Visual-Inertial Sensor Fusion Models and Algorithms for Context-Aware Indoor Navigation

by

Pedram Gharani

Bachelor of Science, University of Isfahan, 2004

Master of Science, University of Tehran, 2008

Master of Arts, The University of Iowa, 2014

Submitted to the Graduate Faculty of

School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Pedram Gharani

It was defended on

July 31, 2019

and approved by

Michael Lewis, Professor, Department of Informatics and Networked Systems

Paul Munro, Associate Professor, Department of Informatics and Networked Systems

Ervin Sejdic, Associate Professor, Bioengineering Department

Thesis Advisor/Dissertation Director: Hassan Karimi, Professor, Department of Informatics and Networked Systems

Visual-Inertial Sensor Fusion Models and Algorithms for Context-Aware Indoor Navigation

Pedram Gharani

University of Pittsburgh, 2019

Abstract

Positioning in navigation systems is predominantly performed by Global Navigation Satellite Systems (GNSSs). However, while GNSS-enabled devices have become commonplace for outdoor navigation, their use for indoor navigation is hindered due to GNSS signal degradation or blockage. For this, development of alternative positioning approaches and techniques for navigation systems is an ongoing research topic. In this dissertation, I present a new approach and address three major navigational problems: indoor positioning, obstacle detection, and keyframe detection. The proposed approach utilizes inertial and visual sensors available on smartphones and are focused on developing: a framework for monocular visual internal odometry (VIO) to position human/object using sensor fusion and deep learning in tandem; an unsupervised algorithm to detect obstacles using sequence of visual data; and a supervised context-aware keyframe detection.

The underlying technique for monocular VIO is a recurrent convolutional neural network for computing six-degree-of-freedom (6DoF) in an end-to-end fashion and an extended Kalman filter module for fine-tuning the scale parameter based on inertial observations and managing errors. I compare the results of my featureless technique with the results of conventional feature-based VIO techniques and manually-scaled results. The comparison results show that while the framework is more effective compared to featureless method and that the accuracy is improved, the accuracy of feature-based method still outperforms the proposed approach.

The approach for obstacle detection is based on processing two consecutive images to detect obstacles. Conducting experiments and comparing the results of my approach with the results of two other widely used algorithms show that my algorithm performs better; 82% precision compared with 69%. In order to determine the decent frame-rate extraction from video stream, I analyzed movement patterns of camera and inferred the context of the user to generate a model associating movement anomaly with proper frames-rate extraction. The output of this model was utilized for determining the rate of keyframe extraction in visual odometry (VO). I defined and computed the effective frames for VO and experimented with and used this approach for context-aware keyframe detection. The results show that the number of frames, using inertial data to infer the decent frames, is decreased.

Table of Contents

List of Tables

List of Figures

Preface

First and foremost, I would like to express my deepest appreciation to my advisor Dr. Hassan Karimi, because of his dedicated help, advice, inspiration, encouragement, and continuous support throughout my Ph.D. His enthusiasm, integral view on research and his mission for providing high-quality work has made a deep impression on me. During our course of interaction in the last five years, I have learned extensively from him, including how to raise new possibilities, how to regard an old question from a new perspective, and how to approach a problem by systematic thinking. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. It has been an honor to be associated with a person like Dr. Hassan Karimi.

I would also like to extend my deepest gratitude to my dissertation committee: Dr. Michael Lewis, Dr. Paul Munro, and Dr. Ervin Sejdic, for their insightful comments and encouragement which incented me to widen my research from various perspectives.

I am also grateful to Dr. Brian Suffoletto for his invaluable insight into novel research problems and helpful advice in our collaborations throughout my Ph.D.

My special words of thanks should go to Dr. Dara Mendez for her relentless support while working with her research group.

Lastly, but by no means least, I would like to thank my family for all their love and encouragement. I am deeply indebted to my mother for raising me with love of science and supported me in all my pursuits. My mother is the most important person in my life, and I dedicate this dissertation to her.

1.0 Introduction

While data from various sources is omnipresent in most scientific and commercial domains, it is necessary to integrate diverse data sources to obtain useful, reliable, and consistent outcomes. Navigation systems, as real-time information systems, can perform navigational tasks by combining various types of data, specifically positioning which is predominately dependent on Global Navigation Satellite Systems (GNSSs), such as GPS. While GNSS-enabled devices are ubiquitous and widely used, GNSS is not a reliable and robust option for positioning inside buildings where the signal is either degraded or blocked. In the presence of poor or blocked GNSS signal in indoor environments, other types of sensors for indoor positioning are considered.

Of possible sensors for indoor positioning, inertial sensors and cameras are widely used on mobile platforms. Inertial data from accelerometers and gyroscopes and images from visual sensors can be used for computing relative displacement and rotation. Inertial data can also reveal the high-level context of the platform, such as activity or gait change, and can enhance other related tasks, such as keyframe detection. Hence, fusing data from inertial and visual sensors can be used not only as a replacement for GNSS, but also for improving the quality of solutions for navigational tasks. It is worth mentioning that this fusion can even be used for improving positioning by GNSS in outdoor environments in places where GNSS signal is either degraded or blocked.

Dealing with the challenges related to navigation has become increasingly important in recent years and they have received considerable attention in the research community, such as in the areas of robotics, human navigation, and location-based services (LBSs). Improving the accuracy and reliability of solutions to such problems as simultaneous localization and mapping (SLAM), odometry, collision avoidance, tracking, autonomous navigation, positioning, wayfinding, and movement pattern analysis in different contexts is still a critical research topic. To meet the objectives of navigation systems, several challenging navigational tasks must be developed. These tasks include positioning and localization, path planning, representation, interaction, wayfinding, SLAM, tracking, collision avoidance, and movement pattern analysis, among others. The navigation environment within which a moving platform moves through plays a vital role in determining how to tackle these challenges.

In the absence of GNSSs, e.g., GPS, commonly used technologies for indoor positioning can be categorized into two main groups: radio-frequency (RF)-based and non-radio frequency (RF)-based positioning methods (Yassin & Rachid, 2015). RF-based positioning methods include Wi-Fi-based positioning, cellular-based positioning, and Bluetooth-based positioning, among others. Non-RF-based positioning methods include signage and maps positioning, inertial navigation, visual-inertial odometry, and acoustic positioning (Karimi, 2015). Currently, microelectromechanical systems (MEMS) sensors are widely available on various devices and navigation systems, where they can be directly used for inertial positioning. However, due to the data integration process in these systems, the errors of the positions derived from MEMS grow quickly as time passes. To overcome this problem, other sensors like visual sensors are considered, resulting in accurate positioning solutions for indoor navigation. Fusing data from visual and inertial sensors is potential for producing enhanced positioning and localization solutions due to the availability and use of contexts. This thesis is focused on a non-RF-based method for positioning indoors by using a visual-inertial odometry. Odometry is a method for estimating change in position over time using data from motion sensors. Motion sensor data for odometry usually provides translation and rotation, and if the values of translation and rotation come from captured visual data, it is called visual odometry.

Visual odometry is a positioning and localization technique that has been used in navigation systems. Its application varies depending on the type of moving objects, including human mobility, movement aid for visually impaired people, self-driving cars, unmanned aerial vehicles (UAVs), automated underwater vehicles (AUVs), and tracking instruments in surgical operations, among others. As one of the positioning techniques, visual odometry's objective is to estimate six degrees of freedom (6DoF) as camera poses from captured frames of a trajectory while traveling through an unknown environment. Moving within an unknown environment implies that a feature map is not available in advance, as in map-based localization methods (Trawny & Roumeliotis, 2005; Wu, Johnson, & Proctor, 2005). With regard to such maps, it is important to mention that the goal of this research is not to build such a map, rather the goal is to reconstruct the trajectory.

In monocular visual odometry, where a single camera is used, computing displacement is more challenging compared to using stereo cameras. Stereo cameras have a known baseline and scale is a known parameter, while in monocular imaging, scale remains an ambiguous parameter. Accordingly, in monocular odometry, it is hard to obtain the absolute scale of the trajectory, based only on visual data and using a sequence of frames. One approach to solve the scale problem is to use other available sources, such as MEMS sensors in an inertial measurement unit (IMU), as input data to the process. Fusing inertial data with visual data to solve the scale ambiguity problem in an efficient way is expected to lead to an accurate trajectory.

Obstacle detection is another challenging navigational task, which is critical for any moving entity such as robots or visually impaired people to avoid collisions with obstacles on their way. There are many approaches and techniques for obstacle detection, such as using ranging sensors. However, in this research widely available non-RF sensors on mobile devices for positioning and localization are used for object detection as well. For this, the captured visual data of the scene, while moving through an unfamiliar indoor environment, is processed to detect obstacles. In other words, obstacles are recognized and detected as areas on the visually captured scene that potentially hinder safe movement. The data processing consists of extracting a subset of all the frames from the video stream, selecting a few significant points on the frames, labeling the selected points as foreground or background, and clustering the foreground points that belong to either close or moving objects.

The last part of this research is focused on context awareness to improve the quality of positioning and obstacle detection solutions. Context generally indicates a set of environmental settings and user's states that determines the behavior of a moving object, and which can be categorized into computing context, user context, physical context, and time context. This research focuses on user context, which considers the impact of user activity as a more efficient solution and presents related knowledge to the user while moving. User context is sensed through inertial data and is used for keyframe detection. Analyzing movement pattern provides contextual data for frame detection and gathers other knowledge about user behavior that can be useful for further processing.

Keyframe detection is an essential part of the visual odometry and obstacle detection approach where context can be used as a mean to determine an efficient rate for extracting frames. In other words, as the standard frame rate of the video stream provides redundant data when detecting obstacles, we need to efficiently set the frame rate to avoid extra computation. One straightforward method is to use a constant rate, but it comes at the cost of computational overhead. A more sophisticated and efficient method is to determine varying rates based on a user's activities. Data from inertial sensors such as accelerometers, gyroscopes, and magnetometers has been used in other work (Ren & Karimi, 2013; Saeedi, Moussa, & El-Sheimy, 2014) to detect the state of a user's activities. Also, inertial data can be used to detect the states of moving platforms, which can be interpreted based on the association of inertial data with measured target behavior, such as blood alcohol level (Gharani, Suffoletto, Chung, & Karimi, 2017; Suffoletto, Gharani, Chung, & Karimi, 2018), where data fusion techniques are used for the analysis.

1.1 Problem statement

The proposed research tackles three challenging navigational tasks and proposes a new framework and a few algorithms for solving them. The main focus is on visual-inertial odometry, obstacle detection, and context-aware keyframe detection. In particular, this research addresses the following research questions:

1- How can the problem of odometry be modeled in a featureless and end-to-end learnable approach that uses both visual and inertial data?

2- What would be a suitable technique to detect obstacles in visual data without taking common supervised learning approaches?

3- What is the efficient rate for context-aware keyframe detection in visual-inertial odometry?

4- How should inertial sensor data-stream be analyzed for detecting a specific context data to help with navigation?

1.2 Contributions

This thesis has three major contributions:

1- A novel framework for visual inertial odometry that uses a deep recurrent convolutional neural network (RCNN) and an extended Kalman filter to solve the tracking problem in a more efficient way.

2- A new algorithm for detecting obstacles in unfamiliar environments.

3- A context-aware methodology for frame detection in video streams for navigational tasks.

1.3 Organization

This dissertation is organized into six chapters. Chapter 1 states the motivation of this research, gives an overview of the challenges in this study, and states its goal, objectives and contributions. Chapter 2 reviews background information and related work. Chapter 3 presents the proposed novel framework for visual inertial odometry using deep learning and sensor fusion. Chapter 4 presents the obstacle detection algorithm for moving through unfamiliar environments. Chapter 5 describes the solution for problem of keyframe detection regarding context awareness. Finally, chapter 6 summarizes the research and its contributions, discusses conclusions, and provides recommendations for future research.

2.0 Background and related work

In this chapter the background of positioning, obstacle detection, and enhancing context-awareness for navigation along with related work are discussed. In the following sections, first, it is explained how to utilize a camera as a visual pose estimator for odometry in order to render the camera into a real-time six degrees of freedom (6DoF). Then, it is presented how the estimated pose can be scaled using inertial data from MEMS. These concepts establish the necessary foundation for the proposed framework for visual inertial odometry in this thesis. Second, the required background of obstacle detection for my proposed algorithm is presented and related work for enhancing context-awareness is discussed. Finally, possible contexts for navigation are explained and their foundations for my work are presented.

2.1 Non-RF-based indoor positioning and localization

Most mobility assisting devices and navigation systems are mainly dependent on GNSS, especially the GPS receivers. However, using GPS for indoor positioning and tracking is not reliable due to signal loss and impossibility to work in GPS-denied environments (Moreno, Shahrabadi, José, du Buf, & Rodrigues, 2012). Although Radio Frequency (RF)-based techniques such as assisted or augmented GPS (A-GPS), Wi-Fi-based positioning, cellular-based positioning, or Bluetooth-based positioning can be utilized for dealing with signal loss issue, for the sake of generality, the focus of this thesis is on non-RF techniques using the available sensors on a mobile device.

Techniques for solving the problem of positioning and localization within indoor environments can be grouped into four general categories: (a) “Dead-reckoning”, a solution to estimate users’ location based on aggregating displacement with the last known location. Odometry as a relative positioning technique implements this approach for finding location where the required data can be obtained through different sensors (Bonarini, Matteucci, & Restelli, 2004); (b) “Direct sensing”, a technique for positioning which determines the location by reading and sensing identifiers or tags which either contain or retrieve, from a database, location information (Ganz, Schafer, Tao, Wilson, & Robertson, 2014). RFID tags and Bluetooth beacon are examples of this technique for absolute indoor positioning; (c) “Triangulation”, where the location of at least three known points are used to determine users’ locations (Tekdas & Isler, 2010). Wireless local area networks (WLANs) positioning triangulates the location of wireless base stations using the provided signal strength of each station; and (d) “pattern recognition-based positioning”, such as computer vision techniques, where a camera captures images or video streams of the environment and by using matching algorithms, the system tries to find known locations from available databases (Fallah, Apostolopoulos, Bekris, & Folmer, 2013). Recently, the last category has become efficient and reasonable for localization as deep learning algorithms and rich image datasets are available on the Internet (Weyand, Kostrikov, & Philbin, 2016).

In this thesis, the focus is on solving the positioning problem by using a dead-reckoning technique and a single camera as the most common visual sensor in integrated moving platforms such as robots and smartphones. Hence, this research addresses indoor positioning by visual-inertial odometry as an incremental approach for relative positioning technique using “monocular camera” for collecting visual data besides inertial data.

Visual Odometry (VO) is used for the estimation of 6DoF for capturing the trajectory of a moving object. In monocular VO approach, obtaining the absolute value of scale of the trajectory just based on visual data is a challenging task. To tackle this scale challenge, I need data from other available sources such as Inertial Measurement Unit (IMU), to address some of the scale ambiguity shortcomings of visual odometry. In the rest of this chapter the background for the discussed tasks in the Introduction chapter are explained and related work is presented.

2.1.1 Localization as recursive state estimation

The main idea of localization is estimating the state of a moving object which is the position in a coordinate system using sensor data. Position is not directly measurable and needs to be estimated via sensed data. In this research state is considered as the collection of all aspects related to position, velocity, scale, and some errors of the moving object that can impact the future. For estimating the state, I consider two fundamental types of interactions: environment sensor measurements and control actions. Environmental measurement is a process by which we can obtain information about the state of the environment and environmental measurement data provides information about a momentary state of the environment such as taking image of surrounding or measuring range using laser scanner which is called measurement. The measurement data at time *t* is denoted $\boldsymbol{z_t}$. The other type of interaction, control action, is able to change the state of the world by forcing the moving object to displace. Therefore, control data includes information about the change of state in the environment. For example, acceleration to the moving object is a control action. In localization, although odometry uses sensor data for estimating the state, it is considered as control data for the moving platform, since it measures a control action. This control data, like accelerometer measurement, is denoted $\boldsymbol{u_t}$ which is usually a sequence of measurements.

2.1.2 Visual odometry

Visual odometry models the procedure of computing the coordinate and orientation of a moving object who carries a camera in a continuous fashion (Scaramuzza & Fraundorfer, 2011). Image acquisition provides a stream of frames from the visual sensor. These frames should be processed (extracting features, finding corresponding features, etc.) and camera pose should be computed. In order to retrieve displacement and motion of moving camera between two moments, we need the corresponding frames where the center of projection of each frame indicates the location at that moment (Figure 2.1). The problem of visual odometry mainly has been tackled by two approaches: feature-based and appearance-based. While in the feature-based approach a sparse set of image features is detected and tracked, the appearance-based approach relies on the pixel intensity values to extract 6DoF.

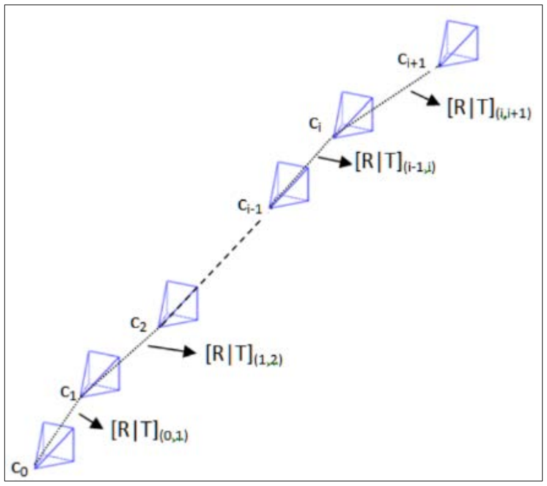


Figure 2.1: Consecutive frames (Ming Ren, 2012)

There are different algorithms for the feature-based approach to detect salient feature points such as FAST (Features from Accelerated Segment Test) (Rosten & Drummond, 2006), SURF (Speeded Up Robust Features) (Bay, Tuytelaars, & Van Gool, 2006), BRIEF (Binary Robust Independent Elementary Features) (Calonder, Lepetit, Strecha, & Fua, 2010), ORB (Oriented FAST and Rotated BRIEF) (Rublee, Rabaud, Konolige, & Bradski, 2011) and Harris (Harris & Stephens, 1988a) corner detectors. In order to track these feature points in the next sequential frame, a feature point tracker such as the Kanade–Lucas–Tomasi (KLT) (Shi & others, 1994; Tomasi & Kanade, 1991) tracker must be used. The result thus obtained is the optical flow, following which ego-motion can then be estimated using the camera parameters as proposed by Nister (Nistér, 2004) which is a general approach for detecting feature points and tracking them in both monocular vision and stereo vision-based approaches (Matthies, 1989) and (Johnson, Goldberg, Cheng, & Matthies, 2008). Parallel tracking and mapping (PTAM) was proposed by (Klein & Murray, 2007) which is a keyframe based approach with two parallel processing threads for the task of robustly tracking a lot of features and producing a 3D point map by Bundle Adjustment technique. In other words, PTAM is a robust feature tracking-based SLAM algorithm in real-time by parallelizing the motion estimation and mapping tasks (Kneip, Chli, & Siegwart, 2011; Mur-Artal, Montiel, & Tardos, 2015; Weiss, 2012; Weiss et al., 2013).

In the feature-based approach by extracting a set of corresponding points in the frames, geometric relations such as rotation matrix **R** and translation vector **t** can be retrieved to explain the motion (Figure 2.2). Equation 2.1 shows the relationship between the camera and the world coordinates (Szeliski, 2010). Accordingly, we need the intrinsic parameters of camera which are denoted by matrix ***K***.

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{p}_w = \boldsymbol{P}\boldsymbol{p}_w \qquad \textbf{2.1}$$

where $\boldsymbol{p}_w$ is 3D world coordinates, $\boldsymbol{P}$ is camera matrix. $\boldsymbol{K}$ is indicating intrinsic parameters of camera which are known by camera calibration process. For recovery of camera pose, matrix $\boldsymbol{K}$ is needed, which needs to perform camera calibration for obtaining the intrinsic matrix $\boldsymbol{K}$ (Ming Ren, 2012).

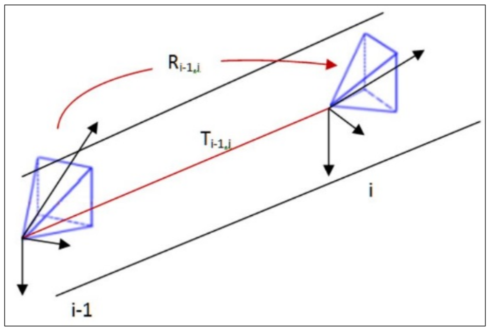


Figure 2.2: Estimation of camera displacement by visual odometry (Ming Ren, 2012)

Matrix $\boldsymbol{K}$ is needed before the camera pose recovery process. In order to obtain the intrinsic matrix $\boldsymbol{K}$, camera calibration must be performed. In this research, we estimate $\boldsymbol{K}$ by using offline camera calibration, due to accuracy and perfomance. $\boldsymbol{K}$ is a 3×3 calibration matrix which describes the camera intrinsic parameters. In the camera calibration process, we try to find the intrinsic parameters of $\boldsymbol{K}$, including calibrating the position of image center, estimating the focal length, using different scaling factors for row pixels and column pixels, and accounting for any skew factor and lens distortion. In order to perform camera calibration, I took pictures of a known object, chessboard, and by knowing the coordinates of given object points in the real world, I obtain internal camera parameters through an optimization algorithm. Equation 2.2 indicates how calibration matrix relates the camera coordinates with the projection center coordinates (Ren, 2012; Szeliski, 2010).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

For the purpose of finding the geometric relationship between two frames we need a different set of image features corresponding to 3D objects. Thus, the initial processing of the frames is to extract a set of salient features that are present in each frame. I used SURF to extract local features and build descriptors as feature vectors. SURF was utilized since it is faster than other algorithms (Xu & Namit, 2008). By using the aforementioned technique for detecting features, the RANSAC (RANdom SAmple Consensus) algorithm (Derpanis, 2010; Fischler & Bolles, 1981) and normalized 8-point algorithm (Hartley & Zisserman, 2003) are used to find the corresponding pairs of detected features on two frames and compute fundamental matrix $\boldsymbol{F}$. Essential matrix $\boldsymbol{E}$ can be obtained by $\boldsymbol{E} = \boldsymbol{K^T F K}$

Once the essential matrix $\boldsymbol{E}$ is obtained, we need to extract rotation and translation from it. Basically, I want to decompose it into ***E=TR*** where $\boldsymbol{T}$ is a skew symmetric matrix and $\boldsymbol{R}$ is a rotation. I will use the two matrices:

$$E = \lfloor t_\times \rfloor R \tag{2.3}$$

where R is the rotation matrix and $\lfloor t_\times \rfloor$ is skew symmetric of translation vector $\boldsymbol{t}$. Taking the singular-value decomposition (SVD) of the essential matrix, and then exploiting the constraints on the rotation matrix yields $\boldsymbol{R}$ and $\boldsymbol{t}$.

While the feature-based approach of visual odometry is widely used for solving the problem of relative positioning, direct or appearance-based visual odometry become more popular (Davison, Reid, Molton, & Stasse, 2007; Engel, Koltun, & Cremers, 2016; Jin, Favaro, & Soatto, 2003; Newcombe, Lovegrove, & Davison, 2011; Pretto, Menegatti, & Pagello, 2011; Silveira, Malis, & Rives, 2008) which relies on processing pixel density data using Convolutional Neural Networks (CNNs) (LeCun & Bengio, 1995). CNN provide the advantage of solving numerous computer vision tasks in more efficiently ways and with higher accuracy compared to traditional geometry-based approaches. Although using CNN, problems like classification, visual recognition, depth regression (Eigen, Puhrsch, & Fergus, 2014), object detection (S. Ren, He, Girshick, & Sun, 2015) and segmentation problems (Long, Shelhamer, & Darrell, 2015) have been efficiently solved, it has not been implemented in the domains of Structure from Motion, SLAM and Visual Odometry using advances in deep learning. Recently, optical flow between images has been computed by deep networks such as FlowNet (Dosovitskiy et al., 2015) and EpicFlow (Revaud, Weinzaepfel, Harchaoui, & Schmid, 2015). Also, homography between two images has been obtained by deep networks (DeTone, Malisiewicz, & Rabinovich, 2016).

Deep learning for odometry was applied by Nicolai, Skeele et al., however, they used laser data from a LIDAR (Nicolai, Skeele, Eriksen, & Hollinger, 2016) in which the point cloud is projected on the 2D image plane and pass it to a neural network for estimating position and orientation. The deep learning approach has also been used for visual odometry for stereo images in work of Konda and Memisevic (Konda & Memisevic, 2015) where a classification approach was adopted to the problem. The classifier is a convolutional neural network a with a softmax layer to determine the relative displacement and change in orientation between two consecutive keyframes using a prior set of discretized velocities and directions. Moreover, Agrawal et al. (Agrawal, Carreira, & Malik, 2015) proposed using vector of egomotion for feature learning where a classification task was used for inferring egomotion. In this research, I treat the visual odometry estimation as a regression problem.

DeepVO was proposed by Mohanty et al. (V. Mohanty et al., 2016b) which is a Siamese AlexNet-based architecture (Du & Shen, 2016). In this technique, the translational and rotational elements are computed using regression through an L2-loss function with equal weight values. Furthermore, Melekhov et al. (Melekhov, Ylioinas, Kannala, & Rahtu, 2017) use a weight factor for balancing both translation and rotation variables of the loss. Moreover, a spatial pyramid pooling layer was added to the architecture of the network which is helpful for making it more robust to varying input image resolutions.

2.2 Obstacle detection

It is a basic and challenging task to avoid obstacles while moving and traveling in unfamiliar environments for moving platforms like robots, self-driving cars, and visually impaired people. Micro-navigation, or obstacle detection, systems are developed to address this challenge (Katz et al., 2012). Micro-navigation systems enable the moving platform to safely navigate through the environment. There are various approaches and techniques to avoid obstacles such as optical triangulation (Benjamin, Ali, & Schepis, 1973), an acoustic-triangulation method (Borenstein, 2001). In this research, I focus on addressing the problem of obstacle detection in indoor environments using computer vision techniques for processing the frames and sensors (accelerometers, gyroscope, magnetometer) on smartphones for activity recognition and contextual-aware frame extraction. However, using these sensors and a single camera, we are not able to directly measure or estimate depth.

In the area of robotics, depth estimation and geometric modeling of the ambient environment are possible by using stereo camera (imagery) (Cheng, Li, & Chen, 2010; Lee, Doh, Chung, You, & Youm, 2004; Murray & Little, 2000; Nakju et al., 2004), and some other sensors such as radar, LiDAR, and RFID. However, in this research I do not use stereo imagery or LiDAR sensors, as they would make detection of obstacles a more challenging task. In spite of these limitations, a moving camera can provide a video stream that consists of time-indexed frames. With two consecutive images (frames), we can track certain points, measure their velocity, and evaluate them to determine whether they are part of an obstacle or not. Embedded cameras typically take videos at 30 fps (frames per second). Obviously, when the scene changes even slightly, processing all those frames is computationally expensive and is not needed.

In this research, the fundamental idea for detecting obstacles is processing the captured visual data of the scene while moving through indoor environments. Obstacles are defined and detected as areas on the visually captured scene which potentially hinder safe movement. The data processing consists of four steps. The first step involves extracting a subset of all the frames from the video stream to reduce data redundancy. The frame extraction is based on a context-aware process which considers user activity. The second step involves selecting a few significant points on the frames to evaluate their motion and displacement pattern. The third step involves labeling the selected points to see whether they are part of foreground or background. Finally, the fourth step involves clustering the foreground points which belong to either close or moving objects. For clustering I take advantage of motion pattern and consider a new parameter as time-to-contact. The clustered points yield areas that are more likely part of an obstacle.

There are numerous studies that have focused on obstacle detection (Boroujeni, 2012; Bousbia-Salah, Bettayeb, & Larbi, 2011; Costa, Fernandes, Martins, Barroso, & Hadjileontiadis, 2012; Rodríguez et al., 2012; Tapu, Mocanu, Bursuc, & Zaharia, 2013). My focus is on optical-flow-based obstacle detection, which is a vision-based technique. Vision-based navigation systems are mostly based on stereo images. A stereo vision system introduced by Martinez and Ruiz (Costa et al., 2012) computes and builds a 3D map of adjacent areas through a six degree of freedom egomotion algorithm. With the resulting map, it is possible to detect head-level obstacles. Stereo-vision techniques are used in head mounted devices as well. A 3D map was made by integrating visual- and feature-based metric-topological simultaneous localization and mapping (SLAM) (Pradeep, Medioni, & Weiland, 2010). Therefore, by using the 3D map, obstacles could be detected. In addition, since the details of the environment were captured, a safe walking path could also be created for the user.

In this research, obstacle detection was accomplished by using optical flow and point tracking algorithms. Detecting obstacles based on optical flow is common in both vehicle navigation (Batavia, Pomerleau, & Thorpe, 1999; Stark, 1991) and robotics (Low & Wyeth, 1998; Ohya, Kosaka, & Kak, 1998; Song & Huang, 2001; Souhila & Karim, 2007; Zingg, Scaramuzza, Weiss, & Siegwart, 2010). Qian et al. used a moving camera for detecting obstacles during navigation. They mounted the camera on a vehicle and proposed a method to detect obstacles on a road, regardless of whether the camera was stationary or moving (Qian, Tan, Kim, Ishikawa, & Morie, 2013). El-Gaaly et al. designed a system using a smartphone for detecting obstacles through optical flow for navigating a watercraft in which optical flow and tracking process just computed for a sparse set of points (El-Gaaly, Tomaszewski, & Valada, 2013). Optical flow-based obstacle avoidance is also used for micro-aerial vehicles (MAVs). Zingg et al. proposed an approach for wall collision avoidance for MAVs in indoor environments based on optical flow (Zingg et al., 2010).

Our proposed approach is to use a moving monocular camera that provides a video stream as part of the input data to the solution, plus other smartphone sensors. A monocular moving-camera approach was developed by Peng et al.(2010) which uses a camera on a smartphone for obstacle detection (Peng, Peursum, Li, & Venkatesh, 2010). Their system can detect obstacles on the floor and interact with users through vibration and vocal feedback. Region of interest (ROI) was also introduced in this system, which was a trapezoid that was computed based on known tilt angle and focal length of the camera. Moreover, José and Rodrigues (José, Buf, & Rodrigues, 2012) proposed path window as ROI which was triangular similar to my ROI. The upper vertex of their triangle is the vanishing point of the scene. Using a vanishing point is not useful for my research, since the vanishing point is independent of the heading of the user.

Tapu et al. (2013) proposed a system for micro-navigation using a smartphone. Their system has two distinct modules for obstacle detection and object recognition. Object recognition is a function to recognize the detected obstacles. The system detects obstacles based on computing optical flow on a point dataset and evaluates them to see if they belong to an obstacle or not. Basically, the point dataset is a predefined grid of points. The points that should be processed and analyzed in these techniques are either predefined or irregular detected by image descriptor, however, in this research, I use the extracted points by an algorithm which considers both predefined and irregular points in an efficient way.

2.3 Enhancing context-awareness

For the purpose of improving efficiency and effectiveness of my proposed methodology and algorithm, I propose to make them context-aware according to activity recognition. There are two different paradigms that use sensor data for activity recognition: data-driven and knowledge-driven (Sara Saeedi et al., 2014). My focus for the first part of this research is on the data-driven paradigm, in which a feature vector is comprised by selecting more efficient features. I classify the feature vector using a supervised classification algorithm to recognize a user's activity. Bao and Intille (2004) developed an algorithm to detect 20 different physical activities from data acquired using five small biaxial accelerometers (Bao & Intille, 2004). They used accelerometers worn simultaneously on different parts of the body of the user and evaluated their performance and accuracy. Ravi et al. (2005) also developed an activity recognition technique that used accelerometers and tried to recognize some common activities, such as standing, walking, running, climbing upstairs, climbing downstairs, sitting, vacuuming, and brushing teeth. They used decision tables, decision trees (C4.5), K-nearest neighbors, Naive Bayes, and SVM algorithms for interpreting the data (Ravi, Dandekar, Mysore, & Littman, 2005).

A user's activity is not the only contextual condition that can influence the process of frame extraction. Device status or orientation is a second contextual factor that can affect the recognition process. Figure 2.3 shows six different possible device orientations. According to these orientations, when the Z-axis is pointing up or down (the device is face up/face down), the initial condition is not appropriate for taking any frame, and thus works as an initial constraint. The gyroscope and accelerometer can be used to detect each of these six orientations (S Saeedi, 2013).

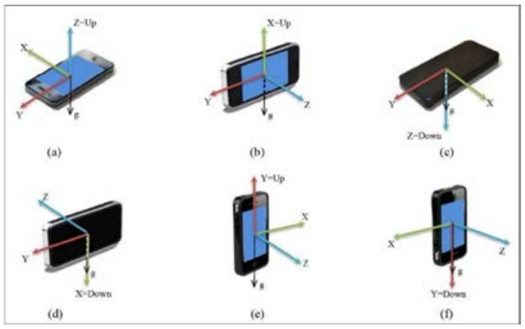


Figure 2.3: Six different orientations of the cellphone (Sara Saeedi et al., 2014)

Extracting frames based on mode of activity was proposed by Ren and Karimi (Ming Ren & Karimi, 2012). They used a 3D accelerometer to comprise the feature vector and a decision tree for classification of activities. They used extracted frames for localization and odometry in map matching. However, their approach does not consider all the available motion sensors. Also, the feature selection procedure is based on evaluating features and reasoning about how impactful the extracted features could be for activity recognition. Hence, those that are considered less informative are ignored. In this research, I have improved upon that research by considering more sensors beside accelerometers, such as magnetometers and gyroscopes. Also, I have improved the feature selection process by selecting features based on their both Information Gain and ReliefF results and have evaluated many different classifiers to find the best one for this particular purpose.

Context of the moving platform can be inferred by analyzing its gait and associate it with other indices. One of the critical indices that can be understood using gait analysis and considered as a user context for navigation is estimation of how sober a pedestrian is. The ability to accurately measure Blood Alcohol Content (BAC) in the real world is vital for understanding the relationship between alcohol consumption patterns and the impairments of normal functioning that occur (such as those related to gait). Smartphone-based alcohol consumption detection that evaluates a gait pattern captured by inertial sensors was proposed by (Kao, Ho, Lin, & Chu, 2012), which labeled each gait signal with a Yes or a No in relation to alcohol intoxication. The study by Kao and colleagues did not examine the quantity of drinks consumed but focused its analyses solely on classifying a subject as intoxicated or not, thus limiting applicability across different ranges of BAC. Park et al. (2017) used a machine learning classifier to distinguish sober walking and alcohol-impaired walking by measuring gait features from a shoe-mounted accelerometer, which is impractical to use in the real world (Park et al., 2016). Arnold et al. (2015) also used smartphone inertial sensors to determine the number of drinks (not BAC), an approach which could be prone to errors given that the association between number of drinks and BACs varies by sex and weight (Arnold, Larose, & Agu, 2015).

Kao et al. (2012) conducted a gait anomaly detection analysis by processing acceleration signals. Arnold et al. (2015) utilized naive Bayes, decision trees, SVMs, and random forest methods, where random forest turned out to be the best classifier for their task. Also, in Virtual Breathalyzer (2016) AdaBoost, gradient boosting, and decision trees were used for classifying whether the subject was intoxicated (yes or no). Furthermore, they implemented AdaBoost regression and regression trees (RT), as well as Lasso for estimation of BrAC (Nassi, Rokach, & Elovici, 2016).

2.4 Keyframe detection

In order to detect the appropriate frames for processing to calculate their pose and add to the topological diagram of environment model three main approaches are generally used; cluster based, energy based, and the sequential techniques (Chatzigiorgaki & Skodras, 2009; Panagiotakis, Doulamis, & Tziritas, 2009). Since cluster and energy based can be applied on the whole video sequence for retrieving the frames, they are called global methods. The global techniques are not proper approaches for vSLAM and VO algorithms, because the keyframe must be detected while the rover is moving.

There are some common methods for keyframe detection in tracking algorithms including uniform sampling in space which means keyframes should be extracted for every traversed distance unit, either linear or angular, (Callmer, Granström, Nieto, & Ramos, 2008; Cummins & Newman, 2008), uniform sampling in time, where frames are captured in a constant time interval (Ho & Newman, 2006; Newman & Ho, 2005), and uniform sampling in appearance (Angeli, Doncieux, Meyer, & Filliat, 2008), which is based on the measuring the amount of change in appearance in consecutive frames. In other words, the amount of change in a frame compared to the last keyframe should exceed a predefined threshold to be introduced as new keyframe. This measurement can be done by some image similarity algorithms or entropy-based sampling.

The main goal of all these techniques is to establish a correlation between appearance change and another factor such as change in time or distance. (Callmer et al., 2008; Cummins & Newman, 2008) used distance-based sampling which assumes a strong correlation between appearance change and the amount of displacement. However, this assumption is dependent on the ambient environment and unknown geometry of the surroundings can cause some unstable results for the detected frames. Likewise, uniform time-based sampling assumes a correlation between the time interval of consecutive captured frames and appearance change. (Ho & Newman, 2006; Newman & Ho, 2005) utilized this approach for frame detection. This assumption can perform well when the mobile agent is moving with a constant velocity within an environment without dramatic changes, however it would not work well when the mobile agent has sudden movements like accelerates, decelerates or stops.

The main idea in appearance-based sampling is straightforward, as it tries to measure the change in appearance in a direct way. In theory, measuring the visual changes seems to be the most reasonable way for managing the frames, however the problem of computational cost is a real barrier. Some of well-known similarity measurements algorithms for the problem of detecting keyframe in vSLAM and VO are pixel-wise, global-histogram, local-histogram, feature matching, Bag-of-Words, and entropy measurement (Mentzelopoulos & Psarrou, 2004; H. Zhang, Li, & Yang, 2010). Although, the best performance is for feature matching technique, it has high computational cost. Dong et al. (2009) proposed a framework for selecting keyframes in order to reduce redundancy which is a proper framework for selecting effective keyframes in offline mode.

Two entropy-based approaches were designed by Das and Waslander (2015) which extract keyframes based on cumulative point entropy reduction and the point pixel flow discrepancy (PPFD).

2.5 Summary

In this chapter I presented the background and related work on the approaches and techniques along with the challenges in this dissertation. In the first part, I overviewed non-RF-based techniques and visual odometry for indoor positioning and localization. I also discussed the background and related work on obstacle detection and tried to depict a comprehensive scope of an unsupervised obstacle detection technique. In the rest of the chapter I focused on understanding and enhancing the solution of navigational tasks. Finally, in the last section I discussed the problem of keyframe detection challenge.

3.0 A framework for visual inertial odometry

Positioning is a core function of location-based services (LBS) and is often implemented by integrating different sensors. The most widely used sensor for positioning is GPS since it is available everywhere and anytime. However, in some places, such as indoor environments, GNSS signals are not accessible where positioning is achieved by utilizing other sensors. In this chapter, a framework for positioning in indoor environments, without GNSS and RF-based systems, is designed and tested.

3.1 Pose estimation and tracking

Design and implementation of a real-time and reliable indoor positioning system is a vital component in most of location-based services (LBSs). There are various applications of LBSs that are dependent on indoor positioning system (IPS). IPSs are designed and implemented using different technologies that compute the position of an object at a wide range of accuracies (Konstantinidis et al., 2015). However, these types of IPS need infrastructures such as beacons, wi-fi, cellular networks, and sensors pre-installed in the environment, which limits their applicability.

One of the most prevalent techniques for non-RF indoor positioning is dead reckoning integrated with visual data from a camera as visual sensor with inertial data. This approach is widely used for relative positioning using "monocular camera" for collecting visual data besides inertial data (Hesch, Kottas, Bowman, & Roumeliotis, 2013; Kasyanov, Engelmann, Stückler, &

Leibe, 2017; Kitt, Chambers, Lategahn, Singh, & Systems, 2011; Leutenegger, Lynen, Bosse, Siegwart, & Furgale, 2015). The conventional featured-based technique for visual odometry when integrated with inertial data is computationally expensive. One proper approach for computing the six-degree-of-freedom for each pair of consecutive images is possible by taking advantage of a learning-based approach. Since I use monocular camera as the visual sensor, I utilize inertial data to solve the problem of scale ambiguity. In this chapter I propose a framework for a learning-based approach for the problem of monocular visual inertial odometry.

To tackle the problem of pose estimation and tracking without necessity of presence of any infrastructure, I propose a framework to acquire visual and inertial data and integrate them with one another to produce reliable results. To achieve this goal, I modularize the solution into three major modules. First, an image acquisition module detects minimum and efficient frames from an incoming video stream; it involves determining the frame rate and provides the next module with the frame sequence. Second, a monocular visual odometry module determines the relative displacement and rotation of the camera, with respect to the immediate last known position. The visual odometry module yields unscaled coordinates of camera (agent) pose by processing consecutive frames. I call this unscaled, because in monocular visual odometry, the scale of the baseline is unknown. I use inertial data for estimating the scale, but I still need to fine-tune the results of this module. The third module is for fine-tuning the coordinates from visual odometry by fusing them with inertial sensor data from IMU using and an extended Kalman filter (EKF). Furthermore, the state vector has updated the scale and bias of the inertial sensors, which are used as feedback to the monocular visual odometry module. The user trajectory could be retrieved by the fine-tuned coordinates. In Figure 3.1, I show a schematic block diagram of the procedure.

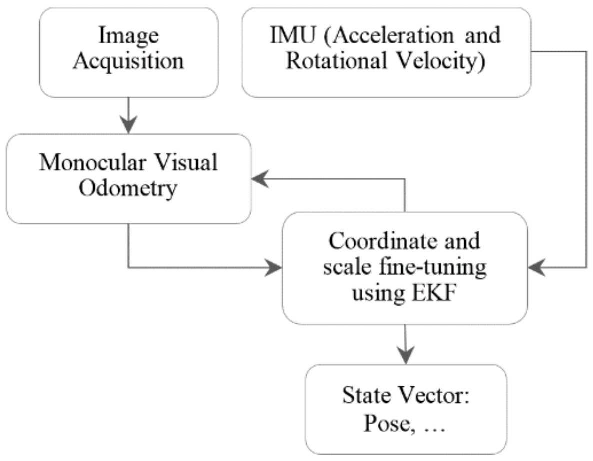


Image Acquisition
IMU (Acceleration and Rotational Velocity)
Monocular Visual Odometry
Coordinate and scale fine-tuning using EKF
State Vector: Pose, …

Figure 3.1: Block diagram of data processing

3.2 Visual odometry with deep recurrent convolutional neural networks

This section presents the proposed framework for visual inertial odometry, which is composed of two major modules. The first one is a monocular visual odometry module, which is a deep RCNN for computing 6DoF in a full and end-to-end approach. Indeed, the visual odometry module is mainly composed two parts: a feature extraction part using a CNN and a sequential modelling part using RNN. The second module is an extended Kalman filter module for fine-tuning the scale parameter.

3.2.1 Architecture of the proposed RCNN

In this section, I discuss the integration of a deep learning monocular visual odometry algorithm using a deep RCNN (Donahue et al., 2015) inspired by (Wang, Clark, Wen, & Trigoni, 2017) with an EKF module for fusing inertial data from IMU with coordinates from visual odometry. Computer vision tasks are usually able to take advantage of some well-known deep neural network architectures, such as VGGNet (Sattler, Leibe, & Kobbelt, 2017) and GoogLeNet (Shotton et al., 2013), which yield remarkable performance. Most architectures concentrate on solving the problems of recognition, classification, and detection (Wang et al., 2017). Computing 6DoF for two images is fundamentally different from the aforementioned computer vision tasks, because visual odometry is a geometrical problem that cannot be coupled with appearance. Therefore, using current deep neural network architectures to solve the problem of visual odometry is not practical.

Since I try to address visual odometry as a geometric problem, I need a framework that can learn geometric feature representations. Also, there are two critical requirements for modeling the problem of visual odometry: first, it is necessary to find connections among consecutive keyframes, such as models of motion, second, since the visual odometry systems work on keyframe sequences which are acquired while moving, it is needed proposed system be able to model that sequential behavior. RCNN can considers these two requirements. Figure 3.1 presents the architecture of the visual odometry system. This network takes a sequence of monocular keyframes as input. A tensor is formed by stacking two consecutive keyframes and passed to the RCNN for extracting motion information and estimating poses. The tensor from input data is fed into the CNN to generate features for the monocular visual odometry. The output features of the CNN are passed through the LSTM RNN in order to learn the sequential information. At each time step and for each pair of frames the network estimates a pose.

3.2.2 RNN-based sequential modeling

For finding the relationships among a sequence of extracted features by CNN and modeling the dynamics, a deep RNN is implemented to perform sequential learning. Modeling by RNN helps detecting proper sequential information; hence, it can provide more information than other techniques we usually use to explain geometry and movement (V. Mohanty et al., 2016a). Visual odometry should be able to model temporality which is due to the motion and sequential dynamics which is because of image sequence; thus, the RNN is a decent approach for modelling dependencies in a sequence. In other words, estimating the pose of the current keyframe can use the information within the previous keyframes(Hartley & Zisserman, 2003). However, it is not a decent way to feed an RNN with high dimensional raw data such as image for learning the sequential model in a direct form. Hence, in this framework instead of passing raw images into the recurrent network, the extracted features by CNN is passed to it. RNN is able to maintain the memory of its hidden states and has feedback loops enabling its current hidden state to be a function of the previous ones (Figure 3.1) (Wang et al., 2017). Hence, the relationship between the incoming keyframe and the former one in the same sequence can be detected by the RNN. It can be shown in Equations 3.1 and 3.2 having a feature x_k from CNN at time k, an RNN updates at time step k by

$$\boldsymbol{h}_k = \mathcal{H}(\boldsymbol{W}_{xh}x_k + \boldsymbol{W}_{hh}x_{k-1} + \boldsymbol{b}_h) \tag{3.3}$$

$$\boldsymbol{y}_k = \boldsymbol{W}_{hy}\boldsymbol{h}_k + \boldsymbol{b}_y \tag{3.4}$$

where $\boldsymbol{h}_k$ is the hidden state and $\boldsymbol{y}_k$ represents the output at time k. Moreover, $\boldsymbol{W}$**’s** denote weight matrices, $\boldsymbol{b}$**’s** show bias vectors, and $\mathcal{H}$ is an activation function (Wang, Clark, Wen, & Trigoni, 2018). In order to determine the correlations between keyframes in longer trajectories, the proposed RNN uses long short-term memory (LSTM), which can learn long-term dependencies by using memory gates and units (Zaremba & Sutskever, 2014). The following equations from (Wang et al., 2018) shows that given the input feature $\boldsymbol{x}_k$ and the hidden state $\boldsymbol{h}_{k-1}$ and the memory cell $\boldsymbol{c}_{k-1}$ of the previous LSTM unit. LSTM updates at time step k, according to

$$\boldsymbol{i}_k = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_k + \boldsymbol{W}_{hi}\boldsymbol{h}_{k-1} + \boldsymbol{b}_i) \tag{3.5}$$

$$\boldsymbol{f}_k = \sigma\left(\boldsymbol{W}_{xf}\boldsymbol{x}_k + \boldsymbol{W}_{hf}\boldsymbol{h}_{k-1} + \boldsymbol{b}_f\right) \tag{3.6}$$

$$\boldsymbol{g}_k = \tanh\left(\boldsymbol{W}_{xg}\boldsymbol{x}_k + \boldsymbol{W}_{hg}\boldsymbol{h}_{k-1} + \boldsymbol{b}_g\right) \tag{3.7}$$

$$\boldsymbol{c}_k = \boldsymbol{f}_k \circ \boldsymbol{c}_{k-1} + \boldsymbol{i}_k \circ \boldsymbol{g}_k \tag{3.8}$$

$$\boldsymbol{o}_k = \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_k + \boldsymbol{W}_{ho}\boldsymbol{h}_{k-1} + \boldsymbol{b}_o) \tag{3.9}$$

$$\boldsymbol{h}_k = \boldsymbol{o}_k \circ \tanh(\boldsymbol{c}_k) \tag{3.10}$$

where $\circ$ is the element-wise product of vectors, $\boldsymbol{i}_k$ is input gate, $\boldsymbol{f}_k$ is forget gate, $\boldsymbol{g}_k$ is input modulation gate, $\boldsymbol{c}_k$ is memory cell, and $\boldsymbol{o}_k$ is output gate at time k. In this case, two LSTM layers are stacked for forming the deep RNN illustrated in Figure 3.1. The outputs of RNN is an estimation of pose vector at each time step using the visual features generated from the CNN as input.

3.2.3 CNN-based feature extraction

Feature extraction is conducted by a CNN to determine the features that are more effective for solving the visual odometry problem. This CNN was developed for feature extraction on the stacked model of two consecutive frames from monocular visual sensor. Ideally, the feature representation is geometric instead of being associated with either appearance or visual context as visual odometry systems need to be generalized and are often used in unknown settings (Wang et al., 2017). The structure of the CNN is inspired by the network for computing optical flow in (Dosovitskiy et al., 2015). The convolutional layers, except for the last one, are followed by a rectified linear unit (ReLU) activation. In order to capture the interesting features, the sizes of the fields in the network are from 7 × 7 to 5 × 5 and then 3 × 3. This learned feature from CNN reduces the dimensionality of RGB frame which too high for RNN and represent it in a compact form. Moreover, it enhances the sequential training procedure. As a result, the final convolutional feature is fed to the RNN for sequential modelling.

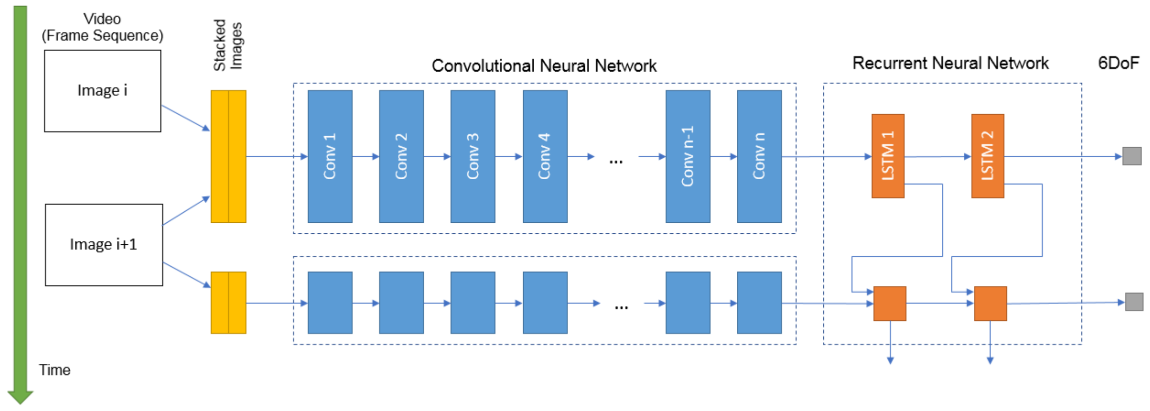


Figure 3.2: Architecture of RCNN for monocular visual odometry

3.2.4 Cost function and optimization

The proposed Recurrent CNN for visual odometry system computes the conditional probability of the poses $\boldsymbol{P}_t = (p_1, \ldots, p_t)$ at time t, given a sequence of frames $\boldsymbol{F}_t = (f_1, \ldots, f_t)$ up to time t (Wang et al., 2018):

$$p(\boldsymbol{P}_t | F_t) = p(p_1, \ldots, p_t | f_1, \ldots, f_t) \qquad 3.11$$

The recurrent convolutional neural network can handle the modeling and probabilistic inference to find the optimal parameters $\boldsymbol{v}^*$ for the VO by maximizing as in (3.12):

$$\boldsymbol{\nu}^* = \underset{\nu}{\operatorname{argmax}}\, p(P_t|\boldsymbol{F}_t;\boldsymbol{\nu}) \qquad 3.12$$

The hyperparameters $\boldsymbol{\nu}$ is learned by minimizing the Euclidean norm between pose including coordinate and orientation in ground truth $(\boldsymbol{p}_k, \boldsymbol{\theta}_k)$ and the estimated one as output of the network $(\widehat{\boldsymbol{p}}_k, \widehat{\boldsymbol{\theta}}_k)$ at time k. Also, the loss function is the mean square error (MSE) of all positions $\boldsymbol{p}$ and orientations $\boldsymbol{\theta}$ using the L2-norm:

$$\boldsymbol{\nu}^* = \underset{\nu}{\operatorname{argmin}} \frac{1}{N}\sum_{k=1}^{t} \|\widehat{\boldsymbol{p}}_k - \boldsymbol{p}_k\|_2^2 + \kappa\left\|\widehat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k\right\|_2^2 \qquad 3.13$$

In order to balance the values of weight for position and orientation a scale factor κ is introduced. Also, N is the number of samples.

3.3 Sensor fusion for visual inertial odometry

Monocular visual odometry uses a single camera; consequently, the baseline between two consecutive frames is unknown, and we have a defect for finding the absolute distance of the baseline. In other words, the scale factor of reconstruction is ambiguous, and the metric scale cannot be recovered by monocular camera. Furthermore, scale drift causes an instability of the navigation system. To find the geometric scale and resolve this ambiguity issue, it is important to compute the geometric scale using another source. I use an accelerometer and gyroscope for estimating the scale factor. The schema of motion estimation for consecutive keyframes is illustrated in Figure 2.1.

The generally travelled distance for an agent that moves with a variable acceleration is computed by the double integral of acceleration over time from t_i to t_{i+1}. From a computational perspective, I assume that the acceleration in each time interval is constant, where having inertial data at 30 Hz is a reasonable assumption, and I can compute the distance for each time interval using Equations 3.14 and 3.15. However, achieving this goal requires more than simply using accelerometer observations. The first challenge is with the noisy data from IMU, which creates a large error over the double integral. Hence, it should be refined, and noise and bias must be removed. Moreover, the observations of inertial sensors are in the inertial frame *{i}*; thus, the attitude of IMU should be considered to transform the acceleration vector to the world frame *{w}*. Finally, I need to consider the gravity component as well, since my observations also contain this element. To solve these challenges and obtain an accurate scale, I use an EKF for dealing with noisy data and solving the scale ambiguity problem, and I call this process coordinate and scale fine-tuning.

$$\Delta d = \iint_{\Delta t} a_m d\tau d\tau \qquad 3.14$$

where:

$$\delta d = v_0 \delta t + \frac{1}{2} a \delta t^2 \qquad 3.15$$

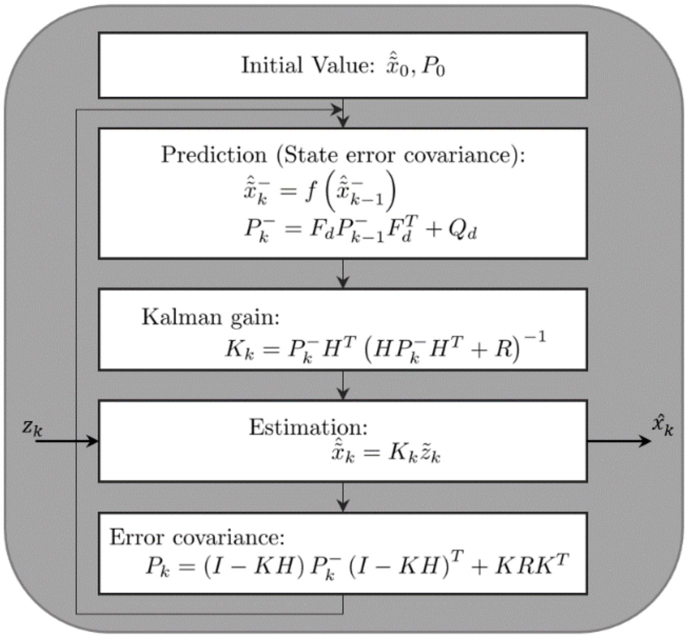
3.3.1 Coordinate and scale fine-tuning

I use an EKF to fuse visual data with inertial data and manage errors to obtain more accurate results. Figure 3.3 shows a schematic and general flow chart of the EKF algorithm. There are two major parts in the Kalman filter: a prediction process and an estimation process. The second box in the block diagram is for the prediction process, and three other boxes are associated with the estimation process. Besides, the filter receives the z_k observation vector and returns an updated state vector, which contains the coordinates of the camera pose and visual scale. The visual scale should be used for the next frame, as well as for scaling the estimated pose by the visual odometry module.

For purposes of modeling the system, I should account for inertial sensors, where accelerometers and gyroscopes yield observations over three axes. These measurements have a bias of b and white Gaussian noise of n, where bias is a non-static as a random variable. As previously mentioned, this system is also provided by visual sensor observations, which are unscaled 3D positions and attitude estimation with respect to the visual coordinate frame *{v}*. Acceleration and rotational velocity could be modeled as follows (Trawny & Roumeliotis, 2005):

$$a_m = a + b_a + n_a \tag{3.16}$$

$$\omega_m = \omega + b_\omega + n_\omega \tag{3.17}$$



Initial Value: $\hat{\tilde{x}}_0, P_0$
Prediction (State error covariance):
$\hat{\tilde{x}}_k^- = f\left(\hat{\tilde{x}}_{k-1}^-\right)$
$P_k^- = F_d P_{k-1}^- F_d^T + Q_d$
Kalman gain:
$K_k = P_k^- H^T \left(H P_k^- H^T + R\right)^{-1}$
z_k
Estimation:
$\hat{\tilde{x}}_k = K_k \tilde{z}_k$
$\hat{x}_k$
Error covariance:
$P_k = (I - KH) P_k^- (I - KH)^T + KRK^T$

Figure 3.3: Extended Kalman filter block diagram

The filter state vector contains the position of the inertial unit in the world frame, denoted as p_w^i, which would be the reported location of the agent. I also have the velocity of IMU in the world frame as v_w^i and the IMU attitude quaternion as $\bar{q}_w^i$. Although these parameters show the state of IMU in the world frame, the camera frame and inertial frame are aligned and are equivalent, and as a result, they explain the data from both cameras and smartphones. Additionally I have to consider visual scale as another parameter in the state vector. An updated visual scale will be used in scaling visual odometry result in each iteration, and it would be refined as the model is updated more and more frequently. Furthermore, acceleration bias and gyro bias, which were considered as random variables, are considered in the state vector (Weiss & Siegwart, 2013). This state vector contains 17 elements.

$$x = \left[{p_w^i}^T \quad {v_w^i}^T \quad {\bar{q}_w^i}^T \quad {b_w}^T \quad {b_a}^T \quad \lambda^T \right]^T \tag{3.18}$$

Differential equations are as follows, where an intuitive.$C_{\bar{q}_w^i}$ denotes the rotational matrix associated with quaternion $\bar{q}_w^i$. Ω(.), and is also used for multiplication of a vector by a quaternion (Trawny & Roumeliotis, 2005).

$$\dot{p}_w^i = v_w^i \tag{3.19}$$

$$\dot{v}_w^d = C_{(q_w^d)}^T (a_m - b_a - n_a) - g \tag{3.20}$$

$$\dot{q}_w^d = \tfrac{1}{2}\Omega(\omega_m - b_\omega - n_\omega) q_w^d \tag{3.21}$$

$$\dot{b}_\omega = n_{b_\omega} \tag{3.22}$$

$$\dot{b}_a = n_{b_a} \tag{3.23}$$

$$\dot{\lambda} = 0 \tag{3.24}$$

The error state vector is defined as $\tilde{x} = x - \hat{x}$, where the difference of the state vector with its estimation is the error state vector. This vector has 16 elements:

$$\dot{\hat{p}}_w^d = \hat{v}_w^d \tag{3.25}$$

$$\dot{\hat{v}}_w^d = C_{(\hat{\bar{q}}_w^d)}^T \left(a_m - \hat{b}_a\right) - g \tag{3.26}$$

$$\dot{\hat{\bar{q}}}_w^d = \tfrac{1}{2}\Omega(\omega_m - \hat{b}_\omega) q_w^d \tag{3.27}$$

$$\dot{\hat{b}}_\omega = 0 \tag{3.28}$$

$$\dot{\hat{b}}_a = 0 \tag{3.29}$$

$$\dot{\hat{\lambda}} = 0 \tag{3.30}$$

where $\hat{\omega} = \omega_m - \hat{b}_\omega$ and $\hat{a} = a_m - \hat{b}_a$. I use these equations for linearization of the error state vector, and show this as Equation 3.31.

$$\dot{\tilde{x}} = F_c \tilde{x} + G_c n \tag{3.31}$$

where the noise vector is $n = \begin{bmatrix} n_a^T & n_{b_a}^T & n_w^T & n_{b_w}^T \end{bmatrix}^T$. The following discretiz ation is used for propagation:

$$F_d = exp(F_c \Delta t) = I + F_c \Delta t + \frac{1}{2!} F_c^2 \Delta t^2 + \cdots \tag{3.32}$$

$$F_d = \begin{bmatrix} I_{3\times3} & \Delta t & A & B & -C^T_{\hat{\bar{q}}^i_w} \frac{\Delta t^2}{2} & 0 \\ 0_{3\times3} & I_{3\times3} & C & D & -C^T_{\hat{\bar{q}}^i_w} \Delta t & 0 \\ 0_{3\times3} & 0_{3\times3} & E & F & 0_{3\times3} & 0 \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0 \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.33}$$

$$A = -C^T_{\hat{\bar{q}}^i_w} \lfloor \hat{a}_\times \rfloor \left(\frac{\Delta t^2}{2} - \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor + \frac{\Delta t^4}{4!} \lfloor \omega_\times \rfloor^2 \right) \tag{3.34}$$

$$B = -C^T_{\hat{\bar{q}}^i_w} \lfloor \hat{a}_\times \rfloor \left(-\frac{\Delta t^3}{3!} + \frac{\Delta t^4}{4!} \lfloor \omega_\times \rfloor - \frac{\Delta t^5}{5!} \lfloor \omega_\times \rfloor^2 \right) \tag{3.35}$$

$$C = -C^T_{\hat{\bar{q}}^i_w} \lfloor \hat{a}_\times \rfloor \left(\Delta t - \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor + \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor^2 \right) \tag{3.36}$$

$$D = -A \tag{3.37}$$

$$E = I - \Delta t \lfloor \omega_\times \rfloor + \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor^2 \tag{3.38}$$

$$F = -\Delta t + \frac{\Delta t^2}{2!} \lfloor \omega_\times \rfloor - \frac{\Delta t^3}{3!} \lfloor \omega_\times \rfloor^2 \tag{3.39}$$

$$Q_d = \int_{\Delta t} F_d(\tau) G_c Q_c^T F_d(\tau)^T d\tau \qquad 3.40$$

where $Q_c = diag\left(\sigma_{n_a}^2, \sigma_{b_a}^2, \sigma_{n_\omega}^2, \sigma_{n_{b_\omega}}^2\right)$ is the noise covariance matrix. Now, it is possible to compute the propagated state covariance matrix, which could be gained by following the presented equations, and by calculating *F_d* and *Q_d*. Using these matrices, I can obtain the propagated state covariance matrix as $P_k^- = F_d P_{k-1}^- F_d^T + Q_d$. According to the EKF diagram, now the prediction process results are ready to be used by the estimation process, which compensates the difference between measurement and prediction and yields a new estimation of state vector *x* and covariance matric P.

The visual odometry module yields the unscaled position and attitude of the camera. These values, both position and quaternion, are observations of the system, and I denote them as z_p and z_q, respectively. Moreover, these values are in the vision frame *{v}*, and I need a geometric relationship between the world frame *{w}* and the vision frame *{v}* as well. Equation 3.41 models the position observation by the visual odometry module. $C_{\bar{q}_v^w}^T$ maps vision and world coordinate frames. According to this formulation, I can demonstrate the position error as Equation 3.42, which should be linearized for the filter; hence I need to compute H_p in $\tilde{z}_{pl} = H_p \tilde{x}$ (Weiss & Siegwart, 2013):

$$z_p = p_v^c = \lambda C_{\bar{q}_v^w}^T p_w^i + n_p \qquad 3.41$$

$$\tilde{z}_p = z_p - \hat{z}_p = \lambda C_{\bar{q}_v^w}^T p_w^i + n_p - \hat{\lambda} C_{\hat{\bar{q}}_v^w}^T \hat{p}_w^i \qquad 3.42$$

$$H_p = \begin{bmatrix} \hat{\lambda} C_{\bar{q}_v^w}^T & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & C_{\bar{q}_v^w}^T \hat{p}_w^i \end{bmatrix} \qquad 3.43$$

For the other observation in the system (rotation measurement), the notion of quaternion and error quaternion is used again. Although the visual odometry module yields the rotation from the vision frame *{v}*, the assumption of equvalency of vision frame and inertial frame leads us to model z_q and its error, as follows:

$$z_q = \bar{q}_v^c = \bar{q}_w^c \otimes \hat{\bar{q}}_v^w \equiv \bar{q}_w^i \otimes \hat{\bar{q}}_v^w \tag{3.44}$$

$$\bar{z}_q = z_q - \hat{z}_q = H_q^{wi} \delta q_w^i \tag{3.45}$$

Ultimately, the measurement can be presented as Equation 3.46, where

$$H_q^{xy} = \begin{bmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & \bar{H}_q^{xy} \end{bmatrix} \tag{3.46}$$

$$\begin{bmatrix} \tilde{z}_p \\ \tilde{z}_q \end{bmatrix} = \begin{bmatrix} & H_p & \\ 0_{3\times 6} & \widetilde{H}_q^{wi} & 0_{3\times 7} \end{bmatrix} \tilde{x} \tag{3.47}$$

Now, I have all the needed matrices and parameters for the update process, which are the three last boxes in Figure 3.3, as a routine procedure in the Kalman filter. Therefore, I compute the residual as $\tilde{z} = z - \hat{z}$ and obtain the Kalman gain as $K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$. Finally, I can compute the correction of the state vector, which is $\hat{\tilde{x}}$, and by using this result, and update the state vector. The following is the updated error state covariance $P_k = (I - KH) P_k^- (I - KH)^T + KRK^T$ (Trawny & Roumeliotis, 2005).

3.4 Experiment

In this section the proposed method is evaluated by comparing its results with the results of another state-of-the-art algorithm and ground truth in different scenarios within different environment settings. I use VISO2, an open-source library (Geiger, Ziegler, & Stiller, 2011) which is one of the most popular visual odometry algorithms for comparison and evaluation. VISO2 uses feature-based matching to realize efficient monocular VO. Since monocular VO does not have an absolute scale, I manually set a scale to recover absolute positions. It is worth noting that the scale could have been recovered by using a fixed baseline of stereo vision if I had access to stereo images, but this was not within the scope of this dissertation.

3.4.1 Dataset

There are various datasets for visual inertial odometry for outdoor and mostly for car navigation. Since, this research concentrated on indoor environment, I either must generate my own dataset or use publicly available datasets for indoor navigation which are very limited. Accurate estimation of trajectory and compare it with the trajectory should be considered for evaluation of the positioning algorithms in indoor environments. This type of evaluation makes generation of a suitable dataset too complicated. To avoid unforeseen errors related to data preparation and be able to compare the results with other algorithms, I decided to use ADVIO dataset, which is for indoor environment and collected by smartphone (Cortés, Solin, Rahtu, & Kannala, 2018). Figure 3.4 shows the devices for collecting the dataset. The dataset was collected by handheld devices and published in 2108. It contains both indoor and outdoor data and I used only indoor data.

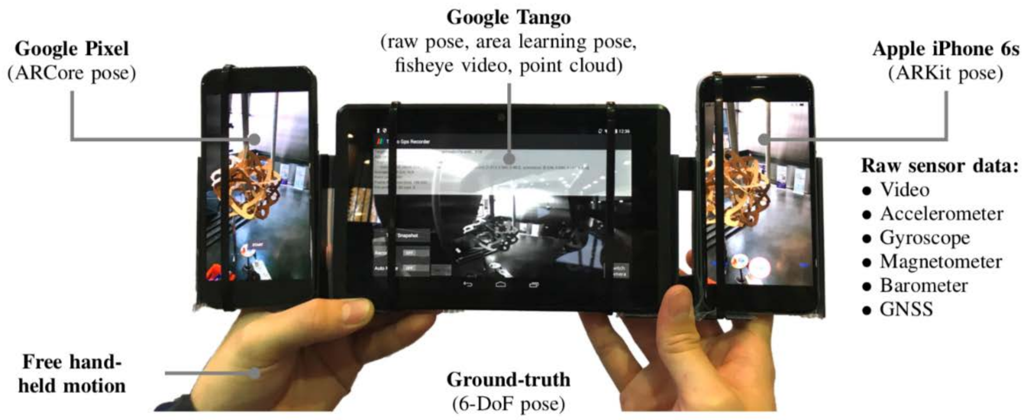
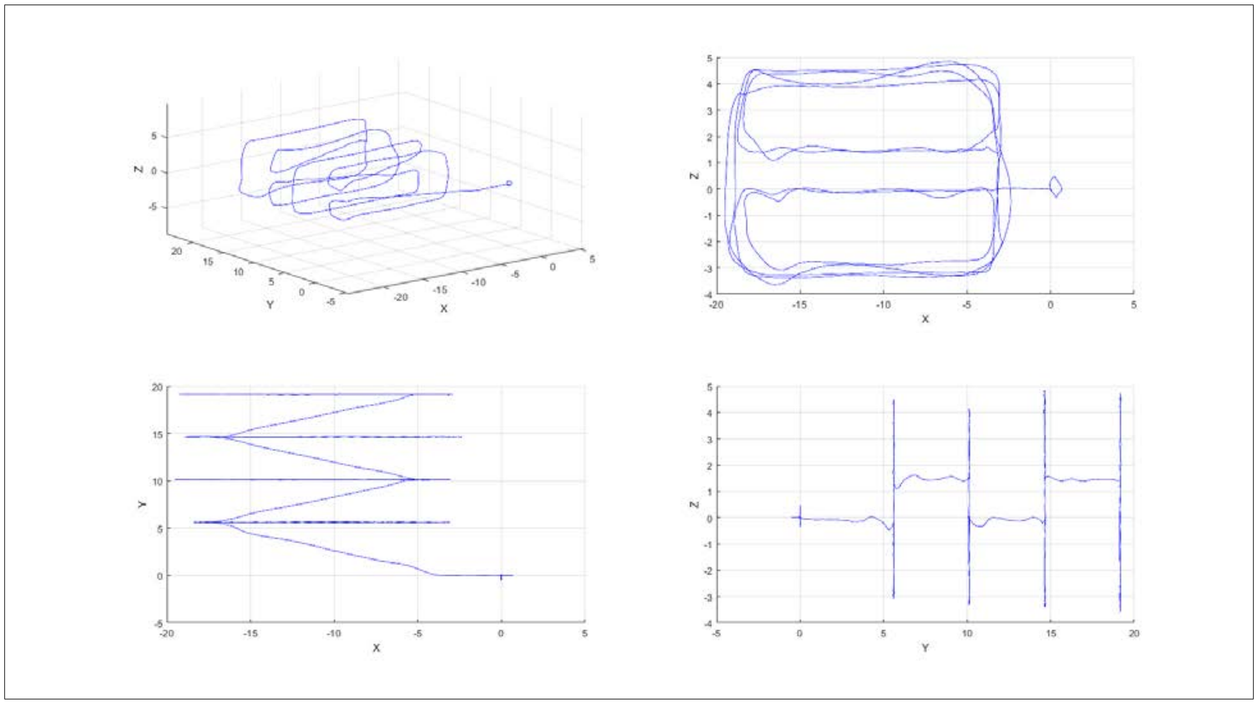
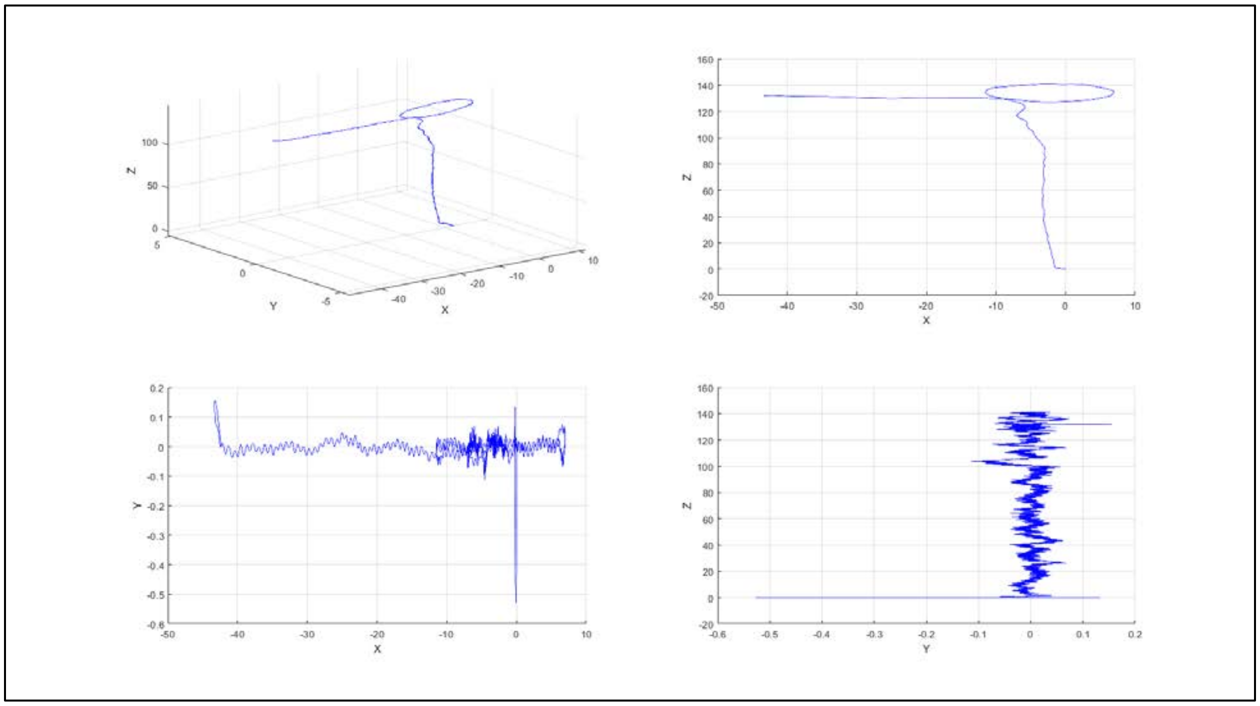


Figure 3.4: Custom-built capture rig with a Google Pixel smartphone on the left, a Google Tango device in the middle, and an Apple iPhone 6s on the right (Cortés et al., 2018)

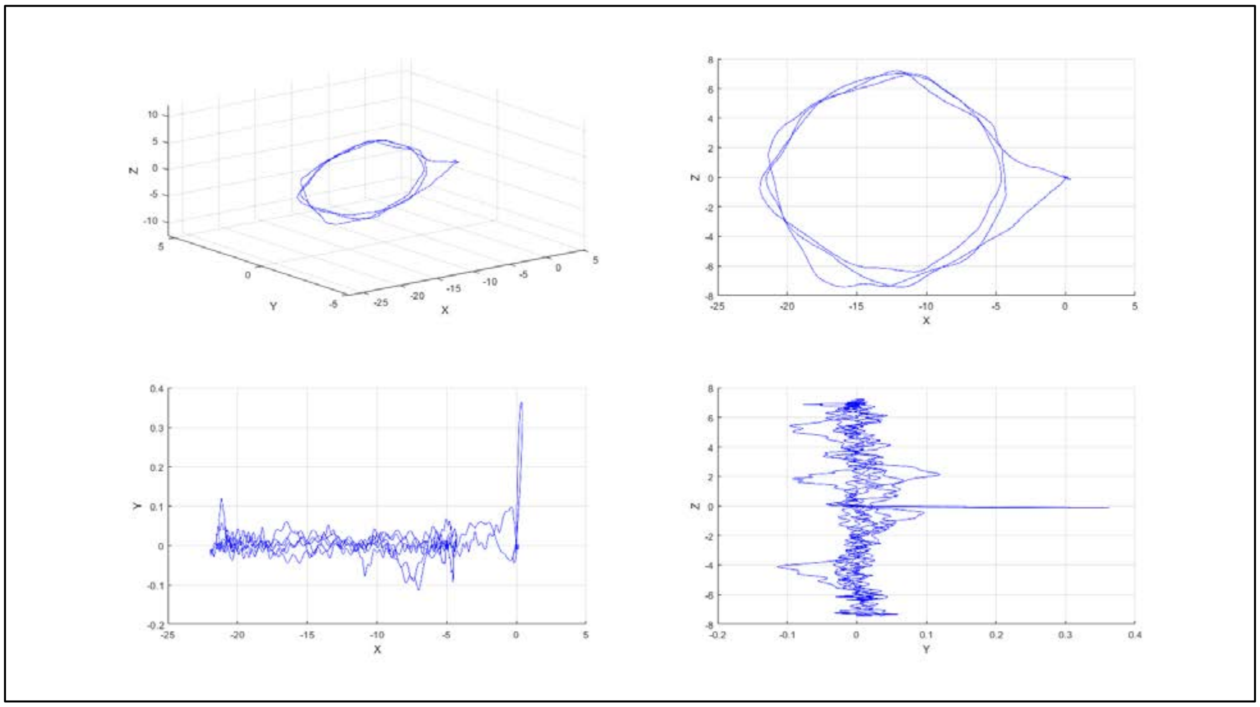
In Figure 3.5: (a) – (s), all sequences are shown in 3D and in XY, XZ, and YZ planes.



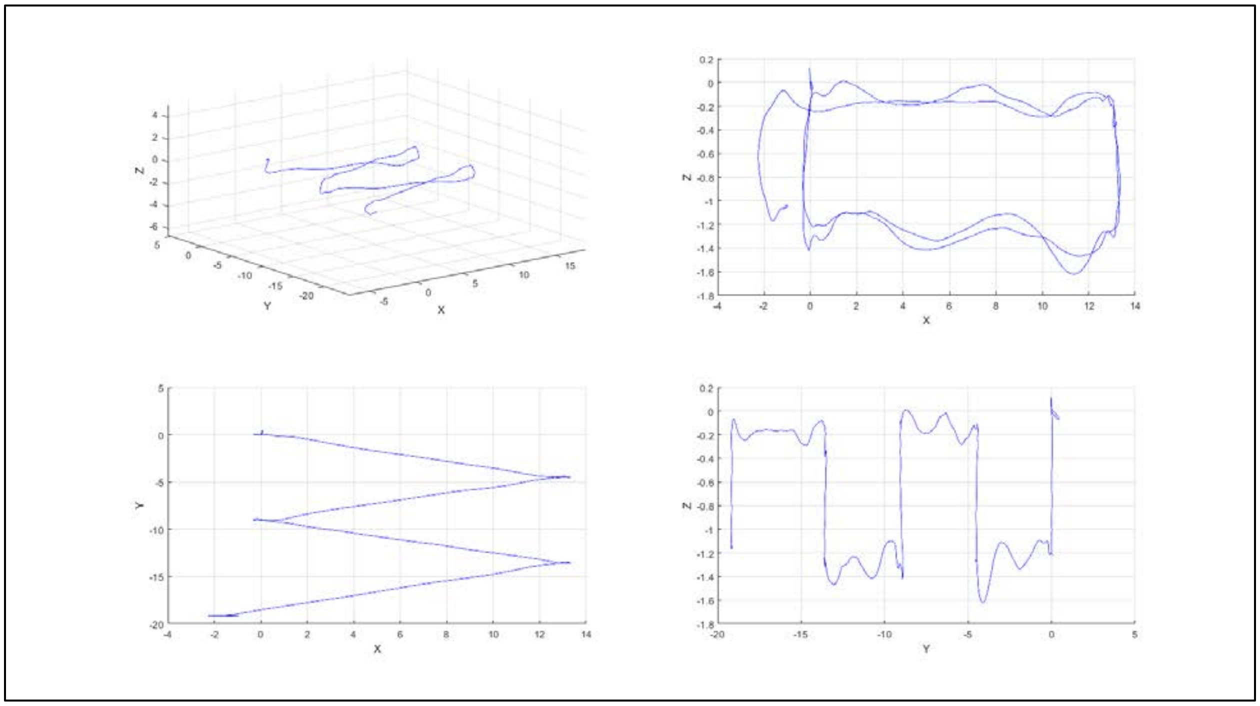
(a) Sequence 01



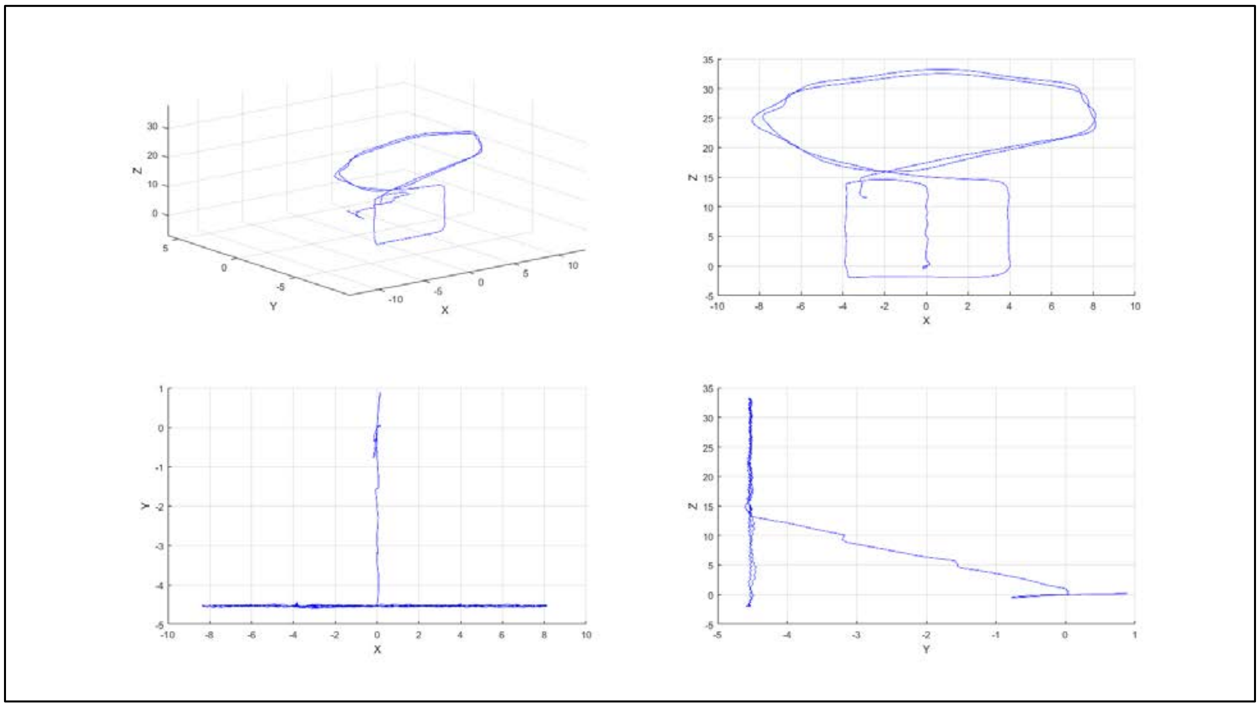
(b) Sequence 02



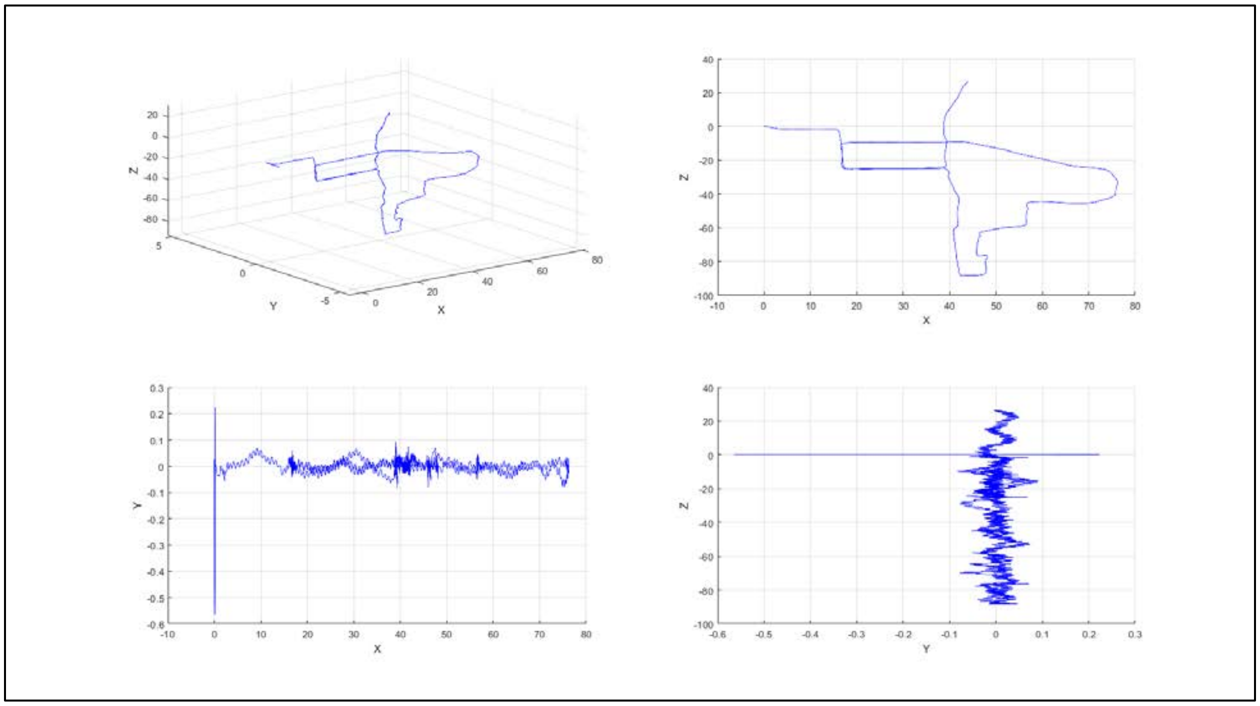
(c) Sequence 03



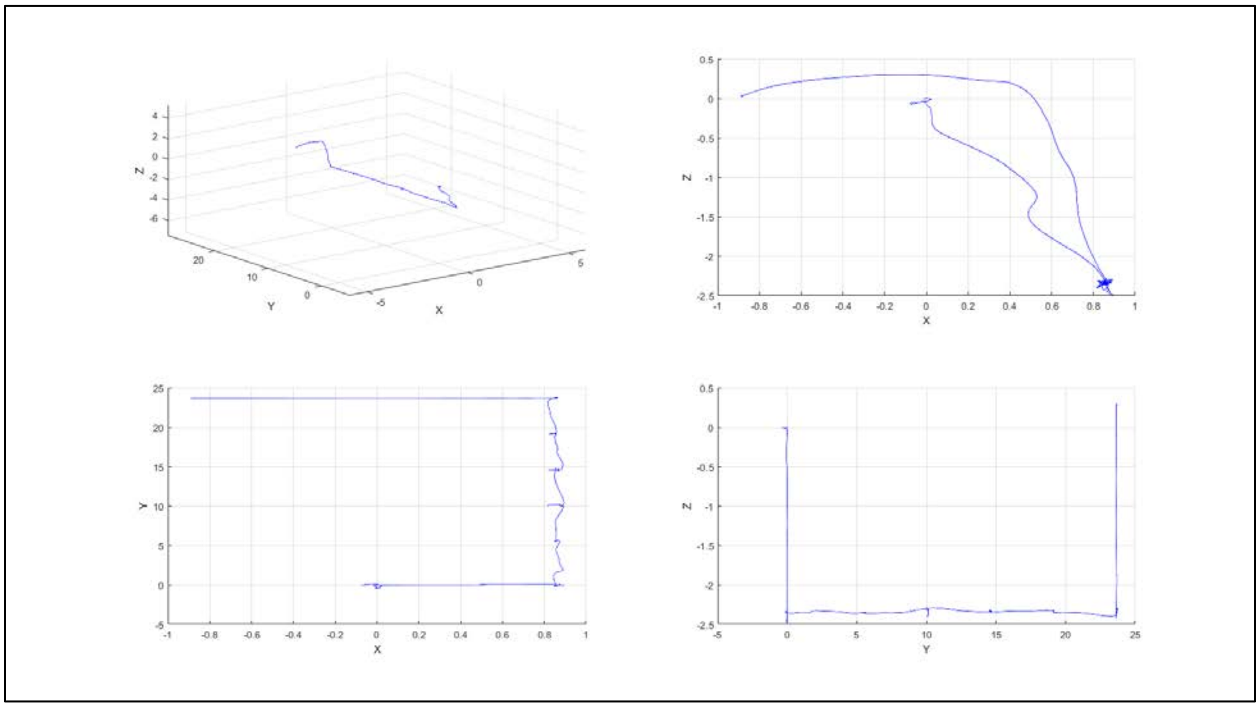
(d) Sequence 04



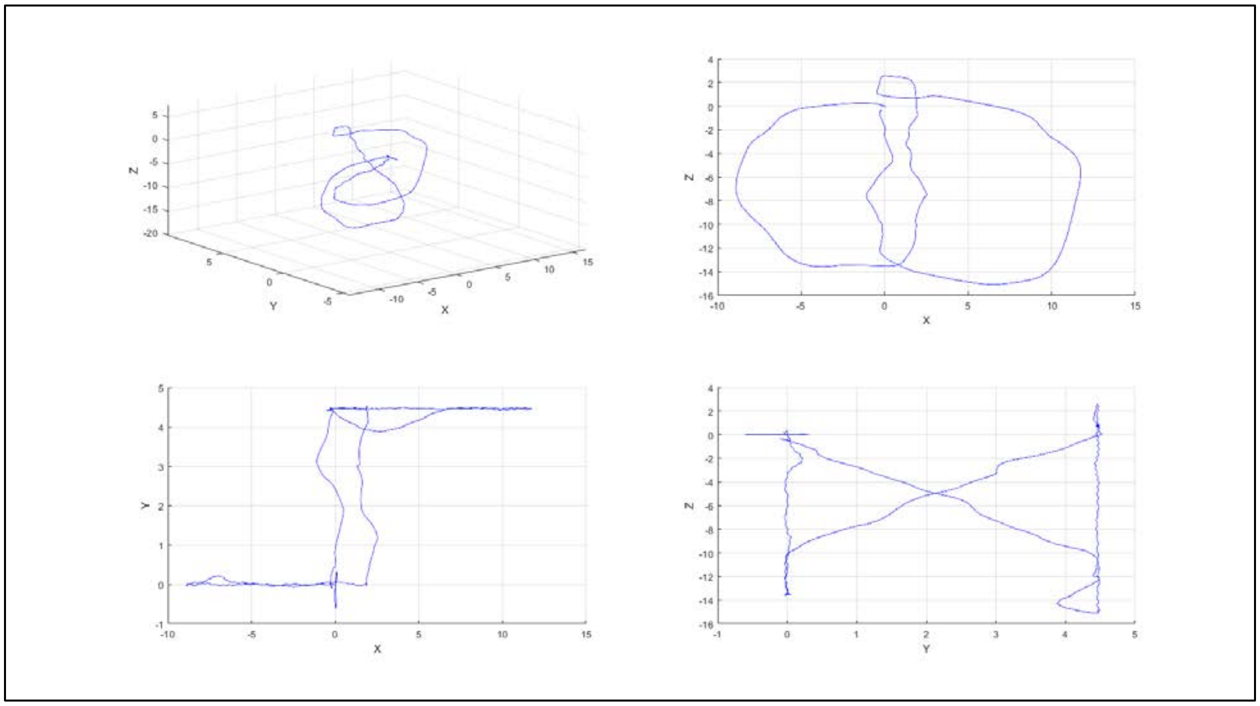
(e) Sequence 05



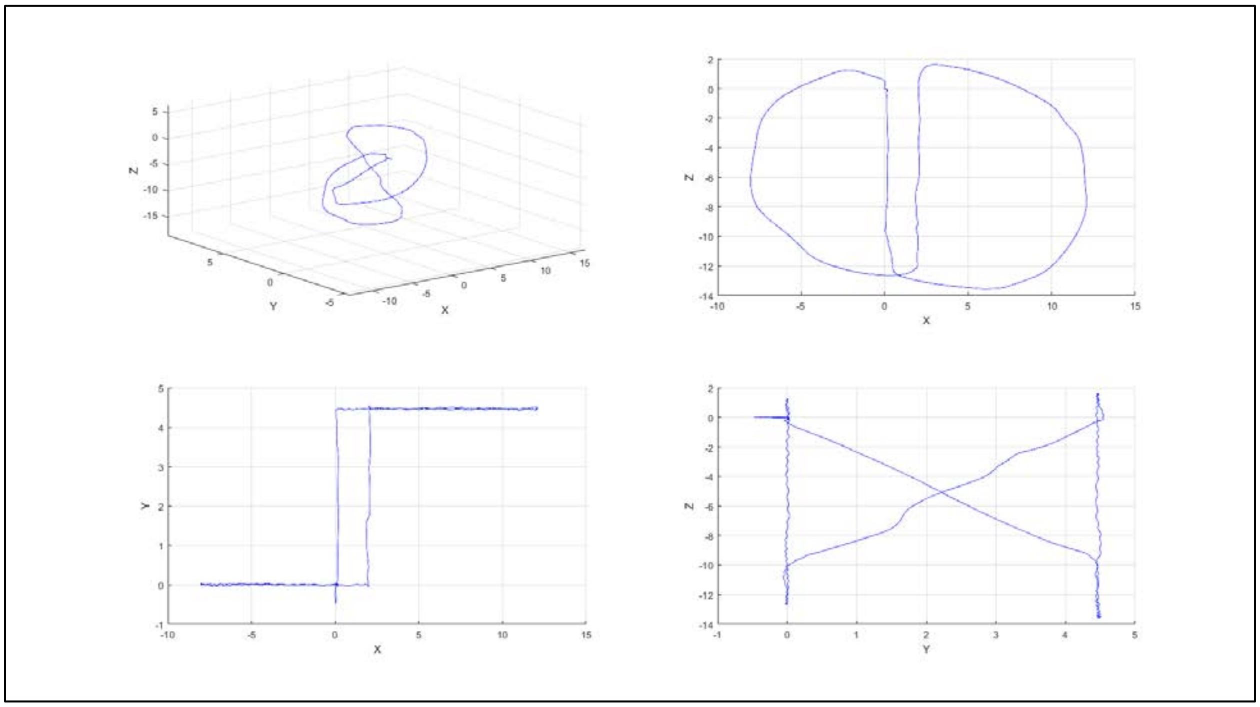
(f) Sequence 06



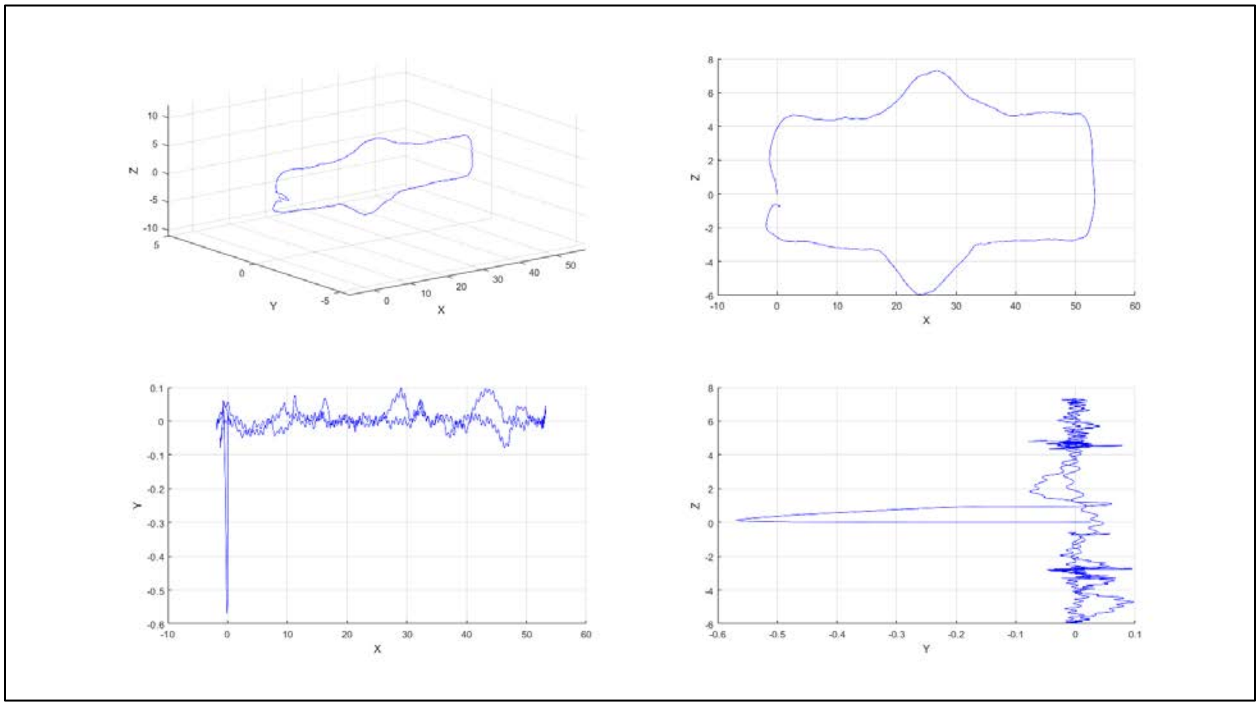
(g) Sequence 07



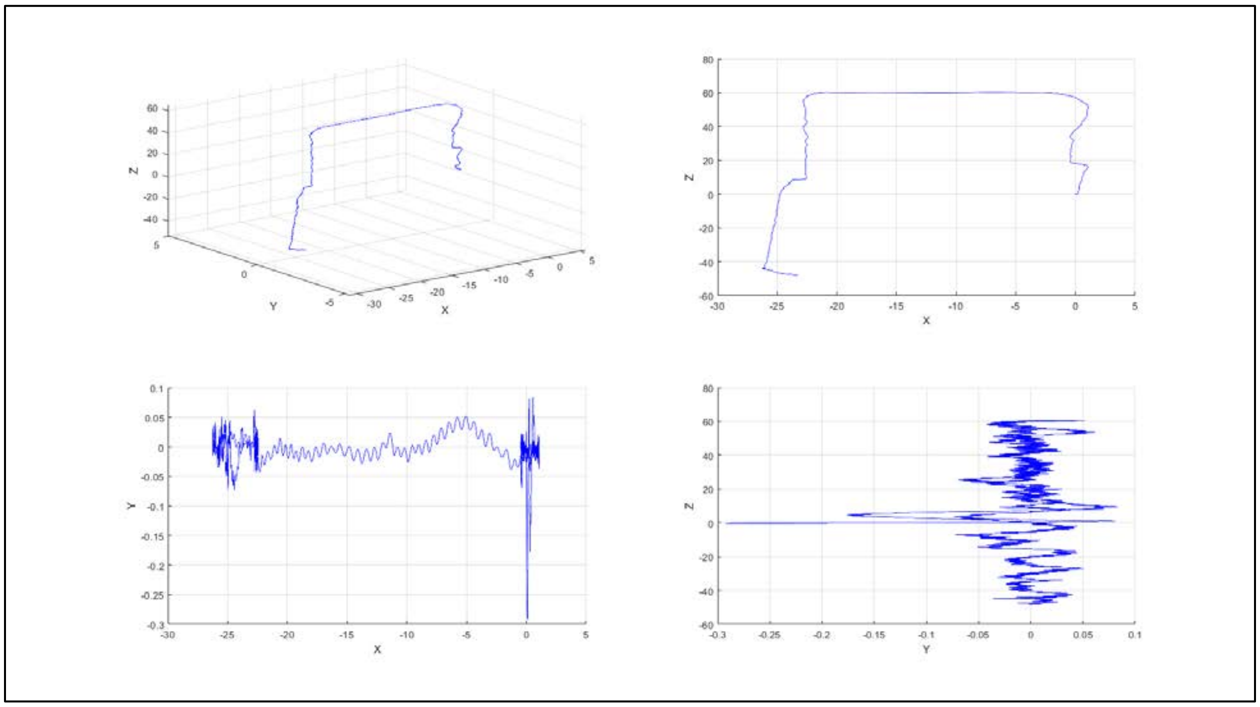
(h) Sequence 08



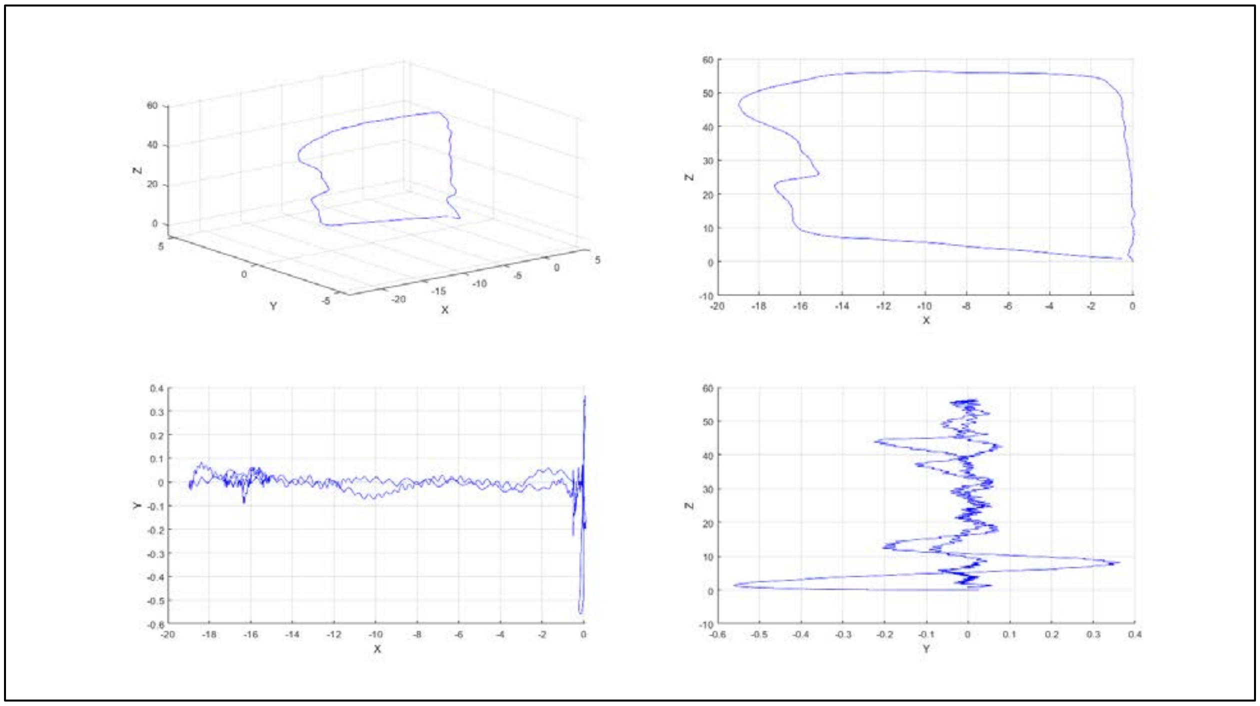
(i) Sequence 09



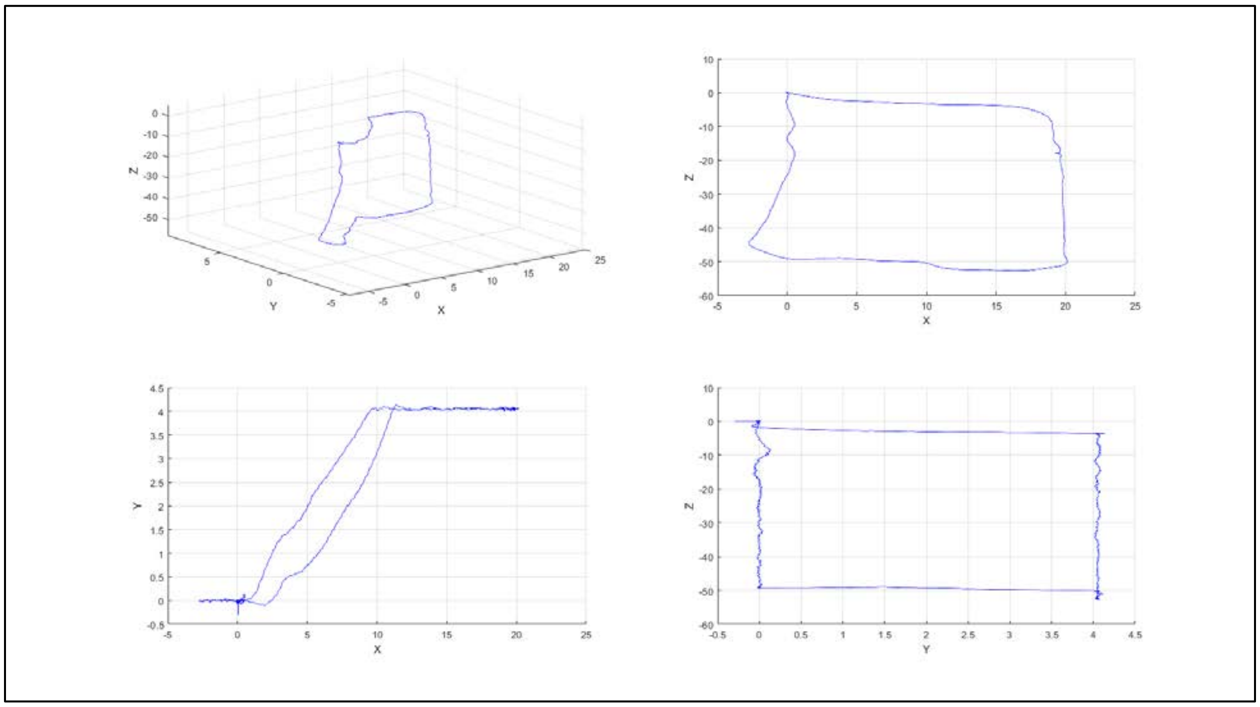
(j): Sequence 10



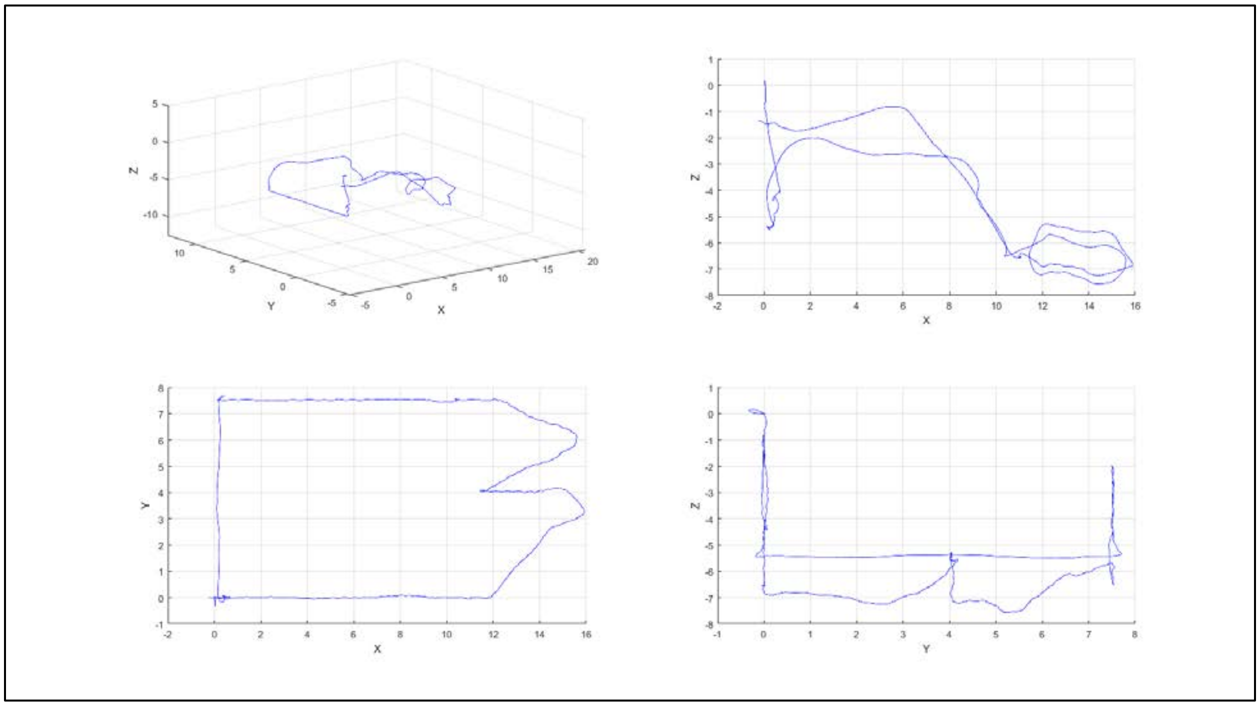
(k) Sequence 11



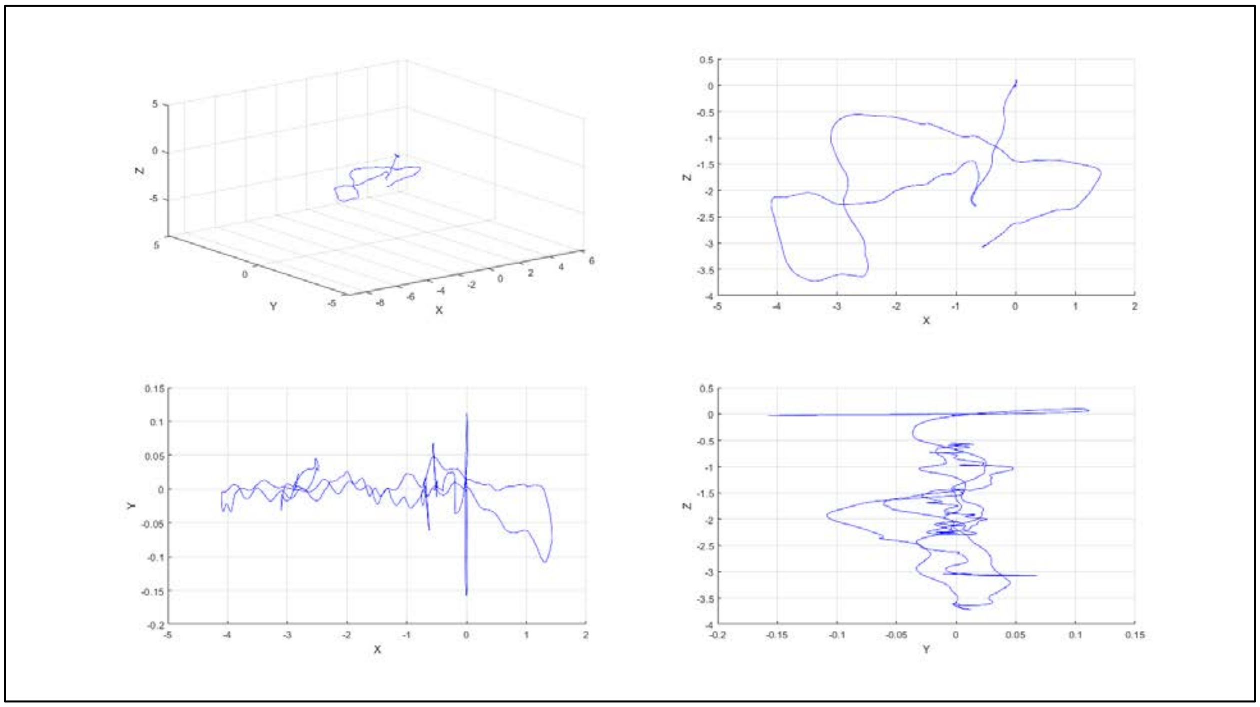
(l) Sequence 12



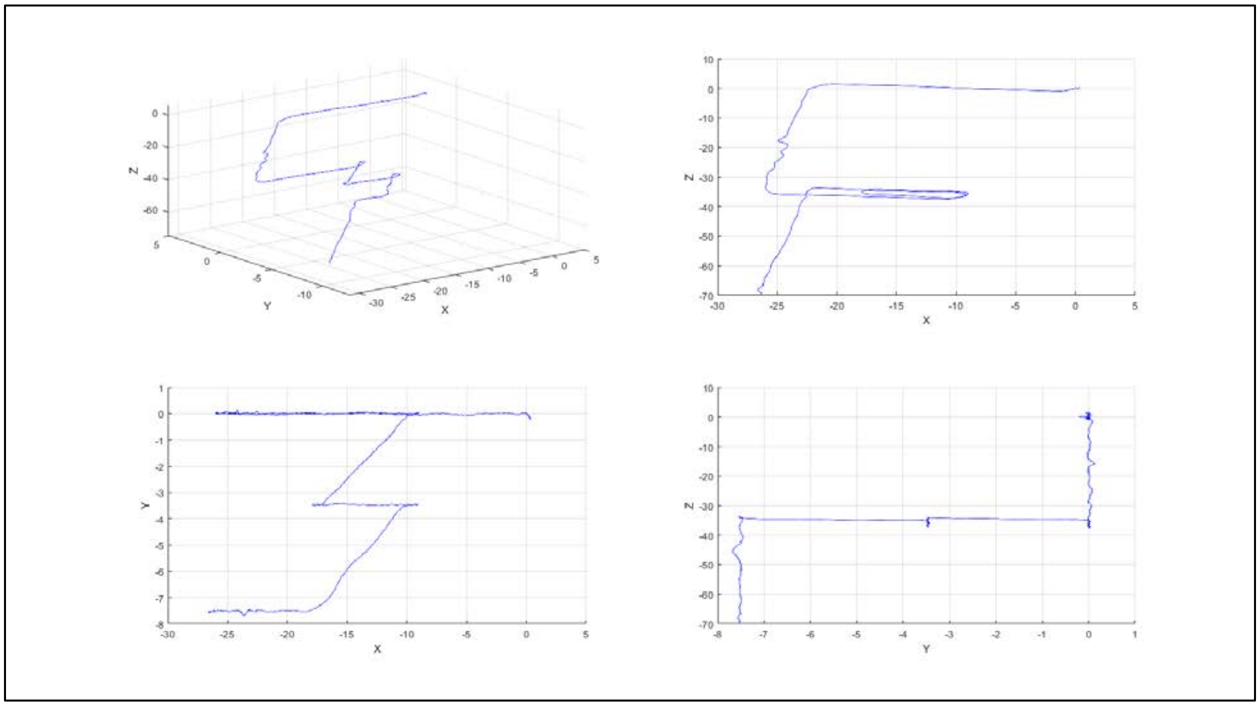
(m) Sequence 13



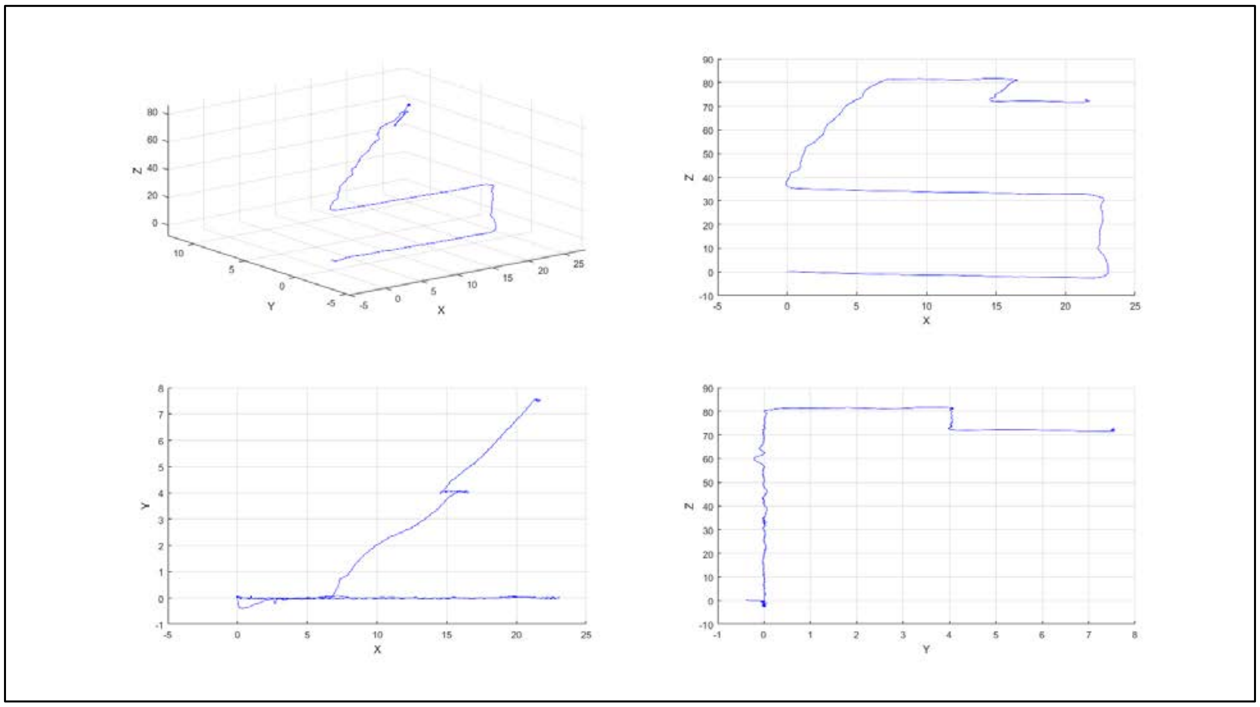
(n) Sequence 14



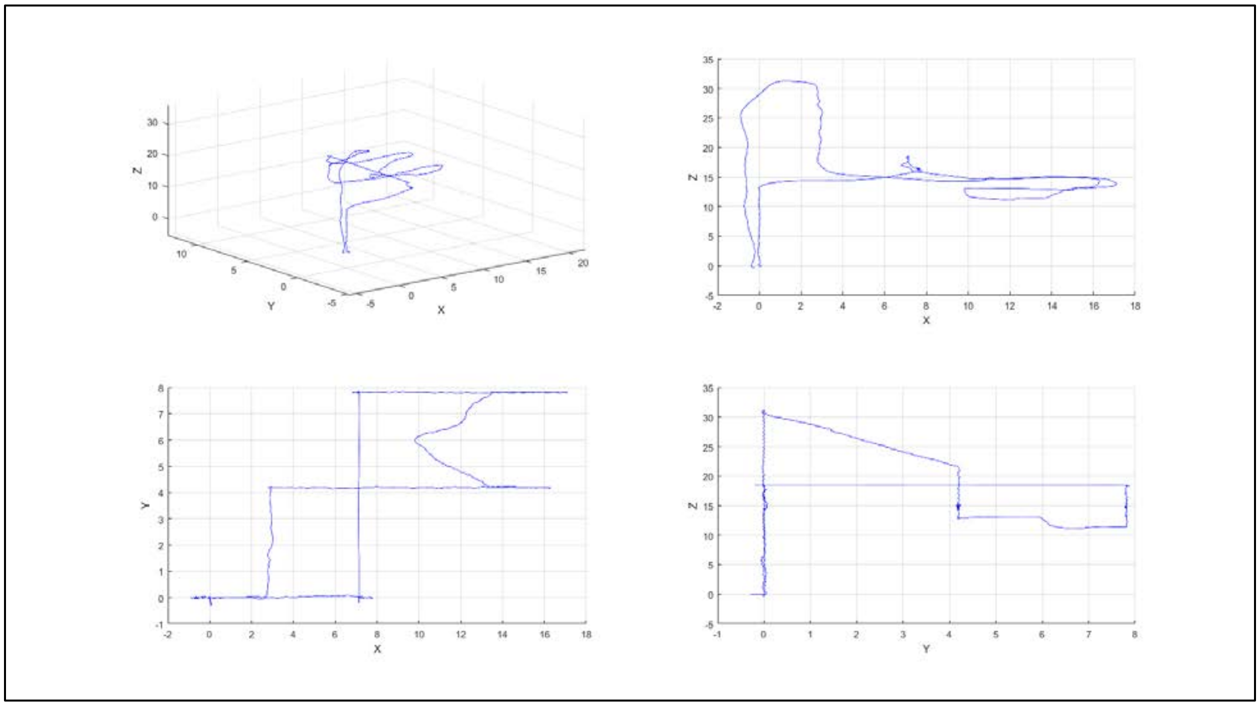
(o) Sequence 15



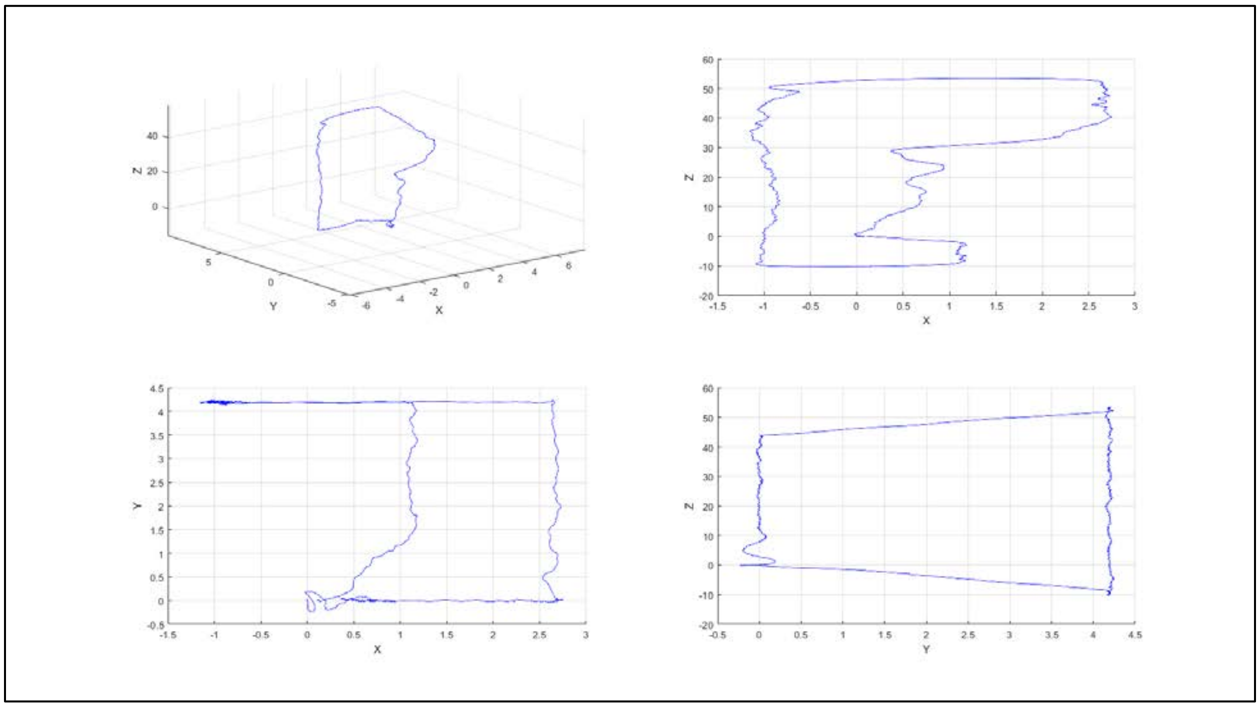
(p) Sequence 16



(q) Sequence 17



(r) Sequence 18



(s) Sequence 19

Figure 3.5: The visualization of pose tracks in different planes

3.4.1.1 Dataset issue

As data is collected separately for each sensor, two synchronizations are needed before their use. The first one involves bringing all timestamps into the same time frame. Although the ADVIO dataset seems to be synchronized, there is an error in the data where the timestamps of frames.csv and arkit.csv are not in the same time frame. This issue causes an offset between the IMU and video data. To fix this issue, the first timestamp from the whole-time vector in those files is subtracted. Moreover, the timestamps for gyroscope, accelerometer, and images are not the same. In other words, for a certain timestamp I do not have all IMU sensor and visual data. I fixed these issues before using condensing the experiments, otherwise there are huge drifts in the data. Since the accelerometer and gyroscope data are in the same time frame, I used linear interpolation to align the inertial data with the frames.

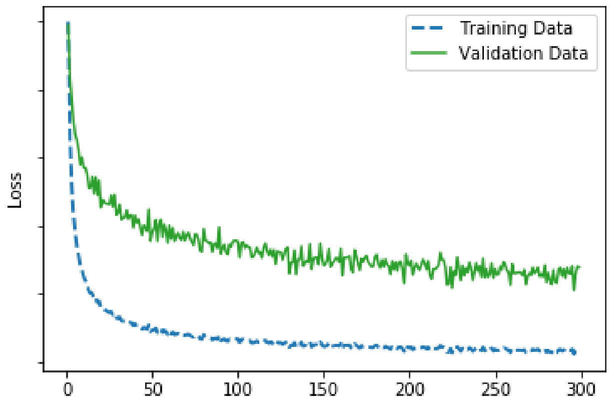
3.4.2 Training and testing

The dataset has 19 sequences of frames for indoor from malls, metros, and offices which are used for training and testing the model. I divided the sequences into two separate groups for training phase and testing phase where experiments are performed to evaluate the framework. The first group is based on the 14 sequences including sequences from all classes of locations (mall, office, and metro) to train the model (five of them) and used the rest for testing the model. Since the ability to generalize well to real data is critical, the next experiment aims to analyze how the framework and the trained models works in an unknown environment. Thus, the five remained sequences which never used for training the model were used for cross validation and avoiding overfitting.

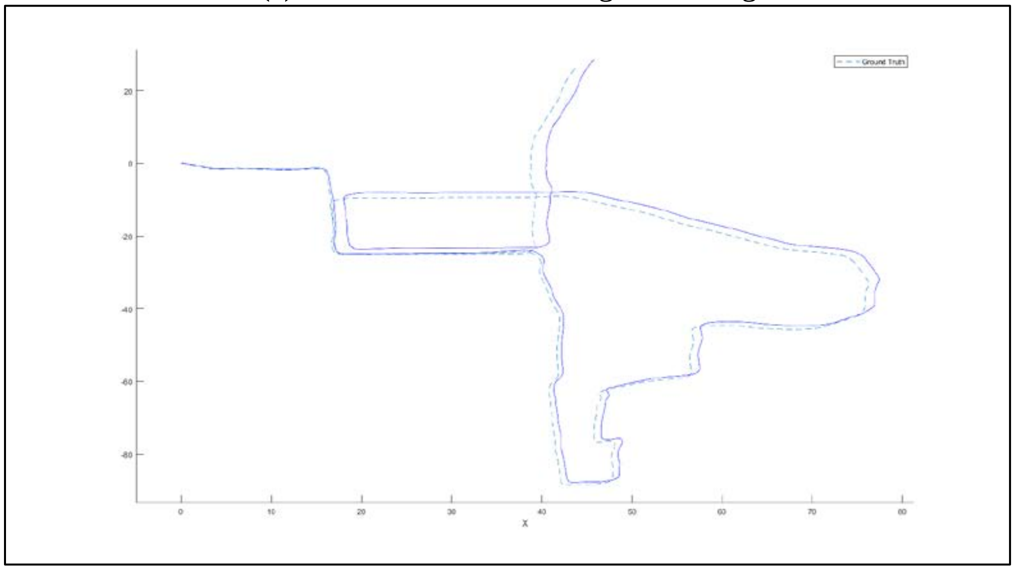
One main issue with training this type of network compared to the conventional convolutional neural networks is that the input is a sequence of images instead of a single image. Moreover, while the number of available images for training CNN-based models is enough for training (millions of images for ImageNet database), such a rich dataset with flexible number of sequences does not, at least currently, exist for this type of RNNs. One approach for training this type of network is proposed by (Wang et al., 2018) where segments of the training sequences are randomly picked. These randomly selected sub-segments have arbitrary lengths with flexible start and end points. This approach is able to generate enough sequences for training the model. I trained the network for up to 300 epochs with learning rate 0.001. In order to avoid overfitting, I used dropout and early stopping techniques. In the next section, I discuss the effect of overfitting and how it impacts the model. Moreover, as the number of available sequences is limited, I use transfer learning for reducing the training time and the required data to converge the training. For this, I used the CNN-based FlowNet model (Dosovitskiy et al., 2015) for CNN which is a pre-trained model.

3.4.2.1 Overfitting effects

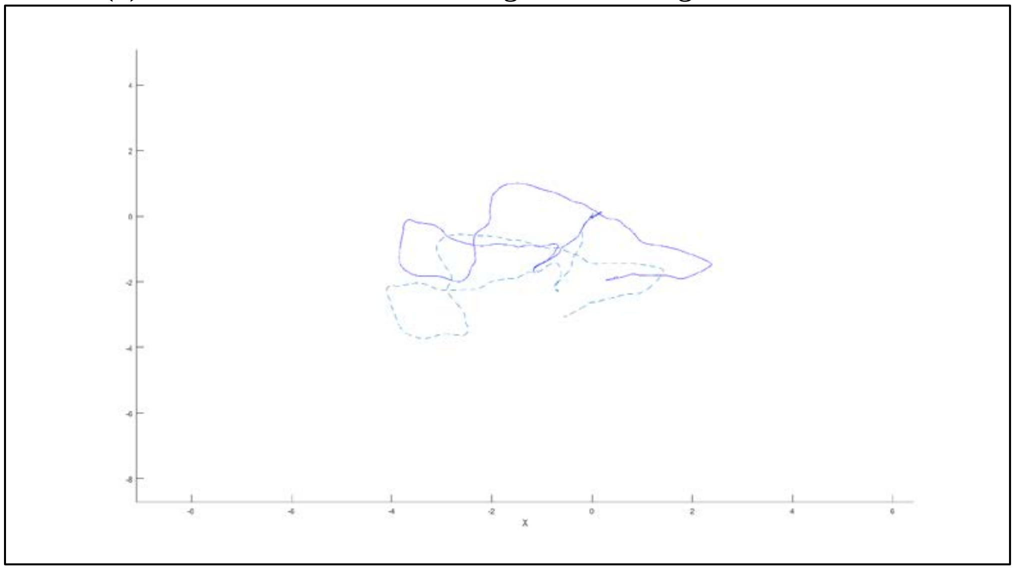
With limited data, it is very likely to have overfitting during the training procedure. Overfitting can have various impacts on the results and here I show how it can influence the outcomes. During the training, I had two cases of overfitting and good fitted models. In Figure 3.6 it can be seen that the gap between training and validation indicates overfitting. Using one of the sequences, which was not used in training as test data shows the overfitted model and how it is useless for generalization. In order to avoid overfitting, I used dropout which is a widely used technique. Figure 3.7 shows the appropriate loss which is a model with good fitting. Although the output of training data for the overfitted model is more accurate, Figure 3.7 shows the results.



(a) Loss function indicating overfitting

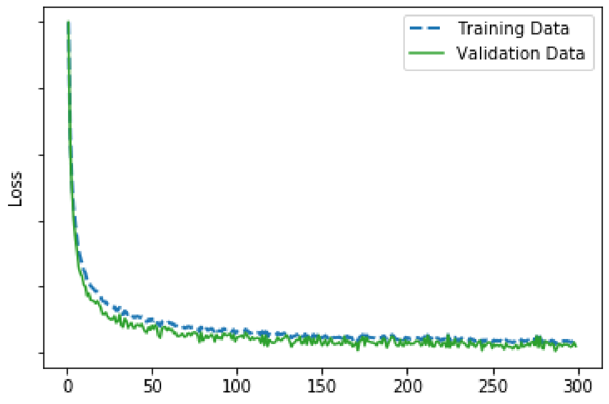


(b) The result of VO for training dataset using overfitted model

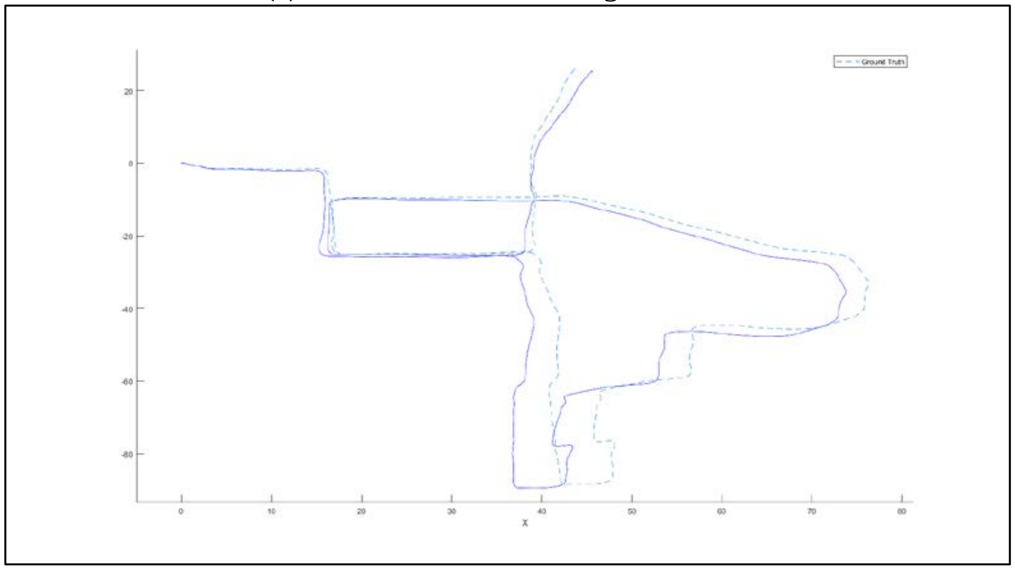


(c) The result of VO for testing dataset using overfitted model

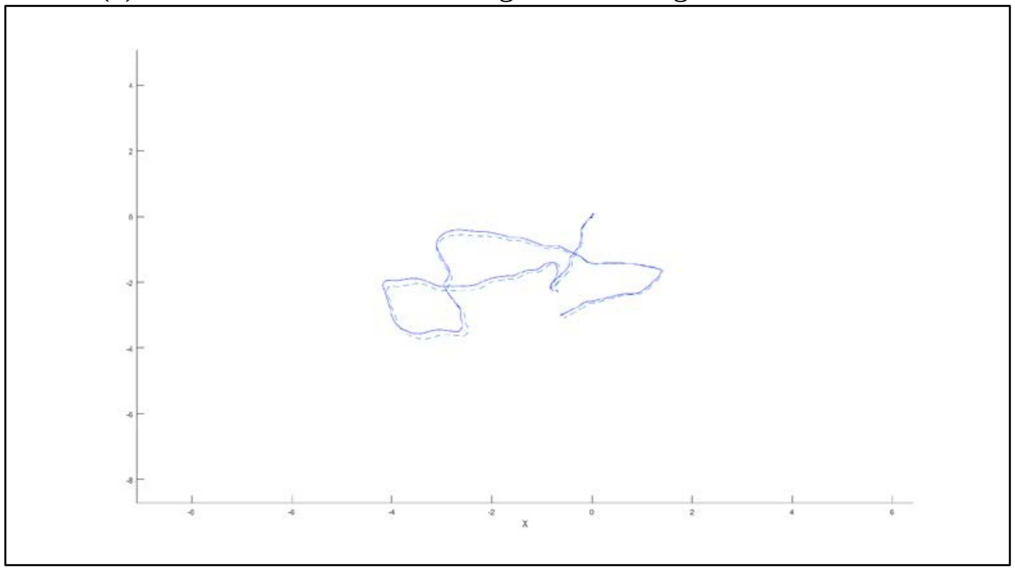
Figure 3.6: Impact of overfitting



(a) Loss function indicating well-fitted model



(b) The result of VO for training dataset using well-fitted model

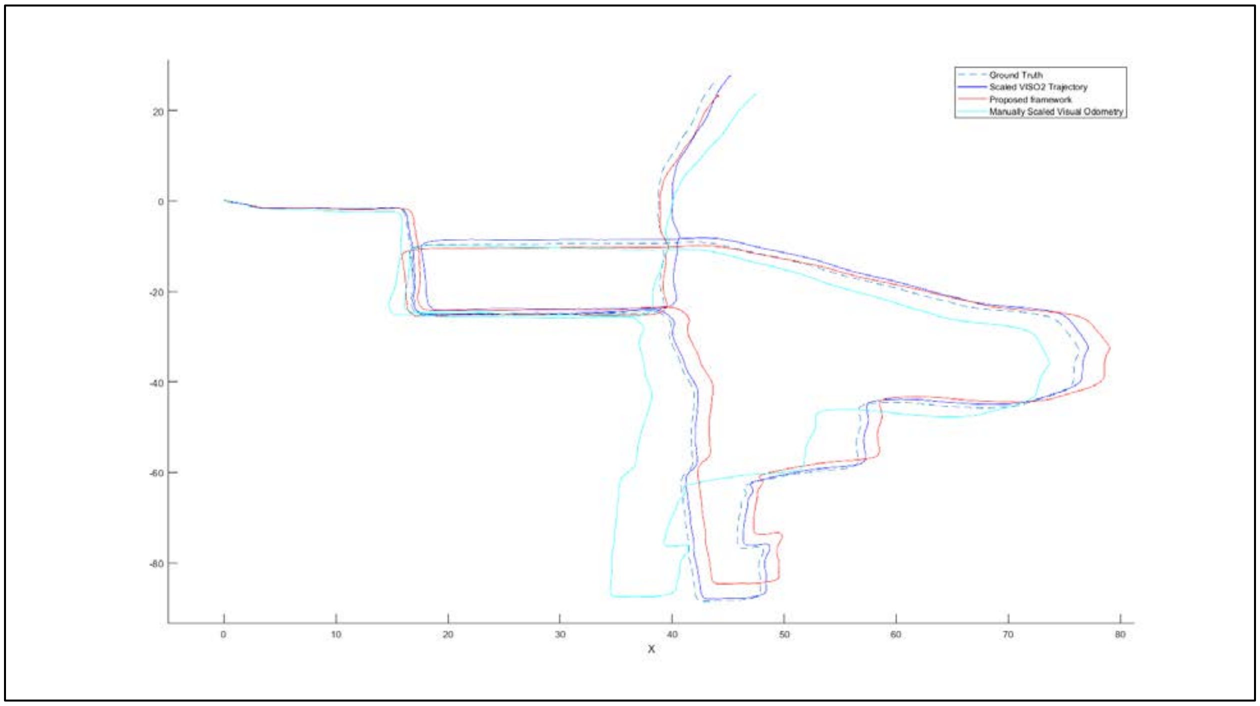


(c) The result of VO for testing dataset using well-fitted model

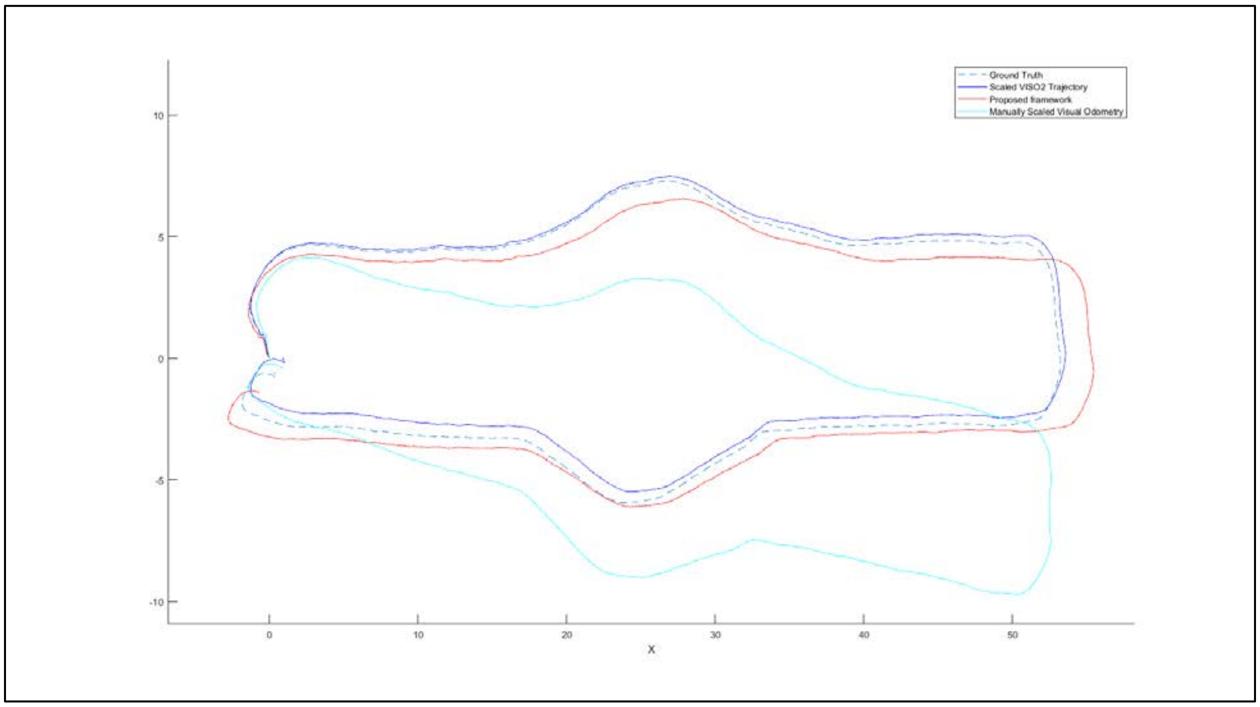
Figure 3.7: A propoer loss function and the impact on training and test data

3.5 Results

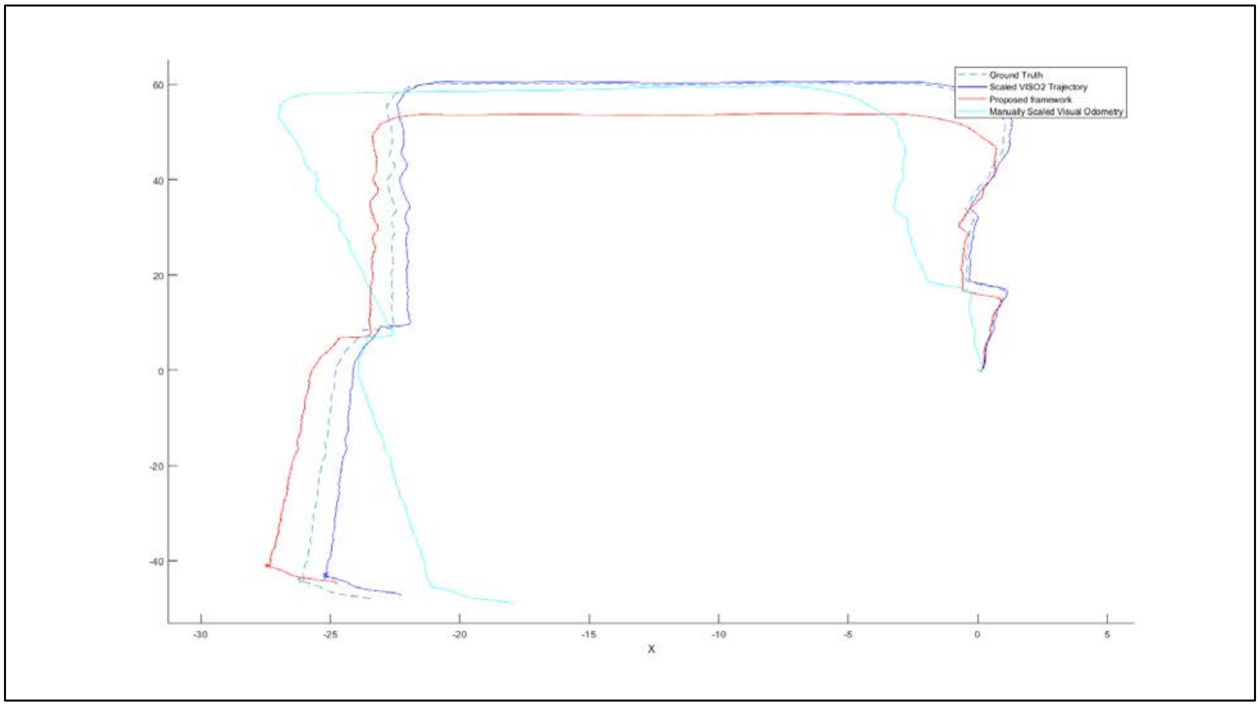
In this section I present the results of comparing positioning and tracking using the proposed framework, the visual odometry with manual scale computation, and a feature-based solution. For the feature-based solution, I used LIBVISO2 (Library for Visual Odometry 2) (Geiger et al., 2011) for monocular odometry. LIBVISO2 uses the 8-point algorithm for estimating fundamental matrix. It assumes that the camera is moving at a known and fixed height over ground for estimating the scale. However, I used an EKF algorithm with inertial data to fine tune the scale factor. For solving the scale of the visual odometry, I considered a set of points and computed an average of scales for manually selected points which are discernible. Figure 3.8 (a) – (e) p show trajectories of the testing sequences (five sequences: 6, 10, 11, 12, and 13) of the ADVIO benchmark with the ground truth. There are four trajectories in each diagram which show the ground truth, LIBVISO2, manually scaled visual odometry, and my framework.



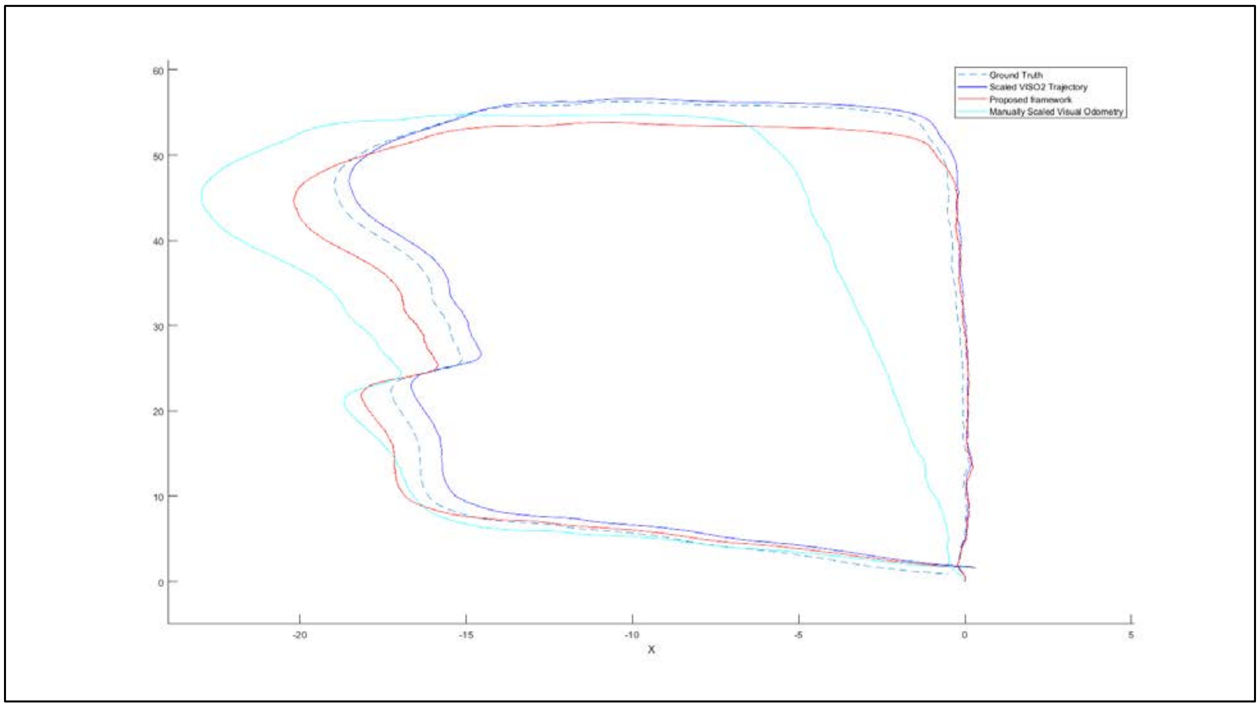
(a) Sequence 06



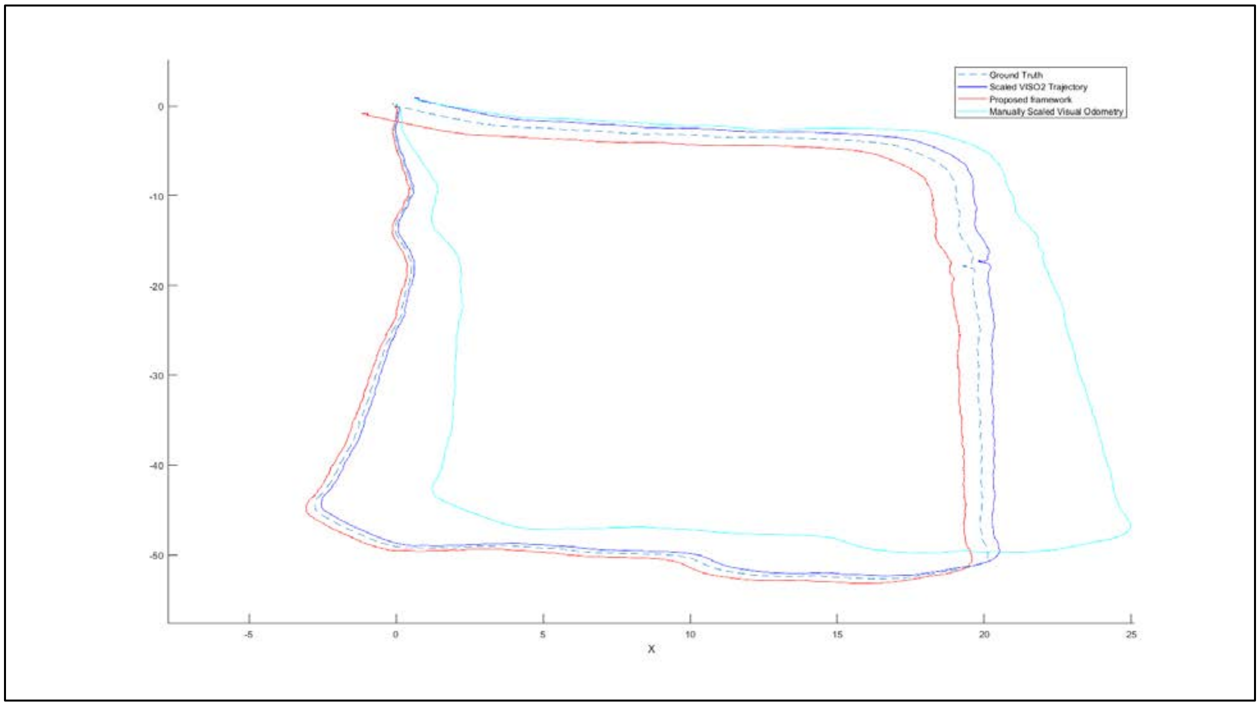
(b) Sequence 10



(c) Sequence 11



(d) Sequence 12

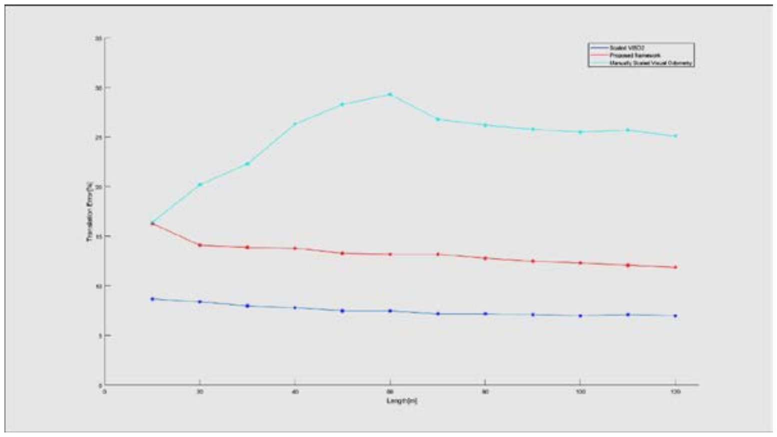


(c) Sequence 13

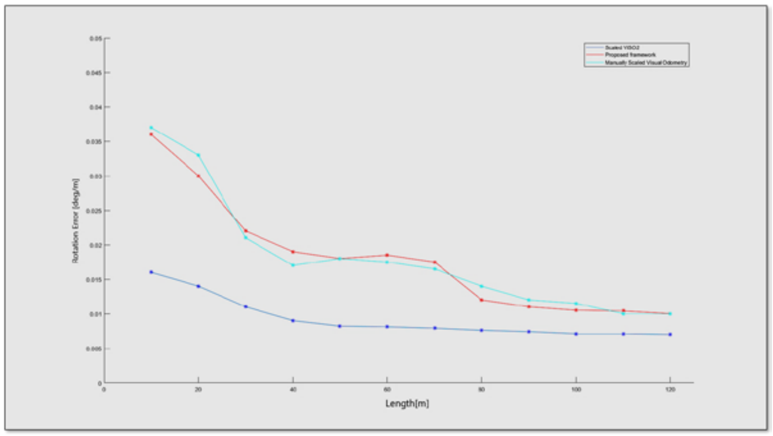
Figure 3.8: Trajectories of testing data for Sequences 06, 10, 11, 12, and 13. Scale in VISO2 model is recovered using inertial sensor data. The red line shows the proposed framework. Manually scaled visual odometry is the result of a monocular learning based model where the scale is manually recovered using one known baseline

3.5.1 Translation and rotation error

In order to compare the methods, I computed the average error of translation and rotation against path lengths measured by RMSE at each time step. Figure 3.9 shows the results of translation and rotation error for sequence 10 which is a test sequence.



(a)



(b)

Figure 3.9: Translation and rotation error for sequence 10

3.6 Summary

This chapter presented a framework for fusing the data from learning-based monocular visual odometry and inertial sensor for managing the problem of indoor positioning in a new approach. In order to find the visual odometry response an RNN was utilized in which CNN is used. In fact, the RCNN is trained to learn estimating 6DoF of each two consecutive frames based on learning not only from two frames, also using the sequence in memory. The results show that the framework is able to solve the problem of positioning and using data-fusion improves the quality of scale estimation and the overall accuracy. However, the quality of the geometry-based method is still better, in fact, these two approaches should not be considered as competitive; they can be seen as complement. For instance, in some scenarios when there are not enough common features to use the 8-points algorithm this technique can be helpful. There are some potentials to take advantage of both of them to make it more accurate and responsive.

There are some challenges in this problem that to the best of my knowledge are still unsolved. The first challenge is integrating learning approach and geometry-based approach where can increase the level of freedom in solving the problem. Moreover, there can be many new approaches to detect loops for managing drift which is a classic problem in SLAM. One major challenge in this approach is dataset. While this data-driven approach needs more sequences, there are just few datasets for this purpose. One decent solution can be utilizing unsupervised and geometry-based approach to solve it.

4.0 Obstacle detection for navigation

One of the main tasks in most navigation systems, particularly the autonomous ones, is to avoid obstacles. In order to steer away from any obstacles, it is necessary that they are detected in time before reaching them. Most obstacle detector approaches are based on supervised training to learn different objects for detecting them and taking appropriate actions once detected. Although these approaches are effective, they are not responsive and reliable enough for indoor environments due to the existence of many different objects, making training almost impossible. Also, the problem becomes more complicated as new objects appear. To address these issues, in this chapter I propose an unsupervised algorithm to detect the regions of the image plane with most likely presence of obstacles.

4.1 Introduction

There are various approaches to tackle the challenge of obstacle detection. Supervised leaning, such as deep learning, is one of the most well-known approaches to effectively detect and recognize obstacles. However, the problem of obstacle detection is more challenging for pedestrians and robots moving in indoor and outdoor environments compared with vehicles driving on roadways. This is because pedestrians and robots face many different types of potential obstacles. This means that utilizing a supervised approach requires the availability of a rich dataset for training a model. In this chapter I concentrate on optical-flow-based obstacle detection which relies on visual data. While most vision-based navigation systems are based on stereo images

(Costa et al., 2012), my proposed approach is based on single image (monocular) for finding the regions on an image plane which are more likely obstacles.

To address the obstacle detection problem, I propose an approach to recognize the user's context, generate a point dataset, and detect obstacles. Figure 4.1 shows the proposed approach for detecting obstacles in a block diagram. The first part of the approach involves determining the frame rate. Frame rate is basically dependent on the mode of activity; therefore, the time interval should correspond to its mode. The second part of the work involves processing consecutive frames. On the image plane, I check a set of sample points to see if they belong to any obstacle. Therefore, the first step in processing the frames is designing a point dataset, which can be based on two different perspectives. The first perspective considers the image texture and decent points to track, which means that the configuration of the points is solely dependent on the properties of the image (El-Gaaly et al., 2013). The other perspective ignores image texture and extracts the points based on a predefined set of points, such as a grid (Tapu et al., 2013).

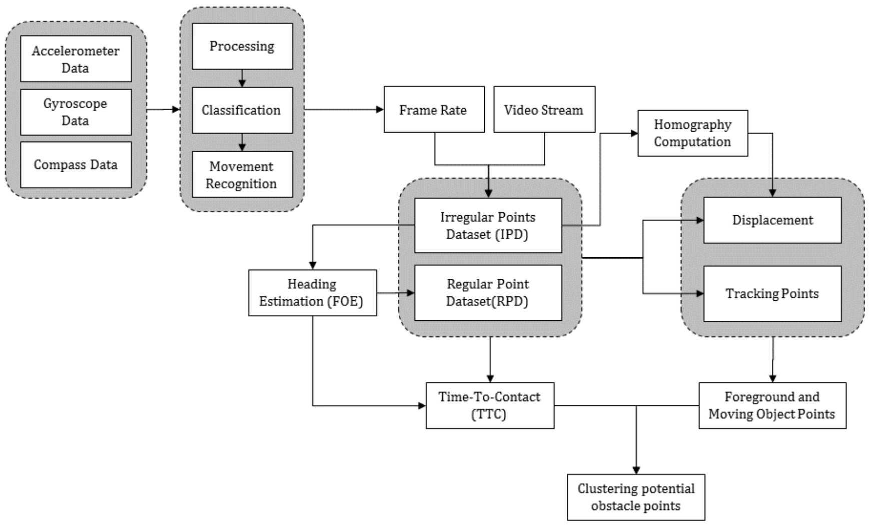


Figure 4.1: Schematic diagram of data processing for obstacle detection

In this chapter, I propose a hybrid method that uses both texture and a regular point set considering the user's movement. This will provide two sets of points. One set is irregularly distributed, based on image texture, and I use an image descriptor for finding the points. The other set of points is regularly distributed, based on the heading and movement of the user. I compute a local displacement or motion for those two sets of points by using the Kanade-Lucas-Tomasi (KLT) feature track algorithm and optical flow technique. The points also have a global displacement, which can be determined by computing a homography matrix. I classify the generated points into background and foreground points by comparing the difference of the motion of the points and their global displacement with a threshold. Finally, I cluster the foreground points, based on time-to-contact (TTC) and motion angle parameters.

In Figure 4.1, I schematically present my proposed procedure that is followed in this research. The figure shows the relationships among the different parts of the procedure in a block diagram. Inertial data stream is processed to extract the most relevant and least redundant features. I use these features to determine the mode of movement using a classifier. Since, frame rate extraction is dependent on the mode of movement, I can obtain the frame rate value as well. Video stream, another data stream source, is also processed based on the determined frame rate to extract consecutive frames which are then used to detect obstacles. In the next steps the extracted consecutive frames are processed. These steps are explained in detail in the following sections.

4.2 Methodology for optical flow-based analysis for potential obstacle detection

With a single moving camera, we need to detect the displacement of the points, due to relative movement between the camera and objects. Optical flow contains information about the velocity of the points, which can be used to detect obstacles (Beauchemin & Barron, 1995). It can be estimated in three different types of analyses: correlation-based, differential-based, and block-based (Beauchemin & Barron, 1995). While dealing with textured backgrounds, utilizing a differential-based type analysis provides more reliable results (Boroujeni, 2012) and it is straightforward to implement. There are two well-known differential-based methods: Lucas-Kanade (Lucas & Kanade, 1981) and Horn-Schunck (Horn & Schunck, 1981). Although Lukas-Kanade does not guarantee either spatial or brightness consistency, it is more effective than other techniques and has the advantage of a substantially lower computational cost (Tapu et al., 2013). Since optical flow tracks pixels or point features over frames of video, the visual processing of the frames initially involves selecting some points and tracking them. There are two main approaches for selecting and tracking those points: dense optical flow (Haiying Liu, Chellappa, & Rosenfeld, 2003) and sparse optical flow (El-Gaaly et al., 2013). By dense, I mean pixel-level features or regular grids of points over the video frames. Since dense optical flow is computationally expensive, it is not considered in this research. In contrast, the sparse technique tracks a number of sparse strongly detected features over multiple frames (such as SURF (Xu & Namit, 2008), SIFT (Lowe, 1999), PCA-SIFT (Ke & Sukthankar, 2004), ORB (Rublee et al., 2011), or Harris Corner detection (Harris & Stephens, 1988b)). However, the sparse technique usually generates just a few points—especially in my case, as the reduced image/frame size influences the number of points. Also, for less textured regions or low-resolution videos, the sparse techniques usually extract few or even no interest points (Tapu et al., 2013). Hence, in order to generate a dataset with sufficient points, I devised an algorithm to extract points, which is discussed in the next section.

4.2.1 Point dataset architecture

I propose an algorithm that uses a hybrid method to extract enough points to process. For this, I define two classes of interest points: irregular point dataset (IPD) and regular point dataset (RPD). IPD consists of the points that are detected by the image descriptor, using the SURF feature detector algorithm, as it is faster than other algorithms (Xu & Namit, 2008). In order to generate RPD, the initial task is to define some areas on the frames where navigation tasks are likely to occur and regions where obstacles are more likely to be found. I call these areas regions of interest (ROI). For the purposes of pedestrian users navigating indoors, the ROI is the area that the user is more likely to pass through next; therefore, it is critical and should be analyzed, regardless of the texture of the image. For the purpose of finding these ROIs, I define a corridor that the user is moving through (Figure 4.2a). Objects in this corridor are more likely to be obstacles that would block the passage of users. Figure 4.2 (b) and (c) show such a corridor, as well as its corresponding ROI on the image. As Figure 4.2(c) highlights, the ROI is a projection of a corridor on the image. As Figure 4.2(c) highlights, the ROI is a projection of a corridor on the frame plane. For delineating this triangular polygon, since the lower vertices are fixed, the unknown vertex is the upper one, which depends on the user’s movement. The user's heading is the upper point of the ROI. Detecting the heading makes it possible to define the ROI, and using optical flow enables the computation of the heading for two consecutive frames (Camus, 1994; O’Donovan, 2005).

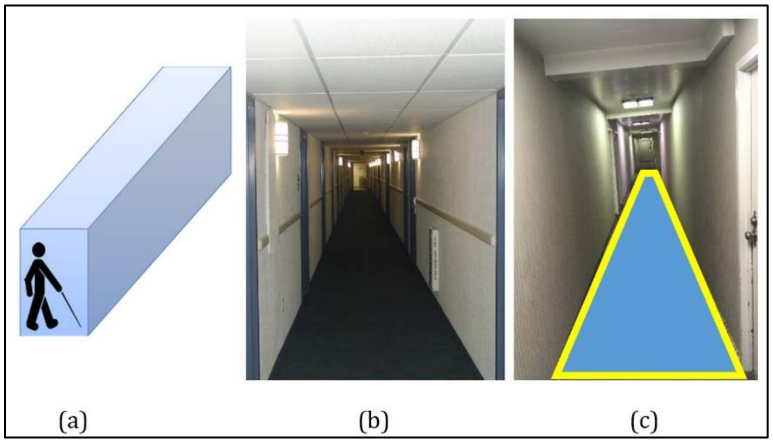


Figure 4.2: (a) Corridor (b) a hallway (c) schematic region of interest.

4.2.1.1 Heading estimation

Heading is defined as the direction of the user traveling through the environment. Translating this concept to an environment results in a radial motion pattern with all motion vectors directed away from a single point (Figure 4.3 and Figure 4.4). That point is the focus of expansion (FOE) which can be obtained by using optical flow. Consequently, in translational movement, FOE can estimate the heading of movement (Browning, Grossberg, & Mingolla, 2009). In other words, the FOE of a motion pattern is situated on the point in the image towards which the observer is moving. To detect the FOE and estimate the heading of the user, I use IPD and calculate the coordinates of the FOE. Tistarelli and Sandini (Tistarelli & Sandini, 1993) used the least squares solution of all flow vectors and used it to find the FOE as follows:

$$FOE = (A^T A)^{-1} A^T \boldsymbol{b} = -\frac{1}{\sum a_{j0}^2 a_{j1}^2 - (\sum a_{i0} a_{i1})^2} \begin{bmatrix} \sum a_{i0} b_i \sum a_{j1}^2 - \sum a_{i1} b_i \sum a_{j0} a_{j1} \\ -\sum a_{i0} b_i \sum a_{j0} a_{j1} + \sum a_{i1} b_i \sum a_{j0}^2 \end{bmatrix} \qquad 4.1$$

where, for each point $p_i = (x, y)$, the associated flow vector $\boldsymbol{v} = (u, v)$ gives $a_{i0} = v$, $a_{i1} = u$, $b_i = xv - yu$

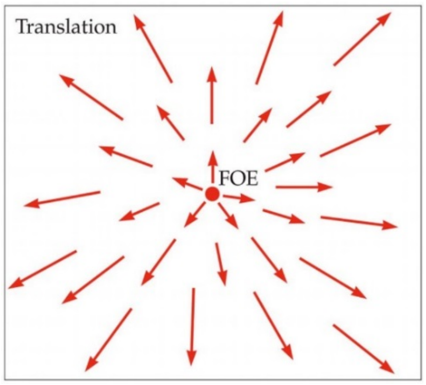


Figure 4.3: Focus of Expansion (FOE)

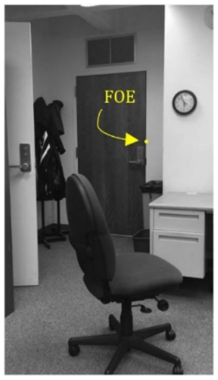


Figure 4.4: Computed FOE

4.2.1.2 Regular point dataset (RPD)

Since FOE estimates the heading, it enables the determination of the ROI on the image plane. For the purpose of enriching the point dataset (including IPD), which was generated by the SURF algorithm, I define a grid over the ROI. The input of the algorithm for extracting the RPD is the number of points for the grid. The algorithm organizes the points in the ROI and returns their coordinates. Thus, the algorithm determines the distance between the points and checks if they are inside the ROI. The grid step is defined as $\Gamma = \left\lfloor \frac{DY}{\sqrt{2N}} + 1 \right\rfloor$, where D is the width of the corridor on the frame and Y is the y coordinate of the FOE, which is the height of the triangle, as computed in the previous step. N is the maximum number of points and the input to the algorithm. To limit the computational cost and obtain a high degree of precision, I set N to 400.

ALGORITHM 1: Point Extraction

```
Data: $frame_{t1}, frame_{t2}, number\,of\,regular\,points$
Result: Set of data Point

$Irregular\,Point\,Dataset(IPD) \longleftarrow SURF(frame_{t1})$;
$N \longleftarrow number\,of\,regular\,points$
$step \longleftarrow \left\lfloor \frac{DY}{\sqrt{2N}} + 1 \right\rfloor$
for $point \in Irregular\,Point\,Dataset$ do
    $OF_{ipd} \longleftarrow OpticalFlow(point)$
end
$FOE(x, y) \longleftarrow Heading(OF_{ipd})$
$(P_1, P_2, P_3) \longleftarrow CoordinatesofROI$
for $i \leftarrow x_{min}$ to $x_{max}$ $(i = +step)$ do
    for $j \leftarrow y_{min}$ to $y_{max}$ $(j = +step)$ do
        if $(i, j)\,isInsideROI$ then
            add $(i, j)$ to $Regular\,Point\,Dataset(RPD)$
        end
    end
end
$Points \longleftarrow IPD + RPD$
```

4.2.1.3 Time-to-contact

Using a single image makes it difficult to obtain information about the depth and distance from different points on the image plane. However, due to having a moving monocular camera, we can get information about time to contact (or collision), which is called the time-to-contact (TTC) parameter. The TTC can be computed from the optical flow (Camus, 1994). This invaluable information plays an important role because it provides data about the third dimension. To compute the TTC, we need to know the FOE, whose calculation was discussed in the previous section using IPD. Figure 4.5 illustrates the geometry of both the FOE and two image planes. Point $p = (x, y, z)$ on image plane at time T is the projected image of point $P = (X, Y, Z)$. The distance of the image plane from the origin is f and the movement of the camera is alongside the Z-axis. At time T+1, point p will be projected onto a new point on the image plane $p' = (x', y', z')$. The following equations, based on simple geometry, show how to obtain $\tau = \frac{y}{\frac{\partial y}{\partial t}}$, which represents TTC.

Therefore, for calculating TTC, we only need the optical values of quantities y and $\frac{\partial y}{\partial t}$. In other words, the optical flow provides $\frac{\partial y}{\partial t}$, and the distance from the FOE is y.

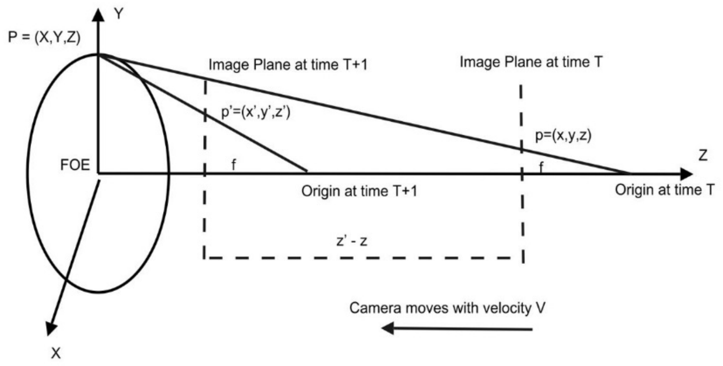


Figure 4.5: Projection of point P onto the image plane of a moving camera (Camus, 1994)

$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z} \Rightarrow \frac{\partial y}{\partial t} = f\frac{\frac{\partial Y}{\partial t}}{Z} - fY\frac{\frac{\partial Z}{\partial t}}{Z} \qquad 4.2$$

As we assume translational movement, $\frac{\partial Y}{\partial t} = 0$, $Y = \frac{yZ}{f}$. Assuming $\frac{Z}{t} = V$ we have

$$\frac{y}{t} = -y\frac{V}{Z} \Rightarrow \frac{y}{\frac{\partial y}{\partial t}} = -\frac{Z}{V} = \tau \qquad 4.3$$

4.2.2 Tracking point dataset and displacement computation

One essential feature in detecting obstacles is determining the displacement of the point dataset by tracking the points in consecutive frames; in this research, these are called motion vectors. For the purpose of computing the motion vector, I use KLT feature tracker. Moreover, I calculate the global geometric transformation between two frames to compute the displacement due to frame transformation. To detect homography transformation, I use two corresponding point datasets on *frame*$_t$ and *frame*$_{t+1}$. The homography transformation, shown by matrix **H** in Equation 4.4, models the transformation, and could be calculated using a least-square adjustment if corresponding pairs of points in two point datasets are detected (Figure 4.6). I use the RANSAC (RANdom SAmple Consensus) algorithm (Fischler & Bolles, 1981) to find corresponding pairs of IPDs on two frames (Figure 4.9). In other words, given matrix **H**, I can map the points in each IPD at which each point on frame$_{t+1}$ is the result of transformation of the same point on frame$_t$ using matrix **H**. In the following equation, p'_i is an estimation of position p_i in the second frame.

$$p'_i = \boldsymbol{H}.p_i \tag{4.4}$$

$$\begin{bmatrix} x'_{2i} \\ y'_{2i} \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_{1i} \\ y_{1i} \\ 1 \end{bmatrix} \tag{4.5}$$

where $w = \frac{1}{(h_{31}.x'_{2i} + h_{32}.y'_{2i} + h_{33})}$

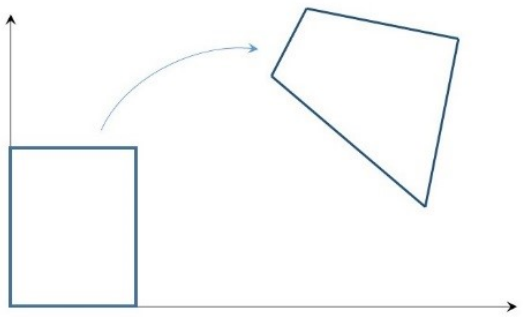


Figure 4.6: Schematic diagram of homography transformation

Next, I discuss vector and geometric displacement computation. For each point in the second frame, I have two pairs of coordinates. The first one is the motion vector and the other pair is the projected vector. Ideally, these two points should be as close to one another as possible. The distance between these two pairs is used as a criterion for defining an error. All moving objects and the stationary object close to the camera are part of the foreground for moving camera, and their movement in the second frame is more than other objects. Therefore, if the error does not exceed the threshold, it is considered to be part of the background. I concentrate on the points that do not belong to the background. In other words, those outliers whose error exceeds the threshold belong to foreground objects and are potentially part of an obstacles. The following equation by Tapu et al. (2013) shows the error value:

$$E(p_i^{t1}) = \left\| p_i^{t2} - \boldsymbol{H}.p_i^{t1} \right\| \qquad 4.6$$

where p_i^{t2} shows the coordinates of point *i* in the frame at time *t2* and $\boldsymbol{H}.p_i^{t1}$ is the projected coordinates of p_i^{t1} which is point *i* at time *t1*.

4.2.3 Clustering

I divide the points into those in the foreground, which are potentially part of an obstacle, and those in the background, which are not. I cluster the foreground points and ignore those in the background. The clustering of the foreground points has two steps. In the first step I compute the related parameters for the purpose of clustering, such as angle of motion, which was proposed by Tapu et al. (Tapu et al., 2013), and use the TTC as my proposed parameter. As I do not know the number of obstacles, I use an agglomerative clustering technique (Cimiano, Hotho, & Staab, 2004), considering the maximum cluster of five possible obstacles.

In the second step, I refine the clustering results. Some of these points might be incorrectly clustered; therefore, I have to reconsider the result of clustering to avoid any unintentional erroneous clustering. For the purpose of refinement, a k-nearest neighbors (kNN) algorithm (M.-L. Zhang & Zhou, 2005) is used for each point, and if half of the points for each k neighbors are in the same cluster, that point would be accepted, otherwise it would be ignored. I use Euclidean distance to define neighbor points

4.3 Results

In this chapter, I show the results of the obstacle detection approach using my proposed algorithm for designing the point dataset. I validated my algorithm through experiments in different buildings where metrics of precision, recall, f-measure, and accuracy are all used to compare my algorithm to sparse points and predefined grid point datasets. I also evaluated and compared the outcomes of considering

In this section the results of some data processing and analyses are shown. Figure 4.7 is a schematic diagram showing how regular points are set in ROI. Figure 4.8 shows the IPD and RPD over an image using Algorithm 1 demonstrating the point extraction process. I also demonstrate the results of my algorithm in three different spaces. Figure 4.10 shows the design of point dataset which is generated based on my algorithm (IPD and RPD). The blue points represent RPD and IPD is shown in yellow. Due to different headings in the three scenarios, RPDs are in different configurations.

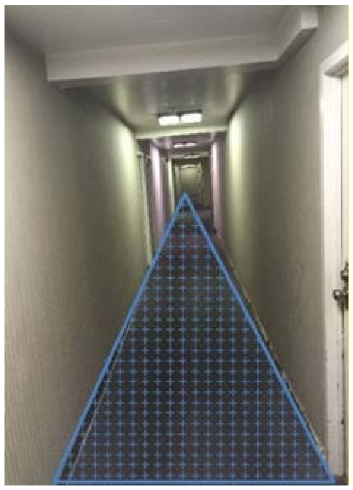


Figure 4.7: Schema of Region of Interest computed by FOE coordinates

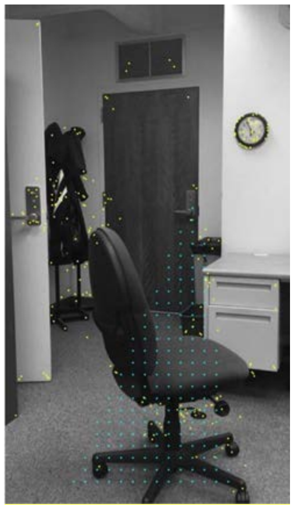


Figure 4.8: IPD and RPD points extracted using Algorithm1

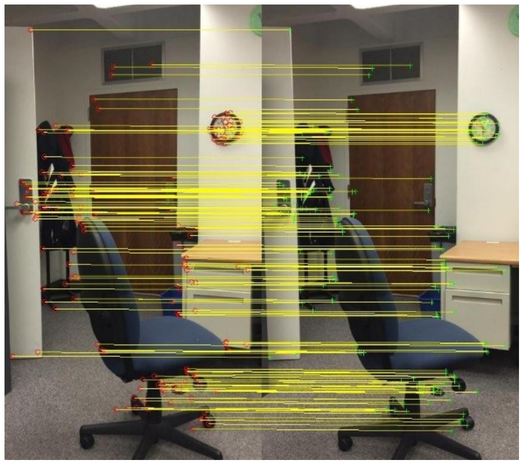


Figure 4.9: Finding corresponding points on the images using RANSAC algorithm

As IPD is based on properties of image, some low textured areas like walls have no or just few points. In Figure 4.10b, the second column shows the result of clustering without refinement. I perform agglomerative clustering detecting clusters and Figure 4.10 shows the refined clusters. I demonstrate the results of my algorithm in three different spaces. Figure 4.10a shows the design of point dataset which is generated based on my algorithm (IPD and RPD). The blue points represent RPD and IPD is shown in yellow. Due to different headings in the three scenarios, RPDs are in different configurations. As IPD is based on properties of image, some low textured areas like walls have no or just few points. In Figure 4.10b, the second column shows the result of clustering without refinement. I perform agglomerative clustering detecting clusters and Figure 4.10c shows the refined clusters.

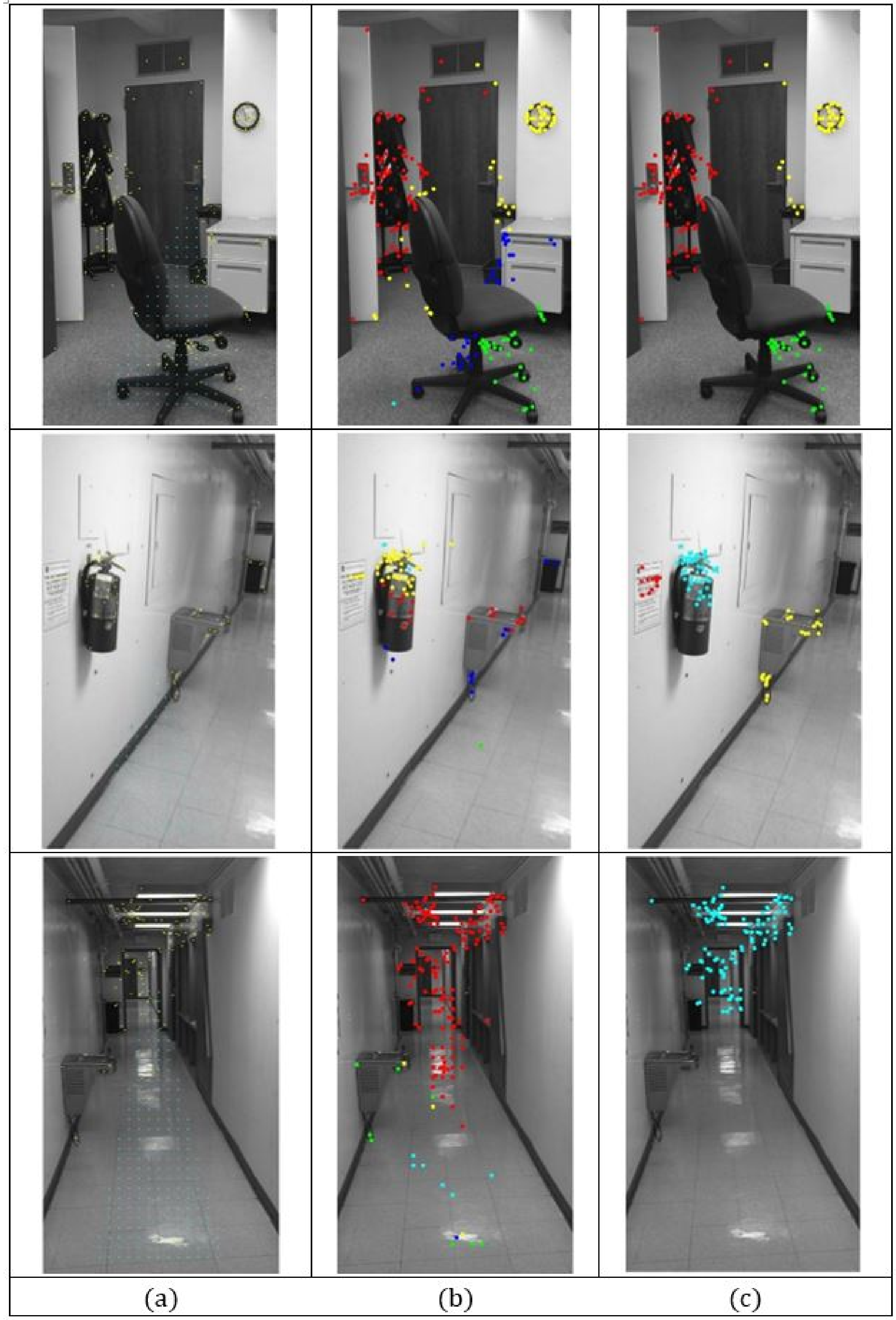


Figure 4.10: Three different scenes for detecting obstacles

In this section, I discuss an experiment to evaluate the precision, recall, f-measure, and accuracy of my algorithm in comparison to two other algorithms: one that used a sparse point dataset and another that used a predefined grid. I do not test the dense point dataset at the pixel level, due to a large number of points (around 330,000 in my experiment) that need to be analyzed; my algorithm needs around 200 regular points and around 400 irregular points, which is less than 2% of the dense-point dataset. The number of points in the predefined grid is the same as the number of points that I use for my algorithm. Although the predefined grid does not use IPD for tracking points, it still needs to extract features to compute homography transformation; hence, there is no advantage to using the predefined grid, as compared with the other two algorithms.

In order to experimentally evaluate the results of my algorithm, its performance is compared with the other two algorithms. In the experiment, I took video at a certain frame rate and analyzed some consequent frames. In the frames, I have many different objects, some of which are obstacles. Intuitively, a higher density of clustered points over these obstacles is preferable, thus, I define polygons around each obstacle in a scene and create them manually. In essence, each clustered point on an obstacle indicates a correct result. Accordingly, if all contributing points are clustered inside any of those polygons, I consider them to be true positive (TP). The points on other objects outside the polygons indicate objects that are not obstacles, therefore, they are classified as false positive (FP). The omitted points are either in the background where they are ignored before clustering, or they are not in the background but are ignored during the refinement procedure. I consider dropping these points as negative. Omitted points outside the polygons are true negative (TN). Consequently, a value of false negative (FN) indicates the ignored points inside the polygons. The precision, recall, accuracy, and f-measure are calculated as follows:

$$precision = \frac{TP}{TP + FP} \qquad 4.7$$

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \qquad 4.8$$

$$recall = \frac{TP}{TP + FN} \qquad 4.9$$

$$f - measure = \frac{2TP}{2TP + FP + FN} \qquad 4.10$$

I tested the designed experiment for 70 videos in 15 different indoor environments. For each video, I tested many different consequent frames. The experiments were conducted in different locations, including the Geoinformatics Laboratory, School of Computing and Information, and the Cathedral of Learning at the University of Pittsburgh; inside a residential building; and in a Giant Eagle Market District grocery store (Figure 4.11). I selected these indoor spaces according to various criteria, such as illumination, image texture, and being busy or not. For example, the study area and the hallways of Cathedral of Learning were dim where lack of illumination dramatically reduces the number of IPD. Using RPD in such locations is more useful. In contrast, the grocery store was too bright and there were some reflections on the floor, as well as other surfaces such as walls, which are incorrectly detected as objects by the algorithm. Moreover, the hallway of the residential building was less textured than other places such as the grocery store which significantly influenced the number of IPD.

My algorithm used 400 regular points and the number of IPD was dependent on the scene, which was considerably high in grocery stores, with highly textured frames (around 750), and in the residential hallway, with slightly textured frames (around 150). I computed precision, recall, and accuracy for all the scenarios, and the means are shown in Table 4.1. I also considered the harmonic mean of precision, recall, and f-measure. All these measures (precision, recall, accuracy, and f-measure) indicate that although the sparse method is more precise than the predefined grid, the predefined grid is more accurate and returns better recall and f-measure values. The sparse method could be even less precise in the case of decreasing the image size, or when I use the technique in a less textured area. My algorithm shows better value for precision, recall, accuracy, and f-measure than the two other algorithms; in particular, the precision of my algorithm is significantly higher than that of the other two algorithms.

For evaluating the role of the TTC parameter and to see if it improved my clustering, I repeated the same experiment two times, using my algorithm to detect obstacles, one time with TTC and one time without TTC. For each case, I conducted the experiment considering TTC and the repeated the experiment without it Table 4.2 shows these results, which indicate that TTC slightly improved the clustering. It is worth mentioning that the time-to-contact (TTC) parameter is computed using equation $\tau = \frac{y}{\frac{\partial y}{\partial t}}$. The absolute value of TTC for each point indicates the needed time to contact the corresponding object point assuming that the user continues to move at the computed velocity $\left(\frac{\partial y}{\partial t}\right)$. In other words, the velocity of movement between consecutive frames remains constant while approaching the object. Since in practice user's velocity for each pair of frames is different, I take advantage of relative values of TTC where the TTC values for points on distant objects are larger than the TTC values for points on close objects.

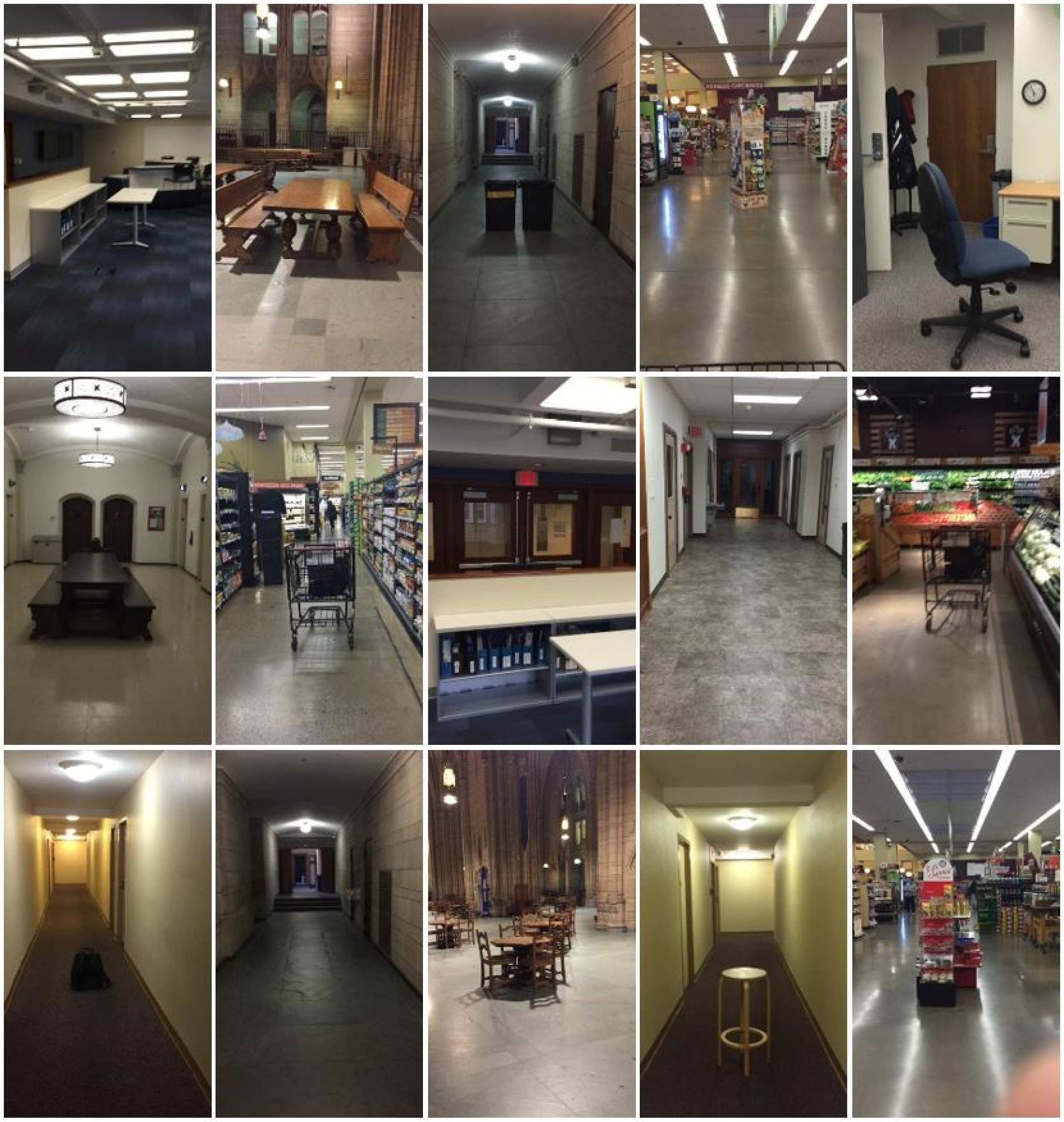


Figure 4.11: Fifteen different indoor settings used to validate the proposed algorithm

Table 4.1: Performance of different algorithms

	Precision	Recall	F-Measure	Accuracy
Sparse points Dataset	69%	43%	53%	65%
Predefined Grid	43%	61%	55%	72%
Our Algorithm	**82%**	**69%**	**72%**	**79%**

Table 4.2: Performance of clustering considering time-to-contact

	Precision	**Recall**	**F-Measure**	**Accuracy**
Clustering with TTC	84%	69%	73%	84%
Clustering without TTC	**82%**	**69%**	**72%**	**79%**

According to these results, my point extraction and tracking algorithms improve the detection of obstacles using optical flow to avoid obstacles. This approach was able to detect moving objects and stationary obstacles close to the camera. my algorithm for extracting the points was based on the image's texture and on the movement of the user. I conducted an experiment to show that this kind of point extraction can improve both the efficiency and precision of the results. I also demonstrated ways in which TTC can improve the overall clustering. Although this is a slight improvement, TTC could be used for some further analyses, such as prioritizing the detected obstacles. Moreover, I used multiple frame rates, according to the user's activity.

4.4 **Summary**

This chapter presents a context-aware smartphone-based visual obstacle detection approach to aid visually impaired people in navigating indoor environments. The approach is based on processing two consecutive frames (images), computing optical flow, and tracking certain points to detect obstacles. The frame rate of the video stream is determined by using a context-aware data fusion technique for the sensors on smartphones. Through an efficient and novel algorithm, a point dataset on each consecutive frame is designed and evaluated to check whether the points belong to an obstacle. In addition to determining the points based on the texture in each frame, my algorithm also considers the heading of user movement to find critical areas on the image plane. I validated my algorithm through experiments by comparing it against two comparable algorithms. The experiments were conducted in different indoor settings and the results based on precision, recall, accuracy, and f-measure were compared and analyzed. The results show that, in comparison to the other two widely used algorithms for this process, my algorithm is more precise. I also considered time-to-contact parameter for clustering the points and presented the improvement of the performance of clustering by using this parameter.

5.0 Sensor fusion and mobile sensing for context-aware keyframe detection

In this chapter movement analysis is introduced not only as a tool to more effectively solve indoor navigation challenges, also for detecting gait changes to understand movement of users. Mobile sensing and computing can be used for detecting the context of user and state of the moving platform. The sensor output is time-series which should be processed to infer the state. In this study I process the signals as the features of signals must be computed and the effective and influencing features must be selected for comprising the input vector to the model. I used this approach for frame detection, and it is explained in detail in the rest of this chapter.

5.1 Introduction

As the standard frame rate of the video stream provides redundant data when processing frames for positioning or detecting obstacles, we need to set the frame rate efficiently to avoid extra computations, especially if it is supposed to be deployed on limited computational resources of mobile devices. As it is discussed in Chapter 2, a straightforward method uses a constant rate; However, at the cost of computational overheads it is inefficient. A more efficient method is determining various rates based on a user's activities. Retrieved data from smartphone sensors such as accelerometers, gyroscopes, and magnetometers has been used in previous work (Bao & Intille, 2004; Gharani & Karimi, 2017; Ming Ren & Karimi, 2012; Sara Saeedi et al., 2014) to detect the state of a user's activity. In these cases, data fusion techniques are used for the recognition analysis.

Our focus in this chapter is on preparing the visual data for the problem of positioning and localization in an effective way by reducing computational cost. Positioning and localization are essential functions for any navigation system on moving platforms such as robots, vehicles, and even pedestrians who uses a smartphone as a navigator. To tackle the problem of pose estimation and tracking, simultaneous localization and mapping (SLAM) (Dissanayake, Newman, Clark, Durrant-Whyte, & Csorba, 2001) and odometry (Nistér, Naroditsky, & Bergen, 2006) are two common approaches which are widely utilized in robotics. SLAM is a technique to build a map of its surrounding while localizing itself with respect to the prepared map. Odometry is also a solution to estimate just the moving platform' position based on aggregating displacement with the last known location. These techniques used to be heavily relied on range-bearing sensors (laser range finder, radar, etc.), however, in recent years with the advent of cameras and becoming ubiquitous, several visual odometry (VO) and visual SLAM (vSLAM) framework have been launched by researches. In the next section the role of keyframes in VO and vSLAM is discussed.

5.2 Keyframes in visual odometry and SLAM

In VO and vSLAM, the trajectory is topologically modeled by a graph where each node shows a significant location that has been visited by the moving object. Moreover, in vSLAM landmarks in environment is modeled as arcs where represents the connectivity between physical locations which can be spatial or visual. In VO and vSLAM determining when and where those nodes should be introduced into the topological graph is a challenge which is keyframe detection. In other words, keyframe detection is an essential module to guarantee visual coverage and avoid disruption while having a simple representation and efficient computation.

As it is discussed in Chapter 2, keyframe detection for vSLAM and VO can be managed based on fixed or adaptive time interval, constant distance sampling, image similarity, or entropy measurement. It is important to notice that in all these approaches researchers consider a correlation between appearance change and the considered parameter. For instance, assuming moving within a static environment, for an interval Δt we can expect to have enough appearance change. This means the change is big enough to have a long distance between two consecutive frames while they have decent amount of overlap and common features enabling us to mathematically model the transformation. This is the same for distance change. Therefore, it can be said that change of appearance is assumed to be proportional to change of time and displacement.

$$\delta_{appearance} \propto \Delta t \tag{5.1}$$

$$\delta_{appearance} \propto \Delta d \tag{5.2}$$

This assumption cannot be consistent for all cases, because change of appearance in same time interval can be very different in environments with different complexities. This is also true for displacement. Hence, measuring similarity seems to be the most significant approach for keyframe detection. The major problem with the appearance-based approach is the cost of computation. In appearance-based approaches and entropy measurement, we must process all frames to see if it exceeds the threshold or not for deciding if it is a keyframe. Since keyframes, as visual data, should be extracted from video frames, which are usually captured from 30-120 fps videos, processing all of them would be computationally too expensive for real-time positioning solutions.

5.2.1 Appearance change detection with inertial data

As it is assumed that the environment is predominantly static, it is a reasonable conclusion that the change of appearance in consecutive frames is majorly due to camera translation and rotation. Therefore, if we can measure relative translation and rotation of two consecutive frames, we should be able to estimate the amount of similarity change. Accelerometer and gyroscope are two common sensors that are used in inertial navigation for positioning and using their data enables us to compute changes in translation and rotation. We do not need necessarily compute translation and rotation and even find the correlation between image similarity change and inertial data as a function is sufficient for solving this problem. This hypothetical unknown function f or g which leads to estimating the amount of appearance change would be like this:

$$\delta_{appearcane} = f(\delta\omega, \delta\varphi, \delta\kappa, \delta x, \delta y, \delta z) = g(S) \qquad 5.3$$

Where $\delta\omega, \delta\varphi, \delta\kappa, \delta x, \delta y, \delta z$ are the rotation and translation with respect to projection coordinate system (Figure 5.1) and $S = \{S_k\}, k \in \{1, \cdots, K\}$ for K different types of input sensors. For instance, $a_x, a_y, a_z, v_x, v_y, v_z$ represent components of two sensors ($K = 2$), accelerometer and gyroscope which measure acceleration and rotational velocity. These are the most common sensors in navigation.

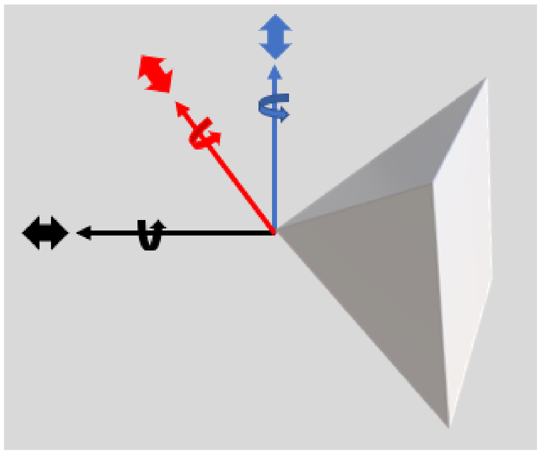


Figure 5.1: Projection coordinate frame and all possible displacement

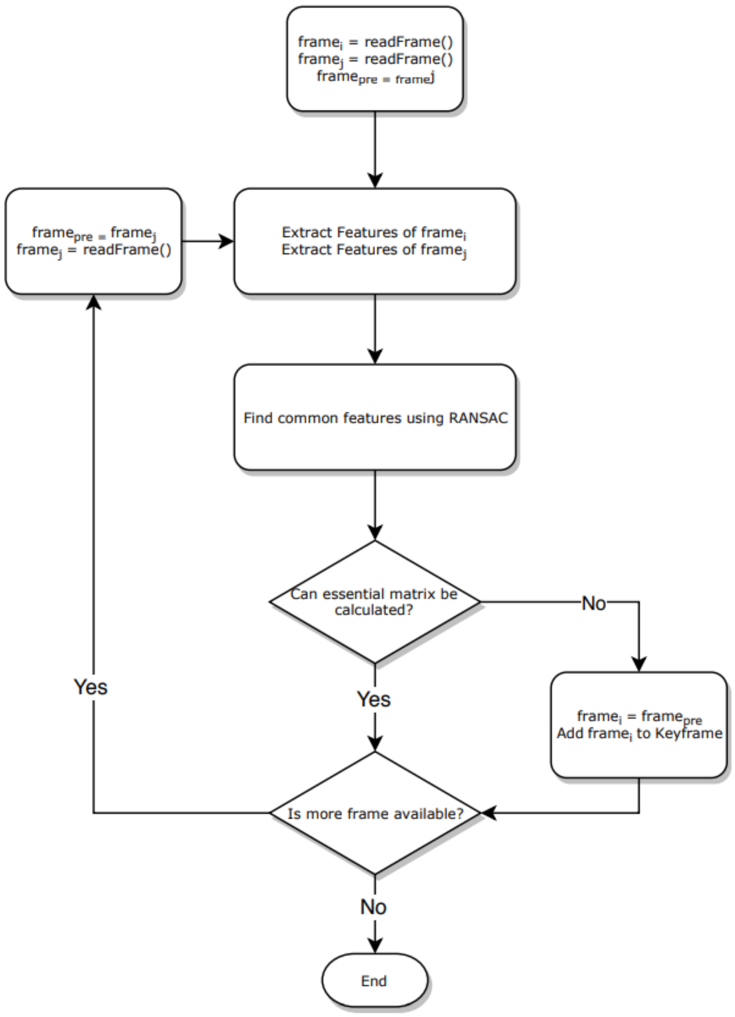
Although it would be very useful if we can define a function which maps the inertial data into image similarity change due to camera motion, defining such a function is too difficult, or even impossible, because the amount of visual change is not only because of translation and rotation. In fact, despite the assumption of a static environment, the level of visual complexity of surroundings has a significant impact on the amount of change in similarity. For instance, visual change of consecutive frames while walking in a prairie is very different compared to the visual change when walking in a grocery store with exact same amount of movement.

In this section, I introduce a learning-based approach with inertial sensor input for detecting keyframes for visual odometry or visual SLAM. The proposed approach to solve this issue is using a supervised learning classification algorithm for training a model based on "effective" keyframes, which are the most appropriate frames for the mentioned purpose, in sequences within various environments. This model gets inertial data as input and the output is the class of frame-rate extraction. In other words, for each trajectory we have a sequence of frames $\mathcal{F}_i = \{f_1^i, f_2^i, \dots, f_n^i\}$ which has a subset of effective keyframes $\varkappa_i = \{\kappa_1^i, \kappa_2^i, \dots, \kappa_m^i\}$ where $\varkappa_i \subseteq \mathcal{F}_i$ and m $\leq n$. Having $\varkappa_i$with their timestamps enables us to determine the rate of extraction for each two consecutive frames which is a correlated to motion and ambient environment. This extraction rate vector forms the output of the machine learning method. In this approach, defining the target vector or frame rate extraction depends on the set of keyframe which is explained in the next section.

5.3 Effective keyframes and framerate extraction

Having a full sequence of frames for a trajectory enables us to process all of them for finding those consecutive ones with least similarity while computing the essential/fundamental matrix for vSLAM and VO. For finding these frames, the first frame is considered as the first keyframe and I examine the next frames one by one until either I cannot solve essential/fundamental matrix or there is no more frame. As I find a frame where essential/fundamental matrix cannot be solved, I consider the previous one as a keyframe and add it to $\varkappa$.

For processing each frame, first the features must be extracted using algorithms such as SURF or SIFT. Then, using RANSAC algorithm, common features are found and selected to be introduced to the Eight-point algorithm for calculating essential matrix. If Eight-point algorithm could yield any result, then I move on to the next frame until there is no result. This process will be iterated through all frames to find a keyframe set. Figure 5.2 shows the steps for finding the effective set of keyframes. Figure 5.3 schematically shows the output of effective keyframes.



$frame_i$ = readFrame()
$frame_j$ = readFrame()
$frame_{pre}$ = framej
$frame_{pre}$ = $frame_j$
$frame_j$ = readFrame()
Extract Features of $frame_i$
Extract Features of $frame_j$
Find common features using RANSAC
Can essential matrix be calculated?
No
Yes
$frame_i$ = $frame_{pre}$
Add $frame_i$ to Keyframe
Yes
Is more frame available?
No
End

Figure 5.2: Flowchart of finding effective keyframes

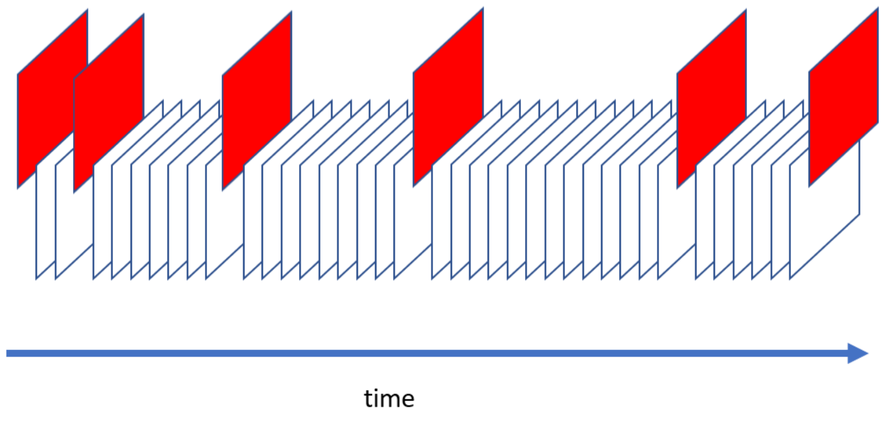


Figure 5.3: Schematic result of effective keyframes

5.4 A supervised learning approach for keyframe detection

Finding the effective keyframes provides us with not only a set of frames and a set of labels indicating if a frame is keyframe or not, it also provides the effective frame rate extraction, shown by different colors in Figure 5.4. Moreover, while capturing frames from video, the inertial data, usually at a higher frequency, including acceleration and rotational velocity are collected by smartphone. Hence, beside the visual data and labels, I also have a set of inertial data. Therefore, for each time interval, I can see a correlation between the camera movement and the rate of keyframes. For instance, in Figure 5.4 the movement of camera in each time window between keyframes can be justified as a mode of change of similarity. This inertial data between each two consecutive frames and the corresponding frames can be evaluated to establish an association which can label frame as a keyframe.

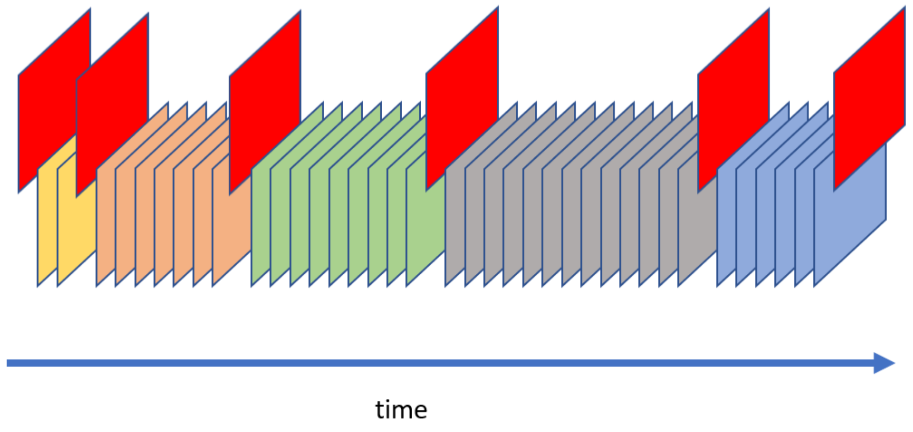


Figure 5.4: Keyframe rate extraction for keyframes

5.4.1 Framerate extraction class

As it was mentioned in the last section, the effective keyframe dataset can define what the ideal frame rate would be for frame extraction to detect keyframes. In order to train a model that can behave in a similar way where for each input feature vector of inertial data, a corresponding framerate must be assigned. There are two possibilities for the framerate: estimating an exact value or selecting a class from a predefine class set. I define a set of classes by ordering all frame rates for effective keyframes and clustering the rates. Each cluster can be labeled with a value for frame rate where I use the minimum value in that cluster. Therefore, for each time window (in a sliding window technique), the class label is assigned to the feature vector. This process is shown in Figure 5.5. It shows that if all the frequencies (or δt's) are mapped to a one-dimensional diagram, they can be clustered, and each cluster is labeled with the minimum time or maximum frequency to guarantee that all frames will have enough common features.

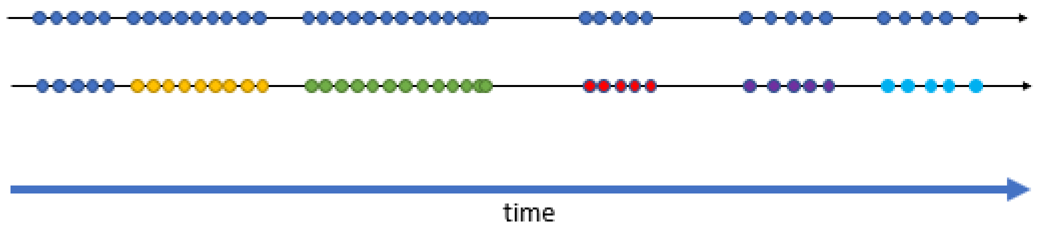


Figure 5.5: Classes of frame extraction

5.4.2 Data preprocessing and feature extraction for input data preparation

The change of consecutive images is due to a wide variety of movements during the mobility. In this research, I consider seven different classes of frame-rate extraction. To detect these classes, I preprocess raw sensor data to select the effective set of features that have the highest gain of information, use a classifier to fuse feature vector data, and map each feature vector to one of these classes. The output of the classification would be one of the seven classes that correspond to the movement modes. Figure 5.6 shows the process of classification using a supervised classification technique. A set of classifiers were examined and are discussed later in this section.

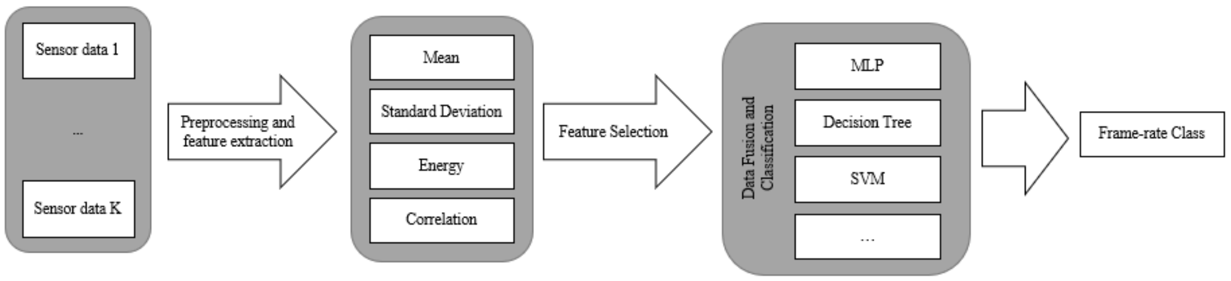


Figure 5.6: Overview of movement recognition

Features are extracted from raw sensor data by using a window size of 128, with 64 samples overlapping between consecutive windows. The efficiency of feature extraction on windows with 50% overlap has been previously discussed (Bao & Intille, 2004). At a sampling frequency of 30 Hz, each window represents data for 4.026 s. This means that each window contains data from several seconds and can sufficiently capture enough data for the process of activity recognition. Mean, standard deviation, energy, and correlation are the features (Bao & Intille, 2004; Ravi et al., 2005; Ming Ren & Karimi, 2012) extracted from the sliding window signals. Moreover, the sensors (accelerometer, gyroscope, and magnetometer) receive data in three dimensions; thus, each observation includes three components, and combining the three sensors results in nine components per observation. These observations are used for feature computations; with four observations, there is a total of 36 possible attributes. Figure 5.7 and Figure 5.8show linear acceleration and linear attitude signals. The extracted mean and standard deviation feature using sliding window are shown in Figure 5.9, and Figure 5.10.

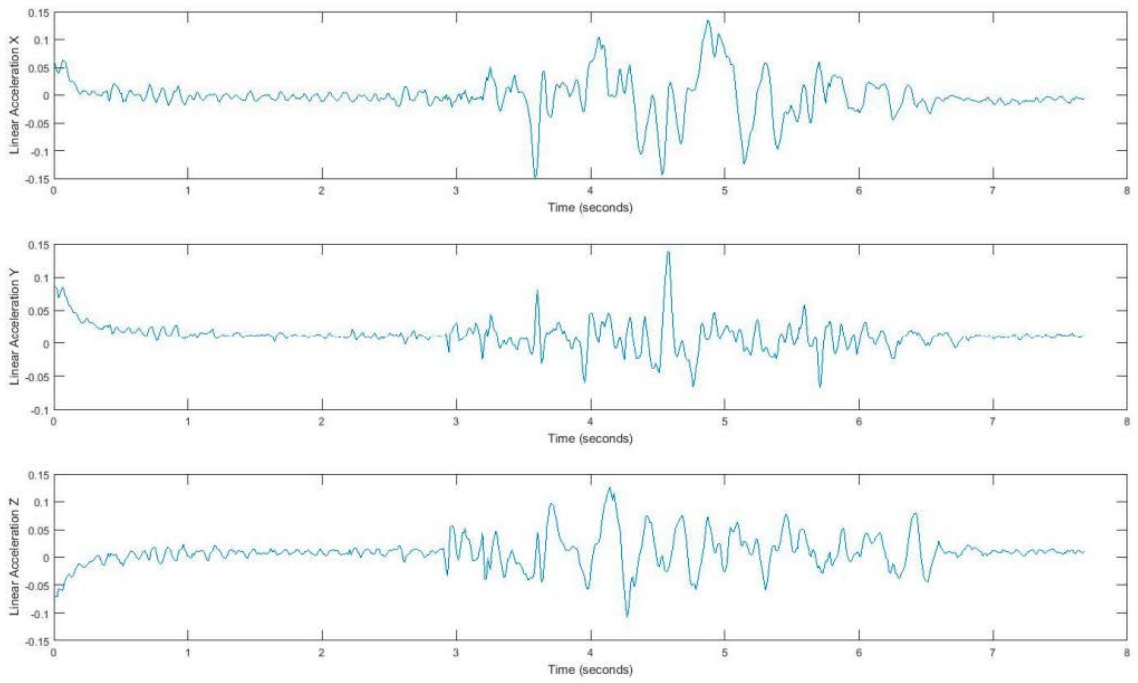


Figure 5.7: Linear acceleration signal

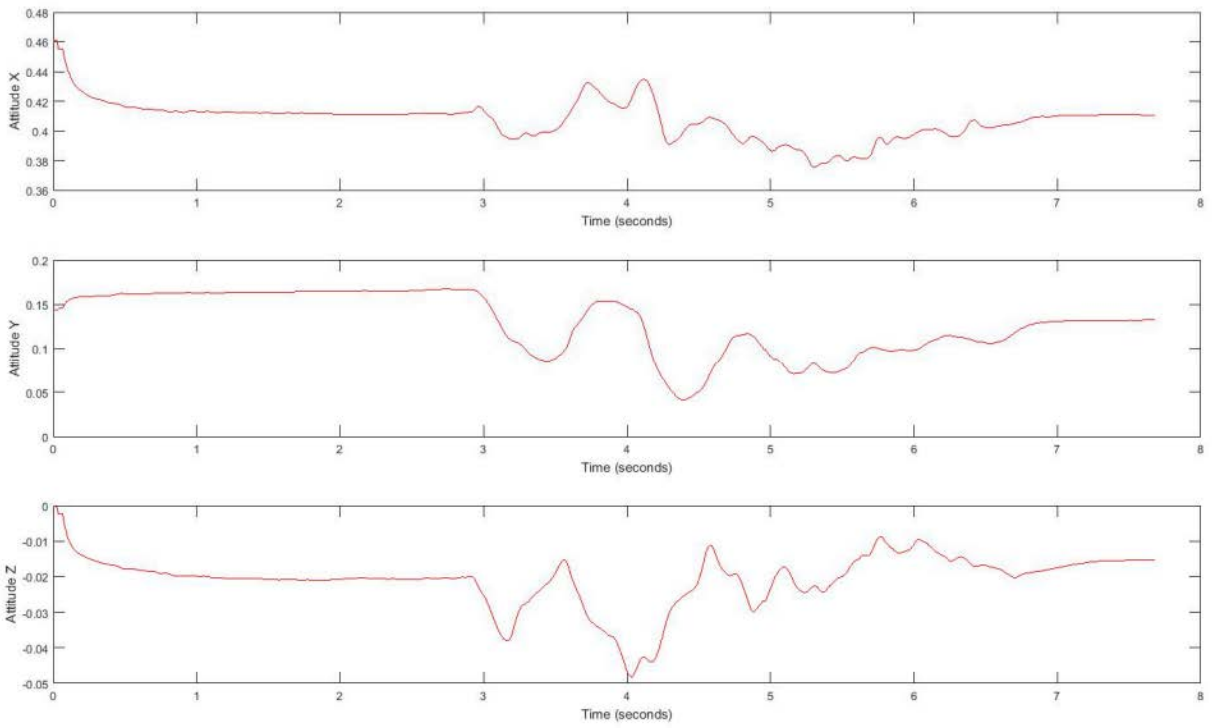


Figure 5.8: Linear attitude signal

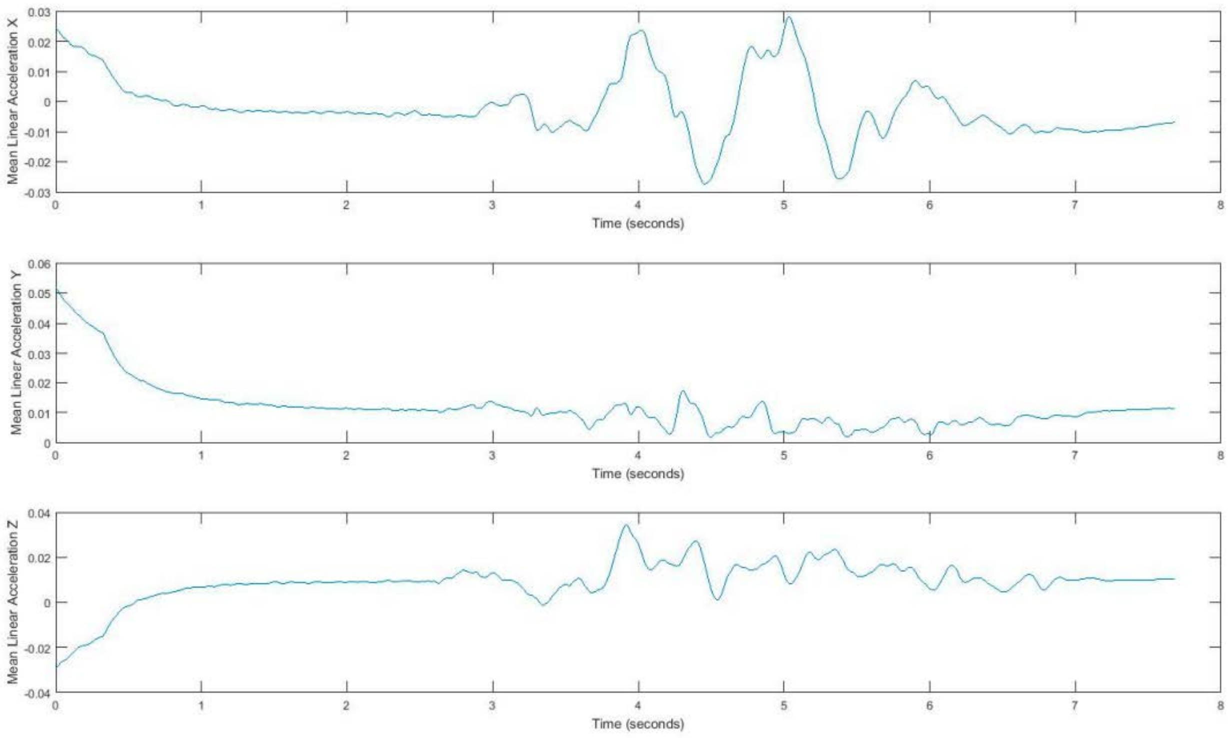


Figure 5.9: Mean of linear acceleration, using a sliding window

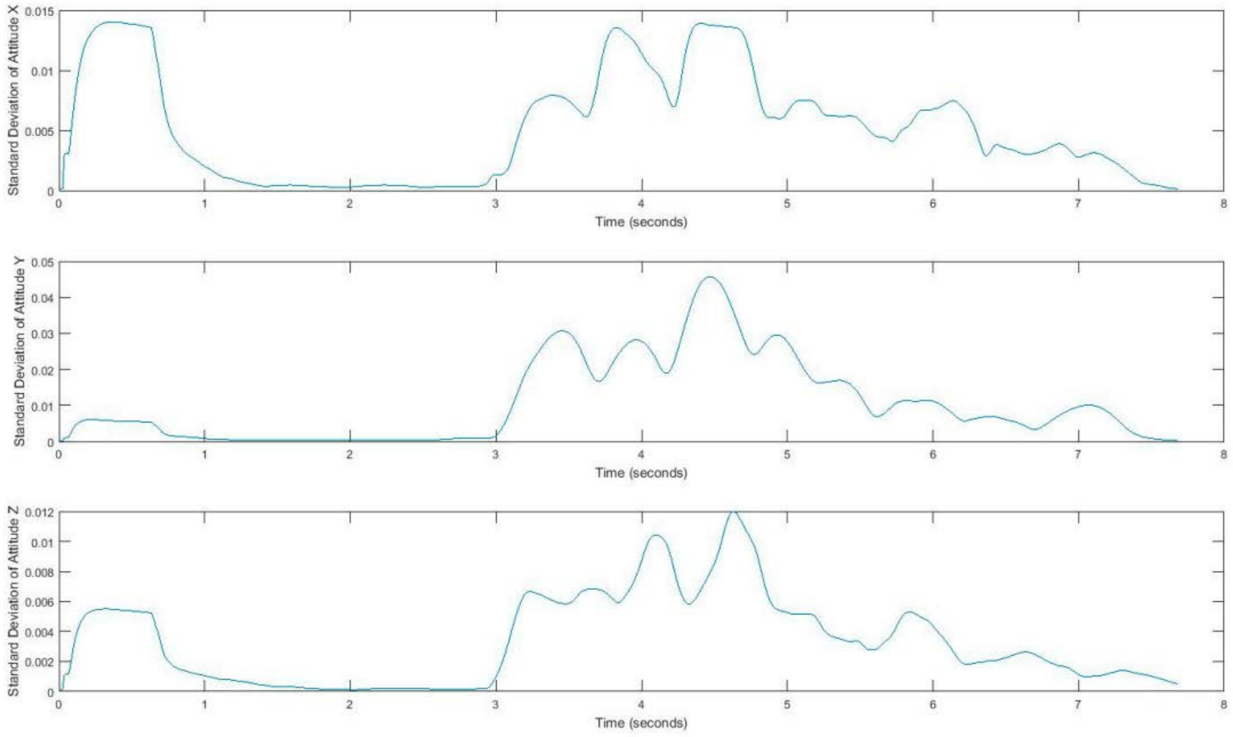


Figure 5.10: Standard deviation of the attitude signal, computed using a sliding window

The usefulness of these features has been previously discussed (Bao & Intille, 2004). The standard deviation is used for capturing the range of possible values, which differs for each activity. Fast Fourier Transform (FFT) converts the signal to frequency, which is used for computing energy parameters. Using the window size of 128 samples enables us to quickly calculate these parameters. In fact, energy feature is the sum of the squared discrete FFT coefficient magnitudes of the signal. The sum is divided by the window length of the window for normalization (Bao & Intille, 2004; Ravi et al., 2005).

If $x_1, x_2, \cdots$ are the FFT components of the window, then $Energy = \frac{\sum_{i=1}^{|w|} |x_i|^2}{|w|}$. Energy and mean values differ for each activity; therefore, they can determine specific activities. In addition, translation in one dimension is clearly recognizable using correlation (Ravi et al., 2005).

5.4.3 Feature selection

After generating feature vector, to reduce over-fitting, improve accuracy, and reduce training time, I perform feature selection. As high dimensionality of the feature vector could mean some of the features might not be relevant to the activity classes, I extract a subset of the computed features to decrease redundancy and increase relevance to the class labels in classification. Since my training dataset is labeled, I use a filter model utilizing a supervised technique where feature selection is separated from classifier learning (Tang, Alelyani, & Liu, 2014). Consequently, the bias of a learning algorithm cannot influence the bias of a feature selection algorithm and the interactions between them are avoided. Generally, feature selection algorithms may use measurement of the various characteristics of the training data such as distance, consistency, dependency, information, and correlation (Kira & Rendell, 1992; Robnik-Siknja & Kononeko, 2003; Tang et al., 2014).

I implemented two feature selection techniques and selected the top-ranked features produced by both techniques. In these techniques, I used feature set $\mathcal{F} = \{f_1, f_1, \dots, f_m\}$, class label set $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, where m is the number of features and K denotes the number of classes, and dataset $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{R}^{m \times n}$, where the label information of the i-th instance xi is denoted by y_i (Tang et al., 2014).

Information Gain (Robnik-Siknja & Kononeko, 2003; Tang et al., 2014) is a computationally efficient technique for feature selection. Since it is simple to interpret the results of Information Gain, it is a well-known and popular feature selection method. This technique measures the dependency between features and class labels. Equation 5.4 shows how to calculate the Information Gain between the *i-th* feature f_i and the class labels $\mathcal{C}$ where $H(f_i)$ is the entropy of f_i and $H(f_i|\mathcal{C})$ is the entropy of f_i after observing $\mathcal{C}$. A higher Information Gain indicates a higher relevancy of a feature

$$IG(f_i, \mathcal{C}) = H(f_i) - H(f_i|\mathcal{C}) \tag{5.4}$$

$$H(f_i) = -\sum_j p(x_j) log_2 \left(p(x_j)\right) \tag{5.5}$$

$$H(f_i|\mathcal{C}) = -\sum_k p(C_k) \sum_j p(x_j|C_k)\, log_2 \left(p(x_j|C_k)\right) \tag{5.6}$$

ReliefF (Robnik-Siknja & Kononeko, 2003; Tang et al., 2014) is a multi-class extension of Relief that selects features to separate an instance from different classes. If I have l random instances sampled from the dataset, the score of the *i-th* feature, S_i, can be calculated by Equation

5.7, where $d(\cdot)$ is the distance function, M_k is a set of nearest point to x_k with the same class label, H_k is a set of nearest points to x_k with class y which is different from class label of x_k, and, $p(y)$ denotes the probability of an instance from class y.

$$S_i = \frac{1}{K}\sum_{k=1}^{l}\left(-\frac{1}{m_k}\sum_{x_j \in M_k} d(\boldsymbol{X}_{ik} - \boldsymbol{X}_{ij}) + \sum_{y \neq y_k}\frac{1}{h_{ky}}\frac{p(y)}{1-p(y)}\sum_{x_j \in H_k} d(\boldsymbol{X}_{ik} - \boldsymbol{X}_{ij})\right) \quad 5.7$$

In this research, representative feature selection techniques include Information Gain and ReliefF.

5.4.4 Classification and best classifier

In order to create a model which, consider both motion and environment, a supervised learning-based approach is proposed for keyframe detection. In this approach the learner is provided with a set of input (feature vector of inertial data)/output (frequency rate of frame extraction class) pairs as follows:

$$(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} \quad 5.8$$

The learned model $f: \mathcal{X} \rightarrow \mathcal{Y}$ should map input examples into their outputs (inertial vector into the frequency rate).

I considered the performance of the following classifiers for the problem of classification, as they have already been used for similar purpose in other research (Bao & Intille, 2004; Ravi et al., 2005; Ming Ren & Karimi, 2012; Sara Saeedi et al., 2014): decision table, decision tree (C4.5), support vector machines (SVM), Naive Bayes, multi-layer perceptron (MLP), RBFNetwork, and Bayes network classifiers, using the selected features for various and different datasets as inputs to the classifiers. I used an indoor dataset which was collected in some different indoor environments, and the locations of data collection are discussed in the results section. I calculated precision, recall, and f-measure of these classifiers to determine the most accurate one. I evaluated Decision Tables, Decision Trees (C4.5), SVM, Naïve Bayes, MLP, RBF Network, and Bayes Network classifiers using the selected features for various and different datasets as inputs to the classifiers. I calculated precision, recall, and f-measure of these classifiers.

5.4.5 Neural network for keyframe detection, architecture and training

Neural networks can model the relationship between an input vector x and an output y. The learning process involves adjusting the parameters in a way that enables the network to predict the output value for new input vectors. I applied a neural network model to find the class of frame rate extraction for keyframe. In order to efficiently design and train a neural network, I must find an appropriate network architecture, determine a suitable training set, and compute the corresponding parameters of the network (such as weights and learning rate) by using an efficient and effective algorithm. In the rest of this section, I explain the overall system architecture and the training process of the parameter

In this work, I used multilayer perceptron (MLP) to model the nonlinear relationships between input vectors, the extracted features, and the output (class value), with nonlinear transfer functions. The basic MLP network is designed by arranging units in a layered structure, where each neuron in a layer takes its input from the output of the previous layer or from an external input. Figure 5.11 shows a schematic diagram of the MLP structure. The transfer functions of the hidden layer in the feedforward network is a sigmoid function. Since I use MLP as a classifier for multiple classes, I should produce reasonable output values using sofrmax function.

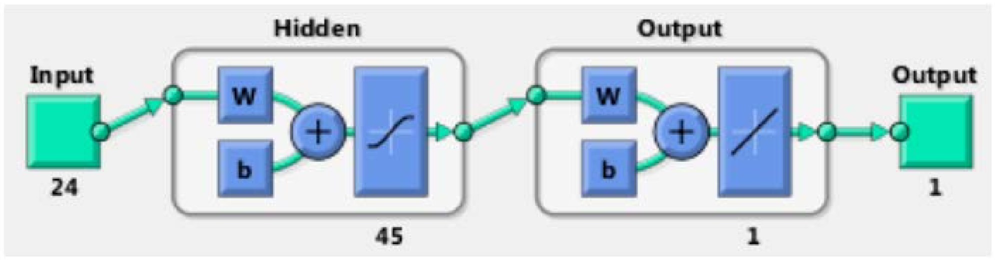


Figure 5.11: The architecture of the feedforward neural network

5.5 Results and comparison

In this section, first I show the results of training and evaluation of the proposed model. Then I compare the uniform keyframe detection, another context-aware keyframe detection, which proposed, and an effective keyframes. Before comparing the models, the context-aware model is briefly discussed.

5.5.1 Training and validation

Numerous training algorithms and learning rules have been proposed for setting the weights and parameters in neural networks; however, it is not possible to determine a global minimum solution. Therefore, training a network is one of the most crucial steps for neural network design. Backpropagation, which is basically a gradient descent optimization technique, is a standard and basic technique for training feedforward neural networks; however, it has some limitations, such as slow convergence, local search nature, overfitting data, and being over-trained, which can cause a loss of the network's ability to correctly estimate the output (Burden & Winkler, 2008). As a result, the validation of the models can be problematic. Moreover, optimization of the network architecture is sometimes time-consuming. There are some modifications to the backpropagation, such as conjugate-gradient and Levenberg-Marquardt algorithms, that are faster than any variant of the backpropagation algorithm (Masters, 1995; S. Mohanty, Jha, Kumar, & Sudheer, 2010). The Levenberg-Marquardt algorithm is for minimizing a sum of squared error (Gavin, 2013; Roweis, 1996) and to overcome some of the limitations in the standard backpropagation algorithm, such as an overfitting problem.

Avoiding the overfitting problem in network architectures can be a serious challenge, because I try to achieve an accurate estimation of the modeled function by a neural network with a minimum number of input variables and parameters. Having too many neurons in the hidden layer can cause overfitting, since the noise of the data is modeled along with the trends. Furthermore, an insufficient number of neurons in the hidden layer can cause problems with the learning data. For the purpose of finding the optimum number of neurons in the hidden layer, I conducted a model selection experiment with different number of neurons ranging from 5 to 50 and the cross-validation error for each setting was calculated. As a result, a hidden layer with 45 neurons is a good fit for the dataset to overcome some of the limitations in the standard backpropagation algorithm, such as an overfitting problem. As neural networks can model the relationship between an input vector x and an output y, the learning process involves adjusting the parameters in a way that enables the network to predict the output value for new input vectors. The trained model is validated by six sequences that has never been used.

5.5.2 Comparison of different detecting approaches

In this section, three approaches to detect the keyframes are compared. In the first approach I extract the effective set of keyframes. The sequences are evaluated in a brute force fashion to determine all frames that belong to this subset. The second approach is the uniform keyframe detection which is straightforwardly prunes the redundant frames. The third approach is the proposed model which is based on the movement change of the user and its impact on the visual data. As shown in Figure 5.13, I tested the models for six different sequences not incorporated in the training process.

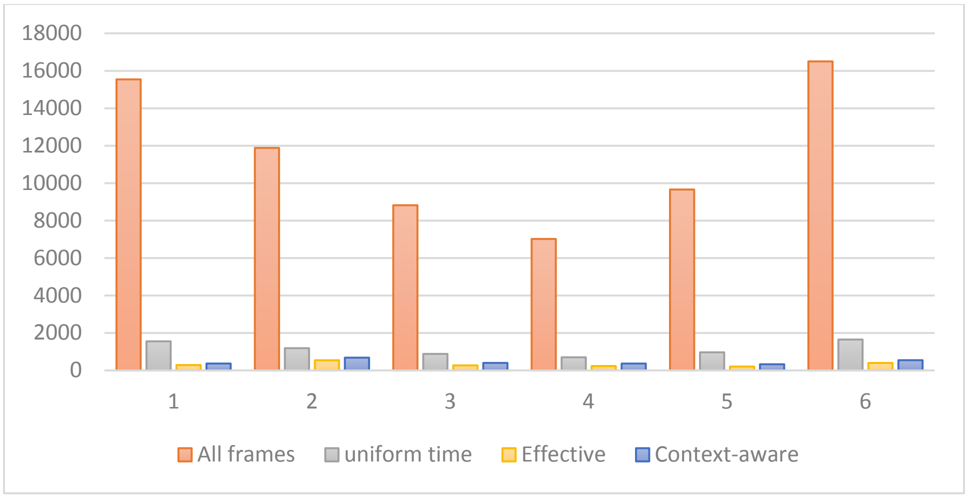


Figure 5.12: Validation of the model for six sequences that are not included in the training procedure, with total number of frames

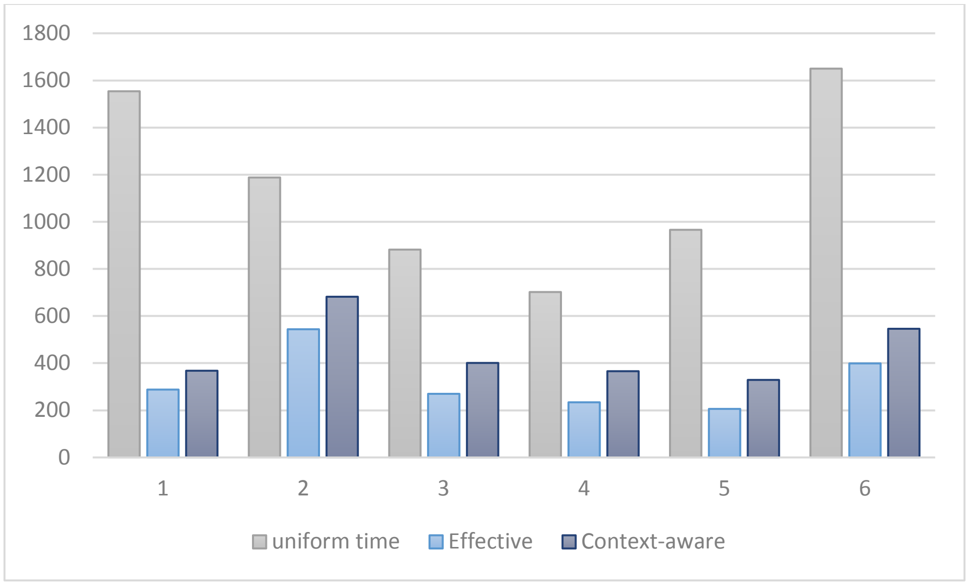


Figure 5.13: Validation of the model for six sequences that are not included in the training procedure

Figure 5.12 and Figure 5.13 show how keyframe detection reduces the number of frames for visual odometry. In fact, about 90% of frames are pruned even in uniform keyframe detection. Moreover, Figure 5.13 shows the effectiveness of the proposed model as the outcome is close to the effective set compared to the uniform keyframe detection.

5.6 **Summary**

Keyframe detection is an essential step in deciding the poses of the moving object to be included in the map for an appropriate coverage of the environment. In this chapter I introduced the effective set of keyframes where in an ideal scenario they should be selected for pose computation. While it is desirable to detect keyframes, it is computationally expensive to process the visual data to obtain them. Hence, since the change in appearance is correlated with the change of inertial data, I defined a machine learning approach for determining whether a frame is a keyframe or not. In this approach, the achieved result is closer to the effective set and more efficient than other approaches such as uniform sampling in time or space. Although the proposed method needs inertial data to perceive the context and infer the class of framerate extraction, unlike other methods, it does not need any threshold.

6.0 Conclusions and future research

6.1 Summary

This dissertation proposes a framework for visual-inertial odometry which fuses the data from a visual coordinate estimator with inertial data utilizes using Extended Kalman Filter (EKF). This was done by integrating a deep learning monocular visual odometry algorithm using a deep RCNN with a Gaussian filter module for fusing inertial data from IMU with coordinates from visual odometry. Moreover, I designed, developed, and implemented an algorithm for obstacle avoidance. Finally, I proposed a context-aware keyframe detection that tries to extract semi-effective keyframes without any visual processing. A summary of the framework and algorithms developed in this dissertation is as follows.

In the first part of the dissertation, to address the problem of scale in deep monocular visual odometry, a framework was proposed which uses deep learning in tandem with Extended Kalman Filter for positioning and localization in GNSS-denied environments. The underlying technique is a deep recurrent convolutional neural network (RCNN) for computing six-degree-of-freedom in an end-to-end fashion and an extended Kalman filter for fine-tuning the scale parameter based on inertial observations. I compared the results of the featureless technique with the results of conventional feature-based VIO techniques

In the second part a context-aware smartphone-based visual obstacle detection approach to aid visually impaired people in navigating indoor environments was proposed. The approach is based on processing two consecutive frames (images), computing optical flow, and tracking certain points to detect obstacles. The frame rate of the video stream is determined using a context-aware data fusion technique for the sensors on smartphones. Through an efficient and novel algorithm, a point dataset on each consecutive frame is designed and evaluated to check whether the points belong to an obstacle. In addition to determining the points based on the texture in each frame, my algorithm also considers the heading of user movement to find critical areas on the image plane. I validated the algorithm through experiments by comparing it against two comparable algorithms. The experiments were conducted in different indoor settings and the results based on precision, recall, accuracy, and f-measure were compared and analyzed. The results show that, in comparison to the other two widely used algorithms for this process, my algorithm is more precise. I also considered time-to-contact parameter for clustering the points and presented the performance improvement by using this parameter.

In the last part I proposed a new algorithm to detect keyframes without processing the frames and just by using inertial data and machine learning. The proposed approach to solve this issue is by using a supervised learning classification algorithm for training a model based on “effective” keyframes, which are the most appropriate frames for the purpose, in sequences within various environments. This model gets inertial data as input and the output is the class of frame-rate extraction.

In this dissertation, experiments were conducted to evaluate the developed algorithms by navigating in different indoor locations of the main campus of the University of Pittsburgh, a resident building, and a grocery store for obstacle detection and using standard publicly available datasets for collecting visual and inertial data while moving through various indoor environments.

6.2 Contributions

Positioning and localization are essential components of any navigation system and play a major role in location-based services. The research conducted considered three main navigational tasks with concentration on indoor navigation. I summarize the novelty of this thesis by examining its three main contributions: (a) a novel framework for visual inertial odometry that uses a deep recurrent convolutional neural network and an extended Kalman filter to solve the tracking problem in a more efficient way; (b) a new algorithm for obstacle detection in unfamiliar environments; and (c) a context-aware methodology for frame detection for navigational tasks and an approach for inferring behavioral context and associating it with moving platforms for indoor movement.

6.3 Limitations

This research provided new frameworks, methods, and algorithms for addressing some of the important challenges in indoor navigation as well as for other types of navigation and location-based services. However, there are some limitations to the research which are discussed below.

In this dissertation I tackled the problem of indoor positioning by integrating learning-based visual odometry with a filtering technique. While there are many datasets available for outdoor navigation, one main limitation is the very limited number of datasets for indoor navigation which makes it difficult to train the model. Although I used transfer learning and trained the model using different subsets of the sequences as a solution to this limitation, having more data can improve the results. Another issue with the current dataset is synchronization. I tried to fix the issue with a linear interpolation, however, it has some negative impacts on the quality of the results.

In this dissertation, I introduced an algorithm to design and extract points to improve the detection of obstacles using optical flow and point track algorithms. This approach was able to detect stationary and moving obstacles close to the camera. My algorithm for extracting the points was based on the image's texture and on the movement of the user. I conducted an experiment to show that this kind of point extraction can improve both the efficiency and precision of the results. I also demonstrated ways in which TTC can improve the overall clustering. Although this is a slight improvement, TTC could be used for some further analyses, such as prioritizing the detected obstacles. Moreover, I used multiple frame rates, according to the user's activity.

Despite these improvements, there are some limitations which future work should consider. The point detection algorithm extracts points on lamps and reflective floors and surfaces, some are likely to be detected as objects. I found some of such points in the output as obstacles, which influenced the precision and accuracy of the results. Removing these types of features could substantially improve the algorithm. Additionally, finding a method to remove the floors and carpets, to avoid errors in obstacle detection, is another future work. Eliminating these errors can help produce less noisy results.

6.4 **Future research**

The proposed framework for visual-inertial odometry integrates convolutional and recurrent neural networks with Extended Kalman Filter to enhance indoor navigation. While the main goal is for a GNSS-denied environment, it is still important to expand this framework to fuse

GPS data with other sensory data for obtaining more reliable and accurate coordinates. Moreover, incorporating Radio Frequency (RF)-based techniques such as Wi-Fi-based positioning, cellular-based positioning, or Bluetooth-based positioning can expand the applicability of the framework. In the second part, I introduced an algorithm to design and extract points to improve the detection of obstacles using optical flow and point track algorithms. This approach was able to detect moving objects and stationary obstacles close to the camera. My algorithm for extracting the points was based on the image texture and the movement of the user. I conducted an experiment to show that this kind of point extraction can improve both the efficiency and precision of the results. I also demonstrated ways in which TTC can improve the overall clustering. Although this is a slight improvement, TTC could be used for some further analyses, such as prioritizing the detected obstacles. Moreover, I used multiple frame rates, according to the user's activity.

Despite these improvements, there are some limitations which future work should consider. The point detection algorithm extracts points on lamps and reflective floors and surfaces, some are likely to be detected as objects. I found some of such points in the output as obstacles, which influenced the precision and accuracy of the results. Removing these types of features could substantially improve the algorithm. Additionally, finding a method to remove the floors and carpets, to avoid errors in obstacle detection, is another option for future work. Eliminating these areas can help produce less noisy results. In addition, with data on the floor, topologic relationships and affordance theory to include information on the floor and related objects can be explored.

In the third part, I used a feature-based approach to analyze the inertial signals and classify it for keyframe detection. There are two major challenges with these introduced techniques for analyzing sensor data. First, sensor measurements on devices like smartphones are too noisy which makes it difficult to model it. Second, although manually feature selection is effective, it is challenging to find the most informative features to use for various sensor noise patterns and heterogeneous user behaviors. Another approach can be an end-to-end learning-based framework that directly addresses the challenges of selecting features and managing the introduced noises by sensors which can form a unified learning-based framework for keyframe detection.

Bibliography

Agrawal, P., Carreira, J., & Malik, J. (2015). Learning to see by moving. In *Computer Vision (ICCV), 2015 IEEE International Conference on* (pp. 37–45).

Angeli, A., Doncieux, S., Meyer, J.-A., & Filliat, D. (2008). Incremental vision-based topological slam. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1031–1036).

Arnold, Z., Larose, D., & Agu, E. (2015). Smartphone inference of alcohol consumption levels from gait. In *Healthcare Informatics (ICHI), 2015 International Conference on* (pp. 417–426).

Bao, L., & Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. In *Pervasive computing* (pp. 1–17). Springer.

Batavia, P. H., Pomerleau, D. a., & Thorpe, C. E. (1999). Overtaking vehicle detection using implicit optical flow. *Computer Standards & Interfaces*, *20*(6–7), 466. https://doi.org/10.1016/S0920-5489(99)91018-8

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404–417).

Beauchemin, S. S., & Barron, J. L. (1995). The computation of optical flow. *ACM Computing Surveys*, *27*(3), 433–466. https://doi.org/10.1145/212094.212141

Benjamin, J. M., Ali, N. A., & Schepis, A. F. (1973). A laser cane for the blind. In *Proceedings of the San Diego Biomedical Symposium* (Vol. 12).

Bonarini, A., Matteucci, M., & Restelli, M. (2004). A kinematic-independent dead-reckoning sensor for indoor mobile robotics. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Vol. 4, pp. 3750–3755).

Borenstein, J. (2001). The GuideCane-applying mobile robot technologies to assist the visually impaired. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions On*, *31*(2), 131–136.

Boroujeni, N. S. (2012). Fast obstacle detection using targeted optical flow. *… (ICIP), 2012 19th IEEE …*, *2*(3), 65–68. https://doi.org/10.1109/ICIP.2012.6466796

Bousbia-Salah, M., Bettayeb, M., & Larbi, A. (2011). A Navigation Aid for Blind People. *Journal of Intelligent & Robotic Systems*, *64*(3–4), 387–400. https://doi.org/10.1007/s10846-011-9555-7

Browning, N. A., Grossberg, S., & Mingolla, E. (2009). A neural model of how the brain computes heading from optic flow in realistic scenes. *Cognitive Psychology*, *59*(4), 320–356. https://doi.org/10.1016/j.cogpsych.2009.07.002

Burden, F., & Winkler, D. (2008). Bayesian Regularization of Neural Networks (pp. 23–42). https://doi.org/10.1007/978-1-60327-101-1_3

Callmer, J., Granström, K., Nieto, J., & Ramos, F. (2008). Tree of words for visual loop closure detection in urban SLAM. In *Proceedings of the 2008 Australasian conference on robotics and automation* (p. 8).

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision* (pp. 778–792).

Camus, T. (1994). *Real-time optical flow*. Society of Manufacturing Engineers. Retrieved from ftp://128.148.32.111/pub/techreports/94/cs94-36.pdf

Chatzigiorgaki, M., & Skodras, A. N. (2009). Real-time Keyframe extraction towards video content identification. *DSP 2009: 16th International Conference on Digital Signal Processing, Proceedings*, 0–5. https://doi.org/10.1109/ICDSP.2009.5201141

Cheng, C. C., Li, C. Te, & Chen, L. G. (2010). A novel 2Dd-to-3D conversion system using edge information. *IEEE Transactions on Consumer Electronics*, *56*(3), 1739–1745. https://doi.org/10.1109/TCE.2010.5606320

Cimiano, P., Hotho, A., & Staab, S. (2004). Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004*.

Cortés, S., Solin, A., Rahtu, E., & Kannala, J. (2018). ADVIO: An authentic dataset for visual-inertial odometry. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11214 LNCS*, 425–440. https://doi.org/10.1007/978-3-030-01249-6_26

Costa, P., Fernandes, H., Martins, P., Barroso, J., & Hadjileontiadis, L. J. (2012). Obstacle Detection using Stereo Imaging to Assist the Navigation of Visually Impaired People. *Procedia Computer Science*, *14*(Dsai), 83–93. https://doi.org/10.1016/j.procs.2012.10.010

Cummins, M., & Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, *27*(6), 647–665.

Das, A., & Waslander, S. L. (2015). Entropy based keyframe selection for multi-camera visual slam. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3676–3681).

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1052–1067. https://doi.org/10.1109/TPAMI.2007.1049

Derpanis, K. G. (2010). Overview of the RANSAC Algorithm. *Image Rochester NY*, *4*(1), 2–3. https://doi.org/10.1002/cne.901000107

DeTone, D., Malisiewicz, T., & Rabinovich, A. (2016). Deep image homography estimation. *ArXiv Preprint ArXiv:1606.03798*.

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, *17*(3), 229–241.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2625–2634).

Dong, Z., Zhang, G., Jia, J., & Bao, H. (2009). Keyframe-based real-time camera tracking. In *2009 IEEE 12th international conference on computer vision* (pp. 1538–1545).

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., … Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2758–2766).

Du, W., & Shen, M. (2016). Siamese Convolutional Neural Networks for Authorship Verification. Retrieved from http://cs231n.stanford.edu/reports/2017/pdfs/801.pdf

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *Nips*, 1–9. https://doi.org/10.1007/978-3-540-28650-9_5

El-Gaaly, T., Tomaszewski, C., & Valada, A. (2013). Visual Obstacle Avoidance for Autonomous Watercraft using Smartphones.

Engel, J., Koltun, V., & Cremers, D. (2016). Direct Sparse Odometry. *Axiv*. Retrieved from http://arxiv.org/abs/1607.02565

Fallah, N., Apostolopoulos, I., Bekris, K., & Folmer, E. (2013). Indoor human navigation systems: A survey. *Interacting with Computers*, *25*(1), 21–33. https://doi.org/10.1093/iwc/iws010

Fischler, M. a., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Ganz, A., Schafer, J. M., Tao, Y., Wilson, C., & Robertson, M. (2014). PE RCEPT - II : Smartphone based Indoor Navigation System for the B lind. *36th Annual International Conference of the IEEE Engineering in Medicince and Biology Society*, 3662–3665. https://doi.org/http://dx.doi.org/10.1109/EMBC.2014.6944417

Gavin, H. P. (2013). The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering, Duke University*, 1–17.

https://doi.org/10.1080/10426914.2014.941480

Geiger, A., Ziegler, J., & Stiller, C. (2011). StereoScan: Dense 3D Reconstruction in Real-time. In *Intelligent Vehicles Symposium (IV)*.

Gharani, P., & Karimi, H. A. (2017). Context-aware obstacle detection for navigation by visually impaired. *Image and Vision Computing*, *64*. https://doi.org/10.1016/j.imavis.2017.06.002

Gharani, P., Suffoletto, B., Chung, T., & Karimi, H. A. (2017). An artificial neural network for movement pattern analysis to estimate blood alcohol content level. *Sensors (Switzerland)*, *17*(12). https://doi.org/10.3390/s17122897

Haiying Liu, Chellappa, R., & Rosenfeld, A. (2003). Accurate dense optical flow estimation using adaptive structure tensors and a parametric model. *IEEE Transactions on Image Processing*, *12*(10), 1170–1180. https://doi.org/10.1109/TIP.2003.815296

Harris, C., & Stephens, M. (1988a). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, pp. 10–5244).

Harris, C., & Stephens, M. (1988b). A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, 23.1-23.6. https://doi.org/10.5244/C.2.23

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Hesch, J. A., Kottas, D. G., Bowman, S. L., & Roumeliotis, S. I. (2013). Camera-IMU-based Localization: Observability Analysis and Consistency Improvement. *Ijrr*, *33*(1), 182–201. https://doi.org/10.1177/0278364913509675

Ho, K. L., & Newman, P. (2006). Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics and Autonomous Systems*, *54*(9), 740–749.

Horn, B. K. P., & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*. https://doi.org/10.1016/0004-3702(81)90024-2

Jin, H., Favaro, P., & Soatto, S. (2003). A semi-direct approach to structure from motion. *The Visual Computer*, *19*(6), 377–394.

Johnson, A. E., Goldberg, S. B., Cheng, Y., & Matthies, L. H. (2008). Robust and efficient stereo feature tracking for visual odometry. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 39–46).

José, J., Buf, J. M. H. du, & Rodrigues, J. M. F. (2012). VISUAL NAVIGATION FOR THE BLIND Path and Obstacle Detection. In *International Conference on Pattern Recognition Applications and Methods* (pp. 515–519).

Kao, H.-L. (Cindy), Ho, B.-J., Lin, A. C., & Chu, H.-H. (2012). Phone-based gait analysis to detect alcohol usage. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing -*

UbiComp '12, 661. https://doi.org/10.1145/2370216.2370354

Karimi, H. A. (2015). *Indoor wayfinding and navigation*. CRC Press.

Kasyanov, A., Engelmann, F., Stückler, J., & Leibe, B. (2017). Keyframe-Based Visual-Inertial Online SLAM with Relocalization, (2). Retrieved from http://arxiv.org/abs/1702.02175

Katz, B. F. G., Kammoun, S., Parseihian, G., Gutierrez, O., Brilhault, A., Auvray, M., … Jouffrais, C. (2012). NAVIG: augmented reality guidance system for the visually impaired. *Virtual Reality, 16*(4), 253–269. https://doi.org/10.1007/s10055-012-0213-6

Ke, Y. K. Y., & Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, *2*, 2–9. https://doi.org/10.1109/CVPR.2004.1315206

Kira, K., & Rendell, L. A. (1992). *A practical approach to feature selection. Proceedings of the ninth international workshop on Machine learning*. https://doi.org/10.1016/S0031-3203(01)00046-2

Kitt, B., Chambers, A., Lategahn, H., Singh, S., & Systems, C. (2011). Monocular Visual Odometry using a Planar Road Model to Solve Scale Ambiguity, 1–6.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* (pp. 225–234).

Kneip, L., Chli, M., & Siegwart, R. (2011). Robust Real-Time Visual Odometry with a Single Camera and an IMU. *Procedings of the British Machine Vision Conference 2011*, 16.1-16.11. https://doi.org/10.5244/C.25.16

Konda, K. R., & Memisevic, R. (2015). Learning Visual Odometry with a Convolutional Network. In *VISAPP (1)* (pp. 486–490).

Konstantinidis, A., Chatzimilioudis, G., Zeinalipour-Yazti, D., Mpeis, P., Pelekis, N., & Theodoridis, Y. (2015). Privacy-preserving indoor localization on smartphones. *IEEE Transactions on Knowledge and Data Engineering, 27*(11), 3042–3055.

LeCun, Y., & Bengio, Y. (1995). *Deep Learning* (Vol. 3361). https://doi.org/10.1109/IJCNN.2004.1381049

Lee, J.-S., Doh, N. L., Chung, W. K., You, B.-J., & Youm, Y. Il. (2004). Door detection algorithm of mobile robot in hallway using PC-camera. In *Proc. of International Conference on Automation and Robotics in Construction*.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, *34*(3), 314–334. https://doi.org/Doi 10.1177/0278364914554813

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Low, T., & Wyeth, G. (1998). Obstacle Detection using Optical Flow. *Australasian Conference on Robotics and Automation*, 1–10. Retrieved from http://www.araa.asn.au/acra/acra2005/papers/low.pdf

Lowe, D. G. (1999). Object Recognition fromLocal Scale-Invariant Features. *IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.1999.790410

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. *IJCAI'81 Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. https://doi.org/Doi 10.1145/358669.358692

Masters, T. (1995). *Advanced algorithms for neural networks: a C++ sourcebook*. John Wiley & Sons, Inc.

Matthies, L. (1989). *Dynamic stereo vision*. Carnegie Mellon University.

Melekhov, I., Ylioinas, J., Kannala, J., & Rahtu, E. (2017). Relative camera pose estimation using convolutional neural networks. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (pp. 675–687).

Mentzelopoulos, M., & Psarrou, A. (2004). Key-Frame Extraction Algorithm using Entropy Difference, 39–45.

Mohanty, S., Jha, M. K., Kumar, A., & Sudheer, K. P. (2010). Artificial neural network modeling for groundwater level forecasting in a river island of eastern India. *Water Resources Management*, *24*(9), 1845–1865. https://doi.org/10.1007/s11269-009-9527-x

Mohanty, V., Agrawal, S., Datta, S., Ghosh, A., Sharma, V. D., & Chakravarty, D. (2016a). DeepVO: a deep learning approach for monocular visual odometry. *ArXiv Preprint ArXiv:1611.06069*.

Mohanty, V., Agrawal, S., Datta, S., Ghosh, A., Sharma, V. D., & Chakravarty, D. (2016b). DeepVO: A Deep Learning approach for Monocular Visual Odometry. Retrieved from http://arxiv.org/abs/1611.06069

Moreno, M., Shahrabadi, S., José, J., du Buf, J. M. H., & Rodrigues, J. M. F. (2012). Realtime Local Navigation for the Blind: Detection of Lateral Doors and Sound Interface. *Procedia Computer Science*, *14*(Dsai), 74–82. https://doi.org/10.1016/j.procs.2012.10.009

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, *31*(5), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

Murray, D., & Little, J. J. (2000). Using real-time stereo vision for mobile robot navigation.

Autonomous Robots, *8*, 161–171. https://doi.org/10.1023/A:1008987612352

Nakju, J. L., Doh, L., Kyun, W., You, C. B., Il, Y., & Youm*, J.-S. L. N. L. D. W. K. C. B.-J. Y. Y. Il. (2004). Door Detection Algorithm of Mobile Robot in Hallway Using Pc-Camera. *Isarc 2004*, *1*(June).

Nassi, B., Rokach, L., & Elovici, Y. (2016). Virtual Breathalyzer. Retrieved from http://arxiv.org/abs/1612.05083

Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 2320–2327).

Newman, P., & Ho, K. (2005). SLAM-loop closing with visually salient features. In *proceedings of the 2005 IEEE International Conference on Robotics and Automation* (pp. 635–642).

Nicolai, A., Skeele, R., Eriksen, C., & Hollinger, G. A. (2016). Deep learning for laser based odometry estimation. In *RSS workshop Limits and Potentials of Deep Learning in Robotics*.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 756–770.

Nistér, D., Naroditsky, O., & Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, *23*(1), 3–20. https://doi.org/10.1002/rob.20103

O'Donovan, P. (2005). Optical Flow: Techniques and Applications. *International Journal of Computer Vision*, 1–26. Retrieved from http://www.dgp.toronto.edu/~donovan/stabilization/opticalflow.pdf

Ohya, A., Kosaka, A., & Kak, A. (1998). Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation*, *14*(6), 969–978. https://doi.org/10.1109/70.736780

Panagiotakis, C., Doulamis, A., & Tziritas, G. (2009). Equivalent key frames selection based on iso-content principles. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*(3), 447–451. https://doi.org/10.1109/TCSVT.2009.2013517

Park, E., Lee, S. I., Nam, H. S., Garst, J. H., Huang, A., Campion, A., … Sarrafzadeh, M. (2016). Unobtrusive and Continuous Monitoring of Alcohol-impaired Gait Using Smart Shoes, 1–9.

Peng, E., Peursum, P., Li, L., & Venkatesh, S. (2010). A smartphone-based obstacle sensor for the visually impaired. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6406 LNCS*, 590–604. https://doi.org/10.1007/978-3-642-16355-5-45

Pradeep, V., Medioni, G., & Weiland, J. (2010). Robot vision for the visually impaired. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 15–22. https://doi.org/10.1109/CVPRW.2010.5543579

Pretto, A., Menegatti, E., & Pagello, E. (2011). Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 3289–3296).

Qian, S., Tan, J. K., Kim, H., Ishikawa, S., & Morie, T. (2013). Obstacles Extraction Using a Moving Camera. In *Computer Vision - ACCV 2012 Workshops* (pp. 441–453). https://doi.org/10.1007/978-3-642-37484-5_36

Ravi, N., Dandekar, N., Mysore, P., & Littman, M. L. (2005). Activity recognition from accelerometer data. In *AAAI* (Vol. 5, pp. 1541–1546).

Ren, M, & Karimi, H. A. (2013). Adaptive Road Candidates Search Algorithm for Map Matching by Clustering Road Segments. *Journal of Navigation*, *66*(3), 435–447. https://doi.org/10.1017/s0373463313000076

Ren, Ming. (2012). Advanced map matching technologies and techniques for pedestrian/wheelchair navigation. Retrieved from http://d-scholarship.pitt.edu/11787/4/Ren,_Ming_Dissertation.pdf

Ren, Ming, & Karimi, H. A. (2012). Movement Pattern Recognition Assisted Map Matching for Pedestrian/Wheelchair Navigation. *Journal of Navigation*, *65*(04), 617–633. https://doi.org/10.1017/S0373463312000252

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *794*, 185–192. https://doi.org/10.1016/j.nima.2015.05.028

Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. (2015). Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1164–1172).

Robnik-Siknja, M., & Kononeko, I. (2003). Theoretical and empirical analysis of RelifF and RReliefF. *Mach Learn*, *53*, 23–69.

Rodríguez, A., Yebes, J. J., Alcantarilla, P., Bergasa, L., Almazán, J., & Cela, A. (2012). Assisting the Visually Impaired: Obstacle Detection and Warning System by Acoustic Feedback. *Sensors*, *12*(12), 17476–17496. https://doi.org/10.3390/s121217476

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430–443).

Roweis, S. (1996). Levenberg-Marquardt Optimization. *Notes, University Of Toronto*.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on* (pp. 2564–2571).

Saeedi, S. (2013). *Context-Aware Personal Navigation Services Using Multi-level Sensor Fusion Algorithms*. *Thesis*. Retrieved from http://theses.ucalgary.ca/bitstream/11023/1098/2/ucalgary_2013_saeedi_sara.pdf

Saeedi, Sara, Moussa, A., & El-Sheimy, N. (2014). Context-Aware Personal Navigation Using Embedded Sensor Fusion in Smartphones. *Sensors*, *14*(4), 5742–5767. https://doi.org/10.3390/s140405742

Sattler, T., Leibe, B., & Kobbelt, L. (2017). Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(9), 1744–1756.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual Odometry [Tutorial]. *IEEE Robotics & Automation Magazine*, *18*(4), 80–92. https://doi.org/10.1109/MRA.2011.943233

Shi, J., & others. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (pp. 593–600).

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., & Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in RGB-D images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (pp. 2930–2937).

Silveira, G., Malis, E., & Rives, P. (2008). An efficient direct approach to visual SLAM. *IEEE Transactions on Robotics*, *24*(5), 969–979.

Song, K., & Huang, J. (2001). Fast optical flow estimation and its application to real-time obstacle avoidance. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, *3*, 2891–2896. https://doi.org/10.1109/ROBOT.2001.933060

Souhila, K., & Karim, a. (2007). Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, *4*(1), 13–16. https://doi.org/10.5772/5715

Stark, L. (1991). Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, *185*(FEBRUARY), 177–185. https://doi.org/10.1007/BF00201978

Suffoletto, B., Gharani, P., Chung, T., & Karimi, H. (2018). Using phone sensors and an artificial neural network to detect gait changes during drinking episodes in the natural environment. *Gait and Posture*, *60*. https://doi.org/10.1016/j.gaitpost.2017.11.019

Szeliski, R. (2010). *Computer Vision : Algorithms and Applications*. Springer Science & Business Media. https://doi.org/10.1007/978-1-84882-935-0

Tang, J., Alelyani, S., & Liu, H. (2014). Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, 37–64. https://doi.org/10.1.1.409.5195

Tapu, R., Mocanu, B., Bursuc, A., & Zaharia, T. (2013). A Smartphone-Based Obstacle Detection and Classification System for Assisting Visually Impaired People. In *2013 IEEE International Conference on Computer Vision Workshops* (pp. 444–451). IEEE. https://doi.org/10.1109/ICCVW.2013.65

Tekdas, O., & Isler, V. (2010). Sensor placement for triangulation-based localization. *IEEE Transactions on Automation Science and Engineering*, *7*(3), 681–685.

Tistarelli, M., & Sandini, G. (1993). On the Advantages of Polar and Log-Polar Mapping for Direct Estimation of Time-To-Impact from Optical Flow. *IEEE Trans. Pattern Anal. Mach. Intell.*, *15*(4), 401–410. https://doi.org/0162-8828/93$03.00

Tomasi, C., & Kanade, T. (1991). Detection and tracking of point features.

Trawny, N., & Roumeliotis, S. I. (2005). Indirect Kalman filter for 3D Attitude Estimation. … , *Dept. of Comp. Sci. & Eng.*, *…*, (2005–002), 1–25. https://doi.org/10.2514/6.2005-6052

Wang, S., Clark, R., Wen, H., & Trigoni, N. (2017). Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (pp. 2043–2050).

Wang, S., Clark, R., Wen, H., & Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *International Journal of Robotics Research*, *37*(4–5), 513–542. https://doi.org/10.1177/0278364917734298

Weiss, S. (2012). *Vision Based Navigation for Micro Helicopters (PhD Thesis - Weiss 2012)*. https://doi.org/10.3929/ethz-a-007344020

Weiss, S., Achtelik, M. W., Lynen, S., Achtelik, M. C., Kneip, L., Chli, M., & Siegwart, R. (2013). Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium. *Journal of Field Robotics*, *30*(5), 803–831.

Weiss, S., & Siegwart, R. (2013). Real-Time Metric State Estimation for Modular Vision-Inertial Systems, *231855*, 4531–4537.

Weyand, T., Kostrikov, I., & Philbin, J. (2016). PlaNet - Photo Geolocation with Convolutional Neural Networks. *Arxiv*. Retrieved from http://arxiv.org/abs/1602.05314

Wu, A. D., Johnson, E. N., & Proctor, A. A. (2005). Vision-Aided Inertial Navigation for Flight Control. *Journal of Aerospace Computing, Information, and Communication*, *2*(9), 348–360. https://doi.org/10.2514/1.16038

Xu, A., & Namit, G. (2008). SURF : Speeded - Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, *110*, 1–30.

Yassin, M., & Rachid, E. (2015). A survey of positioning techniques and location based services in wireless networks. *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, (Ii), 1–5. https://doi.org/10.1109/SPICES.2015.7091420

Zaremba, W., & Sutskever, I. (2014). Learning to execute. *ArXiv Preprint ArXiv:1410.4615*.

Zhang, H., Li, B., & Yang, D. (2010). Keyframe detection for appearance-based visual SLAM. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2071–2076. https://doi.org/10.1109/IROS.2010.5650625

Zhang, M.-L., & Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Granular Computing, 2005 IEEE International Conference on* (Vol. 2, pp. 718–721).

Zingg, S., Scaramuzza, D., Weiss, S., & Siegwart, R. (2010). MAV navigation through indoor corridors using optical flow. *Proceedings - IEEE International Conference on Robotics and Automation*, 3361–3368. https://doi.org/10.1109/ROBOT.2010.5509777