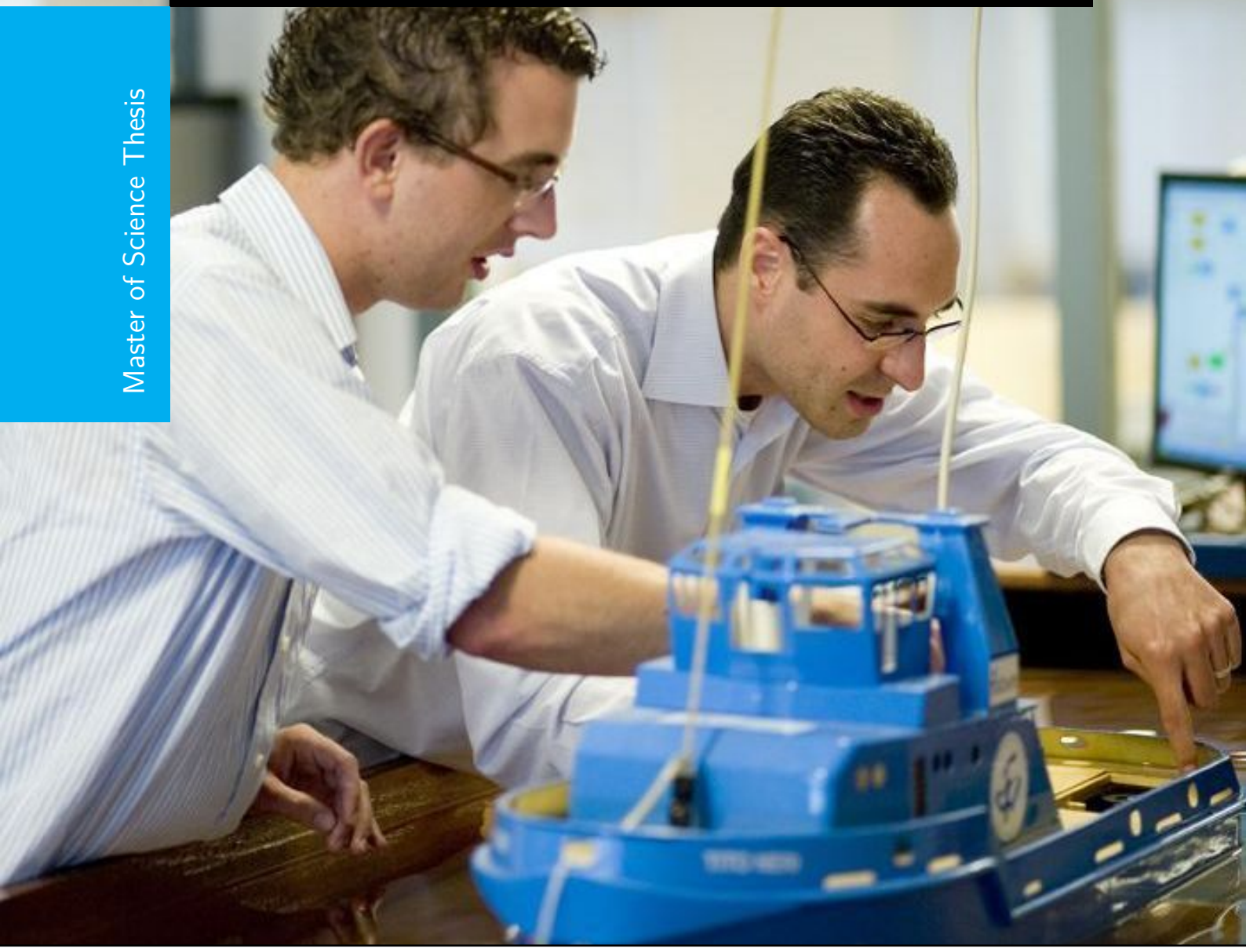
A Sensor Data Fusion Algorithm for Human Motion Estimation

V.S.Raghavan

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

A Sensor Data Fusion Algorithm for Human Motion Estimation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

V.S.Raghavan

October 26, 2016

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

In the elderly population, falls are one of the major causes of injuries. Fall detection algorithms for wearable devices in the literature were found to focus on differentiating Activities of Daily Living (ADL) from falls, rather than on early fall detection and prevention before impact. This thesis work was aimed at providing accurate estimates of human motion parameters like translational velocities of the center of mass, so as to aid early fall detection algorithms in the future. An algorithm which fuses visual and inertial data obtained from a sensor setup consisting of a pair of cameras and an Inertial Measurement Unit (IMU) was developed. An Extended Kalman Filter (EKF) framework was used for sensor data fusion. A neural network was trained to map motions of interesting points or features obtained from image processing, to actual motion of the sensor setup attached to the hip of a person. This neural network provided the correction term to the EKF in the form of 3 dimensional(3D) translational velocities which is an important motion parameter for the detection of falls. First, this algorithm was trained and tested for hand-held sensor setup motions. Acceptable velocity tracking was observed for slow motions. Then the algorithm was trained and tested on motions of a test subject. The accuracy of estimation of 3D translation velocities and in particular the vertical component of the 3D velocity was studied for two distinct sets of activities or motions namely the walking motion and the sitting down/standing up motion. It is shown that the combination of the EKF and the neural network is capable of reacting and tracking the velocities for the sitting down/standing up motion.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors Dr. Pieter Jonker and Dr. Heike Vallery for providing me with the opportunity to work on this thesis. I am extremely grateful to them for everything they have taught me about doing research. They have been a constant source of guidance and have always been there to clear my doubts. I would also like to thank my thesis committee chairman Dr. Robert Babuska, for prompting me to submit initial drafts of the thesis report earlier than I had planned. Without this prompting I might not have completed this report on time.

I would like to thank Daniel Lemus for constant support during the literature survey phase and for teaching me how to use the Qualisys Motion Capture system. I am also thankful to Patricia Baines for the various discussions we had on human movements and analysis for reports during the later stage of the thesis. I would like to also thank Floris Gaisser for providing me with the stereo camera setup and for the initial discussions about using neural networks.

I am very grateful to Dr. Aswin Chandarr for taking time out of his schedule during the process of his PhD graduation, to listen to my ideas and give feedback on them. I am also thankful to my friends Aashish Vatsyayan and Shravan Tata for the memorable discussions we had on topics ranging from neural networks and robotics to philosophy of life. I am thankful to Shravan for also allowing me to use his desktop computer to train neural networks from time to time. I am very grateful to Aswin and Aashish for proofreading my report.

Lastly, I would like to thank my friends and family back home in India, for the constant encouragement they provided throughout the whole process.

Delft, University of Technology V.S.Raghavan
October 26, 2016

"You never change things by fighting the existing reality. To change something, build a new model that makes the existing model obsolete."

— *R. Buckminster Fuller*

Chapter 1

Introduction

Wearable health monitoring devices are expected to revolutionize the healthcare industry. There have been many devices like the Fitbit Blaze watch [1] and Jawbone [2] which can monitor a person's heartbeat, and activity levels. In this era of growing wireless sensor technology and big data processing, one of the major health issues which is still being researched upon is the timely detection of falls and prevention of impacts from fall events. Early fall detection is a very important problem, more so for the elderly population. The study in [3] states that adults over 65 years of age have suffered the most number of fatal falls. Another report by the the World Health Organization(WHO) [4] states that at least 28-35% of the elderly people above the age of 65, experience at least one fall per year. The early detection of falls is a challenging problem primarily due to the similarity in motions of daily life activities and motions that lead to a fall. For example, the motions of slowly slumping onto a chair and the motions at the initial stages of a slow fall are very similar. This gives rise to false positives in early fall detection. To reduce the false positives, it is intended to study motion parameters of the center of mass like translational velocities, height above the ground, and orientation change.

This thesis works towards the goal of developing an algorithm, capable of providing these motion parameters in real time using a low cost sensor setup and requiring minimal computing power. This algorithm will be a base for further motion analysis. If the algorithm can provide reliable estimates of motion parameters, then with further analysis, it would possible to easily distinguish between motions of Activities of Daily Living (ADL) and motions leading to falls. Early detection of the motions leading to falls can be used to deploy an impact reducing or preventing countermeasure like an airbag. This countermeasure will reduce or prevent any injury due to the impact of falls. This could in turn reduce the degradation of mobility of the elderly and also reduce healthcare costs. An Inertial Measurement Unit (IMU) and a stereo camera setup as shown in 1-1 are used in this thesis. An Extended Kalman Filter (EKF) framework is used to fuse data from the IMU with the data obtained from the stereo camera setup. This thesis will look into the implementation concepts of vision based motion estimation or visual odometry, sensor fusion and applying these concepts for estimating human motions.

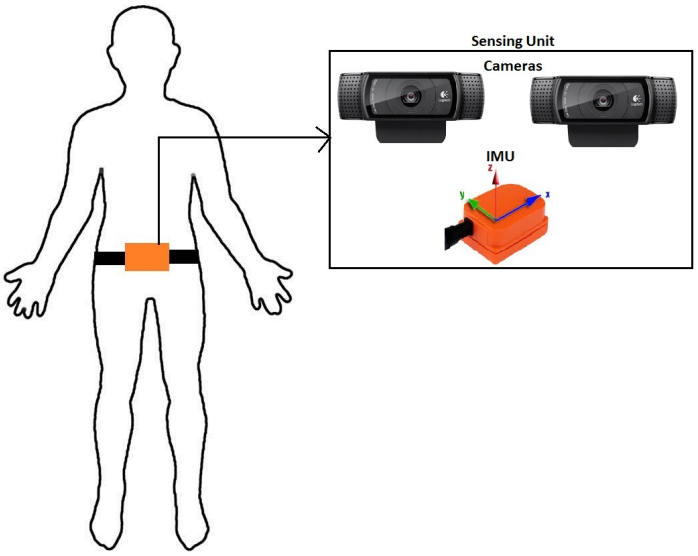


Figure 1-1: Schematic of the proposed sensing system. The camera(s) and IMU are attached to the body of the person using a belt. (Human silhouette source:[5])

1-1 Wearable Device Fall Detection Algorithms in Literature

Previous works on fall detection using wearable sensors have focussed primarily on two methodologies. The first involves thresholding quantities based on raw acceleration values obtained from the IMU. One such example of a popular quantity is the sum vector(SV) defined as $SV = \sqrt{A_x^2 + A_y^2 + A_z^2}$ where A_x, A_y, A_z are the raw acceleration values about the X, Y and Z axes obtained from the IMU. The authors of [6], studied and compared the performance of a few popular thresholding based algorithms on data from real fall events. The accuracy of detection of falls by the algorithms compared in [6] varied from 13% to 96% when tested with data obtained from real falls . Additionally, the study in [7] obtained higher sensitivity of fall detection for the case where the IMU was strapped on to the waist when compared to the cases where the IMU was attached to the head and the wrist. The main advantage of these thresholding based algorithms was that they could be implemented easily on due to their low computational costs.

The second methodology involved using supervised machine learning techniques such as k-nn classifier, bayesian classifier and neural network classifier. The studies in [8] and [9], presented and compared performance of the above mentioned classifiers in distinguishing normal ADL from falls. Features used train the classifiers were typically Fourier transform peaks, wavelet transform coefficients that were extracted based on a sliding window of SV of accelerations. The accuracy obtained for all the methods ranged from 88% to 90% with the neural network giving the highest accuracy. The computation time of these algorithms can be expected to be higher when compared with the thresholding based algorithms.

The major disadvantage of the above discussed methodologies and algorithms studied in the literature is that, they do not focus on early detection of fall but rather on distinguishing of the fall from ADL. It was also concluded that the majority of the algorithms that depended purely on IMU data, relied on the sudden change in orientation and increase in acceleration

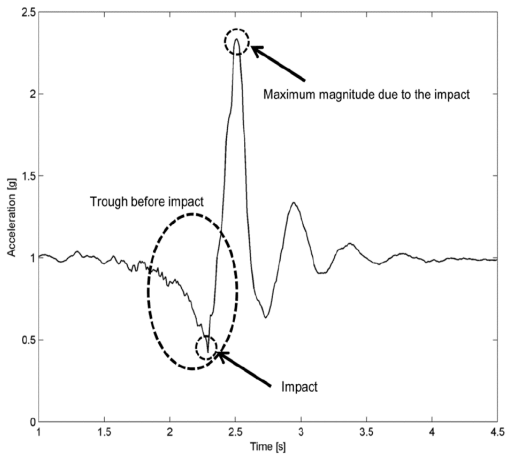


Figure 1-2: Common Pattern of SV during the Fall.(Source: [6])

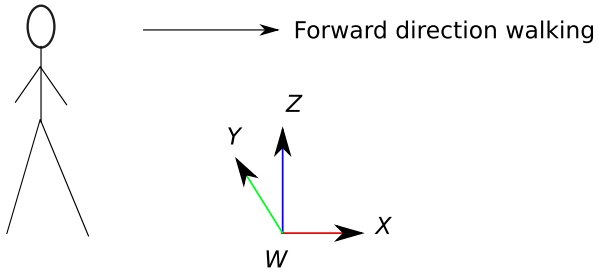


Figure 1-3: Representative diagram of the 3D coordinate axis used in the thesis. The X and Y axes form the horizontal plane and the Z axis constitutes the vertical axis facing upwards. The positive X axis typically faces forward in the direction of vision of the person. W is the origin of the World coordinate frame and it is situated on the ground in the experiments conducted in the thesis. All the measurements are either expressed or transformed to this world coordinate frame.

due to impact of the fall. This in turn led to very little or no time being available to deploy any impact preventing counter measure. Hence these algorithms were not suitable for early fall detection. Furthermore, very scant information is available in the literature about the real time performance of these algorithms, as most of the results have been based on offline performance evaluation.

It is believed that to detect falls early analysis on more human motion parameters is needed. In particular translational velocities are very important in determining how the person is moving between two time instances. The vertical component of the velocity(along the Z axis) is the most important of the three components as it gives us an idea of how quickly the person in descending to the ground. Determining translational velocities and the height of the center of mass above the ground can greatly help in reducing false positives as well. To obtain reliable estimates of motion parameters, we need a an algorithm which tracks or observes human motion. Although, relying on IMU alone, will not provide us with accurate estimates of frame-to-frame motions or velocities. It is proposed to use a pair of cameras or a stereo camera setup along with the IMU. An image processing module will perform operations on the images from the the stereo camera setup. Data from the IMU and the image processing

module will be fused using a sensor data fusion algorithm. The next session briefly explains the need for sensor fusion and various frameworks that perform sensor fusion.

1-2 Sensor Data Fusion Frameworks for Motion Estimation

As mentioned earlier, obtaining translational velocities of the center of mass of a person using just IMU data is difficult. Integrating accelerations and simple low pass or band pass filtering leads to accumulation of drift. One can only expect to obtain good estimates of orientation alone by using only IMU measurements and filters like Kalman, or complementary filters. Whereas for our application, accurate estimates of translational velocities are required. Similarly, relying on information on image processing alone has its own set of disadvantages. The vision system has a inherent weakness towards tracking pure rotations. Furthermore the motion estimates which can be obtained from the vision system are affected by noise as well. It is for this reason that the majority of the researches use sensor fusion tools such as Kalman filters and particle filters for simultaneous localization and mapping of robots and visual inertial sensor fusion. Their aim is to obtain accurate estimates of position and velocity by combining data from multiple sensors.

Human motions are non-linear in nature and cannot be easily generalised. The data obtained from vision sensors and the IMU are non-linear in nature as well. There are three popular frameworks or tools which are used for fusing visual and inertial data and estimating non-linear motion. They are the EKF, Unscented Kalman Filter(UKF) and particle filters. As mentioned earlier, in this thesis we aim to develop an real time computationally efficient algorithm. The algorithm must be capable of giving estimates at a high frame rate as we intend on detecting the falls early. Furthermore, we can expect more computations after the estimates of velocities has been obtained from the sensor fusion algorithm. Hence, it is necessary to have an algorithm which has very low computation time as well. Based on this necessity, we compared the performance of the above mentioned three tools in the literature. Many studies including the the ones in [10] and [11] have found that the EKF has the lowest computation time of the three mentioned filters. Although, the downside to the lower computation time of the EKF is its slightly lower accuracy when compared with particle filters or UKF. In spite of this downside, EKF is the most popular of the three for visual inertial sensor data fusion.

There has been an increasing interest in visual inertial sensor fusion in recent times due to advancement in hardware and easy availability of the sensors and the software platform required to process data in the form of smart phones. Many open source sensor fusion algorithms based on EKF like [12], [13] and [14] exist. Typically these algorithms were developed for deployment on quadcopters and wheeled robots. The algorithm in [15] was tested with setup consisting of the camera and IMU strapped on to a helmet on the head of the person. The algorithm presented accuracy in terms of drift from loop closure and the drift was found to be about 1.01% for walking motions.

In spite of the plethora of algorithms, very few have been explicitly tested on varying human motions like walking, bending down to pick up objects and sitting down. No particular investigation has been done to see if the existing algorithms provide accurate frame-to-frame motion information for different human motions. In this thesis we attempt to develop a computationally efficient algorithm for the specific purpose of estimating motion while walking,

sitting down and standing up. For this we will attempt to use all the possible data available to us from the cameras and IMU. The EKF framework is chosen for the flexibility it offers in fusing different data and its low computation cost. The thesis also uses a simple neural network in the EKF framework to reduce the number of computations and this will be discussed in later sections.

1-3 General Flow of EKF Based Sensor Data Fusion Algorithms

The figure 1-4 shows a general framework of the implementation of EKF for the purpose of sensor fusion. Various EKF based algorithms in literature differ in their definition of prediction models and state correction or measurement update methodologies. These differences and the flow of data in the EKF will be discussed in this section. In this thesis the vectors will be in bold italic font,the scalars in italics and the matrices will be in normal bold font. The axis coordinates will also be written in italics.

1-3-1 IMU Based Prediction Models

For the purposes of prediction, standards kinematic equations of motions like the equations below are used.

$$\begin{aligned} \dot{\boldsymbol{q}} &= \frac{1}{2}\Omega(\boldsymbol{w})\boldsymbol{q} \\ \dot{\boldsymbol{p}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{a} \end{aligned} \tag{1-1}$$

In the above equations $\boldsymbol{q}$ is the quaternion representing orientation of the body with respect to the world coordinate system, $\boldsymbol{v}$ is the 3D velocity vector and $\boldsymbol{p}$ is the 3D position of the body expressed in the world coordinate frame. a and w are the 3D accelerations and angular velocities. Quaternion is commonly used to represent orientation and is preferred over the roll, pitch and yaw angles due to the advantage of efficient computations when compared with rotation matrices and Euler angles and also the absence of the Gimbal lock problem. The accelerations and angular velocities are obtained from the IMU, but some processing is required. The accelerations need to be bias and gravity compensated, and the angular velocities need to bias compensated.

Sensor fusion algorithms in literature also differ in how they define their states. The above mentioned variables of quaternion, velocity and positions are staple in most of the algorithms. Some algorithms like [16] and [17] add the bias of the accelerations and they gyroscopes as states to be estimated. In [16], N previous IMU poses are also added to the state vector.

1-3-2 Image Processing Based Correction/Measurement Update

There are various correction methodologies used in the literature. Typically the outputs from the image processing module, like feature pixel positions, pose or velocity of the camera are used for correcting the IMU based state predictions. For example, the algorithm [16]

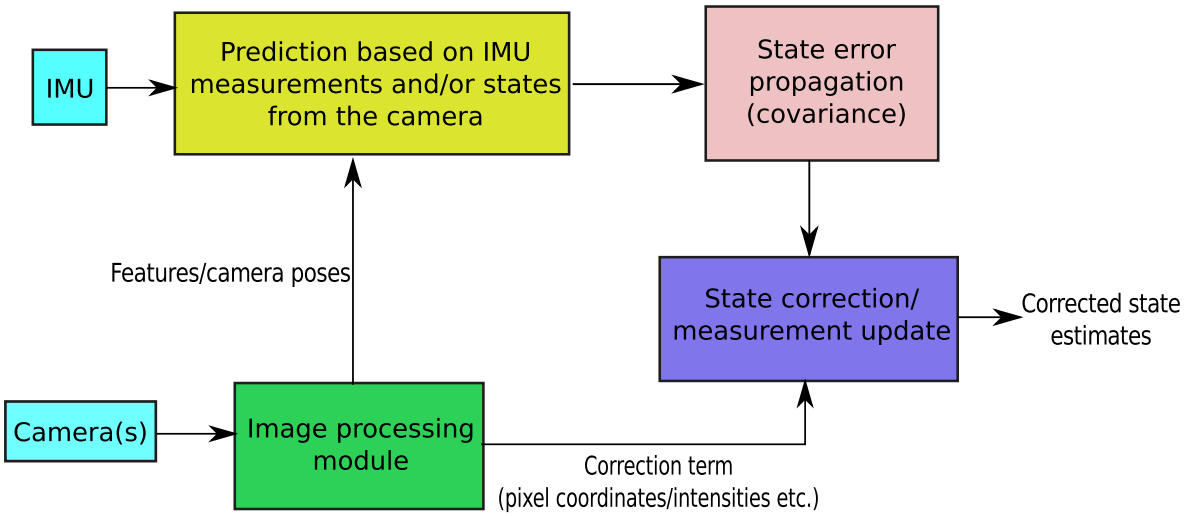


Figure 1-4: General framework of EKF based visual inertial sensor data fusion algorithms

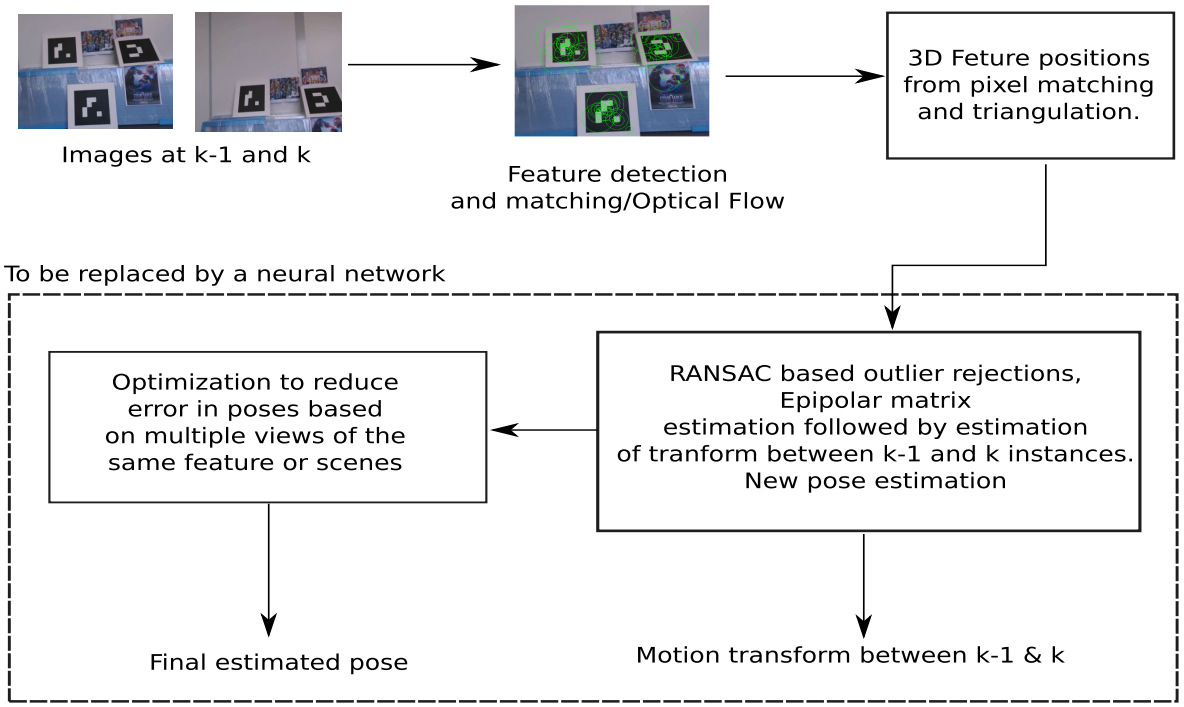


Figure 1-5: General flow of vision based motion estimation. The parts of the algorithm enclosed by the dashed rectangle will be replaced by a neural network.

uses the pixel positions of features observed from different poses, whereas the algorithm [17] uses velocity of camera motion obtained from optical flow estimation for correction. In [12], the measurement update is based on the error in pixel intensities of the re-projected feature points.

Processing of the images from one time instant to another also differs from one algorithm to another. A very general framework of the processes involved in the image processing module is shown in figure 1-5. The process of obtaining motion information from images is termed as visual odometry. Typically images are obtained for two time instances $k-1$ and k. Interesting points or features are found in the images for tracking the points from one frame to another. Once detected these features are matched and the motion of the pixel, along with the calibration of the cameras, is used to obtain information on the motion of the point. There are many feature detection and matching algorithms in the literature like the Scale invariant feature detection (SIFT)[18], Speeded Up Robust Features (SURF) [19], Harris corner [20], Binary Robust Invariant Scalable Keypoints (BRISK) [21], etc. In the literature survey SURF and BRISK were found to be the most suitable as the computation needed for detection and matching happens in real time, and these features provide accurate matching in the presence of image blurs and sudden viewpoint changes as observed in the study [22]. BRISK was found to be computationally faster than SURF when tested in MATLAB but finally SURF was used in the thesis, as it gave more number of features per frame in the scene of where the experiments were carried out when compared to BRISK.

A stereo camera setup is used in this thesis which allows for obtaining the approximate 3D position of a feature point with respect to the camera for every time instant without the need for any motion unlike the case when monocular cameras are cameras are used. Based on the 3D motion of features and 2D pixel motion of the features in the images, an outlier rejection using the popular Random Sampling and Consensus (RANSAC) is done to reduce the noise in the motion estimation. Then, using the inliers, a motion transform T consisting of rotation matrix $\mathbf{R}$ and translation vector $\boldsymbol{t}$ is obtained by either estimating the essential matrix, or by optimizing to reduce the error in the the feature/pixel positions as a result of a given transformation $\mathbf{T}$. Finally, if a feature has been observed from many different poses, then a further optimization is done to correct for the poses based on feature pixel re-projection error.

The optimization/iteration based motion transform estimation from the motion of features is typically never completed in a constant amount of time. When very little features are present, the accuracy of these algorithms using RANSAC decreases. Since we require a constant and steady stream of motion parameter estimates to detect falls early, it is proposed to used a pre-trained neural network to estimate frame-to-frame motion or motion transform between two time instances, as a constant computation time can be expected from a neural network. The next section explains in detail, the motivation behind using neural networks.

1-4 Motivation for Using Neural Networks

Consider a set of features $F_i, i = 1, 2, 3, ...N$. Let the 3D coordinate of feature i in the $k-1$ and k time instances respectively be $\boldsymbol{X}_{ik-1}$ and $\boldsymbol{X}_{ik}$. These 3D coordinates are usually expressed in coordinate frame fixed on the observing camera at $k-1$. Let the pixel coordinates of

feature F_i be $\boldsymbol{x}_{ik-1}$ and $\boldsymbol{x}_{ik}$ in the k and $k-1$ time instances. Let the transformation of the camera pose from the $k-1^{th}$ instant to the k^{th} instant be $\mathbf{T}_{k-1,k}$. In general, two methods can be employed to obtain $\mathbf{T}_{k-1,k}$. If the 3D feature coordinates are available, as is the case for every stereo image pair acquired, then an optimization with the following cost function;

$$\arg\min_{\mathbf{T}_{k-1,k}} \Sigma||\boldsymbol{x}_{ik} - \hat{\boldsymbol{x}}_{ik-1}||^2 \tag{1-2}$$

where $\hat{\boldsymbol{x}}_{ik-1}$ is the reprojection of the of the feature coordinate $\boldsymbol{X}_{ik-1}$ transformed by $\mathbf{T}_{k-1,k}$, on the image at time instance k. The optimization can be performed with RANSAC to eliminate outliers as well. The second method is to optimize the following cost function involving only the 3D feature coordinates;

$$\arg\min_{\mathbf{T}_{k-1,k}} \Sigma||\boldsymbol{X}_{ik} - \mathbf{T}_{k-1,k}\boldsymbol{X}_{ik-1}||^2 \tag{1-3}$$

The paper [23] states that the second of the two optimization methods mentioned above is less advantageous, as noises in estimation of 3D coordinates of the features introduces high uncertainty in the estimate of $\mathbf{T}_{k-1,k}$. The noise is typically caused by camera calibration which is used for estimating 3D coordinates of features in a stereo camera setup, using triangulation. Nevertheless, estimation of the transform between two time instances, or the motion between two time frames requires some form of iterative optimization or outlier rejection. As mentioned earlier, a constant computation time for optimization based motion estimation cannot be expected and sometimes the number of iterations may be too high and the process consumes a lot of processing power. As shown in the figure 1-5, this thesis will attempt to replace the epipolar matrix/transform estimation, RANSAC and optimization iterations, with a neural network. A neural network which is trained using Levenberg-Marquardt back-propagation, is a simple optimization to reduce the error between the predicted outputs and the target outputs by varying the weights. This neural network attempts to replace the optimization cost functions mentioned above by pre-learning the noises affecting the system and the mapping between motion of features and the actual motion of the camera.

The typical noises affecting this estimation are the calibration noise introduced by modelling the camera using the pinhole model, inaccuracies in feature pixel positions during the matching process, inaccurate matchings and image blurs. Knowing this, is it possible for a system to learn these noises, especially the feature matching and calibration noises and estimate the motion parameters between two time frames, specifically the translational velocities, with reasonable accuracy? Can these velocities be used for state correction in the EKF. This brings us to the our research question.

Research Question

How accurate is the estimation of translational velocities for human motions, if an EKF framework is provided with correction updates from a pre-trained neural network which maps image feature motions in the camera frame to actual motion of the camera in the world frame?

The interest in using a neural network spawned from simple intuitive observations from real life. For example, say we are observing a chair with our eyes. An average adult brain does

not know the exact distance of the chair in meters with 100% certainty. What it knows is a rough estimate, which it has learned from various forms of feedback, especially the feedback obtained by moving towards the object. This learning is so advanced that even if we shut one eye off, by moving towards the object, we have a rough idea as to how far the object is or how far we need to walk to reach the object. This is the primary motivation behind using neural networks in this thesis. It is hoped that if the correct inputs are provided from the image processing module, then the neural network might be able to predict the motion between frames.

1-5 Overall Framework of the Algorithm

The framework followed in this thesis will be explained in this section. A standard EKF sensor fusion framework is slightly modified. A prediction model does prediction of motion variable like orientation, velocity and position, based on sensor outputs from the IMU. The state error is propagated through covariance matrix calculations similar to the standard EKF framework. When images are obtained from the stereo camera system, stereo processing and triangulation is done, followed be matching of features in the previous frame with the features in the new frame. The information about the features like 3D motion and 2D pixel motions are sent to the neural network which provides the correction term or measurement update in terms of translational velocities. The overall flow of the algorithm is very similar to the one in 1-4. The idea is to replace the computations required by optimization based motion transform estimation, with the simple low complexity neural network.

The remainder of the thesis in arranged as follows;

- Chapter 2 describes the algorithm presented in the thesis for the purpose of human motion estimation. The image processing module, structure and training of the neural network and the EKF framework are discussed in detail.
- Chapter 3 discusses the results of the experiments conducted in this thesis.
- Chapter 4 presents the conclusions and limitations of the algorithm presented.

Chapter 2

Neural Network Training and Filter Development

In the previous chapter, the general frame work of the motion estimation algorithm was discussed. In this chapter, the details of the algorithm implemented in this thesis are presented. The algorithm mainly involves using the neural network to generate correction values for the states predicted by the Kalman predictor. An EKF based on a models presented in [16], [17] and [24] is implemented. The sensor setup consists of two Logitech C920 cameras and an Xsens MTi-X IMU. The cameras are capable of providing images at 30 Frames Per Second (FPS). The raw accelerations, angular velocities and orientation quaternions are obtained from the IMU at 100 Hz. Both cameras and IMU are fixed rigidly onto a steel plate. Qualisys motion capture cameras are used for obtaining target data for the training of neural network as well as ground truth data. No external hardware is used for syncing the data from the sensor setup and Qualisys motion capture cameras. The sync between the stereo cameras and IMU is achieved though software via the Robot Operating System(ROS). Data is collected from the sensor setup and motion capture cameras and converted to a suitable format for processing in MATLAB. The neural network is trained based on inputs from the image processing module, IMU data and motion capture data from the Qualisys motion capture system. The image processing module uses SURF feature detection and matching, triangulation and Kanade Lucas Tomasi (KLT) optical flow algorithms from the MATLAB Computer Vision toolbox [25]. The neural network is trained using the MATLAB Neural Network toolbox [26]. The first section of this chapter will discuss the coordinate systems and symbols used in the thesis. This will be followed by the details on the neural network training algorithm. Finally the EKF and the fusion of the neural network output with EKF will be discussed in detail.

2-1 Coordinate Systems and Symbols

Figure 2-1 depicts the important coordinate frames used in the algorithm. C represents the frame fixed on the left camera of the sensor setup which consists of two cameras and an

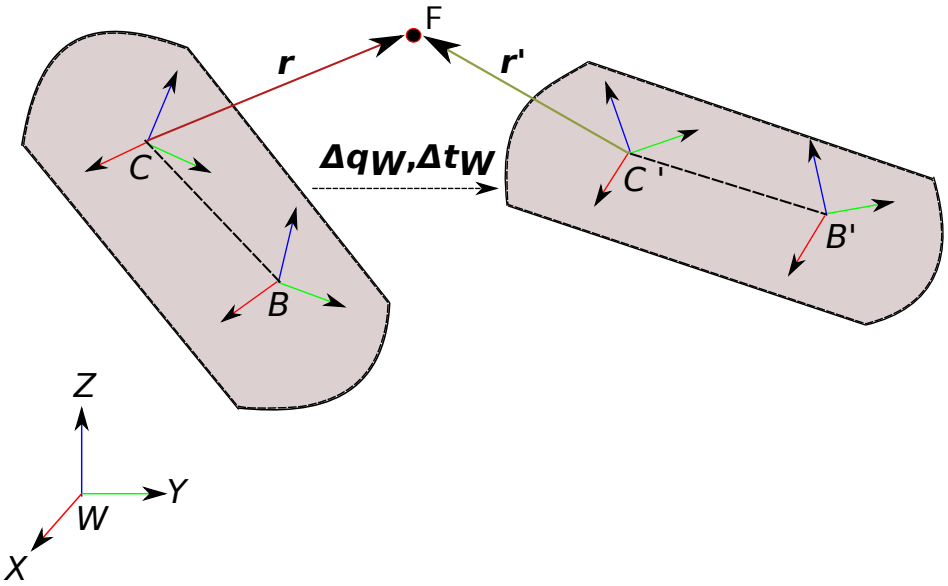


Figure 2-1: Coordinate frames used in this thesis. B is the frame fixed on the IMU and C is the frame fixed on the camera. The figure shows two time instances of the sensor setup represented by the area shaded grey, observing a feature F

X-Sens IMU. B represents the coordinate frame fixed on the IMU and W represents the world coordinate frame system. There is fixed rigid transformation from the IMU frame B to camera frame C as the sensor setup is rigid and fixed. Consider two time instances $k-1$ and k. C and C' represents the position of the frame fixed on the camera at the $k-1^{th}$ and k^{th} instant respectively. The same applies to the IMU fixed frames B and B'. The following symbols represent the various transformations between the coordinate and time frames;

- ${}^{C'}\boldsymbol{q}^{C}_{\boldsymbol{W}}$ represents the orientation quaternion transform of the camera fixed frame from the $k-1^{th}$ time instant to the k^{th} time instant expressed in the world coordinates system W. For purposes of brevity this shall be expressed as $\boldsymbol{\Delta q_W}$.

- ${}^{C'}\boldsymbol{t}^{C}_{\boldsymbol{W}}$ represents the translation of the camera fixed frame between time instants $k-1$ and k expressed in the world coordinate system. For purposes of brevity it shall be represented by $\boldsymbol{\Delta t_W}$.

- In the figure 2-1 the point F represents an interesting feature point which is observed by the camera at two separate instances. The vectors $\boldsymbol{r}$ and $\boldsymbol{r'}$ are the 3D vectors from the origin of the camera fixed frames to the feature point in the $k-1^{th}$ time instant to the k^{th} time instant respectively. In other words, the 3D positions of the feature is expressed with respect to the left camera fixed frames C and C'.

With these coordinate systems, we will discuss the details of the training inputs and outputs for the neural network.

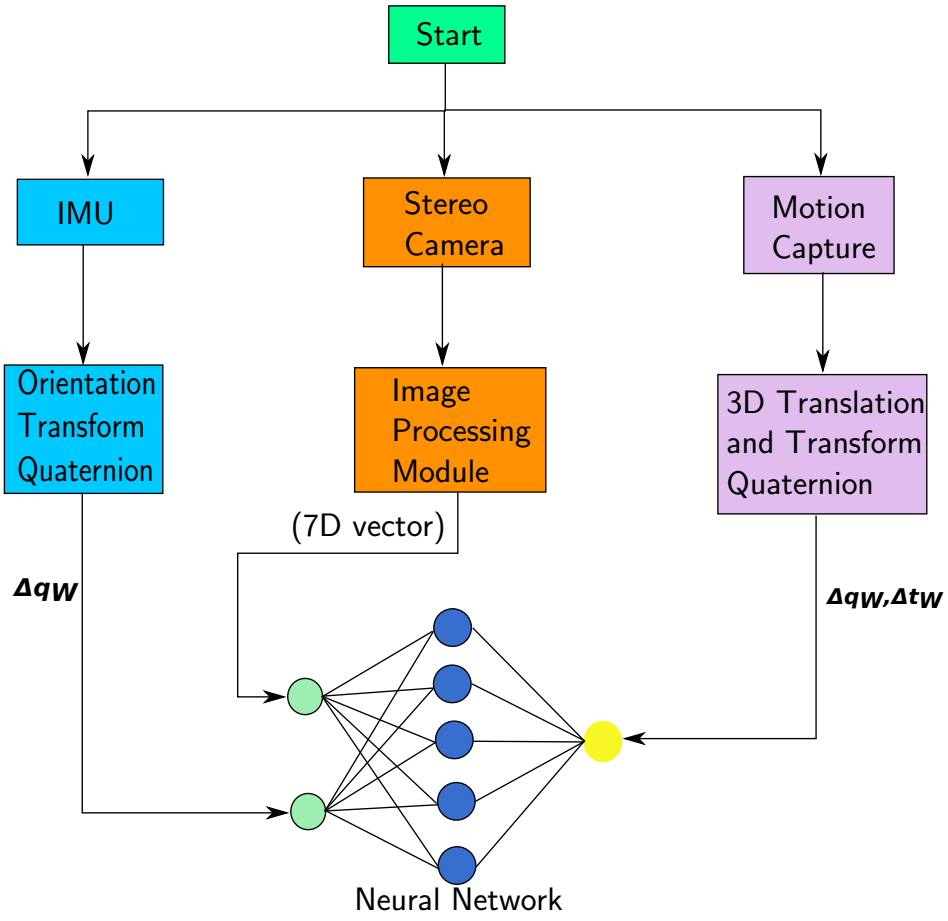


Figure 2-2: Overview of the neural network training algorithm

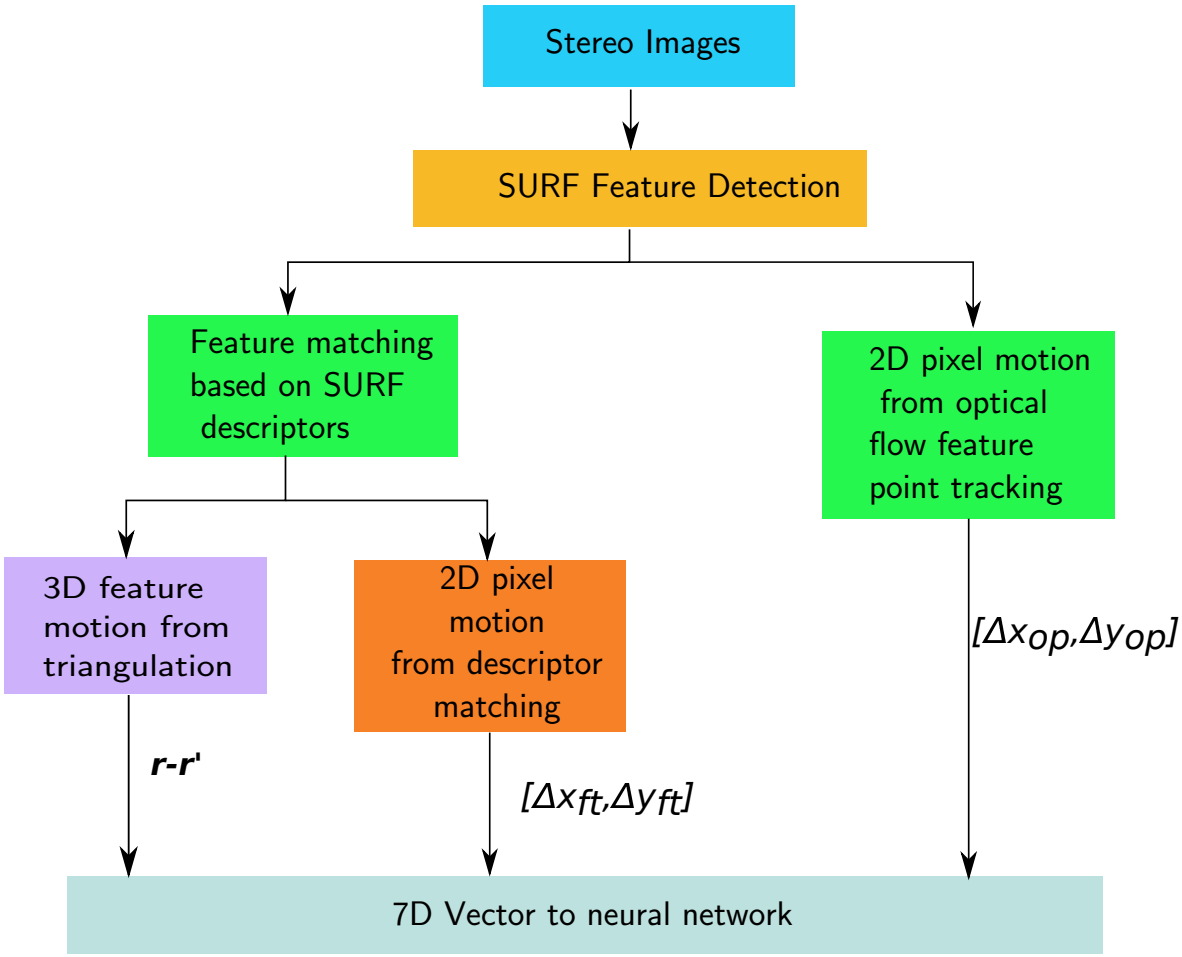


Figure 2-3: Overview of the image processing module

2-2 Neural Network Training

2-2-1 Inputs to the Neural Network

As mentioned earlier, a back propagation based artificial neural network is used in this thesis. The hidden nodes or neurons have sigmoid activation functions. The Xsens IMU provides a 4D quaternion. This can be replaced with a Kalman filter generating this quaternion from raw acceleration and angular velocity values as well. In the future versions of this algorithm, this quaternion will be generated using the open source algorithm presented in [27]. In this thesis though, the quaternion provided by Xsens IMU is used. The motion capture tracks the motion of three markers, one on each of the cameras and one on the IMU. Figure 2-2 gives a brief overview of the algorithm used in training the neural network. The training is done offline.A back propagation neural network of 2 hidden layers of 20 neurons each was chosen. The training inputs and the algorithm which generates the input will now be discussed.

First, the inputs from the image processing module will be discussed. Figure 2-3 depicts the steps.

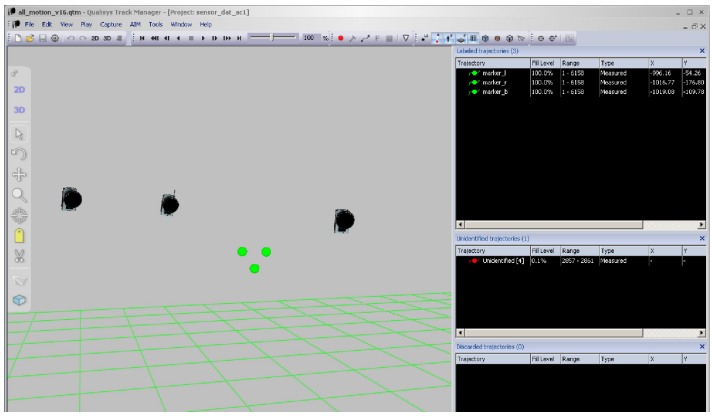


Figure 2-4: Screenshot of Qualisys Motion Capture system tracking three markers. The green circles/spheres are the markers being tracked

- Consider two time instances $k-1, k$. We obtain the image pairs (I_{lk-1}, I_{rk-1}) and (I_{lk}, I_{rk}) where the subscripts l, r stand for the left and right camera images respectively.
- SURF feature detection and matching is first performed between the left and right image pair of each frame, followed by triangulation of successfully matched feature points.
- The above step is repeated for the images obtained in the k^{th} instance. This is followed by matching of features between the images at the two time instances $k-1$ and k. Consider a successfully matched and triangulated feature F as shown in figure 2-1. We obtain two 3D vectors representing $\boldsymbol{r}$ and $\boldsymbol{r'}$ representing the coordinates of the detected features in the two camera frames C and C'. The difference of the 3-axis components of the vectors $\boldsymbol{r}$ and $\boldsymbol{r'}$ is stored. Basically we just subtract the corresponding X, Y and Z coordinates of the triangulated features and store them to be sent to the neural network. Let this difference be denoted by $\boldsymbol{r} - \boldsymbol{r'}$.
- Let the pixel coordinates of feature F at the $k-1^{th}$ instant be $[x_{ft}, y_{ft}]$ and the the corresponding pixel in the kth instant is $[x'_{ft}, y'_{ft}]$. These pixel coordinates are obtained from SURF feature descriptor matching.The difference in 2D pixel positions or pixel motion $[x_{ft} - x'_{ft}, y_{ft} - y'_{ft}] = [\Delta x_{ft}, \Delta y_{ft}]$ is stored.
- Additionally sparse optical flow is performed on features successfully matched between the frames $k-1$ and k and the 2D pixel motion obtained from the optical flow,$[\Delta x_{op}, \Delta y_{op}]$ is also stored.
- Thus every feature gives a 7D vector,$[\boldsymbol{r} - \boldsymbol{r'}, \Delta x_{ft}, \Delta y_{ft}, \Delta x_{op}, \Delta y_{op}]^T$, which is concatenated with orientation transform quaternion between $k-1$ and k time instances, namely $\boldsymbol{\Delta q_W}$ from the IMU and finally it is given as a 11 dimensional input vector to the neural network.

Typically in case of deep learning or neural network learning involving images, the neural network is given the whole image or patches of images and it is allowed to choose the features. In this thesis we don't intend on using deep learning techniques. Furthermore, we are using the neural network as a function fitter and not a classifier. If the inputs were given in the

form of images, or patches of images, then it won't be very generic for every motion, and huge amounts of data would be needed for sufficient generalization of the neural network prediction. By using only difference in 3D feature positions and 2D feature pixel positions in the two frames and further training to predict only the motion between two frames and not absolute positions or orientations, we can have sufficient generalization with very little but representative data.

2-2-2 Outputs of the Neural Network

The motion capture cameras as depicted in the figure 2-4, track three markers, one on each of the two cameras on the sensor setup and one on the IMU. From the motion capture side, two vectors are obtained. First is the 3D translation of the marker on the left camera between the time instances $k-1$ and k. Second is transform quaternion between the orientations of the plane formed by the three markers in the $k-1$ and k time frames. The two quantities are concatenated to form final the target output vector for training the network and they are represented by the $[\boldsymbol{\Delta q_W}, \boldsymbol{\Delta t_W}]^T$. This was the case in initial versions of the algorithm. In the later versions of the neural network though, only $\boldsymbol{\Delta t_W}$ was used for training the network. This choice was made because in the later stages when training for human motions, removing the quaternion term and reducing the number of outputs of the neural network to 3 improved the accuracy of the neural network predictions of Z axis translations. Furthermore, turning about the Z axis or yaw causes the axis in the direction of forward motion to be no-constant. Since the velocities in the $X-Y$ plane is slightly less important than the Z velocities, the motion in the $X-Y$ plane is represented as magnitude of the motion vector in $x-y$ plane and the angle of the vector with respect to the X axis. The motion in the Z direction is not modified, as the direction of the axis remains the same throughout. To illustrate this mathematically let us consider the individual 3D components of the vector $\boldsymbol{\Delta t_W} = \begin{bmatrix}\Delta x_W & \Delta y_W & \Delta z_W\end{bmatrix}^T$. Then the 3 target outputs for which the neural network is trained, form the target vector $\boldsymbol{T}_v$ and the vector is defined as follows;

$$\boldsymbol{T}_v = \begin{bmatrix} \sqrt{\Delta x_W^2 + \Delta y_W^2} \\ tan^{-1}(\frac{\Delta y_W}{\Delta x_W}) \\ \Delta z_W \end{bmatrix} \tag{2-1}$$

For each iteration, the neural network gives a set of possible 3D translation motions of the left camera in terms of $\boldsymbol{T}_v$. Say there are N_f features tracked between the two stereo image pairs. Then we obtain N_f number of possible $\boldsymbol{T}_v$ vectors. Δx_W and Δy_W components are obtained from the first two terms each of the vector $\boldsymbol{T}_v$ and along with the last term we obtain N_f number of $\boldsymbol{\Delta t}_W$ possible translations. These translations are boxed depending on the range of motions and a weighted average is used to obtain the best estimate of a 3D motion between two time frames $k-1$ and k. This weighted average translation $\boldsymbol{\Delta t}_{nn}$, is divided by the sampling time which is 0.1s to give 3D translational velocities $\boldsymbol{v}_{nn}$ where nn is a representative symbol for neural network output. This 3D translational velocity vector obtained from the neural network is used in the extended Kalman filter.

As mentioned earlier, the aim of the neural network is to create a map from the motion of a feature in the camera frames, the pixel motion in the images, and the change in orientation

in the IMU, to the actual motion of the left camera as observed by the motion capture. Typically this is done with multiple feature points using optimization based algorithms like RANSAC and those mentioned in chapter 1 to obtain the pose of the camera with respect to the world coordinate system. The motion of the hip of the person where the sensor setup is expected to be attached to is usually similar and repetitive, more so when comparing motion transformations between frames. Hence it is believed that a neural network trained with such motions will be able to produce a good general consensus of the motion between two frames. The motivation behind using optical flow as another measure of feature pixel motion was to provide every possible information for the neural network. However, after training few neural networks it was observed that removing the optical flow terms from the input reduced accuracy of the neural network. Hence it was decided to keep the optical flow as a part of the input vector. If the training and testing of the neural network is successful then it can potentially reduce the computation time for estimating the motion between two time frames substantially. This is the basis of the algorithm used in this thesis.

In the next section we will describe the extended Kalman filter structure implemented in this thesis and also how the neural network output is fused with the EKF to provide for the correction of the states.

2-3 Extended Kalman Filter Framework

In this section, the extended Kalman framework implemented will be discussed briefly. The state vector, prediction model and error propagation equations implemented are based on the algorithm presented in [16] and [17]. The state consists of 3 pose/motion parameters namely, the quaternion $\boldsymbol{q}_W$ representing the orientation of the IMU frame B with respect to the world frame W, $\boldsymbol{p}_W$ representing the position of the IMU frame in the world coordinates and $\boldsymbol{v}_W$ representing velocity of motion of the world frame. The overall state vector also consists of gyroscope bias $\boldsymbol{b}_g$ and accelerometer bias $\boldsymbol{b}_a$ which are modelled as random walk processes. The overall state vector $\boldsymbol{x}$ is as follows;

$$\boldsymbol{x} = [\boldsymbol{q}_W, \boldsymbol{p}_W, \boldsymbol{v}_W, \boldsymbol{b_g}, \boldsymbol{b_a}]^T \tag{2-2}$$

The continuous time prediction equations governing the states are given by the equation 2-3.

$$\begin{aligned}
\dot{\hat{\boldsymbol{q}}}_W &= \frac{1}{2}\Omega(\hat{\boldsymbol{w}})\hat{\boldsymbol{q}}_W \\
\dot{\hat{\boldsymbol{p}}}_W &= \hat{\boldsymbol{v}}_W \\
\dot{\hat{\boldsymbol{v}}}_W &= \mathbf{C}(\hat{\boldsymbol{q}}_W)\hat{\boldsymbol{a}} \\
\dot{\hat{\boldsymbol{b}}}_a &= 0_{3\times 1} \\
\dot{\hat{\boldsymbol{b}}}_g &= 0_{3\times 1} \\
\Omega(\boldsymbol{w}) &= \begin{bmatrix} [w\times] & \boldsymbol{w} \\ \boldsymbol{w}^T & 0 \end{bmatrix} \\
[\boldsymbol{w}\times] &= \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}
\end{aligned} \tag{2-3}$$

$C(\hat{\boldsymbol{q}}_W)$ is the rotation matrix associated with the quaternion $\hat{\boldsymbol{q}}_W$. $\hat{\boldsymbol{w}}$ is the bias corrected 3D angular velocity vector and $\hat{\boldsymbol{a}}$ is the bias corrected and gravity compensated 3D acceleration vector. They are obtained using the following equations;

$$\begin{aligned} \hat{\boldsymbol{a}} &= \mathbf{C}(\boldsymbol{q}_W)(\boldsymbol{a_m} - \boldsymbol{b_a}) + \boldsymbol{g} \\ \hat{\boldsymbol{w}} &= \boldsymbol{w_m} - \boldsymbol{b}_g \end{aligned} \tag{2-4}$$

where $\boldsymbol{a_m}$ and $\boldsymbol{w_m}$ are raw acceleration measurements, $\boldsymbol{g}$ is the vector representing gravity.

Following the standard EKF framework and equations presented in [16], the linearised continuous time propagation equations of the error in states $\tilde{\boldsymbol{x}}$ are as follows;

$$\dot{\tilde{\boldsymbol{x}}} = \mathbf{F_c}\tilde{\boldsymbol{x}} + \mathbf{G_c}\boldsymbol{n_I} \tag{2-5}$$

Here $\boldsymbol{n_I} = [\boldsymbol{n}_g, \boldsymbol{n}_{bg}, \boldsymbol{n}_a, \boldsymbol{n}_{ba}]$ is the noise parameter vector of the system. $\boldsymbol{n}_g$ and $\boldsymbol{n}_a$ are the gyroscope measurement noise and the accelerometer measurement noise respectively. $\boldsymbol{n}_{bg}$ and $\boldsymbol{n}_{ba}$ are the random walk processes equated to the $\dot{\boldsymbol{b}_g}$ and $\dot{\boldsymbol{b}_a}$ respectively. $\mathbf{F}_c$ and $\mathbf{G}_c$ are linearisation matrices for error propagation and are defined as follows;

$$\begin{aligned} \mathbf{F_c} &= \begin{bmatrix} -[\hat{\boldsymbol{w}}\times] & 0_{3\times3} & 0_{3\times3} & -\mathbf{I}_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -[\hat{\boldsymbol{a}}\times] & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -\mathbf{C}(\hat{\boldsymbol{q}}_W) \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \\ \mathbf{G_c} &= \begin{bmatrix} -\mathbf{I}_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -\mathbf{C}(\hat{\boldsymbol{q}}_W) & 0_{3\times3} \\ 0_{3\times3} & \mathbf{I}_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \end{aligned} \tag{2-6}$$

$\mathbf{I}_{3\times3}$ is an identity matrix while $0_{3\times3}$ is 3×3 matrix filled with zeroes. Consider a continuous noise covariance matrix $\mathbf{Q}_c = diag(\sigma^2_{n_g}\sigma^2_{n_{bg}}\sigma^2_{n_a}\sigma^2_{n_{ba}})$. These matrices are converted to a discrete form to allow for discrete samples based calculations based on the equations in [17] as follows;

$$\begin{aligned} \mathbf{F_d} &= \exp\left(\mathbf{F_c}\Delta t\right) \\ \mathbf{Q_d} &= \int_{\Delta t} \mathbf{F}_d(\tau)\mathbf{G}_c\mathbf{Q}_c\mathbf{G}_c^T\mathbf{F}_d(\tau)^T d\tau \end{aligned} \tag{2-7}$$

Δt is the sampling time. The state error covariance âĂćP is now propagated as follows;

$$\mathbf{P}_{k|k} = \mathbf{F_d}\mathbf{P}_{k|k-1}\mathbf{F}_d^T + \mathbf{Q}_d \tag{2-8}$$

For the measurement update, the output from the neural network ΔT_n is used. The discrete time observation function $\boldsymbol{h}$ and the actual observation $\boldsymbol{z}$ based on the algorithm in [17] are

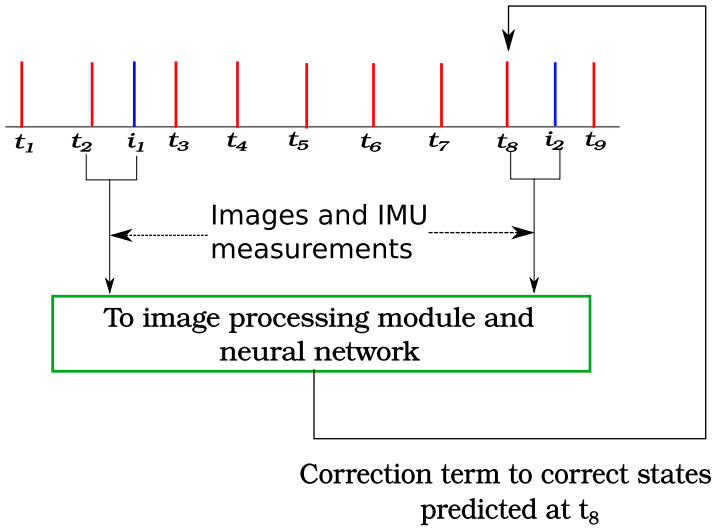


Figure 2-5: Representative diagram of the timing of the prediction and correction computations based on sensor inputs. The red lines represents the incoming IMU measurements and the blue lines represents the incoming stereo images.

described as follows;

$$\begin{aligned} \boldsymbol{h(k)} &= \begin{bmatrix} \hat{\boldsymbol{q}}_W(k) \\ \mathbf{C}(\boldsymbol{q}_B^C)\mathbf{C}(\hat{\boldsymbol{q}}_W(k))\hat{\boldsymbol{v}}_W(k) + \mathbf{C}(\hat{\boldsymbol{q}}_W)([\hat{\boldsymbol{w}}(k)\times]\boldsymbol{p}_B^C \end{bmatrix} \\ \boldsymbol{z(k)} &= [\boldsymbol{q}_W(k), \boldsymbol{v}_{nn}(k)]^T \end{aligned} \tag{2-9}$$

$[\mathbf{C}(\boldsymbol{q}_B^C), \boldsymbol{p}_B^C]$ represent the transform from the IMU frame B to camera frame C. The first versions of the algorithm substitute the actual value of q_W with values obtained from X-Sens IMU directly. Later versions of the algorithm will utilise the open source filter proposed by [27] to get actual estimates of the quaternion.

Finally the correction of the states are carried out using the standard EKF framework using the following set of equations sequentially;

$$\begin{aligned} \boldsymbol{z}(k) - \boldsymbol{h}(k) &= \mathbf{H}_d\tilde{\boldsymbol{x}} + \boldsymbol{n}_v \\ \mathbf{S} &= \mathbf{H}_d\mathbf{P}_{k|k-1}\mathbf{H}_d^T + \mathbf{R} \\ \mathbf{K} &= \mathbf{P}_{k|k-1}\mathbf{H}_d^T\mathbf{S}^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_d)\hat{\boldsymbol{x}}_{k+1|k} + \mathbf{K}\boldsymbol{z(k+1)} \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_d)\mathbf{P}_{k|k-1}(I - K\mathbf{H}_d)^T + \mathbf{K}\mathbf{R}\mathbf{K}^T \end{aligned} \tag{2-10}$$

$cvecn_v$ is the measurement noise of the whole system and $\mathbf{R}$ is its covariance matrix. $\mathbf{H}_d$ is the Jacobian of $\boldsymbol{h}$ with respect to the states.

The figure 2-5 shows timing of the prediction and correction measurements. When stereo images i_1 and i_2 are obtained, the most recently received IMU measurements before the arrival of the image are coupled with the images i_1, i_2 and sent to the neural network. Meanwhile, the prediction continues at all instants when the data from the IMU is received. The timing

interval between the images i_1 and i_2 sent for processing and neural network input generation is approximately 0.1s. The images though are obtained at 30Hz. Hence for every image obtained, the images obtained three time instances earlier and the new image are sent for neural network processing thereby maintaining an overall processing rate 30Hz. For the neural network inputs, the images have to have a time interval of 0.1s between them, as allowing more frequent image inputs, makes it difficult for the neural network to distinguish motion as the motions are much smaller in magnitude. Very low accuracy was observed when the time interval between the images sent to the neural network processing was kept at 0.033s. The IMU predictions run at roughly about 100Hz. Once the neural network provides the motion between i_1 and i_2, the correction velocity update is obtained using simple sampling based interpolation based on the 5 previously predicted velocities. Let the 3D velocities at $t_3, t_4...t_7$ be $\boldsymbol{v}_3, \boldsymbol{v}_4...\boldsymbol{v}_7$. As mentioned earlier the 3D velocity vector obtained from the neural network is $\boldsymbol{v}_{nn}$. The velocity vector $\boldsymbol{v}_c$ sent for the correction update at t_8 is as follows;

$$\boldsymbol{v}_c = \frac{0.1\boldsymbol{v}_n - 0.01(\boldsymbol{v}_3 + \boldsymbol{v}_4 + \boldsymbol{v}_5 + \boldsymbol{v}_6 + \boldsymbol{v}_7)}{0.01} \tag{2-11}$$

This interpolation is needed to ensue that there is no mismatch in the sampling rate of the prediction and the correction velocities and also allows for correctly recognizing and reacting to sudden increase in velocities.

The measurement and process equations used in the Kalman filter are very similar to those used in [24]. Inferring from the observability derivations in [24], the EKF implemented in this thesis makes the absolute 3D positions un-observable. As mentioned in the previous chapter, for the purpose of fall detection, motion between the two time frames or in other words velocity of motion is a very important quantity. 3D positions are not as important as velocities for this thesis. Hence absolute 3D position tracking is not our concern. It is hoped that fusing the neural network output with the EKF will aid in obtaining reliable velocity estimates.

The necessity for using an EKF framework along with a neural network is highlighted when the images are blurred and no proper feature correspondences are obtained. In situations like these , the EKF can still be expected to continue the prediction. The estimation by neural network is also affected by some noise and hence EKF is used in an attempt to smooth the motions obtained from the neural network as well.

This combination of the neural network and EKF shall henceforth be referred to as $\boldsymbol{Neural-EKF}$. The algorithm 1 gives an overview of the $\boldsymbol{Neural-EKF}$ algorithm.The next chapter shall describe the process of experimental data collection for two situations namely the hand held camera motions and motions of a human subject.

Algorithm 1 Algorithm for sensor data fusion using $NeuralEKF$

procedure NEURAL-EKF
  $\boldsymbol{a}, \boldsymbol{w} \leftarrow$ First IMU measurement
  $I_l, I_r \leftarrow$ First stereo image pair.
  Feature detection, matching and triangulation on stereo images
  while Not end of data stream **do**
    if IMU measurements received **then**
      $\boldsymbol{a}, \boldsymbol{w} \leftarrow$ IMU measurements
      $\hat{\boldsymbol{q}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{v}} \leftarrow$ predicted states from prediction model
    end if
    if Stereo image pair received **then**
      $I'_l, I'_r \leftarrow$ stereo image pair.
      Feature detection, descriptor matching and triangulation on I'_l, I'_r
      Feature detection and descriptor matching between $I'_l I_l, I'_r I_r$
      KLT Optical Flow estimation for matched features in I_l, I_r.
      Data to neural network.
      Weighted average of all translations obtained from neural network.
      Rejection of value if number of features is less than 5.
      Sampling and window based interpolation to determine correction velocity v_c.
      Predicted state correction update.
    else
      Continue prediction with next IMU measurement.
    end if
  end while
end procedure

Chapter 3

Experiments and Results

In this chapter we will discuss the experiments conducted with the sensor setup and analyse the results obtained. The experiments can be divided into two parts namely hand-held camera motions and motions of a human subject. First, the implementation details of the experiment will be discussed. This will be followed by results of neural network training followed by results of $\boldsymbol{Neural-EKF}$ with hand-held camera motion data, followed by the results of $\boldsymbol{Neural-EKF}$ with motion data obtained from a human subject. As discussed in the previous chapter, the neural network was trained based on features obtained in the images, the orientation of the sensor setup and the motion of the features. It was proposed to train the neural network for simple generic motions obtained from hand held camera motion as a proof of concept. Then a neural network of similar structure will be trained and tested on human motions.

3-1 Experiment Protocol and Analysis Tools

3-1-1 Protocol

The IMU data and stereo camera images were obtained through USB interfaces and Robot operating System (ROS) on the laptop and the ground truth motion was obtained from the Qualisys Motion Capture system available in the Gait Lab in the Mechanical, Maritime and Materials Engineering (3mE) department at Delft University of Technology (TU Delft). Due to the different devices and operating systems involved, the sync of the data obtained was only approximate and exact sync errors were not concentrated upon in detail.

To perform an approximate sync of the data from the motion capture cameras, IMU and the stereo cameras, two syncing motions were used, once at the beginning and once at the end of data acquisition. Here the sensor setup is turned to an approximate vertical position such that the two markers on the two sensor setup cameras are in a vertical line. Also at the same time the IMU reads an angle of about 90 degrees in the X axis and the motion capture has two of the three markers in a vertical line.

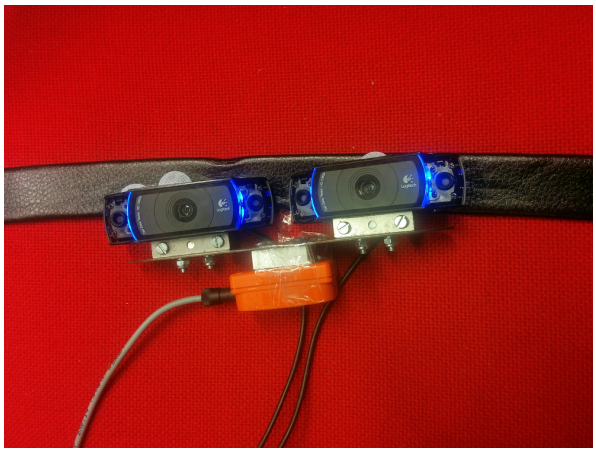


Figure 3-1: Picture of the sensor setup attached to a belt

Figure 3-2: The sensor setup strapped on to the subject

3-1-2 Data Acquisition

For training the neural network for hand held motions, a total of 9 minutes of stereo camera video, IMU data stream and motion capture ground truth motion of the three markers were acquired. The 9 minutes consisted of simple motions listed as follows

- Motions individually in the X, Y and Z axis.
- Planar motions.
- Pure rotations about pitch, roll and yaw axis.
- Combination of rotational and translational motions.

The nerual network was trained using approximately 6 minutes of data and about 3 minutes of data was used for testing. Similar procedure was followed for experiment with a human subject. The sensor setup was attached to a belt and tightened around the hip of the subject such that the setup was on the lower back and the cameras were facing backwards. This was done to ensure minimum occlusions due to the swinging of the arms. About 5 minutes minutes of data consisting of the human subject walking forwards and backwards, and performing forward lunges were obtained.

3-1-3 Data Analysis

The performance of the $\boldsymbol{Neural-EKF}$ was tested primarily by comparing it with the ground truth data obtained from the Qualisys Motion Capture system. For evaluation purposes, the Root Mean Squared Error (RMSE), coefficient of determination R^2 and the 3σ bounds were used. The definition of these quantities are as follows. Let the error between estimated the Z component of translational velocity and ground truth at time instant k be e_k. Let there N instances for which estimation is performed. Let the ground truth estimates for the Z component of the translational velocity be represented by $y_k, k = 1..N$. Let the mean of the ground truth estimate be $\bar{y}$. Let the standard deviation of the errors $e_k, k = 1..N$ be σ_e. Then $\pm 3\sigma_e$ determines the value of the 3σ bounds. If the magnitude of errors fall within this bound then the estimate is considered a good and reliable estimate. RMSE and R^2 measures are defined as follows;

$$\begin{aligned} \text{RMSE} &= \sqrt{\frac{1}{N}\Sigma_{k=1}^{N} e_k^2} \\ R^2 &= 1 - \frac{\Sigma_{k=1}^{N} e_k^2}{\Sigma_{k=1}^{N}(y_k - \bar{y})^2} \end{aligned} \tag{3-1}$$

Lower the RMSE better the quality of the estimate. Closer the value of R^2 is to 1, higher is the quality of the estimate. With these statistical analysis tools we will look into the results obtained from hand held camera motions in the next section.

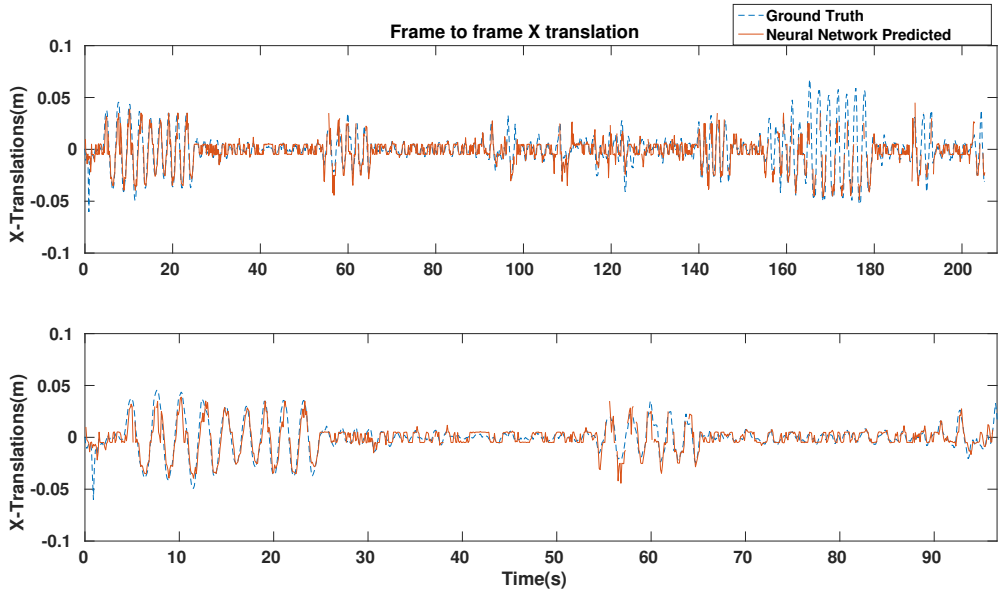


Figure 3-3: Comparison of ground truth x-axis translations with x-axis translations estimated by the neural network. The top figure shows the results for the full of the test data while the bottom figure zooms in on part of the data.

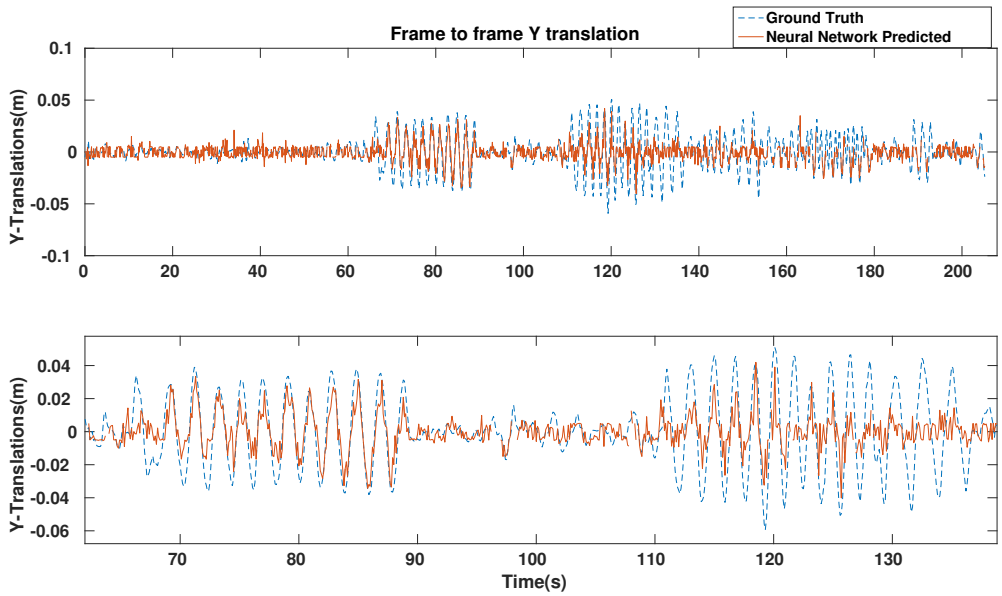


Figure 3-4: Comparison of ground truth y-axis translations with y-axis translations estimated by the neural network. The top figure shows the results for the complete test dataset while the bottom figure zooms in on part of the data.

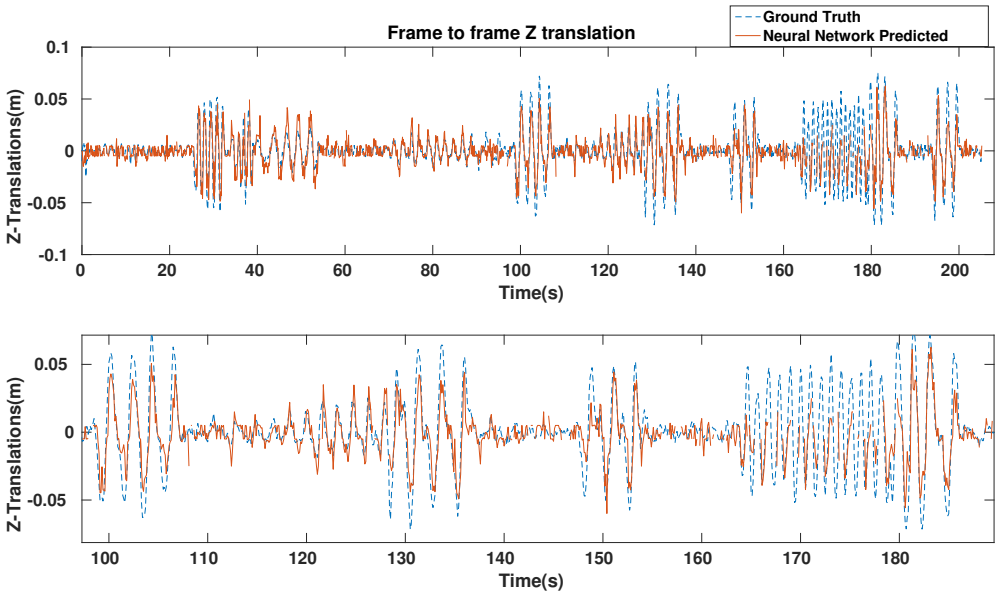


Figure 3-5: Comparison of ground truth z-axis translations with z-axis translations estimated by the neural network. The top figure shows the results for the complete test dataset while the bottom figure zooms in on part of the data.

3-2 Performance of *Neural-EKF* for Hand-Held Camera Motions

3-2-1 Results of Neural Network Estimation

The results of the predictions of translations by the neural network can be seen in figures 3-3, 3-4 and 3-5. For each frame to frame motion, if N features were successfully matched and triangulated, the neural network gives N possible translations of the camera with respect to the world frame. These N translations are boxed and a weighted average is taken as the final prediction of the neural network for comparison with ground truth.

3-2-2 Discussion

The bottom subplots in each of the figures 3-3, 3-4 and 3-5 zoom in on parts of the data where the prediction was found to be more accurate. From the zoom subplots, it can be clearly observed that the neural network performs very well with slow translations but not as well when it comes to fast translations. This can be attributed to the rate of processing of the neural network. This neural network estimates motion between two frames separated by 0.1 seconds. Choosing a higher processing frequency lead to very small frame to frame motions which were found to undistinguishable by the neural network. Hence this prevents the neural network from reacting to faster motions. This problem is solved to some extent in the later stages when fusing the neural network prediction with the EKF, where a the processing of the neural network is done every time a stereo images are acquired and the translational velocities for correction are obtained using simple interpolation as described in the previous chapter. Another factor that needs to be taken into consideration is that for

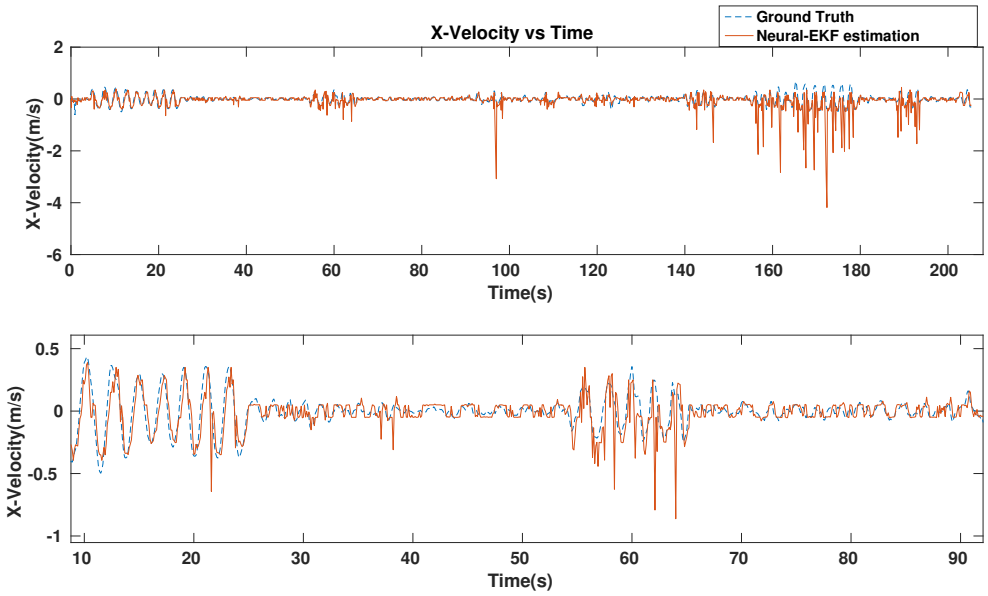


Figure 3-6: Comparison of ground truth x-axis velocities with x-velocities estimated by Neural-EKF.The top figure shows the results for the complete test dataset while the bottom figure zooms in on part of the data.

very fast movements, sometimes the image processing module gives no data due to heavy blurring and lack of features detected. Large blurs also reduce accuracy of feature detection and matching and this seems to affect the prediction by the neural network.

3-2-3 Results of *Neural-EKF* Estimation

Here, the estimation of translation velocities for hand held camera motions using the ***Neural−EKF*** will be discussed in detail. Translation velocities are important for the purposes of fall detection and they are not accurately obtainable from just the IMU. The absolute positions of the setup are not as important to this thesis as the velocity but nevertheless it will be discussed briefly as well.

The comparison of the 3 axis translation velocities with ground truth obtained from the motion capture can be seen in figure 3-6, 3-7 and 3-8.

3-2-4 Discussion

Just as observed in figures 3-3 , 3-4 and 3-5, the zoomed in subplots of figures 3-6, 3-7 and 3-8, clearly show that velocity estimation is quite good for slowly changing velocities and motions which consist of small accelerations. In case of very fast motions, the estimated velocity is observed to drift to a great extent. The same is also observed when there are no correction updates from the neural networks. The RMSE of translation velocity was found to be as follows;

- X-axis RMSE velocity=0.1278m/s.

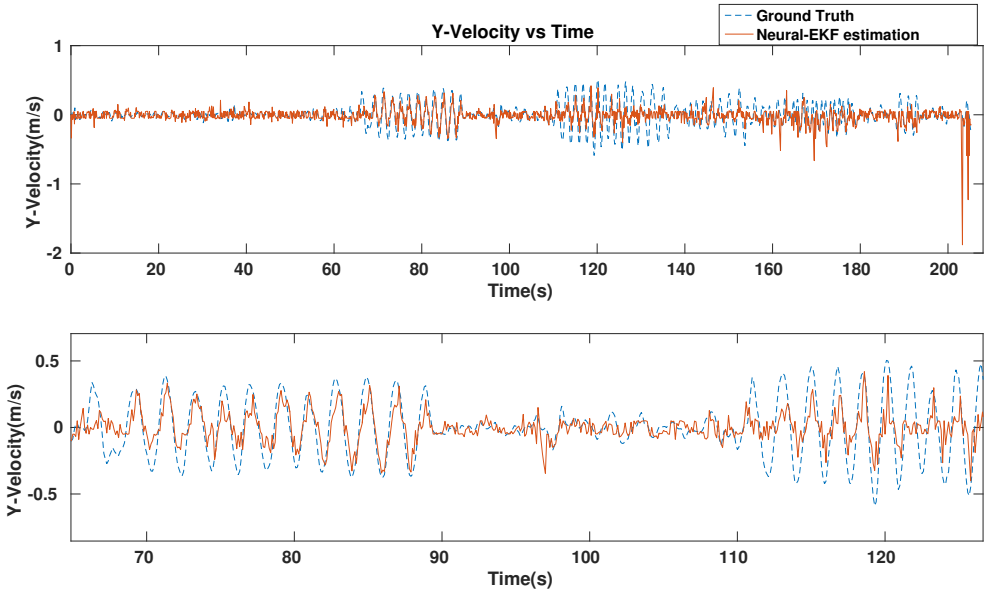


Figure 3-7: Comparison of ground truth y-axis velocities with y-velocities estimated by Neural-EKF.The top figure shows the results for the complete test dataset while the bottom figure zooms in on part of the data.

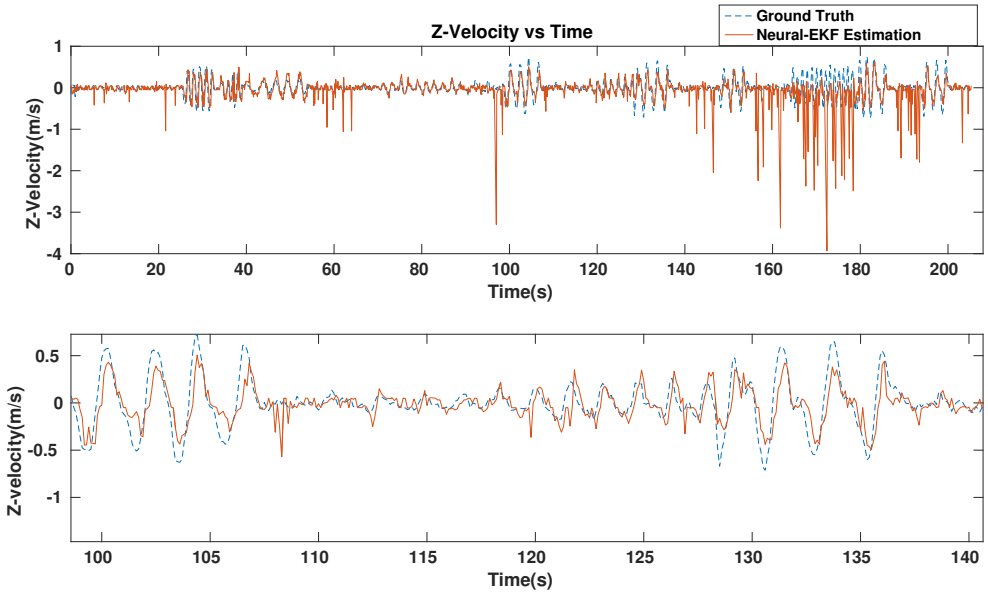


Figure 3-8: Comparison of ground truth z-axis velocities with z-velocities estimated by Neural-EKF.The top figure shows the results for the complete test dataset while the bottom figure zooms in on part of the data.

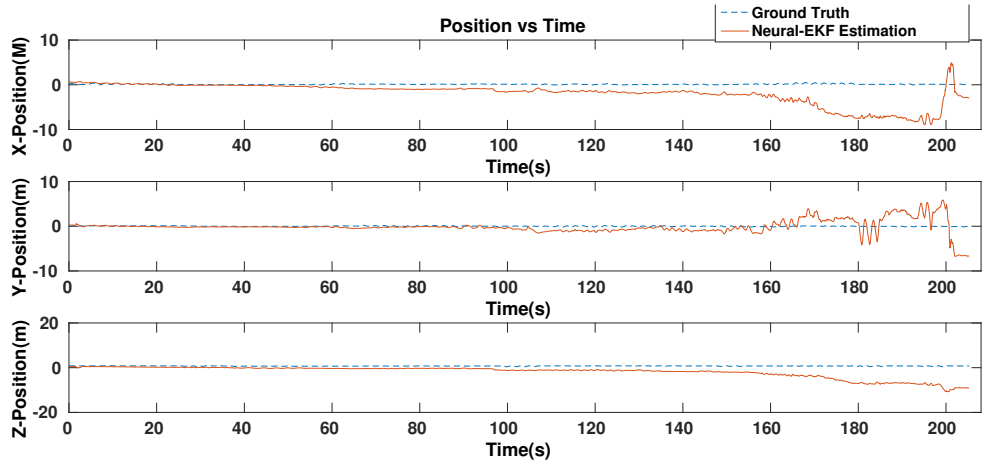


Figure 3-9: Comparison of ground truth positions with positions estimated by Neural-EKF for the 3 axes

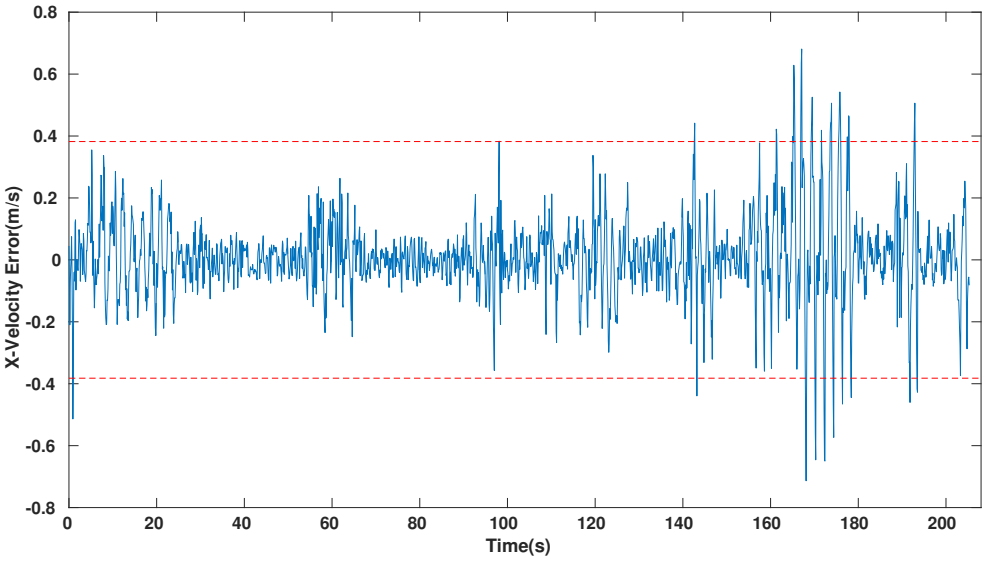


Figure 3-10: Error plot of x-axis velocity and 3σ bounds. The 3σ bounds are represented by the red dashed lines

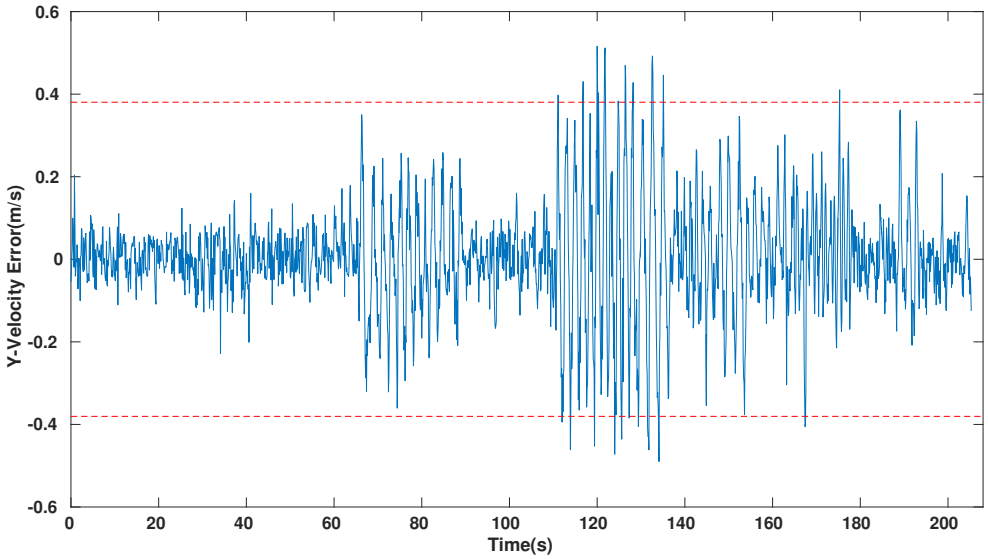


Figure 3-11: Error plot of y-axis velocity and 3σ bounds. The 3σ bounds are represented by the red dashed lines

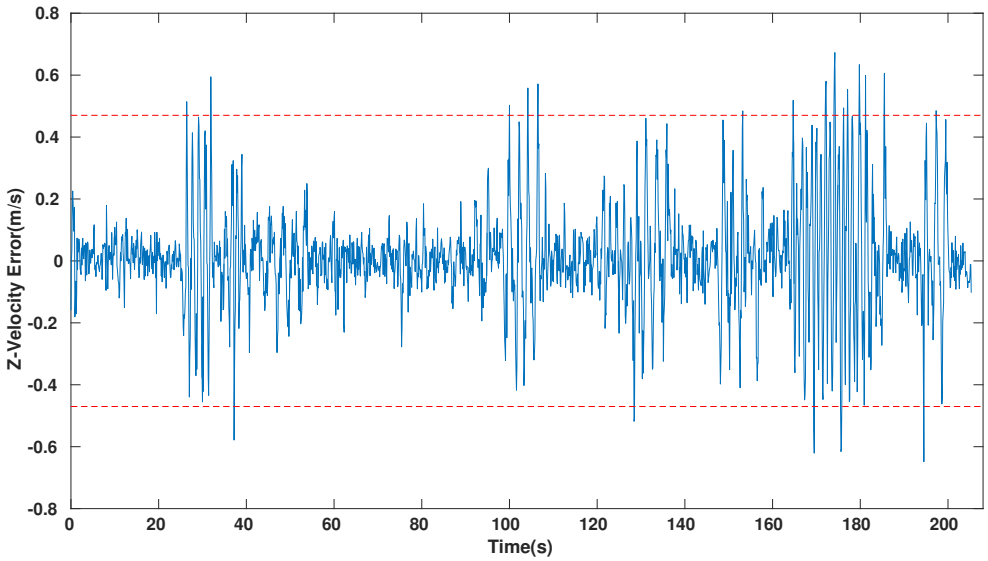


Figure 3-12: Error plot of z-axis velocity and 3σ bounds. The 3σ bounds are represented by the red dashed lines

- Y-axis RMSE velocity=0.1268m/s.
- Z-axis RMSE velocity=0.1571m/s.

It is also important to note that a maximum error of 3.678 m/s is observed in the estimation. As we require velocity to be estimated with very high accuracy, such high maximum error is not acceptable. These errors in velocity also lead to worsening of the position drifts caused by the un-observability of position. This drift can be clearly observed in the figure 3-9. As can be seen, the Z-position drifts the most from the ground truth with the maximum error being 11.56m. Although as mentioned earlier, absolute positions are not our major concerns and we will be focussing on translational velocities. In figures 3-10, 3-11 and 3-12 show the plots of the velocity errors and the 3σ bounds where σ is the standard deviation in the error. From these figures it is clear that the bounds are not met. These results are not of the required accuracy and they clearly demonstrate the limitations of the proposed algorithm.

From the results obtained we can finally conclude that the current algorithm is capable of tracking velocities of slow motions accurately and this is encouraging enough to test the algorithm for motions of a test subject. It will be interesting to see how this combination of neural network and extended Kalman filter performs for normal motions of humans. If good velocity tracking is achieved, then it should be possible to perform the necessary motion analysis on the estimates to detect the start of the fall. With this in mind, the next section will focus on experiments conducted with sensor setup attached to a human subject.

3-3 Performance Analysis of Neural-EKF with Human Motion Data

3-3-1 Data acquisition

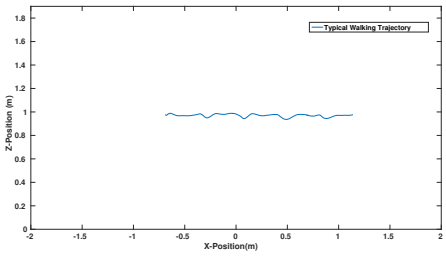
For analysis on actual human motions, a neural network was trained based on on the following motions;

- Slow and medium paced walking.
- vertical lowering of the back and bending of knee.
- Forward step and flexing of knee to bend down.
- Horizontal side steps.

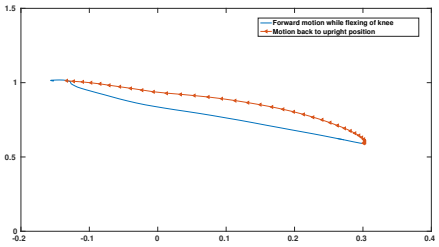
About 5 minutes of video, IMU data and motion capture data were used for training a new neural network to track human motions. This neural network was then tested on a different dataset consisting of the following motions.

- Slow and medium paced walking with more emphasis on stamping the foot on the ground.
- Lowering the hip completely to the floor and then getting up from the sitting down position.

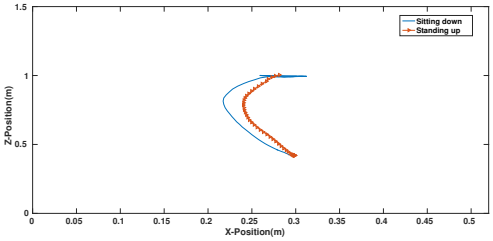
As before the X-axis represents the axis facing in the forward direction with respect to the test subject and Z axis represents the axis facing upwards from the ground. Typical trajectories of the training an test data in the $X-Y$ plane can be seen in the figure 3-14.



(a) Typical $x-z$ plane walking trajectory data used for training.



(b) Typical $x-z$ plane trajectory used for training the neural network. The motion being performed here is slight flexing of the knees while lunging forward.



(c) Typical $x-z$ plane trajectory used for testing the neural network. The motion being performed here complete squat to sit on the floor and then standing up.

Figure 3-14: Typical trajectories used for training and testing the neural network for performance on tracking human motions

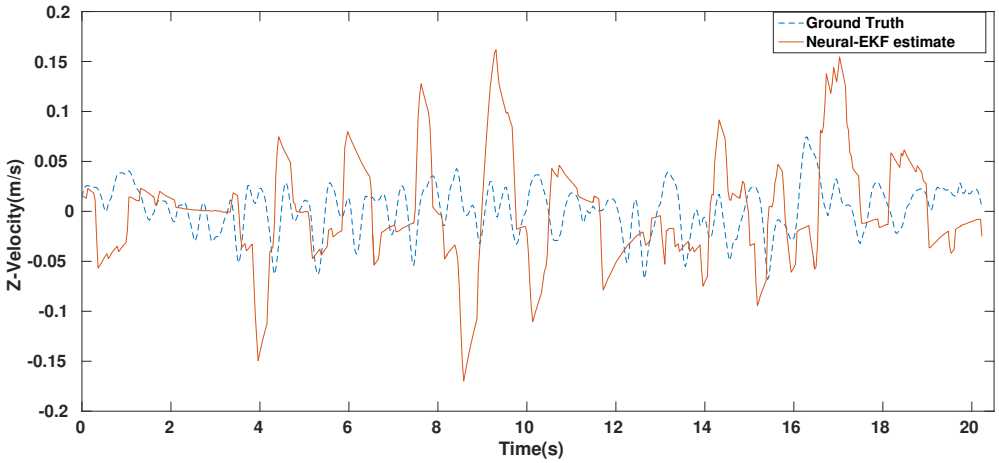


Figure 3-15: Plot comparing the ground truth Z axis velocity with the $\boldsymbol{Neural-EKF}$ estimate while walking.

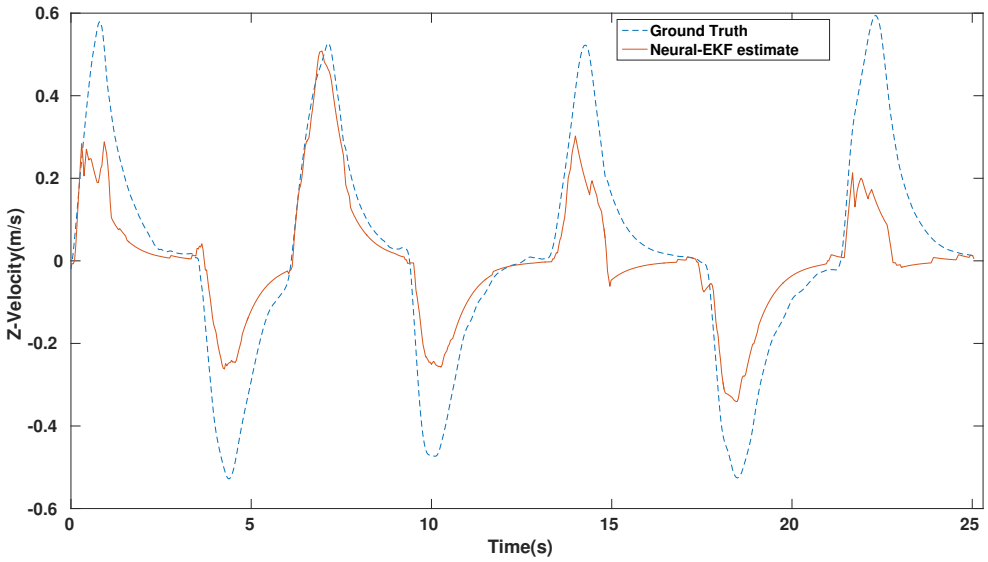


Figure 3-16: Plot comparing the ground truth Z axis velocity with the Neural-EKF estimate while performing the sitting down and getting up motions repeatedly.

Table 3-1: Results of translational velocity estimation by $\boldsymbol{Neural-EKF}$. The magnitude of velocity in the $X-Y$ plane and the Z-velocity are studied.

Activity	Velocity component	RMSE	Max Error	R^2
Walking	$X-Y$ plane	0.1625	0.3755	0.2792
	Z-axis	0.0595	0.1704	-5.152
Sitting Down/Standing up	$X-Y$ plane	0.2319	0.0768	0.1901
	Z-axis	0.1416	0.0.4533	0.7040

3-3-2 Results of *Neural-EKF* with Human Motions

For testing the performance of $\boldsymbol{Neural-EKF}$ with human motions, we are mostly concerned with Z axis velocity tracking and hence the results and analysis shall be done based on Z axis velocity estimates. The figure 3-15 compares the estimate of the $\boldsymbol{Neural-EKF}$ with the ground truth values obtained for a walking experiment. The figure 3-16 compares the ground truth data with Neural-EKF estimates for the case when a complete sitting down and standing up motion was performed.

The table 3-1 lists the various accuracy measures obtained from the estimates of $\boldsymbol{Neural-EKF}$. The velocity estimation accuracy errors are obtained for the two sets of distinct activities of walking and sitting down and standing up.

3-3-3 Discussion

While walking, the hip and the sensor setup attached to the hip, do very small Z axis translations in small repetitive periods. This causes high frequency motion which is not effectively tracked by the $\boldsymbol{Neural-EKF}$ as can be seen in the figure 3-15. Whereas this is

not the case when large motions like sitting down and standing up motions are involved. As can be seen from figure 3-16, the setup responds to every sit down and stand up motions, but the accuracy of the magnitude of velocity is not up to the acceptable standards. Furthermore, as can be seen for the table 3-1, very low and negative R^2 values for walking are obtained, in spite of low RMSE and maximum error values for both components of velocities. Whereas for the sitting down and standing up case, the R^2 values are significantly higher especially for the Z-velocities indicating good velocity tracking for activities involving slow vertical motions.

In spite of low accuracy for walking, we could possibly threshold the Z axis velocities within $\pm 0.2 m/s$ and say that if the velocity oscillates within this threshold, walking is being performed. In case of sitting down/standing up case, we could use overall spatial and temporal characteristics of the curve to determine if the person if performing the sitting down or standing up motion. This is a hypothesis which will be tested in future works once better accuracy is obtained from tracking.

Chapter 4

Conclusions

4-1 Overall Summary

In this thesis, a sensor data fusion algorithm combining a neural network and an EKF was implemented. The goal was to study how accurate the results can be when giving up on costly but accurate optimization/iteration based methods of visual-inertial motion estimation. There is very little data on performance of visual-inertial sensor data fusion algorithm for individual human motions/activity in the literature. Most of the algorithms measure errors in term of loop closure and accuracy of pose estimates. This thesis has attempted to contribute results on performance for two different human activities namely, motions of walking and motions of sitting down and standing up.

A neural network was trained in order to track the motion of the hip while walking, sitting down and standing up. The $\boldsymbol{Neural-EKF}$ provides an confidence or R^2 value of 0.7040 for the sitting down and standing up activity alone. For the walking activity the accuracy was very low and the R^2 value was found to be negative. From this we can conclude that training a neural network to track slow, very low frequency motions is a feasible idea but the inputs and training of the neural network need to be changed to improve the accuracy of the estimation of the $\boldsymbol{Neural-EKF}$. In spite of the low accuracy while tracking Z velocities, a threshold of ± 0.2m/s could be applied and if the Z value oscillated between this threshold, then we could say that the person was performing the walking motion. The estimation results for the sitting down and standing up activities is very encouraging as we observe that the $\boldsymbol{Neural-EKF}$ is capable of reacting to these predominantly vertical axis motions. With slightly more accurate estimates and a supervised classifier which learns the velocity curves for these activities, it would be possible to distinguish whether the person is in control of his motion or whether his motion will ultimately lead to a fall. We know where our algorithm fails. We have identified where the algorithm performs the best. If we exploit these facts, learn the shapes of the velocity curves for various activities, there is potential for this algorithm to be used in early fall detection.

4-2 Limitations of the *Neural-EKF* and Suggestions for Future Work

The main limitation faced by the $\boldsymbol{Neural-EKF}$ is the necessity of presence of good quality features in the image scenes. Without these features, the tracking algorithm will cease to function. To realise the accuracy of velocity estimation we obtained in this thesis, at least 5 features per frames were necessary. To solve this limitation, one could possibly use some form of dense 2D image to 3D mapping techniques like the Large Scale Dense(LSD) SLAM presented in [28]. The dense mapping can be performed on some sparse regions of the image alone to obtain 3D points and motion of 3D points between two frames.

The second limitation is the lack of any explicit outlier rejection of feature points. The outlier rejection is done to some extent when using the weighted values of translations obtained from the neural network but even so, in the presence of low number of feature match is enough to reduce the accuracy of the estimate. Hence, there must a simple technique which pre-filters the inputs sent to the neural network, or rejects the estimate from the neural network if the standard deviation of the translations obtained is higher than a threshold. In such a case where standard deviation of the translations obtained are high, we can say that the noisy feature matches and motions are present in the inputs sent to the neural network, and hence correction need not be performed in this cycle. This may aid in improving the accuracy.

The inputs which were chosen for this neural network, namely frame-to-frame 3D feature coordinate difference, pixel motion and the orientation transform quaternion between the two time frames, gave encouraging results. Nevertheless, to increase the accuracy of estimation further, better parameters or variables need to obtained from the image data to estimate motion.

In this thesis we studied human motion estimation for a single subject. To improve the motion estimation capabilities of the neural network, it needs to be trained with data from more subjects. Also, what needs to be tested is the performance when there is much higher impact on sitting down or when the subject tied to a harness is pushed to fall down.

As mentioned earlier a classifier capable of leaning various velocity curves for different activities on top of the $\boldsymbol{Neural-EKF}$ framework has a potential to to detect falls early. Considering that the neural network has a constant and low complexity of execution, it would be possible to use a slightly more computationally expensive learning algorithm and still obtain real time performance.

The methodology of the $\boldsymbol{Neural-EKF}$ is just a base on which a lot more can be built. With right set of inputs and target outputs, in the near future, it might be possible to estimate motions with very accuracy without the necessity of very deep learning algorithms. The $\boldsymbol{Neural-EKF}$ is a suggestion towards the possibility of using neural networks in multiple low-level capacities to ultimately lead to the development of a successful fall prevention and health monitoring device.

Appendix A

Basics of Kalman Filtering

The key objective of this thesis is to obtain motion parameters like, position, velocity and orientation, but all we get from the IMU and accelerations and rotational velocities, while we get images and points of interest or features from cameras. If we are to obtain any of the motion parameters mentioned, we have to fit a model that maps the sensor data to the motion parameters or states. In control system terms we need an observer of the motion of the person to estimate the various states. In robotics, various Kalman filters/observers have been extensively used for state estimation. The Extended Kalman Filter (EKF) has been used especially for fusing data from various sensors. As both visual odometry and sensor fusion algorithms use the Kalman filter, the formulations of the linear Kalman filter and the EKF are discussed in following sections.

Before going into the Kalman filters it is necessary to discuss Gaussian random variables on which a major part of the formulation of the Kalman filter depends. Consider a random variable X with mean μ and variance σ^2 represented as $x \sim N(\mu, \sigma^2)$. If the variable follows what is known a Gaussian or normal distribution, its Probability Density Function (PDF) is defined as follows

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{A-1}$$

Any linear process on a Gaussian random variable is considered a random process. If we have two independent random variables $x_1 \sim N(\mu_1\sigma_1)$, and $x_2 \sim N(\mu_2, \sigma_2)$ they have the following properties.

$$x_1 + x_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$
$$p(x_2 + x_2) = \frac{1}{\sqrt{2\pi(\sigma_1^2 + \sigma_2^2)}} e^{-\frac{(x-(\mu_1+\mu_2)^2)}{2(\sigma_1^2+\sigma_2^2)}} \tag{A-2}$$

If x is a multi-dimensional random variable then it will be represented as $x \sim N(\mu_x, \Sigma_{xx})$ where μ_x is the mean and Σ_{xx} is known as the covariance matrix of the variable x. Consider

the following;

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim N(\mu, \Sigma) = N\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \tag{A-3}$$

where Σ_{xy} represents the covariance between the variable x and y.

Another important property is the intersection. If have $x \sim N(\mu_x, \Sigma_{xx})$ and $y = Ax+b$ where A is constant and $b \sim N(0, Q)$ where Q is the covariance matrix of the b. Typically the b term is used to represent the assumption that the sensor noise is zero mean white Gaussian noise, an assumption widely used in robotics and in particular with the IMU. We have the following property then;

$$\begin{aligned} p\left(\begin{bmatrix} x \\ y \end{bmatrix} \right) &= N\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \\ &= N\left(\begin{bmatrix} \mu_x \\ A\mu_x + b \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xx}A^T \\ A\Sigma_{xx} & A\Sigma_{xx}A^T + Q \end{bmatrix} \right) \end{aligned} \tag{A-4}$$

A-1 Linear Kalman Filter

Consider a linear system with the following set of state equations

$$\begin{aligned} \boldsymbol{x_{k+1}} &= \mathbf{A}\boldsymbol{x_k} + \mathbf{B}u_k + \boldsymbol{w_k} \\ \boldsymbol{y_k} &= \mathbf{C}\boldsymbol{x_k} + \boldsymbol{v_k} \end{aligned} \tag{A-5}$$

where $\boldsymbol{x_k}$ is the state at time instance k, $\boldsymbol{w_k}$ is the normal variable with $\boldsymbol{w} \sim N(0, \mathbf{Q})$ known as process noise and $\boldsymbol{v}$ is the normal variable $\boldsymbol{v} \sim N(0, \mathbf{R})$ known as the measurement noise. $\mathbf{A}$ is the system matrix and $\mathbf{B}$ is input matrix. $\boldsymbol{u_k}$ is the input at time k.

Suppose we have a state estimator/predictor as follows;

$$\hat{\boldsymbol{x}}_{\boldsymbol{k+1|k}} = \mathbf{A}\hat{\boldsymbol{x}}_{\boldsymbol{k|k}} + \mathbf{B}\boldsymbol{u_k} \tag{A-6}$$

The predicted output is;

$$\hat{\boldsymbol{y}}_{\boldsymbol{k+1}} = \mathbf{C}\hat{\mathbf{x}}_{\mathbf{k+1|k}} \tag{A-7}$$

Then the error is the estimate and the covariance of the error can be derived as follows;

$$\begin{aligned} \tilde{\boldsymbol{x}}_{\boldsymbol{k|k}} &= \boldsymbol{x_k} - \hat{\boldsymbol{x}}_{\boldsymbol{k|k}} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{k+1|k}} &= \boldsymbol{x_{k+1}} - \hat{\boldsymbol{x}}_{\boldsymbol{k+1|k}} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{k+1|k}} &= \mathbf{A}\tilde{\boldsymbol{x}}_{\boldsymbol{k|k}} + \boldsymbol{w_k} \\ \boldsymbol{\Sigma}_{\mathbf{k|k}} &= E[\tilde{\boldsymbol{x}}_{\boldsymbol{k|k}}\tilde{\boldsymbol{x}}^{\boldsymbol{T}}_{\boldsymbol{k|k}}] \\ \boldsymbol{\Sigma}_{\mathbf{k+1|k}} &= E[\tilde{\boldsymbol{x}}_{\boldsymbol{k+1|k}}\tilde{\boldsymbol{x}}^{\boldsymbol{T}}_{\boldsymbol{k+1|k}}] \\ \boldsymbol{\Sigma}_{\mathbf{k+1|k}} &= \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{k|k}}\mathbf{A}^{\mathbf{T}} + \mathbf{Q} \end{aligned} \tag{A-8}$$

where $\boldsymbol{\Sigma}_{\mathbf{k+1|k}}$ is the error covariance for a time instant $k+1$ given corrected estimates till time instant k. $E[]$ is the expectation function.

To minimize the error in the estimate, a Kalman gain is calculated using least squares so as to minimize the error covariance matrix. This Kalman gain $\boldsymbol{K}$ along with the error $\boldsymbol{y_{k+1} - \hat{y_{k+1}}}$ can be used to update the predicted state and the error covariance so as to reduce the prediction error. The equations are as follows.

$$\begin{aligned} \boldsymbol{K_{k+1}} &= \mathbf{\Sigma_{k+1|k}} \mathbf{C^T} (\mathbf{C \Sigma_{k+1|k}} \mathbf{C^T} + \mathbf{R})^{-1} \\ \boldsymbol{\hat{x}_{k+1|k+1}} &= \boldsymbol{\hat{x}_{k+1|k+1}} + \boldsymbol{K_{k+1}} (\boldsymbol{y_{k+1}} - \mathbf{C} \boldsymbol{\hat{x}_{k+1|k}}) \\ \mathbf{\Sigma_{k+1|k+1}} &= \mathbf{\Sigma_{k+1|k}} - \boldsymbol{K_{k+1}} \mathbf{C \Sigma_{k+1|k}} \end{aligned} \tag{A-9}$$

A-2 Extended Kalman Filter

In many cases where motion has to be estimated the process is usually non-linear and using a linear Kalman filter will not yield productive results. So the extended Kalman filter uses a non linear model for the process, linearises the non linear model at the particular time instance of update and follows the process for Kalman filtering on the model linearized around a particular time instant. The following are the non linear process equations.

$$\begin{aligned} \boldsymbol{x_{k+1}} &= f(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{w_k}) \\ \boldsymbol{z_k} &= h(\boldsymbol{x_k}, \boldsymbol{v_k}) \end{aligned} \tag{A-10}$$

where f and h are non linear models and $\boldsymbol{w_k}, \boldsymbol{v_k}$ are the process and measurement noises respectively. To linearise the non linear function the Taylor series expansion is used up to the first order and the prediction step is derived as follows

$$\begin{aligned} f(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{w_k}) &\simeq f(\boldsymbol{\hat{x}_k u_k}) + f'(\boldsymbol{\hat{x}_t u_k})(\boldsymbol{x_k} - \boldsymbol{\hat{x}_k}) ... \\ f(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{w_k}) &\simeq f(\boldsymbol{\hat{x}_k u_k}) + \mathbf{F}(\boldsymbol{\hat{x}_t}, \boldsymbol{u_t}) \boldsymbol{\tilde{x}_k} \end{aligned} \tag{A-11}$$

where $\mathbf{F}(\hat{x}_k, u_k)$ is the Jacobian of the f w.r.t to the states, $\boldsymbol{\tilde{x}_k}$ is the error in estimate of the states,$\boldsymbol{(x_k - \hat{x}_k)}$. $\mathbf{F}(\boldsymbol{\hat{x}_k}, \boldsymbol{u_k})$ will be written as $\mathbf{F_k}$ for brevity.

Using the above equation the error in the estimate of the state can be propagated as follows

$$\boldsymbol{\tilde{x}_{k+1}} = \mathbf{F_k} \boldsymbol{\tilde{x}_k} + \mathbf{G_k w_k} \tag{A-12}$$

where $\mathbf{G_k}$ is the Jacobian of $f(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{w_k})$ with respect to noise.

The error covariance matrix can now be derived as follows

$$\begin{aligned} \mathbf{\Sigma_{k+1}} &= E[\boldsymbol{\tilde{x}_{k+1}} \boldsymbol{\tilde{x}_{k+1}^T}] \\ &= E[\mathbf{F_k} \boldsymbol{\tilde{x}_k} \boldsymbol{\tilde{x}_k^T} \mathbf{F_k^T}] + E[\mathbf{G_k} \boldsymbol{w_k} \boldsymbol{w_k^T} \mathbf{G_k^T}] \\ &= \mathbf{F_k} E[\boldsymbol{\tilde{x}_k} \boldsymbol{\tilde{x}_k^T}] \mathbf{F^T} + \mathbf{G_k} E[\boldsymbol{w_k} \boldsymbol{w_k^T}] \mathbf{G^T} \\ &= \mathbf{F_k \Sigma_k F_k^T} + \mathbf{G_k Q_k G_k^T} \end{aligned} \tag{A-13}$$

It is to be noted that for digital systems which almost always the case, it is necessary to obtain the estimates and errors using numerical integration.

For the update of estimated state we use the Taylor expansion of the measurement equation as follows;

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x_k}) &= \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) + \mathbf{H_k}(\boldsymbol{x} - \hat{\boldsymbol{x}}_{\boldsymbol{k}}) + .. \\ \boldsymbol{h}(\boldsymbol{x_k}) &\simeq \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) + \mathbf{H_k}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} \end{aligned} \tag{A-14}$$

where $\mathbf{H_k}$ is the Jacobian of h w.r.t the states The error in actual measurement and the hypothesis, also known as residual is used to update the estimate, just like the normal Kalman filter using the following equations.

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x_k}) - \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) &= \mathbf{H_k}\tilde{\boldsymbol{x}} - \boldsymbol{v_k} \\ \boldsymbol{z_k} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) &= \mathbf{H_k}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} - \boldsymbol{v_k} \\ \boldsymbol{r_k} &= \mathbf{H_k}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{n_k} \end{aligned} \tag{A-15}$$

where $\boldsymbol{r_k}$ is the residual vector and $\boldsymbol{n_k}$ is the noise.

The state and covariance update equations are as follows

$$\begin{aligned} \boldsymbol{x}_{\boldsymbol{k|k}} &= \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} + \boldsymbol{K_k}(\boldsymbol{z_k} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}})) \\ \mathbf{\Sigma_{k|k}} &= \mathbf{\Sigma_{k|k-1}} - \boldsymbol{K_k}\mathbf{H_k}\mathbf{\Sigma_{k|k-1}} \\ \boldsymbol{K_k} &= \mathbf{\Sigma_{k|k-1}}\mathbf{H_k^T}(\mathbf{H_k}\mathbf{\Sigma_{k|k-1}}\mathbf{H^T} + \mathbf{R_k})^{-1} \end{aligned} \tag{A-16}$$

The full derivations of both the linear Kalman filter and EKF can be found in many online resources, for example in [29] and [30].

Bibliography

[1] "Fitbit." http://www.fitbit.com.

[2] "Jawbone." http://jawbone.com.

[3] "Facts sheet on falls." http://www.who.int/mediacentre/factsheets/fs344/en/.

[4] "Who global report on falls prevention in older age," 2007.

[5] "Human body outline printable." http://cliparts.co/human-body-outline-printable.

[6] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls," *PloS one*, vol. 7, no. 5, p. e37062, 2012.

[7] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait & posture*, vol. 28, no. 2, pp. 285–291, 2008.

[8] S. Z. Erdogan and T. T. Bilgin, "A data mining approach for fall detection by using k-nearest neighbour algorithm on wireless sensor network data," *Communications, IET*, vol. 6, no. 18, pp. 3281–3287, 2012.

[9] H. Kerdegari, K. Samsudin, A. R. Ramli, and S. Mokaram, "Evaluation of fall detection classification approaches," in *Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on*, vol. 1, pp. 131–136, IEEE, 2012.

[10] M. St-Pierre and D. Gingras, "Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 831–835, IEEE, 2004.

[11] J. H. Kotecha and P. M. Djuric, "Gaussian particle filtering," *IEEE Transactions on signal processing*, vol. 51, no. 10, pp. 2592–2601, 2003.

[12] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 298–304, IEEE, 2015.

[13] "Robot pose ekf." wiki.ros.org/robot_pose_ekf.

[14] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3923–3929, IEEE, 2013.

[15] K. Tsotsos, A. Pretto, and S. Soatto, "Visual-inertial ego-motion estimation for humanoid platforms," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 704–711, IEEE, 2012.

[16] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual–inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[17] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4531–4537, IEEE, 2011.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[20] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, p. 50, Citeseer, 1988.

[21] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International conference on computer vision*, pp. 2548–2555, IEEE, 2011.

[22] Y. Jiang, Y. Xu, and Y. Liu, "Performance evaluation of feature detection and matching in stereo visual odometry," *Neurocomputing*, vol. 120, pp. 380–390, 2013.

[23] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–652, IEEE, 2004.

[24] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 957–964, IEEE, 2012.

[25] MATLAB, *MATLAB and Computer Vision Toolbox release R2015b*. Natick, Massachusetts: The MathWorks Inc., 2015.

[26] MATLAB, *MATLAB and Neural Network Toolbox release R2015b*. Natick, Massachusetts: The MathWorks Inc., 2015.

[27] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, IEEE, 2011.

[28] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[29] R. Faragher *et al.*, "Understanding the basis of the kalman filter via a simple and intuitive derivation," 2012.

[30] G. A. Terejanu, "Extended kalman filter tutorial,"

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
TU Delft	Delft University of Technology
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
ADL	Activities of Daily Living
RANSAC	Random Sampling and Consensus
RMSE	Root Mean Squared Error
FPS	Frames Per Second
KLT	Kanade Lucas Tomasi
PDF	Probability Density Function
SURF	Speeded Up Robust Features
BRISK	Binary Robust Invariant Scalable Keypoints