SIMULTANEOUS LOCALIZATION AND MAPPING FOR UNMANNED AERIAL VEHICLES

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Mehmet Kök

August 2008

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

Prof. Dr. Billur Barshan(Supervisor)

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

Prof. Dr. Hitay Özbay

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

Assist. Prof. Dr. Ruşen Öktem

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

SIMULTANEOUS LOCALIZATION AND MAPPING FOR UNMANNED AERIAL VEHICLES

Mehmet Kök
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Billur Barshan
August 2008

Most mobile robot applications require the robot to be able to localize itself in an unknown environment without prior information so that the robot can navigate and accomplish tasks. The robot must be able to build a map of the unknown environment while simultaneously localizing itself in this environment. The *Simultaneous Localization and Mapping (SLAM)* is the formulation of this problem which has drawn a considerable amount of interest in robotics research for the past two decades. This work focuses on the SLAM problem for single and multiple agents equipped with vision sensors. We develop a vision-based 2-D SLAM algorithm for single and multiple Unmanned Aerial Vehicles (UAV) flying at constant altitude. Using the features of images obtained from an on-board camera to identify different landmarks, we apply different approaches based on the Extended Kalman Filter (EKF), the Information Filter (IF) and the Particle Filter (PF) to the SLAM problem. We present some simulation results and provide a comparison between the different implementations. We find Particle Filter implementations to perform better in estimations when compared to EKF and IF, however EKF and IF present more consistent results.

Keywords: UAV, SLAM, Extended Kalman Filter, Information Filter, Particle Filter, FastSLAM, SIFT, multi-agent systems.

ÖZET

İNSANSIZ HAVA ARAÇLARI İÇİN EŞANLI KONUMLANDIRMA VE HARİTALAMA

Mehmet Kök
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Billur Barshan
Ağustos 2008

Gezgin robot uygulamalarının çoğu robotun bilinmeyen bir çevrede önceden bilgi sahibi olmadan kendini konumlandırarak gezinimi sağlamasını ve görevleri gerçekleştirmesini gerektirmektedir. Robot, bilinmeyen ortamın haritasını çıkarıp bu haritada kendisini konumlandırabilmelidir. Eşanlı Konumlandırma ve Haritalama (EKVH) olarak bilinen bu problem son yirmi yıl içerisinde robotbilim araştırmaları arasında oldukça yoğun ilgi görmüştür. Bu çalışma, görü algılayıcıları ile donatılmış tek ve birden fazla robot için EKVH problemi üzerinde durmaktadır. Sabit yükseklikte uçtuğu varsayılan tek ve birden fazla İnsansız Hava Aracı (İHA) için görü-tabanlı bir Eşanlı Konumlandırma ve Haritalama (EKVH) algoritması geliştirilmiştir. Farklı yer işaretleri elde etmek için araç üzerindeki kameradan elde edilen görüntülerin öznitelikleri kullanılarak, Genişletilmiş Kalman Süzgeci (GKS), Bilgi Süzgeci (BS) ve de Parçacık Süzgeci EKVH problemine uygulanmıştır. Bazı benzetim sonuçları sunularak bu yöntemler arasında bir karşılaştırma yapılmıştır. Parçacık Süzgecinin daha iyi kestirim başarımı olduğu ama GKS ve BS'nin daha tutarlı sonuçlar verdiği gösterilmiştir.

Anahtar Kelimeler: İHA, EKVH, Genişletilmiş Kalman Süzgeci, Bilgi Süzgeci, Parçacık Süzgeci, çok-etmenli sistemler.

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. Billur Barshan for her guidance, supervision, endless dedication and especially her patience throughout my graduate studies leading to this thesis.

I am grateful to Prof. Dr. Hitay Özbay and Assist. Prof. Dr. Uluç Saranlı for their help and support with my research and their invaluable advices. I also thank Assist. Prof. Dr. Ruşen Öktem for her revisions and suggestions for my thesis.

My special thanks to Zeynep Yücel for her friendship, support and patience, I am deeply indebted to her help. I will miss our time together where we discussed unimportant matters concerning the meaning of life, universe and everything. I also would like to thank all my friends especially Mehmet Rauf Çelik, Akın Avcı and İmran Akça for their friendship and support, I could not have done without them.

I would also like to thank my parents and also my brother for their endless love, support and encouragement throughout my life.

I am also appreciative of the financial support I received through a fellowship from TÜBİTAK, the Scientific and Technical Research Council of Turkey.

Contents

List of Figures

List of Tables

Dedicated to my mom, dad and my dear brother.

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAV) have become an important and interesting research area with an increasing number of military and civilian applications. With the evolving technology and increasing availability of low-cost components, these vehicles have been transforming from simple radio-controlled aircraft to partially or fully autonomous agents accompanied with new challenges.

UAVs are used in exploration, reconnaissance, surveillance, target identification and tracking missions in applications such as information retrieval, environment monitoring, search and rescue as well as research purposes [38]. The basic requirement in full autonomy in these types of applications is autonomous motion planning and control. The UAV should be able to map the surrounding environment and localize itself in this environment as well, using its on-board sensory equipment. This is the so-called Simultaneous Localization and Mapping (SLAM) problem which has been an important topic in robotics research for the past two decades [22].

In this work, we propose a vision-based SLAM algorithm to realize fully autonomous motion planning and control for an UAV. Using bird's-eye view images of the terrain, obtained via a camera on the UAV, we aim to identify

some landmarks, considered as distinctive features that can be taken as reference points in an area. These landmarks are later on used in SLAM to build up a map of the unknown region while simultaneously keeping track of the correct position of the aircraft with this generated map, which is essential for efficient motion planning and control. We implement and compare the results of three filters: the Extended Kalman Filter (EKF), the Information Filter (IF) and the Particle Filter (PF).

1.1 Literature Review

The probabilistic SLAM research started with the recognition of the importance of simultaneously estimating robot location and pose while at the same time maintaining a map as a fundamental problem in autonomous robotics. Earlier work by Smith *et al.* [59, 60] and Durrant-Whyte [21] provided a statistical basis for describing and representing geometric relationships and uncertainty between *features* or *landmarks* in robotics. Moutarlier and Chatila [50] presented experiments with a real robot as well as introducing two different ways of presenting relationships between landmarks, *relation representation* and *location representation.*

A key paper by Smith *et al.* [61] showed a mobile robot, navigating in an unknown environment while taking observations. These observations demonstrated correlations in the estimations of landmark positions because of the common error in the estimated vehicle location and pose. In [61], a solution based on EKF is presented. These ideas and the solution served as a foundation for most of the current SLAM solutions. Using its on-board sensors, the robot takes observations and identifies landmarks which are then incorporated into the map. A state vector composed of robot pose and landmark locations is maintained using EKF through the recursive structure of the algorithm by updating the states at

discrete movement steps. The importance of the correlations in the estimation and the convergence of the estimation problem is first demonstrated by Csorba with initial results [14, 15] by considering the SLAM problem as a whole. In [18], the proof of convergence of EKF is presented for the linear SLAM problem.

Along with EKF, there are a number of different solutions proposed for SLAM in different probabilistic frameworks. Among these, the Kalman Filter (KF) and its variants are still probably the most popular [22]. However, there are some inherent problems in the EKF approach, some of which are addressed in [18] as well. One problem is the computational complexity of the algorithm which poses a significant problem for large scale areas as the map size increases. There have been a number of different approaches addressing this issue. Among these are partitioned updates [32, 39, 64, 74], global [31, 40] and relative [2, 25, 74] submaps, sparsification using IF [26, 27, 68, 69, 72, 73] and mapping relative quantities [14, 34, 45, 56, 73]. Another problem is the inconsistency introduced due to the linearization process used in the EKF. One solution proposed is unscented filtering [35] where the nonlinear function is approximated as a probability distribution using chosen sigma-points instead of estimating the nonlinear function or transformation by taking the first few terms of the Taylor series expansion. In [5], the inconsistency problem of the EKF-SLAM algorithm is addressed and the ineffectiveness of the previously proposed solutions is mentioned. However they point out that consistency is possible if a small heading variance can be maintained.

Particle Filtering (PF), also known as sequential Monte Carlo Methods, is another popular Bayesian estimation technique, based on estimating a probability distribution using a set of random samples called particles which are propagated over time with *importance sampling* and *resampling* mechanisms. Using particle filters removes the necessity of Gaussian noise models, enabling the use of any probability distribution and makes it possible to directly represent nonlinear

models. Although PF, in its simple form, is not suitable for the SLAM problem due to the high dimensionality of the state-space, in the FastSLAM [70] algorithm, through an approach called Rao-Blackwellization [51] that marginalizes parts of the state space, the algorithm becomes feasible. PF suffers from inconsistency in the long run, because of inherent problems such as particle depletion. However, it produces consistent estimation in the short-term when used with sufficient number of particles, which makes it an attractive estimator for short-term practical uses. Expectation-Maximization (EM) [10, 65, 66] can be considered as another popular approach to SLAM.

These approaches are mostly used in mobile ground vehicles equipped with different types of range sensors such as laser [70, 73], sonar [10], radar [18] by combining the range and bearing data with the information from odometry, GPS or inertial sensors. Vision-based SLAM algorithms have been becoming more popular as the cost- and energy-efficient cameras which also supply richer information become widely available. In vision-based approaches, techniques such as Harris corner detector [33] or Scale Invariant Feature Transform (SIFT) [43] are used to obtain a number of distinctive points on the images obtained by a camera, which later on are used in a SLAM algorithm based on KF or PF approaches [16, 24, 58].

SLAM problem in aerial vehicles is a new topic introduced with the recent popularity of UAV research and presents further challenges because of the motion dynamics and the effect of external factors such as wind, rain, air pockets, drift, etc. There are very few published works on this topic. Kim and Sukkarieh performed an EKF-based SLAM implementation on an UAV using an on-board camera [38]. However, in this work, white markers are manually placed on the terrain as artificial landmarks and the known size of these landmarks is used in order to estimate range along with the bearing to landmarks obtained from the images. This is supplemented by information provided by an on-board INS.

Throughout the ANSER project, other UAV SLAM experiments have been carried out as well, with emphasis on IF [63, 69]. Angeli *et al.* proposed a purely vision-based SLAM algorithm for Micro Air Vehicles [1]. In this work, images from the camera are used in *visual odometry*, along with identifying the landmarks defined as points obtained using Kanade-Lucas-Tomasi (KLT) tracking used with SIFT descriptors.

Multi-agent SLAM problem is a rather less-well-studied problem in the literature. The idea is to be able to complete the exploration and mapping task much faster using multiple robots or agents, and also to increase the robustness of the overall system to failures that can occur in a single robot. For this purpose, the well-known SLAM approaches should be extended to multiple robot cases and the incorporation of multiple estimations should be solved. A decentralized version of KF is used for data fusion in multiple sensor processing nodes [9, 57]. On the other hand, IF provides a better decentralized version of the Kalman Filter since IF has less computational complexity in the fusion of different estimations obtained by different sources. Information Filter is a form of Kalman Filter which uses the information matrix, the inverse of the covariance matrix in KF. IF has been used in multi-sensor data fusion systems with its suitability to decentralized applications [30, 44, 71]. IF is used in SLAM studies with the idea of exploiting the sparseness in the information matrix to decrease the computational complexity inherent in the SLAM problem. For this purpose, Sparse Extended Information Filter (SEIF) is introduced in [69] where a sparse information matrix is maintained using a 'sparsifying' process to achieve computational efficiency. The approach is applied to a real moving vehicle and also multi-vehicle (multi-UAV) simulations are presented. With the existence of multiple agents in a limited communication environment, the information passed along the nodes also becomes an important issue [30].

1.2 Organization

In this thesis, we propose a novel vision-based SLAM solution for an UAV, using the control inputs to the vehicle and natural visual features of terrain images, i.e. without artificial landmarks, which provides fully autonomous behavior. The organization of this thesis is as follows: Chapter 2 gives a description of our system and we describe how we identify landmarks from images. Chapters 3, 4, 5 review the EKF, Extended Information Filter (EIF) and PF respectively. Simulation results are given in Chapter 6. We conclude this thesis with a brief summary of the work presented and provide some future research directions.

Chapter 2

System Description

2.1 The UAV

In this work, we have assumed an UAV equipped with a downward facing camera flying at a constant altitude over a terrain. The bird's-eye view images that are supposedly taken by the downward facing camera mounted on the UAV are represented by 80×80-pixel images captured from *GoogleEarth* [29] software that can be considered as images representing the field of view of the camera. For convenience, we work on an image of size 1280×719 pixels obtained from GoogleEarth software taken at a given altitude of 6000 m above Bilkent University, Ankara, Turkey (Figure 2.2), and instead of capturing 80×80-pixel images from GoogleEarth each time, we take 80×80-pixel sub-images of this large image file. Throughout this work, position of the UAV and the landmarks, and the distances are given in pixel units and the location of the aircraft in a camera image is assumed to be the center of the image.

The aircraft model used for this simulation has three degrees-of-freedom (DOF) which are the x-y coordinates and the bearing θ_r of the aircraft around the

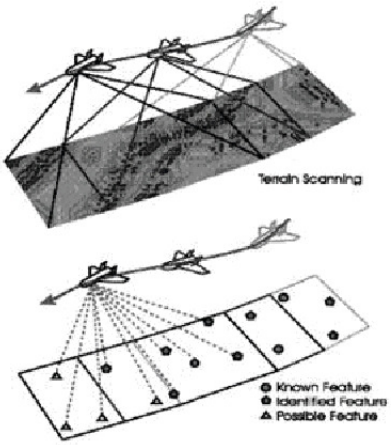


Figure 2.1: Motion of a UAV [37].

Figure 2.2: Virtual area covered by the UAV and zoomed part of the image is the camera view.

Figure 2.3: Three degrees of freedom for the process model [37].

z-axis, since the aircraft is assumed to fly at constant altitude without rotating about the x and y axes. Figure 2.3 displays the three DOF of the aircraft.

With this model and landmark locations, the state vector of the Kalman Filter consists of the position of the UAV and the position of the landmarks. The position of the UAV is represented by x_r and y_r, and θ_r is the orientation of the aircraft on the $x-y$ plane. Each landmark is represented with two states, which are its x and y positions on the terrain recorded in $\mathbf{X}_{map}$. Assuming that the total number of landmarks at the k'th step is n_k, this can be summarized as follows:

$$\begin{aligned} \mathbf{x}_{UAV}(k) &= [x_r(k), y_r(k), \theta_r(k)]^T, \\ \mathbf{x}_{map}(k) &= [x_1(k), y_1(k), x_2(k), y_2(k), \ldots, x_{n_k}(k), y_{n_k}(k)]^T, \\ \mathbf{x}(k) &= \begin{bmatrix} \mathbf{x}_{UAV} \\ \mathbf{x}_{map} \end{bmatrix}. \end{aligned} \tag{2.1}$$

The motion of the aircraft is modeled as:

$$\begin{aligned} x_r(k) &= x_r(k-1) + u_x(k) + q_x(k), \\ y_r(k) &= y_r(k-1) + u_y(k) + q_y(k), \\ \theta_r(k) &= \theta_r(k-1) + u_\theta(k) + q_\theta(k). \end{aligned} \tag{2.2}$$

Here $u_x(k)$, $u_y(k)$, $u_\theta(k)$ are velocities for each DOF given as control input and $q_x(k)$, $q_y(k)$, $q_\theta(k)$ are the zero mean, time-independent Gaussian noise values associated with each DOF.

2.2 Extracting Landmarks from Image Data

In order for SLAM to work, we have to be able to maintain some form of a map, and update it with each step so that we have some form of a reference that the aircraft can localize itself with respect to. In our case, the map consists of a number of reference points, namely the landmarks that can be identified easily from the image data and observed consistently. There are several aspects concerning the landmarks that have to be taken into consideration: Sufficient number of landmarks should be detected and the descriptors of these landmarks should be distinctive enough to decrease the mismatch probability. Excessive number of landmarks should be avoided because the runtime of SLAM would be affected drastically by this number due to the increasing dimensionality of the state vector $\mathbf{x}(k)$. Finally, landmark detection should be fast enough to be able to match the speed of the UAV.

To identify landmarks from the image data, we use image feature extraction methods. Feature extraction is the process of finding a smaller set of data in order to reduce the amount of resources to describe a large set of data accurately. In computer vision, feature extraction methods are used to classify and match images mostly for object recognition purposes. There are a number of methods in computer vision literature developed to extract visual features from an image. Edge detection [11], corner detection [33], blob detection [12], ridge detection [41] and other interest point detectors such as scale-invariant feature

transform (SIFT) [43], speeded up robust features (SURF) [8] and smallest univalue segment assimilating nucleus (SUSAN) [62] are some of the low-level feature extraction methods.

Although SIFT is more popular in vision-based SLAM research [17, 24, 55, 58], other visual features such as Harris Corner Detector [16], KLT [1], edge landmarks [23] are also used in SLAM solutions. Different feature extraction algorithms are compared in [6, 46, 75] which demonstrate the performances of different feature extracting methods in various scenarios. Most of the methods compared are convenient with tuned free parameters, such as matching thresholds.

2.2.1 Scale-Invariant Feature Transform (SIFT)

Among the methods mentioned above, SIFT is found to be more convenient for our work which we found to be more accurate and robust in this type of application. SIFT, devised by Lowe, is a computer vision algorithm for extracting distinctive features from images, to be used in tasks such as matching different views of an object or scene (e.g. for stereo vision) and object recognition. An example image with its SIFT features is given in Figure 2.4. This is also supported by the fact that SIFT is developed for extracting distinctive features from images and features are invariant to image scale, rotation, translation, partially invariant and robust to changing viewpoints and change in illumination, and resistant to partial occlusions. These characteristics make them suitable as landmarks for SLAM algorithms since the landmarks will be observed from different viewpoints with different illumination properties as the UAV moves. Using SIFT to solve data association and loop closing problems has been a popular approach [28, 54].

Outdoor images produce a high number of features (between 5,000−10,000 features) for an image with average resolution of 1280×720 and each feature is

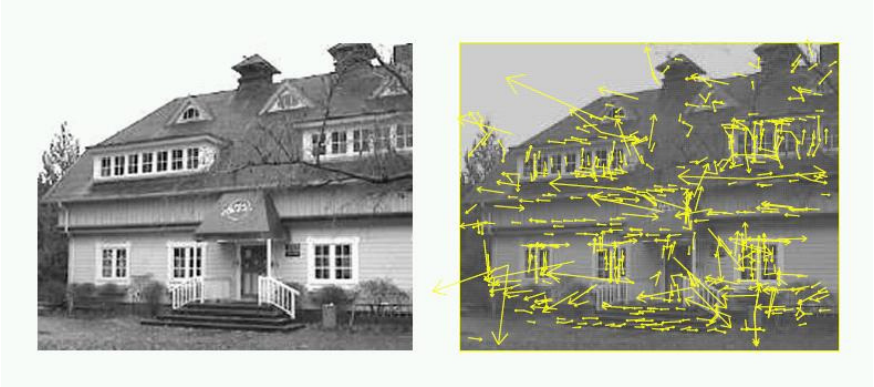


Figure 2.4: SIFT features on a sample image represented with their scales, orientations and locations [42].

represented by a descriptor vector consisting of 128 elements. These 128 elements consist of magnitudes of 4×4 array of histograms with 8 orientation bins obtained from the gradient magnitude and orientation calculated in a 16×16 region around each keypoint. Each point is also assigned a scale and an orientation. Considering that matching two features is done by calculating Euclidean distances of these vectors in a nearest-neighborhood algorithm, using this many features is unnecessary as well as being costly for a SLAM algorithm. Therefore, we tried to reduce the number of feature points.

Along with these considerations, another important issue in using SIFT features is the mismatches. Although SIFT features are highly descriptive, mismatches can still occur which may prevent the Kalman filter in SLAM algorithm from error pruning. Although mismatches can be avoided by modifying certain thresholds, this constitutes a trade-off between matching sufficient number of landmarks and mismatching many landmarks. The small number of mismatches that occur are handled by filtering out observed landmarks that are matched to landmarks that are improbable to be observed by the vehicle at its given pose.

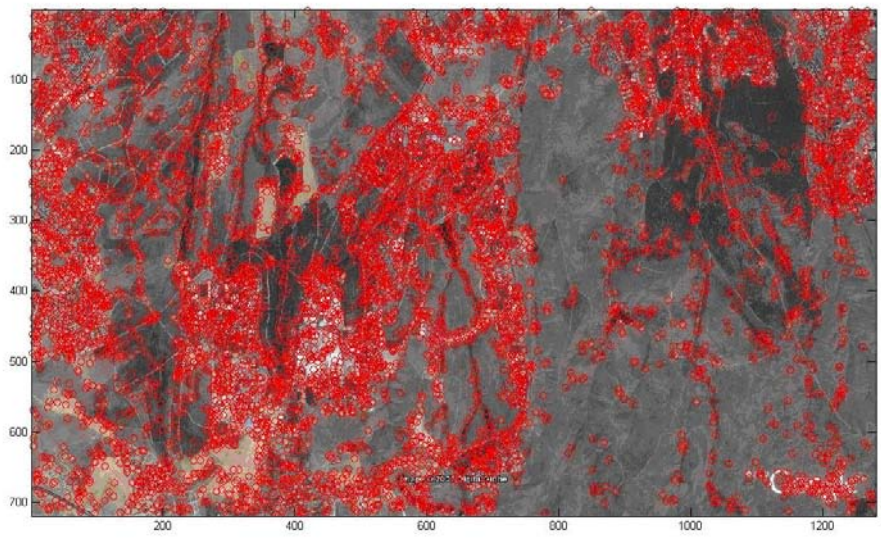


Figure 2.5: Location of all the features obtained with SIFT.

2.2.2 Reducing SIFT results

For the 1280×719 image shown in Figure 2.2, including terrain, buildings and roads, over 9,000 features were extracted using the SIFT algorithm (Figure 2.5). This many features is not necessary and not suitable for use in a Kalman filter, and processing this many features is also a problem. In Figure 2.6, we illustrate a reduced set of features that can be used for off-line processing. In a real life scenario images obtained via camera can be down-sampled to a lower resolution, which enables faster processing with SIFT algorithm, producing fewer features while still identifying landmarks sufficient enough for SLAM. Therefore, in this work, we performed on-line processing where 80×80 images were captured at a time, representing the field of view of the on-board camera.

Along with on-line processing of a smaller sized image to obtain fewer features, we also apply a selection process to further decrease the number of landmarks used. In our experiments, we observed that features with higher scale magnitude appear to be more robust and mismatches among these appear rarely. Therefore, selecting a number of features with higher descriptor scales proved to be a

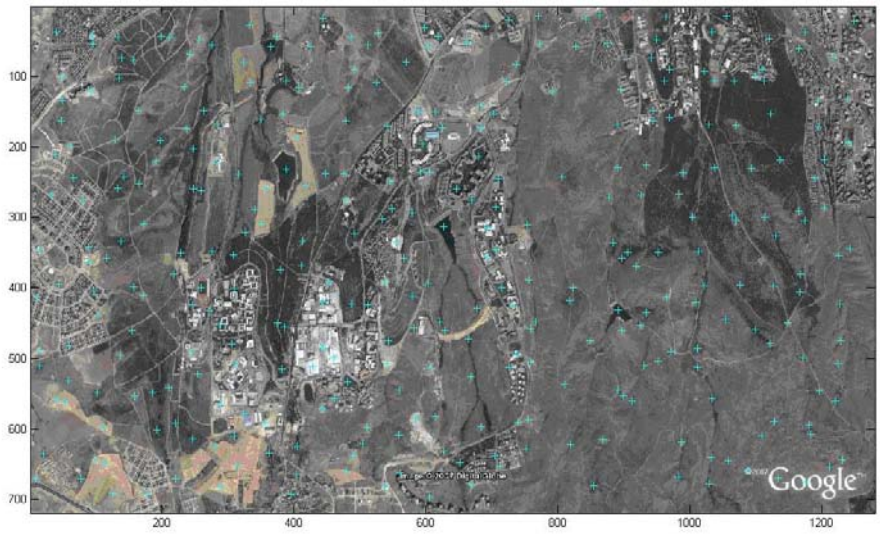


Figure 2.6: Result of the reduction of features for off-line processing.

convenient approach. We also track the features in multiple consecutive images and select the ones that can be consistently observed. About 30 SIFT features were selected per 80×80 image where the density of features is similar to that of Figure 2.6.

Chapter 3

Extended Kalman Filter

3.1 Kalman Filter Primer

The Kalman Filter (KF) has been one of the most commonly used state estimators since its introduction in the seminal paper by Kalman [36]. KF offers an estimator for a linear system modeled in state space form as:

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{G}(k)\mathbf{w}(k),$$
$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k).$$

where $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are zero-mean Gaussian noise sequences distributed according to:

$$\mathbf{w}(k) \sim \mathcal{N}(0, \mathbf{Q}(k)),$$
$$\mathbf{v}(k) \sim \mathcal{N}(0, \mathbf{R}(k)).$$

Using the notation in [7], the estimate of the state $\mathbf{x}(k)$ at time i, given the information up to and including time j, is given by:

$$\hat{\mathbf{x}}(i|j) = E[\mathbf{x}(i)|\mathbf{z}(1), ..., \mathbf{z}(j)],$$

with corresponding covariance

$$\mathbf{P}(i|j) = E[(\mathbf{x}(i) - \hat{\mathbf{x}}(i|j))(\mathbf{x}(i) - \hat{\mathbf{x}}(i|j))^T|\mathbf{z}(1), ..., \mathbf{z}(j)].$$

The corresponding KF is given as:

Time Update (Prediction):

$$\begin{aligned} \hat{\mathbf{x}}(k|k-1) &= \mathbf{F}(k)\hat{\mathbf{x}}(k-1|k-1), \\ \mathbf{P}(k|k-1) &= \mathbf{F}(k)\mathbf{P}(k-1|k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k), \\ \hat{\mathbf{x}}(0|-1) &= \mathbf{x}_0, \quad \mathbf{P}(0|-1) = \mathbf{P}_0 \end{aligned} \tag{3.1}$$

Measurement Update (Observation):

$$\begin{aligned} \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{W}(k)\boldsymbol{\nu}(k), \\ \mathbf{P}(k|k) &= \mathbf{P}(k|k-1)\mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k), \\ \text{where } \boldsymbol{\nu}(k) &= \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1), \\ \mathbf{S}(k) &= \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k), \\ \mathbf{W}(k) &= \mathbf{P}(k|k-1)\mathbf{H}^T(k)\mathbf{S}^{-1}(k), \end{aligned} \tag{3.2}$$

$\boldsymbol{\nu}(k)$ is the innovation or measurement residual sequence, which is the difference between the measurement and predicted measurements, representing the new information. $\mathbf{S}(k)$ is the innovation covariance matrix representing the uncertainty in the measurement. $\mathbf{W}(k)$ is the Kalman gain matrix that adjusts the prediction given the measurement.

3.2 Extended Kalman Filter and SLAM

3.2.1 State and Observation Models

The state model given in Equation (2.2) can be represented with state equations, including the landmark states, as follows:

$$\mathbf{x}(k) = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1(k) \\ y_1(k) \\ x_2(k) \\ y_2(k) \\ \vdots \\ x_{n_k}(k) \\ y_{n_k}(k) \end{bmatrix} = \begin{bmatrix} x_r(k-1) + u_x(k) \\ y_r(k-1) + u_y(k) \\ \theta_r(k-1) + u_\theta(k) \\ x_1(k-1) \\ y_1(k-1) \\ x_2(k-1) \\ y_2(k-1) \\ \vdots \\ x_{n_k}(k-1) \\ y_{n_k}(k-1) \end{bmatrix} + \mathbf{w}(k) \quad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \tag{3.3}$$
$$= \mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k)) + \mathbf{w}(k),$$

Here, $\mathbf{x}(k)$ is the state vector and $\mathbf{u}(k)$ is the control input given to the system at the k'th step with $u_x(k) = \cos(\theta_r(k-1))u_1(k)$, $u_y(k) = \sin(\theta_r(k-1))u_1(k)$ and $u_\theta(k)$ is simply $u_2(k)$ (Figure 3.1). $\mathbf{w}(k)$ is the temporally uncorrelated zero-mean Gaussian process noise with covariance matrix $\mathbf{Q}(k)$ so that $\mathbf{w}(k) \sim \mathcal{N}(0, \mathbf{Q}(k))$ and $E[\mathbf{w}(i)\mathbf{w}(j)] = \delta_{ij}\mathbf{Q}(i)$. $\mathbf{x}(k)$ consists of UAV $x-y$ position and orientation representing the UAV states, and landmark positions representing the map states. Since this is a SLAM application, the state vector keeps growing as new landmarks are detected and are added as states. The positions of landmarks do not change with the motion and process noise is assumed to be zero for the landmark states.

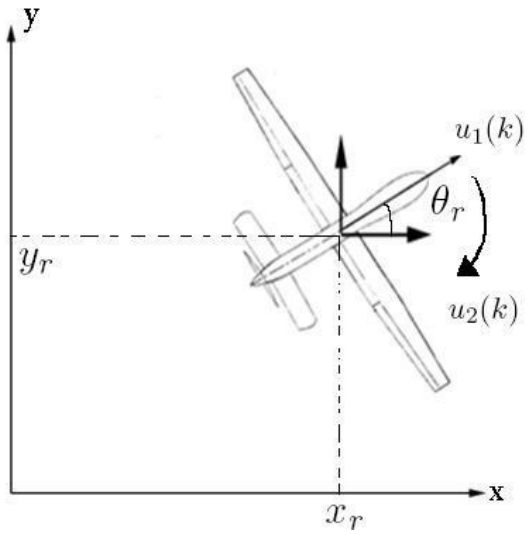


Figure 3.1: UAV states and control input.

The observation model can be stated with $\mathbf{h}_i(k)$ given as a function that returns the range and the bearing of the i^{th} landmark with respect to the vehicle at the k^{th} step and $\mathbf{v}(k)$ here is the temporally uncorrelated zero-mean Gaussian observation noise with covariance $\mathbf{R}(k)$. We assume that the process noise $\mathbf{w}(i)$ and the observation noise $\mathbf{v}(i)$ are uncorrelated:

$$\mathbf{z}(k) = \mathbf{h}(\mathbf{x}(k)) + \mathbf{v}(k),$$

$$\text{where } \mathbf{h}(k) = \begin{bmatrix} \mathbf{h}_1(x_1, y_1) \\ \mathbf{h}_2(x_2, y_2) \\ \vdots \\ \mathbf{h}_{n_k}(x_{n_k}, y_{n_k}) \end{bmatrix}, \quad \mathbf{h}_i(x_i, y_i) = \begin{bmatrix} h_{i_1} \\ h_{i_2} \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta {x_i}^2 + \Delta {y_i}^2} \\ \arctan2\left(-\Delta y_i, -\Delta x_i\right) - \theta_r(k) \end{bmatrix},$$

$$\text{and } \Delta x_i = x_r(k) - x_i(k),\ \Delta y_i = y_r(k) - y_i(k),$$

$$\text{with } \mathbf{v}(k) = \begin{bmatrix} v_1(k) \\ v_2(k) \\ \vdots \\ v_{n_k}(k) \end{bmatrix}, \quad \mathbf{v}(k) \sim \mathcal{N}(0, \mathbf{R}(k)), \quad E[\mathbf{v}(i)\mathbf{v}(j)] = \delta_{ij}\mathbf{R}(i),$$

$$E[\mathbf{w}(i)\mathbf{v}^T(j)] = 0.$$

The first element of $\mathbf{h}_i$ is the distance of the i'th landmark to the UAV. The second element is the relative orientation of the landmark with respect to the vehicle, namely the bearing. One issue to note here is that not each landmark is observed at each state and the number of landmarks keep increasing. The matching of landmarks to the map (data association) is carried out by the SIFT matching algorithm.

Note here that both the motion model and observation model are non-linear, therefore linearized versions of KF and IF will be used, EKF and EIF respectively.

The state and observation equations of the system can be summarized as follows:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k)) + \mathbf{w}(k), \\ \mathbf{z}(k) &= \mathbf{h}(\mathbf{x}(k)) + \mathbf{v}(k). \end{aligned} \tag{3.4}$$

3.2.2 EKF Prediction Step

Since the state model is nonlinear, for EKF we take the $\mathbf{F}$ matrix as the gradient of function $\mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k))$ given in Equation (3.4). $\mathbf{F}$ turns out to be as follows:

$$\begin{aligned} \mathbf{F}(k) = \nabla \mathbf{f} &= \frac{\partial \mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k))}{\partial \mathbf{x}(k)} \\ &= \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial x_r} & \frac{\partial \mathbf{f}_1}{\partial y_r} & \frac{\partial \mathbf{f}_1}{\partial \theta_r} & \frac{\partial \mathbf{f}_1}{\partial x_1} & \frac{\partial \mathbf{f}_1}{\partial y_1} & \cdots & \frac{\partial \mathbf{f}_1}{\partial x_{n_k}} & \frac{\partial \mathbf{f}_1}{\partial y_{n_k}} \\ \frac{\partial \mathbf{f}_2}{\partial x_r} & \frac{\partial \mathbf{f}_2}{\partial y_r} & \frac{\partial \mathbf{f}_2}{\partial \theta_r} & \frac{\partial \mathbf{f}_2}{\partial x_1} & \frac{\partial \mathbf{f}_2}{\partial y_1} & \cdots & \frac{\partial \mathbf{f}_2}{\partial x_{n_k}} & \frac{\partial \mathbf{f}_2}{\partial y_{n_k}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial \mathbf{f}_{2n_k+3}}{\partial x_r} & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial y_r} & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial \theta_r} & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial x_1} & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial y_1} & \cdots & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial x_{n_k}} & \frac{\partial \mathbf{f}_{2n_k+3}}{\partial y_{n_k}} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\sin\theta_r(k-1)u_1(k) & \mathbf{0} \\ 0 & 1 & \cos\theta_r(k-1)u_1(k) & \mathbf{0} \\ 0 & 0 & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{2n_k \times 2n_k} \end{bmatrix}. \end{aligned}$$

Here, it should be pointed out that the map part of the state vector consists of $2n_k$ elements for n_k landmarks since each landmark is represented by its x and y coordinates in the state vector. With this $\mathbf{F}$, the prediction step can be expressed as follows:

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{f}(\hat{\mathbf{x}}(k-1), \mathbf{u}(k)),$$
$$\mathbf{P}(k|k-1) = \mathbf{F}(k)\mathbf{P}(k-1|k-1)\mathbf{F}^T(k) + \mathbf{Q}(k).$$

As mentioned before, the process noise is zero for map states which are landmark positions, therefore, covariance matrix $\mathbf{Q}(k)$ is zero for the map states.

3.2.3 EKF Update Step

Similarly, because of the nonlinearity of the observation model, we use the gradient of the function $\mathbf{h}(\mathbf{x}(k))$, $\mathbf{H}$ in our update equations. Using Equation (2.1), $\mathbf{H}$ can be derived as:

$$\mathbf{H}(k) = \nabla\mathbf{h} = \frac{\partial \mathbf{h}(\mathbf{x}(k))}{\partial \mathbf{x}(k)} = \begin{bmatrix} \mathbf{H}_{UAV}(k) & \mathbf{H}_{map}(k) \end{bmatrix}$$

$$\text{where } \mathbf{H}_{UAV}(k) = \frac{\partial \mathbf{h}(\mathbf{x}(k))}{\partial \mathbf{x}_{UAV}(k)} = \begin{bmatrix} \frac{\Delta x_1}{p_1} & \frac{\Delta y_1}{p_1} & 0 \\ \frac{-\Delta y_1}{p_1^2} & \frac{\Delta x_1}{p_1^2} & -1 \\ \frac{\Delta x_2}{p_2} & \frac{\Delta y_2}{p_2} & 0 \\ \frac{-\Delta y_2}{p_2^2} & \frac{\Delta x_2}{p_2^2} & -1 \\ \vdots & \vdots & \vdots \\ \frac{\Delta x_{n_k}}{p_{n_k}} & \frac{\Delta y_{n_k}}{p_{n_k}} & 0 \\ \frac{-\Delta y_{n_k}}{p_{n_k}^2} & \frac{\Delta x_{n_k}}{p_{n_k}^2} & -1 \end{bmatrix}$$

$$\mathbf{H}_{MAP}(k) = \frac{\partial \mathbf{h}(\mathbf{x}(k))}{\partial \mathbf{x}_{MAP}(k)} = \begin{bmatrix} \frac{-\Delta x_1}{p_1} & \frac{-\Delta y_1}{p_1} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\Delta y_1}{p_1^2} & \frac{-\Delta x_1}{p_1^2} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{-\Delta x_2}{p_2} & \frac{-\Delta y_2}{p_2} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta y_2}{p_2^2} & \frac{-\Delta x_2}{p_2^2} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & & & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & & & & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{-\Delta x_{n_k}}{p_{n_k}} & \frac{-\Delta y_{n_k}}{p_{n_k}} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{\Delta y_{n_k}}{p_{n_k}^2} & \frac{-\Delta x_{n_k}}{p_{n_k}^2} \end{bmatrix},$$

where $p_i = \sqrt{\Delta x_i^2 + \Delta y_i^2}$.

Considering that we do not observe all landmarks at each iteration of the EKF, we do not have observation measurements for some of the states at a given iteration. Furthermore, we may have some previously unobserved states. In order to include only the observed states in our update equations, we process them one by one to obtain $\tilde{\mathbf{H}}(k)$ a sub-matrix of $\mathbf{H}$ and use this sub-matrix in the update equations. If we observe m_k landmarks at step k, $\tilde{\mathbf{H}}(k)$ is obtained as:

$$\tilde{\mathbf{H}}(k) = \begin{bmatrix} \mathbf{H}_1(k) \\ \mathbf{H}_2(k) \\ \vdots \\ \mathbf{H}_{m_k}(k) \end{bmatrix},$$

$$\mathbf{H}_i(k) = \nabla \mathbf{h}_i = \frac{\partial \mathbf{h}_i(x_i, y_i)}{\partial \mathbf{x}(k)},$$

$$= \begin{bmatrix} \frac{\partial \mathbf{h}_{i_1}(k)}{\partial x_r} & \frac{\partial \mathbf{h}_{i_1}(k)}{\partial y_r} & \frac{\partial \mathbf{h}_{i_1}(k)}{\partial \theta_r} & 0 & \dots & 0 & \frac{\partial \mathbf{h}_{i_1}(k)}{\partial x_i} & \frac{\partial \mathbf{h}_{i_1}(k)}{\partial y_i} & 0 & \dots & 0 \\ \frac{\partial \mathbf{h}_{i_2}(k)}{\partial x_r} & \frac{\partial \mathbf{h}_{i_2}(k)}{\partial y_r} & \frac{\partial \mathbf{h}_{i_2}(k)}{\partial \theta_r} & 0 & \dots & 0 & \frac{\partial \mathbf{h}_{i_2}(k)}{\partial x_i} & \frac{\partial \mathbf{h}_{i_2}(k)}{\partial y_i} & 0 & \dots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\Delta x_i}{p_i} & \frac{\Delta y_i}{p_i} & 0 & 0 & \dots & 0 & \frac{-\Delta x_i}{p_i} & \frac{-\Delta y_i}{p_i} & 0 & \dots & 0 \\ \frac{-\Delta y_i}{p_i^2} & \frac{\Delta x_i}{p_i^2} & -1 & 0 & \dots & 0 & \frac{\Delta y_i}{p_i^2} & \frac{-\Delta x_i}{p_i^2} & 0 & \dots & 0 \end{bmatrix}.$$

The update equations for all the states can be listed as follows:

$$
\begin{aligned}
\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{W}(k)\boldsymbol{\nu}(k), \\
\mathbf{P}(k|k) &= \mathbf{P}(k|k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k), \\
\text{where } \boldsymbol{\nu}(k) &= \mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}(k|k-1)), \\
\mathbf{S}(k) &= \tilde{\mathbf{H}}(k)\mathbf{P}(k|k-1)\tilde{\mathbf{H}}^T(k) + \mathbf{R}(k), \\
\mathbf{W}(k) &= \mathbf{P}(k|k-1)\tilde{\mathbf{H}}^T(k)\mathbf{S}^{-1}(k), \\
\mathbf{R}(k) &= \begin{bmatrix} \mathbf{R}_1(k) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_2(k) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_{m_k}(k) \end{bmatrix}.
\end{aligned}
$$

3.2.4 Landmark Augmentation

Since this is a SLAM application, at each iteration, that is at each motion step, we may observe landmarks that match landmarks in our map or observe new landmarks in which case we need to update our map, which is represented by the state vector. We have to update the state vector $\mathbf{x}(k)$ and the covariance matrix $\mathbf{P}(k)$ associated with it as well. This process is called *landmark augmentation.*

Here, we have the landmark initialization function $\mathbf{g}(\mathbf{x}(k), \mathbf{z}_i(k))$ as a function of range r_i and bearing θ_i of a landmark i with respect to the UAV position and

orientation.

$$\mathbf{g}(\mathbf{x}(k), \mathbf{z}_i(k)) = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} x_r(k) + r_i \cos(\theta_i + \theta_r(k)) \\ y_r(k) + r_i \sin(\theta_i + \theta_r(k)) \end{bmatrix},$$

$$\mathbf{x}^*(k|k) = \begin{bmatrix} \hat{\mathbf{x}}(k|k) \\ \mathbf{g}(\hat{\mathbf{x}}(k|k), \mathbf{z}_i(k)) \end{bmatrix},$$

$$\mathbf{P}^*(k|k) = \nabla\mathbf{Y}_{x,z}\mathbf{P}(k|k)\nabla\mathbf{Y}_{x,z}^T,$$

$$\nabla\mathbf{Y}_{x,z} = \begin{bmatrix} \mathbf{I}_{2n_k+3\times 2n_k+3} & \mathbf{0}_{2n_k+3\times 2} \\ \nabla\mathbf{G}_x & \nabla\mathbf{G}_z \end{bmatrix}.$$

where $\nabla\mathbf{G}_x$ and $\nabla\mathbf{G}_z$ are gradients of $\mathbf{g}(\mathbf{x}(k), \mathbf{z}_i(k))$ with respect to $\mathbf{x}$ and $\mathbf{z}$, respectively, given as follows:

$$\nabla\mathbf{G}_x = \begin{bmatrix} \frac{\partial g_1}{\partial x_r} & \frac{\partial g_1}{\partial y_r} & \frac{\partial g_1}{\partial \theta_r} & \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial y_1} & \dots & \frac{\partial g_1}{\partial x_{n_k}} & \frac{\partial g_1}{\partial y_{n_k}} \\ \frac{\partial g_2}{\partial x_r} & \frac{\partial g_2}{\partial y_r} & \frac{\partial g_2}{\partial \theta_r} & \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial y_1} & \dots & \frac{\partial g_2}{\partial x_{n_k}} & \frac{\partial g_2}{\partial y_{n_k}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r_i \sin(\theta_i + \theta_r(k)) & 0 & \dots & 0 \\ 0 & 1 & r_i \cos(\theta_i + \theta_r(k)) & 0 & \dots & 0 \end{bmatrix}$$

$$\nabla\mathbf{G}_z = \begin{bmatrix} \frac{\partial g_1}{\partial r_i} & \frac{\partial g_1}{\partial \theta_i} \\ \frac{\partial g_2}{\partial r_i} & \frac{\partial g_2}{\partial \theta_i} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i + \theta_r(k)) & -r_i \sin(\theta_i + \theta_r(k)) \\ \sin(\theta_i + \theta_r(k)) & r_i \cos(\theta_i + \theta_r(k)) \end{bmatrix}.$$

After obtaining the augmented state vector $\mathbf{x}^*(k|k)$ and augmented covariance matrix $\mathbf{P}^*(k|k)$ we assign these as:

$$\mathbf{x}(k) = \mathbf{x}^*(k|k),$$
$$\mathbf{P}(k) = \mathbf{P}^*(k|k).$$

to complete the k'th step and move on to the next step.

Chapter 4

Extended Information Filter

4.1 The Information Filter

The Information Filter (IF) is a different form of the Kalman Filter represented in terms of measures of *information* about the states rather than using direct state estimates and the covariances associated with them. For a system with Gaussian noise, the inverse of the covariance matrix, Fisher information matrix, provides the measure of information about the state present in the observations [53].

In KF, the Gaussian random vector $\mathbf{x}(k)$ is parametrized by its first and second-order moments, the mean vector $\boldsymbol{\mu}(k)$ and the covariance matrix $\boldsymbol{\Sigma}(k)$ which is called the *covariance form* of the filter. IF makes use of the *canonical parametrization* also known as *canonical form* or *natural form* [26] which relates

to the covariance form as follows:

$$\begin{aligned} p(\mathbf{x}(k)) &= \mathcal{N}(\boldsymbol{\mu}(k), \boldsymbol{\Sigma}(k)), \\ &\propto e^{-\frac{1}{2}((\mathbf{x}(k)-\boldsymbol{\mu}(k))^T \boldsymbol{\Sigma}^{-1}(k)(\mathbf{x}(k)-\boldsymbol{\mu}(k)))}, \\ &= e^{-\frac{1}{2}\mathbf{x}^T(k)\boldsymbol{\Sigma}^{-1}(k)\mathbf{x}(k)-2\boldsymbol{\mu}^T(k)\boldsymbol{\Sigma}^{-1}(k)\mathbf{x}(k)+\boldsymbol{\mu}^T(k)\boldsymbol{\Sigma}^{-1}(k)\boldsymbol{\mu}(k)}, \\ &\propto e^{-\frac{1}{2}(\mathbf{x}^T(k)\boldsymbol{\Sigma}^{-1}(k)\mathbf{x}(k)-2\boldsymbol{\mu}^T(k)\boldsymbol{\Sigma}^{-1}(k)\mathbf{x}(k))}, \\ &= e^{-\frac{1}{2}\mathbf{x}^T(k)\mathbf{Y}(k)\mathbf{x}(k)+\mathbf{y}^T(k)\mathbf{x}(k)}, \\ &\propto \mathcal{N}^{-1}(\mathbf{y}(k), \mathbf{Y}(j)). \end{aligned}$$

where the proportionality sign accounts for the normalization constant. When $\boldsymbol{\mu}(k)$ is $\mathbf{x}(k)$ and covariance matrix $\boldsymbol{\Sigma}(k)$ is $\mathbf{P}(k)$, the information state vector is defined as

$$\mathbf{y}(k) \triangleq \mathbf{P}^{-1}(k)\mathbf{x}(k) \tag{4.1}$$

and the information matrix is

$$\mathbf{Y}(k) \triangleq \mathbf{P}^{-1}(k). \tag{4.2}$$

Substituting (4.1) and (4.2) in the Kalman Filter equations (3.1) and (3.2), the Information Filter equations can be obtained which can be summarized as follows:

Time Update:

$$\hat{\mathbf{y}}(k|k-1) = \mathbf{L}(k|k-1)\hat{\mathbf{y}}(k|k-1), \tag{4.3a}$$

$$\mathbf{Y}(k|k-1) = [\mathbf{F}(k)\mathbf{Y}^{-1}(k-1|k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k)]^{-1}. \tag{4.3b}$$

Measurement Update:

$$\hat{\mathbf{y}}(k|k) = \hat{\mathbf{y}}(k|k-1) + \mathbf{i}(k), \tag{4.4a}$$

$$\mathbf{Y}(k|k) = \mathbf{Y}(k|k-1) + \mathbf{I}(k). \tag{4.4b}$$

Defining

$$\mathbf{i}(k) \triangleq \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) \tag{4.5}$$

as the information state contribution from an observation $\mathbf{z}(k)$,

$$\mathbf{I}(k) \triangleq \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \tag{4.6}$$

as its associated information matrix, and

$$\mathbf{L}(k|k-1) = \mathbf{P}^{-1}(k|k-1)\mathbf{F}(k)\mathbf{P}^{-1}(k-1|k-1).$$

as a propagation coefficient.

This is the information form of the Kalman Filter. The IF is functionally equivalent to the KF, but has some attractive properties. One characteristic that is observed is that the estimation equations (4.4a) and (4.4b) are simpler than that of the KF, at the expense of increased complexity in the prediction step of IF. This enables partitioning of the estimation equations which are simpler in information form for a decentralized scenario. This property will be exploited in this work. Another property is that information matrix used in IF becomes almost sparse when normalized, which provides a way of decreasing the computational costs of the filter by imposing sparseness. This property has been

studied by Thrun *et al.* [69] and Eustice *et al.* [26, 72] with good results. One other property is that the information form enables easier initialization of the state vector and the information matrix [47].

4.2 Extended Information Filter

Similar to the Extended Kalman Filter in Chapter 3, because of the nonlinearity of our system model, we need a linearization process which results in the Extended Information Filter (EIF). Substituting the linearized versions of the process and observation model into Equations (4.3b), (4.5) and (4.6), time update and measurement update steps of the EIF can be obtained as follows:
Time update:

$$\begin{aligned}
\mathbf{x}(k-1) &= \mathbf{Y}^{-1}(k-1|k-1)\mathbf{y}(k-1|k-1), \\
\mathbf{y}(k|k-1) &= \mathbf{Y}(k|k-1)\mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k)), \\
\mathbf{Y}(k|k-1) &= [\mathbf{F}(k)\mathbf{P}(k-1|k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k)]^{-1}.
\end{aligned}$$

Estimation:

$$\begin{aligned}
\hat{\mathbf{y}}(k|k) &= \hat{\mathbf{y}}(k|k-1) + \mathbf{i}(k), \\
\mathbf{Y}(k|k) &= \mathbf{Y}(k|k-1) + \mathbf{I}(k).
\end{aligned}$$

with information state contribution and associated information matrix modified as:

$$\begin{aligned}
\mathbf{i}(k) &\triangleq \mathbf{H}^T(k)\mathbf{R}^{-1}(k)(\boldsymbol{\nu}(k) + \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)), \\
\mathbf{I}(k) &\triangleq \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k),
\end{aligned}$$

and $\boldsymbol{\nu}(k)$ is again the innovation sequence given by

$$\boldsymbol{\nu}(k) = \mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}(k|k-1)).$$

One of the downfalls that appears with the IF is that the indirect propagation of state and covariance vectors becomes a problem when these are needed. Along with the need for state and covariance estimates for realistic use, in EIF, state estimate is also needed for the linearization of the prediction and measurement steps. This situation makes the sparsification methods more attractive [4].

4.3 Multi-UAV SLAM with EIF

As mentioned before, one attractive feature of the information form is its suitability for decentralized applications. The estimation step equations where observations are incorporated into the estimation can be separated into a multi-agent framework, providing the ability to process information from different sources.

4.3.1 Distributing the Information Filter

Assume an observation vector $\mathbf{z}(k)$ consisting of observations from N subvectors corresponding to observations made by each sensor:

$$\mathbf{z}(k) = \left[\mathbf{z}_1^T(k),\ \ldots,\ \mathbf{z}_N^T(k)\right]^T,$$

with observation matrix and observation noise also partitioned corresponding to these observations:

$$\mathbf{H}(k) = \left[\mathbf{H}_1^T(k),\ \ldots,\ \mathbf{H}_N^T(k)\right]^T,$$
$$\mathbf{v}(k) = \left[\mathbf{v}_1^T(k),\ \ldots,\ \mathbf{v}_N^T(k)\right]^T.$$

with the assumption that the observation noise are not correlated

$$E[\mathbf{v}(k)\mathbf{v}^T(k)] = \mathbf{R}(k) = blockdiag\{\mathbf{R}_1(k), \ldots, \mathbf{R}_N(k)\},$$

so that sensor model is partitioned into N equations of the form

$$\mathbf{z}_j(k) = \mathbf{H}_j(k)\mathbf{x}(k) + \mathbf{v}_j(k),$$

with

$$E[\mathbf{v}_p(i)\mathbf{v}_q^T(j)] = \sigma_{ij}\sigma_{pq}\mathbf{R}_p(i).$$

When we define

$$\mathbf{i}_j(k) \triangleq \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{z}_j(k),$$
$$\mathbf{I}_j(k) \triangleq \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k),$$

$\mathbf{i}_j(k)$ being the information state contribution from $\mathbf{z}_j(k)$ and $\mathbf{I}_j(k)$ being its associated information matrix, we obtain:

$$\mathbf{i}(k) = \sum_{j=1}^{N} \mathbf{i}_j(k) = \sum_{j=1}^{N} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{z}_j(k),$$
$$\mathbf{I}(k) = \sum_{j=1}^{N} \mathbf{I}_j(k) = \sum_{j=1}^{N} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k).$$

The resulting estimation equations are:

$$\hat{\mathbf{y}}(k|k) = \hat{\mathbf{y}}(k|k-1) + \sum_{j=1}^{N} \mathbf{i}_j(k),$$
$$\mathbf{Y}(k|k) = \mathbf{Y}(k|k-1) + \sum_{j=1}^{N} \mathbf{I}_j(k).$$

This enables the information from different sources to be incorporated easily with summation of their information state contributions and associated information matrices. This can be extended to EIF as well.

4.3.2 Information Between Agents

Considering a decentralized SLAM application, each agent has its own vehicle and map estimate, but does not need to know the vehicle estimate of the other agents. Each node receives only map information from other agents which can be easily incorporated into the local map estimate. We can define map information in covariance and information form as follows with m and v subscripts meaning map and vehicle respectively:

$$\begin{aligned}
\mathbf{x}(k) &= [\mathbf{x}_v(k)\ \mathbf{x}_m(k)]^T, \\
\mathbf{P}(k) &= \begin{bmatrix} \mathbf{P}_{vv}(k) & \mathbf{P}_{vm}(k) \\ \mathbf{P}_{vm}^T(k) & \mathbf{P}_{mm}(k) \end{bmatrix}, \\
\mathbf{y}_{mm}^*(k) &= \mathbf{P}_{mm}^{-1}(k)\mathbf{x}_m(k), \\
\mathbf{Y}_{mm}^*(k) &= \mathbf{P}_{mm}^{-1}(k).
\end{aligned}$$

When we separate the information form, similarly:

$$\begin{aligned}
\mathbf{y}(k) &= [\mathbf{y}_v(k)\ \mathbf{y}_m(k)]^T, \\
\mathbf{Y}(k) &= \begin{bmatrix} \mathbf{Y}_{vv}(k) & \mathbf{Y}_{vm}(k) \\ \mathbf{Y}_{vm}^T(k) & \mathbf{Y}_{mm}(k) \end{bmatrix}.
\end{aligned}$$

We note that

$$\begin{aligned}
\mathbf{y}_{mm}^*(k) &\neq \mathbf{y}_m(k), \\
\mathbf{Y}_{mm}^*(k) &\neq \mathbf{Y}_{mm}(k).
\end{aligned}$$

This situation is because the vehicle states have effect on the map states. To remove the vehicle information in order to leave map information only, we can

use [63]:

$$\mathbf{y}^*_{mm}(k) = \mathbf{y}_m(k) - \mathbf{Y}_{vm}(k)\mathbf{Y}^{-1}_{vv}(k)\mathbf{y}_v(k),$$
$$\mathbf{Y}^*_{mm}(k) = \mathbf{Y}_{mm}(k) - \mathbf{Y}_{vm}(k)\mathbf{Y}^{-1}_{vv}(k)\mathbf{Y}^T_{vm}(k).$$

With these equations, at each communication step, agents calculate the map information and transmit these, along with the landmark descriptors. At the receiving end, this information is incorporated to the information vector and matrix simply by addition after carrying out a data association step to match the information.

Chapter 5

Rao-Blackwellized Particle Filter: FastSlam

Previous SLAM efforts, based on KF and its variants, mostly focused on improving the performance of EKF-SLAM while keeping the Gaussian assumption on the process and measurement noise models. The FastSlam algorithm proposed by Thrun and Montemerlo [70] provides a fundamentally different solution based on Particle Filtering or recursive Monte Carlo Sampling to the SLAM problem.

5.1 Particle Filtering

The Kalman Filter and its variants represent the probability distributions using a parametrized model, a multivariate Gaussian, as shown in Chapter 3 and 4. Unlike KF, Particle Filters, also called *Sequential Monte Carlo Methods* [19], try to represent a probability distribution, which can be non-Gaussian in general, based on a finite set of weighted samples, called *particles*. The weights signify the reliability of the specific particle and estimate of the random variable is obtained with the weighted sum of all particles. The density of particles in a region

(a) Initially the robot pose is very uncertain.

(b) After a number of steps, particles start to converge.

(c) After many observations the posterior converges to a unimodal posterior.

Figure 5.1: Particle filter localization experiment by Wolfram *et al.* [13].

and their weights represent the probability of that region, high density with high weights meaning high probability. Given sufficient number of samples, this non-parametric representation can approximate an arbitrarily complex multi-modal probability distribution [48]. With infinite number of samples, the true distribution can be obtained exactly [19].

Particle Filters have been implemented for a number of real world applications especially for robot localization [67] and robot mapping [19, 51, 66]. An example of a Monte Carlo localization experiment can be seen in Figure 5.1. At the start, the robot has no prior information about its location in a given map, which is represented by scattering the particles uniformly, displaying the uncertainty shown in Figure 5.1(a). Figure 5.1(b) shows particles after a number of controls and observations have taken place. The converged posterior can be seen in Figure 5.1(c).

One of the drawbacks of the PF is that the number of particles needed to approximate a distribution scales exponentially with the state space size. Therefore, a straight PF implementation is not feasible for a SLAM application, where the length of the state vector tends to keep increasing. However, through a process called Rao-Blackwellization, the SLAM problem can be divided into separate landmark estimation problems, conditioned on the estimate of the robot's path. The resulting algorithm is called FastSLAM [48, 70] which is an example of a Rao-Blackwellized Particle Filter (RBPF) [19, 20].

5.2 Factoring the Posterior

For the system given in Equation (3.4), common SLAM approaches try to compute the posterior probability on robot pose and map given as:

$$p(\mathbf{x}(k) \mid \mathbf{u}^k, \mathbf{z}^k),$$
$$\text{where } \mathbf{u}^k = [\mathbf{u}(1),\ \ldots,\ \mathbf{u}(k)],$$
$$\text{and } \mathbf{z}^k = [\mathbf{z}(1),\ \ldots,\ \mathbf{z}(k)].$$

FastSLAM computes a slightly different posterior, the posterior over map and robot path. Using Equation (2.1), we can write this posterior as:

$$p(\mathbf{x}^k_{UAV},\ \mathbf{x}_{map}(k) \mid \mathbf{u}^k, \mathbf{z}^k),$$
$$\text{where } \mathbf{x}^k_{UAV} = [\mathbf{x}_{UAV}(1),\ \ldots,\ \mathbf{x}_{UAV}(k)].$$

The SLAM problem can be summarized as estimating the location of all landmarks $\mathbf{x}_{map}(k)$ and the path of the vehicle $\mathbf{x}^k_{UAV}$ given all the past control inputs and measurements. Each of the measurements, denoted by $\mathbf{z}(1),\ \ldots,\ \mathbf{z}(k)$, is a function of landmark position and UAV pose $\mathbf{x}_{UAV}(k)$ at the time the measurement was taken. Therefore, with the knowledge of $\mathbf{x}^k_{UAV} = \mathbf{x}_{UAV}(1),\ \ldots\ , \mathbf{x}_{UAV}(k)$, landmark measurements become independent of each other. With this, the SLAM posterior can be factored into a product of simpler terms:

$$p(\mathbf{x}^k_{UAV},\ \mathbf{x}_{map}(k) \mid \mathbf{u}^k, \mathbf{z}^k) = p(\mathbf{x}^k_{UAV} \mid \mathbf{u}^k, \mathbf{z}^k)\ p(\mathbf{x}_{map}(k) \mid \mathbf{u}^k, \mathbf{z}^k), \tag{5.1a}$$

$$p(\mathbf{x}_{map}(k) \mid \mathbf{u}^k, \mathbf{z}^k) = \prod_{i=1}^{n_k} p(\mathbf{x}_i(k) \mid \mathbf{u}^k(k), \mathbf{z}^k(k)), \tag{5.1b}$$

$$p(\mathbf{x}^k_{UAV},\ \mathbf{x}_{map}(k) \mid \mathbf{u}^k, \mathbf{z}^k) = \underbrace{p(\mathbf{x}^k_{UAV} \mid \mathbf{u}^k, \mathbf{z}^k)}_{\text{path posterior}} \underbrace{\prod_{i=1}^{n_k} p(\mathbf{x}_i(k) \mid \mathbf{u}^k, \mathbf{z}^k)}_{\text{landmark estimators}}. \tag{5.1c}$$

This factorization, first introduced by Murphy and Russel [52], states that the SLAM posterior can be separated into a robot path posterior $p(\mathbf{x}_{UAV}^{k} |\ \mathbf{u}^{k}, \mathbf{z}^{k})$ and n_k landmark posteriors conditioned on the robot's path.

5.3 The FastSLAM Algorithm

The factorization of the SLAM posterior in Equation (5.1) shows that SLAM problem can be solved without maintaining cross-correlations explicitly under appropriate conditioning. FastSLAM exploits this feature by maintaining $n_k + 1$ filters, one for each term in Equation (5.1c). With this, each filter becomes low-dimensional.

FastSLAM uses a particle filter to maintain an estimate of the path posterior, the first term in Equation (5.1c). On the other hand, the landmark posteriors are estimated using EKFs. Each particle has its own set of EKFs, one for each landmark which makes each EKF low-dimensional. The landmark EKFs are all conditioned on the particular particle's path posterior. In total, there are $n_k \times M$ EKFs, where M is the particle count in the particle filter. This form is an example of the Rao-Blackwellized Particle Filter mentioned before.

Each FastSLAM particle is in the form:

$$\mathbf{S}^{[m]}(k) = \{\mathbf{x}_{UAV}^{k,[m]}, \boldsymbol{\mu}_1^{[m]}(k), \boldsymbol{\Sigma}_1^{[m]}(k), \ldots, \boldsymbol{\mu}_{n_k}^{[m]}(k), \boldsymbol{\Sigma}_{n_k}^{[m]}(k)\}$$

with $[m]$ representing the index of the particle, $\mathbf{x}_{UAV}^{k,[m]}$ is the m-th particle's path estimate and $\boldsymbol{\mu}_i^{[m]}(k)$ and $\boldsymbol{\Sigma}_i^{[m]}(k)$ represent the mean and covariance of the Gaussian estimate of the i-th landmark conditioned on the path posterior $\mathbf{x}_{UAV}^{k,[m]}$. Filtering, obtaining the posterior at step k from the posterior of step $k-1$, is done by generating a new particle set $\mathbf{S}(k)$ from $\mathbf{S}(k-1)$ incorporating the latest control input $\mathbf{u}(k)$ and the latest measurement $\mathbf{z}(k)$.

1. Sample a new robot pose for each particle given the new control. **2**. Update the landmark EKFs of the observed feature in each particle. **3**. Calculate the importance weight for each particle. **4**. Draw a new unweighted particle set using importance resampling.

Table 5.1: FastSLAM steps summarized.

This process consists of four sub-steps. *Prediction* is the first, where a new pose for each particle is drawn incorporating the control input and added to the particular path estimate $s^{[m]}(k-1)$. In the *update* step, the EKF of the observed landmarks are updated with the observations. With the next step, each particle is given an *importance* weight, reflecting the difference between the prediction and observation. Finally a new set of particles $S(k)$ is drawn from the weighted particles using *importance resampling*. These four steps are summarized in Table 5.1.

5.3.1 Prediction Step

The first step of FastSLAM is to generate probabilistic guesses of the new robot pose at step k given particles at step $k-1$. This prediction is obtained from sampling the probability distribution given as:

$$\mathbf{x}_{UAV}^{[m]}(k) \sim p(\mathbf{x}_{UAV}(k) \mid \mathbf{x}_{UAV}^{[m]}(k-1), \mathbf{u}(k)),$$

Assuming the particles $\mathbf{S}(k-1)$ are distributed according to the known distribution $p(\mathbf{x}_{UAV}^{k-1} \mid \mathbf{u}^{k-1}, \mathbf{z}^{k-1})$, the new particles are distributed according to:

$$p(\mathbf{x}_{UAV}^{k} \mid \mathbf{u}^{k}, \mathbf{z}^{k-1}).$$

which is referred as proposal distribution. Using the motion model given in Equation (3.3) the new pose $\mathbf{x}^{[m]}(k)$ of the m-th particle can be obtained by

adding the new control forward from the previous pose $\mathbf{x}^{[m]}(k-1)$ similar to the prediction step of the EKF.

$$\mathbf{x}_{UAV}^{[m]}(k) = \begin{bmatrix} x_r^{[m]}(k-1) + \cos(\theta_r^{[m]}(k-1))u_1(k) \\ y_r^{[m]}(k-1) + \sin(\theta_r^{[m]}(k-1))u_1(k) \\ \theta_r^{[m]}(k-1) + u_2(k) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}.$$

5.3.2 Update of Landmarks

Since FastSLAM maintains the estimate of landmarks with EKF, the update step is the same as EKF update equations, however, this time there is a separate EKF for each landmark in each particle. The update equations can be summarized as follows:

$$\begin{aligned}
\mathbf{z}_i(k) &= \mathbf{h}_i(x_i, y_i) + \mathbf{v}_i(k), \\
\mathbf{h}_i(x_i, y_i) &= \begin{bmatrix} h_{i_1} \\ h_{i_2} \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta x_i{}^2 + \Delta y_i{}^2} \\ \arctan2\left(-\Delta y_i, -\Delta x_i\right) - \theta_r(k) \end{bmatrix}, \\
&\text{and } \Delta x_i = x_r(k) - x_i(k),\ \Delta y_i = y_r(k) - y_i(k), \\
\mathbf{H}_i(k) &= \nabla \mathbf{h}_i = \frac{\partial \mathbf{h}_i(x_i, y_i)}{\partial \phi_i} = \begin{bmatrix} \frac{-\Delta x_i}{p_i} & \frac{-\Delta y_i}{p_i} \\ \frac{\Delta y_i}{p_i^2} & \frac{-\Delta x_i}{p_i^2} \end{bmatrix}, \\
&\text{where } \phi_i = [x_i, y_i],\ p_i = \sqrt{\Delta x_i^2 + \Delta y_i^2}. \\
\boldsymbol{\mu}_i^{[m]}(k) &= \boldsymbol{\mu}_i^{[m]}(k-1) + \mathbf{W}(k)\boldsymbol{\nu}(k), \\
\boldsymbol{\Sigma}_i^{[m]}(k) &= \boldsymbol{\Sigma}_i^{[m]}(k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k), \\
\boldsymbol{\nu}(k) &= \mathbf{z}_i(k) - \mathbf{h}_i(x_i, y_i), \\
\boldsymbol{\Psi}(k) &= \mathbf{H}_i(k)\boldsymbol{\Sigma}_i^{[m]}(k-1)\mathbf{H}_i^T(k) + \mathbf{R}(k), \\
\mathbf{W}(k) &= \boldsymbol{\Sigma}_i^{[m]}(k-1)\mathbf{H}_i^T(k)\boldsymbol{\Psi}^{-1}(k),
\end{aligned}$$

With these, the update of a landmark becomes a constant time operation, and for the overall particle set, the update time required increases linearly with the

number of landmarks n_k, unlike EKF-SLAM where the update time is quadratic in n_k.

5.3.3 Calculating Importance Weights

Particles drawn in the prediction step, using the motion model are distributed according to the distribution $p(\mathbf{x}_{UAV}^k \mid \mathbf{u}^k, \mathbf{z}^{k-1})$, whereas we need particles from the desired posterior $p(\mathbf{x}_{UAV}^k \mid \mathbf{u}^k, \mathbf{z}^k)$. The importance sampling corrects this difference by modifying particle weights that account for this difference. For FastSLAM, the importance weight for a single particle $w^{[m]}(k)$ is equal to the ratio of the SLAM posterior and the proposal distribution given previously:

$$w^{[m]}(k) = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(\mathbf{x}_{UAV}^{[m]} \mid \mathbf{u}^k, \mathbf{z}^k)}{p(\mathbf{x}_{UAV}^{[m]} \mid \mathbf{u}^k, \mathbf{z}^{k-1})}.$$

Considering that we have landmark estimators in EKF form, this weight can be computed in closed form, commonly computed in terms of the *innovation* $\boldsymbol{\nu}(k)$ which is the difference between the actual observation $\mathbf{z}_i(k)$ and the predicted observation $\mathbf{h}_i(x_i, y_i)$ given in the previous section. The sequence of innovations in the EKF is Gaussian distributed with zero mean and covariance $\boldsymbol{\psi}(k)$, where $\boldsymbol{\psi}(k)$ is the innovation covariance matrix also defined in the previous section. With these, the importance weight can be calculated as:

$$w^{[m]}(k) = \frac{1}{\sqrt{|2\pi\boldsymbol{\psi}(k)|}} e^{-\frac{1}{2}\boldsymbol{\nu}^T(k)\boldsymbol{\psi}^{-1}(k)\boldsymbol{\nu}(k)}$$

The importance weight calculation is constant-time operation per particle, per observed landmark.

5.3.4 Importance Resampling

The particles we obtained in the previous three steps were temporary particles. Once the importance weights are assigned, a new set of samples is drawn, with replacement, with probabilities equal to the importance weights. Naive implementation of sampling requires linear time in the number of the landmarks, but with a sophisticated implementation scheme this can be reduced to $\mathcal{O}(\log(n_k))$ by organizing the landmarks as a binary tree instead of an array.

5.3.5 Landmark Augmentation in FastSLAM

Because of the nature of the SLAM problem, we have an increasing number of landmarks in order to maintain a map estimate, since at each step we may encounter new landmarks which must be added to our estimators. In FastSLAM this step is rather easy, where we just have to initialize new landmark mean and covariance for each particle, using similar relations to that of EKF-SLAM. We can summarize this procedure as follows:

$$\mathbf{g}(\mathbf{x}_{UAV}^{[m]}(k), \mathbf{z}_i(k)) = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} x_r(k) + r_i \cos(\theta_i + \theta_r(k)) \\ y_r(k) + r_i \sin(\theta_i + \theta_r(k)) \end{bmatrix},$$

$$\boldsymbol{\mu}_{n_k+1}^{[m]}(k) = \mathbf{g}(\mathbf{x}_{UAV}^{[m]}(k), \mathbf{z}_i(k))$$

$$\boldsymbol{\Sigma}_{n_k+1}^{[m]}(k) = \nabla\mathbf{G}_z \mathbf{R}(k) \nabla\mathbf{G}_z^T,$$

$$\nabla\mathbf{G}_z = \begin{bmatrix} \frac{\partial g_1}{\partial r_i} & \frac{\partial g_1}{\partial \theta_i} \\ \frac{\partial g_2}{\partial r_i} & \frac{\partial g_2}{\partial \theta_i} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i + \theta_r(k)) & -r_i \sin(\theta_i + \theta_r(k)) \\ \sin(\theta_i + \theta_r(k)) & r_i \cos(\theta_i + \theta_r(k)) \end{bmatrix}.$$

where $\nabla\mathbf{G}_z$ is the gradient of $\mathbf{g}(\mathbf{x}_{UAV}^{[m]}(k), \mathbf{z}_i(k))$ with respect to $\mathbf{x}_{UAV}^{[m]}$ and $\mathbf{z}_i(k)$.

5.4 FastSLAM 1.0 and FastSLAM 2.0

There are two versions of FastSLAM algorithms in the literature, FastSLAM 1.0 [70] and FastSLAM 2.0 [49], which differ only in the proposal distribution-prediction step and importance weights. The FastSLAM 2.0 algorithm is superior to the FastSLAM 1.0. FastSLAM 2.0 proposes solutions to the inherent problems in FastSLAM 1.0 such as particle impoverishment which is a result of the decreasing diversity due to the structure of the algorithm. FastSLAM 2.0 also provides a proof of convergence. In FastSLAM 1.0, the proposal distribution is obtained from the motion model as:

$$\mathbf{x}_{UAV}^{[m]}(k) \sim p(\mathbf{x}_{UAV}(k) \mid \mathbf{x}_{UAV}^{[m]}(k-1), \mathbf{u}(k)),$$

and with this, the weights are calculated according to the marginalized observation model:

$$w^{[m]}(k) \sim w^{[m]}(k) p(\mathbf{z}(k) | \mathbf{x}_{UAV}^{k,[m]}, \mathbf{z}^{k-1})$$

In FastSLAM 2.0, the proposal distribution includes the current observation as well, which results in:

$$\begin{aligned} \mathbf{x}_{UAV}^{[m]}(k) &\sim p(\mathbf{x}_{UAV}(k) \mid \mathbf{x}_{UAV}^{[m]}(k-1), \mathbf{u}(k), \mathbf{z}(k)) \\ &\sim p(\mathbf{x}_{UAV}(k) \mid \mathbf{x}_{UAV}^{k,[m]}, \mathbf{u}^k, \mathbf{z}^k) \end{aligned}$$

Conditioned on all the available information $\mathbf{x}_{UAV}^{k,[m]}$, $\mathbf{u}^k$ and $\mathbf{z}^k$, this gives the smallest possible variance in importance weight $w^{[m]}(k)$ for each particle. The proposal distribution becomes optimal, making FastSLAM 2.0 more advantageous.

5.5 Multi-UAV SLAM with PF

Although PF does not inherently possess characteristics that will enable an easy implementation of multi-UAV SLAM algorithm, an algorithm based on making use of information form to benefit from the additivity can be developed. Similar to the multi-UAV IF case, we are considering a decentralized SLAM application where each UAV has its own estimate of its location and the map but does not maintain estimate of other UAVs. Each UAV receives map information from other UAVs as a set of particles, which can be incorporated into its own set of particles by merging the estimates of each particle with a simple addition in the information form. After a data association step, the receiving UAV adds the information received by firstly merging the matched landmarks using the information form and then by initializing new landmarks in each particle for the landmarks that are not matched.

Chapter 6

Results

For the evaluation and comparison of the four filters EKF, IF, FastSLAM 1.0 and FastSLAM 2.0 described in the previous chapters, MATLAB implementations of these filters are tested on simulated data sets created by the system described in Chapter 2. For FastSLAM 1.0 and 2.0 we modified the sample codes from Tim Bailey's website for our scenario [3]. These simulated datasets consist of three different paths: linear, circular and eight-shaped. Example instances for each filter on each path type are given in Figures 6.1–6.24 which demonstrate how the filters work. FastSLAM 1.0 and 2.0 are executed with a moderate number of 100 particles in these instances. Each run has a set of figures showing the run at intermediate steps (at every 20th step), followed by estimation error plots at the end of the run. In each figure pair, the red path on the left shows the actual path along which the UAV travels through and the black colored path shows the path estimated by the filter. The figures on the right show the real and estimated positions of the landmarks as well as the 3σ confidence level ellipses for that step. The green triangle on the right part indicates the position and the heading of the UAV at the given step. Following each run, error plots are presented showing estimation error in the robot pose given as x_r and y_r coordinates, bearing θ_r and also average estimation errors in the landmark positions x and y.

In Figures 6.1–6.8, a linear path is given, where the UAV goes back and forth on a straight line at a speed of $V = 10$ pixel/s. The process noise has no angular component so the real path of the UAV does not deviate from the line. The control inputs have a process noise with variance of (σ_V^2), where $\sigma_V = 1$ pixel/s and the observation noise variances are (σ_R^2, σ_B^2) with range component $\sigma_R = 1$ pixel and bearing component $\sigma_B = 1.5°$. Figures 6.9–6.16 show an instance of a circular path, and Figures 6.17–6.24 demonstrate an eight-shaped path. For these paths the observation noise is the same as in the linear path, but the process noise has an angular speed component as well, with variances $(\sigma_V^2, \sigma_\omega^2)$ where $\sigma_V = 1$ pixel/s similarly and $\sigma_\omega = 2$ °/s is the standard deviation of the angular velocity noise.

The SLAM algorithm progresses as follows: As the vehicle moves and new observations are acquired, the error in the estimates accumulate since both the process and the observation models are noisy. As the landmarks are observed, they are matched to the existent map, and these errors can be pruned out by the filter. This process continues at each step which can be observed in the given figures. The more the landmarks are observed, the faster the estimated path converges to the actual path. As new landmarks are observed, the estimated path again deviates from the actual path as these new landmarks introduce some estimation error into the process. As the landmarks are re-observed, their error covariance decreases and the corresponding error covariance ellipses shrink. The estimation of the landmarks' convergence to the real positions of the landmarks can be observed in these figures as well. The convergence results from mainly re-observing of the same landmarks in the succeeding steps as the acquired images overlap. We also observe an overall decrease in error at the completion of each loop, which is called 'loop closure' where relatively high number of observed landmarks are matched to the map. However, the effect of loop closure is not significant after the aforementioned overlapping effect. The outer landmarks'

error covariance ellipses do not diminish in size as much as the inner landmarks, as the outer ones tend to be observed less.

However, we also observe that not all error in the estimate can be pruned out. Not all landmarks converge to their actual position with each prediction-update step, some even diverge from the actual positions. The position of the landmarks converge to local minima where the difference between predicted observations and real observations become biased, which appears as a small error in the updates with small innovation. This prevents the landmark positions and the UAV position from converging to their real positions. Instead, they converge to another configuration which appears as actual positions when the prediction error is calculated.

In Figures 6.25 and 6.26, average error levels of the four filters in 80 runs of each noise level for a circular path can be seen. For different range noise levels (Figure 6.25) we observe an increase in estimation errors with the increase in noise levels in range and bearing errors for all the filters, which is an expected result. However, FastSLAM becomes significantly more successful when the noise levels are quite high, whereas EKF and IF start to have large errors when the range noise is high. For the different bearing error case, this behaviour only appears in PF algorithms, whereas in EKF and IF with different bearing errors shows different characteristics, which is probably due to the linearization involved in these. With increased bearing error, EKF and IF become more suitable estimators with less error. We can say that PF based approaches perform similarly if not better than the KF-based algorithms, especially when the noise levels are high.

In Table 6.1, we can also see the average run times of these filters in the same 80 runs. Using these and also the given sample runs we can compare the filters regarding their performance. EKF and IF are essentially equivalent as mentioned

Filter	Average Run Times (sec)		
	Linear Path	Circular Path	Eight-shaped Path
Extended Kalman Filter	51	66	82
Information Filter	76	109	150
FastSLAM 1.0	91	77	71
FastSLAM 2.0	152	131	116

Table 6.1: Average run times of each filter for the three different paths.

before, the slight difference that appears is due to the numerical differences. However, the IF has a bigger run time with a naive implementation because of the matrix inverse operations involved in obtaining the state vectors at each step. The performance can be improved by using special care for these calculations. Using SEIF also can improve this performance. The main advantage of IF is the decentralization ability mentioned before. Looking at FastSLAM 1.0 and FastSLAM 2.0 results, we can conclude that these provide slightly better performance than KF variants. Although they seem to have long run times compared to EKF, this is mainly because of the implementation in MATLAB, which is not optimized to loop operations as opposed to matrix operations used in EKF. This performance can be improved using a different simulation environment which will also enable increasing the number or particles for better estimation. We can assume that FastSLAM 1.0 and 2.0 will offer better performance with less run time when compared to EKF and IF, with FastSLAM 2.0 being a better estimator with the drawback of a longer runtime.

A comparison of FastSLAM 1.0 and 2.0 performance with changing the particle counts is given in Figure 6.27, which shows that increasing the particle count up to a certain point improves the performance slightly. This also shows that especially for FastSLAM 2.0, small number of particles can be sufficient for a fairly good estimation. However, FastSLAM 1.0 and 2.0 also have their drawbacks, with their sampling and resampling processes which involves random sampling of the estimated posterior. Both can converge to a less accurate estimation at

some point and recovering from this situation is hardly possible with their algorithm. FastSLAM also shows a huge variance in the estimation error even on the same data, with the same noise levels. This can be solved with an increased number of particles, along with other techniques such as particle injection at an increased cost of computation.

In Figure 6.28, a sample run for a multi-UAV scenario is presented, with shots given at each 10th step before communication and after communication between UAVs. The plots show each UAV with its own map. Sharing the map information, each UAV can update its own map to a common map including information from the other UAV at the given communication step, which are at each 10th step. The performance of the process can also be seen. When compared to the eight-shaped path by a single UAV (Figures 6.17–6.24) using two UAVs for the same task has its advantages in robustness and also completion of tasks in less time. We also present a sample instance for the particle filter based multi-UAV SLAM algorithm in Figure 6.29.

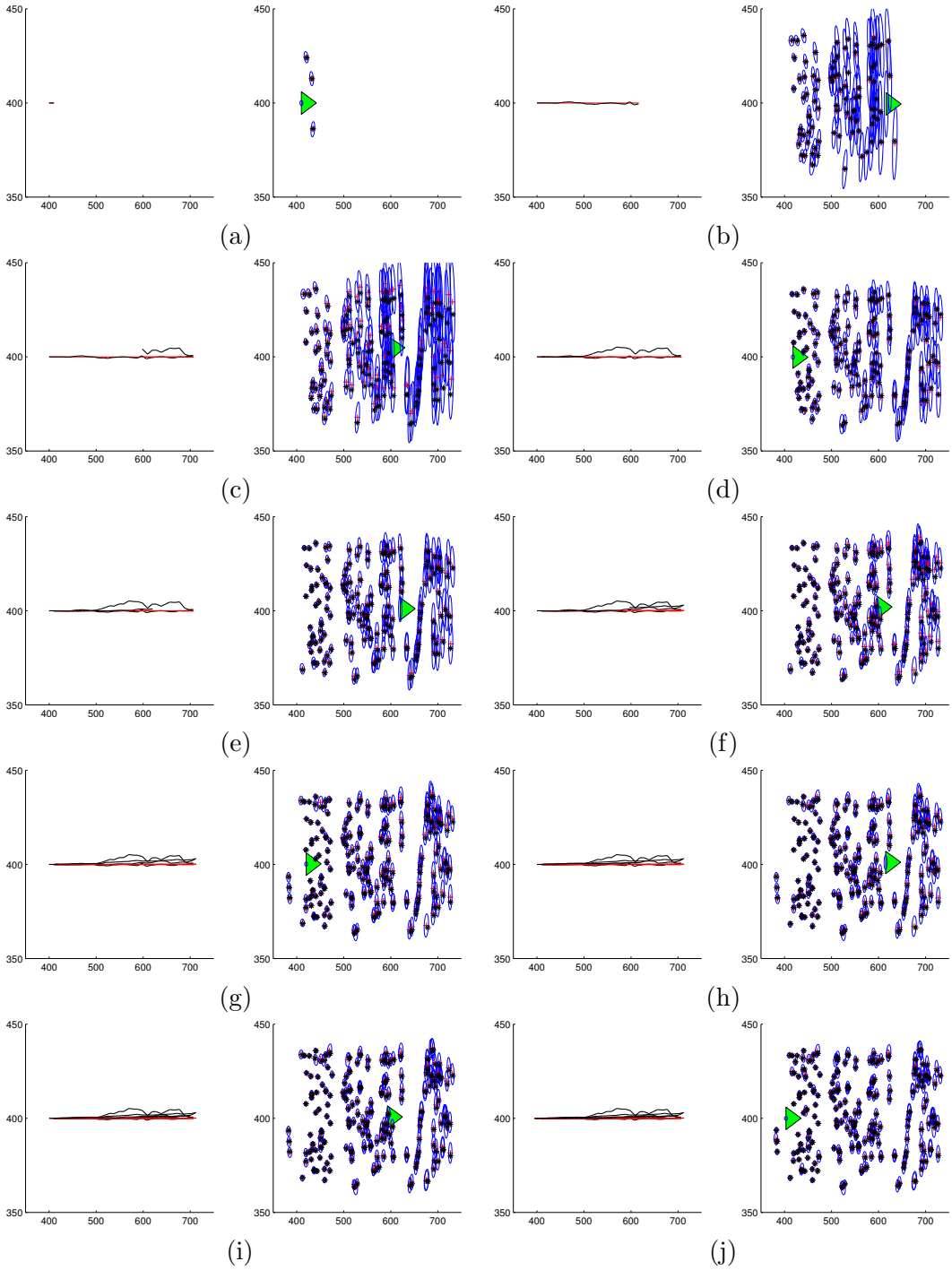


Figure 6.1: EKF-SLAM: sample run for a linear path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

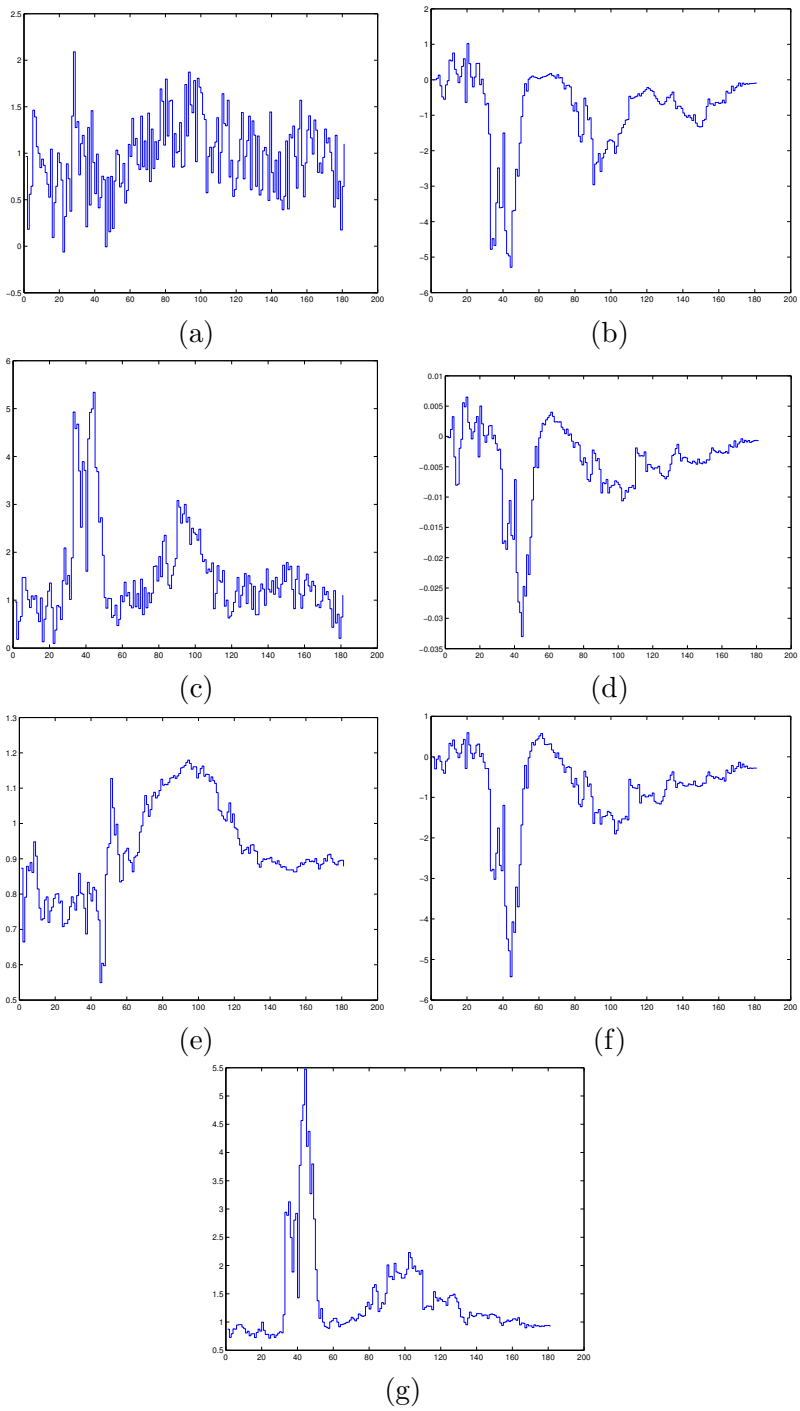


Figure 6.2: EKF-SLAM: error plots for the linear path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

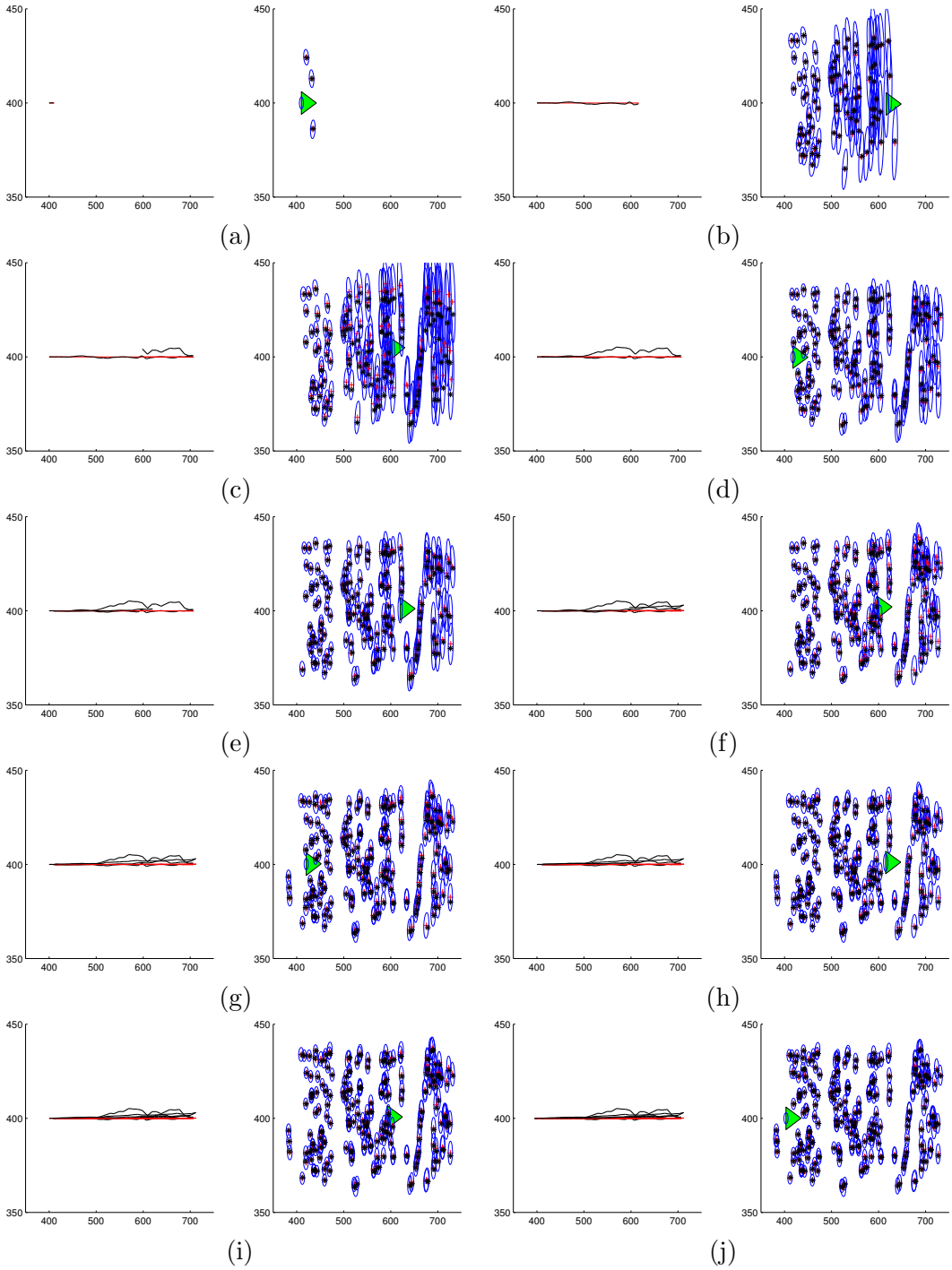


Figure 6.3: IF-SLAM: sample run for a linear path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

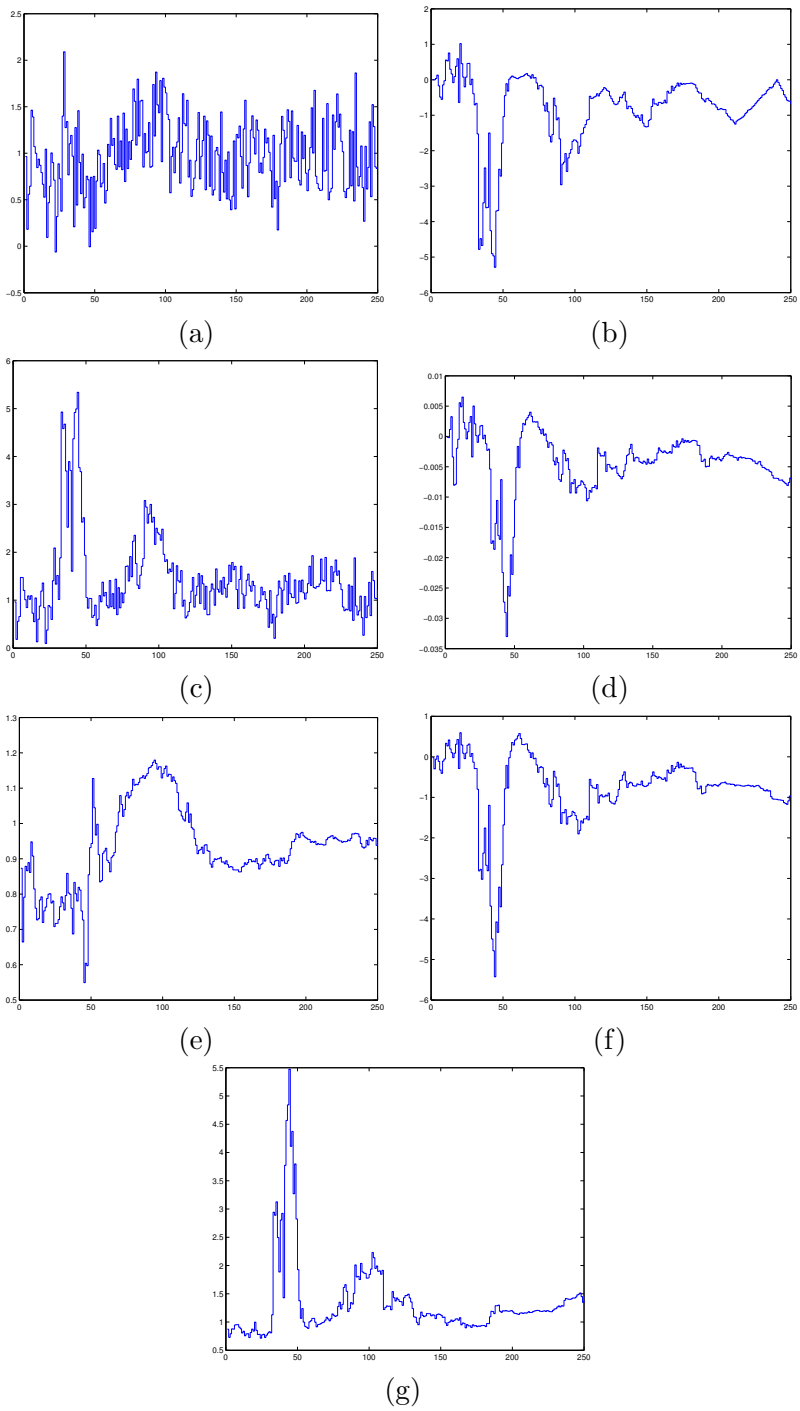


Figure 6.4: IF-SLAM: error plots for the linear path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

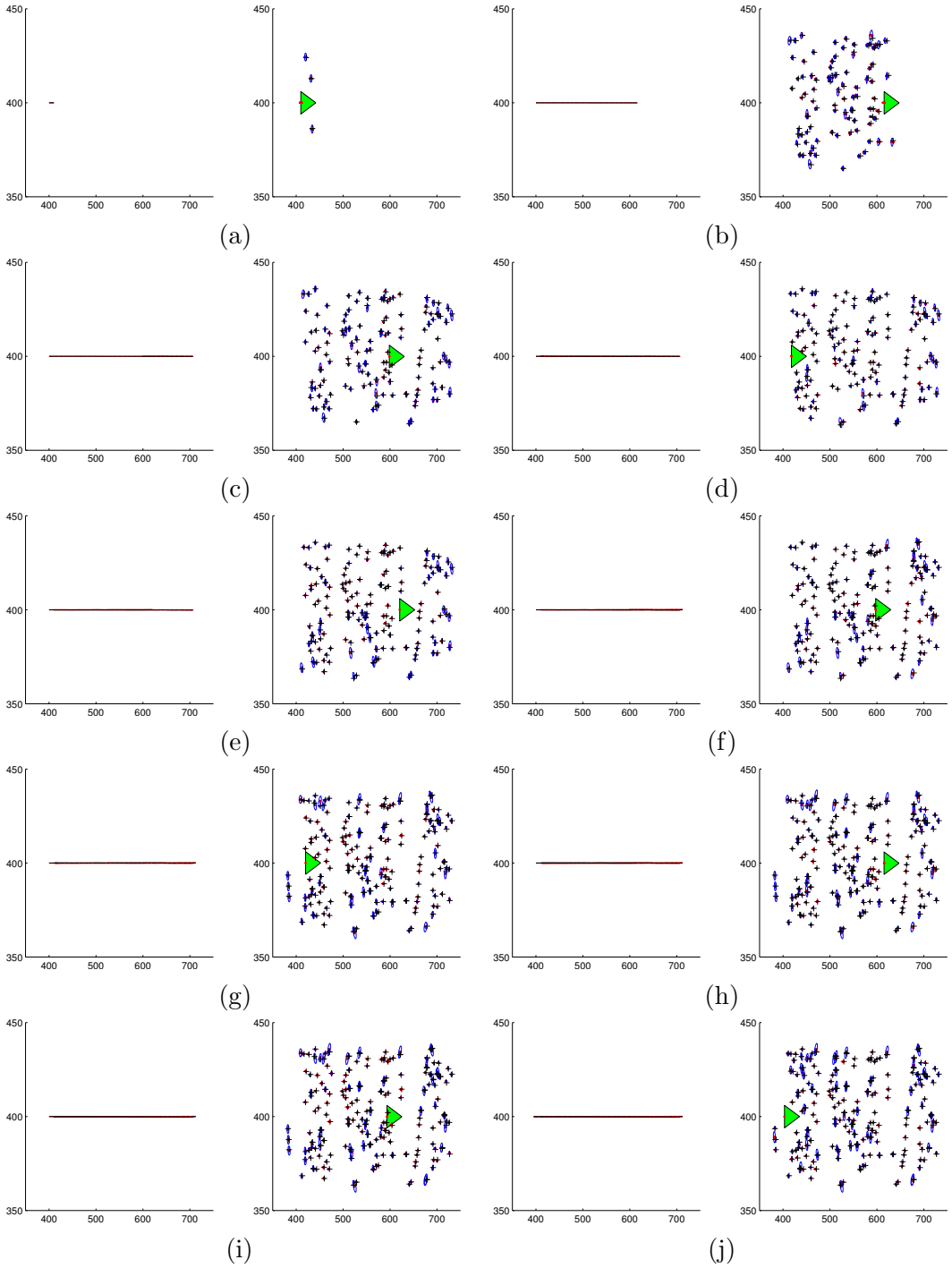


Figure 6.5: FastSLAM 1.0: sample run for a linear path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

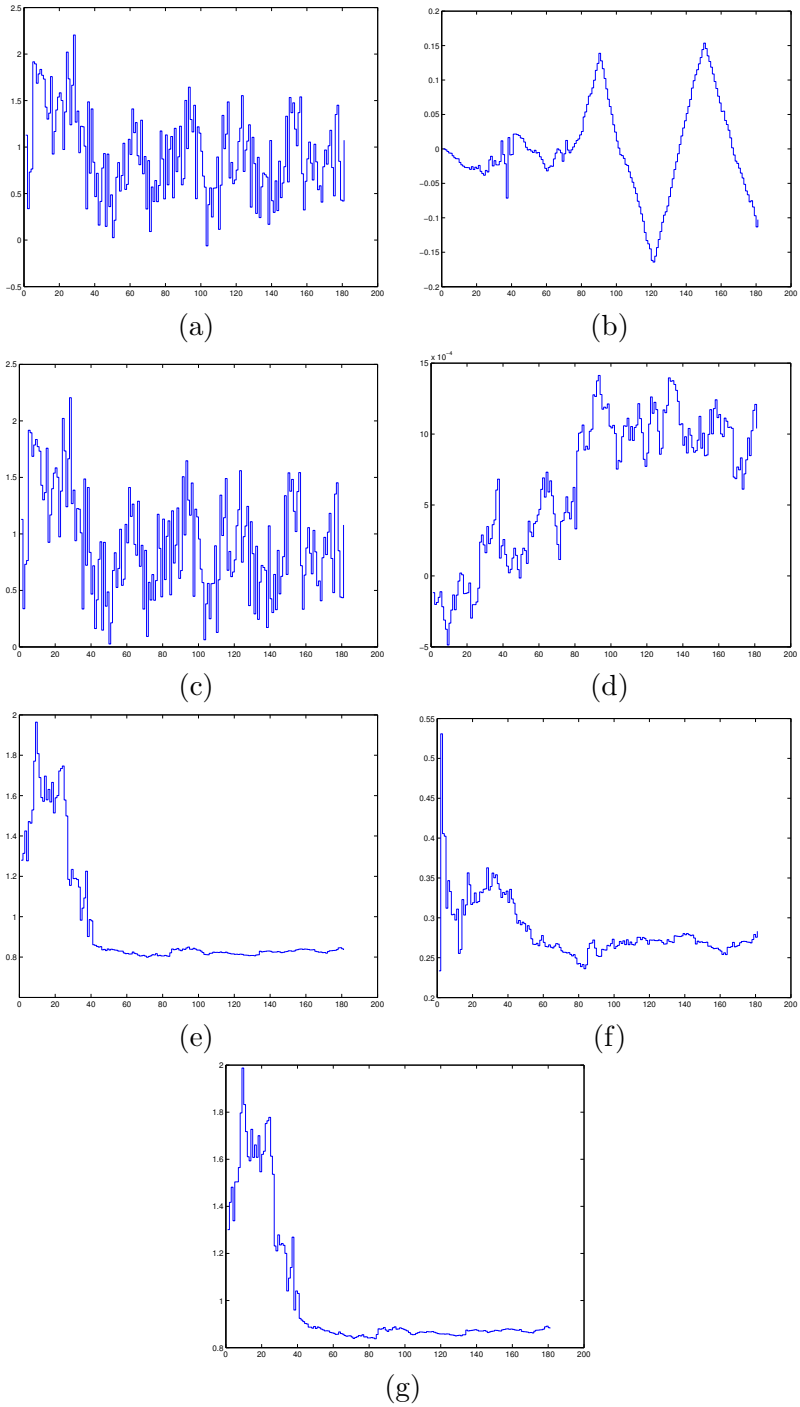


Figure 6.6: FastSLAM 1.0: error plots for the linear path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

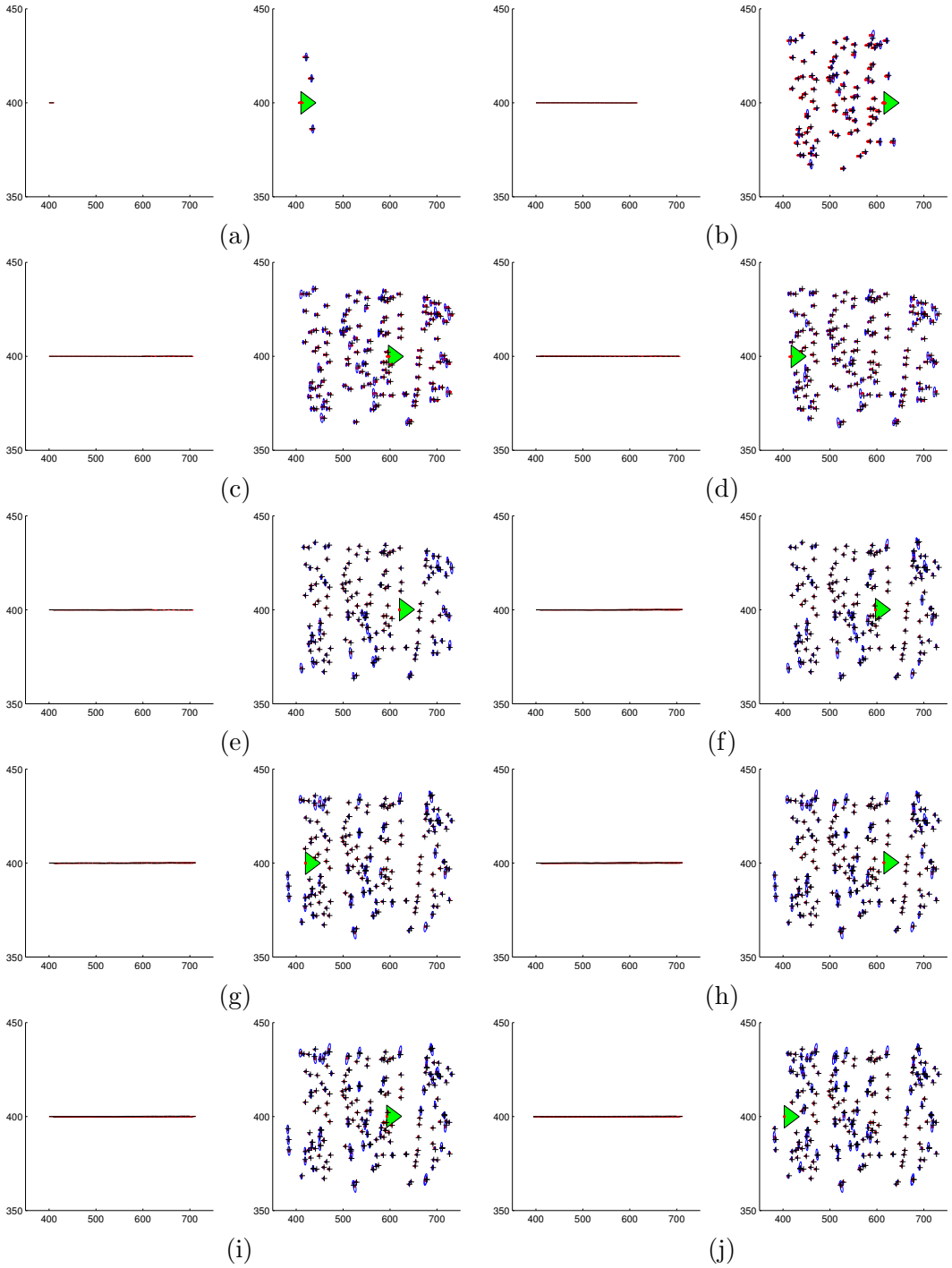


Figure 6.7: FastSLAM 2.0: sample run for a linear path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

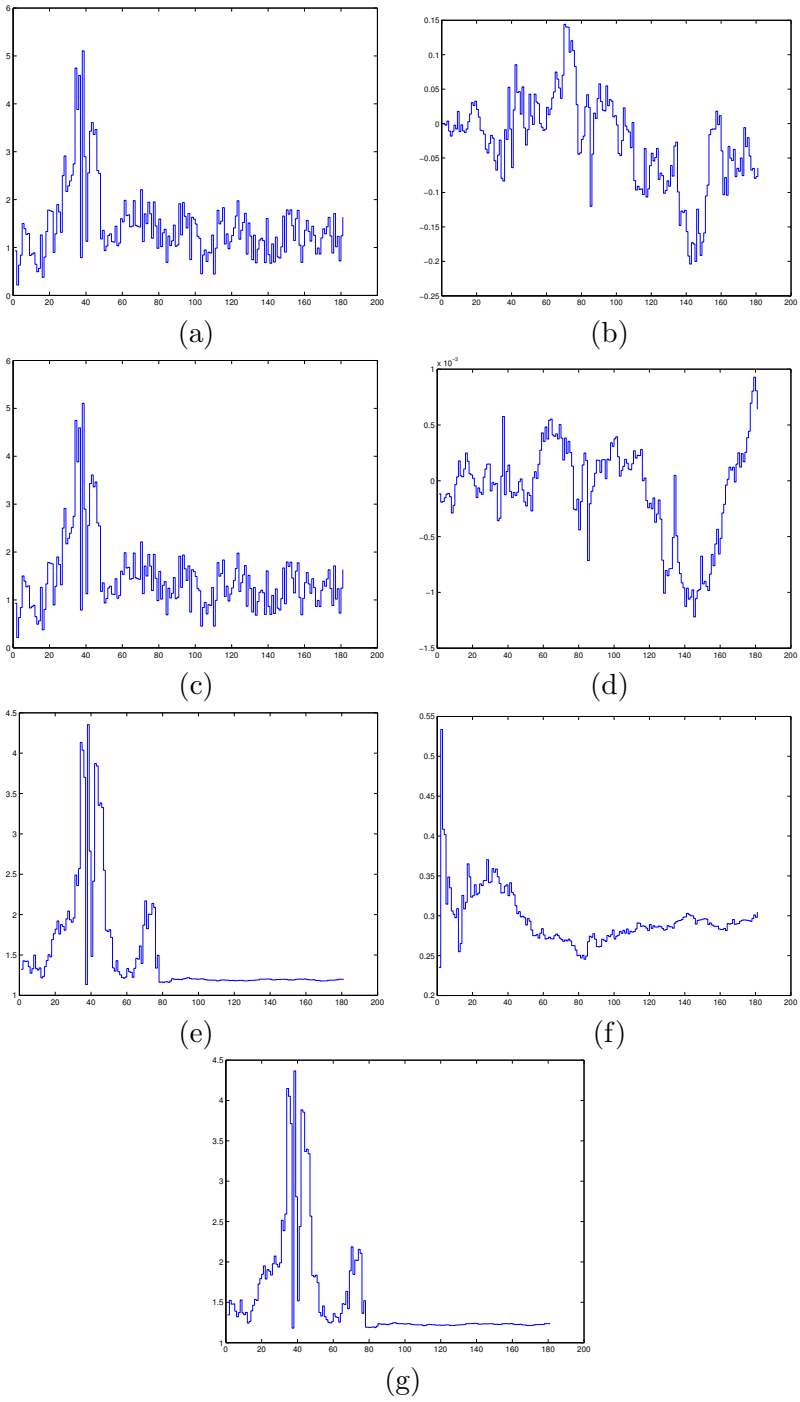


Figure 6.8: FastSLAM 2.0: error plots for the linear path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

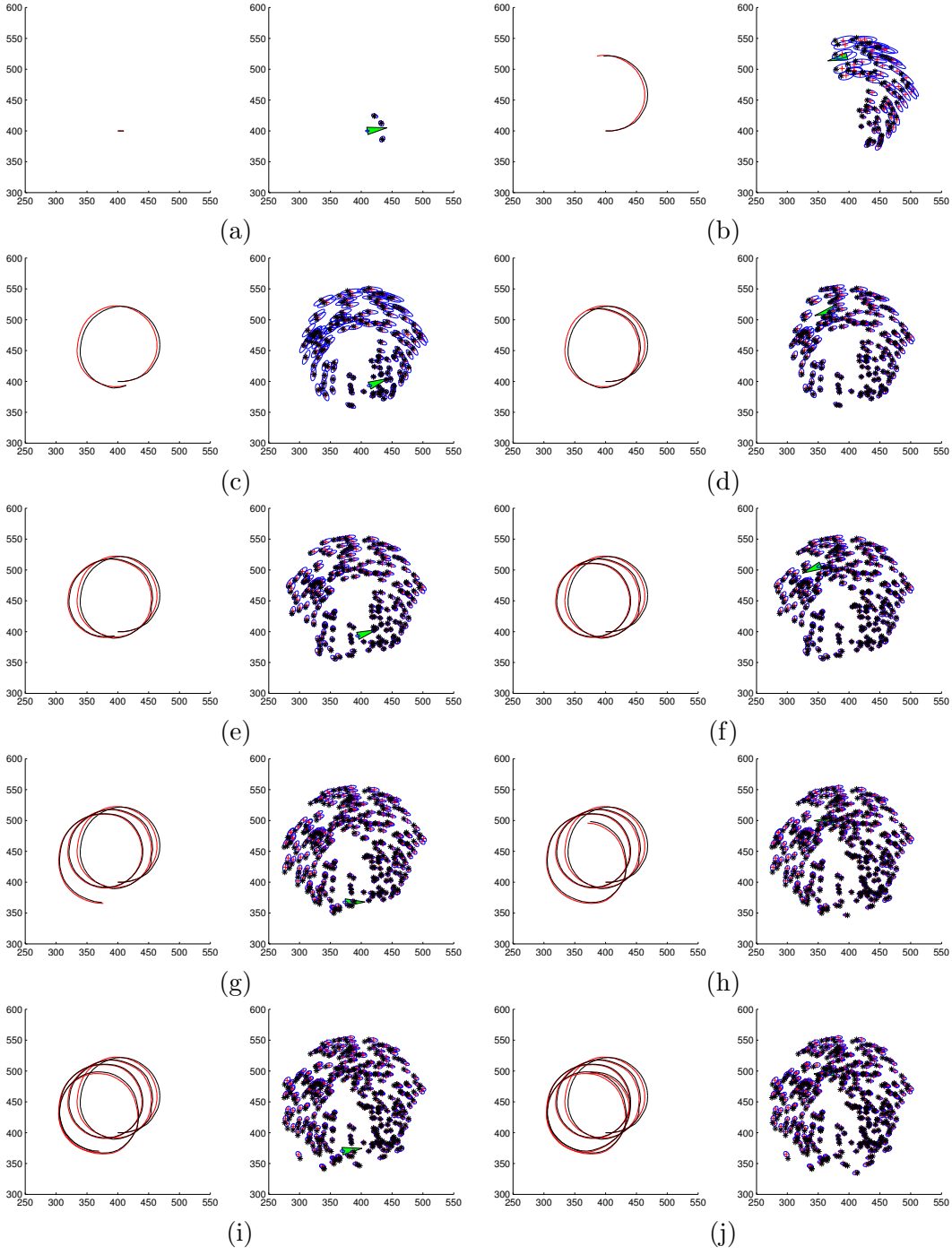


Figure 6.9: EKF-SLAM: sample run for a circular path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

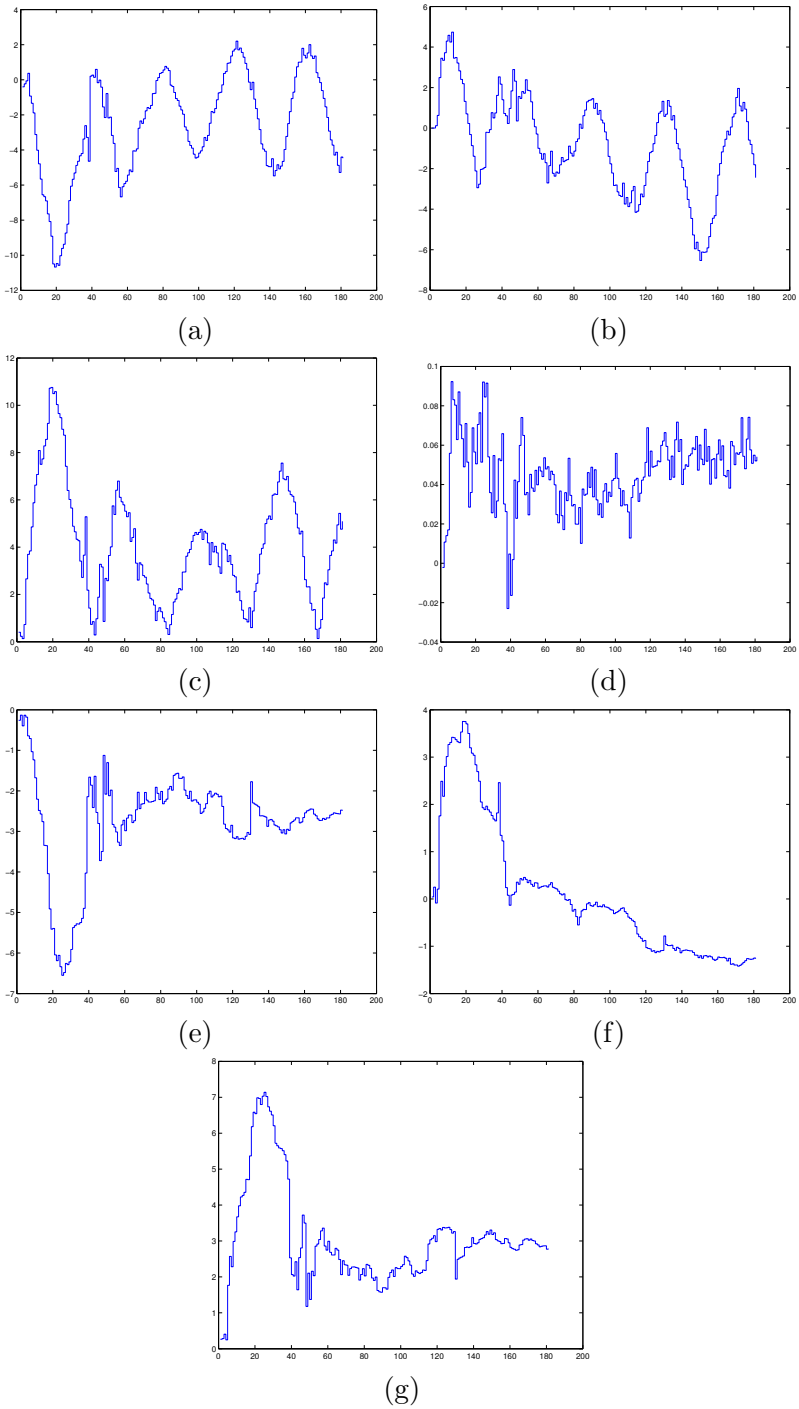


Figure 6.10: EKF-SLAM: error plots for circular path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

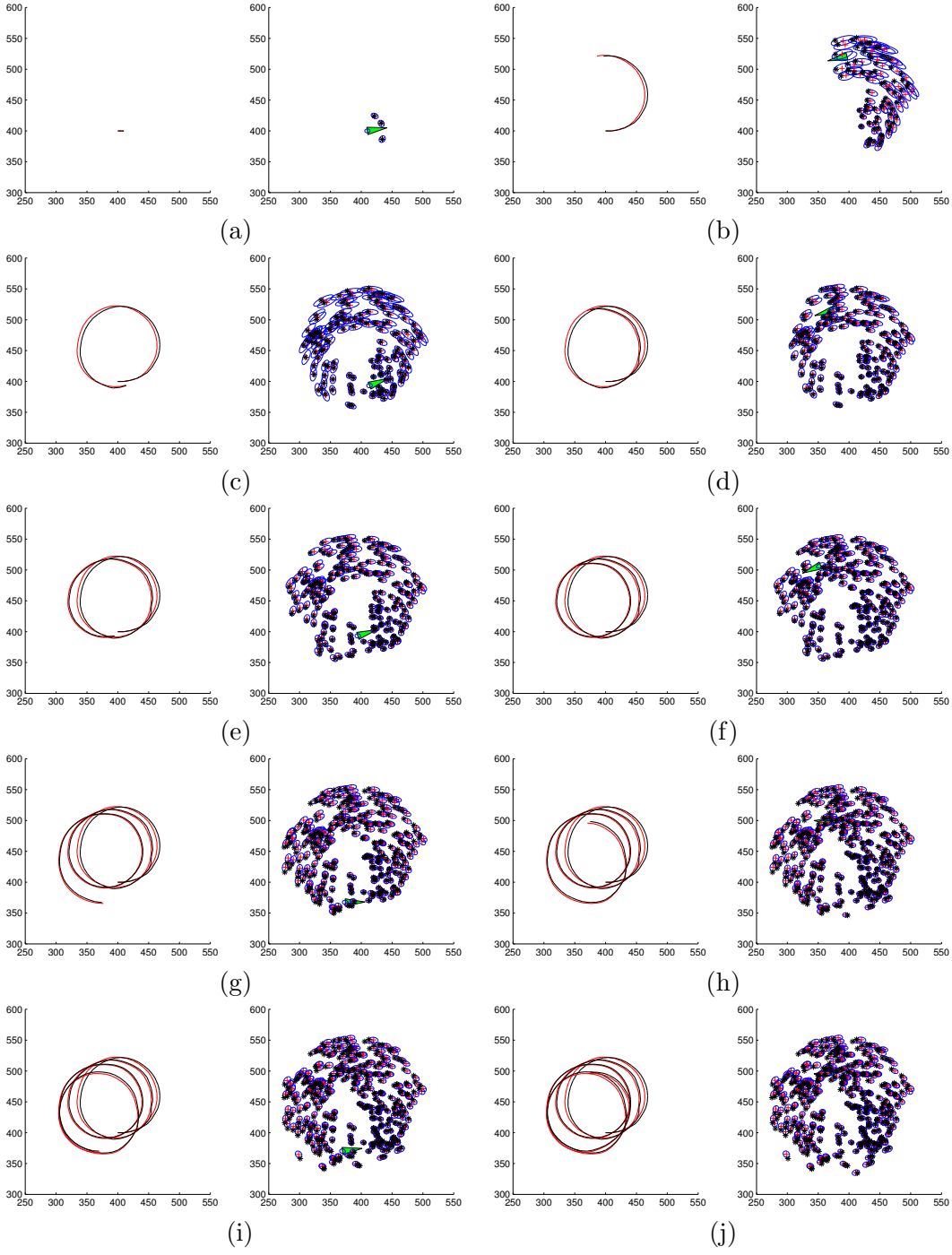


Figure 6.11: IF-SLAM: sample run for a circular path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

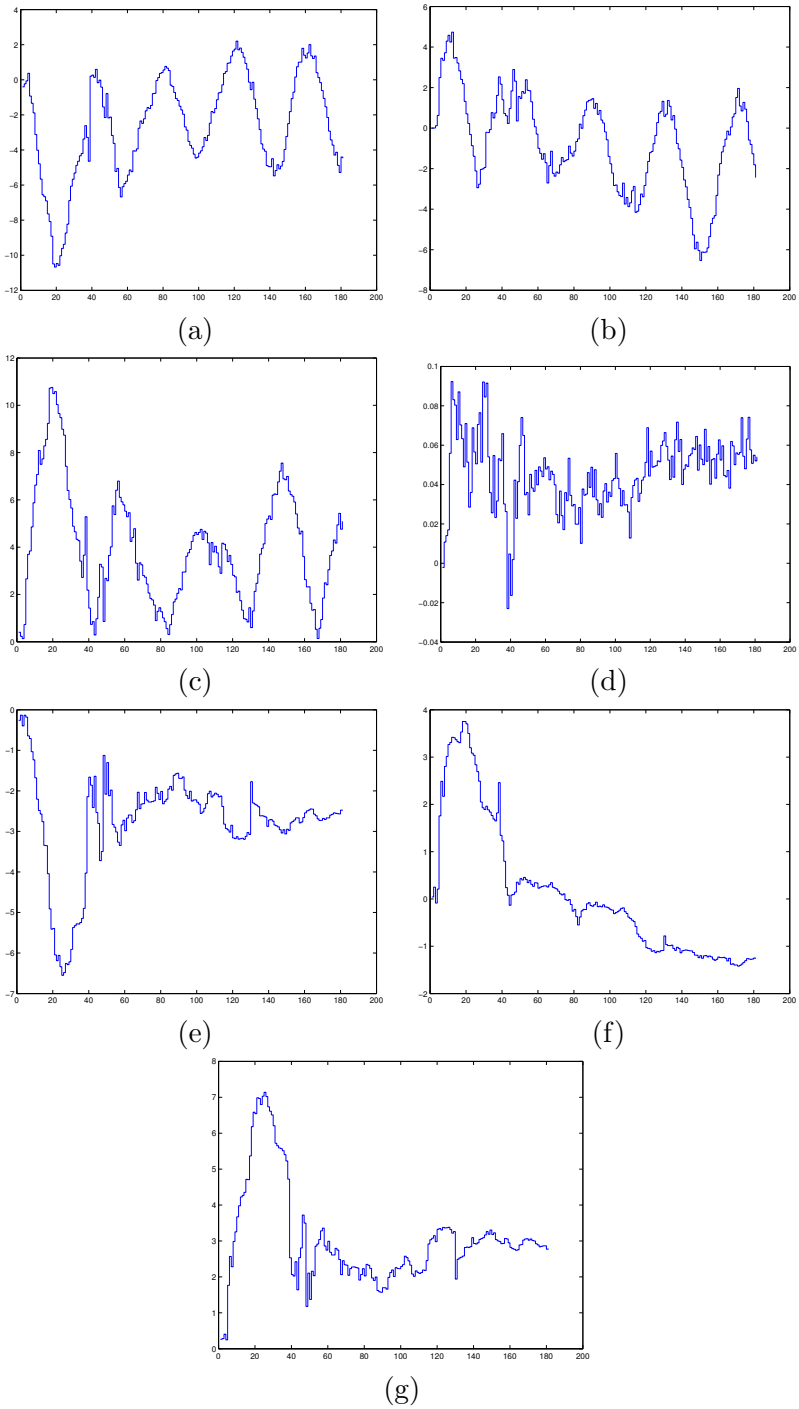


Figure 6.12: IF-SLAM: error plots for circular path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

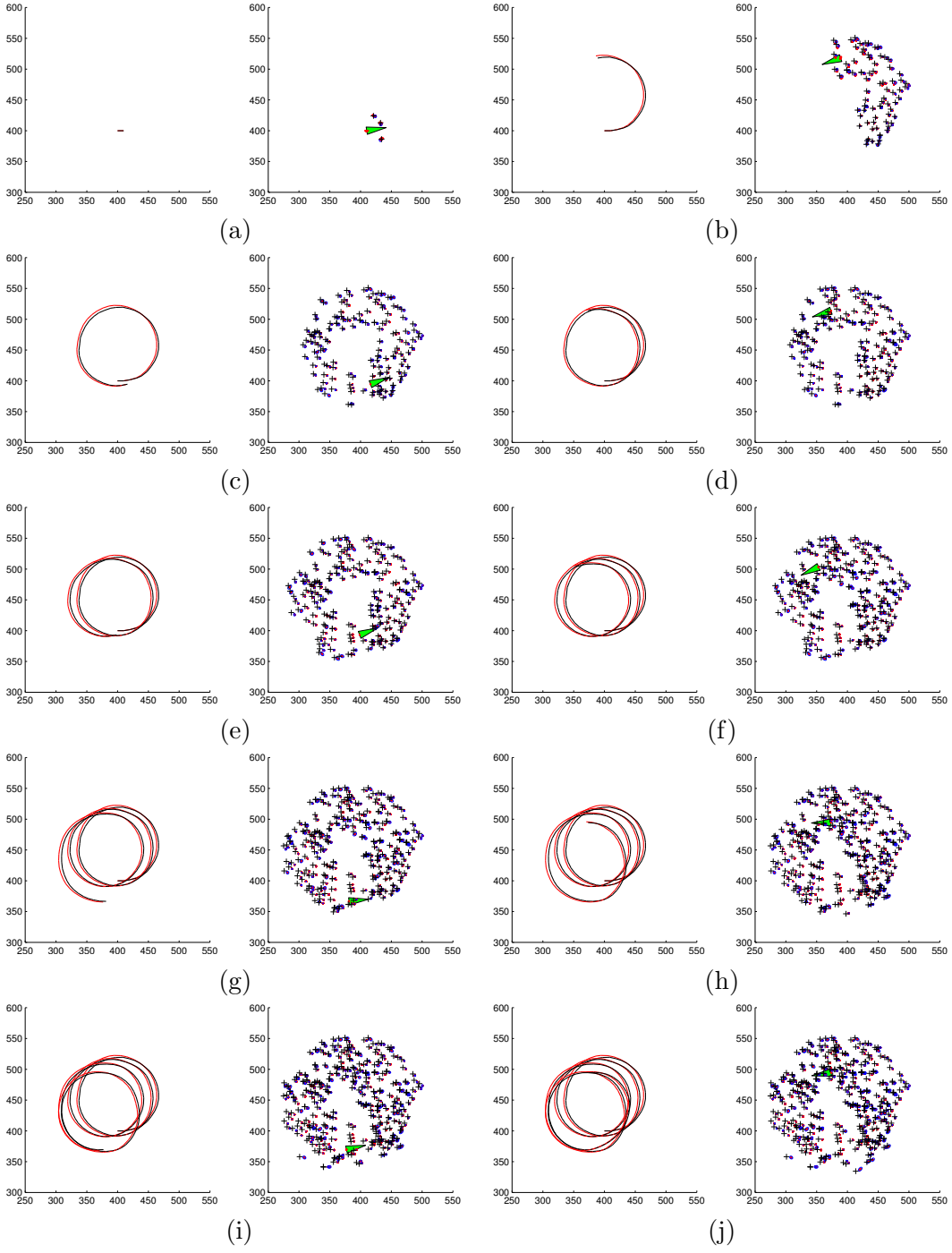


Figure 6.13: FastSLAM 1.0: sample run for a circular path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

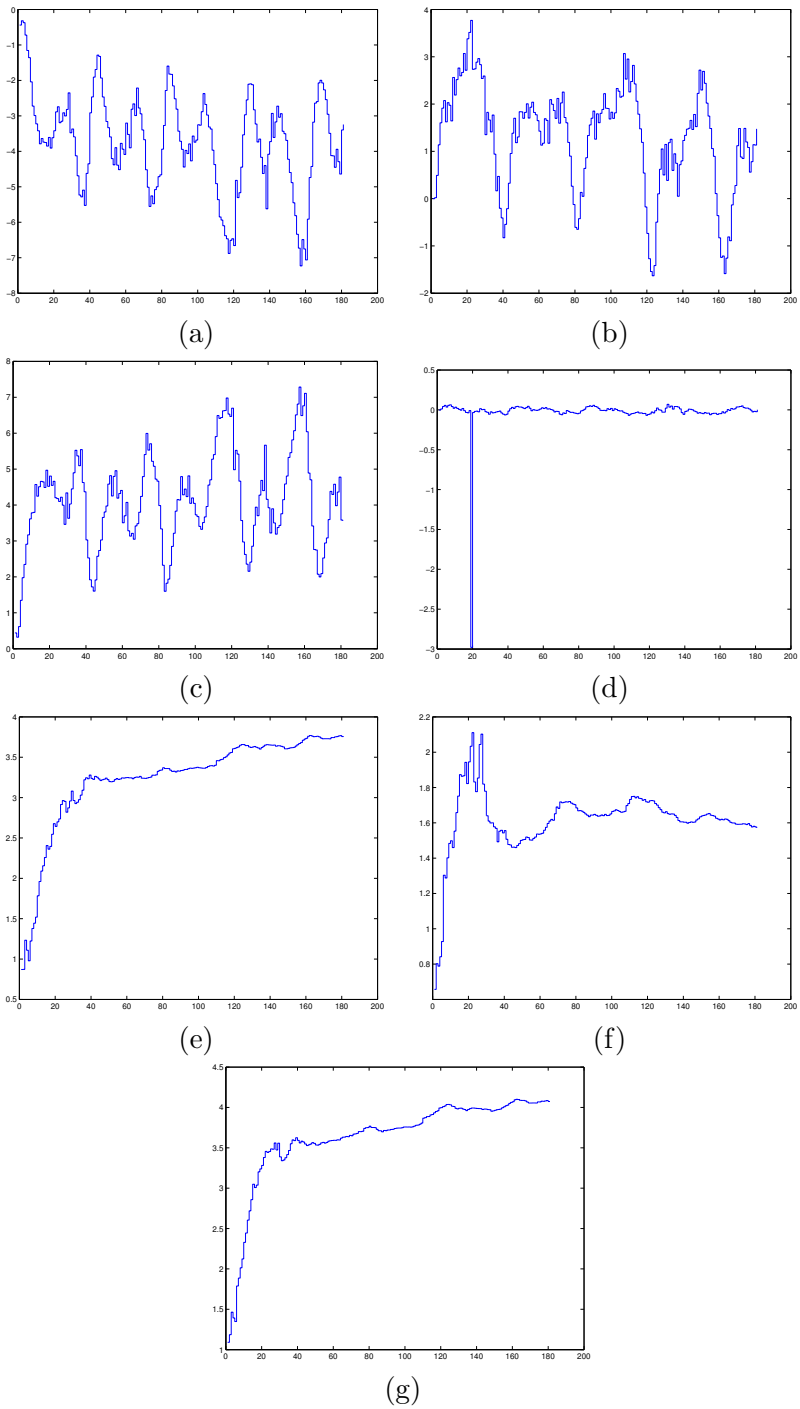


Figure 6.14: FastSLAM 1.0: error plots for circular path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

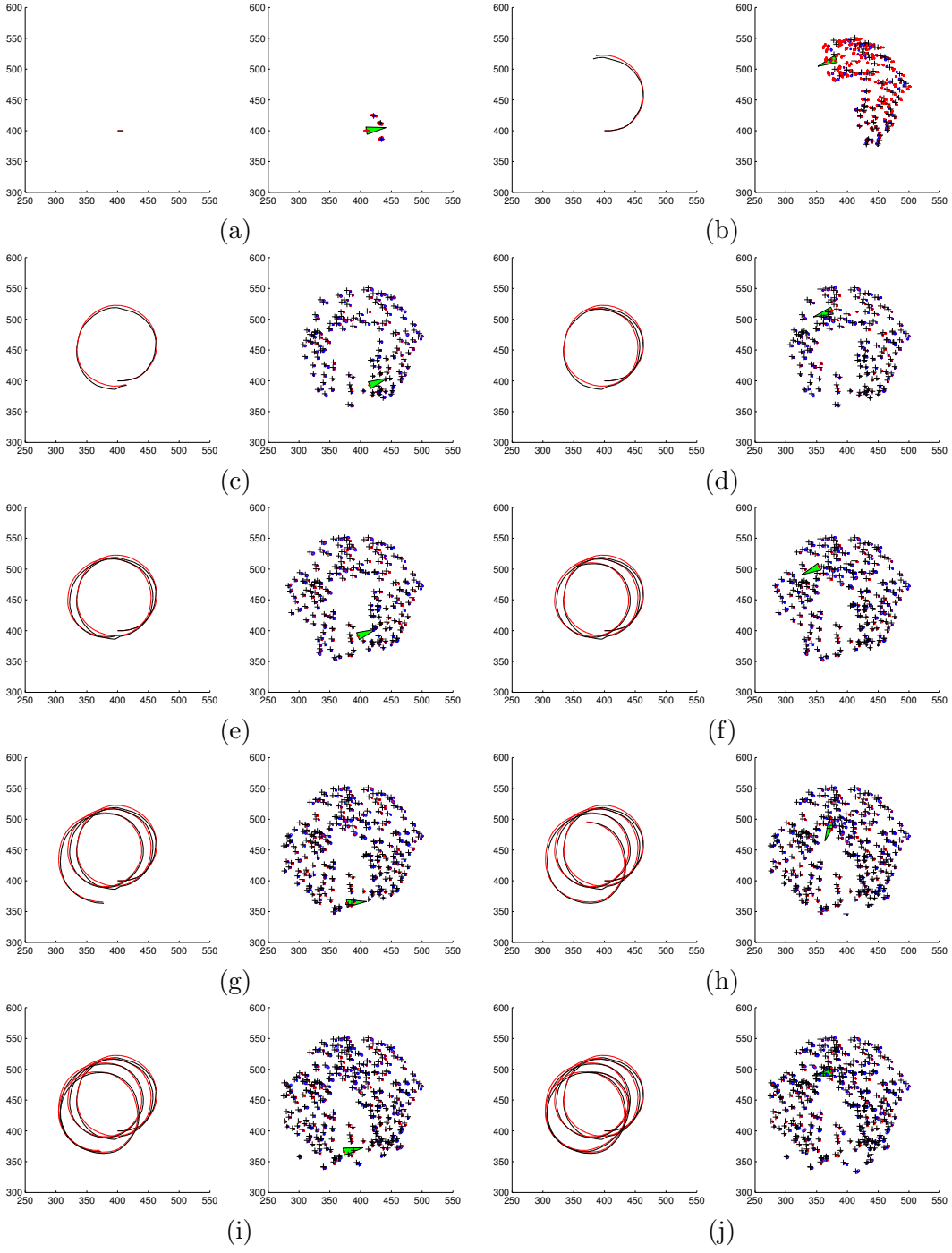


Figure 6.15: FastSLAM 2.0: sample run for a circular path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

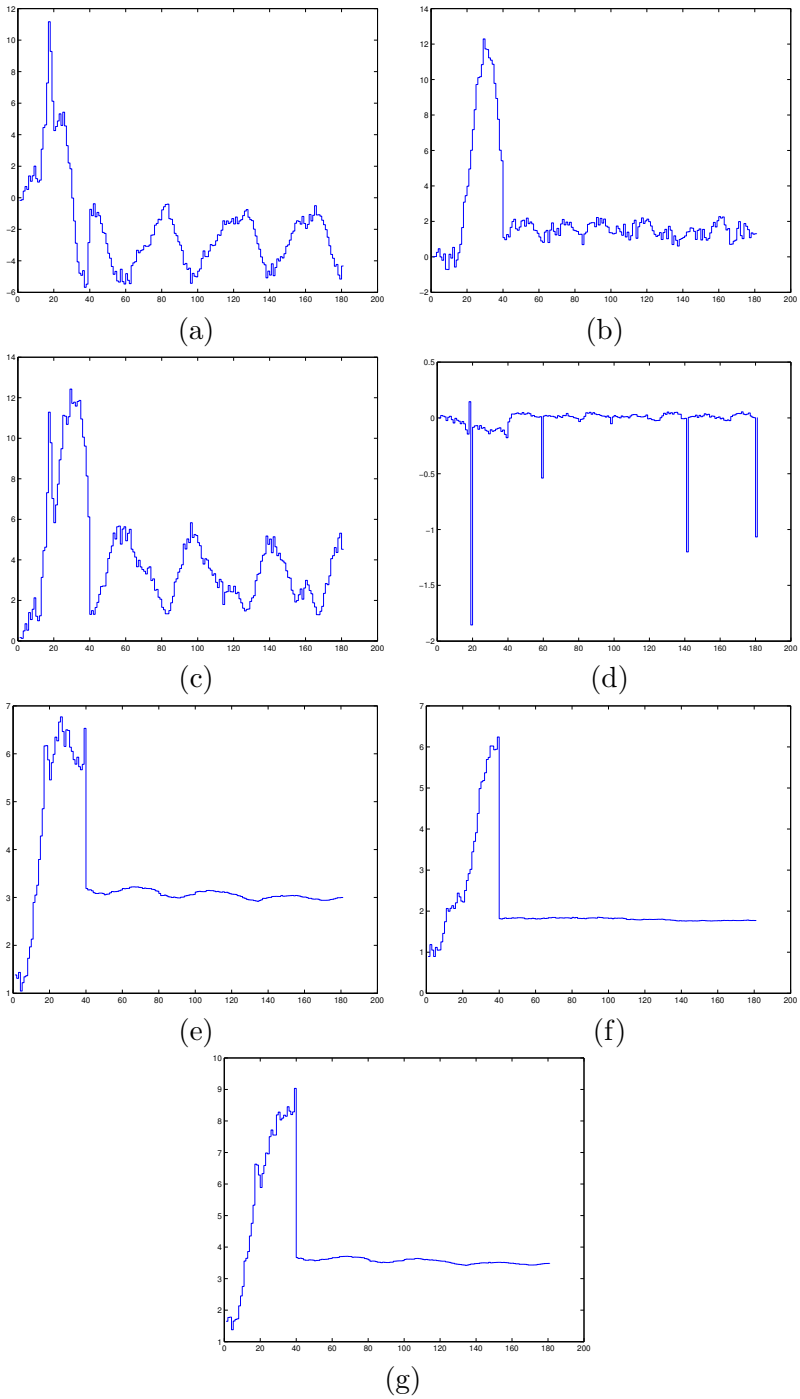


Figure 6.16: FastSLAM 2.0: error plots for circular path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

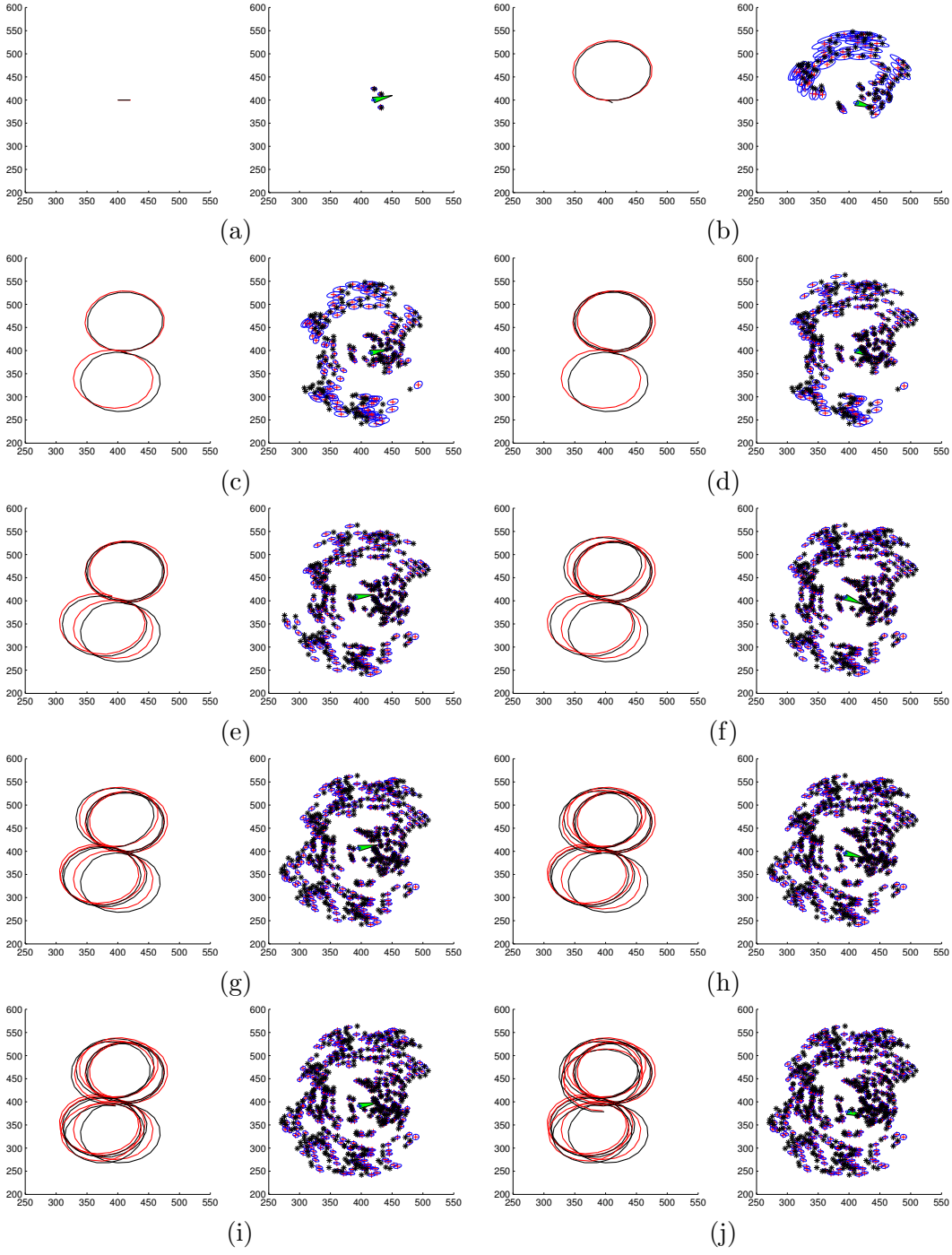


Figure 6.17: EKF-SLAM: sample run for an eight-shaped path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

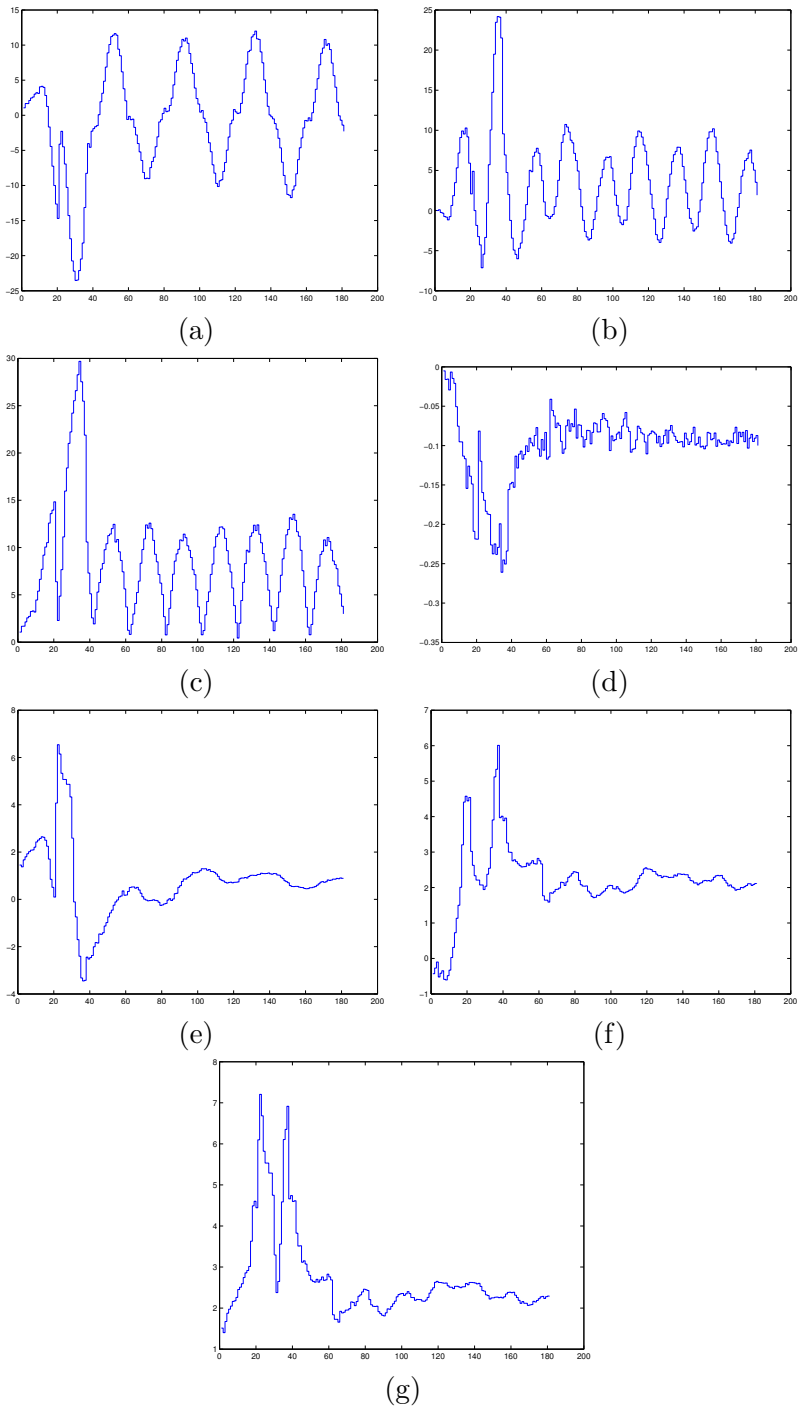


Figure 6.18: EKF-SLAM: error plots for the eight-shaped path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

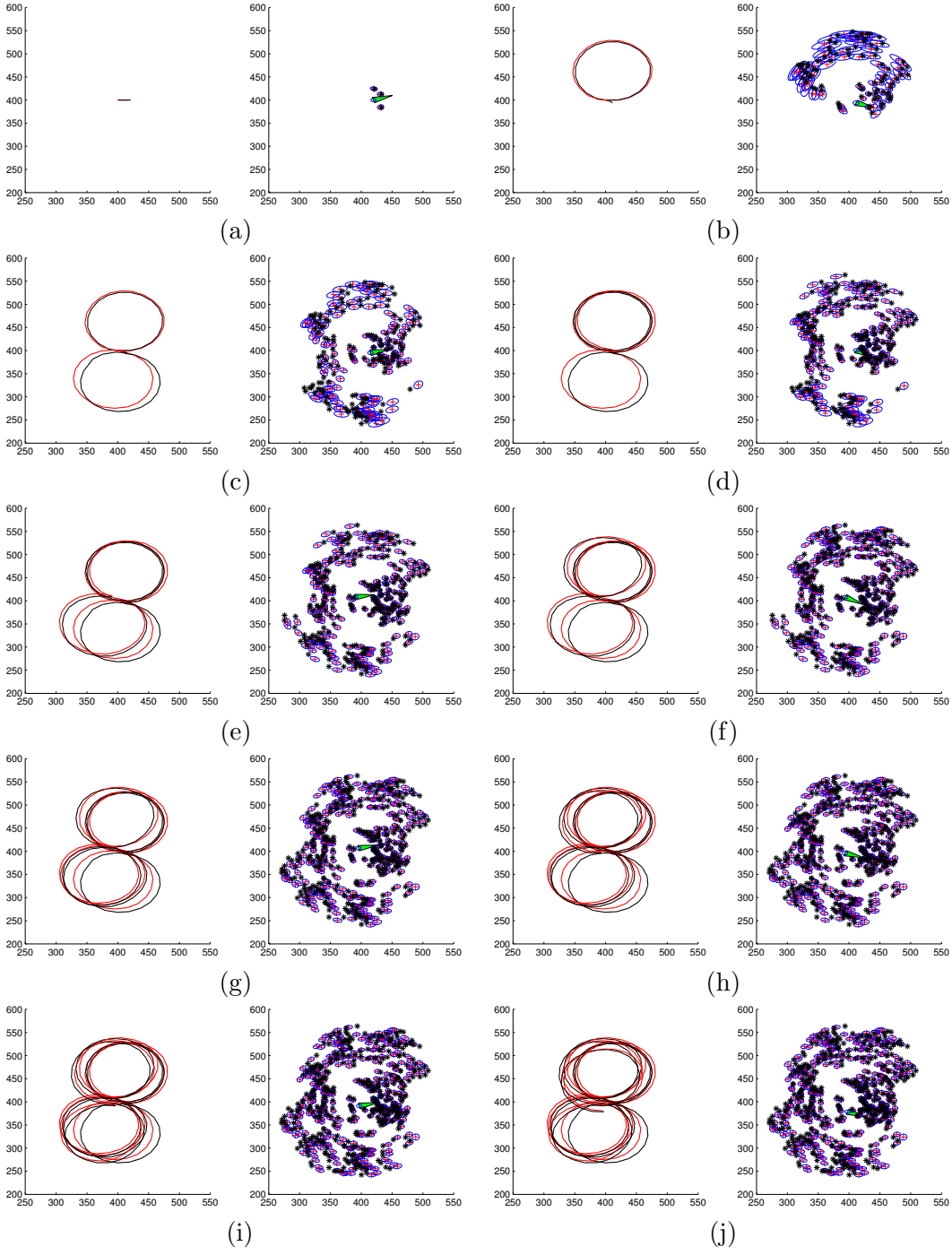


Figure 6.19: IF-SLAM: sample run for an eight-shaped path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

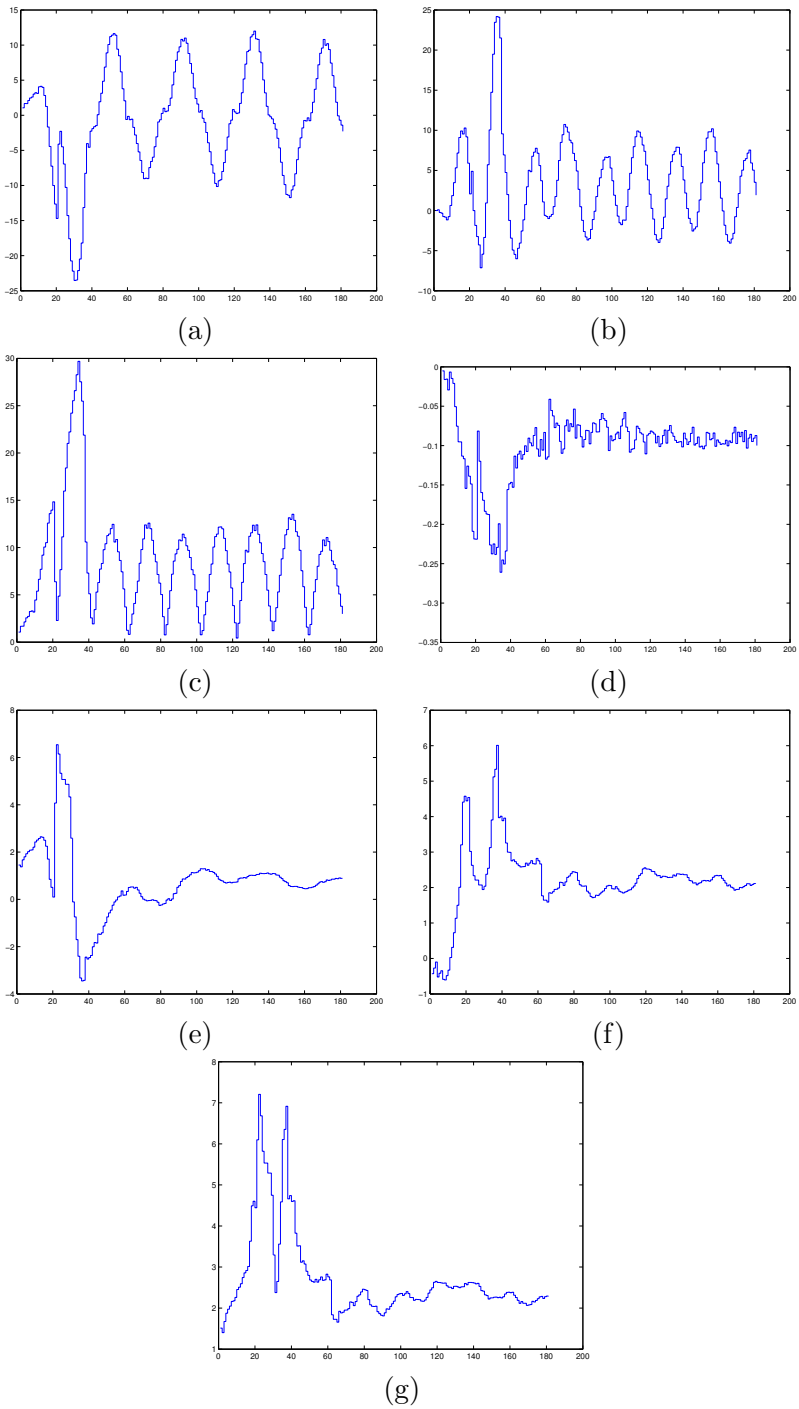


Figure 6.20: IF-SLAM: error plots for the eight-shaped path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

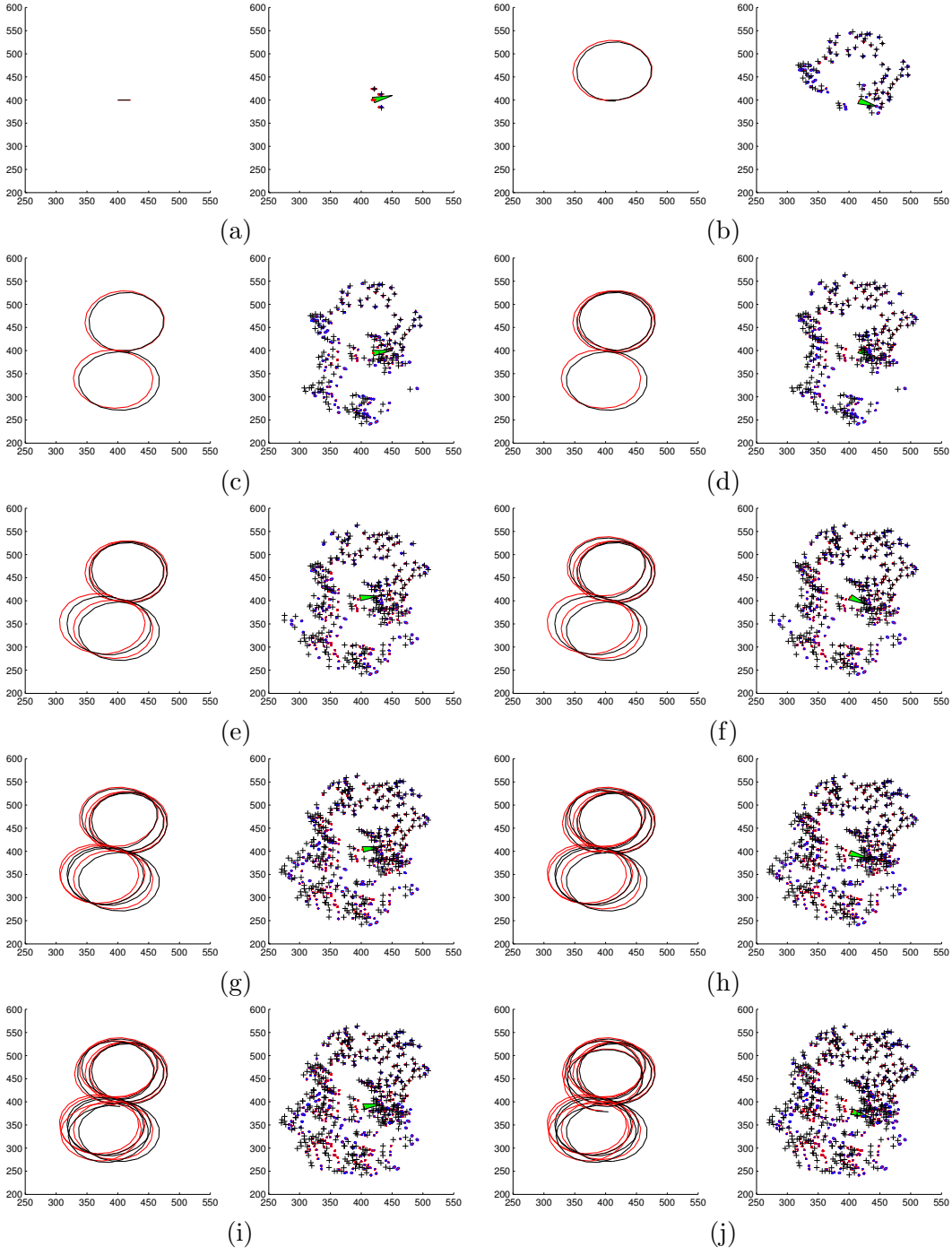


Figure 6.21: FastSLAM 1.0: sample run for an eight-shaped path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

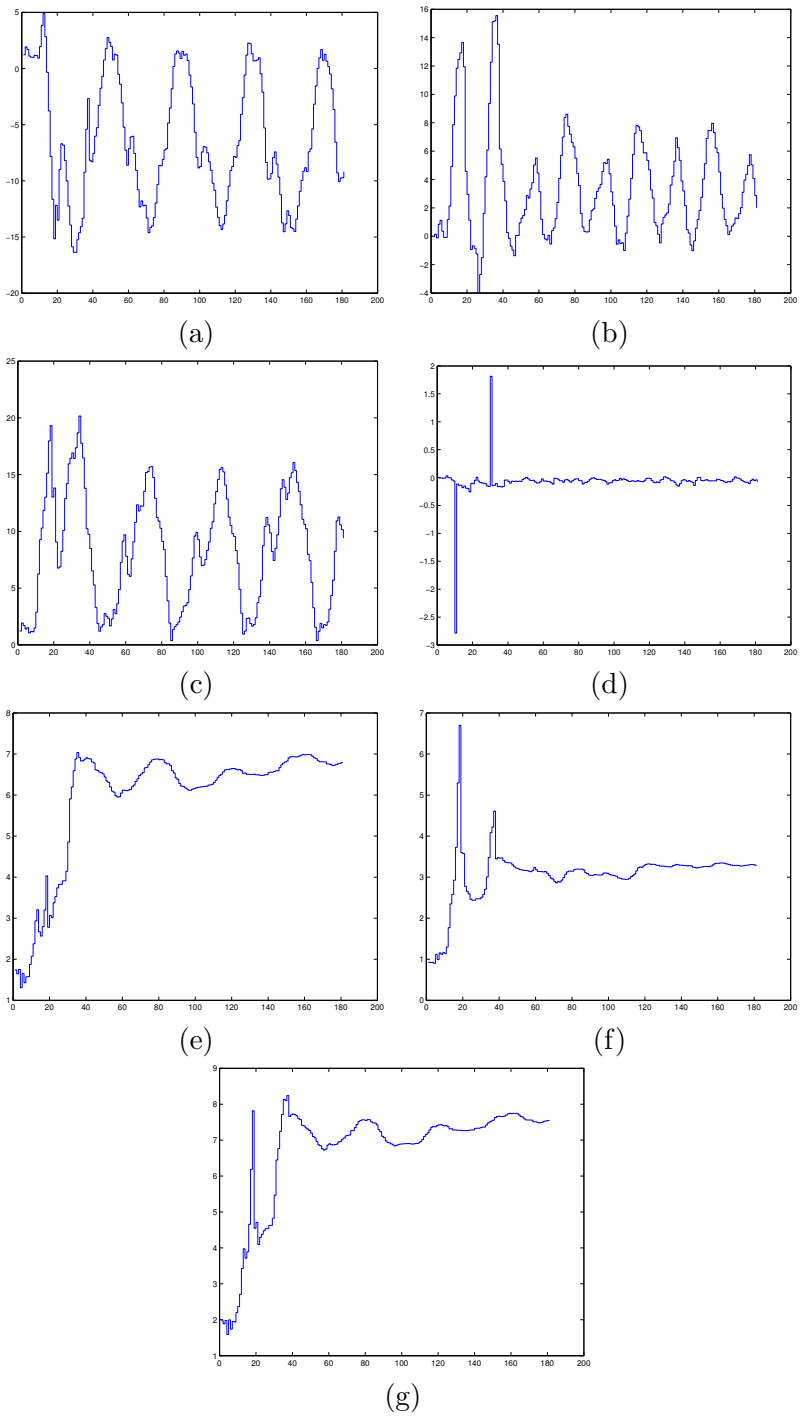


Figure 6.22: FastSLAM 1.0: error plots for the eight-shaped path. Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.

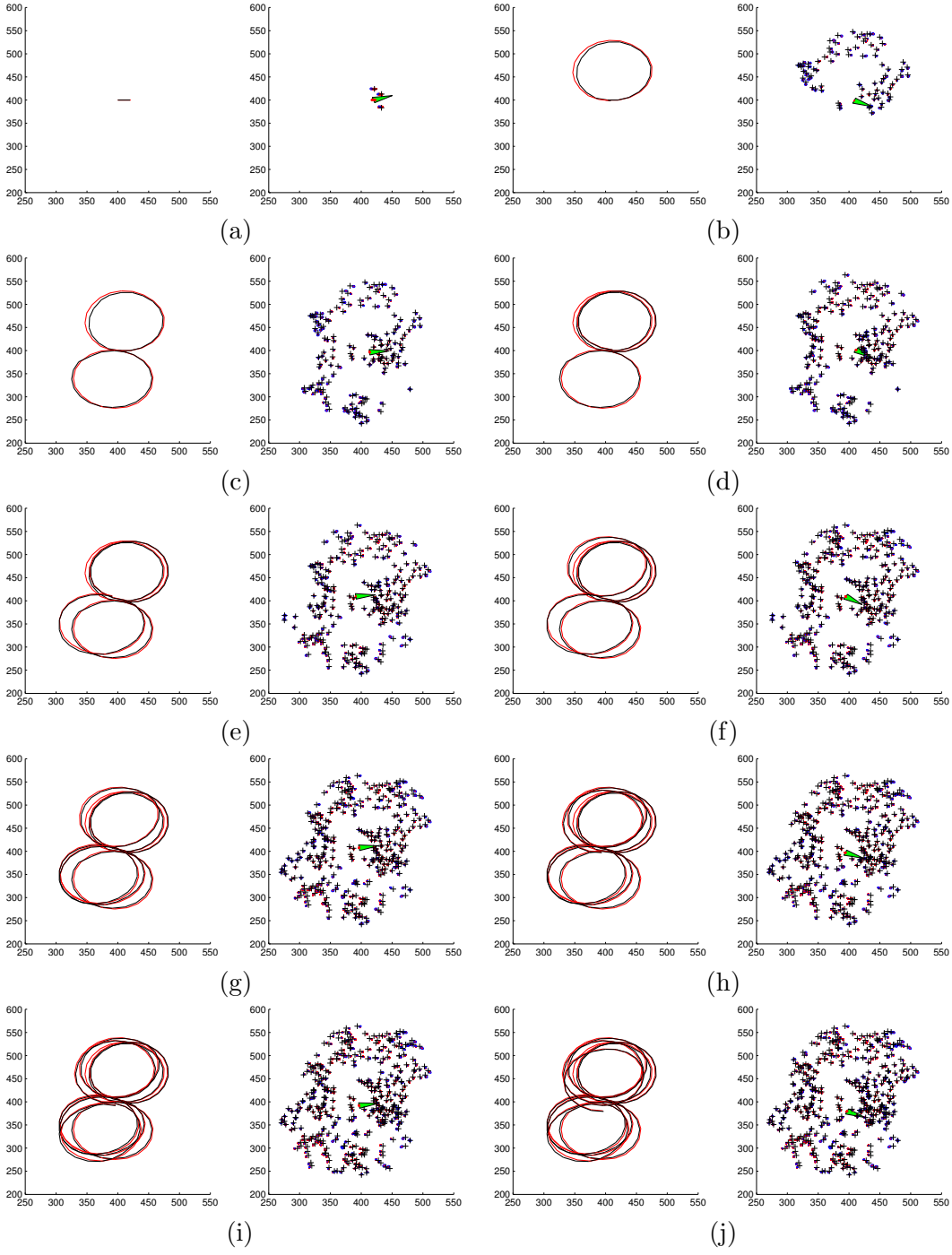


Figure 6.23: FastSLAM 2.0: sample run for an eight-shaped path. Results are shown at every 20th step, the figure on the right shows the UAV's real path and estimated path while the figure on the left shows the real and estimated landmark positions. The units are in pixels.

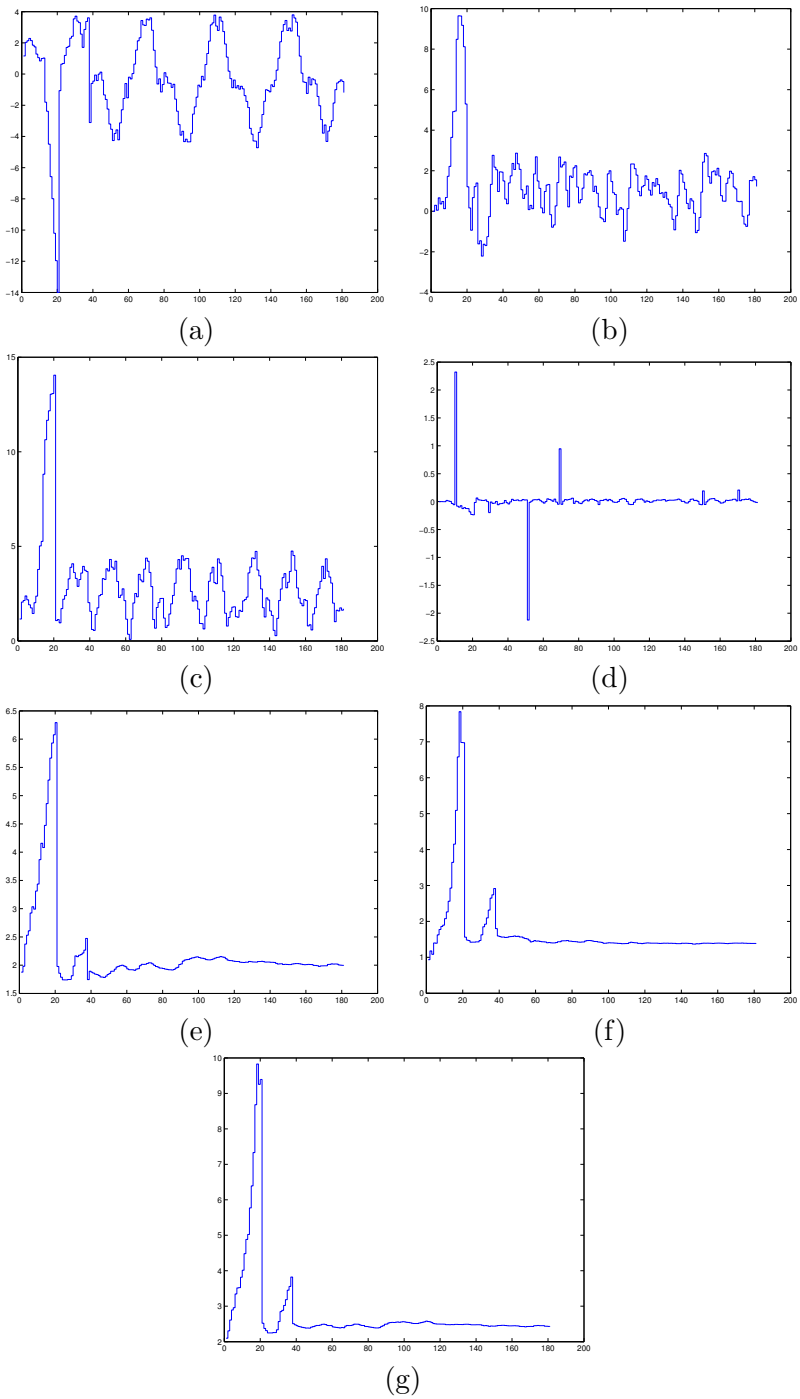
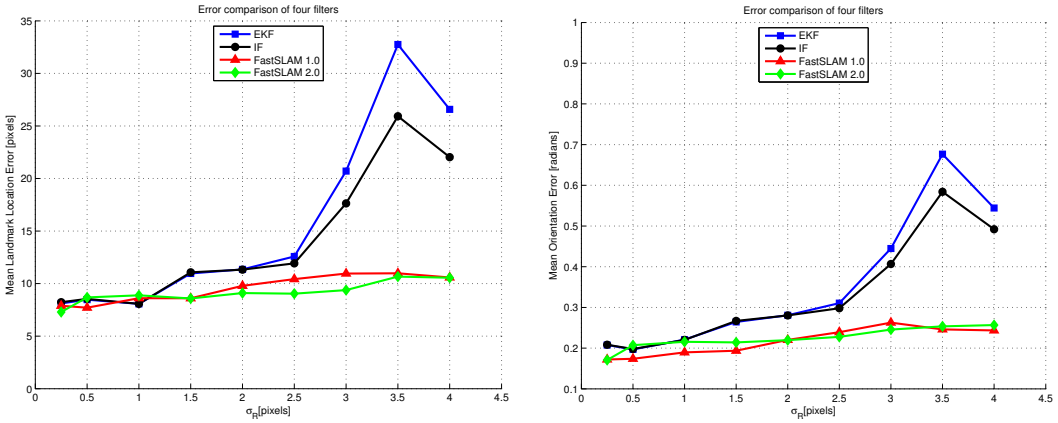
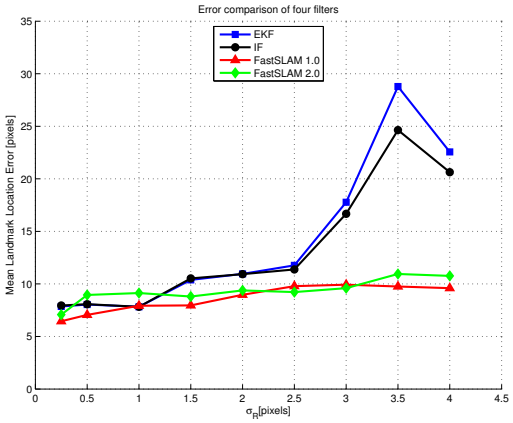


Figure 6.24: FastSLAM 2.0: Parts (a) and (b) give robot pose error in x and y coordinates in pixels, (c) gives the position error as an absolute distance in pixels, (d) shows the error in robot orientation in radians. Parts (e), (f) and (g) give average landmark errors in x and y coordinates, and as absolute distance in pixels respectively.



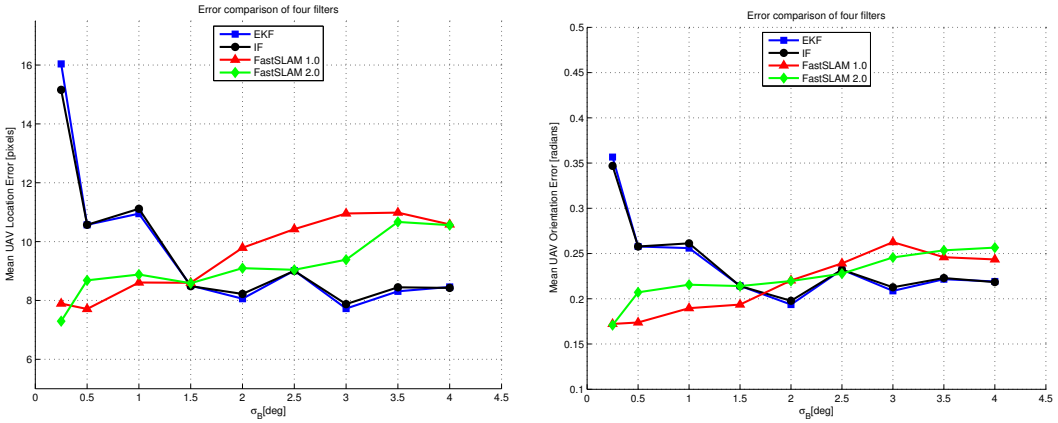
(a)

(b)



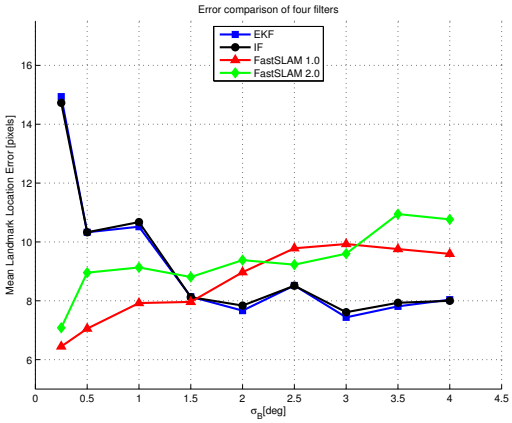
(c)

Figure 6.25: Comparison of error levels of the four SLAM algorithms EKF, IF, FastSLAM 1.0 and 2.0 with different range noise levels with bearing variance σ_B^2 fixed at $0.2^{\circ 2}$. Part (a) is the error in the estimation of the UAV location, (b) shows the error in estimation of UAV orientation and (c) shows the average landmark position error.



(a)

(b)



(c)

Figure 6.26: Comparison of error levels of the four SLAM algorithms EKF, IF, FastSLAM 1.0 and 2.0 with different bearing noise levels with range variance σ_R^2 fixed at 0.5m^2. Part (a) is the error in the estimation of the UAV location, (b) shows the error in estimation of UAV orientation and (c) shows the average landmark position error.

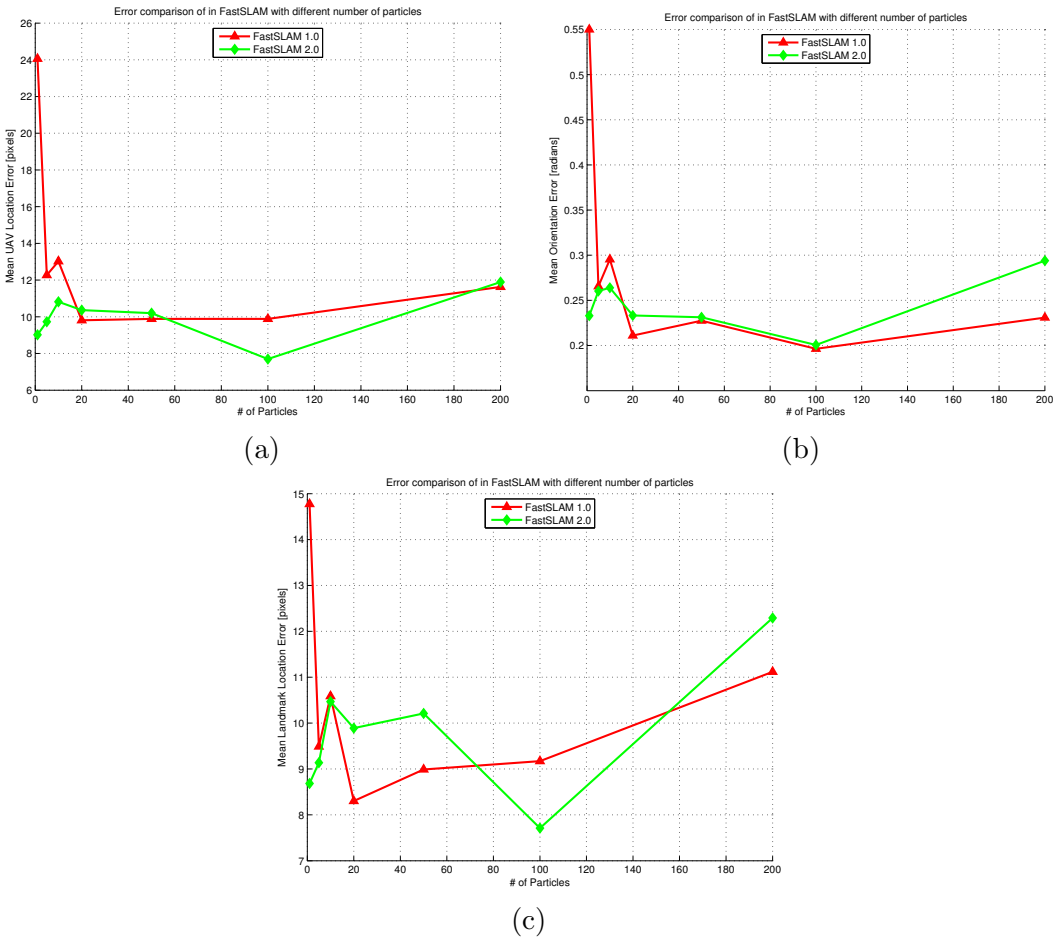


Figure 6.27: Comparison of error levels of the FastSLAM 1.0 and 2.0 algorithms with different number of particles. Part (a) is the error in the estimation of the UAV location, (b) shows the error in estimation of UAV orientation and (c) shows the average landmark position error.

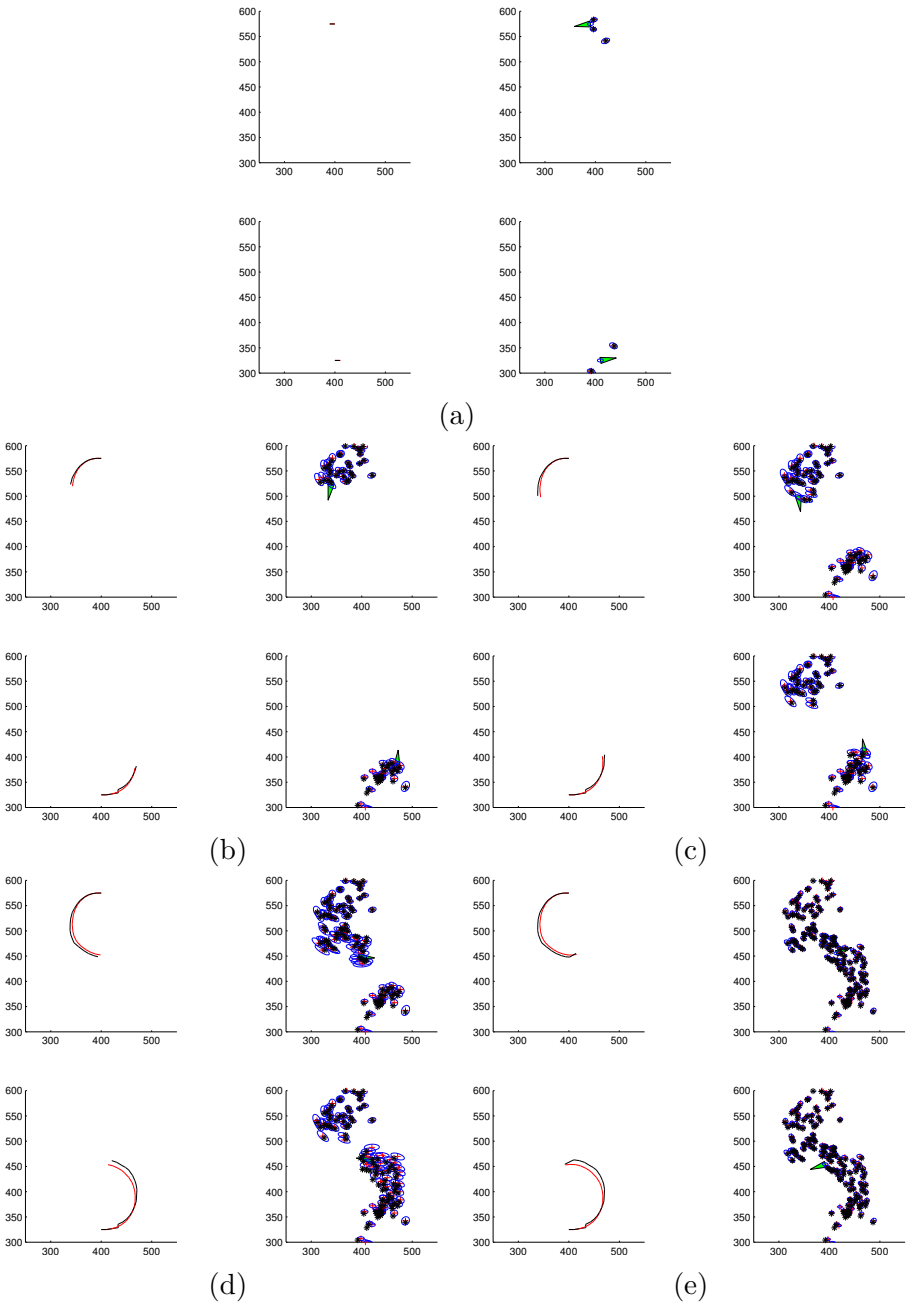


Figure 6.28: A sample run of multiple UAV SLAM with IF. Results are given at every 10th step, just before and after the communication occurs. In each part, the figure on the left shows the real and the estimated paths of each UAV, while the figure on the right shows the individual maps of the UAVs (continued).

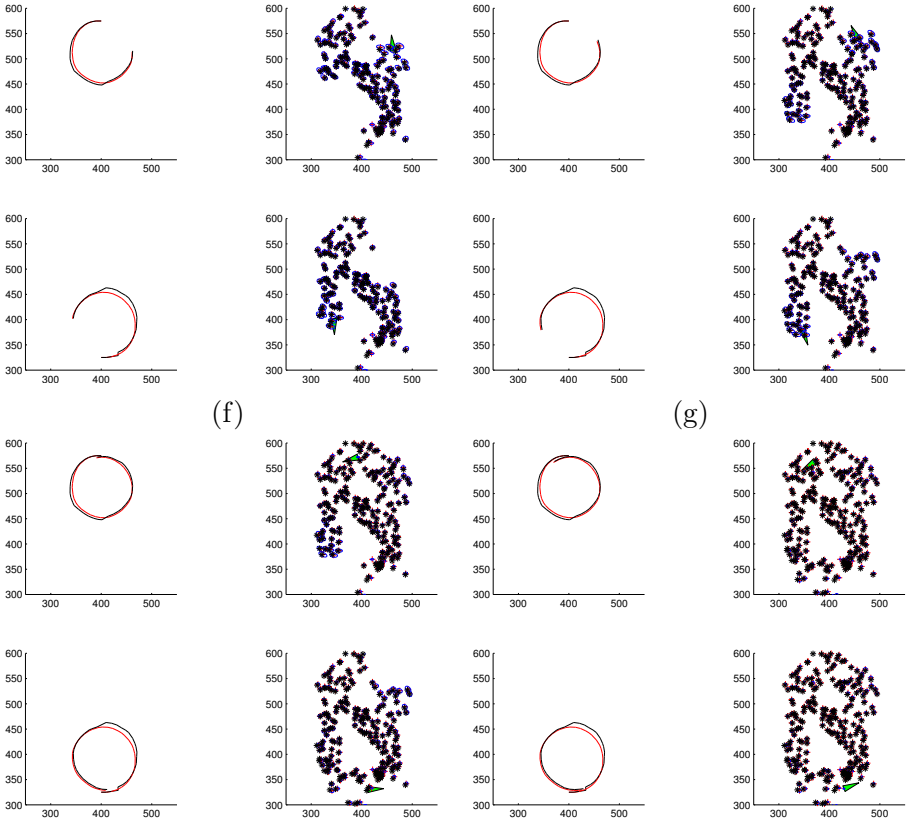


Figure 6.28: (continued) A sample run of multiple UAV SLAM with IF. Results are given at every 10th step, just before and after the communication occurs. In each part, the figure on the left shows the real and the estimated paths of each UAV, while the figure on the right shows the individual maps of the UAVs.

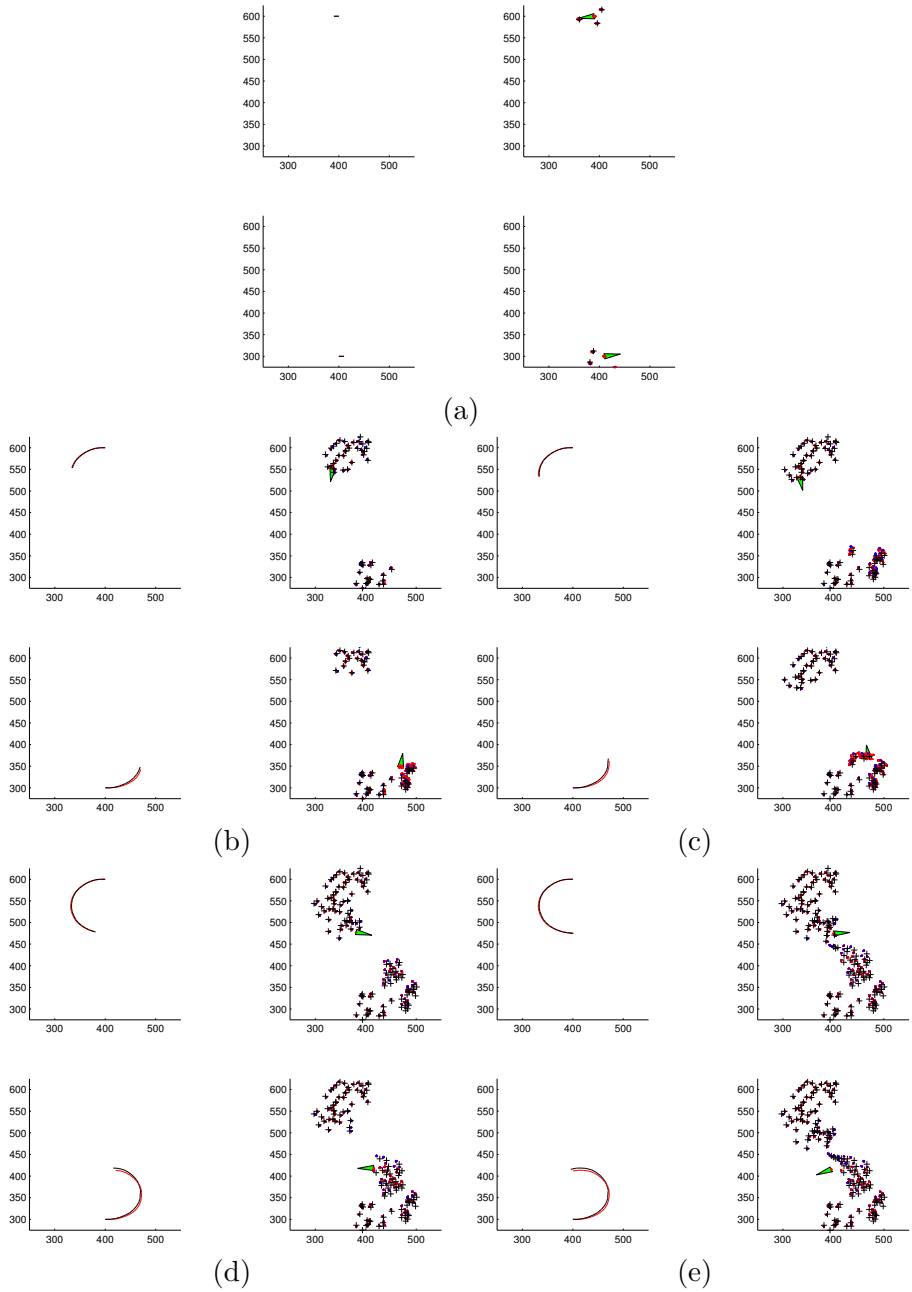


Figure 6.29: A sample run of multiple UAV SLAM with PF based FastSLAM. Results are given at every 10th step, just before and after the communication occurs. In each part, the figure on the left shows the real and the estimated paths of each UAV, while the figure on the right shows the individual maps of the UAVs (continued).

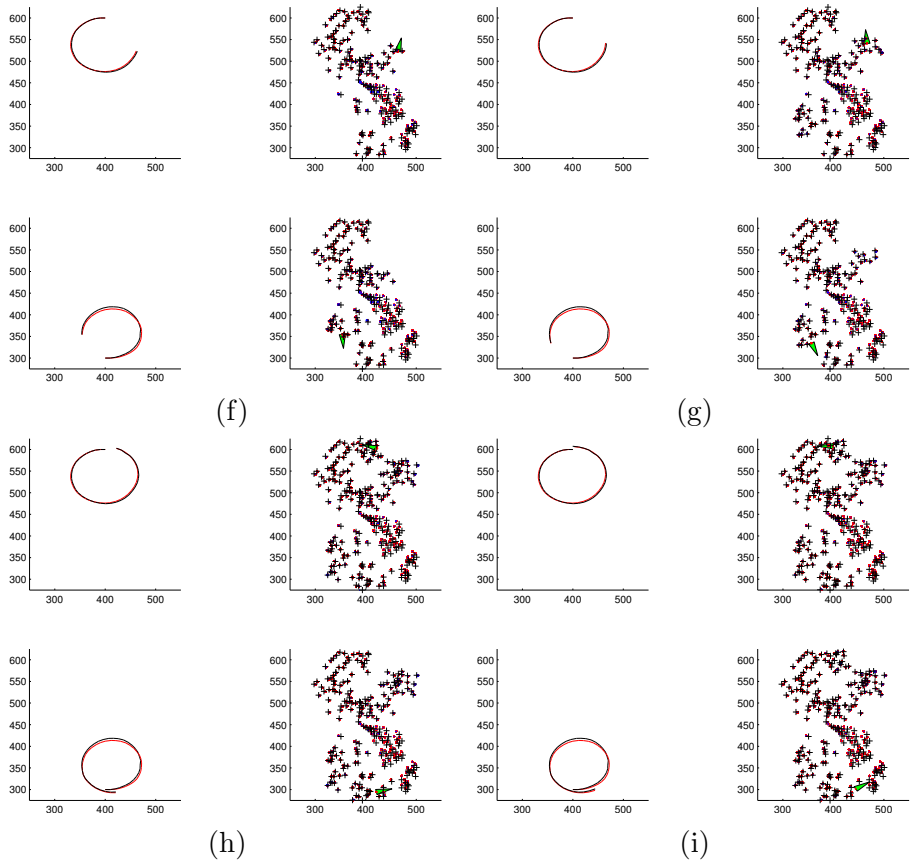


Figure 6.29: (continued) A sample run of multiple UAV SLAM with PF based FastSLAM. Results are given at every 10th step, just before and after the communication occurs. In each part, the figure on the left shows the real and the estimated paths of each UAV, while the figure on the right shows the individual maps of the UAVs.

Chapter 7

Conclusions and Future Work

In this thesis, we constructed a vision-based SLAM algorithm, using visual features of images acquired with a camera as landmarks. The visual features used are SIFT features obtained using the well-known SIFT algorithm. We decreased the number of these features by using a selection process over smaller sized images to obtain a smaller set of landmarks. With this experimental framework, we used four well known SLAM algorithms: EKF, IF, FastSLAM 1.0 and 2.0 and demonstrated their different performance characteristics. We also presented a multi-UAV SLAM application based on IF and PF.

Our algorithms are based on processing visual data which can be easily gathered with low-cost commercial equipment. Using visual data provides easy solutions to problems that appear inherently in the SLAM applications, such as landmark identification and data association. This important aspect constitutes one of the major contributions of our work. Using visual data not only provides a framework for the SLAM, but it also has the advantage of providing rich data for different purposes. In this thesis, we also demonstrated that visual data can be easily incorporated into a framework, to be used with readily available popular SLAM solutions. One other contribution of this thesis is the extension of our

framework to the multi-UAV case, which can provide robustness and efficiency in common UAV applications.

There are still shortcomings of our approach that needs to be addressed along with testing our work in a more realistic framework. The following improvements and extensions can be considered as future work:

- One step is to develop a better simulation environment to incorporate more realistic data.
- We also need to consider real aircraft dynamics and forgo the assumption of constant altitude to have a more practical and realistic implementation.
- A more systematic landmark selection scheme is needed to select useful features only which can reduce the computational costs of the filters drastically. Landmark deletion, i.e. deletion of less observed landmarks, can help in this aspect as well.
- Fusion of data from different sources such as different aircraft with different characteristics or different sensors can also increase efficiency and robustness.

Bibliography

[1] Angeli, A., Filliat, D. and Doncieux, S. and Meyer, J.-A., "2D simultaneous localization and mapping for micro aerial vehicles," in *Proceedings of the European Micro Aerial Vehicles Conference*, 2006.

[2] Bailey, T., "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments," Ph.D. dissertation, University of Sydney, 2002, http://www-personal.acfr.usyd.edu.au/tbailey/techreports/phdthesis.htm.

[3] ——, "Home Page of Tim Bailey," http://www-personal.acfr.usyd.edu.au/tbailey/, Aug 01, 2008.

[4] Bailey, T. and Durrant-Whyte, H. F., "Simultaneous localisation and mapping (SLAM): Part II state-of-the-art," *Robotics and Automation Magazine*, vol. 13, pp. 108–117, 2006.

[5] Bailey, T., Nieto, J., Guivant, J., Stevens, M. and Nebot, E., "Consistency of the EKF-SLAM Algorithm," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3562–3568, 2006.

[6] Ballesta, M., Gil, A., Reinoso, O. and Martinez-Mozos, O., *Evaluation of interest point detectors for visual SLAM*, 2nd ed. International SAR, January 2008, pp. 190–199.

[7] Bar-Shalom, Y. and Li, X. R., *Estimation and Tracking: Principles, Techniques, and Software.* Norwood, MA: Artech House, Inc, 1993.

[8] Bay, H., Tuytelaars, T. and Van Gool, L., "SURF: Speeded up robust features," *European Conference on Computer Vision*, vol. 1, pp. 404–417, 2006.

[9] Berg, T. M. and Durrant-Whyte, H. F., "Model distribution in decentralized multi-sensor data fusion," in *American Control Conference*, vol. 28, 1991.

[10] Burgard, W., Fox, D., Jans, H., Matenar, C. and Thrun, S., "Sonar-based mapping of large-scale mobile robot environments using EM," in *Sixteenth International Conference on Machine Learning*. San Francisco, CA, U.S.A.: Morgan Kaufmann Publishers Inc., 1999, pp. 67–76.

[11] Canny, J., "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[12] Carson, C., Belongie, S., Greenspan, H. and Malik, J., "Blobworld: image segmentation using expectation-maximization and its application to image querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1026–1038, 2002.

[13] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E. and Thrun S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[14] Csorba, M., "Simultaneous Localisation and Map Building," Ph.D. dissertation, University of Oxford, 1998, http://www.cas.edu.au/content.php/292.html?publicationid=157&displaypage=2.

[15] Csorba, M. and Durrant-Whyte, H. F., "A new approach to simultaneous localisation and map building," in *SPIE Aerosense*, vol. 2738, 1996, pp. 26–36.

[16] Davison, A. J. and Murray, D. W., "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[17] Davison, A. J., Reid, I. D., Molton, N.D. and Stasse, O., "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1052–1067, 2007.

[18] Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H. F. and Csorba, M., "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, June 2001.

[19] Doucet, A., de Freitas, J. F. G. and Gordon, N. J., *Sequential Monte Carlo Methods in Practice.* New York: Springer-Verlag, 2001.

[20] Doucet A,, K. M., de Freitas, N., Murphy, K. and Russell, S., "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford-California, 2000, pp. 176–183.

[21] Durrant-Whyte, H. F., "Uncertain geometry in robotics," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, February 1988.

[22] Durrant-Whyte, H. F. and Bailey, T., "Simultaneous localisation and mapping (SLAM): Part I The essential algorithms," *Robotics and Automation Magazine*, vol. 13, pp. 99–110, 2006.

[23] Eade, E. and Drummond, T., "Edge landmarks in monocular SLAM," *British Machine Vision Conference*, 2006.

[24] Elinas, P., Sim, R. and Little, J. J., "σSLAM: stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1564–1570, May 15-19, 2006.

[25] Estrada, C., Neira, J. and Tardos, J. D., "Hierarchical SLAM: real-time accurate mapping of large environments," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 4, pp. 588–596, 2005.

[26] Eustice, R. M., Singh, H. and Leonard, J. J., "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics and Automation*, vol. 22, no. 6, pp. 1100–1114, 2006.

[27] Eustice, R. M., Singh, H., Leonard, J. J., Walter, M. and Ballard, R., "Visually navigating the RMS titanic with SLAM information filters," in *Robotics: Science And Systems*. MIT Press, 2005.

[28] Gil, A., Reinoso, O., Martinez-Mozos, O., Stachniss, C. and Burgard, W., "Improving data association in vision-based SLAM," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[29] Google, "Google Earth," http://earth.google.com/, Aug 01, 2008.

[30] Grime, S., Durrant-Whyte, H. F. and Ho, P., "Communication in decentralized data-fusion systems," in *American Control Conference*, vol. 29, 1991, pp. 3299–3305.

[31] Guivant, J. and Nebot, E., "Improving computational and memory requirements of simultaneous localization and map building algorithms," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2731–2736, 2002.

[32] Guivant, J. E. and Nebot, E. M., "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[33] Harris, C. and Stephens, M., "A combined corner and edge detector," *Proceedings of The Fourth Alvey Vision Conference, Manchester*, vol. 15, pp. 189–192, 1988.

[34] Julier, S. J. and Uhlmann, J. K., "Using multiple SLAM algorithms," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 200–205, 2003.

[35] ——, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[36] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Transactions on ASME: Journal of Basic Engineering*, vol. 82D, pp. 35–45, March 1960.

[37] Kim, J. and Sukkarieh, S., "Autonomous airborne navigation in unknown terrain environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1031–1045, 2004.

[38] ——, "Real-time implementation of airborne inertial-SLAM," *Robotics and Autonomous Systems*, vol. 55, pp. 62–71, 2007.

[39] Knight, J., Davison, A. and Reid, I., "Towards constant time SLAM using postponement," in *RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2001, pp. 405–413.

[40] Leonard, J. J. and Newman, P., "Consistent, convergent, and constant-time SLAM," in *International Joint Conference on Artificial Intelligence*, vol. 18, 2003, pp. 1143–1150.

[41] Lindeberg, T., "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–154, 1998.

[42] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[43] ——, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[44] Manyika, J. M. and Durrant-Whyte, H. F., "On sensor management in decentralized data fusion," in *31st IEEE Conference on Decision and Control*, 1992, pp. 3506–3507.

[45] Martinelli, A., Nguyen, V., Tomatis, N. and Siegwart, R., "A relative map approach to SLAM based on shift and rotation invariants," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 50–61, 2007.

[46] Martinez-Mozos, O., Gil, A., Ballesta, M. and Reinoso, O., *Interest point detectors for visual SLAM*, ser. Current Topics in Artificial Intelligence. Springer-Verlag, 2008, vol. 4788, pp. 170–179.

[47] Maybeck, P. S., *Stochastic Models, Estimation, and Control*. New York, NY: Academic Press, 1979, vol. I,II,III.

[48] Montemerlo, M., "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association," Ph.D. dissertation, Carnegie Mellon University, 2003.

[49] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B., "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 1156. Menlo Park, CA, USA: AAAI Press, 2003, pp. 1151–1156.

[50] Moutarlier, P. and Chatila, R., "Stochastic multisensory data fusion for mobile robot location and environment modelling," *5th International Symposium on Robotics Research*, vol. 1, pp. 85–94, 1989.

[51] Murphy, K., "Bayesian map learning in dynamic environments," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1015–1021, 2000.

[52] Murphy, K. and Russell, S., "Rao-blackwellized particle filtering for dynamic Bayesian networks," *Sequential Monte Carlo Methods in Practice.*, pp. 499–515, 2001.

[53] Mutambara, G. O., *Decentralized Estimation and Control for Multisensor Systems.* 2000 Corporate Blvd., N.W., Boca Raton, Florida: CRC press LLC, 1998.

[54] Newman, P. and Ho, K., "SLAM-loop closing with visually salient features," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 635–642, 2005.

[55] Newman, P., Cole, D. and Ho, K., "Outdoor SLAM using visual appearance and laser ranging," *IEEE Transactions on Robotics and Automation*, pp. 1180–1187, 2006.

[56] Nguyen, V. and Siegwart, R., "Information relative map going toward constant time SLAM," in *European Robotics Symposium 2008 Springer Tracts in Advanced Robotics*, vol. 44. Berlin-Heidelberg: Springer-Verlag, 2008, pp. 134–144.

[57] Rao, B. S. and Durrant-Whyte, H. F., "Fully decentralised algorithm for multisensor Kalman filtering," *Control Theory and Applications, IEE Proceedings D [see also IEE Proceedings-Control Theory and Applications]*, vol. 138, no. 5, pp. 413–420, 1991.

[58] Se, S., Lowe, D. G. and Little, J. J., "Vision-based global localization and mapping for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 3, pp. 364–375, 2005.

[59] Smith, R. and Cheeseman, P., "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[60] Smith, R., Self, M. and Cheeseman, P., "A stochastic map for uncertain spatial relationships," *The Fourth International Symposium of Robotics Research*, pp. 467–474, 1987.

[61] ——, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, vol. 1, pp. 167–193, 1990.

[62] Smith, S. M., "A new class of corner finder," in *3rd British Machine Vision Conference*. University of Leeds, 1992, pp. 139–148.

[63] Sukkarieh, S., Nettleton, E., Kim, J. H., Ridley, M., Goktogan, A. and Durrant-Whyte, H. F., "The ANSER Project: data fusion across multiple uninhabited air vehicles," *International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 505–540, 2003.

[64] Tardos, J., Neira, J., Newman, P. and Leonard, J. J., "Robust mapping and localization in indoor environments using SONAR data," *International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[65] Thrun, S., "Robotic mapping: a survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. San Francisco, CA, USA: Morgan Kaufmann, 2002.

[66] Thrun, S., Burgard, W. and Fox, D., "A probabilistic approach to concurrent mapping and localization for mobile robots," *Autonomous Robots*, vol. 5, no. 3, pp. 253–271, 1998.

[67] Thrun, S., Fox, D. and Burgard, W., "Monte Carlo localization with mixture proposal distribution," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000, pp. 859–865.

[68] Thrun, S., Koller, D., Ghahramani, Z. and Durrant-Whyte, H. F. and Ng, A. Y., "Simultaneous mapping and localization with sparse extended information filters: theory and initial results," *Algorithmic Foundations of Robotics V*, 2003.

[69] Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z. and Durrant-Whyte, H. F., "Simultaneous localization and mapping with sparse extended

information filters," *International Journal of Robotics Research*, vol. 23, no. 7–8, pp. 693–716, 2004.

[70] Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J. and Nebot, E., "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 380–407, 2004.

[71] Vercauteren, T. and Wang, X., "Decentralized sigma-point information filters for target tracking in collaborative sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, Part 2, pp. 2997–3009, 2005.

[72] Walter, M. R., Eustice, R. M. and Leonard, J. J., "Exactly sparse extended information filters for feature-based SLAM," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[73] Wang, Z., Huang, S. and Dissanayake, G., "D-SLAM: A decoupled solution to simultaneous localization and mapping," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 187–204, 2007.

[74] Williams, S. B., "Efficient Solutions to Autonomous Mapping and Navigation Problems," Ph.D. dissertation, The University of Sydney, 2001, http://www.cas.edu.au/content.php/292.html?publicationid=169&display-page=1.

[75] Zhang, H., "Quantitative evaluation of feature extractors for visual SLAM," in *Fourth Canadian Conference on Computer and Robot Vision*. IEEE Computer Society Washington, DC, USA, 2007, pp. 157–164.