40]. Fig. 3.21 shows a finite state machine (FSM) that further details the fusion logic.

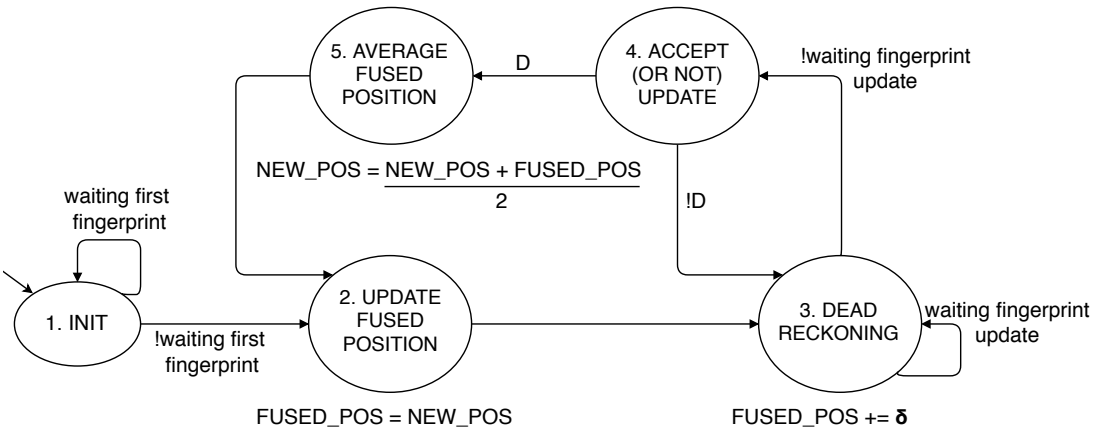


Figure 3.21: Data fusion basic FSM.

The fusion algorithm can be expressed as five different states. Beginning a new position estimation demands an absolute position, because using dead reckoning requires an initial known position. The first state (INIT) represents this behavior by waiting for the fingerprinting component to output an absolute position. After having an absolute position the fusion system transitions to a state where the current fused position (FUSED_POS) is updated to the just acquired absolute position. This state also triggers an UI update,

which means that the virtual indicator that represents the user location moves to the fused position. Unconditionally after this state the system transitions to a new state where it takes advantage of dead reckoning component. In this state, every time a step is detected, its displacement vector, δ, is added to the fused position. Fig. 3.22 shows how the system works so far.

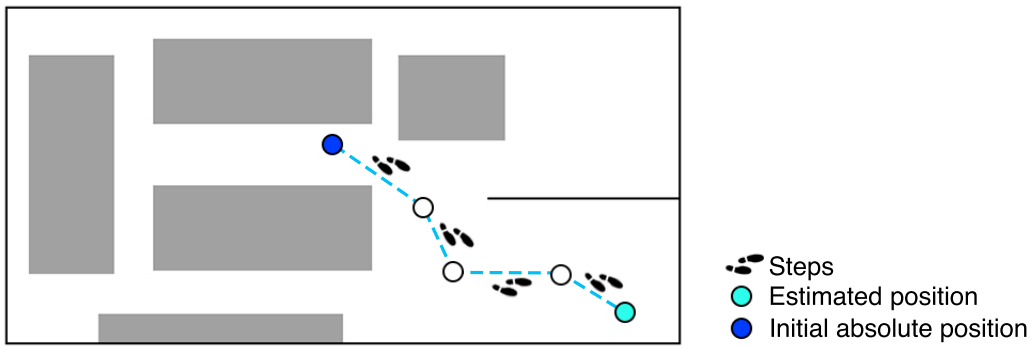


Figure 3.22: Data fusion example using just the dead reckoning technique.

During this state, as it is only based on dead reckoning, the fused position can drift. This is where the long term updates from the fingerprinting component help. Every time the system transitions to this state, a new absolute position is requested. While it is not available, dead reckoning is used, as shown. When it becomes available, its absolute nature is used to correct the drift and the overall fused position. To this extent, the system transitions to the fourth state where the quality of the absolute position is assessed. The position is accepted or not based on a distance condition, D, between the current fused position and the new absolute position. Given the fact that a person can not be in two different places at the same time, D condition models this restriction:

$$D = \begin{cases} 1, dist(FUSED\_POS, NEW\_POS) <= D\_MAX \\ 0, dist(FUSED\_POS, NEW\_POS) > D\_MAX \end{cases}$$

By selecting D_MAX as an appropriate threshold, new absolute positions might be discarded if they are far enough from the current fused position, as represented in Fig. 3.23.

In the case of rejection, the system returns to the third state, and no drift correction is applied. If the absolute position is accepted, the fusion algorithm averages the current fused position with the new absolute position, in the fifth state. By using an arithmetic average, instead of a direct replacement of the fused position for the new absolute position, the system stills converge to the absolute position, but the virtual indicator movement becomes smoother. After which, unconditionally, the system updates the UI and the fused position,

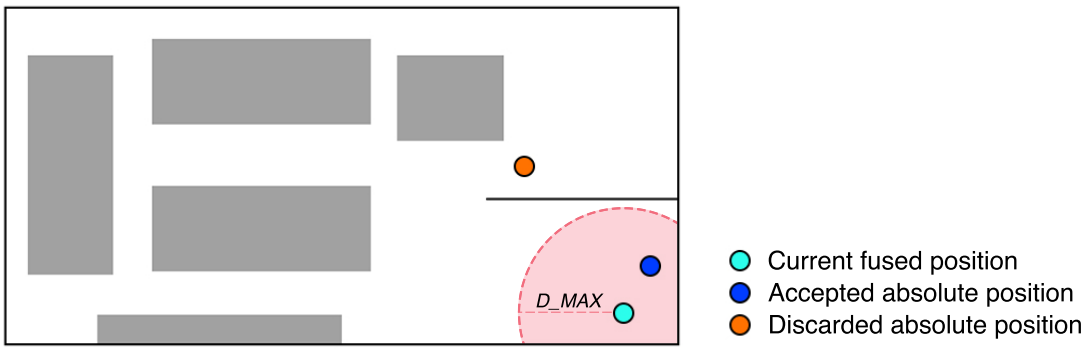


Figure 3.23: Absolute position rejection based on a distance threshold.

back in the second state. Fig. 3.24 shows how the system handles the drift correction, by taking on the example presented previously.

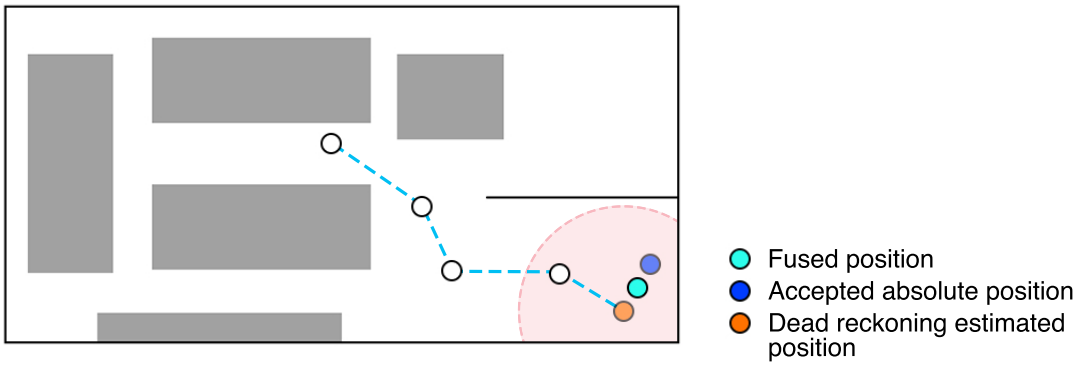


Figure 3.24: Data fusion drift correction example.

This is the basis of how the fusion algorithm operates. However, there is a flaw in this FSM that may lead to only using the dead reckoning component. This happens when the D condition is not satisfied, as shown in Fig. 3.23. When the new absolute position is rejected, the system returns to the dead reckoning state, and the drift is thus not corrected. This can lead to a situation where further new absolute positions will always fail according to D. As of right now, the fusion algorithm is not stable. In order to stabilize the system so that it always converges to absolute positions, correctly eliminating drift, a confidence system is implemented. Fig. 3.25 represents the extended FSM of the fusion algorithm.

The confidence system is represented as the sixth state. It determines the confidence that the fusion algorithm has in positions based only on the dead reckoning component. Each time a new absolute position is rejected, in the fourth state, by failing according to D, the system decrements its confidence about the dead reckoning component. The C condition

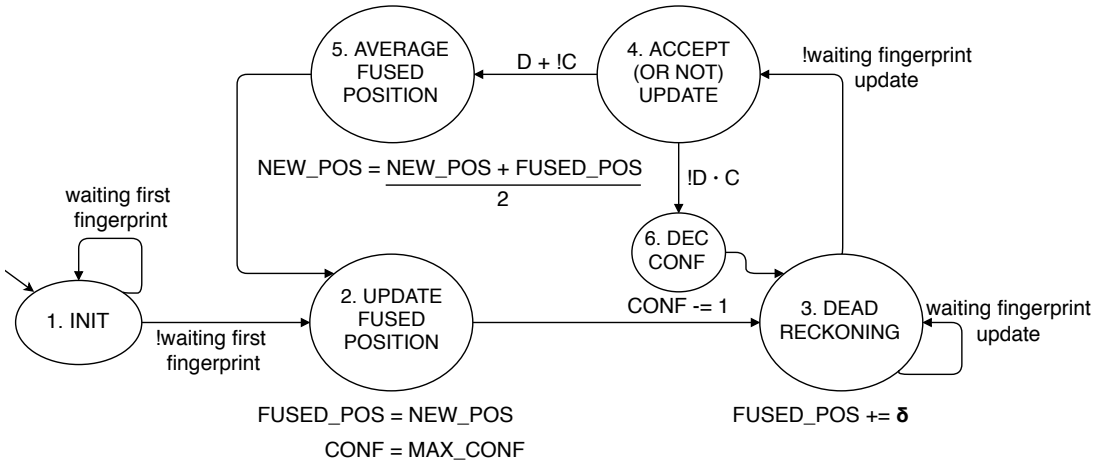


Figure 3.25: Data fusion extended FSM with confidence system.

represents whether the confidence, $CONF$, is above zero or not:

$$C = \begin{cases} 0, CONF <= 0 \\ 1, CONF > 0 \end{cases}$$

In the case of $CONF$ reaching zero, a new absolute position is accepted regardless of the D condition. Every time the system transitions to the second state, the maximum confidence is restored. This addition allows the system to always converge to absolute positions and at the same time preserves the dynamics in position estimation obtained by using dead reckoning. Also, by regulating the MAX_CONF, representing the maximum confidence, the system can be fine tuned as to favor the fingerprinting component over the dead reckoning component or vice-versa.

In terms of the visual representation, the virtual indicator that represents the user location in the area of interest reflects the fused position. States two and three trigger an UI update each time the system transitions to one of them. However, instead of directly repositioning the virtual indicator to the fused position, it is implemented a linear interpolator that translates the position over time, from the last position to the new one. This acts as an animation, but improves the overall aspect and experience of the SmartLocator application.

4 Experimental Results

The SmartLocator system has the ability to allow the fine tuning of each localization component by varying certain algorithm variables. Several experiments were conducted in order to validate and gain a better understanding of how these parameters help in estimating the user position.

In the following sections it is first explained which experiments were conducted and how each one was performed. Later, it is presented and further discussed the localization estimation results of each experiment.

4.1 Setup and method

A different set of experiments was performed for each of the three main components of the SmartLocator localization system: fingerprinting, dead reckoning and data fusion.

The fingerprinting component allows the definition of several parameters of its algorithm. Namely, it allows the position estimation to: either be weighted, by using the WKNN classification algorithm, or not, by using the KNN classification algorithm; use the smartphone orientation category, or not, while performing the radio map matching; and perform the regression of either two (K = 2) or three (K = 3) radio map entries. These 6 variants of the fingerprinting algorithm were tested and their results presented in section 4.2.

The dead reckoning component was first tested in terms of step detection accuracy, as it is crucial for the algorithm overall performance. By establishing a pre-defined initial position in which the user must start the experiment, the component was also put to test regarding position estimation. Section 4.3 presents the results obtained while using only the dead reckoning component.

Lastly, the data fusion component which leverages both the fingerprinting and dead reckoning strengths was tested according to two sets of experiments. First, the influence of the number of radio map entries used for regression, K, is analyzed. In second, as the system

is able to use the dead reckoning in between fingerprints, the influence of the density of the training points is taken into account. Both experiments are detailed in section. 4.4.

In order to extract meaningful information from each of these experiments, a metric must be defined to evaluate each experiment performance. The metric used is the distance between an estimated position and its correspondent real position, called error from now on. This metric allows the estimation of the average, maximum and standard deviation of the error in position estimation of each component.

Each experiment was conducted in the same controlled area room available in the Institute of Systems and Robotics (ISR) of the University of Coimbra. And a common walking path pattern was used throughout all tests, as shown in Fig. 4.1.

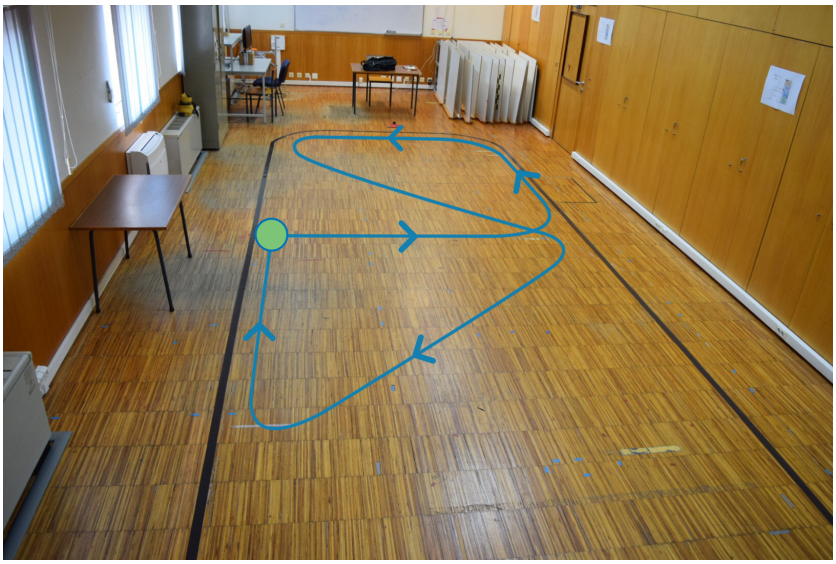


Figure 4.1: Experiments common walking pattern.

The green point indicates both the initial and final points of each experiment, and the arrows indicate the walking direction.

Six runs of the presented path were performed per experiment, while the SmartLocator application was logging each estimated position along with a timestamp into the device storage. Given the fact that different mobile devices have screens with potentially different resolutions, in pixels, each estimated position is normalized before being stored. The normalization is a simple process that allows the coordinates to be independent of the device screen resolution. Let w be the width of the device screen and h its height, an estimated position in the device screen coordinates, (x, y), can be normalized as shown:

$$\text{normalized position} = (\frac{x}{w}, \frac{y}{h})$$

So that each component of the normalized position belongs to the interval $[0, 1]$. Each

log entry has the following format:

{time stamp, normalized position}

This way, a simple Matlab script was developed with the purpose of easily represent estimated positions throughout each experiment, as shown in Fig. 4.2.

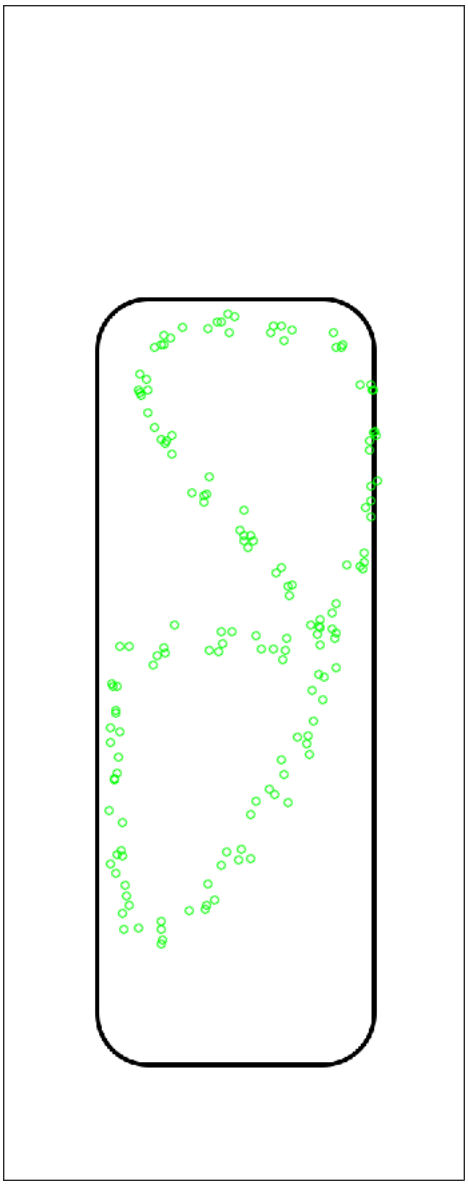


Figure 4.2: Plot of an experiment estimated positions.

In order to compute the distance between an estimated position and the corresponding real position, each experiment was recorded using a Nikon DSLR D5300 camera. Having the video footage of each experiment allowed the ground truth data to be collected by inspection. The camera was mounted on the bottom of the room, above a cabinet. The snapshot represented in Fig. 4.1 was captured from that perspective.

As each log entry contains the timestamp at which that estimated position was obtained,

the experiment video frame corresponding to that exact moment can be obtained and further used to collect the corresponding real position. Another Matlab script was developed that iteratively represents each estimated position available in an experiment log and asks the user to input the corresponding real location by showing the corresponding video frame, as show in Fig. 4.3.

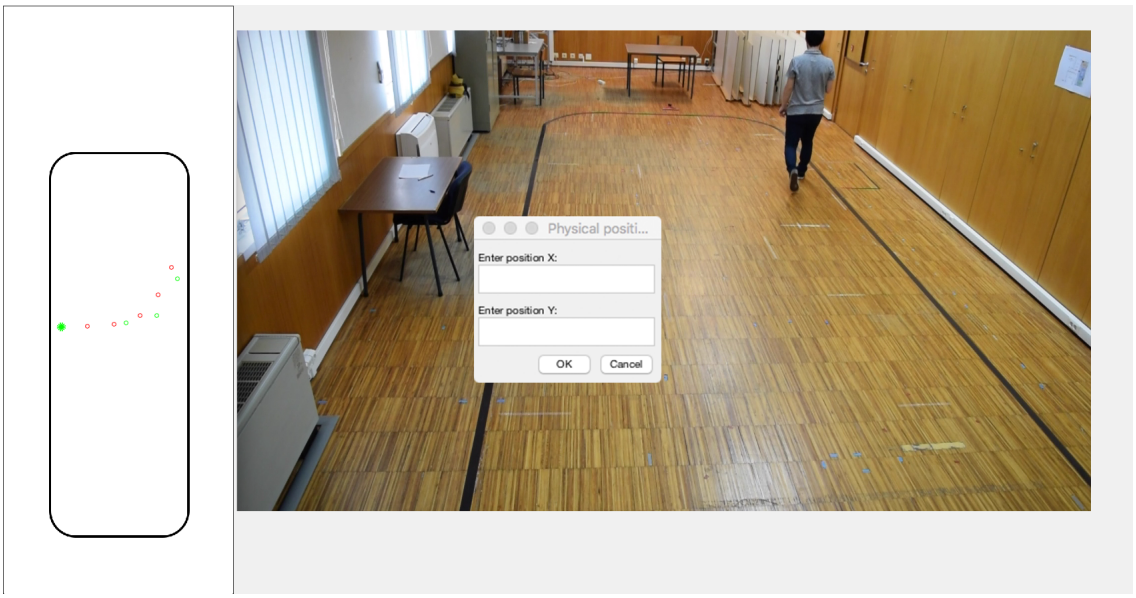


Figure 4.3: Matlab application developed to inspect experiments recorded videos.

Points represented in red are estimated positions and green points are the corresponding real positions input by the user. The output of this Matlab script is two vectors. One represents all the estimated positions and the other the corresponding real ones. By computing the Euclidean distance between each corresponding estimated and real position in the vectors allows the average, maximum and standard deviation of the error to be obtained.

The collected ground truth data accuracy obtained by inspecting the video highly depends on the accuracy of the inspector. To reduce the inspection error a criteria was defined: the user must walk with the smartphone aligned with its waist and thus the real position can be considered as the point in the ground plane corresponding to the vertical projection of the mobile device position, as represented in Fig. 4.4.

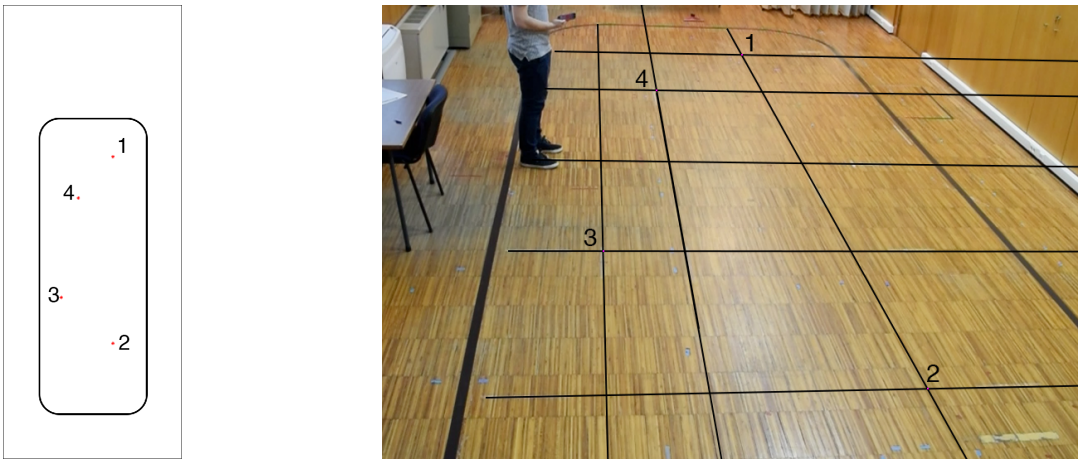
This way, the projection point on the ground plane is close to the center point between the user's feet, which is easier to inspect in the video.

However, while this criteria works flawlessly, this manual inspection consumes a lot of time. To this extent, a computer vision approach was developed to aid the video inspection. Using four control points, whose coordinates in both the layout representation and physical room are known, a homography matrix was estimated. Fig. 4.5 represents the used control

(a) Front view. (b) Side view.

Figure 4.4: Smartphone holding criteria for improved inspection accuracy.

points in both the video frame and the layout.



(a) The control points in the layout image. (b) The corresponding control points in the physical room.

Figure 4.5: Homography matrix estimation between the layout image and the physical room.

This matrix can then be used to project a point in the video frame into the corresponding point in the layout image. This way, the Matlab script that was developed for the video inspection was updated to allow the user to click on the video frame, instead of asking the point coordinates. The position, in pixels, associated with the click is then projected into the layout image using the homography matrix.

Using this technique turned out to be a great improvement in the consumed time to inspect all the data and in the accuracy of the ground truth data collected.

4.2 Fingerprinting

To provide a common baseline between all fingerprinting experiments the same radio map was used across them. The radio map was trained in 6 different positions, where in each one four orientations were recorded (north, east, south and west), as show in Fig. 4.6.

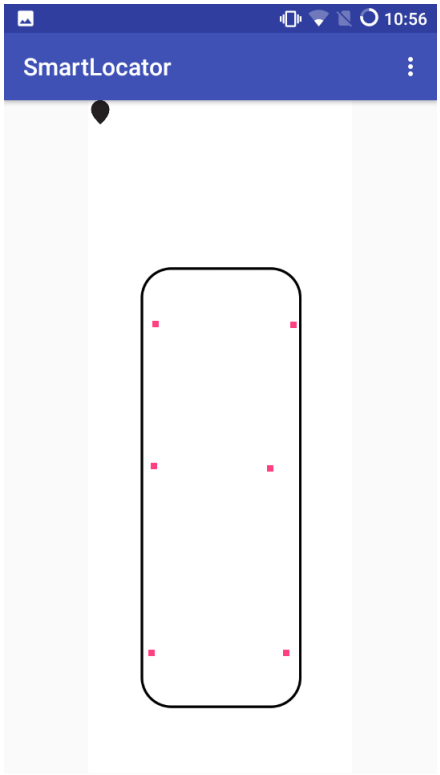
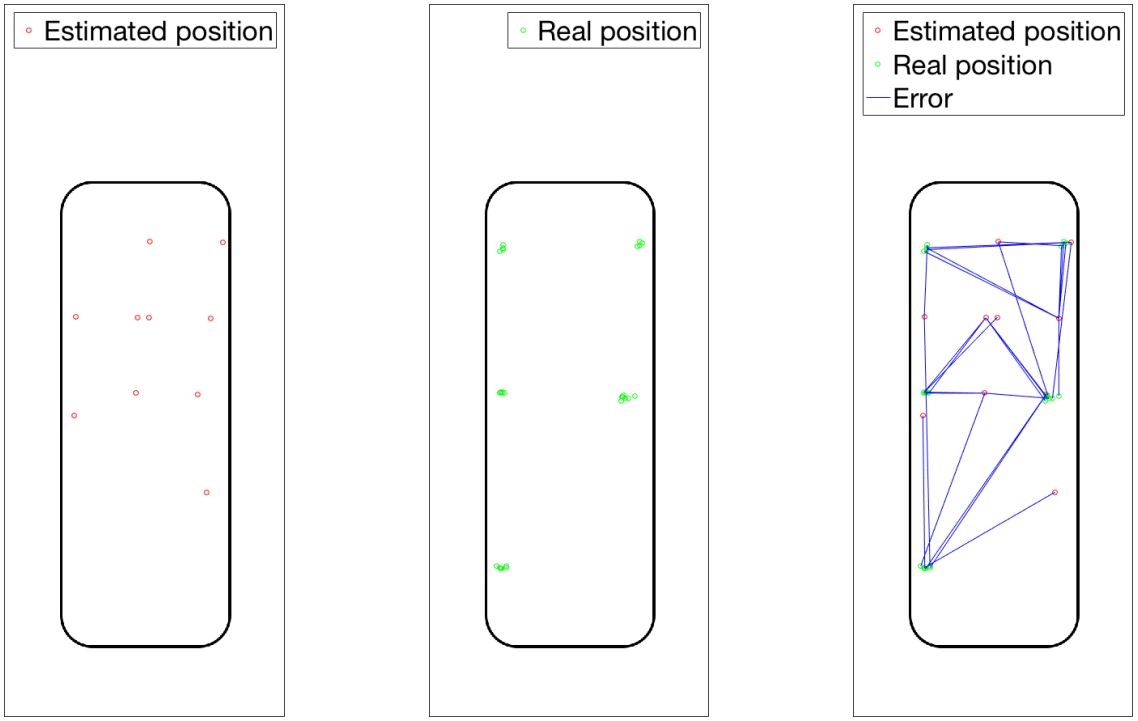


Figure 4.6: Localization of the train points used in fingerprinting experiments.

The layout image was specifically designed to include the rounded corner rectangle that is also present on the real room floor, as can be seen in Fig. 4.1, using the rectangle real coordinates and dimension. The training points are distributed as a 2x3 grid and span across all the room area.

4.2.1 Weighting influence

In order to assess the influence in position estimation between using the WKNN or the KNN algorithm, the system was tested in each situation using K as 2. In Fig. 4.7a the estimated positions can be seen, while performing the walking pattern previously presented, using the KNN algorithm, in Fig. 4.7b the corresponding real positions and in Fig. 4.7c the error between the estimated and real positions. The raw results are presented in appendix B.3, from which the average error in position estimation was computed and is 2.1613 meters, the maximum error 4.6194 meters and the standard deviation (Std), σ, is 0.9692 meters.



(a) Fingerprinting non weighted experiment estimated positions. (b) Fingerprinting non weighted experiment real positions. (c) Fingerprinting non weighted experiment error.

Figure 4.7: Fingerprinting non weighted experiment results.

Using the WKNN algorithm, the estimated positions are shown in Fig. 4.8a, the corresponding real positions in Fig. 4.8b and the error in Fig. 4.8c. In this case the reported average error is 2.2458 meters, the maximum error 3.9699 meters and the standard deviation 1.1457 meters.

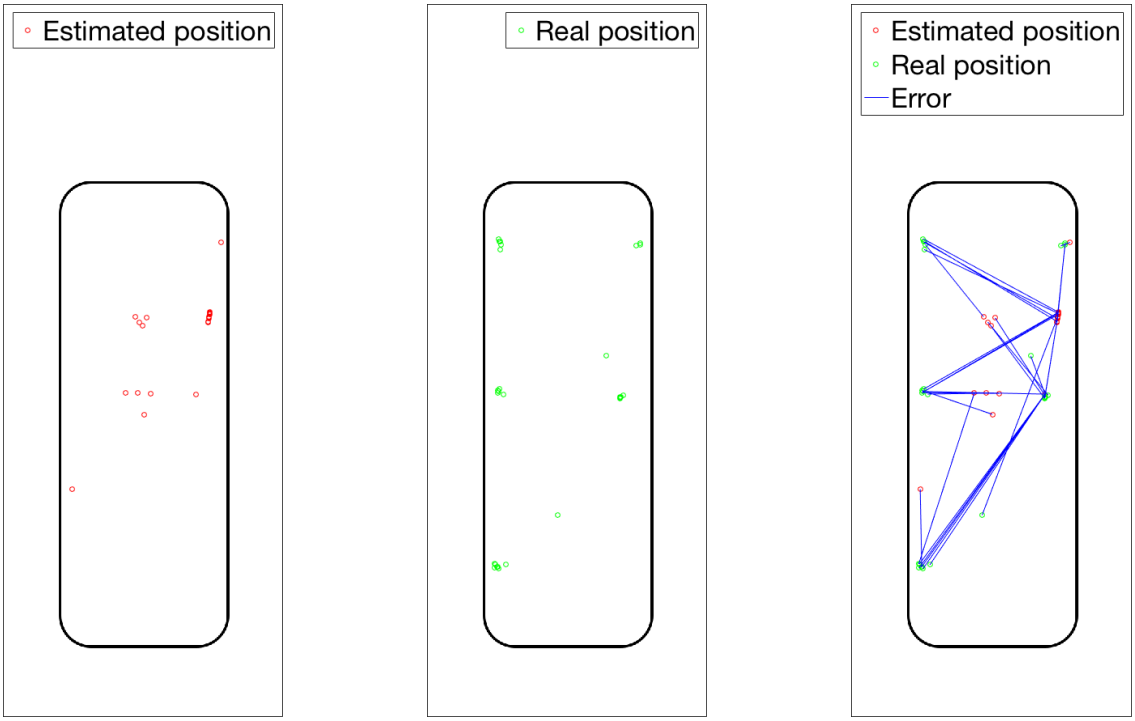
Table 4.1 summarizes the influence in position estimation when using the KNN algorithm or the WKNN algorithm.

Table 4.1: KNN vs WKNN performance comparison summary.

	KNN	WKNN
Average error (m)	2.1613	2.2458
Maximum error (m)	4.6194	3.9699
Std (σ) (m)	0.9692	1.1457

These results were not expected, as the weighted algorithm, WKNN, was implemented using a better regression approach. It is a fact that the average error difference is only $8,45$ centimeters. Also, the maximum error of the WKNN algorithm is 65 centimeters lower than the KNN's. However, in terms of standard deviation, it is again higher in the WKNN case. Overall, is does not outperform the KNN algorithm.

Although both the WKNN and the KNN algorithms have low implementation complexity, the WKNN is still more complex. This result proves that the added complexity is not worth



(a) Fingerprinting weighted experiment estimated positions.

(b) Fingerprinting weighted experiment real positions.

(c) Fingerprinting weighted experiment error.

Figure 4.8: Fingerprinting weighted experiment results.

it, both in implementation time and in computation time.

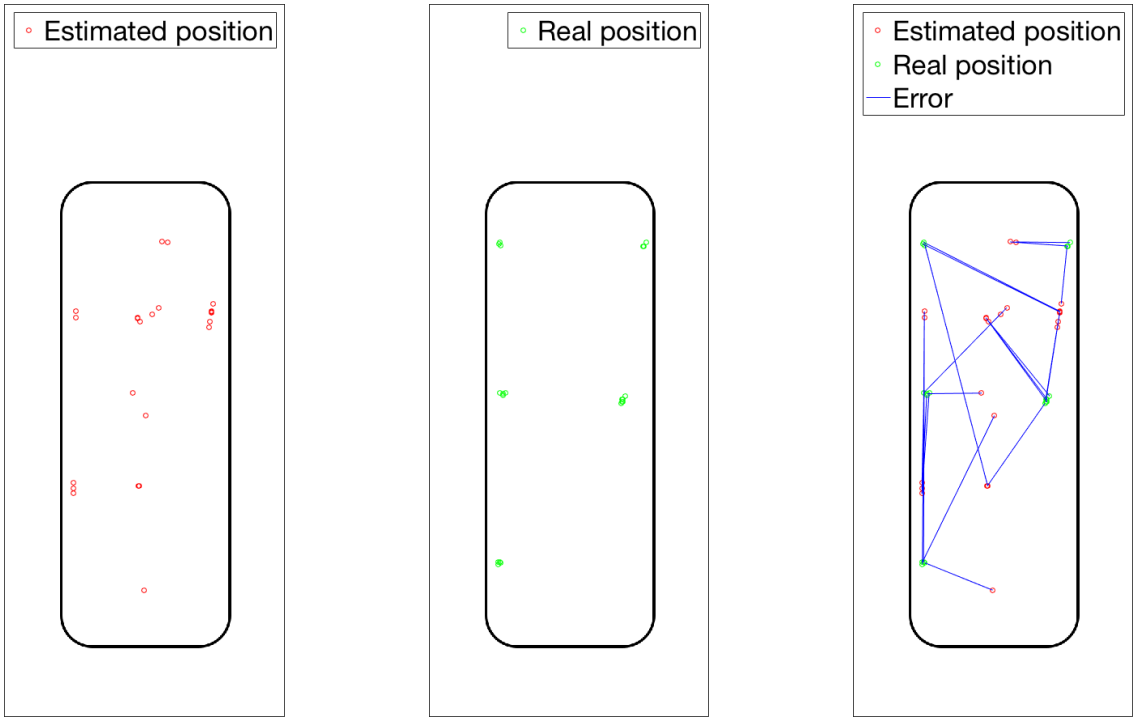
4.2.2 Orientation influence

Another feature of the fingerprinting component is the ability to complement the matching of fingerprints with the orientation category (see Fig. 3.8) in which they were trained. To this extent, two experiments were conducted: one without using the orientation category; the other using it. Both experiments using the WKNN algorithm.

The experiment conducted without orientation category matching yields the same results as the previous experiment using only the WKNN algorithm. Figs. 4.8a, 4.8b and 4.8c show case the estimated points, their corresponding real positions and the associated error between them, respectively.

With orientation category matching, Fig. 4.9a presents the experiment estimated points, Fig. 4.9b the corresponding real positions and Fig. 4.9c the error. The average error, obtained using orientation category matching, is 2.1523 meters, the maximum error is 4.6843 meters and the standard deviation is 1.8620 meters.

The comparison between using orientation category matching or not is presented in table 4.2.



(a) Fingerprinting with orientation category matching experiment estimated positions.

(b) Fingerprinting with orientation category matching experiment real positions.

(c) Fingerprinting with orientation category matching experiment error.

Figure 4.9: Fingerprinting with orientation category matching experiment results.

Table 4.2: Orientation matching vs Without orientation matching performance comparison summary.

	Using orientation matching	Without using orientation matching
Average error (m)	2.1523	2.2458
Maximum error (m)	4.6843	3.9699
Std (σ) (m)	1.1183	1.1457

Using orientation category matching slightly improves the average estimation error, while having a higher maximum error than without using it. The improvement is about $9,35$ centimeters in average.

Regarding the previous best setup (using KNN without orientation) its average error is about 1 centimeter higher than the WKNN algorithm with orientation (2.1613 meters vs 2.1523 meters). Making these two possibilities the best performant so far.

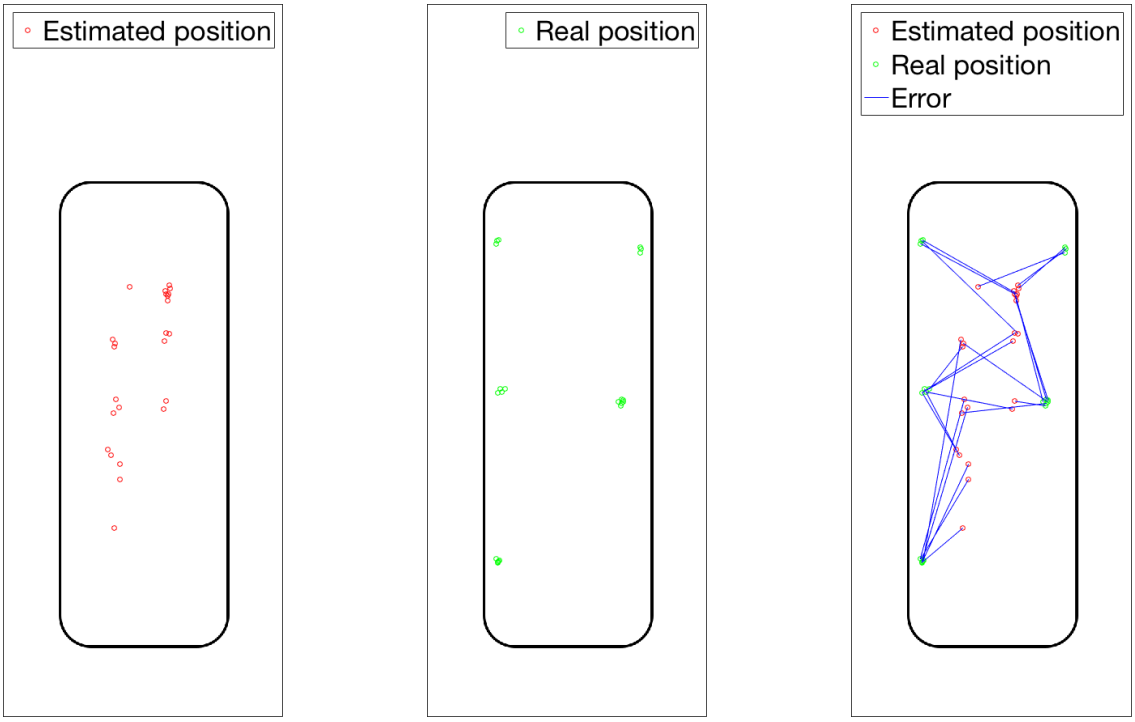
4.2.3 K influence

The previous experiments were conducted using K as 2. However, how does K influence the overall performance of position estimation? To answer this question two experimental tests were made using the most common K constant values: 2 and 3.

The experiment conducted with K as 2 yields the same results as the previous experiment

using the WKNN algorithm with orientation category matching. Figs. 4.9a, 4.9b and 4.9c show case the estimated points, their corresponding real positions and the associated error between them, respectively.

By selecting K as 3, the results are presented in Figs. 4.10a, 4.10b and 4.10c in the same manner. Yielding an interesting result, with an average error of 1.8602 meters, maximum error of 4.1502 meters and a standard deviation of 0.7676 meters. Which is a great improvement from the previous experiment: 13,5% better.



(a) Fingerprinting using K as 3 experiment estimated positions.

(b) Fingerprinting using K as 3 experiment real positions.

(c) Fingerprinting using K as 3 experiment error.

Figure 4.10: Fingerprinting using K as 3 experiment results.

Table 4.3 compares the previous best results with this one.

Table 4.3: K=2 vs K=3 performance comparison summary.

	Using K as 2	Using K as 3
Average error (m)	2.1523	1.8602
Maximum error (m)	4.6843	4.1502
Std (σ) (m)	1.1183	0.7676

Both the average and maximum errors are greatly improved. Resulting in the best combination of the fingerprinting component parameters: using K as 3, using orientation category matching and weighting the regression by using the WKNN algorithm.

4.3 Dead reckoning

The dead reckoning component wast tested regarding its accuracy in detecting steps and position estimation. The same approach of video recording several turns while logging data to the smartphone was used. Fig. 4.1 represents the walking pattern and the initial position used for dead reckoning (green point).

4.3.1 Step detection

Assessing the step detection accuracy is a matter of comparing the estimated step count of each run and manually inspect the corresponding video footage. In this experiment the walking pattern was not used. The user walked freely in the area.

Three different types of experiments were performed. In the first type the user walked slowly, in the second the user walked at regular speed and in the third the user walked in an accelerated pace. In each type were performed 6 runs where in each the user walked 10 steps. Table 4.4 shows the results of each experiment.

Table 4.4: Step detection results.

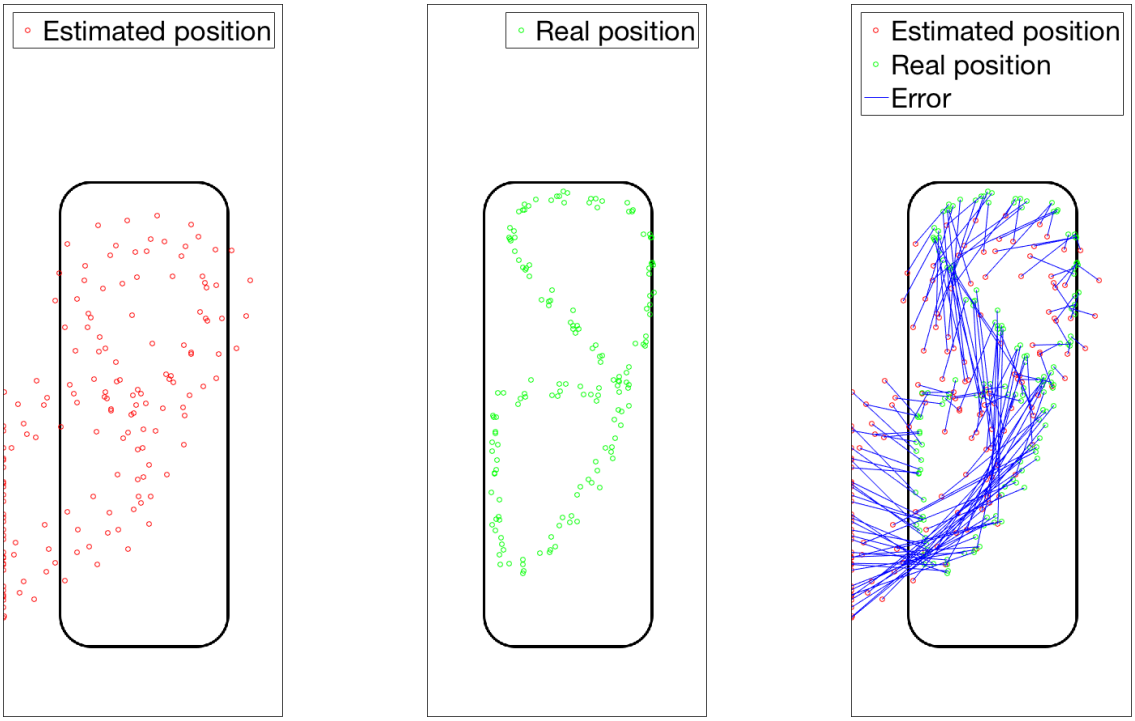
	Slower speed	Regular speed	Accelerated speed
Estimated steps	61	61	61
Real steps	60	60	60
Mispredicted steps	1	1	1

The algorithm predicted the same number of steps in each run, interestingly. Having estimated one step more than what the user took in each experiment. One misprediction in a total of 60 steps yields an accuracy in step estimation of $98,(3)\%$.

4.3.2 Accuracy

Regarding the overall position estimation accuracy, six experiments were conducted in which the user started from the same initial position and completed the circuit, as in previous experiments. The estimated positions were then compared with the ground truth location obtained from inspecting the video footage of each test.

Figs. 4.11a, 4.11b and 4.11c, respectively, represent the estimated points, their corresponding real positions and the associated error between them. The raw data is presented in B.5.



(a) Dead reckoning estimated positions.

(b) Dead reckoning real positions.

(c) Dead reckoning error.

Figure 4.11: Dead reckoning results.

Using only the dead reckoning component, the average error obtained is 1.6607 meters, the maximum error is 4.1616 meters and the standard deviation is 1.0632 meters. Which is more accurate than using only the fingerprinting component. It was not expected that the dead reckoning component was this accurate. In table 4.5, the dead reckoning component accuracy is compared against the best performing fingerprinting setup.

Table 4.5: Dead reckoning vs Best fingerprinting setup comparison.

	Dead reckoning	Fingerprinting with K as 3, WKNN and orientation
Average error (m)	1.6607	1.8602
Maximum error (m)	4.1616	4.1502
Std (σ) (m)	1.0632	0.7676

The dead reckoning component is 10.72% more accurate than the best fingerprinting setup. However, given the nature of the type of localization obtained while using dead reckoning, if the tests were longer, the drift effect would be more noticeable. Impairing this method accuracy.

4.4 Fusion

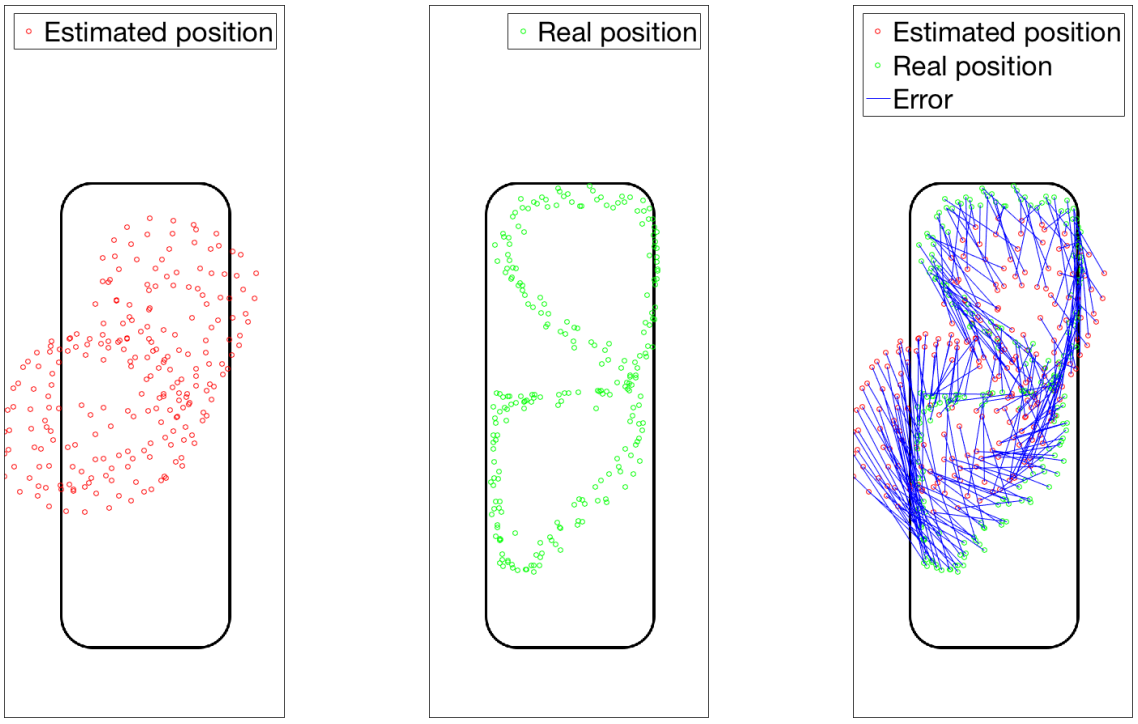
Sections 4.2 and 4.3 discuss the accuracy in position estimation while using the fingerprinting technique or the dead reckoning technique individually, respectively. These are the two techniques used in the data fusion component. Having well performant individual components improves the overall system accuracy. The fusion algorithm uses fingerprinting with WKNN and orientation category matching.

In this section performance gains obtained by integrating these two different types of localization technique are presented and further discussed.

4.4.1 K influence

Similarly to the experiment conducted in the fingerprinting component, this experiment evaluates the influence that the number of radio map entries used for regression may have on the final accuracy. To this extent, six turns over the pre-defined path (see Fig. 4.1) were performed while using K as 2 and K as 3.

The estimated positions while using K as 2 are represented in Fig. 4.12a, their corresponding real positions in Fig. 4.12b and the error between them in Fig. 4.12c.



(a) Fusion using K as 2 estimated positions.

(b) Fusion using K as 2 real positions.

(c) Fusion using K as 2 error.

Figure 4.12: Fusion using K as 2 results.

The fusion algorithm using K as 2 is capable of an average error of 1.3955 meters, a maximum error of 2.9191 meters and a standard deviation of 0.6557 meters. It provides a great improvement over using only one type of localization method. Particularly, it is $24,98\%$ more performant than using only fingerprinting and $15,97\%$ better than using only dead reckoning. Table 4.6 compares the fusion against both the fingerprinting and dead reckoning techniques when used individually.

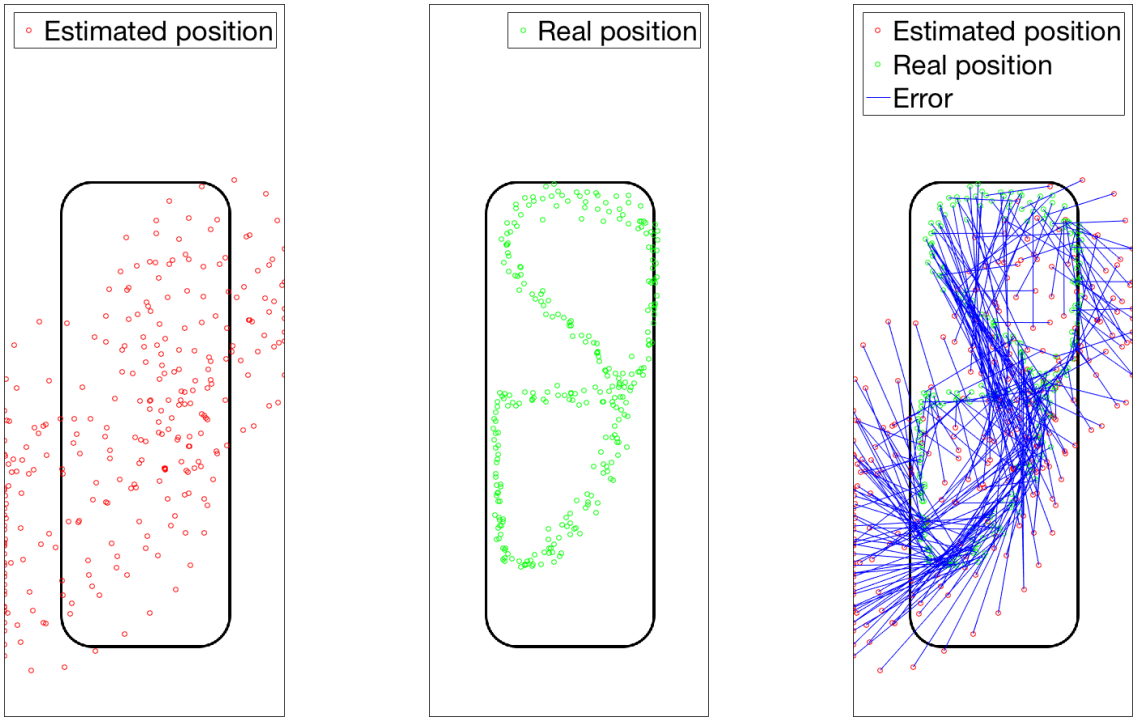
Table 4.6: Fusion using K as 2 vs Dead reckoning vs Best fingerprinting setup comparison.

	Fusion (K=2)	Dead reckoning	Fingerprinting
Average error (m)	1.3955	1.6607	1.8602
Maximum error (m)	2.9191	4.1616	4.1502
Std (σ) (m)	0.6557	1.0632	0.7676

The standard deviation improvement reflects the elimination of the dead reckoning drift by using the absolute position updates from the fingerprinting component. It can be seen in Fig. 4.12a that the estimated points almost do not reach the room boundaries, which, on the contrary, is common in the dead reckoning case (see Fig. 4.11a). Which, naturally, presents an improvement over 1 meter in maximum error.

Previously, fingerprinting (individually) saw an improvement when used with K as 3, instead of 2. In Fig. 4.13a the estimated positions while using the fusion algorithm with K as 3 is represented, their corresponding real positions in Fig. 4.13b and the error between them in Fig. 4.13c.

Just by analyzing Fig. 4.13c it is clear that using K as 2 yields better results. In fact, the average error is 2.0191 meters, the maximum 4.602 meters and the standard deviation 1.0564 meters. Also, by comparing these results with the results available in table 4.6, using fusion with K as 3 even performs worse than using fingerprinting or dead reckoning individually.



(a) Fusion using K as 3 estimated positions.

(b) Fusion using K as 3 real positions.

(c) Fusion using K as 3 error.

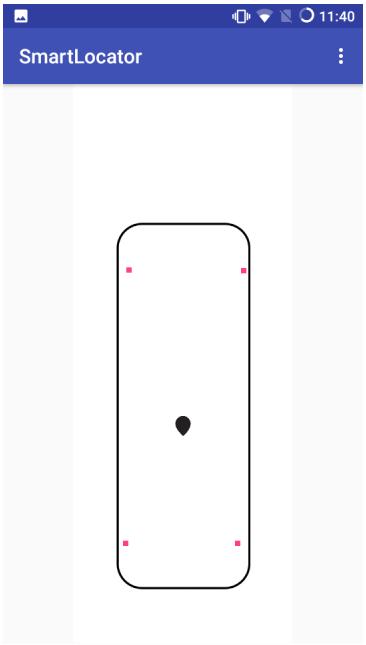
Figure 4.13: Fusion using K as 3 results.

4.4.2 Training points density influence

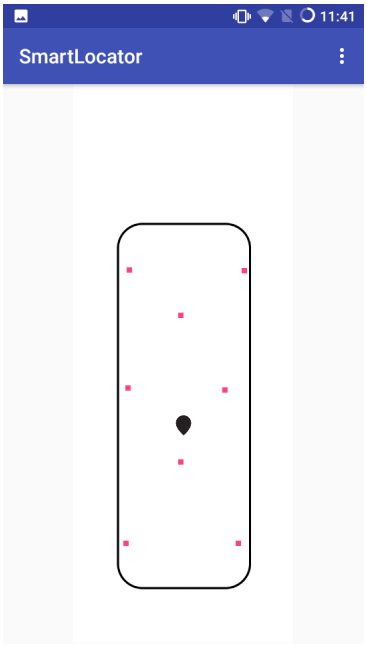
As can be seen in Fig. 4.6, all experiments were carried while using a radio map with 6 points. To better understand the influence of the amount of training points in the overall accuracy, the fusion algorithm was tested using 2 different radio maps. One map with fewer training points, 4, and the other with more training points, 8, as shown in Fig. 4.14. The experiments were performed using K as 3.

While using the radio map with lower density with only 4 training points, the average error was 2.1703 meters, the maximum error 5.0684 meters and the standard deviation 1.0997 meters. On the other hand, using the more dense radio map with 8 points, the average error was 1.9593 meters, the maximum error 5.2504 meters and the standard deviation 1.0664 meters.

The results regarding the use of each radio map (4 points, 6 points and 8 points) is summarized in table 4.7.



(a) Lower density map with 4 training points.



(b) Higher density map with 8 training points.

Figure 4.14: Radio maps with different densities.

Table 4.7: Fusion using a lower density radio map vs Fusion using regular density radio map vs Fusion using higher density radio map comparison.

	Low density	Regular density	High density
Average error (m)	2.1703	2.0191	1.9593
Maximum error (m)	5.0684	4.602	5.2504
Std (σ) (m)	1.0997	1.0564	1.0664

Increasing the density yields best results, with the highest density map achieving $9,72\%$ better accuracy than the lower density one. However, the improvement is not significative. The main conclusion of this experiment is that there is no need to create radio maps with large amounts of training points requiring labour intensive work. A simple, low density radio map, can perform almost identical.

5 Conclusion and Future Work

5.1 Conclusion

Nowadays ubiquitous computing applications are a popular topic. Besides rich UIs, applications developers start to gather context information in order to provide better user experiences. One of the fundamental dimensions of the surrounding context is localization. To that extent, GPS has been used, but fails in indoor environments.

Using alternative localization methods can enhance the data quality regarding the context, so that developers can take advantage of it. Several types of localization solutions already exist, including, but not limited to: fingerprinting, motion sensing, range-based and vision-based. Each having its pros and cons.

This work focuses on a hybrid approach that fuses two localization techniques: fingerprinting and motion sensing. By using two techniques, estimation quality was greatly improved, as the fusion algorithm takes advantage of the best features of each technique yet minimizing their negative aspects. The proposed algorithm is stable and always converges to absolute positions inside the area of interest, successfully eliminating drift over time. Also, by analyzing the algorithm performance while using radio maps with different densities, it was concluded that a lower density map can perform almost identical to a higher density map. Yielding a great result that reduces the training effort regarding the fingerprinting technique.

Several parameters of the fusion algorithm are fine tunable, making it flexible to its user and area of interest independent. The best setup result, in terms of average error in position estimation, was achieved using K as 2, orientation category matching and weighted regression. Interestingly and unexpected, dead reckoning estimated positions were more accurate than fingerprinting ones, when using each localization technique individually.

5.2 Future work

Independently of GPS lower accuracy it would be interesting to incorporate this source of data into the fusion algorithm, to be used in situations where Wi-Fi is not available and the user is outside.

Besides representing user's estimated location the SmartLocator application could make use of that information to present curated information in the form of pop-ups, regarding specific locations in the area of interest. Real applications include points of interest in museums, stores branding in malls, an assistant in navigation systems and advertisement which has great commercial interest.

A negative aspect of the solution is requiring the user to walk with the smartphone aligned with its motion heading. It would be interesting to explore alternative solutions that allow the mobile device to be in the pocket or wallet. That way, the SmartLocator application could run as a background service and take advantage of the Android OS notifications to present relevant information regarding the context without requiring interaction.

References

[1] S. Nikitaki, G. Tsagkatakis, and P. Tsakalides. Efficient Multi-Channel Signal Strength Based Localization via Matrix Completion and Bayesian Sparse Learning. *IEEE Transactions on Mobile Computing*, 14(11):2244–2256, Nov 2015.

[2] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise Indoor Localization Using Smart Phones. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 787–790, New York, NY, USA, 2010. ACM.

[3] Mauro Brunato and Roberto Battiti. Statistical Learning Theory for Location Fingerprinting in Wireless LANs. *Comput. Netw.*, 47(6):825–845, April 2005.

[4] Feng Yu, Ming Hua Jiang, Jing Liang, Xiao Qin, Ming Hu, Tao Peng, and Xin Rong Hu. An Indoor Localization of WiFiBased on Support Vector Machines. In *Progress in Applied Sciences, Engineering and Technology*, volume 926 of *Advanced Materials Research*, pages 2438–2441. Trans Tech Publications, 7 2014.

[5] S. Chieochan, E. Hossain, and J. Diamond. Channel Assignment Schemes for Infrastructure-based 802.11 WLANs: A Survey. *Commun. Surveys Tuts.*, 12(1):124–136, January 2010.

[6] Y. Chapre, P. Mohapatra, S. Jha, and A. Seneviratne. Received signal strength indicator and its analysis in a typical WLAN system (short paper). In *38th Annual IEEE Conference on Local Computer Networks*, pages 304–307, Oct 2013.

[7] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2, pages 775–784 vol.2, 2000.

[8] Moustafa Youssef and Ashok Agrawala. The Horus Location Determination System. *Wirel. Netw.*, 14(3):357–374, June 2008.

[9] Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses. In *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, WiNTECH '06, pages 34–40, New York, NY, USA, 2006. ACM.

[10] Disha Adalja and Girish Khilari. Fingerprinting Based Indoor Positioning System using RSSI Bluetooth. *IJSRD - International Journal for Scientific Research & Development*, 1(4), 2013.

[11] L. E. Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009. revision #136646.

[12] S. H. Fang, C. H. Wang, T. Y. Huang, C. H. Yang, and Y. S. Chen. An Enhanced ZigBee Indoor Positioning System With an Ensemble Approach. *IEEE Communications Letters*, 16(4):564–567, April 2012.

[13] ITU Recommandation. Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 mhz to 100 ghz, 2015.

[14] Cyril Brignone, Tim Connors, Geoff Lyon, and Salil Pradhan. SmartLOCUS: An autonomous, self-assembling sensor network for indoor asset and systems management. 2003.

[15] Esmond Mok, Linyuan Xia, Guenther Retscher, and Hui Tian. A case study on the feasibility and performance of an UWB-AoA real time location system for resources management of civil construction projects. *Journal of Applied Geodesy*, 4(1):23–32, 2010.

[16] Sergio Caccamo, Ramviyas Parasuraman, Fredrik Båberg, and Petter Ögren. Extending a ugv teleoperation flc interface with wireless network connectivity information. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4305–4312. IEEE, 2015.

[17] L. Klingbeil and T. Wark. A Wireless Sensor Network for Real-Time Indoor Localisation and Motion Monitoring. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 39–50, April 2008.

[18] B. Krach and P. Robertson. Integration of foot-mounted inertial sensors into a Bayesian location estimation framework. In *2008 5th Workshop on Positioning, Navigation and Communication*, pages 55–61, March 2008.

[19] Alberto Serra, Davide Carboni, and Valentina Marotto. Indoor Pedestrian Navigation System Using a Modern Smartphone. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, pages 397–398, New York, NY, USA, 2010. ACM.

[20] Oliver Woodman and Robert Harle. Pedestrian Localisation for Indoor Environments. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 114–123, New York, NY, USA, 2008. ACM.

[21] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

[22] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 421–430, New York, NY, USA, 2012. ACM.

[23] Y. Liu, M. Dashti, M. A. Abd Rahman, and J. Zhang. Indoor localization using smartphone inertial sensors. In *2014 11th Workshop on Positioning, Navigation and Communication (WPNC)*, pages 1–6, March 2014.

[24] Henning Lategahn and Christoph Stiller. Vision-only localization. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1246–1257, 2014.

[25] Atanas Georgiev and Peter K Allen. Vision for mobile robot localization in urban environments. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 472–477. IEEE, 2002.

[26] Niklas Karlsson, Enrico Di Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E Munich. The vSLAM algorithm for robust localization and mapping. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 24–29. IEEE, 2005.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-DOF Localization on Mobile Devices. In *ECCV (2)*, pages 268–283, 2014.

[29] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[30] C. Jiang, M. Fahad, Y. Guo, J. Yang, and Y. Chen. Robot-assisted human indoor localization using the Kinect sensor and smartphones. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4083–4089, Sept 2014.

[31] Kejie Qiu, Tianbo Liu, and Shaojie Shen. Model-Based Global Localization for Aerial Robots Using Edge Alignment. *IEEE Robotics and Automation Letters*, 2(3):1256–1263, 2017.

[32] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[33] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[34] IEEE Standards Association et al. 802.11-2012-IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *Retrived from http://standards. ieee. org/about/get/802/802.11. html*, 2012.

[35] Gints Jekabsons and Vadim Zuravlyov. Refining Wi-Fi based indoor positioning. In *Proceedings of 4th International Scientific Conference Applied Information and Communication Technologies (AICT), Jelgava, Latvia*, pages 87–95, 2010.

[36] David Titterton and John Weston. Strapdown inertial navigation technology. *IEEE Aerospace and Electronic Systems Magazine*, 20(7):33–34, 2005.

[37] H Weinberg. Using the ADXL202 in pedometer and personal navigation applications. Application Notes AN-602, Analog Devices, 2002.

[38] Zhi-An Deng, Guofeng Wang, Ying Hu, and Di Wu. Heading estimation for indoor pedestrian navigation using a smartphone in the pocket. *Sensors*, 15(9):21518–21536, 2015.

[39] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which way am I facing: Inferring horizontal device orientation from an accelerometer signal. In *Wearable Computers, 2009. ISWC'09. International Symposium on*, pages 149–150. IEEE, 2009.

[40] Shane Colton and FRC Mentor. The balance filter. *Presentation, Massachusetts Institute of Technology*, 2007.

[41] Paul Lawitzki. Android sensor fusion tutorial. Code Project, 2014. [Online: `https://www.codeproject.com/Articles/729759/Android-Sensor-Fusion-Tutorial`. Accessed 15-June-2017].

Appendices

A ITU Recommendation factors

Table A.1: Power loss coefficients, N, for indoor transmission loss calculation.

Frequency	Residential	Office	Commercial
900 MHz	-	33	20
1.2-1.3 GHz	-	32	22
1.8-2.0 GHz	28	30	22
4 GHz	-	28	22
60 GHz[1]	-	22	17

[1]60 GHz values assume propagation within a single room or space, and do not include any allowance for transmission through walls. Gaseous absorption around 60 GHz is also significant for distances greater than about 100 m which may influence frequency re-use distances. (See Recommendation ITU-R P.676.)

Table A.2: Floor penetration loss factors, Lf (dB) with n being the number of floors penetrated, for indoor transmission loss calculation.

Frequency	Residential	Office	Commercial
900 MHz	-	9 (1 floor) 19 (2 floors) 24 (3 floors)	-
1.8-2.0 GHz	4n	15 + 4 (n – 1)	6 + 3 (n – 1)

B Raw results

The following sections include tables with raw estimated positions, real positions and their corresponding error. The coordinates frame of the room used during the experiments is shown in Fig. B.1.

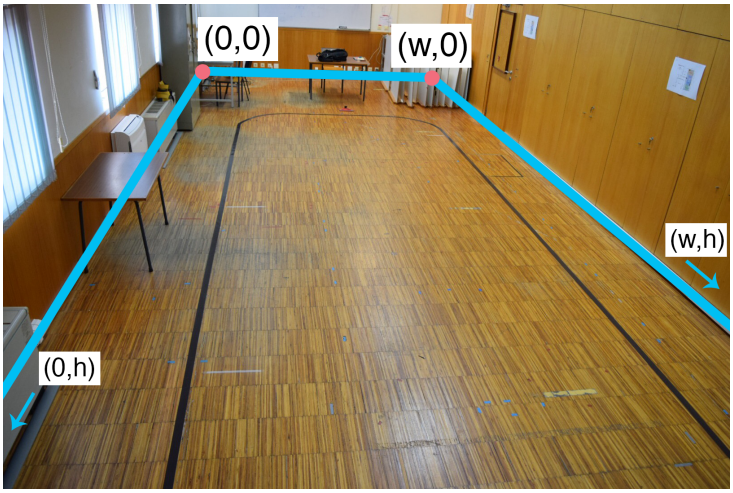


Figure B.1: Experiments room coordinates frame.

The width, w, is 518 centimeters and the height, h, is 1321 centimeters.

B.1 Fingerprinting results

Table B.1: KNN error results (cm).

Estimated position	Real position	Error
(245.6495, 581.2756)	(131, 720)	179.9693
(380.9347, 582.1658)	(380.9761, 725.8911)	143.7253
(380.9347, 582.1658)	(389.8937, 439.9019)	142.5457
(380.9347, 582.1658)	(385.9083, 449.2181)	133.0406
(380.9347, 582.1658)	(131.4894, 458.2348)	278.5352
(380.9347, 582.1658)	(136.2347, 452.1196)	277.1103

Estimated position	Real position	Error
(404.0756, 441.5202)	(136.3185, 455.0117)	268.0968
(404.0756, 441.5202)	(369.1559, 730.5622)	291.1437
(243.8694, 721.031)	(362.719, 730.6072)	119.2347
(357.7938, 722.8113)	(142.5919, 1044.3485)	386.9083
(357.7938, 722.8113)	(133.9052, 1046.5824)	393.6418
(129.055, 762.8686)	(132.6517, 1046.1014)	283.2557
(245.6495, 581.2756)	(140.4565, 720.4437)	174.4515
(243.8694, 721.031)	(131, 720)	112.8741
(245.6495, 581.2756)	(357.6804, 728.25)	184.8037
(268.7904, 440.6301)	(390.4349, 448.054)	121.8708
(131.7251, 579.4953)	(137.1487, 446.5922)	133.0137
(245.6495, 581.2756)	(356.157, 735.7643)	189.9439
(131.7251, 579.4953)	(142.2454, 1041.3168)	461.9413
(243.8694, 721.031)	(125.2899, 1040.7018)	340.9553
(243.8694, 721.031)	(135.9368, 720.919)	107.9327
(243.8694, 721.031)	(135.2777, 719.4211)	108.6036
(243.8694, 721.031)	(135.875, 719.4242)	108.0064
(245.6495, 581.2756)	(131, 720)	179.9693
(268.7904, 440.6301)	(360.2319, 724.4965)	298.231
(380.9347, 582.1658)	(393.9565, 443.4683)	139.3074
(404.0756, 441.5202)	(137.0516, 451.673)	267.2169
(245.6495, 581.2756)	(358.671, 727.605)	184.895
(373.8144, 904.4043)	(131.3847, 1044.771)	280.1338
(267.0103, 580.3854)	(132.695, 718.4603)	192.6273

Table B.2: WKNN error results (cm).

Estimated position	Real position	Error
(382.5779, 572.1787)	(131, 720)	291.792
(382.2829, 573.9715)	(357.8201, 727.6045)	155.5684
(381.3879, 579.4111)	(357.8201, 727.6045)	150.0558
(265.7591, 581.6531)	(359.1089, 727.5946)	173.2429
(379.6955, 589.6975)	(394.8471, 445.603)	144.8889
(380.8047, 582.9559)	(133.3914, 440.5594)	285.4648

Estimated position	Real position	Error
(379.9469, 588.1694)	(135.8943, 440.3716)	285.3171
(357.7938, 722.8113)	(331.4355, 652.1764)	75.3927
(381.3925, 579.3836)	(241.6915, 946.8131)	393.0914
(357.7938, 722.8113)	(132.6517, 1046.1014)	393.9612
(357.7938, 722.8113)	(129.9469, 1043.1135)	393.0747
(127.3236, 899.2821)	(130.2538, 1043.1336)	143.8814
(260.628, 760.2414)	(131.3934, 716.0408)	136.5844
(272.5121, 721.4786)	(131, 720)	141.5198
(257.6805, 596.4597)	(356.6104, 729.7611)	166.0012
(404.0756, 441.5202)	(386.8325, 448.1985)	18.4912
(382.7509, 571.1278)	(132.5781, 434.8861)	284.8652
(357.7938, 722.8113)	(357.0434, 728.2459)	5.4862
(357.7938, 722.8113)	(145.2736, 1038.9795)	380.9556
(357.7938, 722.8113)	(125.6062, 1037.7929)	391.3112
(227.3276, 720.7725)	(124.612, 1037.3107)	332.7866
(357.7938, 722.8113)	(131.6253, 717.9025)	226.2217
(382.7554, 571.1)	(133.8178, 713.3716)	286.7248
(382.1304, 574.8987)	(131, 720)	290.036
(252.1862, 589.5255)	(363.5184, 724.4352)	174.9156
(381.2567, 580.2088)	(394.8883, 443.3764)	137.5097
(382.3719, 573.431)	(134.6958, 454.8145)	274.6149
(244.1337, 579.3626)	(136.8911, 445.8382)	171.2593
(381.2314, 580.3623)	(357.125, 730.6839)	152.2422
(357.7938, 722.8113)	(125.4436, 1044.6972)	396.9851
(248.6051, 721.105)	(141.0134, 722.7335)	107.604

Table B.3: Using orientation category while matching results (cm).

Estimated position	Real position	Error
(127.4963, 887.4205)	(131, 720)	167.4571
(246.3593, 582.1715)	(359.4025, 732.5424)	188.1228
(385.3417, 555.3812)	(396.2394, 448.5781)	107.3576
(249.0022, 892.2515)	(132.9535, 447.174)	459.9579
(379.8444, 588.792)	(356.7107, 737.0779)	150.0796

Estimated position	Real position	Error
(131.8505, 569.6488)	(128.1353, 1038.0661)	468.4321
(261.5897, 762.6659)	(128.5227, 1033.4782)	301.7386
(127.2091, 907.1434)	(140.7027, 720.5746)	187.0561
(284.7307, 562.4324)	(131, 720)	220.1378
(245.5732, 581.1793)	(359.3372, 737.1025)	193.0137
(301.4946, 440.8453)	(401.6766, 441.9521)	100.1882
(383.0053, 569.5813)	(130.2203, 441.3696)	283.4404
(378.1366, 599.1722)	(356.7582, 732.5211)	135.0517
(258.3633, 1085.9973)	(132.9041, 1035.4793)	135.2481
(237.8673, 720.9372)	(137.2262, 722.0334)	100.647
(272.5788, 574.7439)	(131, 720)	202.8396
(251.4656, 588.6161)	(363.9474, 726.3933)	177.8615
(290.8711, 440.7753)	(398.1507, 448.4396)	107.5531
(382.8463, 570.548)	(128.6582, 444.5137)	283.7186
(382.6354, 571.8291)	(356.7107, 737.0779)	167.27
(248.4812, 893.063)	(355.156, 740.0663)	186.514
(131.7133, 580.4238)	(130.3751, 1034.056)	453.6342
(127.3481, 897.5999)	(136.1819, 724.9588)	172.8669

Table B.4: K = 3 results (cm).

Estimated position	Real position	Error
(199.9409, 836.612)	(131, 720)	135.4666
(304.0139, 549.0499)	(363.6883, 734.0933)	194.4277
(309.4154, 527.0507)	(394.6488, 451.2944)	114.034
(301.7419, 536.9309)	(127.992, 444.0669)	197.0096
(204.1358, 757.5659)	(360.5333, 740.052)	157.3751
(214.6521, 747.2996)	(130.3552, 1033.5721)	298.4257
(206.2462, 970.849)	(130.2236, 1035.0974)	99.5354
(194.0169, 825.7796)	(135.0933, 712.8846)	127.3471
(302.4709, 609.3146)	(131, 720)	204.092
(300.3554, 532.2097)	(360.2218, 732.4666)	209.0139
(307.5552, 521.4911)	(396.6971, 454.3009)	111.6279
(306.204, 537.0086)	(131.9866, 437.8168)	200.4763

Estimated position	Real position	Error
(302.4517, 736.0159)	(358.5287, 744.4551)	56.7084
(202.3844, 621.9409)	(133.1055, 1031.133)	415.0153
(216.5999, 852.7449)	(131.7036, 1032.1607)	198.4879
(297.9812, 750.2296)	(138.9465, 718.9722)	162.0773
(298.2578, 624.985)	(131, 720)	192.3617
(206.5214, 628.3366)	(362.7548, 737.1061)	190.3672
(234.2878, 523.3245)	(394.2124, 460.5087)	171.8188
(308.192, 610.6182)	(129.4361, 437.9327)	248.5437
(304.0435, 541.0842)	(354.2983, 736.8615)	202.1245
(208.6718, 732.8405)	(127.3865, 1028.555)	306.6828
(216.9099, 880.4534)	(128.2597, 1028.2485)	172.3434
(205.6357, 634.8626)	(144.6965, 713.1193)	99.1851

B.2 Dead reckoning results

Table B.5: Dead reckoning results (cm).

Estimated position	Real position	Error
(189.4867, 718.8524)	(131, 720)	58.498
(249.9174, 714.4319)	(283.1587, 708.3963)	33.7848
(308.1217, 695.2885)	(344.6702, 696.2517)	36.5612
(348.9006, 648.0972)	(385.2412, 628.7035)	41.1917
(377.5724, 586.732)	(412.1585, 540.4799)	57.7534
(395.1215, 521.6718)	(417.16, 478.3278)	48.625
(393.0401, 455.1236)	(401.387, 427.2562)	29.0907
(348.2339, 408.6869)	(370.3089, 368.5986)	45.7644
(284.764, 392.2461)	(303.3033, 360.599)	36.6776
(229.1234, 400.7152)	(244.5725, 355.37)	47.9047
(176.3376, 409.9197)	(239.9351, 355.7773)	83.5226
(118.2218, 443.9409)	(180.2436, 371.1379)	95.6399
(103.1702, 497.832)	(153.6776, 414.892)	97.1084
(95.9136, 550.2602)	(155.4047, 438.7074)	126.4248
(114.1708, 599.2901)	(169.8783, 474.3739)	136.7749
(134.0688, 643.116)	(176.8375, 487.3537)	161.5273
(168.5759, 694.7723)	(211.4264, 547.5411)	153.3401
(199.8243, 750.2514)	(265.1662, 589.7446)	173.2974
(222.9099, 804.4818)	(306.6241, 638.6362)	185.7763
(212.0334, 871.6466)	(354.1767, 697.7589)	224.5922
(166.7932, 911.8571)	(359.9004, 755.4998)	248.4713
(118.2152, 941.532)	(342.6608, 821.4822)	254.5345
(76.7784, 965.5829)	(340.2276, 829.3867)	296.5718
(22.9789, 998.5684)	(303.987, 886.3365)	302.5913
(0, 1025.6742)	(263.5499, 960.5397)	271.4794
(0, 1049.0925)	(227.9069, 1011.192)	231.0368
(0, 1038.366)	(179.0394, 1050.4926)	179.4497
(0, 985.0229)	(133.7108, 1019.9475)	138.1966
(0, 921.9554)	(127.2921, 954.3131)	131.3404
(0, 859.1596)	(125.1931, 868.3609)	125.5308

Estimated position	Real position	Error
(0, 790.2399)	(120.4572, 811.2336)	122.2729
(27.2039, 742.0327)	(121.4542, 762.7248)	96.495
(81.1622, 729.9548)	(167.8793, 740.8133)	87.3944
(130.544, 722.3405)	(181.1111, 727.7595)	50.8567
(191.6765, 730.3399)	(231.2081, 724.3438)	39.9837
(246.903, 742.9024)	(289.7697, 723.7998)	46.9305
(295.2343, 749.0745)	(303.5408, 723.6718)	26.7263
(344.5919, 725.4984)	(352.1776, 707.5298)	19.5041
(394.0091, 694.0979)	(373.5516, 671.8588)	30.2174
(431.9349, 637.6225)	(405.8879, 616.0803)	33.8011
(450.6439, 577.5737)	(406.0776, 564.271)	46.5093
(457.5706, 512.2576)	(418.2406, 483.7401)	48.5809
(423.3531, 455.9447)	(413.3187, 426.4188)	31.1844
(362.0852, 430.4893)	(373.4772, 385.0453)	46.8501
(299.3603, 440.4747)	(311.8032, 359.871)	81.5584
(244.3555, 459.8529)	(253.5443, 368.2433)	92.0693
(220.6633, 518.1318)	(180.0201, 381.0849)	142.9466
(239.0037, 576.2851)	(152.0606, 433.0706)	167.5395
(276.6189, 629.1064)	(181.4736, 492.3417)	166.605
(310.1705, 688.2482)	(224.8912, 558.4263)	155.3264
(333.7753, 748.0704)	(275.359, 610.142)	149.7889
(334.141, 816.0944)	(324.9189, 651.2731)	165.0792
(303.7327, 871.7592)	(356.2538, 718.5408)	161.9702
(250.3964, 912.4385)	(358.4199, 779.9128)	170.9741
(196.3108, 943.0772)	(343.132, 842.575)	177.9246
(138.6935, 973.1356)	(319.5286, 896.6302)	196.3528
(83.8513, 1009.2354)	(278.3649, 958.4112)	201.0438
(19.5529, 1020.0378)	(236.6685, 1005.6852)	217.5895
(0, 991.6796)	(176.5946, 1054.1562)	187.3205
(0, 945.2748)	(135.6799, 1037.8299)	164.2422
(0, 894.0949)	(126.7578, 975.6941)	150.7514
(0, 845.2309)	(119.5904, 964.9344)	169.2063
(0, 782.3119)	(118.235, 905.2219)	170.5473

Estimated position	Real position	Error
(2.2255, 719.3118)	(119.9996, 828.6861)	160.728
(62.4619, 697.2373)	(122.687, 765.3326)	90.9068
(123.7801, 705.5828)	(180.7214, 721.8378)	59.2159
(188.0611, 728.3237)	(246.0198, 718.1468)	58.8453
(254.6055, 721.8927)	(313.4008, 735.5679)	60.3646
(302.3606, 685.5948)	(369.8514, 700.8177)	69.1863
(335.3261, 632.0561)	(400.0688, 630.2463)	64.768
(364.7337, 571.0048)	(412.9274, 557.0491)	50.1736
(369.2638, 504.6715)	(410.6624, 489.5898)	44.0602
(337.5007, 448.5186)	(415.3921, 432.3401)	79.5539
(273.3459, 434.8857)	(380.094, 384.5204)	118.0331
(208.7878, 447.8317)	(324.9353, 365.5934)	142.3143
(152.1113, 485.1236)	(259.8105, 350.5572)	172.3578
(137.4793, 546.9416)	(188.2083, 373.7918)	180.428
(156.7095, 598.2747)	(161.3069, 420.4104)	177.9237
(177.3023, 644.5091)	(162.7577, 458.2568)	186.8193
(209.5611, 699.9639)	(183.134, 489.6149)	212.0026
(237.3123, 760.378)	(225.4349, 551.5245)	209.1909
(249.9801, 825.1187)	(270.8975, 595.865)	230.206
(249.2714, 881.0996)	(306.6241, 638.6362)	249.1543
(241.1876, 936.7598)	(320.3524, 653.2244)	294.3797
(193.432, 979.6796)	(355.9862, 691.0903)	331.2215
(136.41, 1016.5728)	(353.9792, 752.696)	342.0048
(80.1448, 1051.9968)	(330.7624, 822.4923)	339.8257
(30.3994, 1090.6826)	(298.1234, 880.6607)	340.2725
(0, 1118.1258)	(266.8258, 948.1528)	316.365
(0, 1094.6259)	(226.3799, 1015.8594)	239.6915
(0, 1036.3859)	(177.7454, 1038.947)	177.7639
(0, 974.4553)	(138.9585, 1000.19)	141.3214
(0, 909.7536)	(133.9763, 955.2451)	141.4889
(0, 841.9318)	(124.4288, 871.1019)	127.8022
(43.67, 797.2232)	(125.6662, 792.7428)	82.1185
(106.7227, 782.8714)	(141.5368, 720.7428)	71.2179

Estimated position	Real position	Error
(172.7074, 793.1471)	(192.7477, 696.9187)	98.293
(231.5723, 793.2211)	(245.2743, 704.1404)	90.1284
(282.7556, 791.1299)	(256.6362, 703.9863)	90.9738
(336.6618, 761.7831)	(317.67, 711.1318)	54.0947
(370.5766, 708.1634)	(373.0377, 705.6547)	3.5144
(399.7725, 648.7253)	(404.9871, 626.1548)	23.165
(405.2519, 583.2626)	(420.1748, 535.2412)	50.2867
(376.1227, 525.4218)	(415.5204, 481.1697)	59.2488
(313.4794, 504.7064)	(413.3187, 426.4188)	126.8733
(247.0853, 505.5605)	(370.3089, 368.5986)	184.2352
(185.4416, 526.4617)	(299.5457, 367.7314)	195.4866
(161.81, 580.4278)	(230.2106, 363.4712)	227.4836
(188.7967, 637.2841)	(169.3708, 385.2346)	252.797
(216.0867, 695.7896)	(153.737, 435.8964)	267.2676
(241.7597, 757.4158)	(188.956, 504.8357)	258.0405
(257.8599, 822.0315)	(228.6476, 549.0578)	274.5323
(259.3136, 876.1587)	(270.0333, 602.1386)	274.2296
(255.9878, 924.0048)	(277.4729, 595.6255)	329.0813
(213.456, 974.2693)	(320.774, 664.0788)	328.2306
(158.8029, 1005.9604)	(355.7162, 699.3789)	364.3721
(96.3991, 1035.653)	(347.3204, 769.2746)	365.9494
(38.5279, 1070.4705)	(312.5274, 847.3814)	353.3334
(0, 1097.9838)	(283.647, 893.9)	349.4364
(0, 1103.5999)	(250.7368, 951.769)	293.1238
(0, 1085.6677)	(245.2444, 967.1018)	272.4017
(0, 1024.5502)	(208.6054, 1016.9851)	208.7425
(0, 958.2982)	(151.8431, 1036.978)	171.0171
(0, 889.1271)	(137.02, 988.4549)	169.2351
(0, 822.1262)	(133.5035, 919.0259)	164.9628
(38.3155, 777.9685)	(129.1924, 845.3266)	113.1181
(76.0563, 743.0587)	(130.3831, 816.9052)	91.6772
(136.2526, 738.1871)	(128.1692, 765.3296)	28.3206
(199.726, 753.3864)	(173.3395, 730.7475)	34.7673

Estimated position	Real position	Error
(264.527, 740.3029)	(242.0096, 725.7411)	26.8157
(314.673, 700.3743)	(316.4131, 725.0737)	24.7606
(348.0449, 646.1465)	(371.7599, 712.0807)	70.0694
(375.0143, 583.1863)	(403.9887, 634.0492)	58.5367
(372.9626, 516.712)	(412.1976, 575.3379)	70.5435
(330.6747, 467.0071)	(411.1454, 500.201)	87.0481
(265.4785, 458.2032)	(414.4768, 432.4032)	151.2155
(198.4943, 465.7037)	(380.8158, 381.1575)	200.9706
(149.9309, 511.443)	(315.8329, 376.4517)	213.8834
(157.187, 572.9194)	(251.6993, 347.7848)	244.1683
(179.284, 616.963)	(201.1956, 362.5891)	255.3159
(209.0169, 663.6865)	(177.3632, 381.3067)	284.1484
(239.5351, 724.4235)	(162.9247, 432.2806)	302.0208
(260.4832, 787.2867)	(189.6922, 483.9133)	311.5233
(270.525, 854.5962)	(230.9904, 529.9477)	327.0469
(272.7348, 912.0255)	(269.5194, 567.7615)	344.2791
(270.6244, 963.7402)	(280.1904, 601.7822)	362.0844
(231.9379, 1009.1348)	(312.6831, 632.7246)	384.9733
(174.9948, 1038.5477)	(369.1667, 682.3881)	405.6506
(113.8145, 1068.7739)	(374.0242, 744.002)	416.1561
(57.3426, 1102.6967)	(347.7258, 805.0279)	415.8475
(2.5736, 1134.0203)	(315.271, 863.7413)	413.3164
(0, 1137.4629)	(277.2133, 909.168)	359.118
(0, 1078.4631)	(230.0245, 987.6352)	247.3075
(0, 1015.828)	(177.2162, 1030.0886)	177.7891
(0, 951.5929)	(141.489, 1011.3593)	153.5942
(0, 885.1454)	(132.1724, 950.5152)	147.4543
(16.0873, 823.028)	(128.1475, 862.8221)	118.9161
(60.9909, 802.5853)	(125.7821, 795.1224)	65.2196

B.3 Fusion results

Table B.6: Fusion using K as 2 results (cm).

Estimated position	Real position	Error
(127.3433, 897.9277)	(131, 720)	177.9653
(169.6195, 904.1707)	(175.3871, 747.0524)	157.2241
(212.2924, 908.8152)	(239.4703, 741.7753)	169.2365
(258.2772, 902.5896)	(308.7525, 747.67)	162.9351
(299.4482, 877.1455)	(362.8695, 707.5228)	181.0915
(299.4482, 877.1455)	(385.0024, 672.475)	221.8322
(327.046, 839.5736)	(405.6406, 631.7922)	222.1492
(347.2356, 796.7934)	(417.8381, 582.7564)	225.381
(367.249, 755.6094)	(423.003, 491.9564)	269.4837
(387.0855, 706.5361)	(421.8095, 419.9396)	288.6924
(402.113, 664.7528)	(413.7411, 395.8668)	269.1373
(415.6452, 619.3486)	(408.7276, 376.9144)	242.5329
(421.7503, 579.5008)	(396.5805, 362.0316)	218.9209
(399.148, 542.1147)	(372.479, 355.9836)	188.032
(357.7091, 524.8788)	(317.6493, 355.6495)	173.9061
(312.44, 520.1617)	(256.1612, 349.8561)	179.3635
(266.948, 528.7069)	(194.0476, 365.2539)	178.973
(231.4694, 555.7246)	(155.1949, 392.3844)	180.2715
(218.8473, 589.5871)	(121.6553, 443.9926)	175.0543
(210.4286, 625.2455)	(126.0052, 474.1222)	173.1057
(232.8312, 661.0878)	(152.8569, 504.4031)	175.9147
(232.8312, 661.0878)	(192.3863, 559.3724)	109.4615
(262.6641, 692.9763)	(207.8688, 571.0309)	133.6907
(294.3496, 724.7768)	(252.1459, 608.8354)	123.3838
(318.6647, 764.019)	(300.629, 624.5358)	140.6444
(332.4381, 805.5494)	(356.0183, 672.5486)	135.0749
(316.5682, 846.974)	(377.9506, 746.6261)	117.6329
(295.0054, 873.5788)	(394.0912, 787.3472)	131.3541
(271.0591, 897.8484)	(385.4609, 808.8062)	144.97
(233.7262, 918.0264)	(354.0413, 863.9996)	131.8886

Estimated position	Real position	Error
(193.6067, 932.5569)	(312.7932, 918.2029)	120.0477
(149.0614, 939.9554)	(275.9213, 956.2489)	127.902
(149.0614, 939.9554)	(224.2644, 985.737)	88.0423
(114.3571, 937.2128)	(224.2644, 985.737)	120.1425
(83.2518, 929.2771)	(206.0592, 996.481)	139.9929
(62.7662, 891.0244)	(159.2605, 978.5869)	130.301
(64.4131, 855.3259)	(133.3101, 939.9971)	109.1605
(68.2337, 822.7389)	(130.9153, 920.2709)	115.9374
(77.9441, 782.0321)	(129.6094, 863.9248)	96.8283
(100.6107, 747.3832)	(126.704, 797.3812)	56.3973
(142.7916, 740.8893)	(144.1855, 749.5973)	8.8188
(183.9529, 745.277)	(194.4917, 727.8844)	20.3364
(228.2321, 752.5224)	(257.5794, 718.9716)	44.5748
(265.2612, 752.5073)	(321.2329, 720.5538)	64.4504
(306.2823, 751.1295)	(335.2742, 721.4464)	41.4924
(335.7827, 726.4291)	(382.481, 689.9565)	59.2536
(354.6327, 686.7557)	(414.3056, 633.0881)	80.2562
(372.6776, 643.6201)	(419.1585, 559.9439)	95.7193
(382.5018, 597.878)	(420.9823, 467.4163)	136.0184
(373.6143, 554.2596)	(415.2599, 427.1065)	133.7994
(340.8998, 522.3431)	(385.6277, 380.8897)	148.3565
(300.4943, 502.6628)	(328.5786, 370.7925)	134.8277
(291.5775, 471.6921)	(264.8914, 365.4757)	109.5174
(244.6303, 471.681)	(249.2441, 354.6755)	117.0964
(211.3618, 489.6527)	(208.8421, 355.5386)	134.1378
(181.9689, 511.6858)	(199.3352, 367.6343)	145.0946
(168.895, 553.8558)	(159.466, 385.1112)	169.0078
(182.6342, 589.9638)	(148.6261, 473.4149)	121.4092
(212.8373, 621.8265)	(172.5557, 519.3051)	110.151
(241.3725, 653.5971)	(220.1932, 571.9034)	84.3944
(269.9512, 687.6723)	(275.0792, 598.4834)	89.3362
(286.09, 728.2648)	(333.2724, 627.7295)	111.0564
(290.1295, 772.2215)	(372.6963, 685.4139)	119.8033

Estimated position	Real position	Error
(270.4626, 812.6498)	(389.3311, 741.82)	138.3711
(236.789, 840.3771)	(388.3654, 802.6449)	156.2021
(236.789, 840.3771)	(383.2104, 814.8349)	148.6326
(198.9408, 858.927)	(351.2762, 854.8196)	152.3907
(161.1488, 876.7075)	(327.6273, 915.3602)	170.9068
(120.4263, 888.7999)	(284.4768, 957.9969)	178.0472
(76.547, 894.2777)	(242.7962, 1010.6129)	202.9105
(46.9824, 876.6752)	(203.5447, 1038.1013)	224.878
(19.0871, 855.6357)	(194.246, 1050.1565)	261.7613
(7.8864, 824.5897)	(152.7319, 1037.5364)	257.5393
(0.62082, 790.4637)	(145.2885, 1029.5783)	279.4718
(7.0663, 750.0545)	(130.3983, 988.5853)	268.5288
(17.1447, 710.4365)	(120.1644, 934.0446)	246.1984
(35.2864, 671.1048)	(120.7447, 876.2627)	222.2451
(60.558, 644.2671)	(123.6901, 805.3718)	173.0329
(87.6086, 622.677)	(123.329, 774.7987)	156.2593
(119.6803, 618.9892)	(130.7101, 754.0992)	135.5594
(153.5144, 629.121)	(160.2036, 736.5626)	107.6496
(191.817, 635.2677)	(176.8009, 734.4321)	100.2949
(191.817, 635.2677)	(180.3031, 732.8524)	98.2617
(127.3117, 900.1)	(131, 720)	180.1378
(156.1537, 884.7353)	(125.0947, 719.5305)	168.099
(197.801, 880.1699)	(169.4472, 724.6745)	158.0594
(243.1069, 883.3726)	(240.6374, 721.2929)	162.0985
(290.0345, 880.1713)	(315.1121, 716.2046)	165.8733
(290.0345, 880.1713)	(368.2754, 712.0585)	185.428
(329.3552, 853.6949)	(382.8335, 686.5329)	175.5081
(351.0589, 811.9791)	(400.5235, 645.7179)	173.4634
(366.0845, 769.3203)	(412.0917, 569.8668)	204.6909
(381.9555, 730.466)	(418.8176, 518.6278)	215.0215
(403.1689, 684.1233)	(423.8017, 489.156)	196.0561
(419.8588, 649.0376)	(422.2858, 475.6461)	173.4085
(435.7161, 612.1375)	(421.1646, 450.1443)	162.6455

Estimated position	Real position	Error
(445.6933, 571.7606)	(417.4668, 411.2457)	162.9778
(440.833, 527.1555)	(408.7276, 376.9144)	153.6331
(406.6671, 498.4669)	(357.0688, 355.3199)	151.4961
(361.856, 488.771)	(297.5646, 334.6939)	166.9525
(317.7582, 495.2649)	(241.6099, 345.6493)	167.8791
(279.6667, 521.7517)	(183.0848, 377.405)	173.678
(266.7792, 563.9453)	(147.2151, 430.9318)	178.8524
(264.2654, 596.2078)	(145.3058, 462.9092)	178.6614
(274.8445, 640.5906)	(144.9554, 487.3687)	200.8684
(285.6081, 673.2233)	(152.124, 507.0425)	213.1527
(310.944, 709.1477)	(173.1844, 528.9904)	226.7915
(341.2756, 739.8607)	(216.392, 595.8149)	190.6439
(294.5317, 661.9176)	(243.964, 605.5275)	75.7425
(326.6139, 693.9665)	(268.5832, 634.9085)	82.7974
(341.9639, 736.3307)	(318.9812, 664.3484)	75.5623
(334.2256, 781.6908)	(366.628, 699.1838)	88.6415
(307.3516, 815.5851)	(377.0953, 760.8813)	88.6379
(268.7051, 837.2963)	(348.5419, 839.9971)	79.8825
(228.3662, 855.7052)	(307.9831, 893.5118)	88.1373
(187.2828, 871.6294)	(284.0931, 916.5557)	106.7268
(148.2347, 892.5022)	(241.2588, 968.4287)	120.0763
(105.0942, 897.761)	(197.8499, 1018.1423)	151.9713
(117.2156, 820.075)	(178.2069, 1045.507)	233.537
(86.0765, 790.0894)	(154.7087, 1043.5663)	262.6041
(82.086, 746.1141)	(134.8137, 1017.9081)	276.8613
(86.7168, 704.1133)	(125.9983, 968.0399)	266.8338
(100.2462, 661.8454)	(118.8892, 890.5525)	229.4657
(132.9563, 634.1759)	(119.0286, 822.7841)	189.1218
(178.8832, 627.9893)	(142.5717, 769.241)	145.8443
(220.922, 637.5067)	(189.4858, 735.3084)	102.7299
(263.589, 653.2588)	(251.3433, 719.2715)	67.1389
(302.0935, 651.4067)	(312.8429, 714.7576)	64.2564
(338.5838, 647.6398)	(328.2517, 715.5726)	68.7141

Estimated position	Real position	Error
(372.769, 620.9631)	(374.852, 703.6691)	82.7322
(394.937, 584.214)	(402.4773, 651.2436)	67.4525
(416.3518, 543.5481)	(414.4099, 580.8703)	37.3727
(428.8505, 501.2264)	(417.4156, 508.5171)	13.5614
(421.8104, 456.2335)	(417.2236, 447.7848)	9.6135
(391.9237, 424.1067)	(395.3247, 405.3033)	19.1085
(349.8334, 406.6794)	(341.0433, 375.3058)	32.5817
(311.1026, 423.6665)	(310.3104, 360.4716)	63.1999
(264.8863, 424.8129)	(278.8409, 373.053)	53.608
(234.2358, 442.4137)	(235.1665, 371.012)	71.4078
(204.5347, 465.0116)	(219.6545, 373.9011)	92.3566
(195.8581, 509.1848)	(176.3007, 419.9721)	91.3313
(207.8625, 548.7421)	(161.2371, 487.653)	76.8492
(230.5833, 578.0875)	(177.5127, 538.0141)	66.5009
(252.5075, 604.4839)	(183.0666, 553.6302)	86.0706
(280.0746, 638.4104)	(219.5467, 591.3244)	76.6858
(306.6238, 675.1927)	(266.6331, 623.4847)	65.3679
(322.4245, 718.9514)	(332.2232, 658.0302)	61.7042
(319.6513, 761.9671)	(370.0382, 715.1464)	68.7824
(296.7666, 801.6007)	(389.048, 775.5916)	95.8767
(258.54, 825.628)	(390.3236, 844.3461)	133.1063
(258.54, 825.628)	(378.4579, 868.2278)	127.2597
(211.7381, 833.907)	(362.8847, 892.7936)	162.2126
(168.7269, 841.4341)	(328.0948, 921.0868)	178.1648
(126.0492, 854.7422)	(275.3246, 963.6124)	184.7589
(82.2771, 853.67)	(233.5203, 998.7359)	209.5677
(56.1177, 819.0197)	(189.8293, 1037.8994)	256.4901
(50.3482, 784.7653)	(145.4211, 1032.6552)	265.4963
(47.6227, 749.3234)	(134.8759, 1027.8931)	291.9147
(54.099, 711.1958)	(130.6375, 991.8021)	290.8573
(63.013, 669.9585)	(115.0758, 944.5395)	279.4732
(85.6715, 631.9861)	(114.9272, 883.8447)	253.5521
(121.7372, 616.6279)	(118.1916, 797.5367)	180.9436

Estimated position	Real position	Error
(157.3463, 618.7586)	(140.2133, 745.6888)	128.0813
(192.6196, 622.8642)	(194.7763, 721.9921)	99.1514
(159.9994, 759.1616)	(191.2132, 722.1635)	48.4062
(254.2009, 721.1925)	(131, 720)	123.2066
(293.5047, 720.7972)	(175.0449, 735.9269)	119.4221
(337.7362, 720.3083)	(245.15, 734.3772)	93.649
(380.2701, 702.9384)	(318.7417, 714.4704)	62.5998
(412.0348, 673.2846)	(381.7158, 661.913)	32.3814
(412.0348, 673.2846)	(398.1867, 606.0472)	68.6487
(433.5159, 630.8057)	(400.9176, 588.7451)	53.2141
(450.7685, 587.5024)	(403.2329, 537.8593)	68.7318
(461.4698, 543.2742)	(408.5533, 463.4736)	95.7512
(464.6261, 497.6086)	(406.771, 397.0152)	116.0442
(437.0158, 463.3053)	(361.0434, 364.9588)	124.2733
(396.4133, 445.3331)	(303.0196, 344.3695)	137.5356
(350.5493, 440.4757)	(239.3047, 338.9992)	150.575
(305.1729, 451.3536)	(169.1754, 359.8312)	163.9258
(278.9196, 488.3353)	(138.9869, 429.3414)	151.8599
(264.5892, 517.984)	(137.2085, 423.6406)	158.5135
(269.3085, 562.0724)	(141.285, 495.7468)	144.1842
(294.8918, 601.6046)	(165.7602, 520.1939)	152.6521
(323.0805, 638.9094)	(213.2164, 554.7232)	138.4104
(353.5554, 673.5861)	(269.4567, 596.9314)	113.7916
(373.5165, 715.1874)	(325.5765, 639.9249)	89.2339
(373.7449, 761.6574)	(372.3903, 688.8372)	72.8328
(378.5612, 664.4884)	(370.9936, 700.5967)	36.8928
(354.8452, 705.8239)	(372.7814, 771.9707)	68.5355
(322.2777, 732.7032)	(363.0832, 822.0398)	98.2147
(280.7372, 747.4243)	(333.2879, 868.7564)	132.2235
(238.7876, 762.4994)	(288.5097, 924.6127)	169.5671
(197.0347, 780.3901)	(243.4277, 969.8044)	195.013
(168.913, 794.8166)	(227.1653, 991.6455)	205.2679
(130.7006, 815.1787)	(205.5916, 1023.053)	220.9534

Estimated position	Real position	Error
(90.7311, 806.6804)	(163.915, 1046.7954)	251.0201
(75.4575, 769.4278)	(131.0708, 1034.5289)	270.8716
(75.4575, 769.4278)	(123.9248, 984.9485)	220.9032
(81.4912, 724.3179)	(126.7024, 963.2949)	243.2161
(88.6446, 682.2503)	(123.2515, 919.2922)	239.5548
(102.4649, 638.6045)	(119.8705, 846.9781)	209.0993
(134.3703, 609.9089)	(122.2728, 766.0538)	156.6128
(182.0394, 610.3481)	(151.7634, 739.717)	132.8644
(226.0167, 618.114)	(206.4354, 725.8709)	109.5216
(271.4853, 624.6803)	(276.4727, 718.0629)	93.5157
(315.4041, 611.294)	(348.7287, 700.1426)	94.8926
(346.5081, 579.4903)	(389.5366, 650.2864)	82.8465
(366.7549, 537.3738)	(407.7137, 592.4469)	68.6343
(382.2482, 496.4733)	(421.0015, 508.0762)	40.453
(381.1171, 451.4101)	(418.7239, 438.6752)	39.7045
(353.9122, 416.5005)	(397.8364, 382.1223)	55.7782
(312.6997, 399.4942)	(349.1453, 367.0968)	48.7635
(268.2576, 395.5966)	(281.6674, 372.5377)	26.6746
(226.1207, 411.5356)	(215.8359, 361.0782)	51.4949
(226.1207, 411.5356)	(172.5637, 376.0642)	64.2383
(203.2847, 436.2819)	(172.5637, 376.0642)	67.6014
(181.3339, 469.1723)	(165.4307, 409.5336)	61.7226
(192.8022, 513.107)	(148.1971, 442.5348)	83.4868
(206.717, 546.1915)	(154.232, 486.02)	79.8454
(235.7181, 584.4091)	(168.057, 514.9385)	96.9752
(267.6443, 618.5721)	(206.4915, 541.5782)	98.3246
(295.752, 658.2204)	(255.2621, 587.7649)	81.2613
(319.8954, 699.5141)	(306.1004, 629.9433)	70.9254
(329.5569, 745.8741)	(362.295, 659.5891)	92.2869
(312.2096, 790.1688)	(391.6819, 725.1052)	102.7088
(270.9425, 811.9863)	(396.5343, 774.0603)	131.1932
(228.3642, 827.2108)	(373.0234, 834.4634)	144.8409
(183.5469, 841.0075)	(317.642, 880.9642)	139.9215

Estimated position	Real position	Error
(143.207, 863.9789)	(278.483, 914.7834)	144.5015
(143.207, 863.9789)	(269.4243, 926.3127)	140.7704
(103.9224, 889.2361)	(237.5852, 959.196)	150.8646
(61.0844, 909.9807)	(208.8515, 1015.9574)	181.841
(28.2144, 901.5165)	(193.9898, 1038.6673)	215.1553
(0, 873.4855)	(180.6849, 1046.1283)	249.9051
(0, 831.8695)	(136.3003, 1039.5714)	248.4308
(8.7505, 788.5459)	(131.8534, 989.7471)	235.8734
(16.8074, 744.473)	(129.7216, 932.3379)	219.1867
(26.3943, 700.7258)	(127.7728, 877.2315)	203.5482
(44.1337, 660.9859)	(119.1465, 810.9708)	167.6974
(84.3908, 642.8877)	(119.3592, 745.1587)	108.084
(117.4544, 641.474)	(136.181, 726.2546)	86.8241
(150.5957, 648.6412)	(145.233, 724.3749)	75.9233
(183.976, 656.961)	(148.0926, 724.2374)	76.2478
(227.244, 616.8995)	(145.8044, 724.3474)	134.8238
(234.125, 668.9429)	(145.233, 724.3749)	104.7592

Table B.7: Fusion using K as 3 results (cm).

Estimated position	Real position	Error
(297.9272, 619.7776)	(131, 720)	194.7029
(350.8269, 641.8755)	(192.5835, 720.2797)	176.6018
(409.2763, 656.9376)	(259.4778, 732.4395)	167.7501
(468.002, 643.6615)	(333.4167, 717.4001)	153.462
(518, 607.6817)	(374.7063, 653.466)	150.4304
(518, 564.2943)	(404.5204, 567.7073)	113.5309
(518, 514.5006)	(407.0873, 514.2566)	110.913
(518, 453.4803)	(406.0792, 450.6199)	111.9574
(518, 453.4803)	(392.5865, 412.6422)	131.895
(504.728, 401.7147)	(368.6532, 407.1741)	136.1843
(483.5044, 351.0489)	(358.4018, 390.6388)	131.2175
(425.2486, 325.8989)	(311.6398, 379.3635)	125.5605
(364.6366, 337.4159)	(245.5141, 354.7666)	120.3795

Estimated position	Real position	Error
(305.3099, 365.1495)	(176.5531, 359.401)	128.885
(280.6117, 424.4761)	(144.4635, 425.275)	136.1506
(305.1336, 482.6143)	(159.8946, 502.5939)	146.6068
(336.9715, 531.1726)	(210.2892, 532.1411)	126.686
(364.1072, 590.2782)	(260.3662, 589.9327)	103.7415
(377.8529, 653.9407)	(310.2347, 629.6504)	71.8487
(374.8031, 711.9197)	(344.6541, 684.2816)	40.9002
(369.2269, 762.4562)	(351.3974, 693.0376)	71.6717
(340.4429, 819.0032)	(367.7632, 764.465)	60.9985
(295.4886, 859.581)	(365.6454, 821.4958)	79.8277
(243.8488, 895.5725)	(340.8499, 882.914)	97.8236
(191.1537, 924.6954)	(311.7663, 923.6594)	120.617
(138.3193, 956.7807)	(271.5378, 971.458)	134.0246
(83.0238, 981.7099)	(228.5212, 1015.8112)	149.4403
(106.2394, 861.3267)	(207.0706, 1037.984)	203.4078
(54.1769, 862.6104)	(190.8498, 1041.0636)	224.7777
(7.2143, 852.928)	(177.5011, 1039.2674)	252.4282
(0, 809.568)	(139.7508, 1031.7714)	262.4969
(0, 754.1479)	(129.6755, 990.9388)	269.9735
(3.7998, 695.289)	(127.6411, 920.8633)	257.3333
(17.9777, 632.5712)	(123.2115, 870.0833)	259.7809
(64.9409, 588.2156)	(123.4188, 795.4184)	215.2966
(115.3162, 592.1019)	(161.2631, 743.9426)	158.6402
(167.0583, 617.3848)	(211.419, 732.6109)	123.4704
(217.2654, 641.118)	(266.1045, 734.0983)	105.0266
(261.0815, 662.7594)	(278.3051, 743.2044)	82.2682
(322.8373, 677.1097)	(339.2724, 737.5896)	62.6732
(268.9517, 751.1673)	(379.3657, 702.0137)	120.8608
(326.052, 733.4628)	(386.6502, 681.3407)	79.9303
(382.777, 704.0878)	(406.2839, 631.8239)	75.9912
(436.1802, 660.9602)	(417.5518, 557.8246)	104.8045
(480.3602, 614.0232)	(421.0831, 505.6752)	123.5033
(498.6852, 551.1157)	(413.9086, 456.4619)	127.0686

Estimated position	Real position	Error
(467.6155, 494.0839)	(393.0176, 402.4972)	118.1226
(408.3805, 470.0154)	(327.1243, 367.6928)	130.6617
(341.1261, 473.7619)	(256.2659, 364.9758)	137.9699
(285.3042, 507.7953)	(193.3803, 365.4695)	169.4303
(265.682, 560.5303)	(144.0078, 406.0698)	196.6283
(256.5037, 617.2404)	(143.2779, 446.8138)	204.6101
(291.8531, 669.6269)	(137.5574, 478.8667)	245.35
(333.0928, 718.5262)	(175.8355, 517.798)	254.9935
(372.4146, 766.4208)	(228.1547, 579.1642)	236.3809
(392.3911, 828.5127)	(282.6871, 613.3191)	241.5435
(385.7303, 880.5995)	(326.7095, 680.5191)	208.604
(374.7986, 928.2135)	(329.5129, 689.2531)	243.2136
(334.2607, 980.3609)	(362.4346, 709.9733)	271.8514
(334.2607, 980.3609)	(355.9781, 751.3159)	230.0722
(283.3418, 1021.6564)	(348.7118, 804.1066)	227.1589
(227.0119, 1058.4688)	(328.4564, 857.2478)	225.3462
(175.2882, 1095.7233)	(299.6344, 911.0275)	222.6533
(121.6691, 1130.2502)	(261.2951, 961.3471)	219.1431
(72.8249, 1145.9409)	(225.518, 1008.3831)	205.5172
(23.9611, 1150.4672)	(213.0908, 1016.6102)	231.7061
(0, 1115.8377)	(177.2032, 1038.8443)	193.207
(0, 1060.8997)	(141.5202, 1009.1056)	150.7004
(0, 997.721)	(136.8704, 958.7143)	142.3201
(0, 997.721)	(131.7655, 894.5344)	167.3607
(0.92847, 935.2233)	(132.2846, 889.8147)	138.9833
(13.2529, 867.7681)	(126.053, 842.8635)	115.5167
(40.1691, 821.1394)	(119.7888, 768.883)	95.2367
(76.157, 779.3796)	(123.7797, 756.2)	52.9642
(129.161, 768.3053)	(135.1075, 726.2171)	42.5062
(189.2515, 784.2285)	(191.5173, 732.0696)	52.208
(252.8127, 807.0276)	(262.8997, 721.9823)	85.6413
(320.599, 803.0456)	(321.9512, 737.1121)	65.9474
(375.6378, 769.8552)	(374.0102, 710.2132)	59.6642

Estimated position	Real position	Error
(407.5908, 734.0899)	(397.6375, 674.2623)	60.65
(450.0727, 686.7686)	(407.7212, 629.7758)	71.0059
(497.7542, 634.8879)	(416.4936, 574.6794)	101.1353
(518, 582.5547)	(418.741, 516.6521)	119.1449
(512.4499, 521.5113)	(423.4695, 421.0649)	134.1902
(495.7983, 473.4578)	(419.1427, 407.9173)	100.8545
(447.4224, 431.5368)	(397.9786, 402.2326)	57.4754
(381.5199, 417.5294)	(352.5045, 380.4608)	47.0741
(323.333, 435.0972)	(288.7126, 377.2192)	67.4421
(278.5851, 479.2681)	(290.0362, 366.4209)	113.4267
(230.5022, 525.3194)	(235.5185, 373.4371)	151.9651
(230.2833, 583.6563)	(160.6816, 371.1398)	223.6239
(258.4511, 635.5708)	(145.7521, 443.6914)	222.5281
(295.1763, 684.5141)	(169.583, 488.7854)	232.5583
(322.8633, 742.3318)	(207.6667, 537.1707)	235.2899
(342.1553, 801.5756)	(262.1505, 587.7144)	228.3361
(336.7015, 864.6123)	(312.6415, 643.4651)	222.4522
(301.204, 916.3147)	(352.747, 703.1825)	219.2761
(249.4025, 958.6287)	(365.205, 738.3044)	248.9036
(198.098, 998.8627)	(351.6138, 811.4586)	242.2548
(198.098, 998.8627)	(338.459, 849.6524)	204.8533
(154.5102, 1030.8203)	(330.154, 865.2552)	241.3764
(106.6622, 1067.759)	(294.5061, 922.393)	237.5218
(46.2678, 1089.0202)	(255.6401, 968.2801)	241.6918
(0, 1075.5528)	(230.6478, 1001.4497)	242.2595
(0, 1054.3941)	(222.2322, 1017.9707)	225.1973
(0, 1000.293)	(167.3038, 1031.8304)	170.2503
(0, 946.2511)	(139.3593, 1010.6911)	153.5367
(0, 898.1453)	(131.7219, 983.5546)	156.9886
(1.7397, 837.1975)	(130.7431, 956.424)	175.6611
(9.3553, 773.0472)	(126.0237, 907.0871)	177.7026
(9.3553, 773.0472)	(128.0932, 817.6924)	126.8538
(72.1312, 781.5522)	(154.7959, 738.1752)	93.3542

Estimated position	Real position	Error
(134.3567, 796.9874)	(218.5504, 719.3906)	114.4983
(186.5515, 802.113)	(270.5139, 722.1667)	115.9357
(241.1557, 802.4371)	(286.9645, 716.3465)	97.5194
(285.1379, 774.3425)	(333.4167, 717.4001)	74.6544
(333.6426, 741.4426)	(351.5988, 709.7488)	36.427
(380.5341, 696.6685)	(390.9477, 681.3996)	18.4819
(420.6801, 644.7317)	(413.0523, 627.7186)	18.6448
(452.0796, 585.3)	(410.7759, 557.9324)	49.5477
(443.3701, 524.3091)	(420.3157, 480.1466)	49.818
(397.8469, 481.7687)	(411.6749, 421.6372)	61.7009
(334.714, 464.3599)	(368.6814, 368.7285)	101.4847
(269.3861, 477.0822)	(312.3585, 353.9953)	130.3726
(226.5315, 529.5047)	(231.3821, 333.2761)	196.2886
(270.0873, 568.3249)	(210.5774, 360.7426)	215.944
(282.1504, 631.7007)	(169.8793, 391.2762)	265.3465
(313.0914, 681.9725)	(144.1251, 446.7753)	289.5985
(343.7678, 739.8194)	(163.1167, 505.1418)	296.1561
(363.7348, 795.394)	(207.7282, 551.7319)	289.3255
(363.0434, 856.7954)	(260.335, 596.4672)	279.8568
(340.3607, 905.9256)	(311.1484, 647.3122)	260.258
(315.9515, 948.3026)	(315.2295, 662.3808)	285.9227
(264.1453, 986.4764)	(359.3584, 684.4957)	316.6352
(208.4036, 1023.0101)	(362.3813, 761.4881)	303.4846
(155.7207, 1057.2338)	(348.1559, 811.3016)	312.2723
(109.0375, 1095.7648)	(322.7709, 868.684)	311.8455
(109.0375, 1095.7648)	(294.8236, 927.6972)	250.5258
(53.473, 1124.3312)	(290.3128, 933.9718)	303.8582
(6.171, 1127.9337)	(268.1995, 971.7733)	305.0328
(0, 1124.7446)	(261.3065, 983.418)	297.0762
(0, 1093.4091)	(241.2791, 1023.9549)	251.0767
(0, 1039.9693)	(192.6829, 1037.6918)	192.6963
(0, 974.4725)	(146.3165, 1018.4069)	152.7702
(2.4456, 911.2413)	(136.5222, 953.9351)	140.71

Estimated position	Real position	Error
(44.8999, 869.8981)	(127.8673, 835.1331)	89.9566
(108.107, 869.8893)	(123.3231, 772.7581)	98.3159
(169.5765, 887.056)	(140.2177, 721.9673)	167.6789
(233.6787, 897.3476)	(191.4616, 733.5152)	169.1844
(286.2801, 884.2837)	(238.0184, 724.4095)	166.9999
(339.0937, 867.7362)	(262.4912, 720.4379)	166.0264
(393.8219, 833.2159)	(316.5794, 718.595)	138.2185
(441.4186, 788.8396)	(375.7583, 686.4932)	121.598
(489.679, 744.7974)	(406.621, 623.5593)	146.9602
(518, 686.9948)	(417.814, 565.1206)	157.7674
(506.9083, 628.0433)	(414.1627, 489.02)	167.1204
(506.9083, 628.0433)	(421.4965, 456.1571)	191.9376
(457.2211, 588.2789)	(415.8146, 427.9784)	165.5619
(390.8017, 578.0499)	(390.0466, 402.6557)	175.3958
(333.2706, 600.5272)	(333.0823, 385.1225)	215.4047
(289.1213, 647.7746)	(273.3777, 349.1822)	299.0072
(286.955, 695.4565)	(227.0829, 356.0052)	344.6909
(289.7217, 753.1585)	(192.621, 358.3212)	406.6018
(319.0708, 807.9746)	(158.2173, 405.2764)	433.6354
(351.2761, 859.3131)	(146.1571, 461.445)	447.6302
(369.5707, 916.7459)	(169.5419, 502.2947)	460.197
(368.6389, 979.1654)	(211.2154, 558.7682)	448.9052
(343.8257, 1039.4624)	(263.3433, 596.4463)	450.2673
(316.0253, 1084.5967)	(305.4495, 627.6334)	457.0857
(316.0253, 1084.5967)	(311.5311, 645.3921)	439.2277
(268.8095, 1128.4546)	(339.2264, 707.8619)	426.4467
(221.8507, 1166.6848)	(351.0831, 769.4022)	417.7733
(167.8958, 1198.4522)	(348.9855, 836.394)	404.8204
(111.4136, 1228.1495)	(336.7488, 879.8956)	414.7972
(49.9315, 1234.498)	(309.6596, 942.612)	390.7123
(0.46126, 1208.4799)	(278.6517, 980.9478)	359.3894
(0, 1161.1603)	(244.0176, 1017.7975)	283.015
(0, 1117.7069)	(231.155, 1033.4276)	246.0399

Estimated position	Real position	Error
(0, 1054.862)	(180.6363, 1038.812)	181.3479
(2.2961, 991.5317)	(136.8484, 1007.7403)	135.525
(2.7576, 926.518)	(131.2397, 966.2738)	134.4923
(22.3446, 865.3644)	(125.1717, 911.4259)	112.6725
(78.2848, 835.3217)	(125.1577, 849.4739)	48.9628
(140.4753, 833.0169)	(123.0132, 822.6852)	20.2896
(195.5484, 841.7828)	(118.5892, 787.0744)	94.4231
(248.7323, 851.5301)	(126.9918, 745.6287)	161.3563
(297.963, 859.76)	(137.6547, 730.6046)	205.8637
(358.312, 848.4321)	(177.9893, 718.4239)	222.3025
(411.2754, 818.0337)	(237.33, 707.3809)	206.1578
(460.2428, 782.3132)	(307.815, 710.452)	168.5178
(505.2836, 737.9998)	(362.8069, 701.7037)	147.0273
(518, 687.7973)	(405.0451, 608.6743)	137.9103
(518, 624.8578)	(421.2536, 540.2219)	128.5422
(488.6067, 570.2614)	(416.7736, 511.2828)	92.9434
(428.8923, 542.7973)	(414.6402, 450.2633)	93.6251
(367.395, 542.4855)	(386.9125, 416.2557)	127.7298
(301.1984, 543.3766)	(391.4956, 374.5655)	191.4439
(246.1646, 575.5916)	(346.1073, 355.4832)	241.7359
(222.1542, 634.172)	(278.5155, 356.2388)	283.5903
(237.9834, 685.0474)	(216.422, 338.159)	347.5578
(270.097, 733.9501)	(160.7251, 378.1272)	372.2528
(303.2723, 787.5091)	(134.4053, 435.0926)	390.7856
(327.6328, 847.188)	(146.9222, 494.988)	395.855
(333.9387, 910.1428)	(190.4093, 537.5495)	399.2824
(307.6459, 965.6815)	(238.4135, 574.5321)	397.2292
(260.5085, 1008.3329)	(289.6038, 625.6073)	383.83
(207.3027, 1044.0505)	(326.4197, 678.7518)	384.2291
(162.9665, 1079.578)	(343.2769, 730.1188)	393.2348
(125.2491, 1109.7394)	(346.0092, 748.1138)	423.684
(76.8177, 1147.901)	(327.831, 803.1949)	426.4153
(29.6849, 1182.3718)	(305.2162, 856.9219)	426.4214

Estimated position	Real position	Error
(0, 1187.0771)	(273.0873, 920.0614)	381.9347
(0, 1146.1325)	(245.797, 965.0046)	305.3252
(0, 1146.1325)	(239.7763, 988.1399)	287.1486
(0, 1088.3864)	(222.2848, 1008.5559)	236.1851
(0.036364, 1018.3296)	(182.4688, 1032.9935)	183.0208
(3.1957, 952.8454)	(140.2017, 1025.7718)	155.206
(7.8233, 892.6495)	(127.7729, 971.5165)	143.5545
(16.3793, 843.7345)	(128.8642, 952.0862)	156.1825
(62.9122, 795.6857)	(128.2864, 904.2672)	126.7428
(128.2892, 814.1195)	(128.4703, 793.3348)	20.7855
(195.7508, 826.9199)	(145.1606, 744.8264)	96.4298
(261.5234, 810.4828)	(193.4388, 729.2067)	106.0251
(308.5042, 766.3846)	(251.1372, 724.737)	70.8907
(348.3134, 716.4953)	(322.5901, 737.1297)	32.9768
(348.3134, 716.4953)	(372.6625, 716.6842)	24.3499
(381.3766, 661.5272)	(395.6377, 686.7912)	29.0112
(411.5516, 601.2576)	(406.9139, 657.5376)	56.4707
(413.7355, 533.243)	(415.4764, 590.8792)	57.6625
(367.7796, 487.2309)	(411.6377, 516.835)	52.9144
(311.4448, 474.0146)	(422.9074, 437.3152)	117.3489
(244.9226, 484.9116)	(401.1195, 405.4687)	175.2388
(193.6565, 527.9511)	(368.5987, 354.0126)	246.697
(187.4838, 578.27)	(309.8755, 350.4412)	258.6226
(187.7003, 626.3991)	(292.3074, 347.8927)	297.5037
(217.5213, 681.4705)	(239.2286, 347.8018)	334.3741
(250.3236, 735.727)	(184.7873, 373.1142)	368.4875
(285.1367, 794.6544)	(150.49, 421.8119)	396.4105
(297.9727, 860.9403)	(168.7705, 486.0618)	396.5187
(303.4597, 734.704)	(197.0938, 547.1413)	215.6234
(291.2099, 796.4423)	(220.3109, 549.1022)	257.3009
(253.4819, 844.286)	(249.2027, 581.171)	263.1497
(213.5543, 880.6111)	(295.0159, 615.3666)	277.4719
(163.7495, 918.0394)	(332.8463, 675.2789)	295.8486

Estimated position	Real position	Error
(118.5154, 959.2381)	(359.2649, 727.4371)	334.2036
(67.1607, 1000.9329)	(343.0094, 797.714)	342.6229
(12.4072, 1035.0841)	(323.137, 851.8902)	360.7118
(0, 1034.6291)	(292.8578, 908.1353)	319.0085
(0, 1003.2541)	(261.0691, 943.645)	267.7878
(0, 967.1328)	(253.9492, 951.103)	254.4546
(0, 900.0377)	(221.4244, 992.345)	239.8946
(0, 834.5986)	(188.3925, 1030.3065)	271.6493
(0, 834.5986)	(170.0301, 1043.1035)	269.0438
(5.4207, 771.0009)	(137.427, 1029.539)	290.2889
(22.1455, 711.1077)	(124.4292, 992.4114)	299.3221
(82.6835, 697.3193)	(122.3129, 946.6627)	252.473
(144.4064, 709.3553)	(124.1632, 890.8544)	182.6245
(204.062, 731.4332)	(119.7807, 807.5858)	113.5894
(268.7979, 735.5419)	(127.0377, 737.3699)	141.772
(326.5607, 717.5307)	(167.0645, 739.9375)	161.0624
(376.2635, 676.6689)	(215.9381, 737.0823)	171.3301
(421.0723, 632.7786)	(291.8555, 737.7596)	166.4872
(453.0988, 585.968)	(340.8617, 743.5732)	193.4853
(481.2046, 544.7912)	(353.9841, 727.3052)	222.4779
(489.0086, 484.0514)	(392.6951, 676.0215)	214.7762
(453.2046, 428.0858)	(406.2839, 631.8239)	209.0712
(393.3766, 399.913)	(418.3143, 555.3509)	157.4256
(340.5473, 396.7174)	(414.1964, 514.0689)	138.5481
(281.4677, 400.7418)	(413.6271, 474.5132)	151.355
(224.938, 432.0158)	(402.049, 425.3753)	177.2355
(214.5267, 495.0136)	(353.1209, 397.8283)	169.273
(262.5184, 551.9296)	(323.4152, 402.8476)	161.0399
(286.2985, 592.7467)	(282.351, 398.346)	194.4408
(323.4644, 640.9355)	(210.7547, 399.0059)	266.8959
(353.0161, 698.3192)	(161.8204, 411.6011)	344.6202
(364.138, 758.9145)	(147.1182, 467.1795)	363.6027
(342.6087, 820.1364)	(167.1184, 530.6618)	338.515

Estimated position	Real position	Error
(296.9429, 863.0226)	(215.6179, 574.8159)	299.4609
(240.9756, 889.3774)	(266.1208, 615.4309)	275.0981
(187.6619, 923.4531)	(313.593, 653.0373)	298.3008
(136.7163, 959.9743)	(352.2878, 704.8327)	334.0185
(94.6716, 1006.8291)	(344.0286, 782.4825)	335.4255
(94.6716, 1006.8291)	(338.6078, 802.448)	318.2397
(34.6959, 1028.7406)	(317.6433, 829.3147)	346.1646
(30.0941, 970.5749)	(220.7179, 972.9661)	190.6388
(32.2848, 906.7876)	(195.0377, 1020.2346)	198.3903
(33.9383, 844.1916)	(152.8133, 1028.2981)	219.1495
(55.5515, 784.6926)	(133.48, 967.0328)	198.2948
(104.2031, 769.094)	(131.2991, 921.1417)	154.4432
(159.3189, 768.2496)	(122.9196, 876.8852)	114.5714
(159.3189, 768.2496)	(149.2078, 725.1273)	44.2919

Table B.8: Fusion using low density radio map results (cm).

Estimated position	Real position	Error
(288.5788, 707.863)	(131, 720)	158.0455
(337.5317, 716.5297)	(194.0251, 712.3232)	143.5683
(397.0177, 726.0621)	(255.8365, 721.353)	141.2598
(460.262, 722.0908)	(324.6881, 716.6604)	135.6826
(517.2036, 692.2746)	(365.1264, 687.7743)	152.1438
(518, 647.5884)	(387.3721, 628.2371)	132.0535
(518, 595.5143)	(388.2739, 567.1365)	132.7937
(518, 595.5143)	(403.4315, 514.5185)	140.3078
(518, 536.9006)	(402.9417, 487.1608)	125.3493
(518, 470.9975)	(389.2808, 452.3904)	130.0572
(481.8969, 419.0299)	(346.2008, 385.8688)	139.6893
(417.7001, 403.2367)	(289.483, 367.0226)	133.2332
(353.7343, 412.1741)	(218.4559, 359.1983)	145.2814
(305.5268, 423.4026)	(197.032, 377.619)	117.7593
(249.6534, 457.1408)	(168.3468, 388.9023)	106.1473
(237.872, 511.8446)	(147.3723, 431.3671)	121.1067

Estimated position	Real position	Error
(233.9987, 561.5257)	(147.7709, 452.3944)	139.0858
(268.45, 612.166)	(171.2351, 504.7112)	144.9043
(303.7336, 668.4162)	(216.4166, 549.2451)	147.7363
(334.7133, 725.5407)	(273.9541, 587.8937)	150.4606
(355.8174, 786.0477)	(323.3964, 639.6913)	149.9044
(348.2464, 846.0381)	(364.5457, 694.5928)	152.3199
(305.7201, 896.6445)	(358.1186, 770.0653)	136.996
(265.3061, 931.2645)	(337.0654, 815.7033)	136.0286
(226.3272, 961.0175)	(329.4327, 830.1051)	166.6397
(174.9252, 999.2867)	(307.4803, 870.0318)	185.1423
(174.9252, 999.2867)	(298.0141, 895.7009)	160.8754
(132.4126, 1044.2239)	(271.4842, 935.0938)	176.7775
(85.2923, 1087.4457)	(244.3625, 978.2376)	192.95
(30.6485, 1104.0003)	(216.2377, 1013.0481)	206.6776
(0, 1068.4549)	(163.9301, 1035.136)	167.2819
(0, 1008.7989)	(133.465, 982.4533)	136.0404
(1.0272, 944.2746)	(125.1906, 933.9398)	124.5927
(4.2004, 883.2017)	(119.5156, 875.257)	115.5885
(13.66, 817.4756)	(121.3254, 809.6229)	107.9514
(42.4427, 770.7801)	(123.2979, 745.5575)	84.698
(92.8968, 763.0499)	(157.2354, 732.3777)	71.2757
(140.8352, 759.4697)	(178.0234, 705.7641)	65.3243
(140.8352, 759.4697)	(191.9189, 718.4026)	65.5443
(194.5784, 766.6409)	(216.8029, 724.9961)	47.2039
(253.202, 786.6968)	(293.2392, 728.2888)	70.8129
(314.9421, 782.8784)	(348.6296, 724.9875)	66.9791
(361.8375, 761.5092)	(393.8405, 655.6793)	110.5629
(401.3241, 734.1572)	(397.7755, 638.316)	95.9069
(447.0687, 691.0991)	(405.6776, 605.3486)	95.2175
(488.202, 649.2105)	(402.1163, 583.2581)	108.4457
(518, 597.9794)	(404.2925, 514.5069)	141.057
(518, 536.0471)	(409.9285, 439.5814)	144.8623
(471.9731, 491.8721)	(389.4101, 391.2113)	130.1892

Estimated position	Real position	Error
(411.4409, 476.5216)	(321.8576, 376.3493)	134.3861
(347.8641, 488.3284)	(258.1008, 353.9379)	161.6114
(301.4468, 533.3907)	(210.6894, 363.0836)	192.9803
(301.6045, 597.0342)	(154.1237, 415.5346)	233.8648
(311.1882, 646.6254)	(153.0514, 466.7427)	239.51
(338.594, 702.8193)	(163.1703, 496.8974)	270.5131
(307.233, 719.9925)	(188.2653, 532.5109)	222.042
(341.9101, 774.9825)	(200.4447, 542.171)	272.4218
(372.5481, 832.0392)	(248.5733, 585.7275)	275.7521
(387.2822, 884.2022)	(289.92, 611.3886)	289.6664
(396.1264, 930.8987)	(301.479, 637.5748)	308.2159
(377.9096, 989.1821)	(338.6201, 669.7724)	321.8171
(337.6127, 1034.985)	(360.5028, 726.8027)	309.0313
(287.8283, 1069.8121)	(347.3432, 789.3743)	286.6834
(240.4743, 1106.3655)	(319.2073, 864.8324)	254.0416
(192.952, 1143.6003)	(288.9064, 915.7484)	247.2321
(150.3855, 1177.6328)	(261.9465, 961.654)	243.0898
(87.6073, 1199.0634)	(229.9403, 1005.6114)	240.1715
(87.6073, 1199.0634)	(201.3693, 1031.2428)	202.745
(30.7941, 1171.314)	(181.3382, 1042.269)	198.283
(9.3676, 1113.5244)	(133.7762, 1026.8535)	151.6223
(0, 1053.9366)	(129.2175, 983.238)	147.2938
(3.0835, 990.3989)	(124.069, 930.4477)	135.0245
(11.6733, 928.4827)	(119.4598, 853.5319)	131.2843
(49.9019, 883.3507)	(112.5924, 780.2006)	120.7064
(109.2743, 875.4951)	(134.3476, 736.0891)	141.6429
(166.3641, 894.7432)	(186.9501, 719.8047)	176.1456
(226.4342, 909.7154)	(250.4709, 728.7777)	182.5273
(290.225, 904.5063)	(308.7248, 730.1643)	175.3209
(335.5827, 889.3263)	(333.5227, 723.099)	166.2401
(385.2711, 853.5066)	(369.6785, 698.0514)	156.2352
(427.1087, 811.8008)	(392.1117, 651.864)	163.7209
(466.2616, 762.6277)	(403.1109, 601.0197)	173.5084

Estimated position	Real position	Error
(497.4305, 707.474)	(401.953, 535.2976)	196.8773
(477.0667, 653.8866)	(418.444, 442.4684)	219.3953
(477.0667, 653.8866)	(409.6782, 433.3157)	230.6355
(426.5638, 619.75)	(385.4683, 421.2435)	202.7158
(367.4132, 615.2756)	(326.9485, 386.5597)	232.2678
(302.7085, 636.1436)	(265.9404, 381.9109)	256.8777
(301.5319, 689.6212)	(201.7435, 401.0266)	305.3597
(305.935, 738.2808)	(192.6678, 401.3363)	355.4731
(331.7145, 778.7064)	(167.3548, 430.7873)	384.7881
(359.9409, 819.9569)	(170.5044, 491.3897)	379.2658
(391.5669, 874.0887)	(200.5704, 519.8154)	402.4789
(407.7175, 938.3238)	(243.9637, 563.1761)	409.3301
(397.9496, 998.9236)	(287.3782, 613.4501)	401.0185
(357.4076, 1052.3729)	(337.8901, 655.0818)	397.7703
(312.258, 1095.0153)	(355.7318, 729.7599)	367.8335
(312.258, 1095.0153)	(346.6609, 750.2397)	346.4878
(259.9492, 1135.8603)	(336.7149, 785.1903)	358.9741
(214.5378, 1182.8994)	(305.5366, 848.2331)	346.8174
(165.447, 1223.5198)	(284.0502, 889.7834)	354.1846
(113.9159, 1258.1949)	(253.5293, 958.5438)	330.5793
(55.3053, 1266.5042)	(223.756, 1012.6318)	304.675
(2.9125, 1264.0961)	(210.1052, 1036.2373)	307.9747
(0, 1230.8652)	(180.0633, 1049.6882)	255.4367
(0, 1173.4655)	(140.5692, 1041.009)	193.1435
(0, 1112.3361)	(128.1591, 1012.1568)	162.6673
(0.86771, 1047.5111)	(128.5863, 972.1109)	148.3146
(2.8526, 984.8336)	(122.7035, 915.3988)	138.5115
(40.4383, 935.4222)	(120.8872, 864.6913)	107.1209
(103.0761, 919.8432)	(120.8091, 802.8736)	118.3062
(163.894, 937.2875)	(140.2673, 737.6563)	201.0245
(221.9419, 948.69)	(198.4735, 723.0743)	226.833
(269.8503, 959.4603)	(207.5413, 715.693)	251.6047
(319.232, 962.3747)	(247.5579, 727.2135)	245.8414

Estimated position	Real position	Error
(379.8205, 955.3958)	(267.6534, 730.6801)	251.1546
(128.1378, 829.4642)	(269.133, 729.2219)	172.9976
(177.8468, 795.6364)	(322.4868, 725.9279)	160.5616
(216.6116, 764.9673)	(361.8842, 706.1864)	156.7142
(257.2087, 732.3597)	(374.5793, 686.1979)	126.122
(300.4953, 687.1236)	(391.9704, 649.95)	98.7398
(334.8739, 631.5856)	(399.6056, 594.4468)	74.6289
(331.7338, 569.1454)	(398.5394, 503.8418)	93.4213
(288.4258, 523.6579)	(401.6849, 442.9004)	139.1022
(226.7742, 510.4787)	(382.6015, 391.4476)	196.0881
(159.652, 519.198)	(319.3875, 379.9287)	211.923
(114.9348, 539.2728)	(308.437, 355.4718)	266.8818
(86.4297, 595.2112)	(252.0387, 368.503)	280.7542
(90.0065, 645.786)	(208.9481, 359.5921)	309.9259
(109.6905, 692.7275)	(195.8951, 384.5087)	320.047
(124.0434, 741.6151)	(191.3744, 381.2676)	366.584
(144.6286, 786.1965)	(185.7663, 384.8843)	403.4151
(173.5515, 834.587)	(168.2446, 424.5521)	410.0693
(204.1303, 891.4856)	(191.0174, 491.0165)	400.6837
(221.797, 951.4878)	(230.0252, 544.3281)	407.2428
(222.5567, 1014.7173)	(275.4706, 598.7218)	419.3472
(204.4813, 1075.4008)	(315.9254, 633.7656)	455.4793
(160.9656, 1119.2771)	(355.7486, 689.3383)	472.004
(110.9742, 1156.5338)	(377.048, 748.1695)	487.3978
(53.0244, 1183.0846)	(378.9672, 804.1755)	499.8108
(5.5343, 1207.791)	(367.2116, 858.1404)	503.0567
(0, 1230.4263)	(362.7932, 876.4995)	506.8364
(0, 1252.237)	(337.8044, 921.6599)	472.6447
(0, 1265.0835)	(304.5806, 964.4091)	427.9888
(0, 1245.1132)	(258.9587, 1016.2195)	345.6182
(0, 1198.7992)	(212.5565, 1049.5794)	259.7053
(0, 1198.7992)	(138.644, 1040.0586)	210.7622
(0, 1137.5257)	(130.8877, 1030.025)	169.3753

Estimated position	Real position	Error
(2.7769, 1080.8533)	(129.3819, 1007.5538)	146.293
(2.8236, 1022.0314)	(129.6485, 962.8612)	139.9489
(21.0204, 967.4686)	(127.094, 890.8137)	130.8724
(72.5392, 940.2044)	(125.9166, 816.3819)	134.8375
(100.497, 873.3259)	(149.8339, 720.4661)	160.6246

Table B.9: Fusion using high density radio map results (cm).

Estimated position	Real position	Error
(333.9744, 502.5261)	(131, 720)	297.4786
(393.9494, 515.9401)	(189.1361, 727.3177)	294.328
(393.9494, 515.9401)	(195.6832, 716.9646)	282.3479
(440.8207, 526.0434)	(213.6558, 717.3589)	296.994
(503.1917, 540.9718)	(254.7846, 719.7987)	306.0803
(518, 542.2847)	(318.85, 716.5355)	264.6207
(518, 513.0277)	(364.5457, 694.5928)	237.727
(518, 466.3353)	(390.8075, 644.1446)	218.6185
(518, 411.4746)	(401.3012, 596.6489)	218.8792
(518, 352.5713)	(412.5949, 506.3421)	186.4288
(503.9366, 290.9154)	(380.1375, 452.6117)	203.6465
(450.0707, 254.4569)	(334.0248, 406.5194)	191.2843
(385.4924, 245.3681)	(263.1308, 428.008)	219.8402
(321.5933, 257.9905)	(202.2113, 435.9428)	214.2873
(280.3734, 306.9472)	(163.8796, 437.0131)	174.608
(281.2511, 358.5195)	(152.0077, 507.6258)	197.3234
(287.9995, 409.619)	(153.6735, 512.7765)	169.3663
(306.4267, 456.4419)	(163.27, 535.2343)	163.4077
(334.0939, 499.5207)	(177.1648, 542.3826)	162.6773
(361.6859, 538.9726)	(187.0442, 549.4746)	174.9572
(353.5693, 515.7548)	(203.3657, 563.3921)	157.5768
(381.2808, 568.7921)	(234.512, 583.5494)	147.5089
(398.6367, 624.9858)	(283.6994, 627.6171)	114.9674
(407.2261, 685.1002)	(330.3745, 656.8659)	81.874
(352.9775, 707.5509)	(342.6598, 802.8597)	95.8656

Estimated position	Real position	Error
(294.2592, 733.4237)	(308.4204, 877.5612)	144.8315
(242.7537, 772.8832)	(276.1908, 915.0326)	146.029
(196.6621, 820.1401)	(242.1911, 959.7399)	146.8367
(162.0581, 871.7963)	(211.1122, 996.4914)	133.9969
(162.0581, 871.7963)	(181.0943, 1040.6232)	169.8967
(117.987, 918.0575)	(181.0943, 1040.6232)	137.8582
(96.6821, 871.7489)	(126.2341, 1024.9772)	156.052
(84.8924, 815.243)	(121.3942, 993.1753)	181.6378
(81.8305, 751.6453)	(120.3386, 933.6427)	186.0267
(85.4421, 689.046)	(120.4208, 864.6685)	179.072
(110.9104, 629.0875)	(112.6055, 767.72)	138.6429
(163.9825, 596.2678)	(128.8406, 731.6723)	139.8904
(217.9799, 599.896)	(166.7849, 722.3451)	132.7205
(263.7837, 612.3438)	(217.1527, 720.4835)	117.7652
(308.8924, 626.6078)	(241.1027, 707.0264)	105.1789
(370.1875, 645.8392)	(302.3901, 723.9358)	103.4194
(433.5261, 633.369)	(351.4345, 717.2326)	117.3547
(486.8048, 599.4855)	(392.2524, 653.7686)	109.0267
(518, 566.3957)	(404.7121, 615.9618)	123.6565
(518, 518.0319)	(404.8618, 590.0061)	134.0915
(518, 518.0319)	(413.2115, 542.7416)	107.6624
(518, 460.7719)	(410.9658, 509.0656)	117.4249
(518, 407.0229)	(411.7093, 484.1694)	131.3366
(518, 359.4446)	(410.6607, 457.9132)	145.6633
(507.9539, 304.4503)	(405.504, 423.8767)	157.3488
(490.525, 259.9264)	(405.4679, 400.8513)	164.6042
(434.75, 235.1063)	(375.0842, 395.1124)	170.7687
(370.4192, 238.5855)	(313.7461, 366.0634)	139.5079
(321.7086, 261.5646)	(259.6796, 368.2009)	123.3649
(275.6381, 286.8905)	(249.4704, 365.0655)	82.4383
(244.2081, 337.8491)	(188.6035, 381.3721)	70.6125
(231.3198, 387.3359)	(182.661, 404.9471)	51.7477
(238.5945, 438.122)	(170.7428, 448.8538)	68.6952

Estimated position	Real position	Error
(249.9778, 486.8298)	(159.7227, 463.7279)	93.1648
(284.975, 535.3448)	(166.4226, 488.7509)	127.38
(306.3953, 521.6152)	(255.8149, 560.7833)	63.9728
(330.8071, 575.6714)	(300.6283, 605.1339)	42.1758
(338.9074, 634.6177)	(332.2123, 655.0211)	21.4738
(320.768, 692.3991)	(368.5091, 699.7028)	48.2965
(277.2925, 737.918)	(381.8603, 758.3272)	106.5409
(228.2156, 773.392)	(369.9693, 801.4018)	144.4945
(175.5133, 808.5323)	(332.2689, 849.4194)	162.0003
(126.2579, 847.7589)	(297.5384, 895.6759)	177.8568
(77.3161, 888.4826)	(255.796, 941.2396)	186.1139
(32.3445, 921.7523)	(221.8824, 990.8215)	201.7305
(0, 927.6617)	(185.738, 1032.1906)	213.1312
(0, 927.6617)	(144.1016, 1038.0836)	181.5441
(0, 880.5476)	(133.1586, 1030.6232)	200.6337
(0, 816.4484)	(125.3203, 997.5016)	220.194
(0, 753.6847)	(120.0782, 938.9935)	220.8124
(2.2131, 689.2217)	(117.2701, 880.2946)	223.0402
(17.193, 640.3531)	(117.677, 802.7555)	190.9753
(74.7953, 623.2468)	(122.0398, 752.3084)	137.437
(133.3642, 632.6341)	(173.2834, 737.0863)	111.8204
(193.434, 653.7641)	(230.6988, 728.307)	83.3385
(254.5482, 665.3424)	(287.9236, 742.9223)	84.4544
(220.9179, 732.2911)	(287.9236, 742.9223)	67.8438
(275.5092, 723.2734)	(342.226, 729.442)	67.0014
(313.5793, 689.7473)	(374.9578, 716.1443)	66.8141
(348.0168, 655.3542)	(392.318, 682.9852)	52.2117
(384.5809, 604.1742)	(403.4852, 634.3799)	35.6337
(416.3321, 550.9419)	(408.8804, 557.5004)	9.9268
(429.6529, 491.0511)	(408.6334, 495.4467)	21.4742
(391.282, 439.9819)	(404.1877, 436.6092)	13.3392
(330.8301, 415.3354)	(366.5273, 408.777)	36.2947
(268.5021, 422.8253)	(314.6473, 373.114)	67.8277

Estimated position	Real position	Error
(217.7579, 458.1348)	(244.1362, 396.253)	67.2694
(220.0971, 509.8225)	(179.9626, 405.0376)	112.2081
(232.1519, 559.2092)	(176.9995, 427.4079)	142.8753
(255.5459, 607.6191)	(162.1919, 449.0742)	183.9877
(279.2235, 653.7069)	(163.9899, 502.1941)	190.3547
(294.0794, 547.3006)	(179.5117, 491.2259)	127.5544
(320.088, 601.4396)	(201.308, 501.5863)	155.1753
(335.1027, 651.7455)	(235.795, 511.5921)	171.7702
(348.3366, 702.1203)	(245.3125, 549.0194)	184.5369
(352.9771, 759.8488)	(286.4178, 585.6764)	186.4569
(326.6546, 818.4467)	(330.774, 645.5429)	172.9528
(281.8009, 859.1591)	(352.7695, 702.7272)	171.7775
(234.9497, 891.9208)	(360.6507, 747.733)	191.2874
(188.0331, 934.1594)	(332.8205, 810.8644)	190.1711
(143.2278, 983.3984)	(306.0428, 852.3319)	209.0147
(143.2278, 983.3984)	(285.2964, 884.5587)	173.0687
(105.0038, 1034.9579)	(274.0518, 897.8285)	217.6733
(57.9064, 1075.5441)	(241.288, 953.204)	220.4448
(3.2692, 1085.2776)	(219.7978, 986.8235)	237.8609
(0, 1080.3843)	(215.9384, 1004.5929)	228.8531
(0, 1033.4928)	(178.3065, 1046.8137)	178.8034
(0, 976.5248)	(131.6859, 1031.7617)	142.8016
(3.1565, 916.2793)	(125.1614, 985.6299)	140.3379
(6.7257, 852.555)	(128.8904, 934.8453)	147.2953
(33.7013, 793.3139)	(117.5059, 858.1169)	105.937
(80.1537, 767.2653)	(120.3642, 790.0998)	46.2417
(144.3946, 768.3492)	(136.6291, 734.7265)	34.5078
(197.2899, 780.6526)	(169.1932, 725.3688)	62.0139
(245.8684, 795.9084)	(184.9356, 725.7394)	92.9327
(295.3784, 809.6902)	(231.6125, 725.3409)	105.7398
(351.6008, 821.1355)	(241.8037, 728.5714)	143.609
(402.9727, 800.2329)	(297.686, 731.4028)	125.789
(452.9191, 762.2885)	(357.4644, 728.2701)	101.3353

Estimated position	Real position	Error
(452.9191, 762.2885)	(366.6504, 712.7685)	99.4712
(494.5741, 714.3271)	(392.778, 670.4885)	110.8344
(518, 659.423)	(402.3825, 628.3268)	119.7262
(518, 604.2631)	(409.9209, 562.2929)	115.9422
(496.3716, 543.6523)	(420.5568, 481.1508)	98.2564
(467.4172, 505.902)	(422.3514, 448.5351)	72.9512
(417.3675, 483.3242)	(405.2624, 439.7041)	45.2686
(352.731, 477.1312)	(367.0593, 415.3164)	63.4537
(303.7588, 500.106)	(318.8281, 351.3844)	149.4831
(263.4299, 525.3474)	(292.462, 373.9638)	154.1424
(250.8435, 576.0942)	(261.8788, 375.1352)	201.2618
(256.9941, 625.0037)	(238.6506, 382.923)	242.7747
(286.589, 675.877)	(176.6275, 381.8235)	313.9411
(318.9735, 731.8092)	(165.6107, 436.9648)	332.3452
(329.7718, 614.2976)	(172.2838, 471.9939)	212.2565
(354.2809, 672.0463)	(188.0685, 509.6841)	232.3532
(359.5478, 733.7486)	(240.6838, 539.361)	227.8491
(342.0662, 791.6211)	(284.6872, 576.7399)	222.4101
(301.1027, 838.9366)	(320.2773, 633.7962)	206.0346
(256.0669, 879.3277)	(345.1333, 695.9356)	203.8762
(213.5224, 914.3817)	(350.6123, 751.778)	212.6819
(170.0649, 961.7987)	(329.5743, 814.2536)	217.2851
(127.715, 1011.4834)	(304.2158, 878.2734)	221.1276
(92.3972, 1054.8312)	(278.1892, 914.3573)	232.9198
(27.3, 1069.3502)	(245.0252, 959.9228)	243.6773
(0, 1038.7742)	(211.8722, 1008.3087)	214.0513
(0, 1002.6489)	(205.9104, 1020.3427)	206.6692
(0, 944.0228)	(176.9167, 1041.5109)	201.9986
(0, 883.5335)	(131.1397, 1035.8348)	200.9809
(2.1705, 817.8284)	(124.1701, 982.8845)	205.2497
(8.8352, 760.9511)	(121.2028, 946.237)	216.6965
(8.8352, 760.9511)	(115.1329, 902.8892)	177.3291
(57.7345, 723.3196)	(114.5647, 861.6566)	149.5554

Estimated position	Real position	Error
(116.5449, 721.0347)	(112.1498, 810.3706)	89.444
(180.1418, 739.8347)	(118.4017, 741.2843)	61.7571
(231.6363, 751.7742)	(152.6524, 740.7957)	79.7432
(285.4786, 762.6772)	(171.1055, 713.3748)	124.5469
(347.7303, 755.5254)	(230.4924, 717.7282)	123.1801
(394.2007, 738.9664)	(255.8202, 712.0622)	140.9716
(443.5645, 707.226)	(292.6586, 731.2818)	152.8112
(488.7975, 667.301)	(348.8516, 726.5346)	151.9654
(518, 618.8676)	(388.5429, 679.4001)	142.9102
(518, 566.2146)	(411.6037, 615.9884)	117.4632
(518, 504.5271)	(410.9502, 567.0342)	123.9628
(518, 504.5271)	(411.1895, 514.4137)	107.2671
(476.2248, 457.1879)	(415.4839, 489.7493)	68.9181
(416.1672, 443.4993)	(409.4974, 451.901)	10.7273
(352.203, 451.5571)	(379.6885, 391.5487)	66.0034
(302.5108, 492.385)	(322.7905, 351.2177)	142.6165
(294.5221, 539.3347)	(270.7275, 353.407)	187.4441
(294.4186, 593.0633)	(252.5523, 339.4568)	257.0389
(319.3927, 635.8169)	(201.4901, 356.3184)	303.3487
(343.852, 678.1505)	(189.943, 363.9298)	349.8896
(372.4677, 730.2645)	(147.8988, 412.5472)	389.07
(394.6375, 788.6594)	(158.9239, 469.4551)	396.8025
(403.5967, 852.4298)	(201.3703, 519.8049)	389.2747
(401.6675, 900.9013)	(240.0364, 546.6578)	389.3752
(391.0264, 954.0962)	(253.7621, 565.4399)	412.1835
(354.0252, 1003.8303)	(289.9614, 592.256)	416.5304
(311.9972, 1039.863)	(336.7969, 647.5109)	393.1351
(262.0639, 1080.6553)	(368.9371, 691.269)	403.7866
(214.5271, 1126.0213)	(355.9111, 767.2817)	385.595
(214.5271, 1126.0213)	(343.028, 787.9573)	361.6625
(164.051, 1169.738)	(329.5025, 822.2863)	384.8336
(119.1393, 1211.1118)	(305.2072, 876.4787)	382.8846
(57.5717, 1234.6892)	(281.2798, 922.3003)	384.2293

Estimated position	Real position	Error
(10.6599, 1245.5224)	(272.5442, 938.0257)	403.903
(0, 1225.3362)	(250.0153, 959.6044)	364.8576
(0, 1171.1536)	(218.805, 1004.8014)	274.8612
(0, 1116.3628)	(172.5, 1040.3464)	188.5066
(2.3149, 1053.1906)	(127.389, 1020.6992)	129.2255
(13.0145, 990.8417)	(118.686, 965.6716)	108.6278
(265.0068, 771.2812)	(120.9428, 939.7091)	221.6358
(297.5535, 727.1961)	(118.6769, 908.4475)	254.6546
(339.4403, 704.8317)	(121.2767, 863.8037)	269.9397
(399.3257, 711.9375)	(122.5855, 791.3618)	287.9121
(455.8572, 713.2729)	(134.4828, 738.9147)	322.3957
(508.9969, 699.3402)	(170.548, 705.6145)	338.5071
(518, 684.1522)	(189.6218, 713.771)	329.7112
(518, 651.1835)	(238.8802, 727.0092)	289.2359
(518, 614.1345)	(300.2518, 740.3381)	251.6776
(518, 571.5288)	(358.7011, 712.605)	212.7877
(518, 514.1753)	(388.4537, 642.174)	182.1151
(508.5557, 452.8331)	(401.3012, 596.6489)	179.406
(458.5146, 408.0112)	(410.997, 530.0996)	131.0096
(396.545, 401.7993)	(417.3822, 466.6885)	68.1527
(367.546, 450.173)	(414.1116, 451.7893)	46.5936
(306.4269, 469.9239)	(394.218, 407.896)	107.4929
(265.4733, 522.6037)	(353.6635, 360.8814)	184.2053
(258.2364, 570.5148)	(305.4683, 369.9402)	206.0607
(270.4601, 625.157)	(276.7618, 360.389)	264.843
(303.0142, 674.0524)	(206.7626, 366.7805)	321.9944
(332.3208, 730.5974)	(162.0843, 418.454)	355.5474
(349.9989, 787.9683)	(149.7128, 466.8183)	378.4862
(350.4697, 853.5018)	(190.1928, 519.9508)	370.0607
(326.3688, 910.4365)	(239.9671, 556.1886)	364.6324
(326.3688, 910.4365)	(261.9142, 581.2821)	335.4058
(289.4221, 949.9146)	(268.0112, 598.7149)	351.8518
(244.422, 990.9785)	(322.2684, 641.6199)	357.9267

Estimated position	Real position	Error
(191.7701, 1035.7731)	(348.6747, 682.3145)	386.7196
(146.4825, 1080.3693)	(354.3506, 751.8803)	388.734
(100.3764, 1123.4435)	(333.5109, 802.5281)	396.659
(51.4471, 1158.2642)	(302.4459, 862.0988)	388.2195
(0, 1160.8118)	(270.173, 902.6204)	373.7061
(0, 1120.1118)	(239.8338, 954.4223)	291.5017
(0, 1057.3402)	(210.2632, 997.4818)	218.6175
(0, 1057.3402)	(204.7983, 1011.2277)	209.9255
(0, 999.5785)	(173.9576, 1032.5531)	177.0553
(4.5328, 932.3506)	(130.3485, 1034.1374)	161.8337
(23.1448, 876.1747)	(123.4422, 992.8113)	153.83
(78.1727, 839.9333)	(118.9164, 931.5097)	100.2312
(138.2295, 836.783)	(118.5795, 869.9494)	38.5504
(198.3885, 856.1204)	(111.5327, 809.242)	98.6991
(253.9632, 865.6082)	(120.6181, 748.2151)	177.6571
(302.5973, 872.9641)	(127.4093, 737.3238)	221.5606
(362.6975, 852.121)	(158.3778, 728.0504)	239.0398
(402.1779, 814.7837)	(211.5717, 729.3305)	208.885
(443.2072, 768.747)	(279.6551, 726.4591)	168.9306
(473.6094, 728.6209)	(345.436, 729.5175)	128.1765
(505.5864, 686.2891)	(356.4745, 715.754)	151.9952
(518, 625.2834)	(387.2897, 681.149)	142.1484
(506.114, 566.4235)	(403.3691, 632.3739)	122.0901
(506.114, 566.4235)	(410.1373, 567.0379)	95.9787
(454.7384, 525.7058)	(411.856, 550.1799)	49.3749
(407.4895, 503.0495)	(409.9775, 545.2583)	42.2821
(343.1267, 490.0714)	(412.269, 519.689)	75.2187
(280.7463, 514.1752)	(415.855, 448.6958)	150.1396
(242.5992, 543.8624)	(408.4791, 426.9969)	202.913
(212.6372, 599.6273)	(388.8598, 414.6474)	255.4837
(234.1715, 653.729)	(328.4029, 365.484)	303.2569
(261.359, 696.3437)	(298.6608, 363.0901)	335.3348
(291.0053, 735.5004)	(261.593, 368.1253)	368.5507

Estimated position	Real position	Error
(323.423, 777.3277)	(243.7388, 365.2956)	419.6666
(358.0055, 831.7189)	(197.359, 367.1581)	491.5527
(382.0115, 888.3481)	(150.7424, 425.0793)	517.787
(382.2793, 952.9504)	(161.4653, 477.8345)	523.9217
(357.4531, 1015.5667)	(199.9737, 514.6954)	525.0446
(336.951, 667.0224)	(242.3945, 546.6384)	153.0792
(300.0837, 710.9152)	(242.3945, 546.6384)	174.1118
(252.3472, 750.7429)	(292.7605, 603.0024)	153.1682
(207.7873, 791.5182)	(334.4669, 660.6149)	182.1632
(167.1905, 837.437)	(352.7614, 707.6386)	226.4601
(125.4954, 887.2367)	(355.1286, 761.7572)	261.6802
(84.2686, 935.7599)	(329.4741, 816.5679)	272.6398
(19.8049, 952.065)	(305.2394, 863.2034)	298.9469
(0, 914.5797)	(264.5051, 928.8397)	264.8892
(0, 855.3899)	(231.2443, 984.2525)	264.7253
(0, 792.5382)	(203.4638, 1017.353)	303.2148
(0, 727.3555)	(163.0446, 1045.5813)	357.5629
(2.7786, 663.8318)	(121.3477, 1024.1666)	379.3412
(30.8401, 618.9093)	(123.7302, 985.5289)	378.2043
(67.5966, 585.5491)	(121.5753, 957.5104)	375.8575
(117.0943, 567.1091)	(119.1967, 934.2506)	367.1476
(157.939, 603.3908)	(119.3967, 911.4983)	310.5089
(157.939, 603.3908)	(147.4607, 721.9005)	118.972