Fraunhofer IOSB

Ernst-Abbe-Hochschule Jena
University of Applied Sciences

# Analysis and prediction of high-resolution eye-tracking data

## Master Thesis Dissertation

Srikanth Venkatramani, (2021)
Matriculation No. 645243

**M.Sc Scientific Instrumentation**
**Ernst Abbe Hochschule, Jena, Germany**

**Academic Supervisor:**
Prof. Dr. rer. nat. Robert Brunner,
robert.brunner@eah-jena.de,
SciTec department,
Ernst Abbe Hochschule, Jena

**Internal Mentor:**
Dr. Alexey Pak,
alexey.pak@iosb.fraunhofer.de,
MRD department,
Fraunhofer IOSB, Karlsruhe

srikanth.v0610@gmail.com
+49-17636974394

# ACKNOWLEDGEMENTS

The past 6 months have presented a truly unique opportunity to gain immense knowledge and first hand experience at the prestigious Fraunhofer IOSB research institute. Thank you for allowing me to carry out my Master Thesis project at this esteemed organization.

I would like to express my deepest gratitude to Dr. Alexey Pak, my thesis mentor, who in spite of being extraordinarily busy with his duties, took time out to provide me with valuable input, guidance, encouragement and support over the past 7 months.

I would like to thank my thesis supervisor Prof. Dr. Rober Brunner for accepting to be my mentor and providing me with his valuable guidance.

To my family and friends, thank you for your continuous support, encouragement, patience and love over this time.

# NOTATION

| | |
|---|---|
| Deterministic Forecasting Approach: | |
| $x_k$ | the current eye position, where the indexing k represents the present discrete time sample number |
| $\hat{x}_k$ | the predicted eye position for the $k_{th}$ time step |
| $e_k$ | the prediction error for the $k_{th}$ time step |
| $t_k$ | the discrete time step such that $t_{k+1} = t_k + T$, where T is the sampling time |
| $n$ | the number of previous data points used to fit the model |
| $A^{-1}$ | the inverse of a matrix $A$ |
| $adjA$ | the adjoint of a matrix $A$ |
| Probabilistic Forecasting Approach: | |
| $X$ | the state matrix |
| $\hat{x}_{k\|k-1}$ | the system predicted state at sample $k$, using information only up to the $(k-1)_{th}$ measurement |
| $P_{k\|k-1}$ | the system predicted covariance at sample $k$, using information only up to the $(k-1)_{th}$ measurement |
| $z_k$ | the measured eye position for the $k_{th}$ time step |
| $\bar{z}_k$ | the measurement residual |
| $R_k$ | measurement noise covariance |
| $K_k$ | the Kalman filter gain |
| $\hat{x}_{k\|k}$ | the system estimate state at sample $k$, using information up to the $(k)_{th}$ measurement |
| $P_{k\|k}$ | the system estimate covariance at sample $k$, using information up to the $(k)_{th}$ measurement |
| $L_k$ | the likelihood function |
| $\mu_k$ | the model probability |
| $\pi_{ij}$ | the model transition probability matrix where $i$ and $j$ represents each model |
| $A^T$ | the transpose of a matrix $A$ |

# ABSTRACT

In Human visual system studies, it remains an unresolved debate on how the fixational eye movements, together with the neural activity they generate, help in the visual perception. In experimental studies of visual performance, the need often emerges to modify the stimulus according to the eye movements performed by the subject. Our project "Visual Scrambler" aims to develop a novel eye-movement-contingent display (EMCD) to communicate visual information to people exploiting the current knowledge about the human visual system, utilizing state-of-the-art equipment.

The response latency is the most important characteristic of an EMCD system. In order to reduce the perceived stimulation latency, we built a fast Field Programmable Gate Arrays (FPGA) platform to generate image transformations at the level of electronics. However, an important parameter for conducting our planned experiments would require an interactive stimulus generation with a delay of at most 5-6 ms between an eye position reading and the stimulus display. Therefore to achieve the goal parameters, we need to further reduce the system latency. This paper focuses on the development of eye movement prediction system, that could accurately predict the subjects eye positions in real-time for up to 10 ms in advance, thereby allowing the reduction in the effective latency. The prediction models, once integrated into the FPGA would be able to fill the latency gap, thereby generating faster gaze-contingent stimuli.

# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ii

**NOTATION** iii

**ABSTRACT** iv

**1 INTRODUCTION** 1

1.1 Human Visual System . . . 1

1.1.1 The Unsteady Eye . . . 1

1.1.2 Fixational Eye Movements . . . 2

1.2 Eye Movement Contingent Display (EMCD) . . . 5

1.3 State-of-the-art in EMCD . . . 8

1.3.1 Are Video-based Eye Trackers Accurate? . . . 8

1.3.2 A Modern EMCD Control: EyeRIS . . . 9

1.3.3 EMCD for Retinal Stabilization . . . 11

1.3.4 EMCD in Neuroscience . . . 11

1.4 The Visual Scrambler Project . . . 12

1.5 Problem Statement . . . 12

**2 METHODOLOGY** 14

2.1 Overview . . . 14

2.1.1 Visual Scrambler . . . 14

2.2 Model Constraints . . . 16

2.3 Algorithms . . . 16

2.3.1 Deterministic Forecasting Approach . . . 16

2.3.2 Probabilistic Forecasting Approach . . . 25

2.4 Quality Metrics . . . 54

2.4.1 Forecast Errors . . . 54

2.4.2 Percentage Accuracy (PA) . . . 55

**3 EXPERIMENTAL SETUP** 56

3.1 Survey . . . 56
3.1.1 Apparatus . . . 56
3.1.2 Participants . . . 56
3.1.3 Procedure and Task . . . 56
3.1.4 Pre-processing . . . 57
3.2 Test Environment . . . 57
3.3 Plots . . . 58

**4 RESULTS** **61**
4.1 Delta Coding Technique . . . 61
4.2 Linear Regression . . . 65
4.3 Polynomial Regression . . . 69
4.4 Kalman Constant Velocity Estimator . . . 73
4.5 Kalman Constant Acceleration Estimator . . . 77
4.6 Adaptive Filter . . . 82
4.7 Multiple Model Adaptive Estimator . . . 86
4.8 Interactive Multiple Model Estimator . . . 90
4.9 Model Comparison . . . 95
4.10 Discussion . . . 98

**5 CONCLUSIONS AND RECOMMENDATIONS** **99**
5.1 Restatement of Research Goal . . . 99
5.2 Solution . . . 99
5.3 Summary of Results . . . 99
5.4 Outlook . . . 100

**REFERENCES** **105**

# CHAPTER 1

# INTRODUCTION

## 1.1 HUMAN VISUAL SYSTEM

### 1.1.1 The Unsteady Eye

Since the early 1900s *eye movements*[1] have been measured and studied widely. Eye tracking technology has been successfully used to draw inferences about perception, cognition and brain function in numerous areas of psychology, cognitive science and applied research fields [1]. The *acuity*[2] throughout the visual field is varying, rapidly declining with increasing distance from the *foveola*[3]. As a result of this phenomenon, humans acquire visual information during brief periods of *fixations* separated by *saccades*, which helps in the inspection of an object of interest using the high-acuity region of the eye [2]. But what are saccades and fixations?

**Saccades:** As illustrated in Figure 1.1 saccades are rapid eye movements intended to change the portion of the visual field being fixated. In eye-tracking environments, saccades generally occur when a subject switches from one stimulus point to another.

**Fixations:** These are low-speed eye movements that occur in-between saccades, when a subject attempts to fix their eyes on a specific point in space. Refer Figure 1.1.

The term "fixation" is misleading as our eyes are continuously in motion even when we fix our gaze on an object. The fixational eye movements have a magnitude that is large enough to be visible to us but still we are unaware of their existence. Motion signals with such speeds would immediately be visible to us, had they originated from the objects in the scene rather than our own eyes [2]. So what are these inter-saccadic fixational eye movements and what is their purpose in visual perception?

[1] **Eye movement** includes the voluntary or involuntary movement of the eyes, helping in acquiring, fixating and tracking visual stimuli.

[2] **Visual acuity** refers to your ability to discern the shapes and details of the things you see.

[3] **Foveola** is located within a region called the macula, a yellowish, cone photoreceptor filled portion of the human retina.

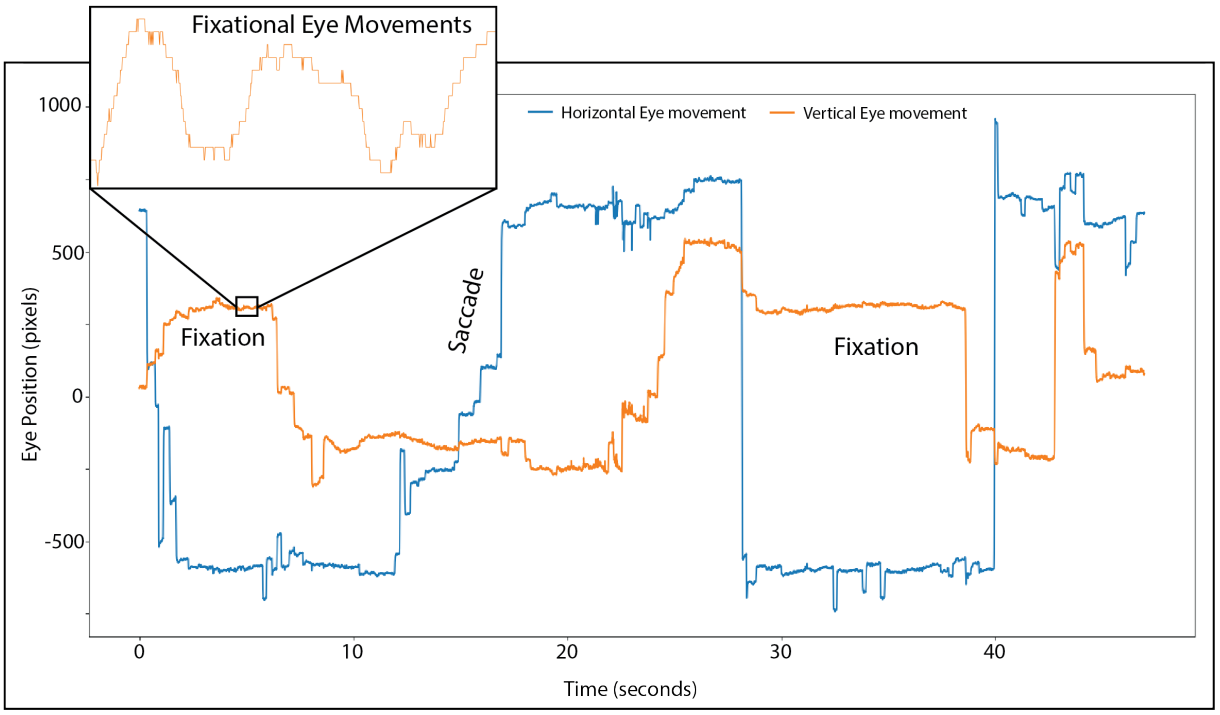


**Figure 1.1: Types of Eye Movements:** The figure shows an analysis of a typical eye tracking session, recorded at Fraunhofer IOSB with the TRACKPixx3 eye tracker. The horizontal and vertical eye motion are plotted with respect to time. The regions illustrating the different types of eye movements are characterized here.

### 1.1.2 Fixational Eye Movements

#### *1.1.2.1 Types of Fixational Eye Movements*

Scientists today agree on the occurrence of three main types of fixational eye movements: tremor, drifts and microsaccades [3]:

**Tremor:** These are high frequency (40-100 Hz), low amplitude (~8.5 sec) [4], periodic movements in the eyes which are usually in the range of the recording systems noise. Being the smallest eye movements, visual tremor is difficult to record accurately and their contribution for the maintenance of vision is unclear. It has been debated that due to the high frequency of the tremor movements, they might be ineffective as a stimulus since they lie above the *Flicker Fusion Frequencies*[4] in humans. However, recent studies have shown that early visual neurons can follow the high-frequency

[4]**Flicker Fusion Frequencies** is the rate of flicker at which the flickering stimulus being viewed appears non-flickering (approximately 50–60 Hz in humans).

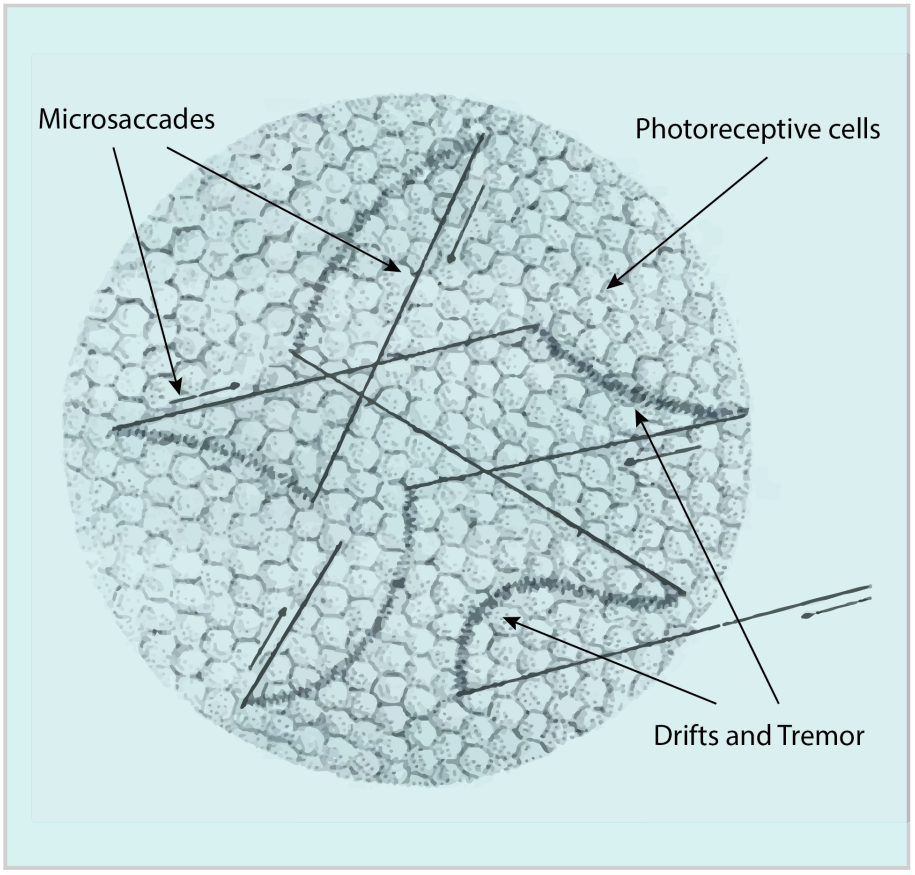


**Figure 1.2: Fixational Eye Movement:** A 0.05 mm diameter of the patch of the fovea is shown here. The curved lines show the High-frequency tremor, superimposed on slow drifts. The straight lines illustrates Microsaccades, the fast jerk-like movements, which generally bring the image back towards the centre of vision. ("The figure and the caption adopted from: [3]").

flickering and so it might be possible that tremor movements play a role in the visual perceptions [3].

**Drifts:** These are slow eye motions that occur simultaneously with tremor during the epochs between microsaccades (95-97% fixation time). During drifts, the image of the object being fixated can move across dozens of *photoreceptors*[5]. Initially, drifts seemed to be generated by the instability of the *oculomotor system*[6], due to their random motions. However, later studies have found out that in the absence of microsaccades, drift movements have a compensatory role in maintaining accurate visual fixation [3] [2]. Due to the small amplitude (3–12 min [5]) and the unavailability of

[5]**Photoreceptors** are the cells in the retina that respond to light.

[6]**Oculomotor system** consists of interconnected regions throughout the central nervous system that interact to control various eye movements.

a marked boundary between the drifts and tremor movements, characterizing them is challenging.

**Microsaccades:** These are small, fast, jerk-like involuntary eye movements that occur during voluntary fixations. Microsaccades are 20-30 ms in duration [6] and may carry the retinal image across a range of several dozens to several hundreds of photoreceptor widths. The role of microsaccades is still being argued upon. A possible role of microsaccades is to correct the eye movements caused by drifts as illustrated in the Figure 1.2. For instance, if drifts carry the fixated image away from the *fovea*[7], microsaccades will tend to bring it back. Recent studies have found that microsaccades might help to counteract receptor adaptation on a short time-scale and correct the fixational errors on longer time scales. Further studies on the velocity and amplitude shows similarity between the microsaccades and larger saccades. As a consequence, it has been proposed that the two motions are generated by the same mechanism [3].

#### *1.1.2.2 Role of Fixational Eye Movements*

The function of microsaccades during visual fixation has been the subject of debate for more than thirty years:

- *Retinal stabilization*[8] experiments have shown that elimination of the retinal image motion causes the image to fade away [7]. Fixational eye movements are commonly thought to refresh neural responses in order to prevent the image from fading away [2].
- Cornsweet originally proposed that the purpose of microsaccades is to act as a correction to the drift movements and thereby bringing the eyes to the fixation target [8].
- It was also postulated that microsaccades help in maintaining the vision by counteracting the retinal fatigue [9].
- Ditchburn and Foley-Fisher's research has shown that microsaccades were necessary to perceive hue difference at low contrast [10].
- Westheimer suggested that microsaccades enhance the perception of depth [11].
- Carpenter hypothesised that amongst the three types of fixational eye movements, only mi-

[7] **Fovea** is the retinal region with maximal concentration of photoreceptors, where visual acuity is highest.

[8] **Retinal Stabilization** involves a visual stimulus, shifting in sync with the subjects eye movements such that all the eye movements are effectively cancelled and the retinal image remains stable.

crosaccades might contribute significantly to the maintenance of vision [12].

- Horwitz et al. research has shown that the onset of a peripheral stimulus generates microsaccades with a latency of up to 70 ms. If the role of microsaccades were to increase the visibility of the stimulus then turning off the peripheral stimuli should evoke microsaccades rather than turning it on [13].
- In the 1990s it was proposed that microsaccades could produce temporary neural responses. Now the focus has shifted towards the studies on neurophysiological properties of microsaccade-induced activities [2].
- Further researchers have found that microsaccades produce responses in every single visual area, indicating a potentially important physiological role [2].
- Another theory is that there should be a neural mechanism that suppresses these microsaccades, thereby stabilizing the visual perception [3].

Although we now know that microsaccades induce neural activity, the role of microsaccades and other fixational eye movements for visual perception still remains a mystery. The questions still remains open: Why does the world around us remains perceptually stable during fixations, despite the presence of fixational eye movements? How can we control fixational eye movements? Many questions regarding the fixational eye movements and their role in the visual perception are still unanswered [2].

## 1.2 EYE MOVEMENT CONTINGENT DISPLAY (EMCD)

For studying the human visual system, one starts with presenting some controlled visual stimuli and recording the responses. The next step involves designing a system that can modify visual stimuli according to the subject's eye movements [14]. Eye movement contingent display (EMCD) control is a methodology that allows accurate control of the stimulus position and motion on the retina [2]. It has been successfully used in many areas of vision science, ranging from the study of visual attention to the physiological characterization of neuronal response properties. Apart from the vision research, EMCD control is also crucial for development of augmented information displays and image enhancement aids for the visually impaired [14].

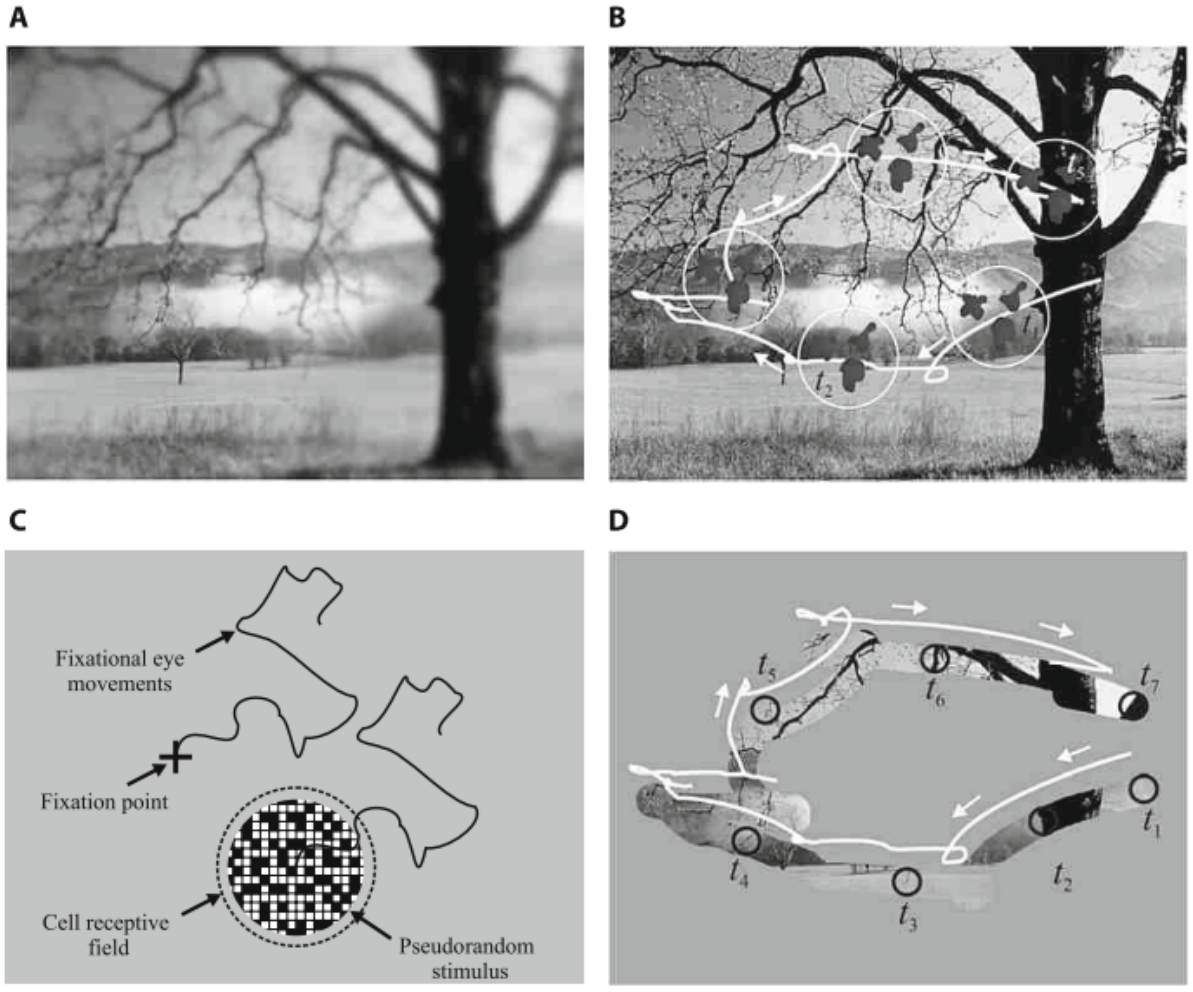


**Figure 1.3:** Examples of EMCD experiments. **(A) Foveated Imaging:** Useful for studying the visual acuity and attention by varying the image resolution to follow the reduction in visual acuity with eccentricity. **(B) Simulation of a Scotoma:** For reproducing the impaired regions of the visual field, the dark spots on the image follow the trajectory of the eyes. Each circle shows the scotomas at different time $t_i$. **(C) Retinal Stabilization in Neurophysiology:** For neurophysiological studies a pseudorandom spatiotemporal signal is used to stimulate a cell in the eyes and a EMCD control enables the stimulus to follow the cell receptive field as it moves with the physiological inability of visual fixation. **(D) Physiological characterization of retinal and extraretinal influences on cell responses:** The experiment involves two phases. In the first phase a subject follows a sequence of guided fixation dot, moving on the scene and the cell responses are recorded. The stimulus is set to grey and only the area covered by the cell receptive field is dynamically made visible. The second phase involves the recording of cell responses while displaying the same stimulation obtained during the first phase. Comparing the two phases allows characterization of proprioceptive influences. Varying the dimensions of the stimulus window enables in determining the influences in the classical and extraclassical receptive fields.("The figure and the caption adopted from: [2]")

The Figure 1.3 shows four applications of the EMCD control in different areas of vision science [14]. Figure 1.3 (A) is a real-time space-variant sampling experiment where the regions of the scene are dynamically modified with respect to the subjects real-time gaze position with the help of EMCD control. The scene is resampled with space-variant resolution relative to the point of fixation, to replicate the eccentricity-dependent resolving power of the visual system [15]. This kind of experiment is useful for studying visual acuity and attention, and quantifying the variables that determine saliency [14]. In Figure 1.3 (B) a simulation of a visual *scotoma*[9] is illustrated where an EMCD system is programmed to enable simulations of various types of visual impairments. Such experiments are useful for medical evaluation of visual performance, patient rehabilitation, and for ophthalmologists training purposes [14]. Figure 1.3 (C) shows simulation of retinal stabilization in the field of neurophysiology. For improving the quality of neurophysiological recordings and to characterize the neural responses in alert subjects, the challenges of unpredictable shifts in the locations of cell receptive fields caused by the oculomotor activities must be overcome [16]. The EMCD system is a powerful tool that can be utilised to overcome these challenges. Despite the presence of fixational jitter, the EMCD control enables accurate positioning of the stimulus in the receptive field of the recorded cell. The experiment technique has helped in characterizing cell response properties in alert monkeys with a level of accuracy comparable to that obtained under anesthesia [14]. As illustrated in Figure 1.3 (D), EMCD control can be used to recognize and segregate the influences on neuronal responses originating from different sources, such as contributions from the *classical*[10] and the *extraclassical*[11] receptive field, as well as *extraretinal*[12] modulations [14].

Studies show that EMCD control can offer valuable assistance in numerous areas of vision science but despite their numerous advantages, there are still some limitations that have restricted the widespread use of this technique. One of the major limitations has been the availability of a general purpose system that enables a flexible gaze-contingent manipulation of the stimulation. Another drawback of the current EMCD techiques has been the difficulty of a guaranteed real-time performance [14] and ensuring an upper limit on the latency between the occurrence of oculomotor

[9] **Scotoma** is an area of partial alteration in the field of vision consisting of a partially diminished or entirely degenerated visual acuity that is surrounded by a field of normal or relatively well-preserved vision.

[10] **Classical Receptive field** is a portion of sensory space that can elicit neuronal responses when stimulated.

[11] **Extraclassical Receptive field** is the region of visual space where stimuli cannot elicit a spiking response but can modulate the response of a stimulus in the classical receptive field.

[12] **Extraretinal:** situated or occurring outside the retina

events to the display change in the stimulus. A real-time EMCD experiment usually requires advanced hardware development and programming expertise that are usually unavailable in vision science laboratories.

With the recent advancements in the technology, there has been a tremendous improvement in eye tracking equipment, video hardware and computational power that has paved the way to a flexible and economical approach for conducting EMCD experiments. The current digital signal processor boards have increased speed and performance, and are much simpler to program and interface with other devices as per experimental requirements. The availability of fast, high quality cathode ray tubes with refresh rates up to 200 Hz for stimulus projection has opened up ways for many experiments in visual neuroscience [14]. Miniaturising the devices and reduction in their cost has enlarged the circle of potential users.

## 1.3 STATE-OF-THE-ART IN EMCD

A gaze-contingent display system integrates an eye tracker, digital signal processing board and a display device that rapidly updates the projected stimulus on the basis of the dynamic eye movements. The vital parameters for research studies in vision science using the EMCD system involves the accurate measurements of the oculomotor events and a minimum system and transmission latency from the change in the gaze location to a relative change in the stimulus displayed.

### 1.3.1 Are Video-based Eye Trackers Accurate?

The accuracy of eye movement measurements is crucial for research studies on fixational eye movements. The *search coil technique*[13] was considered to be the gold standard for measuring the fixational eye movements. The measurements were significantly less noisy in comparison to the video trackers, but it was considered as an invasive technique that may also interfere with the natural vision and cause discomfort to the subject. Due to these complications, most contemporary researchers rely on safer, non-invasive video tracking methods to record eye movements. A detailed study on the reliability of fixational eye movement data obtained by using the contemporary video trackers (EyeLink 1000) in comparison to the search coil technique is discussed in the paper [17]

[13]**Search Coil Technique** is an invasive eye tracking technique which is based on the recording of small electric currents induced by a magnetic field in a coil of very narrow-gauge wire embedded in a pliable donut-shaped plastic ring that is placed on the eye.

by McCamy et al. The studies have shown that 95% of microsaccades detected by the search coil were also detected by the video tracker and 95% of the microsaccades detected by the video tracker were also detected by the search coil. Although there are some differences in the microsaccades and drift properties such as the magnitude, mean peak velocities, and overshoot size between the two systems, the measurement accuracy of a video based eye tracker now approaches the gold standards of the search coil technique for measuring fixational eye movements [17].

### 1.3.2 A Modern EMCD Control: EyeRIS

The article on EyeRIS (Eye movement Real-time Integrated System) describes a state-of-the-art general purpose system for EMCD experiments [14]. Recent advancements in technology have been integrated to build this system that enables flexible designs of EMCD experiments with a maximum latency of 10 ms. Figure 1.4 illustrates the general architecture of the EyeRIS system. The eye-tracker measures the eye movements and feeds the data to the EyeRIS Hardware board. The DSP-based board is programmed with sets of predefined rules as per the experimental requirements. It is equipped with analog and digital inputs for real-time processing of the data. The oculomotor signals and events are continuously transmitted to the host PC, where they are integrated with conditions and functions of stimulus manipulation as per the experimenter's specifications. A CRT monitor with a refresh rate of 200 Hz is used to display the stimulus on the screen thereby inducing a delay <10 ms (typical delay 7.5 ms) between subsequent frames [14].

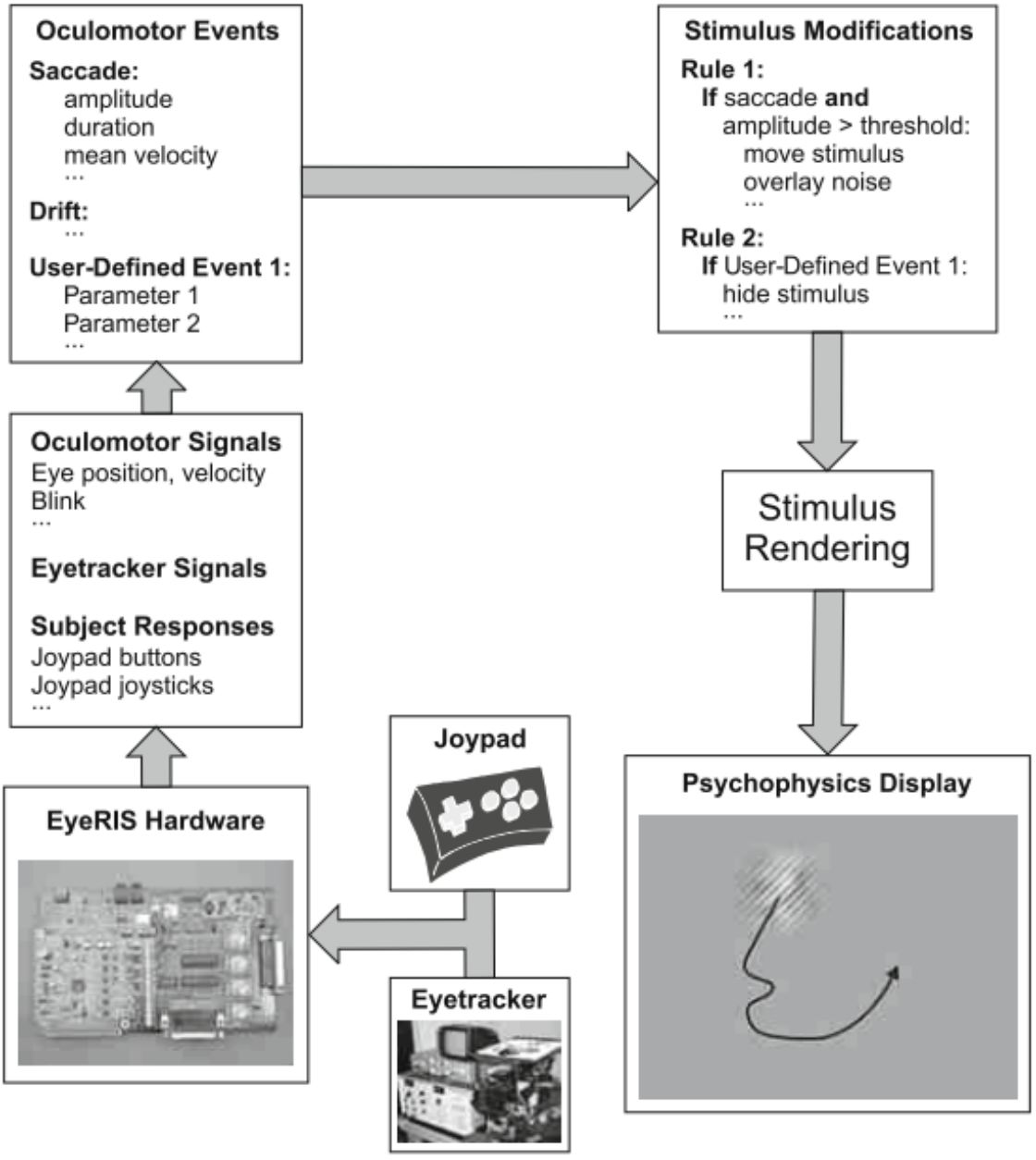


**Figure 1.4: EyeRIS general architecture:** The Eye-tracker records the gaze position which are then sampled and processed by a dedicated board. The digital signal processor board is fed with the tracker outputs and is programmed to generate real-time gaze-contingent paradigms. Upon successive screen refresh, the resultant modifications in the stimuli are displayed on the psychophysical display monitor.("The figure and the caption adopted from: [2]")

### 1.3.3 EMCD for Retinal Stabilization

Retinal Stabilization is an EMCD procedure that involves a human subject holding the eyes physically stationary (with paralysing drugs, for instance) to eliminate the retinal image motion [18]. Given the technical difficulties and potential health hazards, an alternate approach is generally used. It involves a visual stimulus, shifting in sync with the subjects eye movements such that all the eye movements are effectively cancelled and the retinal image remains stable [16]. Retinal stabilization has also been useful for neurophysiological studies, where it has been used to reduce the variability of cell responses [19]. Unfortunately the elimination of retinal motion requires expensive and accurate systems, thus a relatively small number of laboratories have systematically used retinal stabilization. The EyeRIS system provides a simple and fast approach for achieving retinal stabilization with a mean delay of 12 ms [14].

### 1.3.4 EMCD in Neuroscience

One of the most promising developments in neuroscience is the combined recording and analysis of eye movements and the brain activity they generate. Studies have shown that fixational eye movements induce firing of visual neurons [3]. Eye movements have been used as markers for brain responses related to perceptual and cognitive processes during reading [20], image and object recognition [21], and scene perception [22]. The EMCD control have been particularly beneficial for developing diverse experiments to change paradigms such as moving window, moving mask and invisible boundary paradigms. The fundamental principle of these paradigms is changing the stimulus with respect to the fast eye movements [23].

It still remains an unresolved debate of how fixational eye movements together with the neural activities they generate help in the visual perception. Several techniques under development will further improve our understanding in this field. The development of fast and reliable non-invasive retinal stabilization techniques can help in carrying out crucial experiments for decoding the language that our brain uses to represent a stable visual perception. The experiments would include an accurate recording of saccades, microsaccades, drifts and tremor movements. Designing fast retinal stabilization conditions for characterizing the different eye movements and recording the neural responses generated during each of the eye movements [3].

## 1.4 THE VISUAL SCRAMBLER PROJECT

Previous EMCD devices work with high quality CRT monitors with typical refresh rates up to 200 Hz for stimulus projection and can achieve retinal stabilization with a mean latency of 12 ms. A combination of high-speed eye tracking and high-speed visual stimulation is generally required for studying the neural responses related to eye movements. Fast and real-time display changes is a vital factor because of the high susceptibility of the brain to visual stimulation [23].

Studies on fixational eye movements have shown that the natural response rate of retinal ganglion cells can be up to 1000 Hz in humans [2]. With a faster EMCD control having high-speed eye tracker, high-speed visual stimulation and faster digital signal processing board, crucial experiments can be conducted for studying the response of the cells at this frequency. Stimuli generated at this frequency would enable us to discover new effects on brain activity that could further improve our understandings of fixational eye movements and the neural activity they generate.

The project "Visual Scrambler" at the Fraunhofer IOSB supported by the Volkswagen foundation aims to develop a novel EMCD system to communicate visual information to people exploiting the current knowledge about the human visual system. The eye-tracking laboratory has state-of-the-art equipment capable of recording subjects eye movements with the sampling rate of 2 kHz and angular resolution below 0.6 degrees. A specialized projector PROPixx by the VPixx technologies is capable of presenting visual stimuli at rates up to 1440 frames per second. Digital image processing and interfaces introduce an unavoidable latency of at least 4 ms. In order to reduce the perceived stimulation latency, the image generator is built on a fast FPGA platform and implements image transformations at the level of electronics. However, an important parameter for conducting our planned experiments would require an interactive stimulus generation with a delay of at most 5-6 ms between an eye position reading and the stimulus display.

## 1.5 PROBLEM STATEMENT

To achieve the goal parameters for the "Visual Scrambler" project, we need to further reduce the system latency. My Master Thesis focuses on reducing the latency in EMCD down to 2-3 ms (generally below 8-10 ms). With the available digital electronics, an unavoidable latency of at least (7-16 ms) would be introduced. To achieve this goal, we propose an eye movement prediction

system, that could accurately predict the subjects eye positions in real-time for up to 10 ms in advance, thereby reducing the effective latency. An important parameter for developing such an eye-position prediction system is its computational cost which should be less than 2-3 ms for taking in the measurements and giving out the predicted eye positions.

# CHAPTER 2

# METHODOLOGY

## 2.1 OVERVIEW

Interactive stimulus generation with a delay of at most 5-6 ms between an eye position reading and the stimulus display is an important enabling step in many planned experiments. Signal transmission and processing introduces a latency of 10-12 ms between the eye tracker and the stimulus generator. Therefore, to fill the latency gap and generate faster gaze-contingent stimuli, we design prediction algorithms.

The objective of this study is to build time-series forecasting models that can accurately predict eye positions in real-time for 10 ms ($\sim$20 frames) ahead using the present and/or past eye position measurements.

### 2.1.1 Visual Scrambler

The overall architecture of the Visual Scrambler is shown in Figure 2.1. In the Visual Scrambler project, the TRACKPixx3 eye tracker records eye movements. The recorded gaze positions are transmitted through an analog interface to a fast FPGA board introducing transmission delay. The FPGA board is connected to a control PC that programs the board with sets of instructions for performing crucial tasks as per experimental requirements. Although the FPGA board is capable of generating the gaze contingent stimuli at the level of electronics, there is still an unavoidable latency of at least 4 ms due to digital image processing and interfaces. The prediction model, once ready, will be integrated into the FPGA firmware and with the dynamic inflow of the eye positions, the algorithm will be able to predict the future readings to fill the latency gap and generate faster gaze-contingent stimuli. The generated stimuli are then projected onto a screen using the PROPixx projector which is capable of projecting stimuli with a refresh rate of 1440 Hz.

We have conducted eye tracking sessions with multiple participants and collected their eye movement data. A detailed explanation of the experimental procedure will be discussed in a later section. The eye tracking data are saved in the form of a CSV (comma-separated values) file which contains information of the recorded timestamp and the corresponding horizontal and vertical eye positions

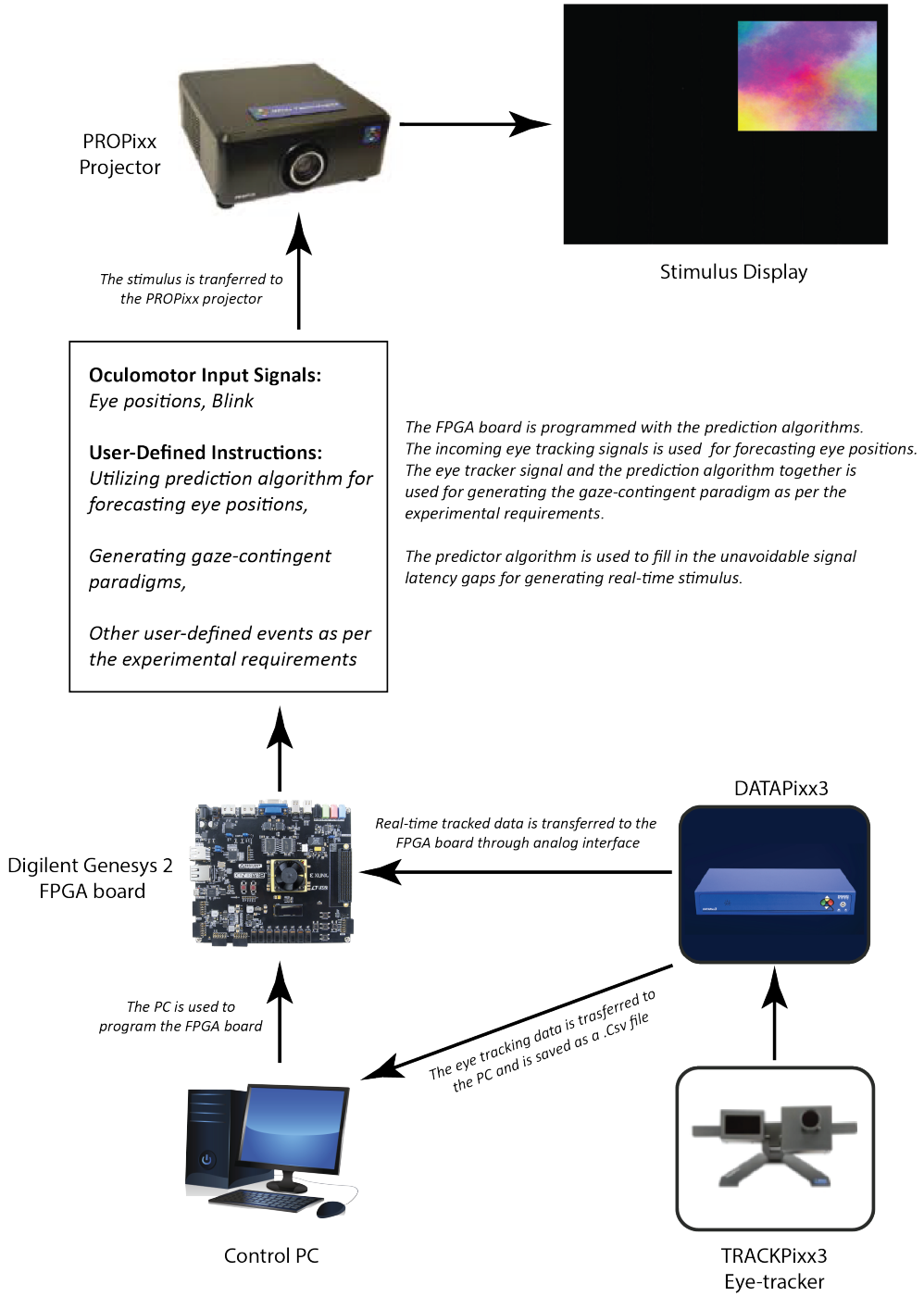


**Figure 2.1: Visual Scrambler architecture and dataflow:** The TRACKPixx3 eye-tracker records a subject's eye movements. The signal is processed and transferred to a multi-functional data hub, DATAPixx3. The signal can be transferred in real-time using analog interface to the FPGA board or can be saved in the control PC for further analysis. The FPGA board is programmed with sets of instructions for generating real-time gaze-contingent stimuli. The PROPixx Projector is capable of projecting stimuli with a refresh rate of up to 1440 Hz.

for both eyes, recorded with a sampling rate of 2000 Hz. The TRACKPixx3 eye tracking system provides other features such as the pupil diameter, blink detection, and saccades and fixational eye movement flags that can be used for conducting various vision science experiments. We use the collected data for designing and testing the quality of the predictor models using the Python programming language.

For developing our prediction models, we make use of deterministic as well as probabilistic forecasting techniques.

## 2.2 MODEL CONSTRAINTS

For designing a real-time eye movement prediction model, we should first consider the model limitations. First the prediction algorithm should be computationally inexpensive to minimize the processing latency. The FPGA board has a weak processor (SoC: 100 MHz MicroBlaze), incapable of multithreading. Therefore, a complex model will be ineffective since it would require a high computation time and memory. Using advanced forecasting techniques such as Particle Filtering and Nonlinear Gaussian Filtering will give us highly accurate predictions, but their complex algorithm increases the computation time. Therefore, here we only consider simpler forecasting algorithms for designing the eye movement prediction models.

## 2.3 ALGORITHMS

### 2.3.1 Deterministic Forecasting Approach

In a deterministic approach we predict the future eye positions using the previous values of eye positions that are measured. In a sliding window approach we use a fixed window of the previous eye positions (1-10 frames) to train the model and at each step, the training window is moved forward as illustrated by the black bar in the Figure 2.2. There are various approaches used for the purpose of training the model. The trained model is then utilized to predict the future eye positions (next 20 frames) in each step as illustrated by the orange bars in the Figure 2.2.

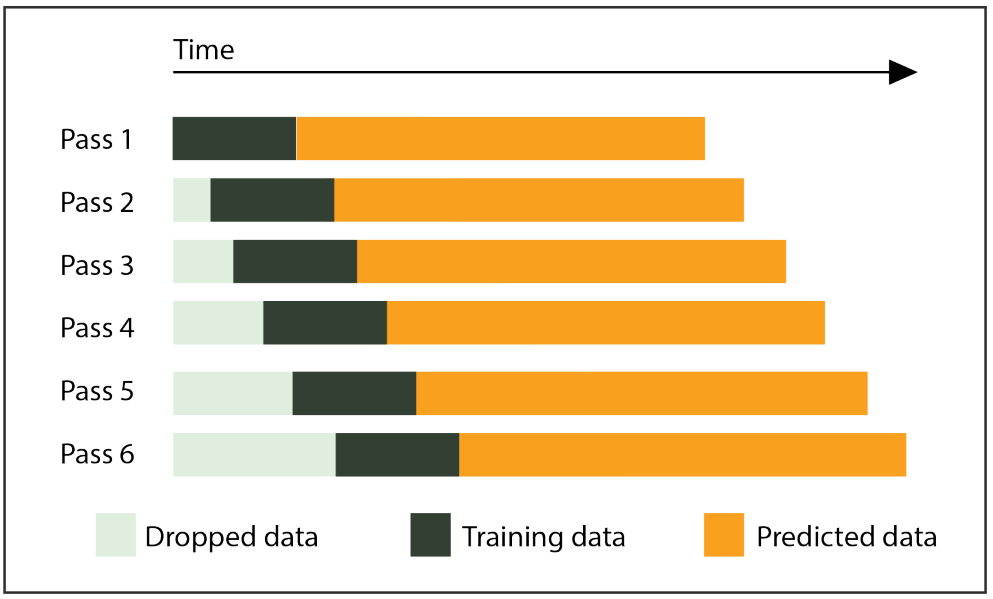


**Figure 2.2: The Sliding Window Approach:** A fixed window of the measured eye position is used to train the model. The model, once trained, is used to forecast the subsequent eye positions based on the resulting trained model. We then slide the training window forward and use the subsequent measurements to train and predict the next set of eye positions.

#### *2.3.1.1 Delta Coding*

The delta coding is a naive forecasting model where we assume that the present eye position measurement holds true for the predicted eye positions as illustrated in Figure 2.3. Here, we take the current eye position in pixel as the fixed window, represented by the black bar in Figure 2.2 and predict the subsequent 20 frames using the delta coding algorithm:

$$\hat{x}_{k+\Delta t} = x_k \tag{2.1}$$

Here, $x_k$ is used to denote the present eye position (pixel) at time step $t_k$ and $\hat{x}_{k+\Delta t}$ is the predicted eye position. $k$ is the discrete time step and is related to the continuous time via the equation $t_{k+1} = t_k + T$, where $T$ is the sampling time. The value of $\Delta t$ increases from 1 to 20, thereby giving us eye position for the next twenty time steps.

To evaluate the prediction efficiency of the model, we determine the prediction error for each pass. The difference between the predicted data and the actual measured data gives us the prediction

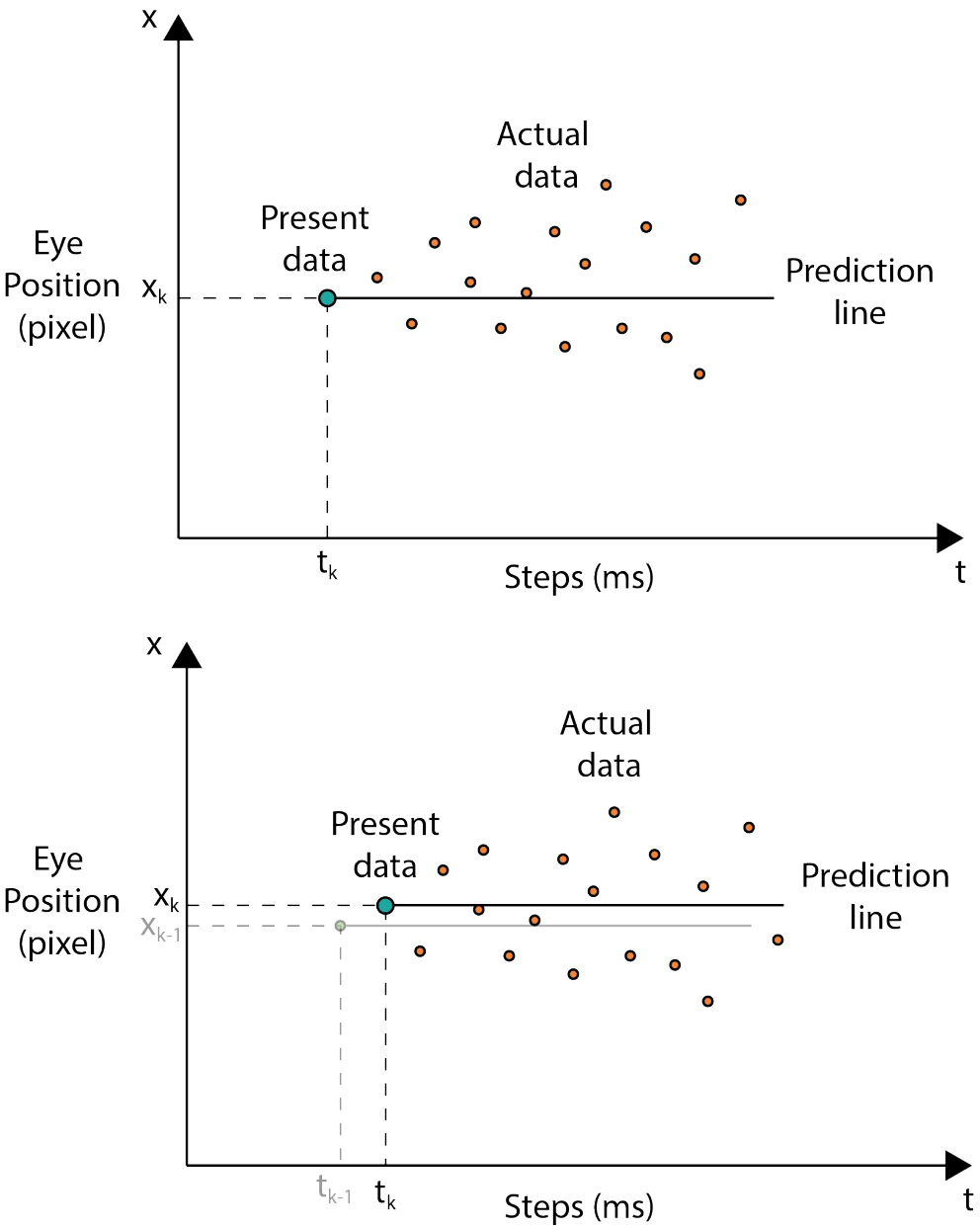


**Figure 2.3: Delta Coding:** The straight line represents the prediction of the delta coding model. The top graph represents the prediction of the delta coding model for Pass 1. The present eye position data $x_k$ holds true for the predicted eye positions and we get a horizontal straight prediction line. The graph below illustrates how the prediction line shifts from the the previously recorded eye position $x_{k-1}$ to the new eye position $x_k$ during Pass 2 and now, the new horizontal line predicts the eye positions for the subsequent 20 frames. The difference in the observed eye positions and the predicted straight line gives us the error in prediction.

error:

$$e_{k+\Delta t} = \hat{x}_{k+\Delta t} - x_{k+\Delta t} \tag{2.2}$$

During each pass, the model predicts the subsequent 20 eye positions and we use the Equation 2.2 to calculate the prediction error $e_{k+\Delta t}$ by subtracting each predicted eye positions $\hat{x}_{k+\Delta t}$ with the corresponding measured eye positions $x_{k+\Delta t}$ for that frame.

**Algorithm 1:** Delta Coding Algorithm

```
1 N = Measured eye position;        ▷ N is a row vector with all the measured eye positions;
2 for k in range len(N) do
3 |   for Δt in range 20 do
4 |   |   x̂_{k+Δt} = x_k;
5 |   end
6 end
```

#### *2.3.1.2 Linear Regression*

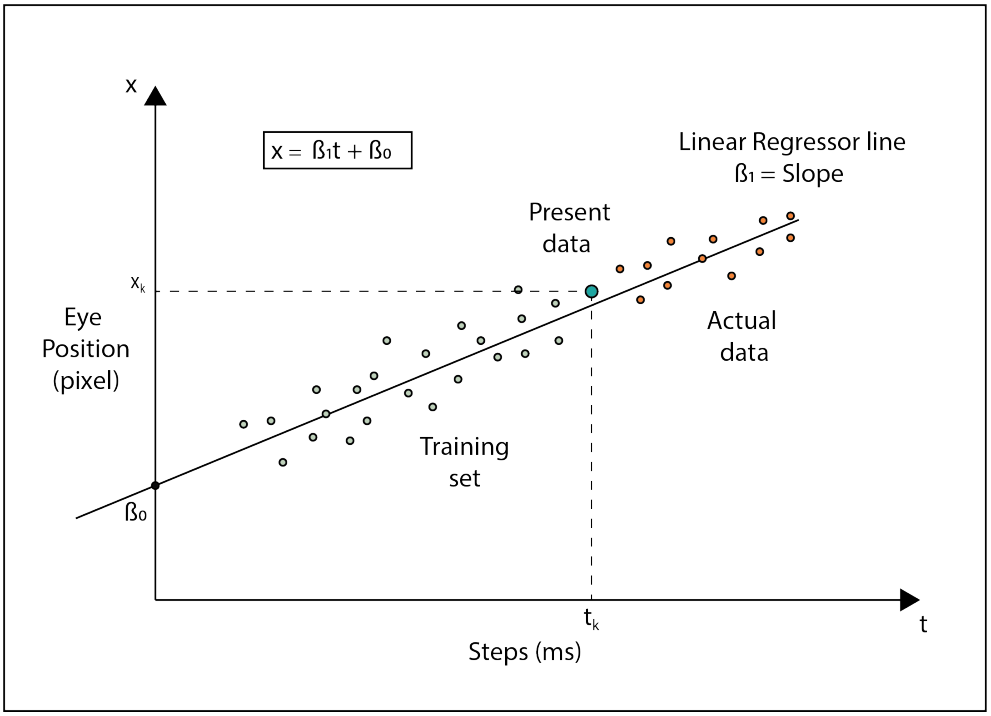


**Figure 2.4: Linear Regression Model:** Here, we plot the eye position $x$ with respect to time steps $t$. $k$ is used to represent the present instance such that at the present time step $t_k$, the measured eye position is $x_k$. The straight line represents the linear regression line with the Y-axis intercept of $\beta_0$ and a slope of $\beta_1$. The previously measured eye positions before the time step $t_k$ are used to train the model and compute the future prediction. The error in prediction is the vertical distance between the regression line from the measured eye positions.

Linear Regression is a statistical method used to model the relationship between two variables by fitting a linear equation to observed data [24]. One variable is considered to be the dependent variable and the other is considered to be an independent variable and we assume a linear relation between them. This method can be utilized for forecasting under the assumption of continuing the correlation between the variables in the future [25].

A single variable simple linear regression model hypothesis equation is represented by [26]:

$$x = \beta_1 t + \beta_0 \tag{2.3}$$

Here, $t$ is the time steps which represents the independent variable and $x$ is the eye position

which represents the dependent variable. In the case of eye movement prediction, the previous eye positions (pixel) with respect to the time step are used to train the regression model. $\beta_0$ and $\beta_1$ together constitutes the weight matrix which defines our regression line.

As illustrated in Figure 2.4 $\beta_0$ is the Y-axis intercept and $\beta_1$ is the slope of the regression line. We use a training window $n$ of prior measured data to compute the regression parameters at each pass. Once the regression parameters are computed, the model is ready to predict the future eye positions. (See Figure 2.4). The hypothesis equation can be rewritten as follows:

$$x_{k-\Delta n} = \beta_1 t_{k-\Delta n} + \beta_0 \tag{2.4}$$

Here, $x_{k-\Delta n}$ is the previous eye position data used to train our regression model with respect to time $t_{k-\Delta n}$. $k$ is the present time step, after which the model is trained and ready to give us future predictions. To get the best fit regression line, we first need to compute the model parameters $\beta_0$ and $\beta_1$. We will be using the least-square approach to compute $\beta_0$ and $\beta_1$, making use of the training data set. The equations are given below [27]:

$$\beta_1 = \frac{\sum_{i=1}^{n}(t_{k-i} - \bar{t}_k)(x_{k-i} - \bar{x}_k)}{\sum_{i=1}^{n}(t_{k-i} - \bar{t}_k)^2} \tag{2.5}$$

$$\beta_0 = x_k - \beta_1 t_k \tag{2.6}$$

Here, $n$ is the number of previous data points used to train (fit) our model. The equations 2.5 and 2.6 are used to compute the model parameters $\beta_1$ and $\beta_0$. $\bar{x}_k$ is the arithmetic mean of all the training set eye positions and $\bar{t}_k$ is the mean time step which is equal to 0.5 ms. Once we have our model parameters, we can substitute the values in our hypothesis equation and get future eye positions. We use the equation given below for predicting the eye positions.

$$\hat{x}_{k+\Delta t} = \beta_1 t_{k+\Delta t} + \beta_0 \tag{2.7}$$

Where, $\hat{x}_{k+\Delta t}$ is the predicted eye position and $t_{k+1} = t_k + T$, where $T$ is the sampling time. The

**Algorithm 2:** Linear Regression Algorithm

$N$ = Measured eye position; ▷ N is a matrix with all the measured eye positions with respect to time step
$n = 5$; ▷ Length of the training data;
**for** *k in range len(N)* **do**
    $\bar{x}_k = 0$;
    $\bar{t}_k = 0.5ms$;
    $s_x = 0$;
    $s_t = 0$;
    **for** *i in range n* **do**
        $\bar{x}_k = \bar{x}_k + x_{k-i}$;
    **end**
    $\bar{x}_k = \frac{\bar{x}_k}{n}$; ▷ Calculate the mean eye position;
    **for** *i in range n* **do**
        $s_{xi} = x_{k-i} - \bar{x}_k$;
        $s_x = s_x + s_{xi}$;
        $s_{ti} = t_{k-i} - \bar{t}_k$;
        $s_t = s_t + s_{ti}$;
    **end**
    $\beta_1 = \frac{s_x s_t}{s_t^2}$; ▷ Computing model parameter $\beta_1$;
    $\beta_0 = x_k - \beta_1 t_k$; ▷ Computing model parameter $\beta_0$;
    **for** $\Delta t$ *in range 20* **do**
        $x_{k+\Delta t} = \beta_1 t_{k+\Delta t} + \beta_0$; ▷ The Hypothesis equation used for prediction;
    **end**
**end**

value of $\Delta t$ increases from 1 to 20, thereby giving us eye position for the next twenty time steps. The complete Linear regression algorithm is given in the Algorithm 2.

#### *2.3.1.3 Polynomial Regression*

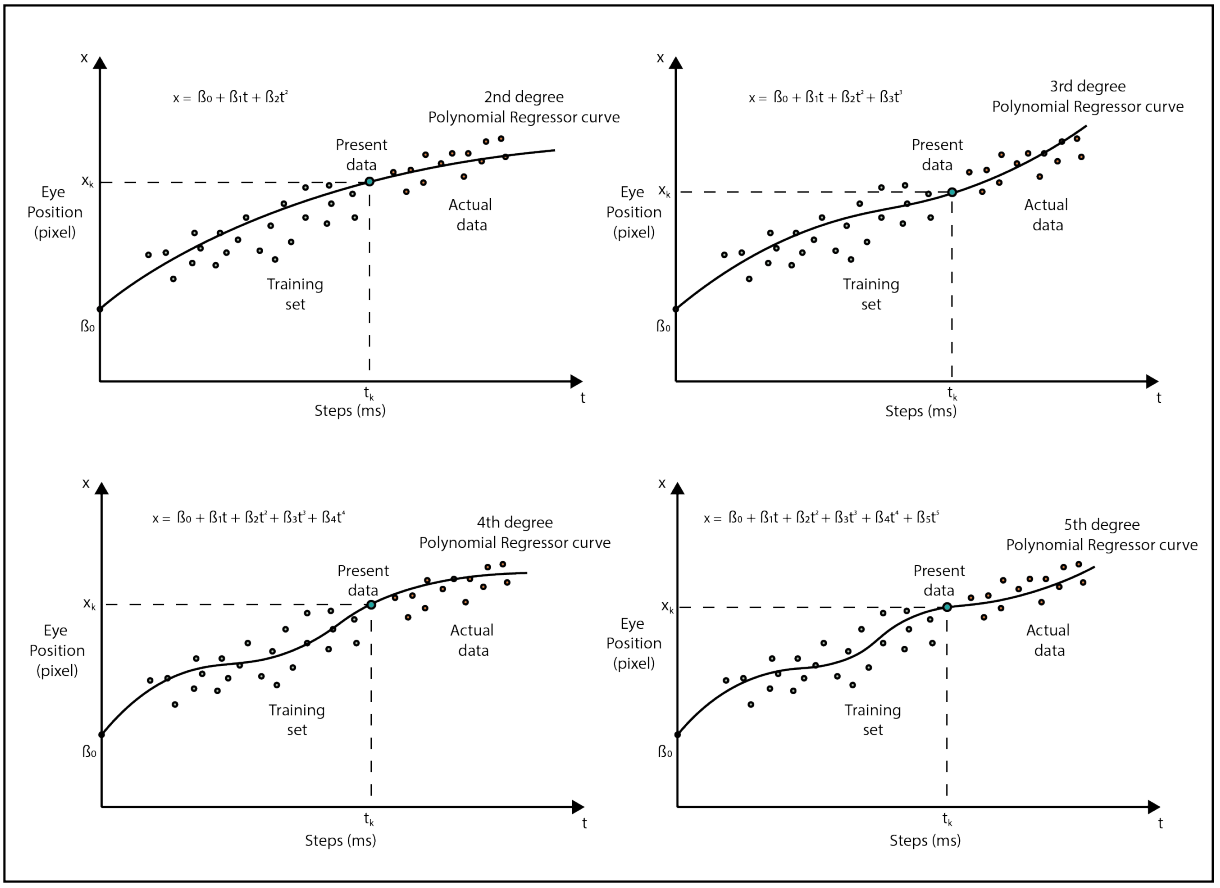


**Figure 2.5: Polynomial Regression Model:** Illustrates the polynomial regression curve with increasing order of polynomial. The measured eye positions are used to train the model and compute the prediction. The error in prediction is the vertical distance between the regression curve from the actual eye positions.

Instead of using a straight line prediction, we can also use a curve for prediction. A polynomial regression, similar to a simple linear regression, has a dependent and an independent variable. A single variable polynomial regression model can be represented as [26]:

$$x = \beta_0 + \beta_1 t + \beta_2 t^2 + .... + \beta_k t^m \tag{2.8}$$

Where $x$ is the eye positions that represents the dependent variable and $t$ is the time steps which represents the independent variable. $m$ is the degree of the polynomial. The shape of the regression curve depends on the order of our polynomial regression. For a first degree polynomial regression, we take $m$=1 in the equation 2.9 and we obtain $x = \beta_0 + \beta_1 t$ which is the same as a simple linear regression equation 2.3 giving us a straight regression line. As we increase the degree of polynomial, we get a wider range of curvature as illustrated in Figure 2.5. A second order polynomial forms a quadratic expression, hence giving us a parabolic curve. Similarly, a third order polynomial forms a cubic expression, and fourth order polynomial gives us a quartic expression [28]. The difference in the shape of the curves with the increasing polynomial degree is illustrated in the Figure 2.5.

For designing our prediction model, we will be using a second order polynomial regression model here. The hypothesis equation used for training the model is given below:

$$x_{k-\Delta n} = \beta_0 + \beta_1 t_{k-\Delta n} + \beta_2 t^2_{k-\Delta n} \tag{2.9}$$

The notation used in the equation above is same as in the case of linear regression model. For computing the model parameters $\beta_0$, $\beta_1$ and $\beta_2$, we use the least-square approach, taking the previous five measured data to train our model. Cramer's rule allows us to solve the linear system of equations to find the regression coefficients [26]:

$$\begin{bmatrix} 1 & x_{k-1} & x^2_{k-1} \\ 1 & x_{k-2} & x^2_{k-2} \\ 1 & x_{k-3} & x^2_{k-3} \\ 1 & x_{k-4} & x^2_{k-4} \\ 1 & x_{k-5} & x^2_{k-5} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} t_{k-1} \\ t_{k-2} \\ t_{k-3} \\ t_{k-4} \\ t_{k-5} \end{bmatrix} \tag{2.10}$$

The equation 2.10 takes the form of $X\vec{\beta} = \vec{Y}$, where $\vec{\beta}$ is a row vector that contains the regression parameters. Using Cramer's rule, we can compute the regression parameters $\beta_0$, $\beta_1$ and $\beta_2$.

$$\vec{\beta} = X^{-1}\vec{Y} = \frac{1}{\det X}.adjX.\vec{Y} \tag{2.11}$$

Once we have our model parameters, we can substitute the values in our hypothesis equation and get future eye positions. We use the equation given below for predicting the eye positions.

$$\hat{x}_{k+\Delta t} = \beta_0 + \beta_1 t_{k+\Delta t} + \beta_2 t^2_{k+\Delta t} \tag{2.12}$$

Here, $\hat{x}_{k+\Delta t}$ is the predicted eye position (pixel) and $t_{k+1} = t_k + T$, where $T$ is the sampling time. The value of $\Delta t$ increases from 1 to 20, thereby giving us eye position for the next twenty time steps. The complete algorithm of our polynomial regression model is given below (See Algorithm 3)

**Algorithm 3:** Polynomial Regression Algorithm

$N$ = Measured eye position; ▷ N is a matrix with all the measured eye positions with respect to time step

$n = 5$; ▷ Length of the training data;

**for** *k in range len(N)* **do**

$$X = \begin{bmatrix} 1 & x_{k-1} & x^2_{k-1} \\ 1 & x_{k-2} & x^2_{k-2} \\ 1 & x_{k-3} & x^2_{k-3} \\ 1 & x_{k-4} & x^2_{k-4} \\ 1 & x_{k-5} & x^2_{k-5} \end{bmatrix};$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix};$$

$$Y = \begin{bmatrix} t_{k-1} \\ t_{k-2} \\ t_{k-3} \\ t_{k-4} \\ t_{k-5} \end{bmatrix}$$

$\beta = X^{-1}.Y$; ▷ Computing model parameters;

**for** $\Delta t$ *in range 20* **do**

$x_{k+\Delta t} = \beta_0 + \beta_1 t_{k+\Delta t} + \beta_2 t^2_{k+\Delta t}$; ▷ The Hypothesis equation used for prediction;

**end**

**end**

### 2.3.2 Probabilistic Forecasting Approach

Since the future is uncertain, a prediction may output a probability distribution over the future events [29]. Unlike a deterministic forecasting technique where we get a single point estimation,

a probabilistic model gives us distributional or probabilistic forecasts [30]. Probabilistic forecasts serve to quantify the uncertainty in a prediction which forms an essential ingredient for optimal decision making [29].

We utilize probabilistic forecasting techiques to predict the eye motions, using the past and the present eye position data, taking into account the various measurement uncertainties such as the process noise, inaccuracy in eye tracking due to the trackers angular resolution and also the experimental conditions. The algorithm predicts the eye positions while making use of the prior eye movement trends.

#### *2.3.2.1 Kalman Filter Constant Velocity Model*

The Kalman Filter belongs to a family of Bayesian filters which probabilistically estimate a dynamic system's state using noisy observations [31]. The technique was originally published in 1960 by RE Kalman [32] and was initially designed for spacecraft navigation system. Since then, due to its general nature, the Kalman filter technique has been vastly used in many fields [33]. Presently, Kalman filters are the most widely used variant of Bayes filters [34] with applications in object navigation, instrumentation, demographic modelling, manufacturing industry, fuzzy logic, and neural network training [35][33]. The filter combines the limited knowledge of how a system behaves with the noisy and limited sensor measurements to produce an estimate of the state of the system [36].

For predicting the eye movements, we take a Kalman filter constant velocity approach where, as the name suggests, we assume the eye to move at a constant velocity with respect to time, thereby assuming the acceleration to be zero. Although the assumption of only considering a constant velocity for eye motion tracking might be misleading, still it would be interesting to evaluate the results obtained through such a model. The measured eye positions from the eye tracker contains uncertainties if we consider the angular resolution ($0.6^o$) of our device. We will be considering this measurement noise while modelling our Kalman filter model.

In the Figure 2.6, we see the continuous cycle of Kalman filter model. The Kalman filter uses a form of feedback control for estimating the eye positions [37]. We divide the filter process into two groups: The predict step is responsible for projecting the current state and error covariance estimates to obtain prior estimates for future eye positions. The update step acts as the estimate

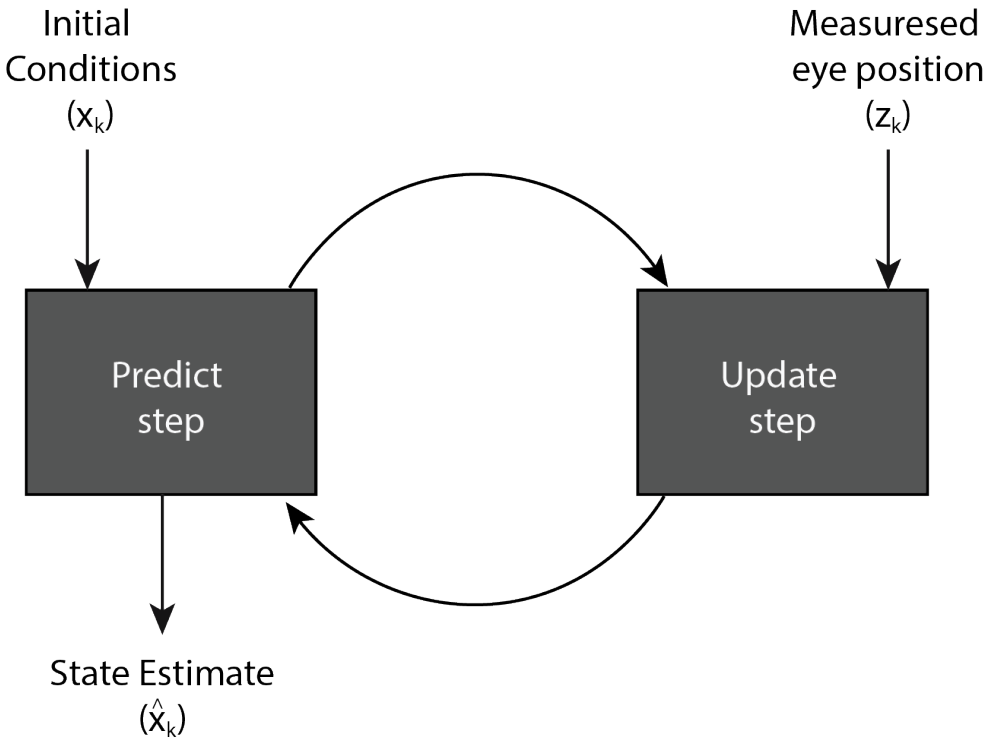


**Figure 2.6: Kalman Filter Algorithm:** [36] A continuous Kalman filter cycle is illustrated. During the predict step, the model projects the current state estimate ahead in time. In the update step, the model uses the incoming measurement data from the eye tracker and adjusts the projected estimate using the actual measurement data at that time.

corrector by incorporating the newly measured eye position with the prior estimates to obtain an improved a posteriori estimate. The Kalman filter being a recursive estimator, takes in this a posteriori estimate and utilizes this information to give us the next prediction [37]. Due to this continuous predict and update loop, the Kalman filter, within few iterations is able to estimate the true eye positions.

For the purpose of simplicity, we model a 1D Kalman Filter which describes the horizontal eye movements. The predict step equations for our constant velocity Kalman filter model comprises of the target predicted state and the predicted covariance equations which follow Newton's laws of motion. The discrete-time version of the target state vector and process covariance matrix is described by the following equations:

$$\hat{x}_{k|k-1} = A_{k-1}\hat{x}_{k-1|k-1} \tag{2.13}$$

$$P_{k|k-1} = A_{k-1}P_{k-1|k-1}(A_{k-1})^T + Q_{k-1} \tag{2.14}$$

Here $k$ is the discrete time step and is related to the continuous time via $t_k = t_{k-1} + T$, where T is the sampling time. For the TRACKPixx3 eye tracker, the sampling time $T = 0.5$ ms. Here, Equation 2.13 is used to compute the predicted state and equation 2.14 computes the predicted covariance. We use the Equation 2.13 to model our system dynamics. Here $\hat{x}_{k-1|k-1}$ is the system prior state estimate at discrete time step $k-1$. $\hat{x}_{k|k-1}$ is the system predicted state at time step $k$. $P_{k|k-1}$ is the state prediction covariance. For a 1D constant velocity Kalman filter, the state matrix is described by $X = \begin{bmatrix} x & \dot{x} \end{bmatrix}^T$ which contains the position $x$ and the velocity $\dot{x}$ of the eye motion. The velocity $\dot{x}$ is the derivative of position with respect to time [38].

$$x_{0|0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.15}$$

$$P_{0|0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.16}$$

We initialize the model with the initial state $x_{0|0}$ and the initial uncertainty $P_{0|0}$. By utilizing the incoming eye tracker measurements, the Kalman filter is able to quickly adapt itself and converge to the true position and linear velocity value [38].

$A_{k-1}$ is the discrete-time system transition matrix [38] which we use to compute the new predicted state. $Q_{k-1}$ is the covariance matrix of the discrete-time process noise sequence. $A_{k-1}$ and $Q_{k-1}$ for a 1D constant velocity Kalman filter is given by:

$$A_{k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{2.17}$$

$$Q_{k-1} = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 \\ 1/2(\Delta t)^3 & \Delta t^2 \end{bmatrix} \sigma_x^2 \tag{2.18}$$

Here $\Delta t$ is the prediction time. For predicting the subsequent 20 frames we increment $\Delta t$ from 1 to 20 and predict the eye positions using equation 2.13. $\sigma_x$ is the process noise covariance factor for our model.

The update step takes in the new measurement of the eye position and adjusts the projected estimate by the actual measurement at that time, thereby computing the a posteriori variables [38]. The update state steps involves measurement residual, residual covariance, calculating the Kalman filter gain, Update state and finally the covariance update. The update step equations are given by:

$$\bar{z}_k = z_k - H_k \hat{x}_{k|k-1} \tag{2.19}$$

$$S_k = H_k P_{k|k-1} (H_k)^T + R_k \tag{2.20}$$

$$K_k = \frac{(H_k)^T P_{k|k-1}}{S_k} \tag{2.21}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \bar{z}_k \tag{2.22}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T \tag{2.23}$$

Where $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$ is a transition matrix, $z_k$ is the actual measured eye position and $\bar{z}_k$ is the measurement residual which is used to correct the filter prediction and compute the a posteriori state estimate using equation 2.22 [38]. $R_k$ is the measurement noise covariance which is equal to $\sigma_L^2$. We consider the angular resolution $(0.2^o - 0.6^o)$ of our device to calculate $\sigma_L$. For a distance of 60 cm between a subjects eye and the display monitor, we can calculate $\sigma_L$, the standard deviation in pixels. As illustrated in the Figure 2.7, we use trigonometry to find the measurement error.

Equation 2.21 is the Kalman filter gain equation. The equation takes into account the stochastic nature of the process and the measurement dynamics, in order to produce an optimal linear esti-

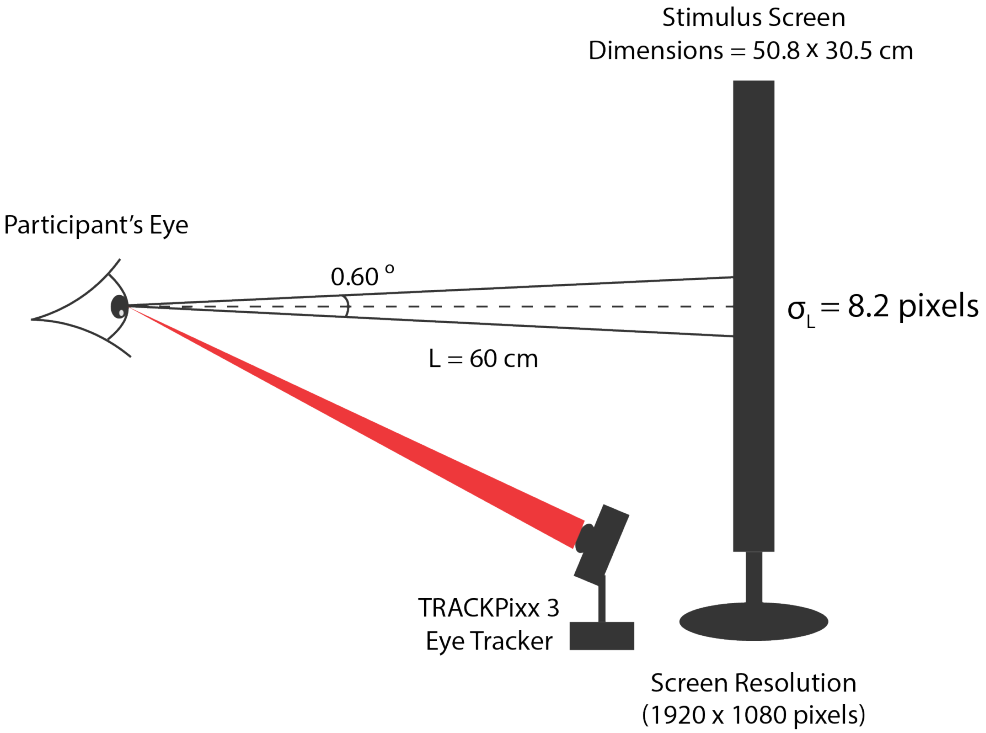


**Figure 2.7:** Illustrating the measurement noise induced by the angular resolution of our TRACKPixx3 eye tracking system for a typical sample recording. For an eye to monitor distance of 60 cm and angular resolution of $0.6^o$ we can compute the measurement uncertainty $\sigma_L$.

mator $\hat{x}_{k|k}$ [38]. The Kalman gain decides what fraction of the measured input and what fraction of the predicted state to use, combining the two informations to then update the new estimate.

After one complete cycle of a constant velocity Kalman filter, we end up with the a posteriori estimate state $\hat{x}_{k|k}$ and covariance $P_{k|k}$. For the subsequent cycle, we use these in the equation 2.13 and 2.14 to compute the new prediction states and covariance. Algorithm 4 illustrates the algorithm for the Kalman constant velocity model.

**Algorithm 4:** Kalman Filter Constant Velocity Model Algorithm

$N$ = Measured eye position; ▷ $N$ is a row vector with all the measured eye positions;

$x_k = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix;

$P_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix;

$\sigma_x = 0.1$;

$\sigma_l = 8$;

$R = \sigma_l^2$; ▷ Measurement noise covariance;

$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$;

**for** *i in range len(N)* **do**

  **for** $\Delta t$ *in range 20* **do**

    $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$; ▷ Transition Matrix;

    $Q = \begin{bmatrix} 1/3(\Delta t)^4\sigma_x^2 & 1/2(\Delta t)^3\sigma_x^2 \\ 1/2(\Delta t)^3\sigma_x^2 & \Delta t^2\sigma_x^2 \end{bmatrix}$; ▷ Process Noise covariance matrix;

    $\hat{x}_k = Ax_k$; ▷ The predict state;

    $\hat{P}_k = AP_kA^T + Q$; ▷ The predict covariance;

    **if** $\Delta t == 1$ **then**

      $\hat{x}_0 = \hat{x}_k$;

      $\hat{P}_0 = \hat{P}_k$;

    **end**

  **end**

  $z_k = N[i]$; ▷ New measured eye position;

  $\bar{z}_k = z_k - H\hat{x}_0$; ▷ Residual;

  $S = H\hat{P}_0(H)^T + R$; ▷ System uncertainty;

  $K = \frac{H^T\hat{P}_0}{S_k}$; ▷ Kalman gain;

  $x_k = \hat{x}_0 + K\bar{z}_k$; ▷ The estimate state;

  $P_k = \hat{P}_0 - KSK^T$; ▷ The estimate covariance;

**end**

#### *2.3.2.2 Kalman Filter Constant Acceleration*

A popular approach used for tracking maneuvering targets is to estimate the target velocity and acceleration on the basis of the targets measured position using Kalman filter constant acceleration algorithm [39]. Unlike the constant velocity model, the key idea here is to assume a constant accelerated eye motion. The eye movements are assumed to be moving at a constant acceleration and is acted upon by a system noise which disturbs the constant acceleration motion. In the real world the acceleration will not be perfectly constant, hence the system noise accounts for the inaccuracies in our constant acceleration assumptions [39].

For the 1D Kalman filter constant acceleration model, the state vector is expressed by [40]:

$$X = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \tag{2.24}$$

where matrix $X$ contains the position $x$, velocity $\dot{x}$ and acceleration $\ddot{x}$ of the eye motion. The velocity $\dot{x}$ and acceleration $\ddot{x}$ are the first and second order derivative of the position with respect to time. The state transition matrix $A_{k-1}$ and the process noise covariance matrix $Q_{k-1}$ together are used to model the system dynamics of the Kalman constant acceleration model [39] [41]:

$$A_{k-1} = \begin{bmatrix} 1 & \Delta t & \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

$$Q_{k-1} = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 & 1/2(\Delta t)^2 \\ 1/2(\Delta t)^3 & (\Delta t)^2 & \Delta t \\ 1/2(\Delta t)^2 & \Delta t & 1 \end{bmatrix} \sigma_x^2 \tag{2.26}$$

Here, similar to the constant velocity model, $\Delta t$ and $\sigma_x$ are the prediction steps and process noise covariance factor respectively for our model. The measurement transition matrix $H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, instructs the model to select the incoming measurement data as an eye position. The initial state vector and uncertainty matrix are given by:

$$x_{0|0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.27}$$

$$P_{0|0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

Once the initial parameters are initialized, we can use the same sets of equations that were used by the Kalman filter constant velocity model to predict and update our model. The stepwise equation of the predict and update step are given below:

$$\hat{x}_{k|k-1} = A_{k-1}\hat{x}_{k-1|k-1} \tag{2.29}$$

$$P_{k|k-1} = A_{k-1}P_{k-1|k-1}(A_{k-1})^T + Q_{k-1} \tag{2.30}$$

$$\bar{z}_k = z_k - H_k\hat{x}_{k|k-1} \tag{2.31}$$

$$S_k = H_kP_{k|k-1}(H_k)^T + R_k \tag{2.32}$$

$$K_k = \frac{(H_k)^TP_{k|k-1}}{S_k} \tag{2.33}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\bar{z}_k \tag{2.34}$$

$$P_{k|k} = P_{k|k-1} - K_kS_kK_k^T \tag{2.35}$$

For a detailed understanding of the working principle of the equations, refer to the previous section. The predict and the update equations together form the recursive algorithm that estimates the future eye positions. The system and measurement noise parameters can be adjusted, depending upon the experimental setup used, for example the eye to display screen distance or room illumination. In a later section we visualize and compare the prediction results with the results from other models. The complete algorithm can be seen in Algorithm 5.

**Algorithm 5:** Kalman Filter Constant Acceleration Model Algorithm

$N$ = Measured eye position; ▷ $N$ is a row vector with all the measured eye positions;

$x_k = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix;

$P_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix;

$\sigma_x = 0.1$;

$\sigma_l = 8$;

$R = \sigma_l^2$; ▷ Measurement noise covariance;

$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$;

**for** *i in range len(N)* **do**

  **for** $\Delta t$ *in range 20* **do**

    $A_{k-1} = \begin{bmatrix} 1 & \Delta t & \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Transition matrix;

    $Q_{k-1} = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 & 1/2(\Delta t)^2 \\ 1/2(\Delta t)^3 & (\Delta t)^2 & \Delta t \\ 1/2(\Delta t)^2 & \Delta t & 1 \end{bmatrix} \sigma_x^2$; ▷ Process noise covariance matrix;

    $\hat{x}_k = Ax_k$; ▷ The predict state;

    $\hat{P}_k = AP_kA^T + Q$; ▷ The predict covariance;

    **if** $\Delta t == 1$ **then**

      $\hat{x}_0 = \hat{x}_k$;

      $\hat{P}_0 = \hat{P}_k$;

    **end**

  **end**

  $z_k = N[i]$; ▷ New measured eye position;

  $\bar{z}_k = z_k - H\hat{x}_0$; ▷ Residual;

  $S = H\hat{P}_0(H)^T + R$; ▷ System uncertainty;

  $K = \frac{H^T\hat{P}_0}{S_k}$; ▷ Kalman gain;

  $x_k = \hat{x}_0 + K\bar{z}_k$; ▷ The estimate state;

  $P_k = \hat{P}_0 - KSK^T$; ▷ The estimate covariance;

**end**

#### *2.3.2.3 Adaptive Filter Model*

A Kalman filter constant velocity model considers the eye motion to be well behaved in relation to our model, where we take the assumption that our eyes are moving at a reasonably constant speed or varies its velocity very slowly. The model may work well while predicting the fixational eye movements due to its steady behaviour, but in the real-world, eye movements include fast moving saccadic eye motions as well. In these situations the constant velocity filter perform quite poorly [36]. The Adaptive Kalman filtering algorithm is used for achieving the best performance from a Kalman filter, using adaptive techniques for the stochastic model to integrate the eye motion dynamics and the noisy measurement conditions [42].

One method involves varying the process noise parameter $Q$ of our constant velocity Kalman filter to adapt the different eye motion dynamics [42]. During constant speed, the filter performs optimally but during fast moving saccadic motions and microsaccades, the filter breaks down and as a result we get faulty predictions. The constant acceleration Kalman filter adapts itself to the fast movements and is able to give us a precise estimate. By varying the process noise parameter $Q$ with an artificially large value, the constant velocity Kalman filter can give a fair prediction for varying velocity of eye motion [36]. Initializing a lower process noise $Q$ will command the filter to trust its predictions over the measured data, thereby using a greater fraction of the predicted state for computing the estimate state. On the other hand, a higher process noise will increase the uncertainty in the filter prediction and hence the filter is more inclined towards the measured eye position while computing the estimate state [42]. In addition, a larger process noise would result in a longer estimation transition and a noisy output. While this can work in the sense of providing a non-diverging filter, the result is not optimal [36].

Before we discuss how to create an adaptive filter we have to first find a way to detect a maneuver. Without knowing when a maneuver is happening, we cannot reasonably adapt a filter to respond to swift eye movements. A tracked object is said to be maneuvering during the onset of an accelerated motion. In general, we can say that the object is maneuvering if its behaviour is different than the process model being used by our Kalman filter, thereby giving us a larger residual. The residual $\bar{z}_k$ is the difference between the filters current prediction and the measurement [36]:

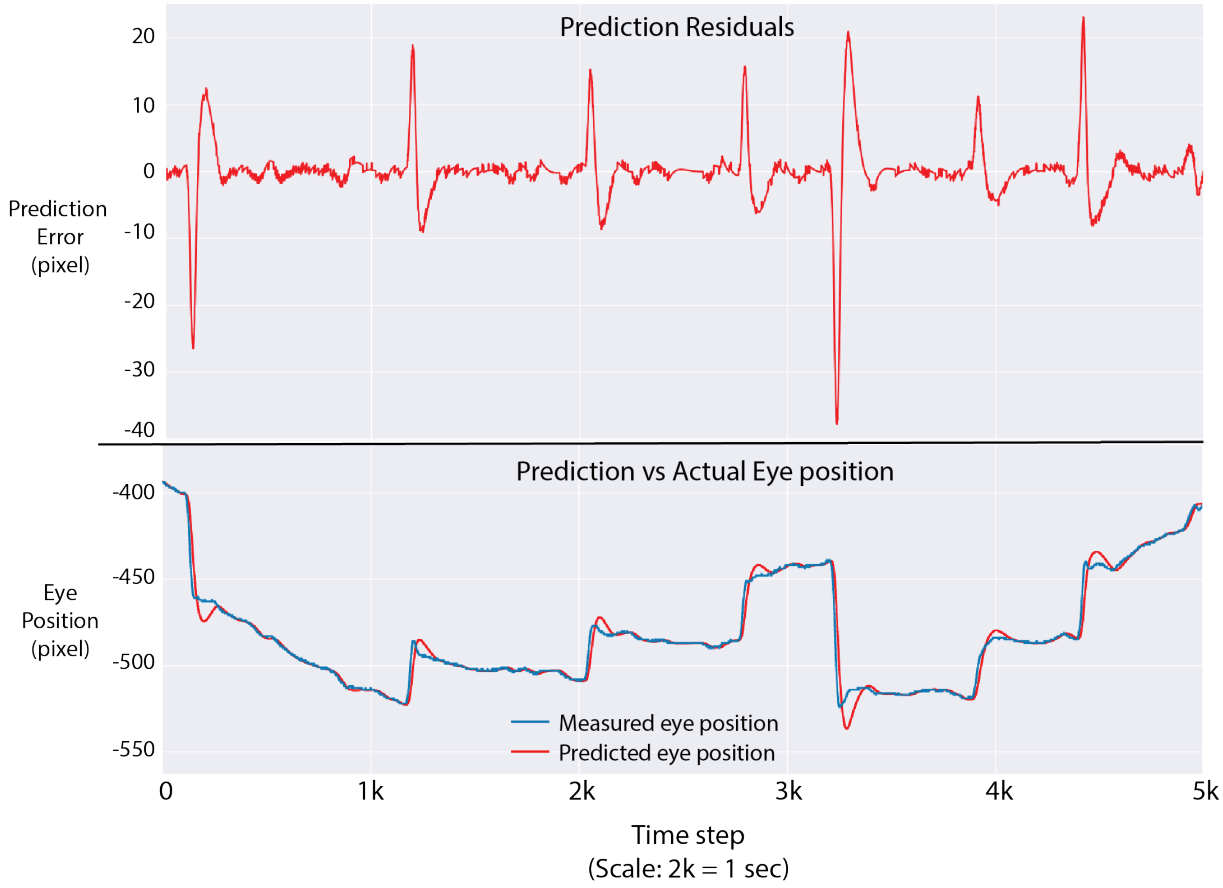


**Figure 2.8: Maneuver detection using the filter residual:** The top graph is the residual plot $\bar{z}_k$ and the bottom plot illustrates the measured eye positions against the constant velocity Kalman filter prediction. During a constant or slowly varying speed, the filter predictions match the measurements and therefore the residual is around 0. Once the eye motion starts accelerating, the predictions of the filter becomes uncertain and an overshoot is observed. This leads to an increase in the residual as illustrated in the Residual plot.

$$\bar{z}_k = z_k - H_k \hat{x}_{k|k-1} \tag{2.36}$$

Figure 2.8 illustrates the residual $\bar{z}_k$ plotted with respect to the time-frame. Every time there is a fast eye motion, there is a divergence in the residual from its mean zero position. This change in the error value can be used to detect if a maneuver is taking place. We consider a lower process noise to model our Kalman filter and after each prediction step, we compute the residual. If the residual is larger than a pre-initialized residual limit, we increase the process noise. Due to the increased process noise, the kalman filter starts to favor the measured eye positions over the process

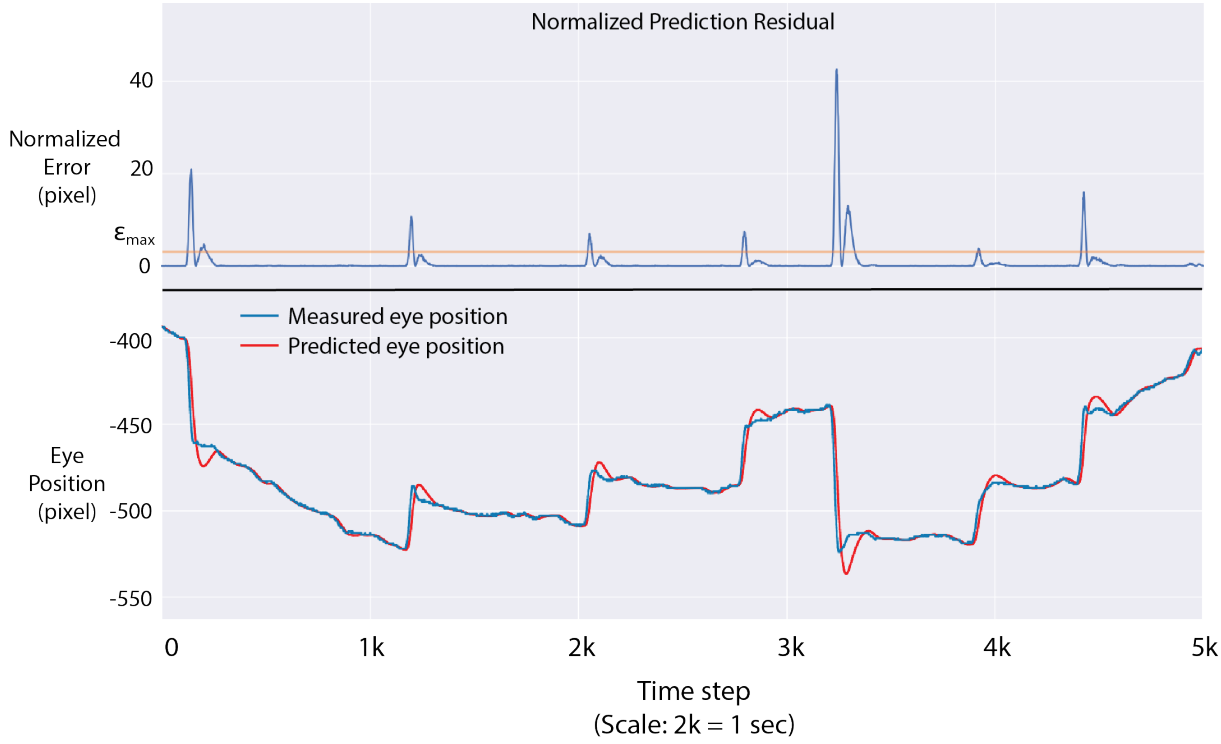


**Figure 2.9: Normalized residual:** The top graph is the normalized residual plot $\epsilon$. The orange line represents the $\epsilon_{max}$ we choose for our filter. When $\epsilon \leq \epsilon_{max}$, we consider a smaller process noise. For $\epsilon > \epsilon_{max}$, we scale up the process noise parameter.

prediction and the filter will track the signal closely [36]. Once the residual becomes smaller, we will scale back to the lower process noise.

We use a continuous adjustment method [43] where we first normalize the square of the residual using the following equation:

$$\epsilon = z_k^T S^{-1} z_k \tag{2.37}$$

Where $z_k$ is the residual and $S$ is the system uncertainty that is computed using equation 2.20. By squaring the residual we can ensure that the error is always greater than zero and normalizing by the measurement covariance scales the signal as illustrated in Figure 2.9. We will start to scale up our process noise $Q$ once the value of $\epsilon$ exceeds a certain limit say $\epsilon_{max}$ (the orange line in Figure 2.9). When $\epsilon$ falls below $\epsilon_{max}$, we bring the process noise back to its initial value. The value of $\epsilon_{max}$ can be chosen according to the observations and the experimental requirements. Usually, once

the residual is equal to 5 standard deviations, we scale up the process noise [43]. There is no single correct value for $\epsilon_{max}$ and one could chose a value that best matches the required performance. The algorithm for the Adaptive filter model is shown in Algorithm 6.

**Algorithm 6:** Adaptive Kalman Filter Model Algorithm

$N$ = Measured eye position; ▷ $N$ is a row vector with all the measured eye positions;

$x_k = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix;

$P_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix;

$\sigma_x = 0.1$;

$\sigma_l = 8$;

$R = \sigma_l^2$; ▷ Measurement noise covariance;

$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$;

scale factor = 1;

$\epsilon_{max} = 5$;

**for** *i in range len(N)* **do**

  **for** $\Delta t$ *in range 20* **do**

    $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$; ▷ Transition Matrix;

    $\sigma_x = \sigma_x \times (scalefactor)$; ▷ Process noise adjustment;

    $Q_{k-1} = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 \\ 1/2(\Delta t)^3 & \Delta t^2 \end{bmatrix} \sigma_x^2$; ▷ Process Noise covariance matrix;

    $\hat{x}_k = Ax_k$; ▷ The predict state;

    $\hat{P}_k = AP_kA^T + Q$; ▷ The predict covariance;

    **if** $\Delta t == 1$ **then**

      $\hat{x}_0 = \hat{x}_k$;

      $\hat{P}_0 = \hat{P}_k$;

    **end**

  **end**

  $z_k = N[i]$; ▷ New measured eye position;

  $\bar{z}_k = z_k - H\hat{x}_0$; ▷ Residual;

  $S = H\hat{P}_0(H)^T + R$; ▷ System uncertainty;

  $K = \frac{H^T\hat{P}_0}{S_k}$; ▷ Kalman gain;

  $x_k = \hat{x}_0 + K\bar{z}_k$; ▷ The estimate state;

  $P_k = \hat{P}_0 - KSK^T$; ▷ The estimate covariance;

  $\epsilon = z_k^T S^{-1} z_k$; ▷ Normalized Residual;

  **if** $\epsilon \leq \epsilon_{max}$ **then**

    scale factor = 1;

  **else**

    scale factor = 100;

  **end**

**end**

#### *2.3.2.4 Multiple Model Adaptive Estimator*

The idea of building a Multiple Model Adaptive Estimator (MMAE) is to use banks of Kalman filters to output an overall better estimation. The MMAE algorithm uses multiple models at a time to match the uncertain motion of the eyes [38]. Unlike the adaptive filter technique, here we choose, either the prediction, or measurement of a single filter, but we use a blend of two filters in proportion to their likelihoods [36].

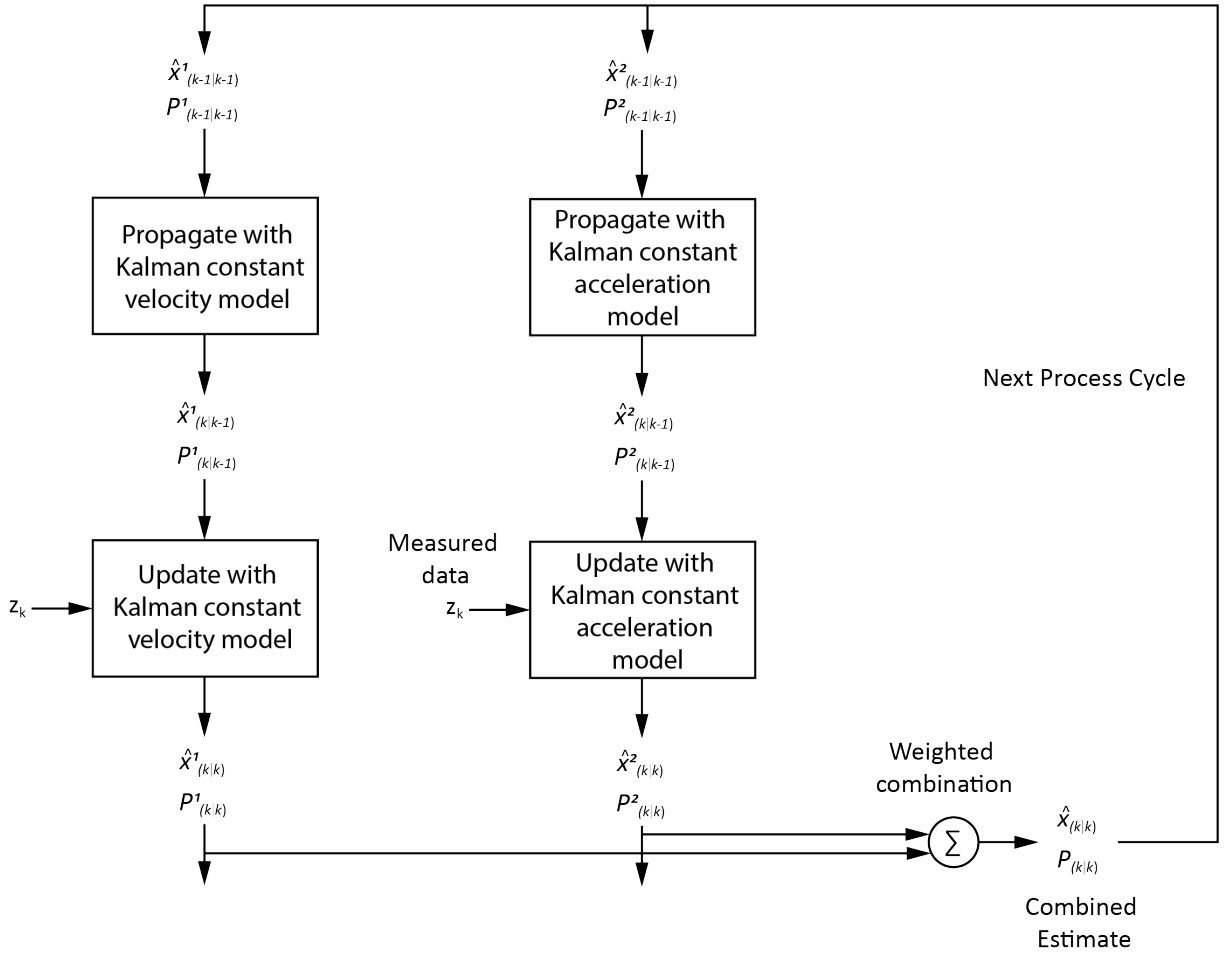


**Figure 2.10: Block Diagram of Multiple Model Adaptive Estimator [44]:**

Figure 2.10 illustrates the block diagram of the MMAE [44]. We initialize the constant velocity and the constant acceleration Kalman filters and then run them together in a predict and update loop. At each update step we use Equation 2.31 and 2.32 to compute the residual $\bar{z}_k$ and the system uncertainty $S_k$ for each filter. We then compute the likelihood function for both filters using the

equation:

$$L_k = \frac{1}{\sqrt{2\pi S_k}} \exp\left[\frac{1}{2}\bar{z}_k^T S_k \bar{z}_k\right] \tag{2.38}$$

In statistics, the likelihood function measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters [45]. In our case, we compute the likelihood function for the Kalman constant velocity model and the Kalman constant acceleration model. If the residual is large, then the likelihood function gives us a low value, suggesting that our algorithm has failed to correctly predict the eye motion [36]. We can use this to compute the probability that each filter is the best fit to the data. For N filters, we can compute the probability that the $i_{th}$ filter is correct in relation to the other filters.

$$\mu_k^i = \frac{L_k^i \mu_{k-1}^i}{\sum_{j=1}^{N} L_k^j \mu_{k-1}^j} \tag{2.39}$$

Here, the numerator is the likelihood function of a filter multiplied by the previously computed probability of that filter and the denominator is used to normalize all the filter probabilities to sum to one. Since we take a bank of two filters, in our case, $N$=2. We set an initial probability $\mu_{k-1}^i$ to 0.5 for each filter. We can then compute the estimated state as the sum of the state from each filter multiplied by the the probability of that filter being correct.

$$\hat{x}_{k|k} = \sum_{i=1}^{2} \hat{x}_{k|k}^i \mu_k^i \tag{2.40}$$

$$P_{k|k} = \sum_{i=1}^{2} [P_{k|k}^i + (\hat{x}_{k|k} - \hat{x}_{k|k}^i)(\hat{x}_{k|k} - \hat{x}_{k|k}^i)^T]\mu_k^i \tag{2.41}$$

Where we compute $\hat{x}_{k|k}^i$ and $P_{k|k}^i$ using the Equation 2.34 and 2.35 respectively for the two filters. The combined estimate $\hat{x}_{k|k}$ and the combined process uncertainty $P_{k|k}$ are then used as the initial state $\hat{x}_{k-1|k-1}^i$ and process covariance matrix $P_{k-1|k-1}^i$ by each filter for the next predict and update cycle. The Algorithm 7 gives the complete algorithm for the MMAE model.

**Algorithm 7:** Multiple Model Adaptive Estimator Model Algorithm

$N$ = Measured eye position; ▷ $N$ is a row vector with all the measured eye positions;

$x_k(cv) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix for KFCV model;

$P_k(cv) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix for KFCV model;

$x_k(ca) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix for KFCA model;

$P_k(ca) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix for KFCA model;

$\sigma_x = 0.1$;

$\sigma_l = 8$;

$R = \sigma_l^2$; ▷ Measurement noise covariance;

$H(cv) = \begin{bmatrix} 1 & 0 \end{bmatrix}$;

$H(ca) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$;

**for** *i in range len(N)* **do**

  **for** $\Delta t$ *in range 20* **do**

    $A(cv) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$; ▷ Transition Matrix for KFCV model;

    $Q(cv) = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 \\ 1/2(\Delta t)^3 & \Delta t^2 \end{bmatrix} \sigma_x^2$; ▷ Process Noise covariance matrix for KFCV;

    $A(ca) = \begin{bmatrix} 1 & \Delta t & \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Transition matrix for KFCA model;

    $Q(ca) = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 & 1/2(\Delta t)^2 \\ 1/2(\Delta t)^3 & (\Delta t)^2 & \Delta t \\ 1/2(\Delta t)^2 & \Delta t & 1 \end{bmatrix} \sigma_x^2$;

    $\hat{x}_k(cv) = A(cv)x_k(cv)$;

    $\hat{P}_k(cv) = A(cv)P_k(cv)A(cv)^T + Q(cv)$;

    $\hat{x}_k(ca) = A(ca)x_k(ca)$;

    $\hat{P}_k(ca) = A(ca)P_k(ca)A(ca)^T + Q(ca)$;

    **if** $\Delta t == 1$ **then**

      $\hat{x}_0(cv) = \hat{x}_k(cv)$;

      $\hat{P}_0(cv) = \hat{P}_k(cv)$;

      $\hat{x}_0(ca) = \hat{x}_k(ca)$;

      $\hat{P}_0(ca) = \hat{P}_k(ca)$;

    **end**

  **end**

**end**

▷ Continue the for loop;;

**for** *i in range len(N)* **do**

$z_k = N[i]$;

$\bar{z}_k(cv) = z_k - H(cv)\hat{x}_0(cv)$;

$S(cv) = H(cv)\hat{P}_0(cv)(H(cv))^T + R$;

$K(cv) = \frac{H(cv)^T \hat{P}_0(cv)}{S_k(cv)}$;

$x_{k0}(cv) = \hat{x}_0(cv) + K(cv)\bar{z}_k(cv)$;

$P_{k0}(cv) = \hat{P}_0(cv) - K(cv)S(cv)K^T(cv)$;

$\bar{z}_k(ca) = z_k - H(ca)\hat{x}_0(ca)$;

$S(ca) = H(ca)\hat{P}_0(ca)(H(ca))^T + R$;

$K(ca) = \frac{H(ca)^T \hat{P}_0(ca)}{S_k(ca)}$;

$x_{k0}(ca) = \hat{x}_0(ca) + K(ca)\bar{z}_k(ca)$;

$P_{k0}(ca) = \hat{P}_0(ca) - K(ca)S(ca)K^T(ca)$;

$L(cv) = \frac{1}{\sqrt{2\pi S(cv)}} \exp[\frac{1}{2}\bar{z}_k(cv)^T S(cv)\bar{z}_k(cv)]$;

$L(ca) = \frac{1}{\sqrt{2\pi S(ca)}} \exp[\frac{1}{2}\bar{z}_k^T(ca)S(ca)\bar{z}_k(ca)]$;

$\mu_k(cv) = \frac{L(cv)\mu_{k-1}(cv)}{L(cv)\mu_{k-1}(cv)+L(ca)\mu_{k-1}(ca)}$;

$\mu_k(ca) = 1 - \mu_k(cv)$;

$x_k = x_{k0}(cv) \times \mu_k(cv) + x_{k0}(ca) \times \mu_k(ca)$; ▷ New combined estimate state;

$P_k = \mu_k(cv).(P_{k0}(cv) + (x_k - x_{k0}(cv).(x_k - x_{k0}(cv))^T) + \mu_k(ca).(P_{k0}(ca) + (x_k - x_{k0}(ca).(x_k - x_{k0}(ca))^T)$; ▷ New combined covariance;

$x_k(cv) = x_k(ca) = x_k$;

$P_k(cv) = P_k(ca) = P_k$;

**end**

**Table 2.1:** One Cycle of MMAE Algorithm

| Model-conditioned filtering: | |
|---|---|
| Predicted state: | $\hat{x}^i_{k\|k-1} = A^i_{k-1}\hat{x}^i_{k-1\|k-1}$ |
| Predicted covariance: | $P^i_{k\|k-1} = A^i_{k-1}P^i_{k-1\|k-1}(A^i_{k-1})^T + Q^i_{k-1}$ |
| Measurement Residual: | $\bar{z}^i_k = z_k - H^i_k\hat{x}^i_{k\|k-1}$ |
| Covariance Residual: | $S^i_k = H^i_k P^i_{k\|k-1}(H^i_k)^T + R^i_k$ |
| Filter Gain: | $K^i_k = P^i_{k\|k-1}(H^i_k)^T(S^i_k)^{-1}$ |
| Update state: | $\hat{x}^i_{k\|k} = \hat{x}^i_{k\|k-1} + K^i_k\bar{z}^i_k$ |
| Update covariance: | $P^i_{k\|k} = P^i_{k\|k-1} - K^i_k S^i_k(K^i_k)^T$ |
| **Model probability update:** | |
| Model likelihood: | $L^i_k = \frac{1}{\sqrt{2\pi S^i_k}}\exp[\frac{1}{2}(\bar{z}^i_k)^T S^i_k \bar{z}^i_k]$ |
| Model probability: | $\mu^i_k = \frac{L^i_k\mu^i_{k-1}}{\sum_{j=1}^{N} L^j_k\mu^j_{k-1}}$ |
| **Estimate Fusion:** | |
| Overall estimate: | $\hat{x}_{k\|k} = \sum_{i=1}^{2}\hat{x}^i_{k\|k}\mu^i_k$ |
| Overall covariance: | $P_{k\|k} = \sum_{i=1}^{2}[P^i_{k\|k} + (\hat{x}_{k\|k} - \hat{x}^i_{k\|k})(\hat{x}_{k\|k} - \hat{x}^i_{k\|k})^T]\mu^i_k$ |

#### *2.3.2.5 Interactive Multiple Model Estimator*

The Interactive Multiple Model (IMM) Estimator was original proposed by Bloom in [46] to solve the combinatorial explosion problem of multiple models. A subsequent paper [47] is the most cited paper on the IMM algorithm. The IMM Estimator is one of the most cost-effective class of estimators for tracking maneuvering targets using different filter models. In order to give us a satisfactory prediction with eye movements, the tracking algorithm needs to be capable to detect the beginning and the end of maneuvers. The IMM estimator is able to achieve this by computing and updating the mode probabilities based on received data. As a result of this adaptive capability, that is the ability to switch between different filter models, in the presence or absence of maneuvers, that gives the IMM estimator its advantage over simple estimators like the Kalman filter [48].

For predicting the eye motion, we use the Kalman filter constant velocity and constant acceleration models. When the eyes are fixated or moving slowly, the constant acceleration filter will lag the

signal, while the constant velocity filter is able to give us a good estimate. The likelihood function of each model, tells us which filter is most probable. The constant velocity filter will have a higher likelihood in comparison to the constant acceleration filter, so we adjust the constant acceleration filter estimate with the constant velocity model estimate. Once the eye motion starts to accelerate during saccades and microsaccades, the constant velocity filter estimate starts to lag while the constant acceleration model is able to give us a better estimation. Since we had revised the constant acceleration models estimate with the constant velocity estimate and within a few epochs, it will be producing an adequate estimation.

We first initialize a set of model probabilities for each of the systems: $\mu = \{$constant velocity, constant acceleration$\}$ [36]:

$$\mu_{0|0} = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} \tag{2.42}$$

Here, $\mu_{0|0}$ gives us a vector of mode probabilities for each probability mode. Initially we assume that 70% of the eye motions are fixational or slow velocity eye movements and the other 30% comprises of fast movements. The mode probability matrix can be represented by:

$$\mu^i_{k-1} = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix} \tag{2.43}$$

We next consider the model transitions as a Markov chain as illustrated in Figure 2.11 where we consider the eyes to move with different motions, thereby transitioning between the two modes. We represent the Markov chain illustrated in Figure 2.11 with a transition probability matrix:

$$\pi_{ji} = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix} \tag{2.44}$$

The model transition matrix and the initialized mode probability matrix together are used to compute the new mode probabilities:

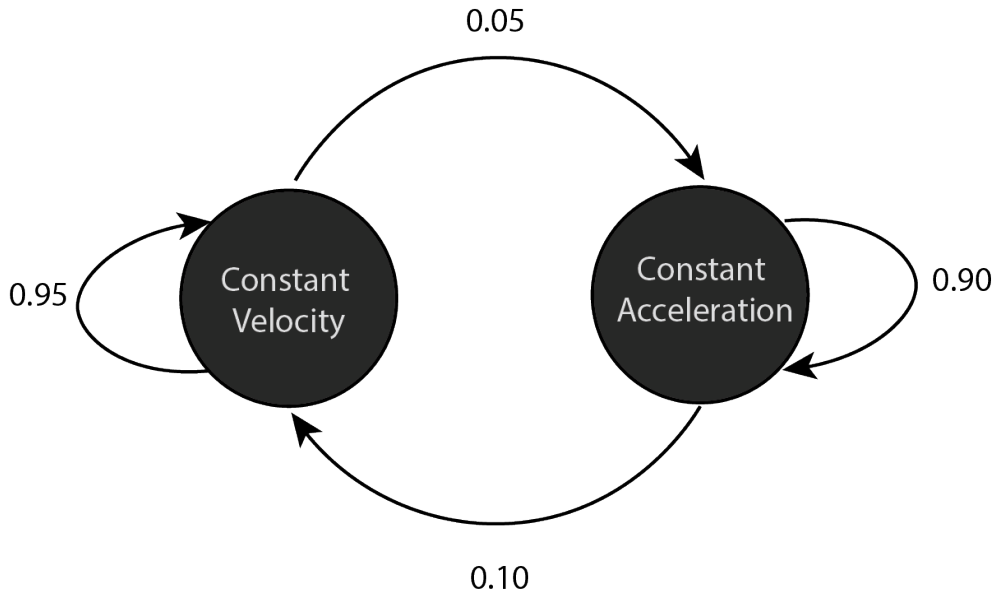


**Figure 2.11: A Markov Chain:** Illustrates two modes for the eye motions, one mode is the constant velocity and the other is the constant acceleration. If the current eye motion is with a constant velocity, we predict that there is a 95% chance that it stays in the same motion and a 5% chance that the eyes start to move with an accelerated motion. Once the eyes start to move in a varying speed, we predict that there is a 90% chance that it stays in the same motion, and a 10% chance of returning to a constant velocity.

$$\mu^i_{k|k-1} = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix} \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} \tag{2.45}$$

$$\mu^i_{k|k-1} = \sum_{j=1}^{2} \pi_{ji} \mu^i_{k-1} \tag{2.46}$$

Next, we compute the model probability using the Bayes theorem:

$$\mu^{j|i}_{k-1} = \frac{\pi_{ji} \mu^j_{k-1}}{\mu^i_{k|k-1}} \tag{2.47}$$

The IMM estimator utilizes the mixing probabilities to interact between the different models. Instead of having the Kalman filter to compute its state as the weighted average of all the filters in the filter bank, the filter having the higher probability during the current eye motion gets weighted more than the other filter. As a result, the information from the more probable filter is able to

improve the accuracy of the other filter at each epoch. As a result of the interaction between the filters, the likely filter will be slightly adjusted by the unlikely filter, and the unlikely filter will be strongly adjusted by the likely one. The mixed estimate and the mixed covariance is computed using the following equations [36]:

$$\bar{x}^i_{k-1|k-1} = \sum_j \hat{x}^j_{k-1|k-1} \mu^{j|i}_{k-1} \tag{2.48}$$

$$\bar{P}^i_{k-1|k-1} = \sum_j [P^j_{k-1|k-1} + (\bar{x}^i_{k-1|k-1} - \hat{x}^j_{k-1|k-1})(\bar{x}^i_{k-1|k-1} - \hat{x}^j_{k-1|k-1})^T] \mu^{j|i}_{k-1} \tag{2.49}$$

The mode probabilities describe the current probability of each mode, while the transition probabilities describe how likely we are to change modes. The Kalman filters perform the prediction step to compute the new prior, making use of the mixed estimate $\bar{x}^i_{k-1|k-1}$ and mixed covariance $\bar{P}^i_{k-1|k-1}$ using equations 2.29 and 2.30. Then we perform the update step, making use of the new measured data $z_k$ to get the new estimates using equations 2.34 and 2.35. We then compute the model likelihood and the model probability for the two filters, making use of the residual $\bar{z}^i_k$ and system uncertainty $S^i_k$:

$$L^i_k = \frac{1}{\sqrt{2\pi S^i_k}} \exp\left[\frac{1}{2}(\bar{z}^i_k)^T S^i_k \bar{z}^i_k\right] \tag{2.50}$$

$$\mu^i_k = \frac{L^i_k \mu^i_{k-1}}{\sum_{j=1}^{N} L^j_k \mu^j_{k-1}} \tag{2.51}$$

Making use of the individual model probability, now we can get the overall estimate and the overall covariance of the IMM estimator:

$$\hat{x}_{k|k} = \sum_{i=1}^{2} \hat{x}^i_{k|k} \mu^i_k \tag{2.52}$$

$$P_{k|k} = \sum_{i=1}^{2} [P^i_{k|k} + (\hat{x}_{k|k} - \hat{x}^i_{k|k})(\hat{x}_{k|k} - \hat{x}^i_{k|k})^T]\mu^i_k \tag{2.53}$$

For the next cycle, the new model probability $\mu^i_k$ now becomes our prior model probability $\mu^i_{k-1}$ and using our Markov transition matrix, we can now compute the new predicted model probability using equation 2.45 and new mixing probabilities using equation 2.46. Once we have our prior overall estimate $\hat{x}_{k-1|k-1}$ and covariance matrix $P_{k-1|k-1}$, and the new mixing probabilities $\mu^i_{k-1}$, we calculate our mixed estimates and mixed covariance for the two models using equations 2.48 and 2.49.

The block diagram of an IMM estimator is illustrated in the Figure 2.12 [44]. The complete algorithm used for designing our IMM estimator model is given below (See Algorithm 8),

**Algorithm 8:** Interactive Multiple Model Model Algorithm

$N$ = Measured eye position; ▷ $N$ is a row vector with all the measured eye positions;

$x_k(cv) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix for KFCV model;

$P_k(cv) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix for KFCV model;

$x_k(ca) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$; ▷ Initial state matrix for KFCA model;

$P_k(ca) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Initial covariance matrix for KFCA model;

$\sigma_x = 0.1$;

$\sigma_l = 8$;

$R = \sigma_l^2$; ▷ Measurement noise covariance;

$H(cv) = \begin{bmatrix} 1 & 0 \end{bmatrix}$;

$H(ca) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$;

$\mu_k = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$

$\pi = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix}$; ▷ The Markov transition matrix;

**for** *i in range len(N)* **do**

  $\mu_{k|k-1} = \mu_k \pi$; ▷ Predicted model probability;

  **for** *m in range len(2)* **do**

    **for** *n in range len(2)* **do**

      $(\mu_{k-1})_{m,n} = (\pi_{m,n} * (\mu_{k|k-1})_m)/(\mu_{k|k-1})_n$; ▷ Mixing probabilities ($m \times n$ matrix);

    **end**

  **end**

  $\bar{x}_k(cv) = ((\mu_{k-1})_{0,0} \times x_k(cv)) + ((\mu_{k-1})_{1,0} \times x_k(cv))$; ▷ Mixing estimates;

  $\bar{x}_k(ca) = ((\mu_{k-1})_{0,1} \times x_k(ca)) + ((\mu_{k-1})_{1,1} \times x_k(ca))$;

  $\bar{P}_k(cv) = [(\mu_{k-1})_{0,0}(P_k(cv) + (x_k(cv) - \bar{x}_k(cv))(x_k(cv) - \bar{x}_k(cv))^T)] + [(\mu_{k-1})_{1,0}(P_k(ca) + (x_k(ca) - \bar{x}_k(ca))(x_k(ca) - \bar{x}_k(ca))^T)]$;

  $\bar{P}_k(ca) = [(\mu_{k-1})_{0,1}(P_k(cv) + (x_k(cv) - \bar{x}_k(cv))(x_k(cv) - \bar{x}_k(cv))^T)] + [(\mu_{k-1})_{1,1}(P_k(ca) + (x_k(ca) - \bar{x}_k(ca))(x_k(ca) - \bar{x}_k(ca))^T)]$; ▷ Mixing covariance;

**end**

▷ Continue the for loop;;

**for** *i in range len(N)* **do**

  **for** $\Delta t$ *in range 20* **do**

    $A(cv) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$; ▷ Transition Matrix for KFCV model;

    $Q(cv) = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 \\ 1/2(\Delta t)^3 & \Delta t^2 \end{bmatrix} \sigma_x^2$; ▷ Process Noise covariance matrix for KFCV;

    $A(ca) = \begin{bmatrix} 1 & \Delta t & \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$; ▷ Transition matrix for KFCA model;

    $Q(ca) = \begin{bmatrix} 1/4(\Delta t)^4 & 1/2(\Delta t)^3 & 1/2(\Delta t)^2 \\ 1/2(\Delta t)^3 & (\Delta t)^2 & \Delta t \\ 1/2(\Delta t)^2 & \Delta t & 1 \end{bmatrix} \sigma_x^2$;

    $\hat{x}_k(cv) = A(cv)\bar{x}_k(cv)$;

    $\hat{P}_k(cv) = A(cv)\bar{P}_k(cv)A(cv)^T + Q(cv)$;

    $\hat{x}_k(ca) = A(ca)\bar{x}_k(ca)$;

    $\hat{P}_k(ca) = A(ca)\bar{P}_k(ca)A(ca)^T + Q(ca)$;

    **if** $\Delta t == 1$ **then**

      $\hat{x}_0(cv) = \hat{x}_k(cv)$;

      $\hat{P}_0(cv) = \hat{P}_k(cv)$;

      $\hat{x}_0(ca) = \hat{x}_k(ca)$;

      $\hat{P}_0(ca) = \hat{P}_k(ca)$;

    **end**

    $z_k = N[i]$;

    $\bar{z}_k(cv) = z_k - H(cv)\hat{x}_0(cv)$;

    $S(cv) = H(cv)\hat{P}_0(cv)(H(cv))^T + R$;

    $K(cv) = \frac{H(cv)^T\hat{P}_0(cv)}{S_k(cv)}$;

    $x_{k0}(cv) = \hat{x}_0(cv) + K(cv)\bar{z}_k(cv)$;

    $P_{k0}(cv) = \hat{P}_0(cv) - K(cv)S(cv)K^T(cv)$;

    $\bar{z}_k(ca) = z_k - H(ca)\hat{x}_0(ca)$;

    $S(ca) = H(ca)\hat{P}_0(ca)(H(ca))^T + R$;

    $K(ca) = \frac{H(ca)^T\hat{P}_0(ca)}{S_k(ca)}$;

    $x_{k0}(ca) = \hat{x}_0(ca) + K(ca)\bar{z}_k(ca)$;

    $P_{k0}(ca) = \hat{P}_0(ca) - K(ca)S(ca)K^T(ca)$;

    $L(cv) = \frac{1}{\sqrt{2\pi S(cv)}}\exp[\frac{1}{2}\bar{z}_k(cv)^T S(cv)\bar{z}_k(cv)]$;

    $L(ca) = \frac{1}{\sqrt{2\pi S(ca)}}\exp[\frac{1}{2}\bar{z}_k^T(ca)S(ca)\bar{z}_k(ca)]$;

    $\mu_k(cv) = \frac{L(cv)\mu_{k-1}(cv)}{L(cv)\mu_{k-1}(cv)+L(ca)\mu_{k-1}(ca)}$;

    $\mu_k(ca) = 1 - \mu_k(cv)$;

    $x_k = x_{k0}(cv) \times \mu_k(cv) + x_{k0}(ca) \times \mu_k(ca)$; ▷ New combined estimate state;

    $P_k = \mu_k(cv).(P_{k0}(cv) + (x_k - x_{k0}(cv).(x_k - x_{k0}(cv))^T) + \mu_k(ca).(P_{k0}(ca) + (x_k - x_{k0}(ca).(x_k - x_{k0}(ca))^T)$; ▷ New combined covariance;

    $x_k(cv) = x_k(ca) = x_k$;

    $P_k(cv) = P_k(ca) = P_k$;

    $\mu_k = \begin{bmatrix} \mu_k(cv) \\ \mu_k(ca) \end{bmatrix}^T$

  **end**

**end**

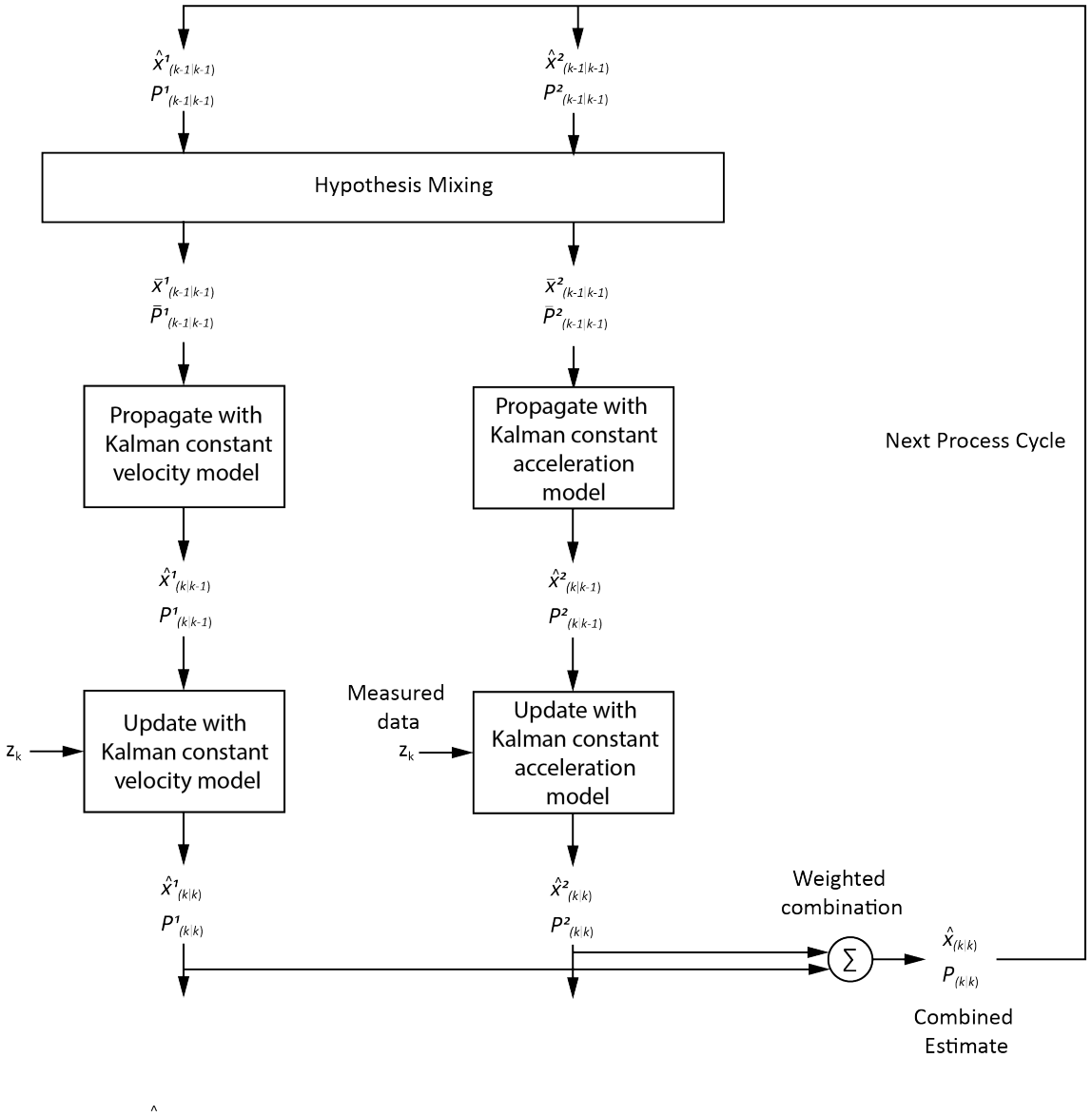


**Figure 2.12: Block Diagram of Interactive Multiple Model Estimator [44]**

**Table 2.2:** One Cycle of the IMM Estimator Algorithm

| | |
|---|---|
| Model-conditioned re-initialization: | |
| Predicted model probability: | $\mu^i_{k\|k-1} = \sum_j \pi_{ji}\mu^j_{k-1}$ |
| Mixing probabilities: | $\mu^{j\|i}_{k-1} = \pi_{ji}\mu^j_{k-1}/\mu^i_{k\|k-1}$ |
| Mixing estimates: | $\bar{x}^i_{k-1\|k-1} = \sum_j \hat{x}^j_{k-1\|k-1}\mu^{j\|i}_{k-1}$ |
| Mixing covariance: | $\bar{P}^i_{k-1\|k-1} = \sum_j [P^j_{k-1\|k-1} + (\bar{x}^i_{k-1\|k-1} - \hat{x}^j_{k-1\|k-1})$ $(\bar{x}^i_{k-1\|k-1} - \hat{x}^j_{k-1\|k-1})^T]\mu^{j\|i}_{k-1}$ |
| Model-conditioned filtering: | |
| Predicted state: | $\hat{x}^i_{k\|k-1} = A^i_{k-1}\bar{x}^i_{k-1\|k-1}$ |
| Predicted covariance: | $P^i_{k\|k-1} = A^i_{k-1}\bar{P}^i_{k-1\|k-1}(A^i_{k-1})^T + Q^i_{k-1}$ |
| Measurement Residual: | $\bar{z}^i_k = z_k - H^i_k\hat{x}^i_{k\|k-1}$ |
| Covariance Residual: | $S^i_k = H^i_k P^i_{k\|k-1}(H^i_k)^T + R^i_k$ |
| Filter Gain: | $K^i_k = P^i_{k\|k-1}(H^i_k)^T(S^i_k)^{-1}$ |
| Update state: | $\hat{x}^i_{k\|k} = \hat{x}^i_{k\|k-1} + K^i_k\bar{z}^i_k$ |
| Update covariance: | $P^i_{k\|k} = P^i_{k\|k-1} - K^i_k S^i_k (K^i_k)^T$ |
| Model probability update: | |
| Model likelihood: | $L^i_k = \frac{1}{\sqrt{2\pi S^i_k}} \exp[\frac{1}{2}(\bar{z}^i_k)^T S^i_k \bar{z}^i_k]$ |
| Model probability: | $\mu^i_k = \frac{L^i_k \mu^i_{k-1}}{\sum_{j=1}^{N} L^j_k \mu^j_{k-1}}$ |
| Estimate Fusion: | |
| Overall estimate: | $\hat{x}_{k\|k} = \sum_{i=1}^{2} \hat{x}^i_{k\|k}\mu^i_k$ |
| Overall covariance: | $P_{k\|k} = \sum_{i=1}^{2} [P^i_{k\|k} + (\hat{x}_{k\|k} - \hat{x}^i_{k\|k})(\hat{x}_{k\|k} - \hat{x}^i_{k\|k})^T]\mu^i_k$ |

## 2.4 QUALITY METRICS

To comprehensively assess the overall performance of the proposed predictor models, several errors for describing the performance are employed as follows:

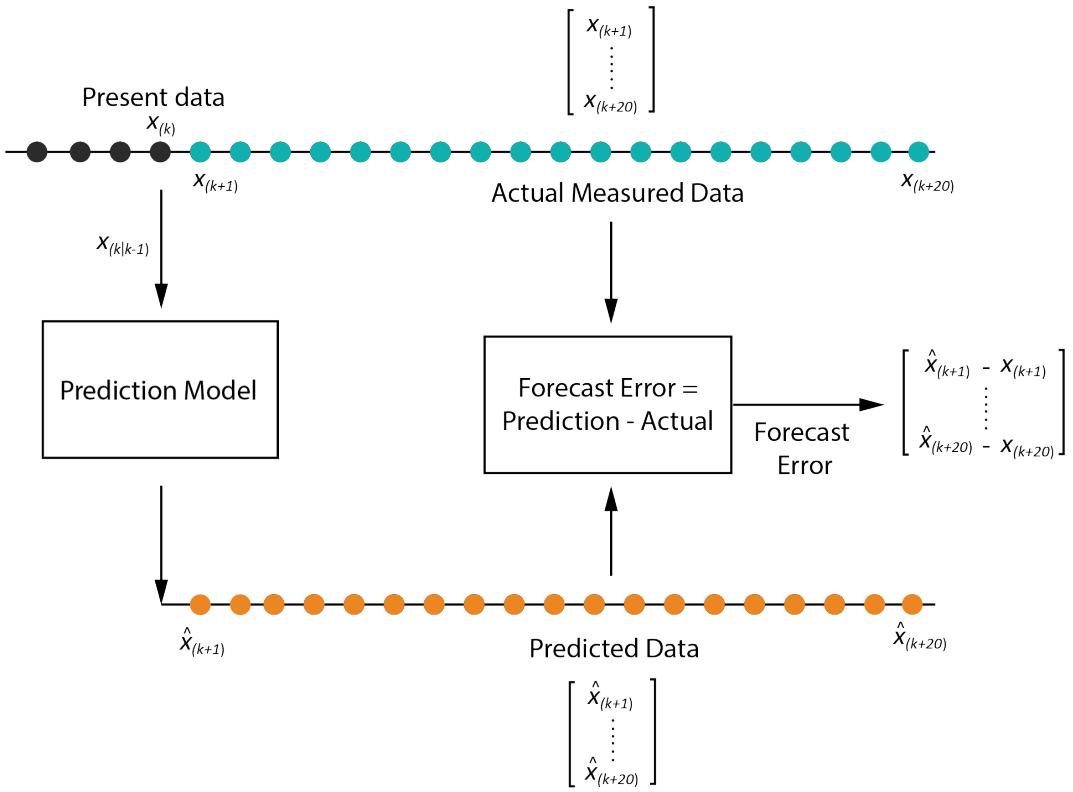


**Figure 2.13: Error Estimation:** At each step, the prediction models predict eye positions for the next 20 frames shown with orange dots. The Error in forecast is computed my subtracting the predicted data with the subsequent 20 measured data giving us a 20 $\times$ 1 error matrix. Therefore, for a total of $N$ steps, we obtain an error matrix of dimension 20 $\times N$.

### 2.4.1 Forecast Errors

A forecast error is the difference between an observed value $x_{k+\Delta t}$ and its forecast $\hat{x}_{k+\Delta t}$ for the time step $k$ as illustrated in Figure 2.13. The equation can be written as:

$$e_{k+\Delta t} = \hat{x}_{k+\Delta t} - x_{k+\Delta t} \tag{2.54}$$

Where $\Delta t$ ranges from 1,2,3,...,20. An ideal prediction model should give us an error of 0 for each

prediction, but in real world this is impossible to achieve.

### 2.4.2 Percentage Accuracy (PA)

We use the percentage accuracy to evaluate the quality of the prediction models. The percentage accuracy metrics gives us the relative accuracy of a models prediction in percentage. We compute the percentage accuracy using the equation:

$$PA_{\Delta t} = \frac{N_{\Delta t}}{N_{Total}} \times 100\% \tag{2.55}$$

Where $\Delta t$ ranges from 1,2,3,...,20. Here, $N_{\Delta t}$ is the total number of predictions with a forecast error equal to zero and the notation $N_{Total}$ is the total number of predictions. As the name suggests, the model with the highest percentage accuracy will be the best suited for predicting the eye movements.

# CHAPTER 3

# EXPERIMENTAL SETUP

## 3.1 SURVEY

### 3.1.1 Apparatus

The crucial parameters required for designing an eye movement predictor system is the eye gaze-position information, accurately tracked with respect to changing time. The TRACKPixx3 video eye tracker (VPixx Technologies, Quebec, Canada) is used to collect the eye data of both the eyes. The tracker is capable of achieving a sampling rate of 2000 Hz with an angular resolution of 0.2°-0.6° while the subjects viewed a 27" display (1920 × 1080 pixel) from 60 cm. A chin rest is used to rest the participants head to eliminate head movements during the experimental procedure. The TRACKPixx3 system is connected to the control PC though a low-latency USB interface and the gaze information is saved in the form of a .csv file.

### 3.1.2 Participants

Nine male participants ranging from 25 to 60 years participated in this experiment with normal or corrected-to-normal vision. Six participants either wore eye glasses or contact lenses during the experiment. Difficulty in accurate calibration of the eye-tracker for a participant wearing contact lens was experienced where the participant was requested to remove the contact lens for the experiment. The participation was voluntary and we obtained a written informed consent from all participants prior to inclusion in the study. All collected data were completely confidential. Due to the Covid19 pandemic situation, all individuals present in the laboratory had to wear face masks and all equipment were disinfected before and after the experiment.

### 3.1.3 Procedure and Task

The participant rested their head on the chin rest, viewing a console monitor placed at a distance of 60 cm away from the eyes. The position and focus of the TRACKPixx3 eye-tracker is adjusted to track both eyes. The PyPixx software provided by VPixx Technology is used for the calibration process. The participants have to visually track a succession of 13 white targets that appear one

after the other on a black background. The calibration is then validated by a gaze position indication stimuli and if unsuccessful, the process is repeated. Once the tracker is successfully calibrated, we start with our experiment.

During the first tracking session, the participants are instructed to follow the cursor on the screen, placed in front of them and their gaze movements are recorded. The cursor is moved on a dark background in a way that different types of eye motions are covered. First we start by moving the cursor slowly to record the fixational eye movements, followed by fast moving cursor that would induce saccades. Each stimulation lasts for 20-30 seconds and the eye gaze positions are recorded. The second tracking session is a natural viewing of a high-definition video, where the participants are presented with a city walk stimulus. The participants are instructed to visually inspect the the stimulation without any constrains for a period of 10-15 secs and their eye gaze positions are recorded for further analysis and development of the prediction models.

### 3.1.4 Pre-processing

A total of 351.5 seconds (703k frames) of eye position data were collected from the nine participants and used for analysing the prediction models. From the recorded data sheet we extract the timestamp and the corresponding right/left eye, horizontal position for developing our model. All the prediction models are built in a way that both, horizontal as well as vertical movements are predicted using the same algorithm.

## 3.2 TEST ENVIRONMENT

A total of eight prediction models have been built using different algorithms as discussed in our previous section. The linear regression and polynomial regression models requires the predictive model to be trained first using prior data, and the model uses the trained knowledge to predict the future eye positions. The probabilistic predictor models are built using algorithm that do not require training data. After each prediction, the model compares its predictions with the actual measurement to converge towards an accurate prediction.

The performance of each prediction models are analysed for the complete collected data. We will also examine the individual model performance exclusively during saccades and fixations. For this,

we will utilize the saccade and fixation flags which are produced automatically by the TRACKPixx3 eye-tracker to separate the two types of eye movements.

The video eye tracker is capable of recording at a sampling rate of 2000 Hz but the stimulus generator can work on a different sampling rate. For this, we down sample the data and analyze the performance of our prediction models for different sampling rates.

## 3.3 PLOTS

The plot in the Figure 3.1 is used to visually analyze the prediction quality of the different predictor models in application to gaze positions on the screen. The interpretation of the plots are as follows:

- Plot A is a 2D Heat map along with the contour lines displaying the error distribution over the magnitude of forecast error in pixels and the prediction steps in future. An ideal prediction algorithm should give us a distribution which would only have the central zero error row populated. Therefore the wider the distribution, the poorer is the model prediction.
- The plot B provides the histogram of the number of eye positions over a varying magnitude of the forecast error for a fixed prediction step. For each prediction algorithm, we will plot the histogram for different intervals of prediction steps and analyze the results. An ideal predictor should only have the central, zero error line populated for each fixed prediction step. The vertical red dotted line in plot A represents the axis along which, this histogram is plotted.
- The plot C represents the histogram of the number of eye positions over varying prediction steps for a fixed error magnitude. The horizontal red dotted line in plot A represents the axis along which, this histogram is plotted.

As seen in Figure 3.1, the difference between the maximum and minimum is large. To respond to the skewness toward large values we use a logarithmic scale. In Figure 3.2 the plots display the wide range of values in a compact way for better visualization of our results

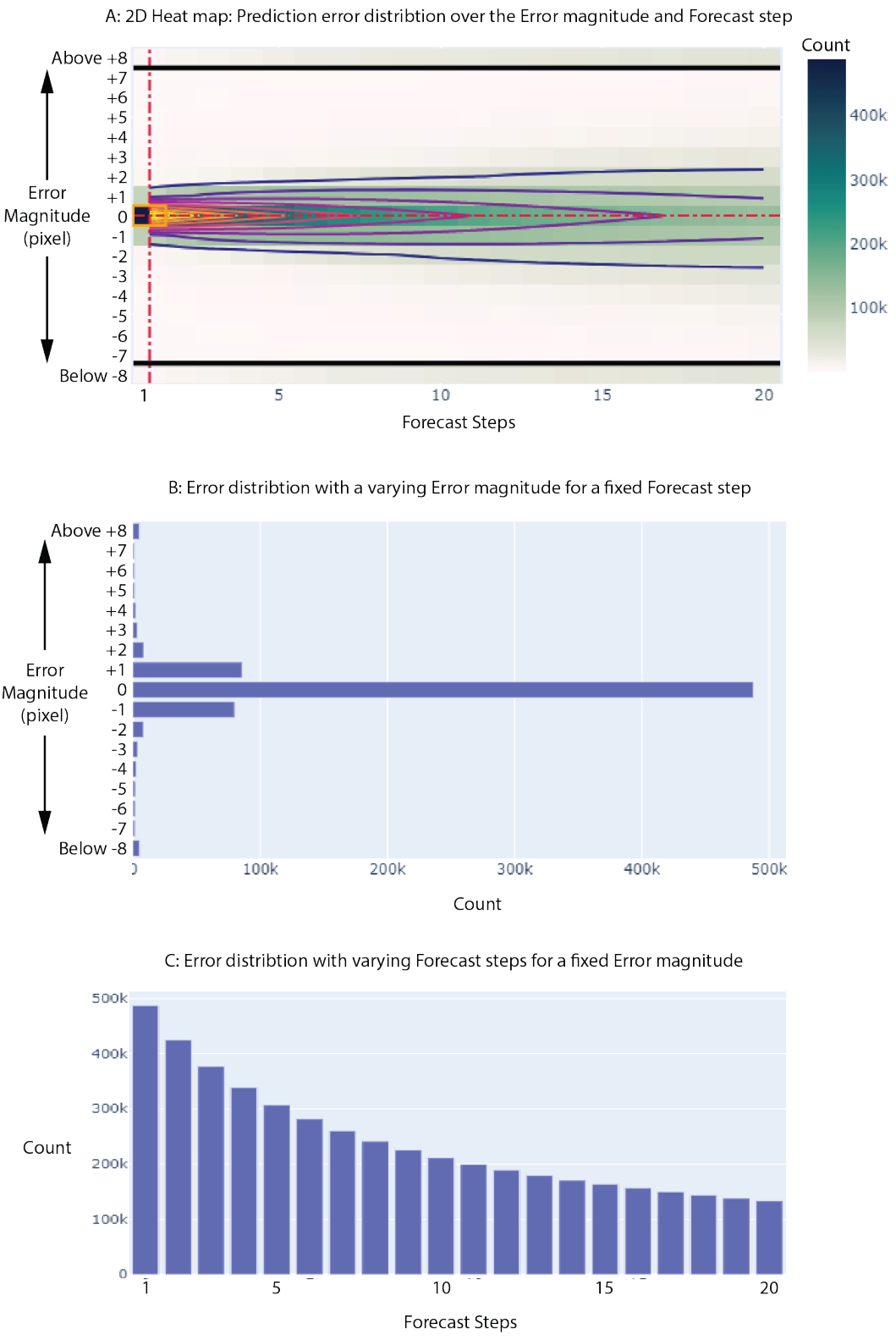


**Figure 3.1: Plot for analyzing the prediction quality (linear scale)**

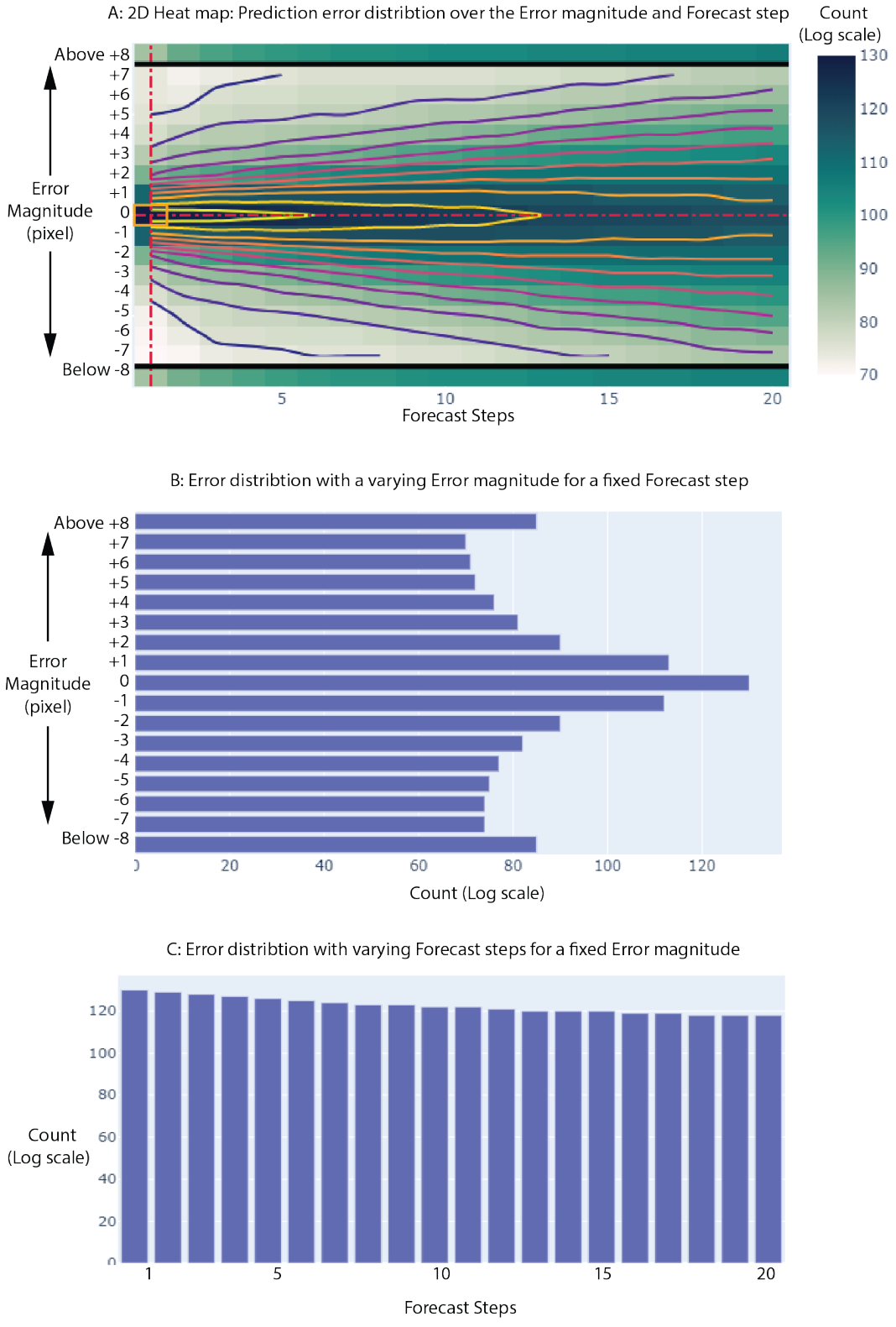


**Figure 3.2: Plot for analyzing the prediction quality (logarithmic scale)**

# CHAPTER 4

# RESULTS

## 4.1 DELTA CODING TECHNIQUE

To evaluate the prediction models, we divide the results into three major parts.

- The Figure 4.1 shows the distribution of the error with respect to the varying prediction steps and the error magnitude. Here, Figure 4.1 (A) is plotted on a linear scale and the Figure 4.1 (B) is plotted on a logarithmic scale. The contour lines gives us an idea of the distribution of the error for increasing prediction steps. For a forecasting model normally we expect that as the prediction steps increases, the prediction accuracy decreases. This is evident by looking at the plots where we observe that the contour lines widen as the prediction steps increases.

- Now in Figure 4.2, we compare the percentage prediction accuracy of our Delta coding algorithm for different prediction steps, increasing from Figure 4.2 (A-E). We evaluate our model for the data collected from all nine participants. Figure 4.2 (F) shows the total prediction accuracy (percentage) of the Delta coding algorithm using all the participants collected data. It also showcases the normal trend of the prediction model where the prediction accuracy keeps falling down for an increasing number of prediction steps.

  We observe here that for the first prediction step in plot (A), the delta coding algorithm is able to accurately predict up to 69% of the total predictions. However, the prediction accuracy falls to 43% when the algorithm is predicting five steps ahead. The prediction accuracy further keeps decreasing as the number of of prediction steps increases. We observe a 30%, 23% and 19% forecast accuracy for the tenth, fifteenth and twentieth respectively.

- In Figure 4.3, we evaluate the forecast accuracy of the model with respect to different parameters. Figure 4.3 (A) compares the forecast accuracy of the Delta coding algorithm for different error ranges. The blue bars represent the zero error range while the orange bar considers an error ranging from +2 pixels to -2 pixels. Taking an error deviation of $\pm$ 2 pixel, would give us a clearer idea of our model performance. When we consider the possible measurement error due to the angular resolution of our TRACKPixx3 eye-tracker, a deviation of $\pm$ 2 pixel range can be considered an accurate prediction. Considering the error range from +2 to -2 pixels,

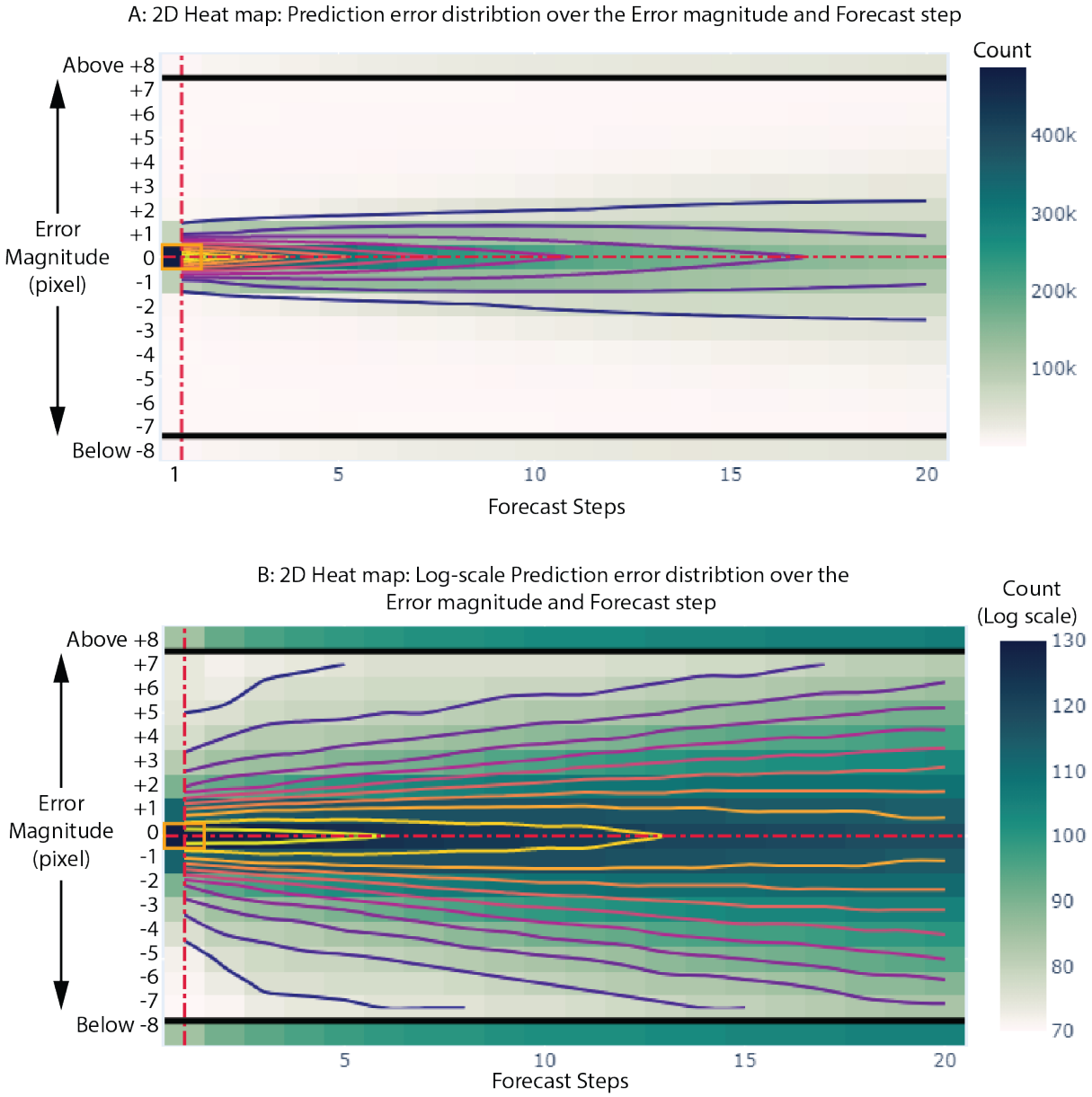


**Figure 4.1: Delta Coding Technique:** A 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps: **A** Linear scale, **B** Logarithmic scale

we get a prediction accuracy of up to 96% for prediction step one. As the prediction steps increases, we observe a steady fall in the prediction accuracy and for the twentieth prediction step, 66% of all the predictions, lie between an error range of +2 and -2 pixels.

Plot 4.3 (B) compares the percentage forecast accuracy of our algorithm for the different eye movements. Here the first bar considers the complete data, the second bar only considers

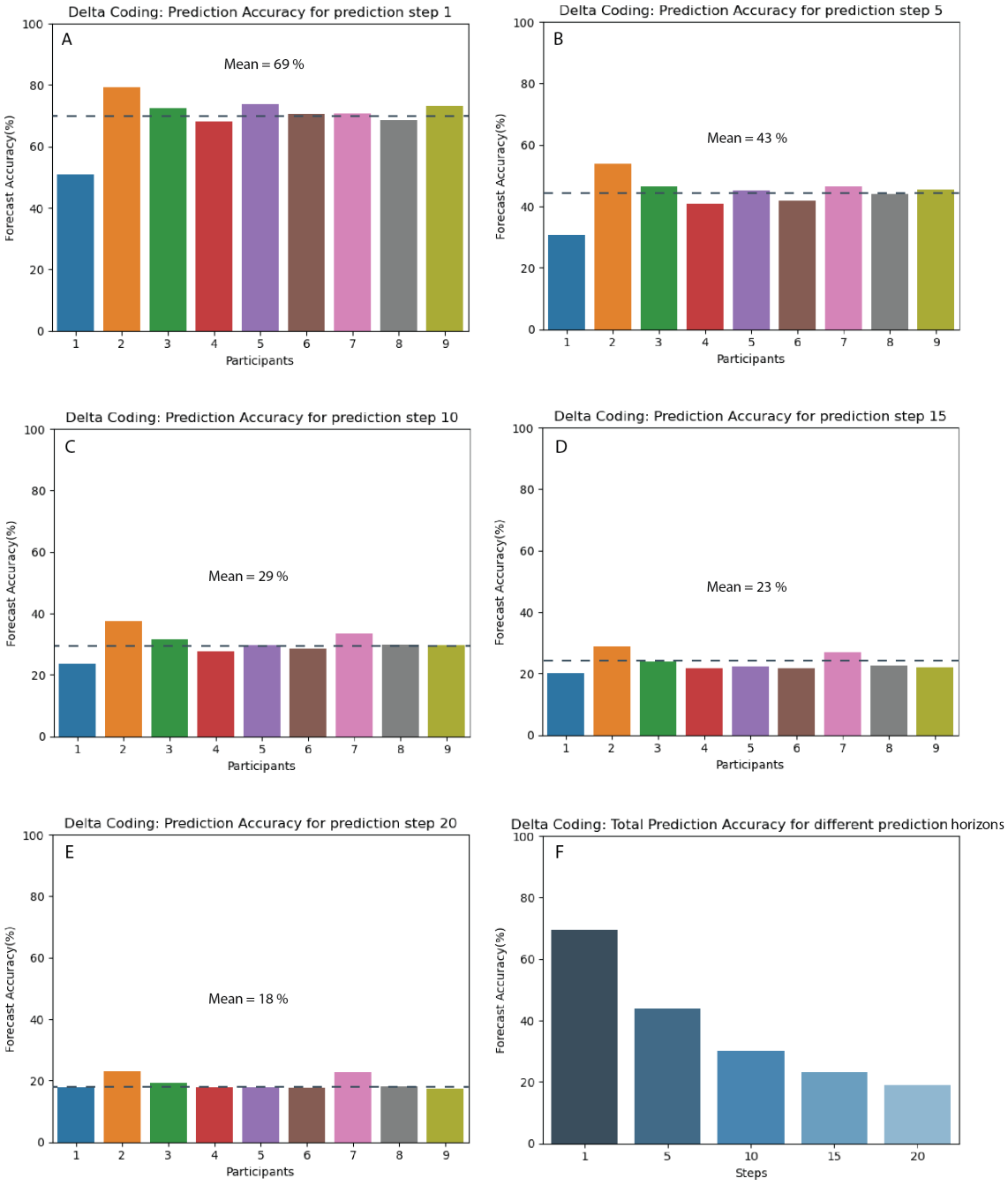


**Figure 4.2:** The plot illustrates the percentage forecast accuracy for the Delta coding algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

the forecast accuracy for the fixation eye movements and the third bar examines the forecast

accuracy for saccade eye movements. We observe the difference in the model performance while predicting saccade eye movements using this algorithm. For the first prediction step, the model predicts with an accuracy of 72% for all fixation eye movements and 5% for the saccade motion.

In the Plot 4.3 (C), we compare our model performance for the different sampling rates. We choose three different sampling rates for evaluating the effects of down sampling on our model performance. A visible decline in the model performance can be observed for a lower sampling rate. The forecast accuracy falls from 69% to 61% and 19% to 11% by changing the sampling rate from 2000 Hz to 1000 Hz for the first prediction step and twentieth prediction step respectively.

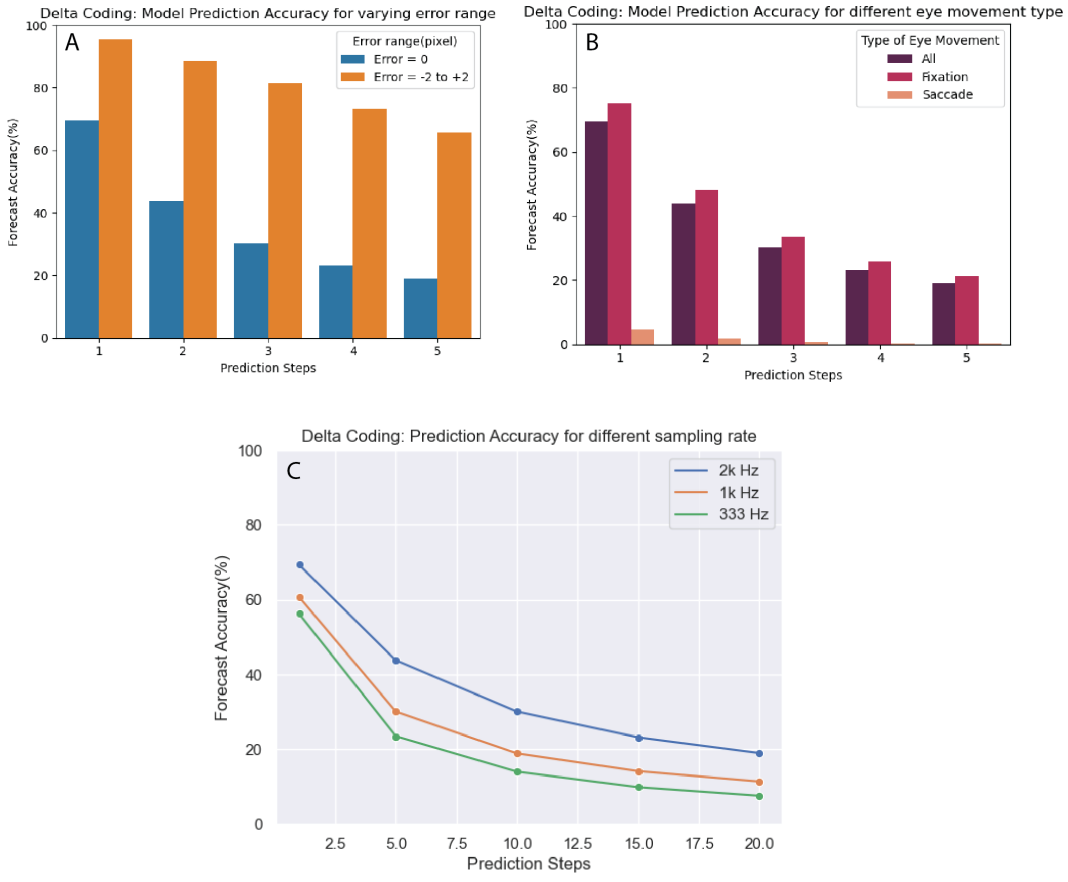


**Figure 4.3:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Delta coding algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.2 LINEAR REGRESSION

- Figure 4.4 illustrates the error distribution for the Linear regression model. The contour lines shows an evenly distributed error on either sides of the zero error line, widening continuously as the prediction steps increase.

- In Figure 4.5 (A-E) we evaluate the percentage forecast accuracy of our Linear regression model for the individual participants for the different prediction steps. Plot (F) shows the prediction accuracy taking the total data of all the participants with increasing prediction steps. As the prediction steps increases, the model accuracy decreases. For the first prediction step, the model is accurately able to predict 72% of the total data giving us a zero error. For the fifth prediction step the prediction accuracy falls to 51%. We obtain a prediction accuracy of 37%, 29% and 24% for the tenth, fifteenth and twentieth prediction step respectively.

- Now in Figure 4.6 (A) we compare the Linear regression models forecast accuracy for an error bandwidth of +2 to -2 pixels. We observe that 88.5% of the predictions lie between this error range for the first prediction step. For the twentieth prediction step, 66.42% of the predictions lie within an error bandwidth of $\pm 2$ pixels.

  In Plot (B), we observe that the model can accurately predict 81% of fixation eye movements but it performs poorly, when it comes to predicting saccade motions. The model could only predict 2% of the saccade eye movements for the first prediction step and the performance further deteriorates for a larger prediction step.

  The percentage forecasting accuracy of the model for different sampling rates is shown in the Plot (C). By decreasing the sampling rate from 2 kHz to 1 kHz, the prediction accuracy decreases from 72% to 63% and 24% to 20% for the first and twentieth prediction step respectively. Hence, by decreasing the sampling rate, the model accuracy decreases.

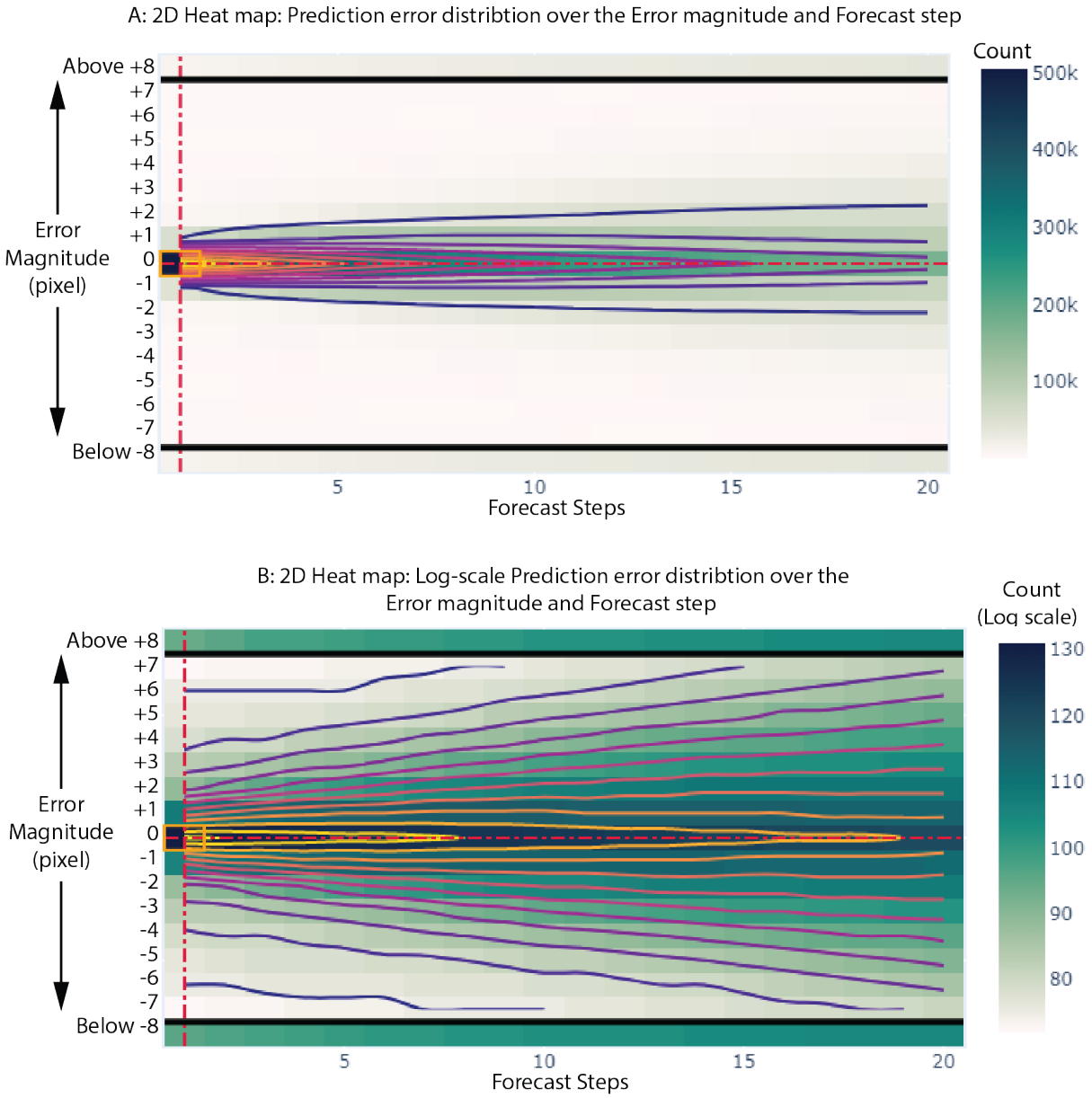


**Figure 4.4: Linear Regression Model:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

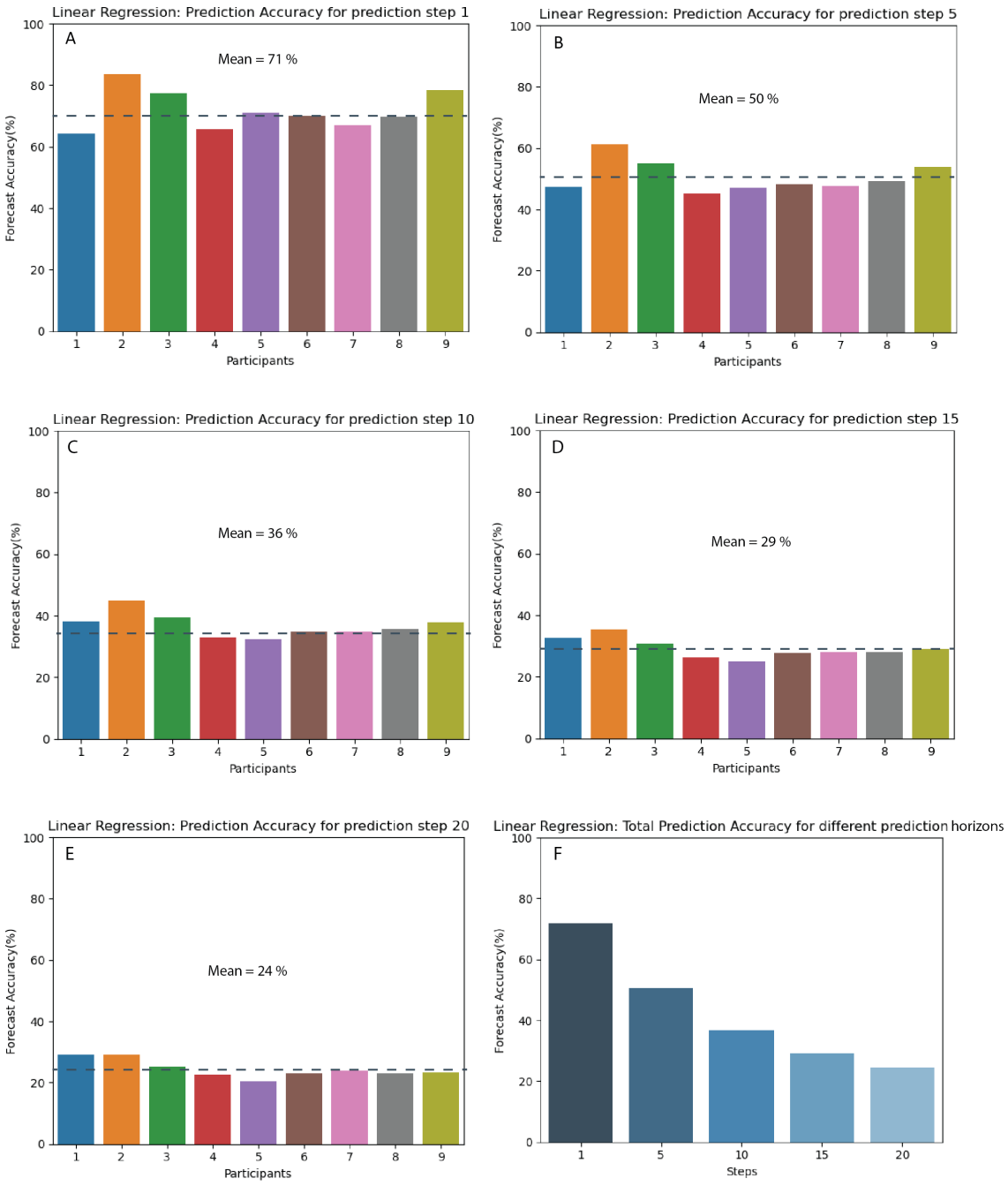


**Figure 4.5:** The plot illustrates the percentage forecast accuracy for the Linear regression algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

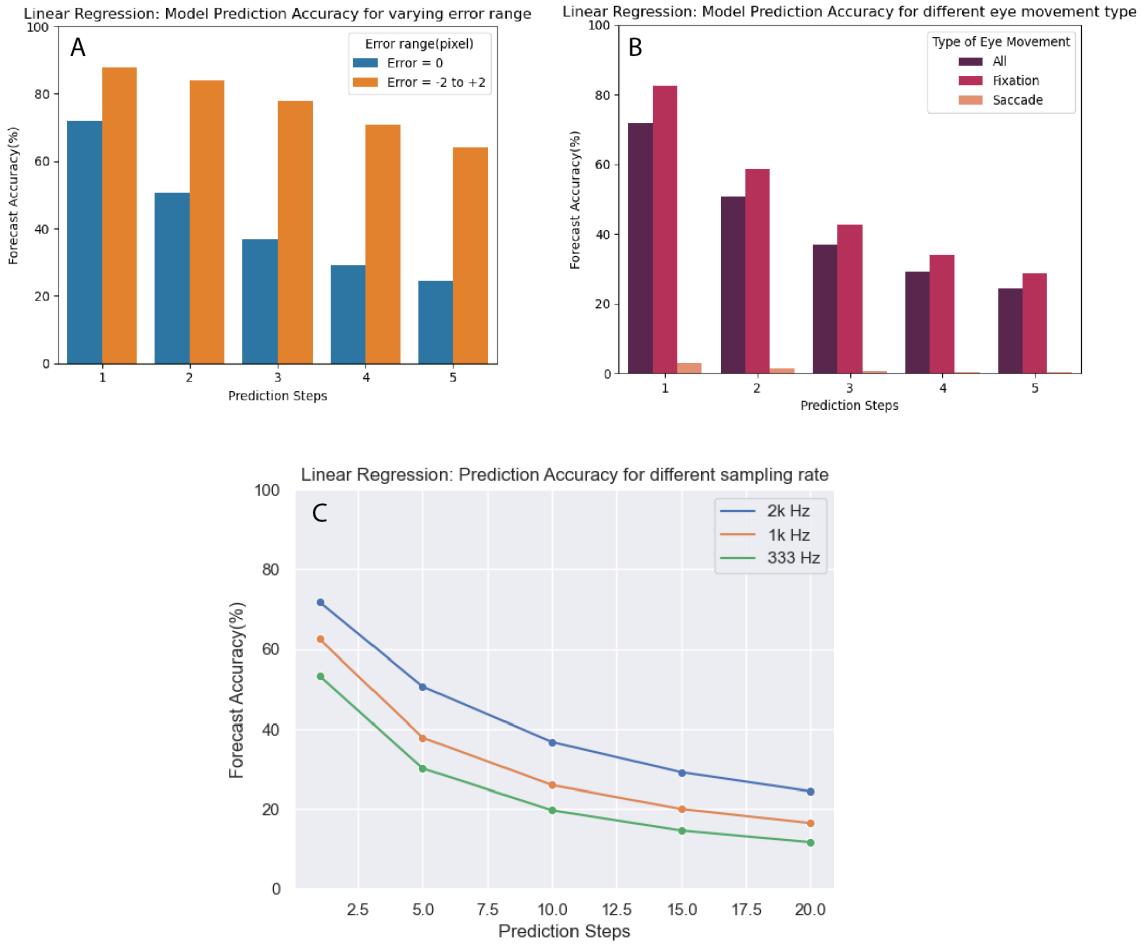


**Figure 4.6:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Linear Regression algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.3 POLYNOMIAL REGRESSION

- Figure 4.7 illustrates the error distribution for the second order Polynomial regression model. Similar to the Linear regression model, the contour lines widen continuously as the prediction steps increase.
- As illustrated in Figure 4.8 (A-E) the mean forecast accuracy for the polynomial regression model for the first prediction step is 71%, fifth prediction step is 51%, tenth prediction step is 37%, fifteenth prediction step is 29% and twentieth prediction step is 28%
- In Figure 4.9 (A) we evaluate our second order polynomial regression model forecast accuracy for an error bandwidth of +2 to -2 pixels. We observe that 88% of the predictions lie between this error range for the first prediction step. For the tenth and twentieth prediction step we get a prediction accuracy of 762% and 65% respectively.

  Similar to out Linear regression algorithm, here in Plot (B), we observe that the model can accurately predict 82% of fixation eye movements but it performs poorly, when it comes to predicting saccade motions.

  The percentage forecasting accuracy of the model for different sampling rates is shown in the Plot (C). By decreasing the sampling rate from 2 kHz to 1 kHz, the prediction accuracy decreases from 71% to 61% and 28% to 19% for the first and twentieth prediction step respectively.

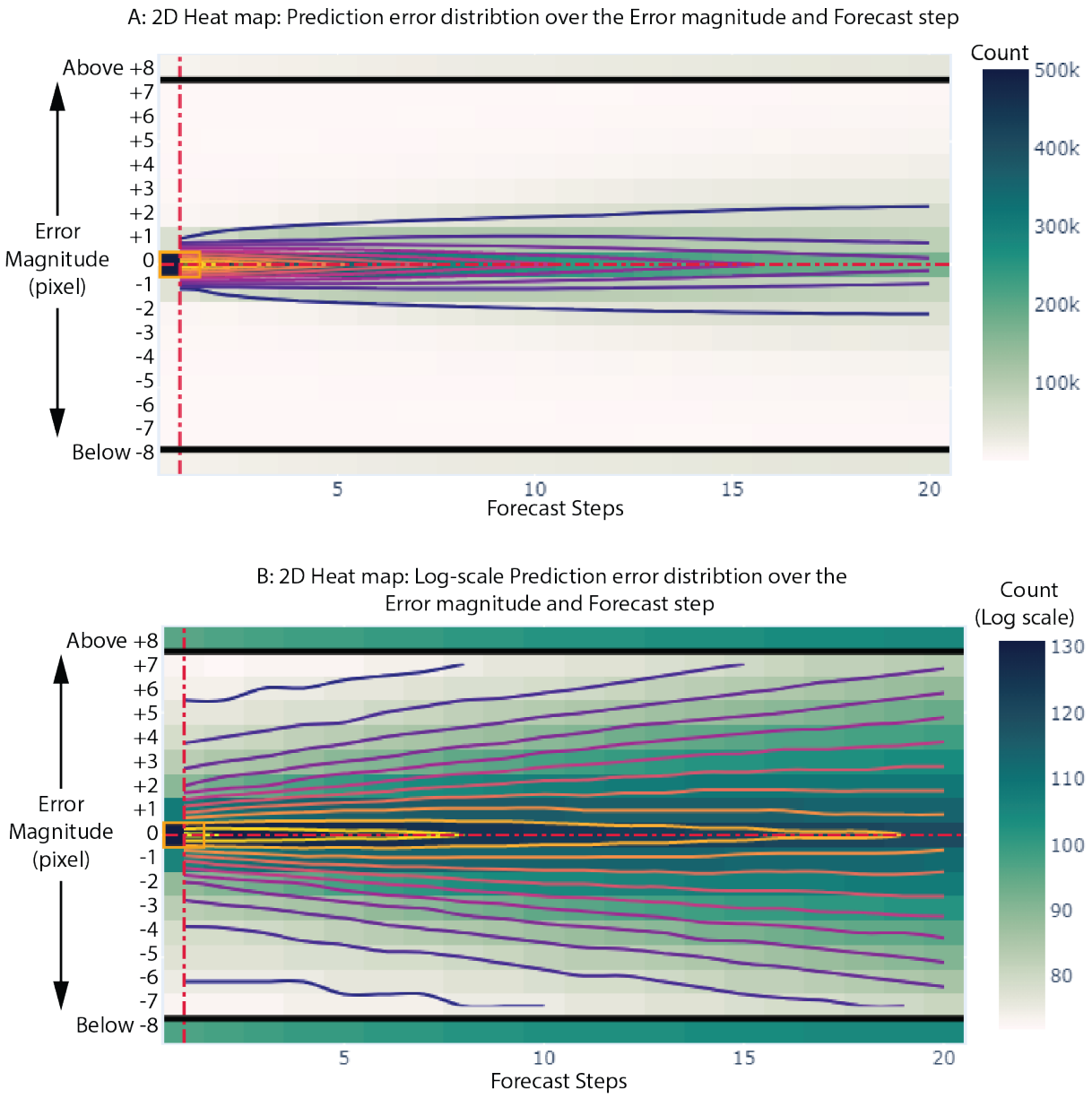


**Figure 4.7: Polynomial Regression Model:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

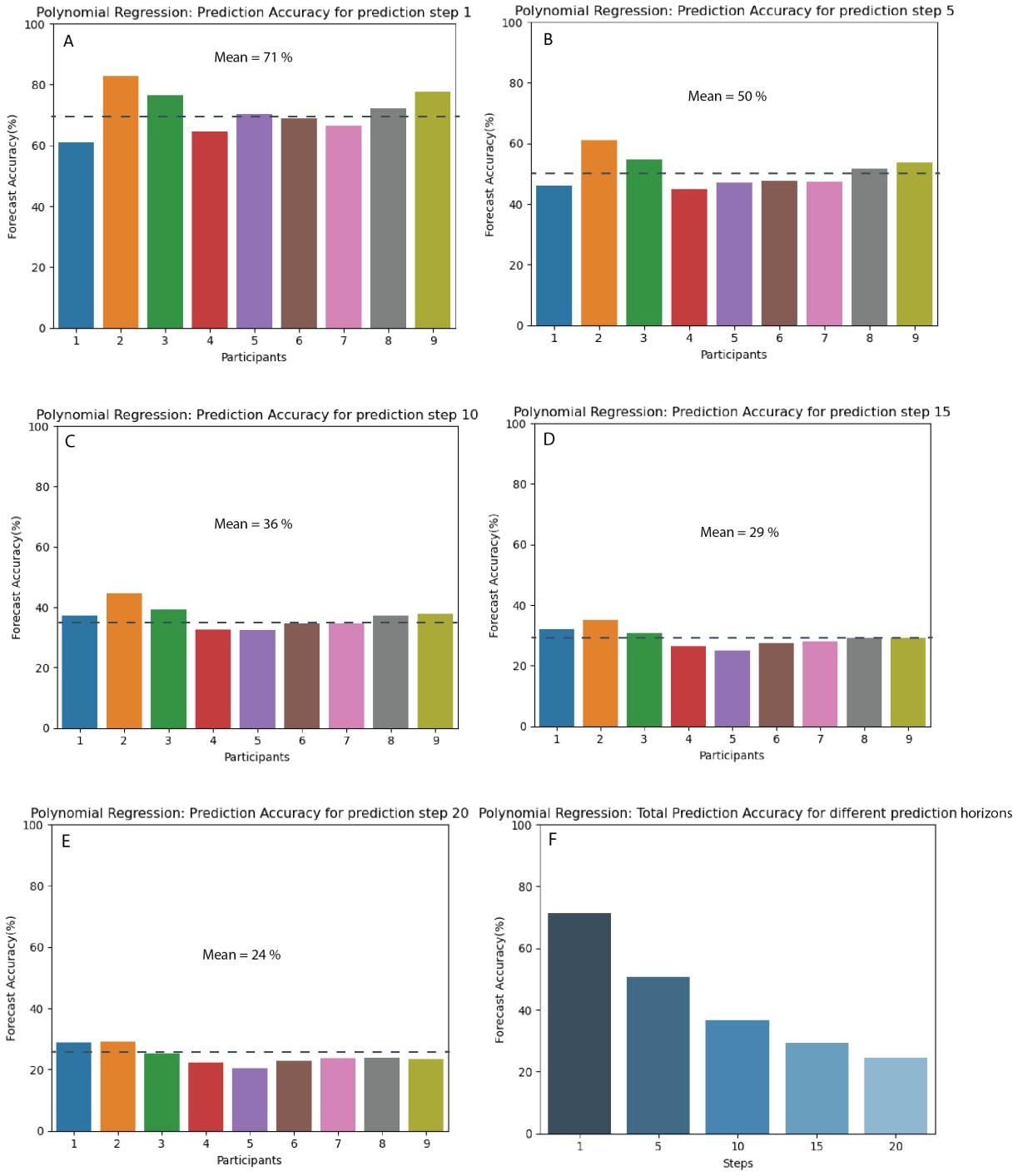


**Figure 4.8:** The plot illustrates the percentage forecast accuracy for the Polynomial regression algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

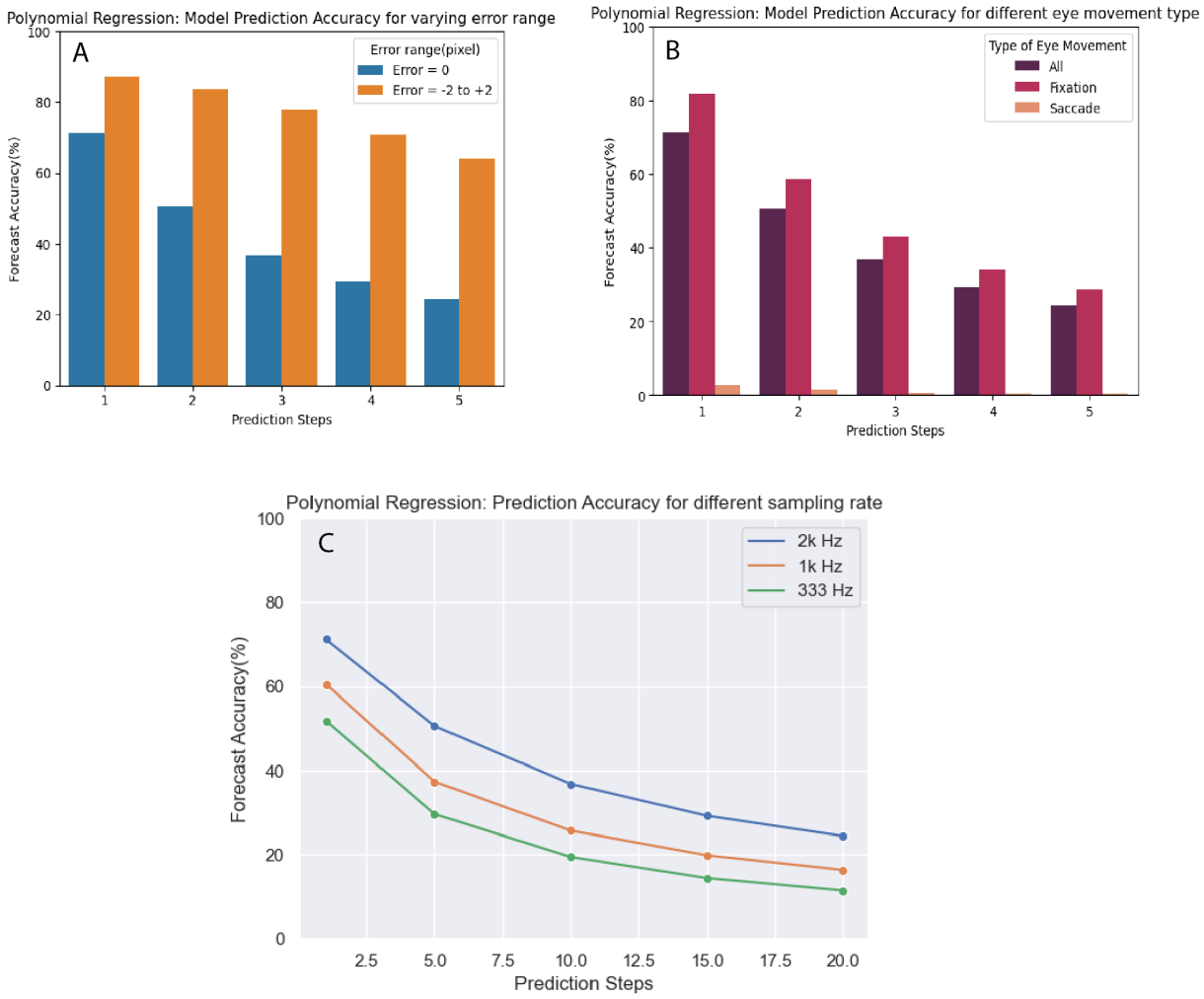


**Figure 4.9:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Polynomial Regression algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.4 KALMAN CONSTANT VELOCITY ESTIMATOR

- Figure 4.10 shown the 2D Heat map and the contour error distribution lines for the Kalman constant velocity estimator. The contour lines are much wider from the start for lower prediction steps. For higher prediction steps, the error widens, but at a much slower magnitude. The percentage accuracy steadily drops as the prediction steps increase, but this fall in the forecast accuracy is much smaller.

- In Figure 4.11 the percentage prediction accuracy with respect to the increasing prediction steps for individual participants have been plotted. The forecast accuracy for all the participant combined is given below:

  - Prediction Step 1: Forecast Accuracy = 46%
  - Prediction Step 5: Forecast Accuracy = 41%
  - Prediction Step 10: Forecast Accuracy = 35%
  - Prediction Step 15: Forecast Accuracy = 31%
  - Prediction Step 20: Forecast Accuracy = 28%

  The performance of the Kalman constant velocity model is much lower than our regression models for the lower prediction steps but due to its gradual decrease, the model accuracy is better than the regression models performance for higher prediction steps.

- Now in the Figure 4.12 (A) we compare the model performance for a higher error range of +2 to -2, 75% of the predictions lie between this error range for the first prediction step. As the prediction steps increases, the percentage prediction falls to 70% and 60% for the tenth and the twentieth prediction steps respectively.

  Plot (B) illustrates the model performance during fixation and saccade movements. The model completely fails to predict the eye positions during a saccade motion. During fixation movements, the model accurately predicts 53% for the first prediction step, and falls to 31% for the twentieth prediction step.

  For a lower sampling rate of 1 kHz, the prediction accuracy further decreases to 28% for the first prediction step. The model performs much poorly for a higher prediction step, giving us

an accuracy of 20% and 17% for the tenth and the twentieth prediction steps respectively.

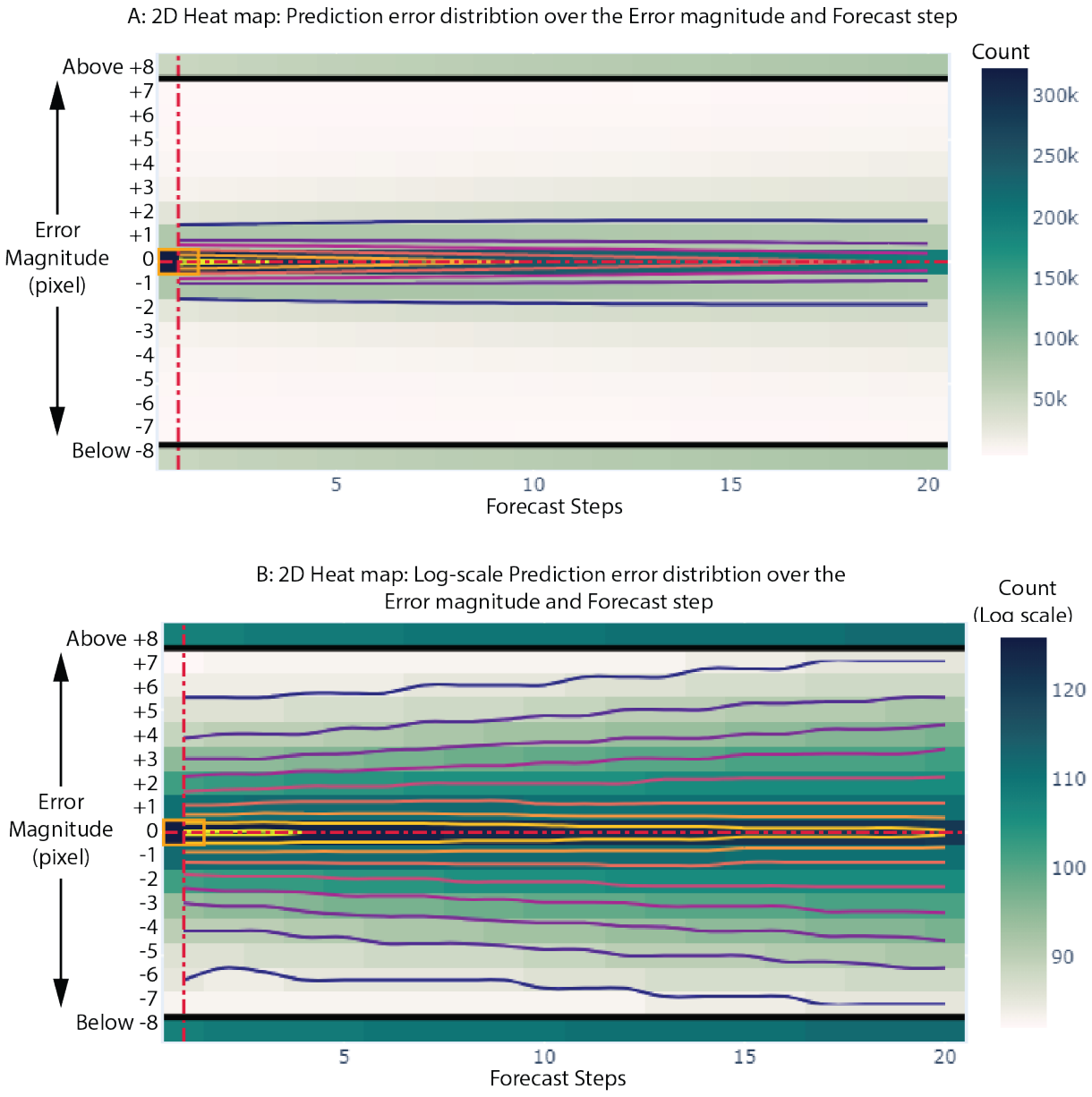


**Figure 4.10: Kalman Constant Velocity Estimator:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

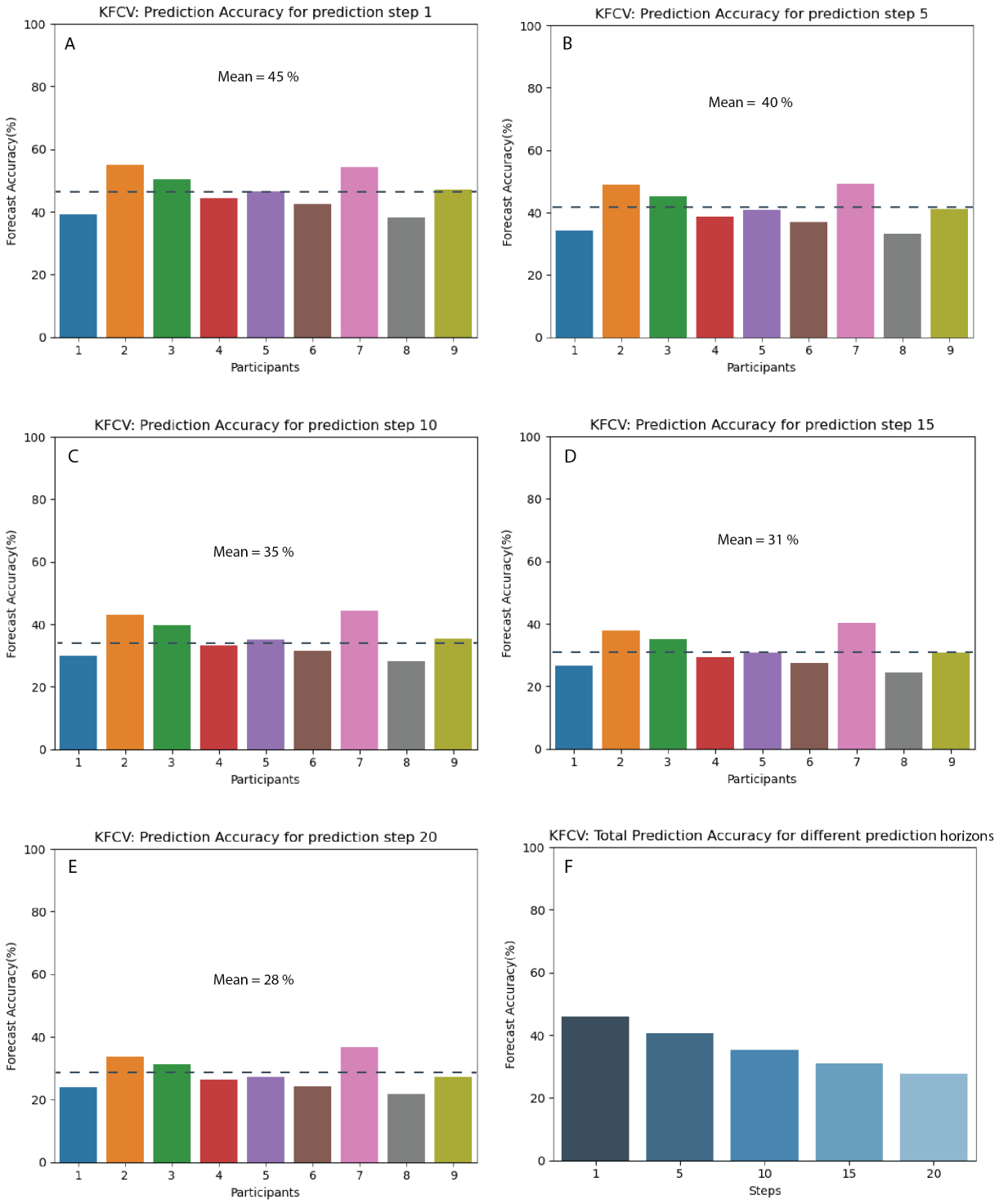


**Figure 4.11:** The plot illustrates the percentage forecast accuracy for the Kalman constant velocity estimator algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

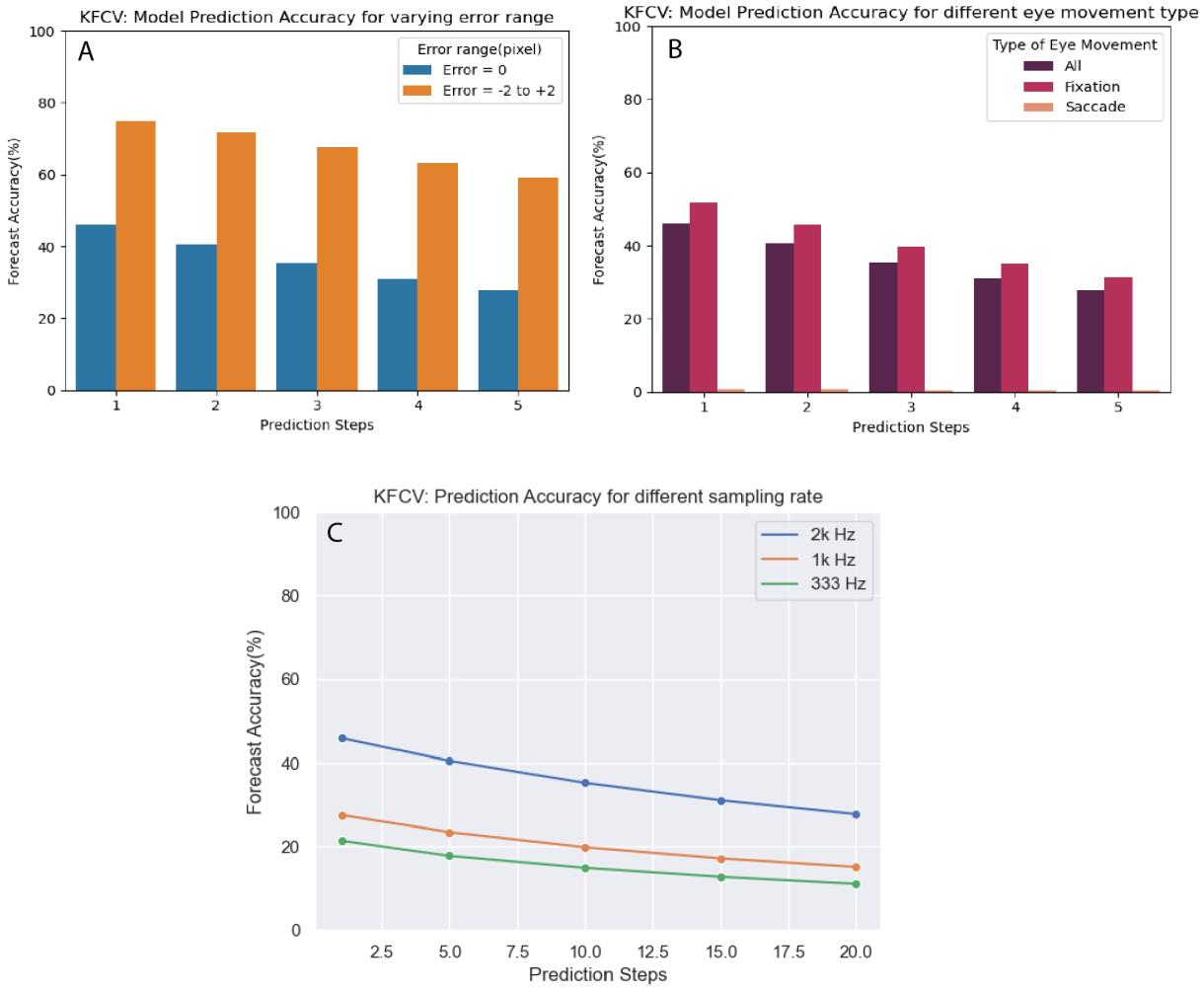


**Figure 4.12:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Kalman constant velocity estimator algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.5 KALMAN CONSTANT ACCELERATION ESTIMATOR

- Figure 4.13 shown the 2D Heat map and the contour error distribution lines for the Kalman constant acceleration estimator. Observing the contour lines, it is visible that the model performs well for the first prediction step, but as the prediction steps increases, the algorithm fails to predict accurately and as a result the contour lines are much wider.
- In Figure 4.14 the percentage prediction accuracy with respect to the increasing prediction steps for individual participants have been plotted. The forecast accuracy for all the participant combined is given below:
  - Prediction Step 1: Forecast Accuracy = 87%
  - Prediction Step 5: Forecast Accuracy = 26%
  - Prediction Step 10: Forecast Accuracy = 20%
  - Prediction Step 15: Forecast Accuracy = 13%
  - Prediction Step 20: Forecast Accuracy = 10%

  For the first prediction step, the Kalman constant acceleration estimator performs well, but then the forecast accuracy decreases exponentially for a higher prediction step. The exponential drop in the model performance is visible in the Figure 4.14 (F).
- Now in the Figure 4.15 (A) we compare the model performance for an error range of +2 to -2:
  - Prediction Step 1: Forecast Accuracy = 98%
  - Prediction Step 5: Forecast Accuracy = 77%
  - Prediction Step 10: Forecast Accuracy = 44%
  - Prediction Step 15: Forecast Accuracy = 29%
  - Prediction Step 20: Forecast Accuracy = 21%

  and we observe that 98% of the predictions lie between this error range for the first prediction step. As the prediction steps increases, the percentage prediction falls to 44% and 60% for the tenth and the twentieth prediction steps respectively.

Plot (B) illustrates the model performance during fixation and saccade movements. For the first prediction step, the model is able to accurately predict 69% of all the saccade movement position and 90% pf all the fixation movement positions. As the prediction steps increase, the model performance decreases.

For a lower sampling rate of 1 kHz, the prediction accuracy further decreases to 79% for the first prediction step. The model performs much poorly for a higher prediction step, giving us an accuracy of 13% and 5% for the tenth and the twentieth prediction steps respectively.

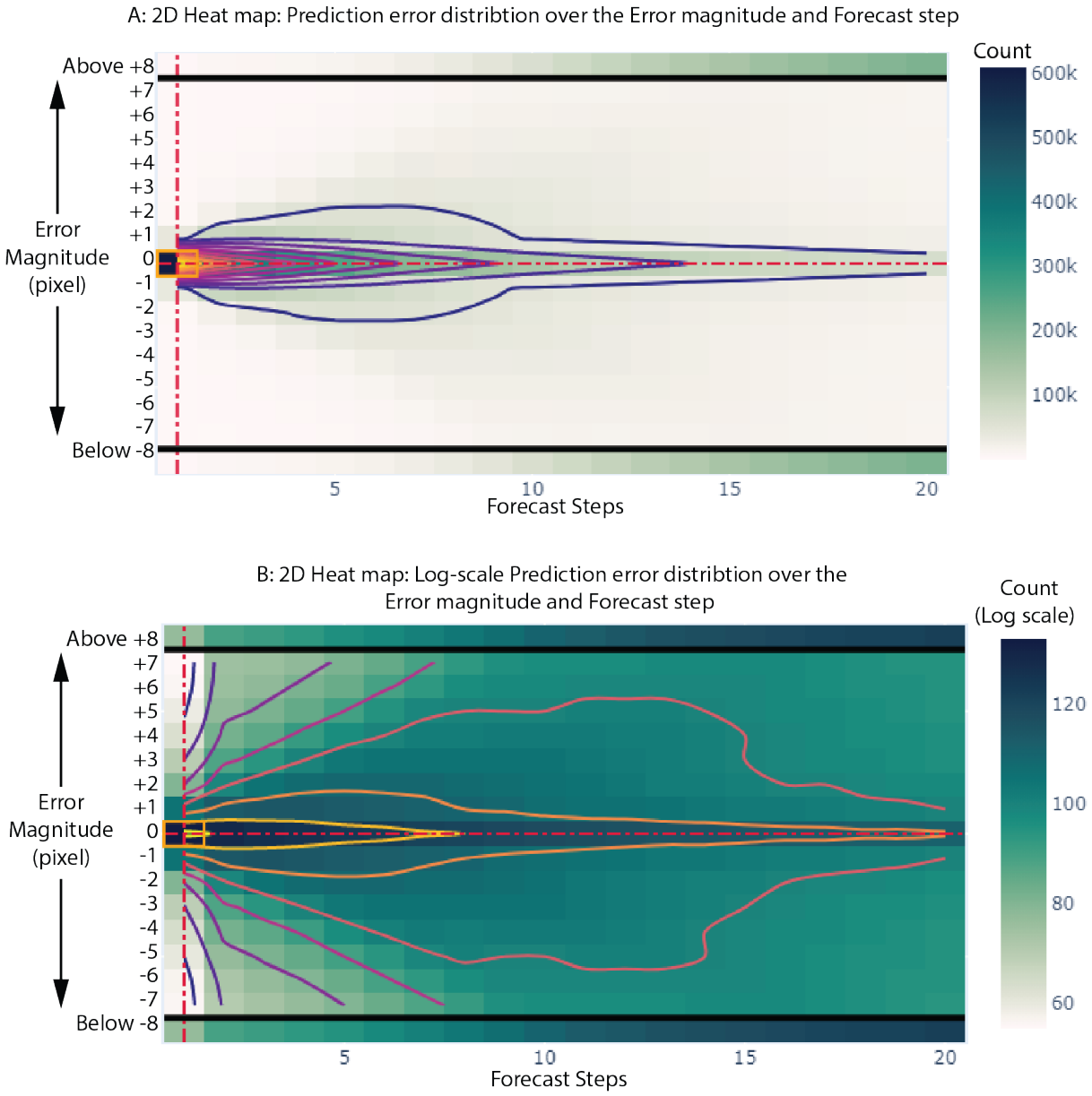


**Figure 4.13: Kalman Constant Acceleration Estimator:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

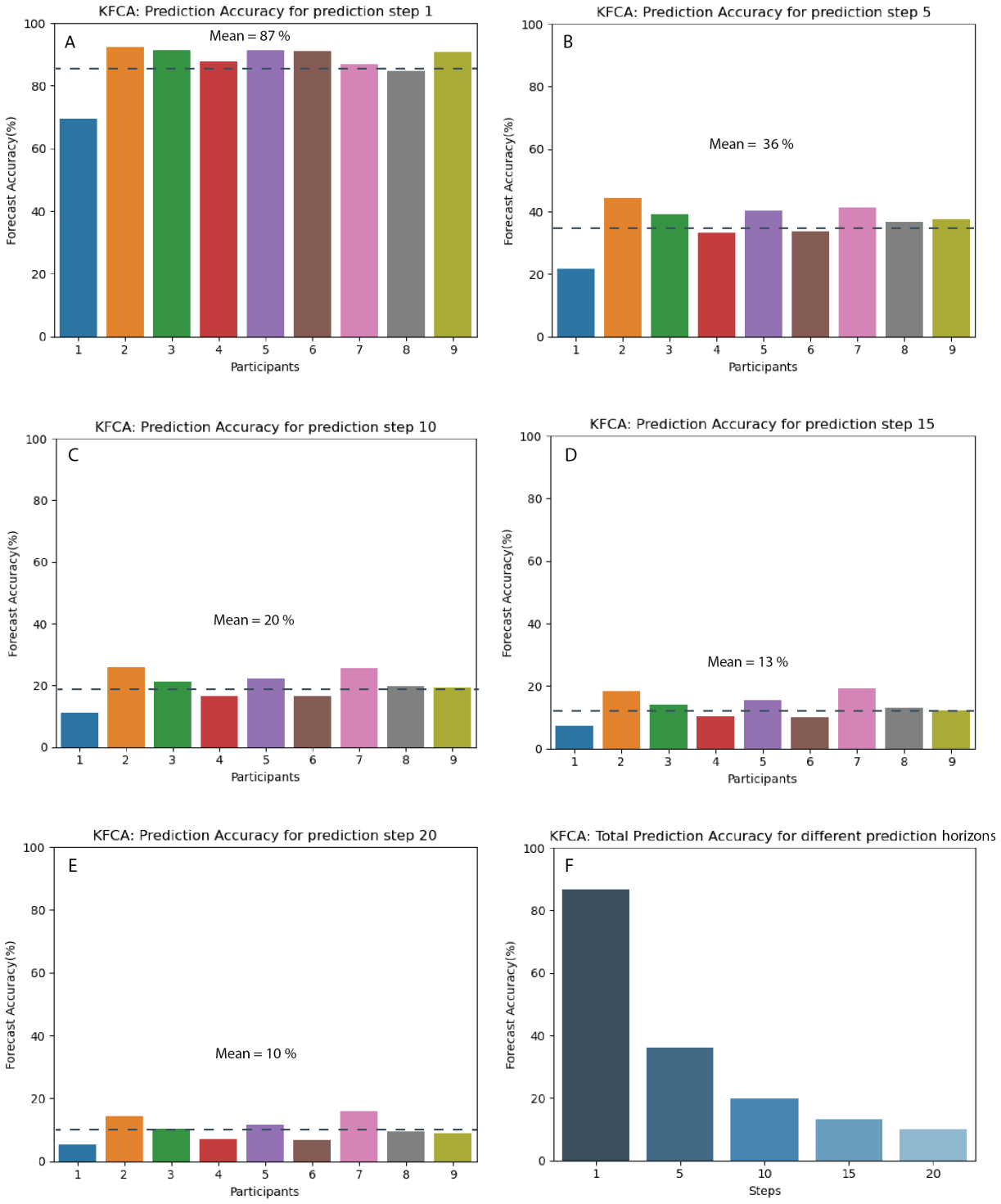


**Figure 4.14:** The plot illustrates the percentage forecast accuracy for the Kalman constant acceleration estimator algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

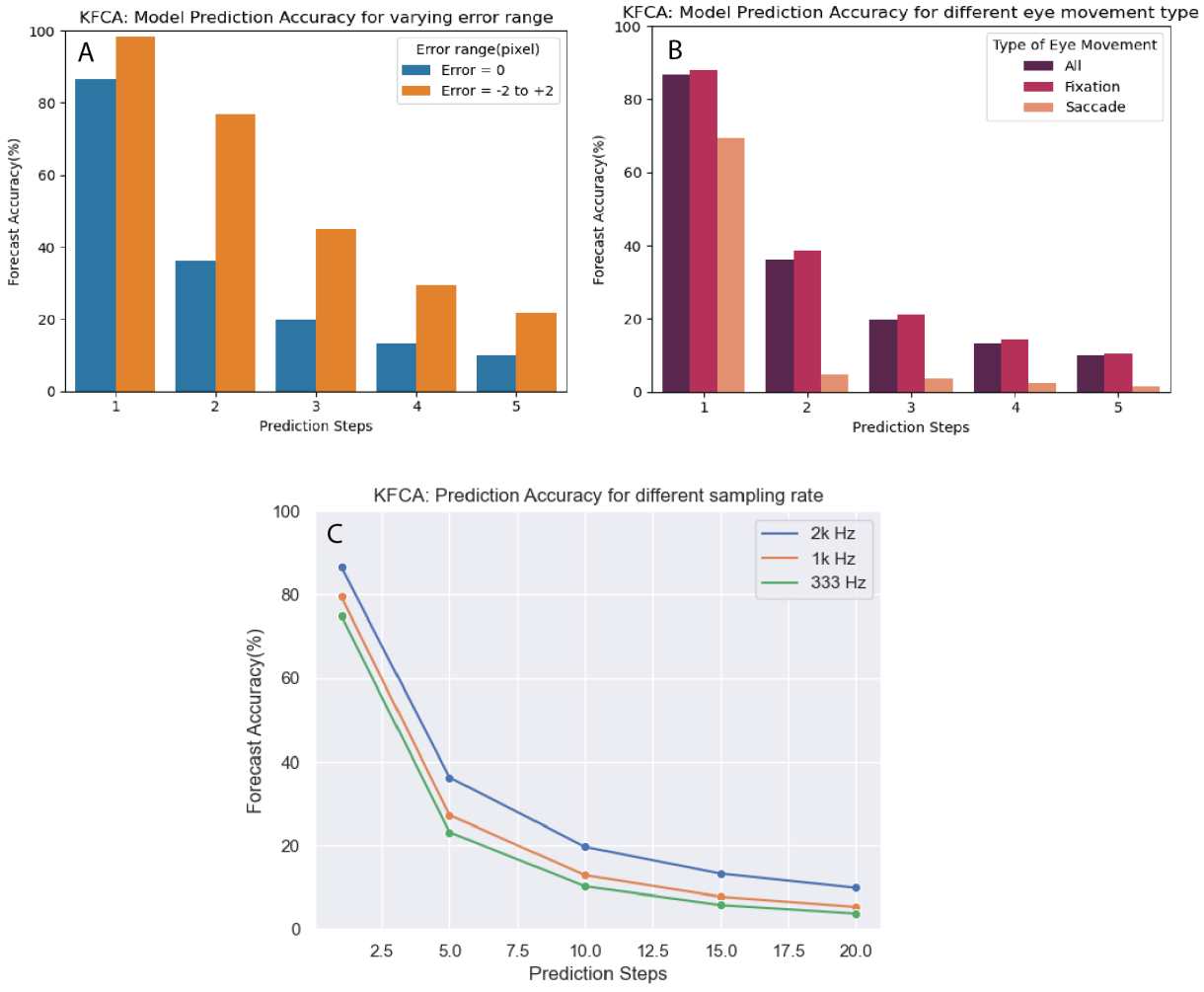


**Figure 4.15:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Kalman constant acceleration estimator algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.6 ADAPTIVE FILTER

- Figure 4.16 shown the 2D Heat map and the contour error distribution lines for the Adaptive filter model. The forecast error is distributed evenly with increasing prediction steps with a steady decrease in the forecast accuracy.
- In Figure 4.17 the percentage prediction accuracy with respect to the increasing prediction steps for individual participants have been plotted. The forecast accuracy for all the participant combined is given below:
  - Prediction Step 1: Forecast Accuracy = 48%
  - Prediction Step 5: Forecast Accuracy = 42%
  - Prediction Step 10: Forecast Accuracy = 37%
  - Prediction Step 15: Forecast Accuracy = 32%
  - Prediction Step 20: Forecast Accuracy = 29%
- Now in the Figure 4.18 (A) we compare the model performance for an error range of +2 to -2:
  - Prediction Step 1: Forecast Accuracy = 79%
  - Prediction Step 5: Forecast Accuracy = 76%
  - Prediction Step 10: Forecast Accuracy = 70%
  - Prediction Step 15: Forecast Accuracy = 68%
  - Prediction Step 20: Forecast Accuracy = 62%

  Again, we observe a steady decline in the percentage of prediction accuracy for an increasing prediction step. A drop of 18% prediction accuracy is acquired from first to the twentieth prediction step.

  Plot (B) illustrates the model performance during fixation and saccade movements. During fixation movements, the model accurately predicts 54% for the first prediction step, and falls to 38% for the twentieth prediction step.

In Plot (C), for a lower sampling rate of 1 kHz, the prediction accuracy falls to 37% for the first prediction step and further falls down to 23% for the tenth prediction step and 17% for the twentieth prediction step.

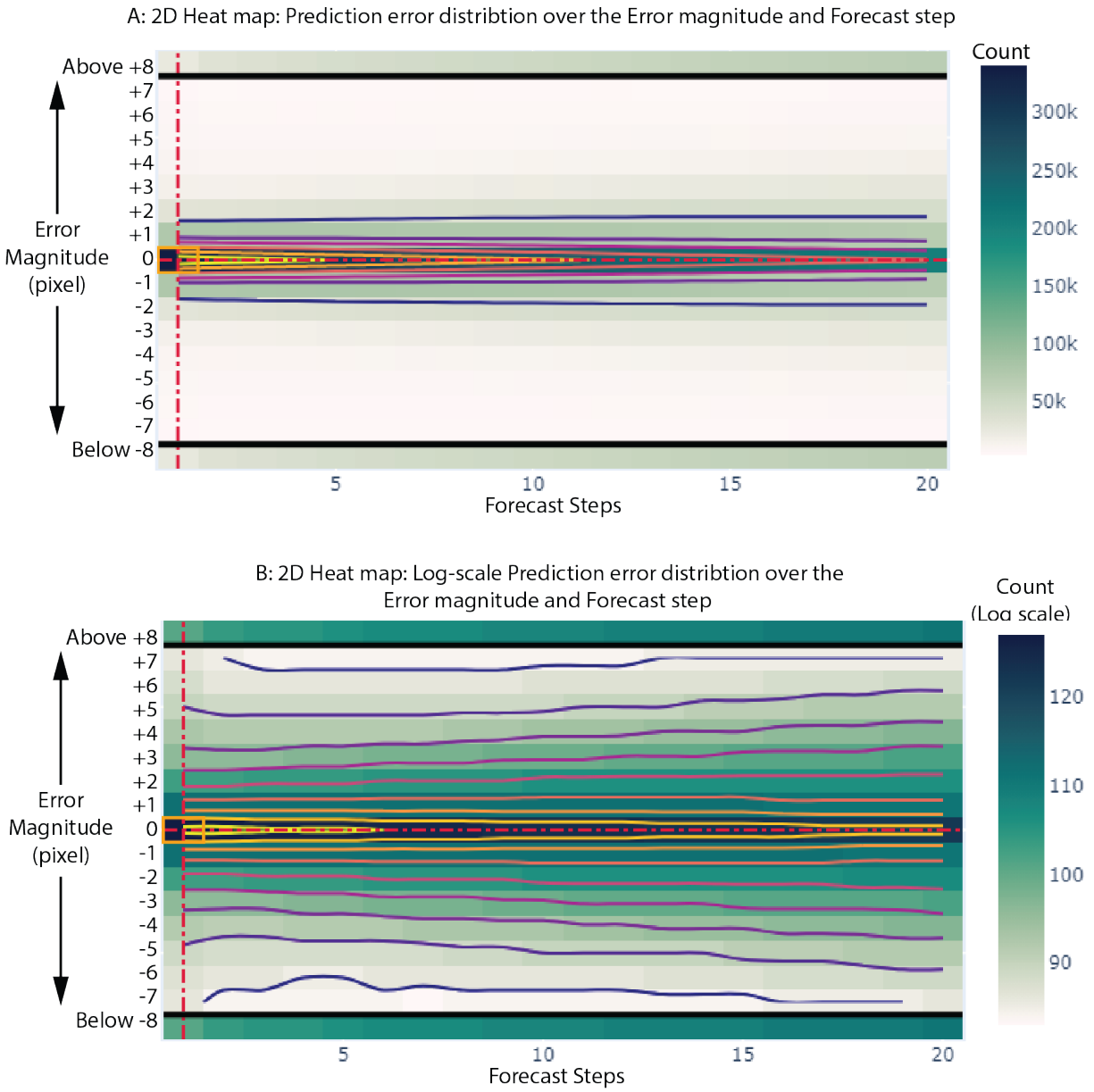


**Figure 4.16: Adaptive Filter Estimator:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

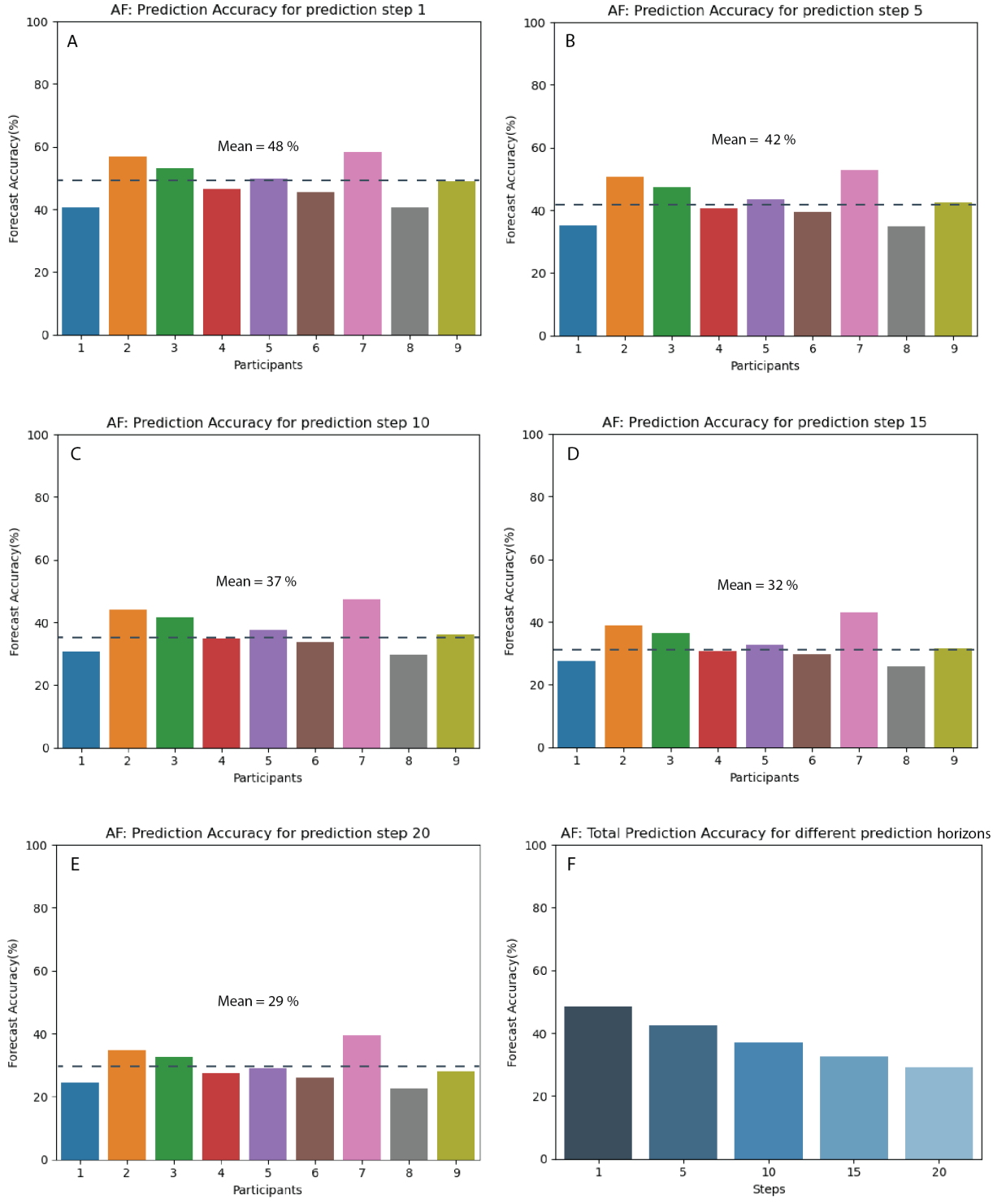


**Figure 4.17:** The plot illustrates the percentage forecast accuracy for the Adaptive filter algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

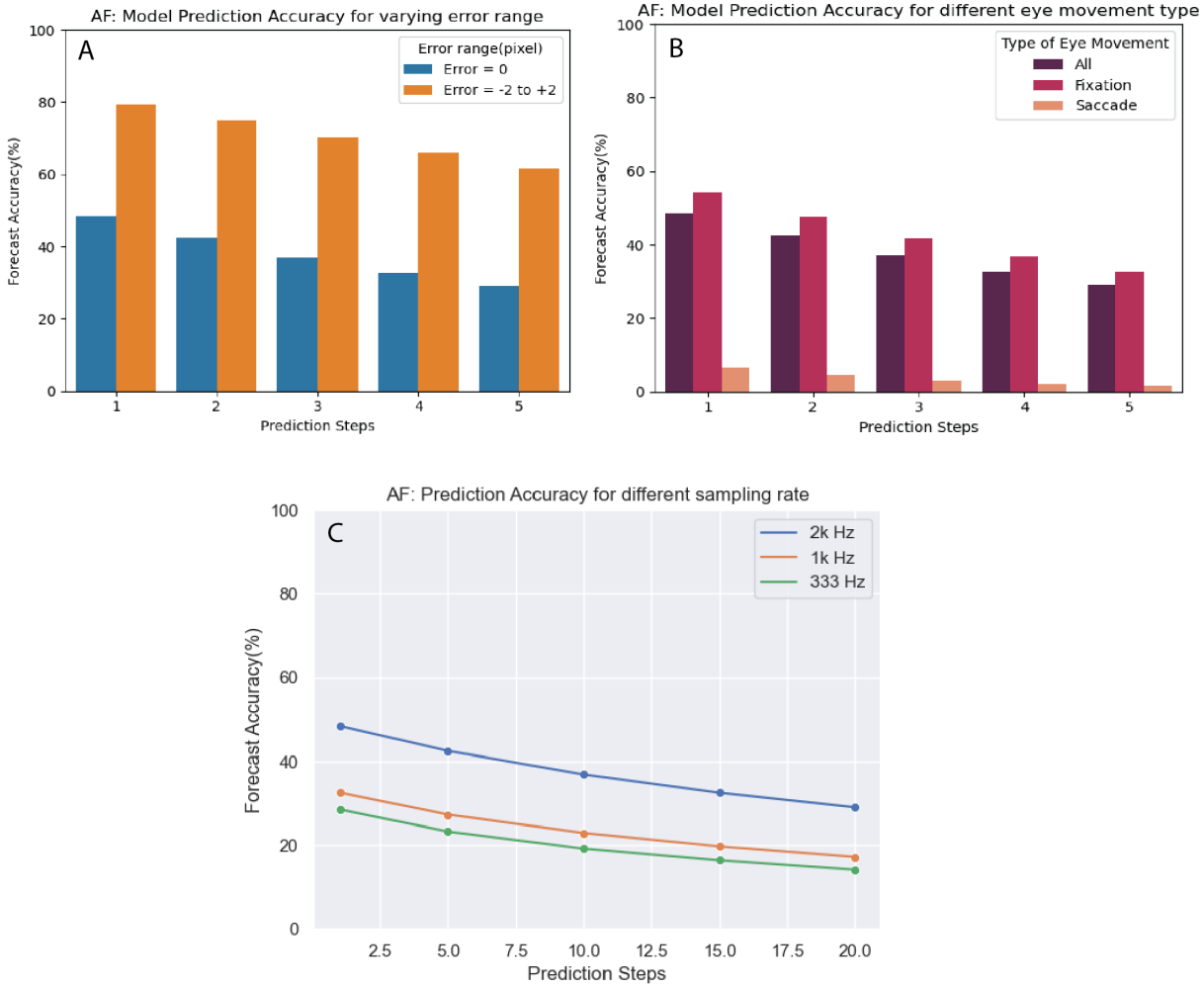


**Figure 4.18:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Adaptive filter algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.7 MULTIPLE MODEL ADAPTIVE ESTIMATOR

- The Figure 4.19 illustrates the 2D Heat map and the contour error distribution lines for the Multiple model adaptive estimator. We observe an evenly distributed forecast error distributed along the zero error axis, widening with increasing prediction steps.
- In Figure 4.20 the percentage prediction accuracy with respect to the increasing prediction steps for individual participants have been plotted. The forecast accuracy for all the participant combined is given below:
  - Prediction Step 1: Forecast Accuracy = 73%
  - Prediction Step 5: Forecast Accuracy = 51%
  - Prediction Step 10: Forecast Accuracy = 37%
  - Prediction Step 15: Forecast Accuracy = 30%
  - Prediction Step 20: Forecast Accuracy = 25%
- Now in the Figure 4.21 (A) we compare the model performance for an error range of +2 to -2:
  - Prediction Step 1: Forecast Accuracy = 88%
  - Prediction Step 5: Forecast Accuracy = 77%
  - Prediction Step 10: Forecast Accuracy = 68%
  - Prediction Step 15: Forecast Accuracy = 60%
  - Prediction Step 20: Forecast Accuracy = 57%

  The MMAE algorithm is able to give us an 88% prediction accuracy for an error bandwidth of +2 to -2 pixels. A drop of 31% prediction accuracy is acquired from first to the twentieth prediction step.

  Plot (B) illustrates the model performance during fixation and saccade movements. During fixation movements, the model accurately predicts 80% for the first prediction step, which falls to 29% for the twentieth prediction step. However the model fails to accurately predict the eye position during saccade movements, giving us a 7% forecast accuracy.

In Plot (C), we observe that by lowering the sampling rate from 2 kHz to 1 kHz, the MMAE algorithm decreases by 4% from 72% to 68% for the first prediction step. As the prediction steps increase, the forecast accuracy falls down to 25% for the tenth prediction step and 20% for the twentieth prediction step.

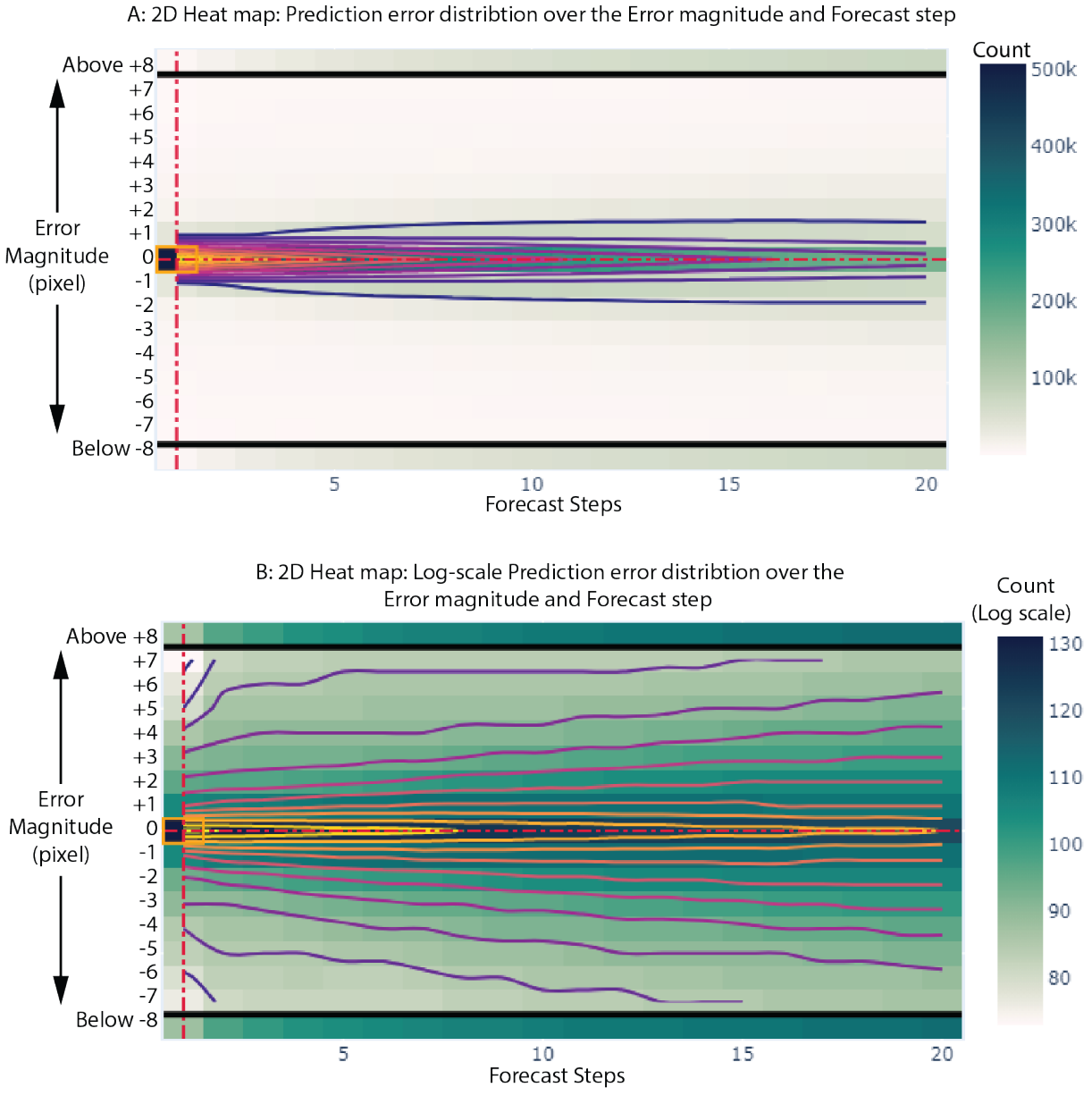


**Figure 4.19: Multiple Model Adaptive Estimator:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

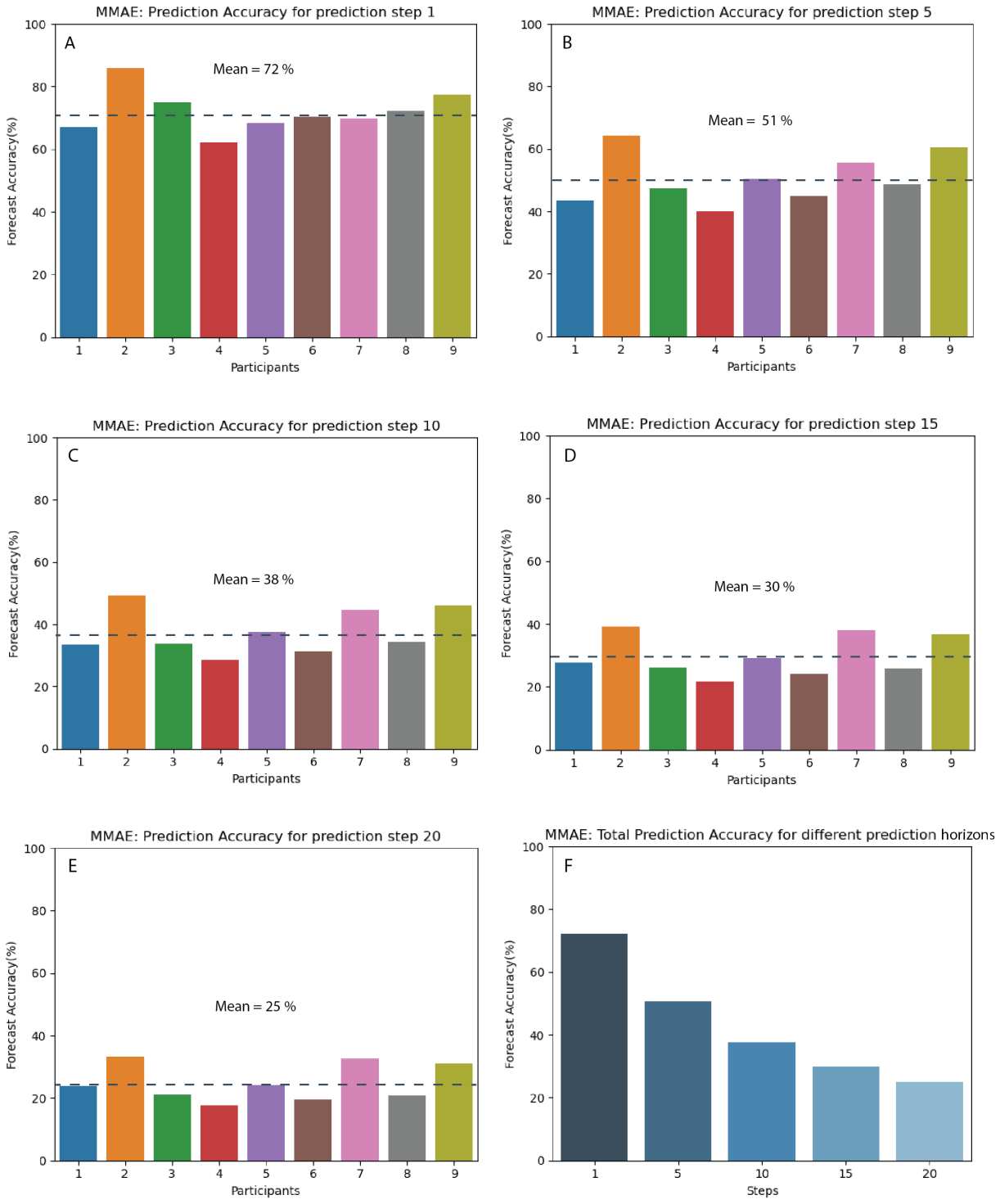


**Figure 4.20:** The plot illustrates the percentage forecast accuracy for the Multiple model adaptive estimator algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

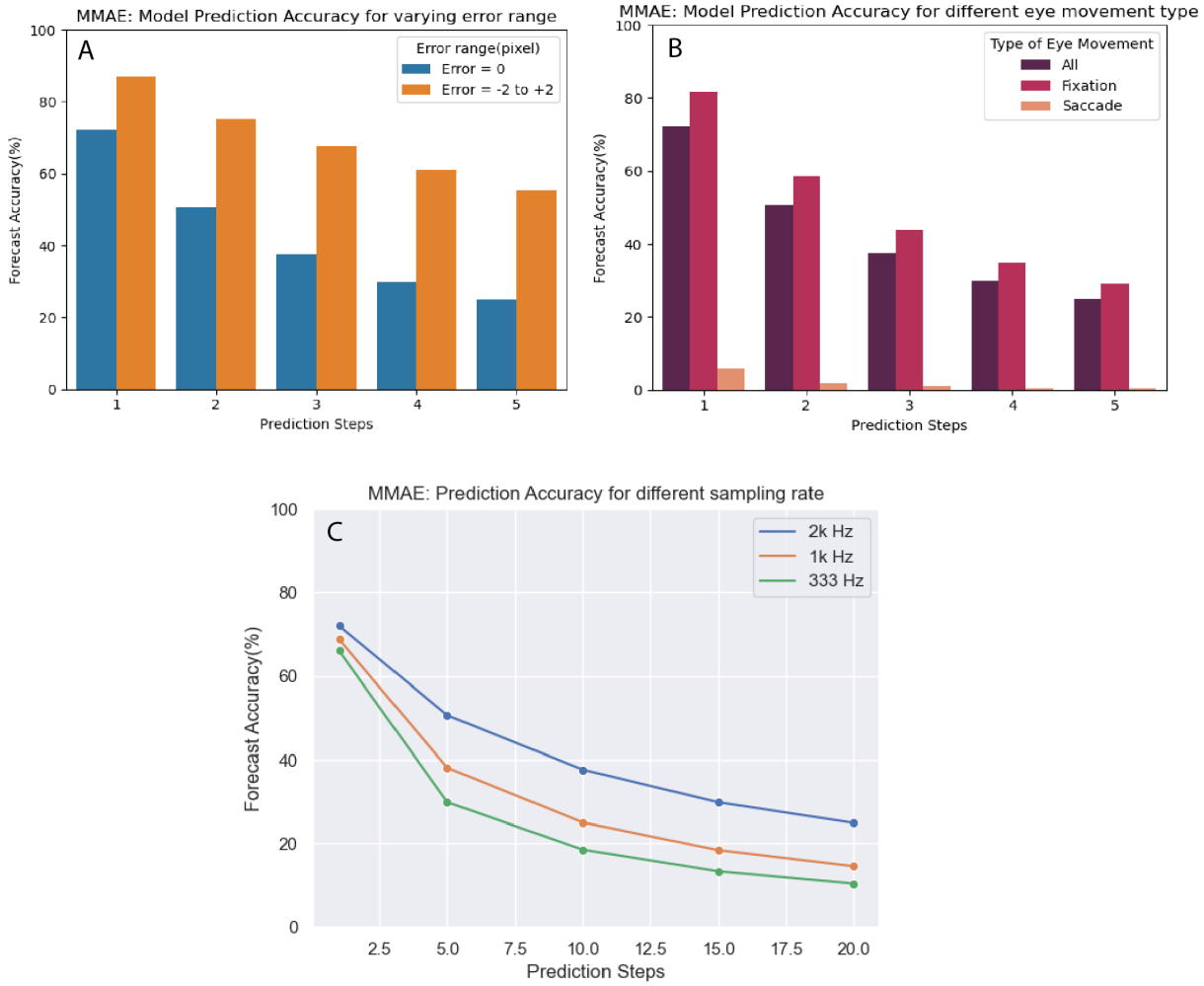


**Figure 4.21:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Multiple model adaptive estimator algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.8 INTERACTIVE MULTIPLE MODEL ESTIMATOR

- Our final prediction model is the Interactive multiple model estimator. The Figure 4.22 illustrates the 2D Heat map and the contour error distribution lines for the IMM algorithm. The model gives us an evenly distributed forecast error along the zero error axis, widening with increasing prediction steps.
- In Figure 4.23 the percentage prediction accuracy with respect to the increasing prediction steps for individual participants have been plotted. The forecast accuracy for all the participant combined is given below:
  - Prediction Step 1: Forecast Accuracy = 88%
  - Prediction Step 5: Forecast Accuracy = 60%
  - Prediction Step 10: Forecast Accuracy = 39%
  - Prediction Step 15: Forecast Accuracy = 30%
  - Prediction Step 20: Forecast Accuracy = 23%

  The numbers show that the IMM estimator has the best performance for the first prediction step with a prediction accuracy of 88%.
- Now in the Figure 4.24 (A) we compare the model performance for an error range of +2 to -2:
  - Prediction Step 1: Forecast Accuracy = 99%
  - Prediction Step 5: Forecast Accuracy = 91%
  - Prediction Step 10: Forecast Accuracy = 83%
  - Prediction Step 15: Forecast Accuracy = 67%
  - Prediction Step 20: Forecast Accuracy = 58%

  The IMM algorithm is able to give us an 99% prediction accuracy for an error bandwidth of +2 to -2 pixels. However, a drop of 41% prediction accuracy is obtained from first to the twentieth prediction step.

Plot (B) illustrates the model performance during fixation and saccade movements. During fixation movements, the model accurately predicts 90% for the first prediction step, which falls to 26% for the twentieth prediction step. During saccade movements, the model has an accuracy of 62% for the first prediction step. As the prediction steps increase, the error in prediction also increases.

In Plot (C), we observe that by lowering the sampling rate from 2 kHz to 1 kHz, the IMM algorithm decreases by 7% from 88% to 80% for the first prediction step. As the prediction steps increase, the forecast accuracy falls down to 41% for the tenth prediction step and 18% for the twentieth prediction step.

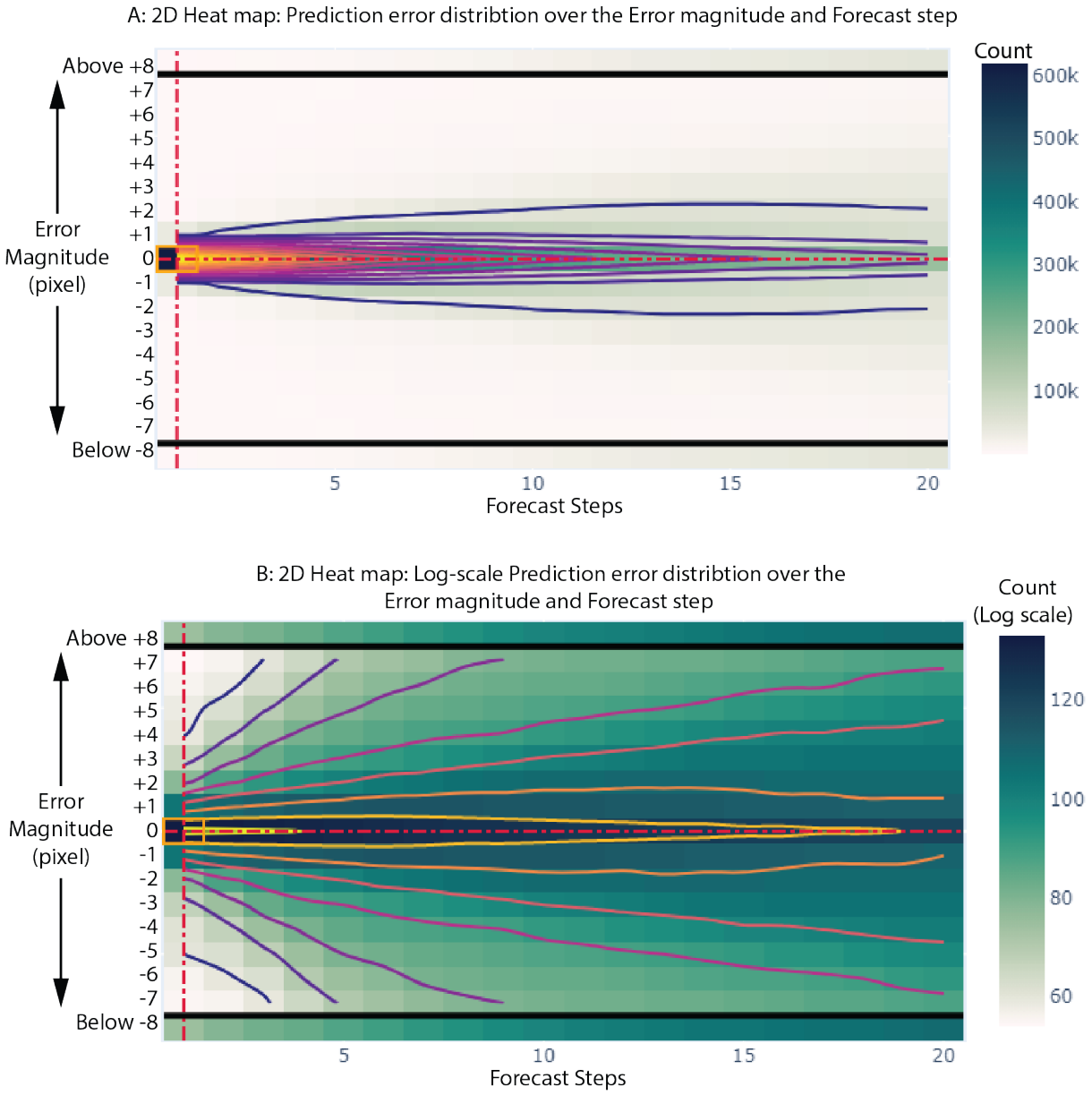


**Figure 4.22: Interactive Multiple Model Estimator:** 2D Heat map illustrating the distribution of error over the error magnitude (pixels) and the prediction steps:**A** Linear scale, **B** Logarithmic scale

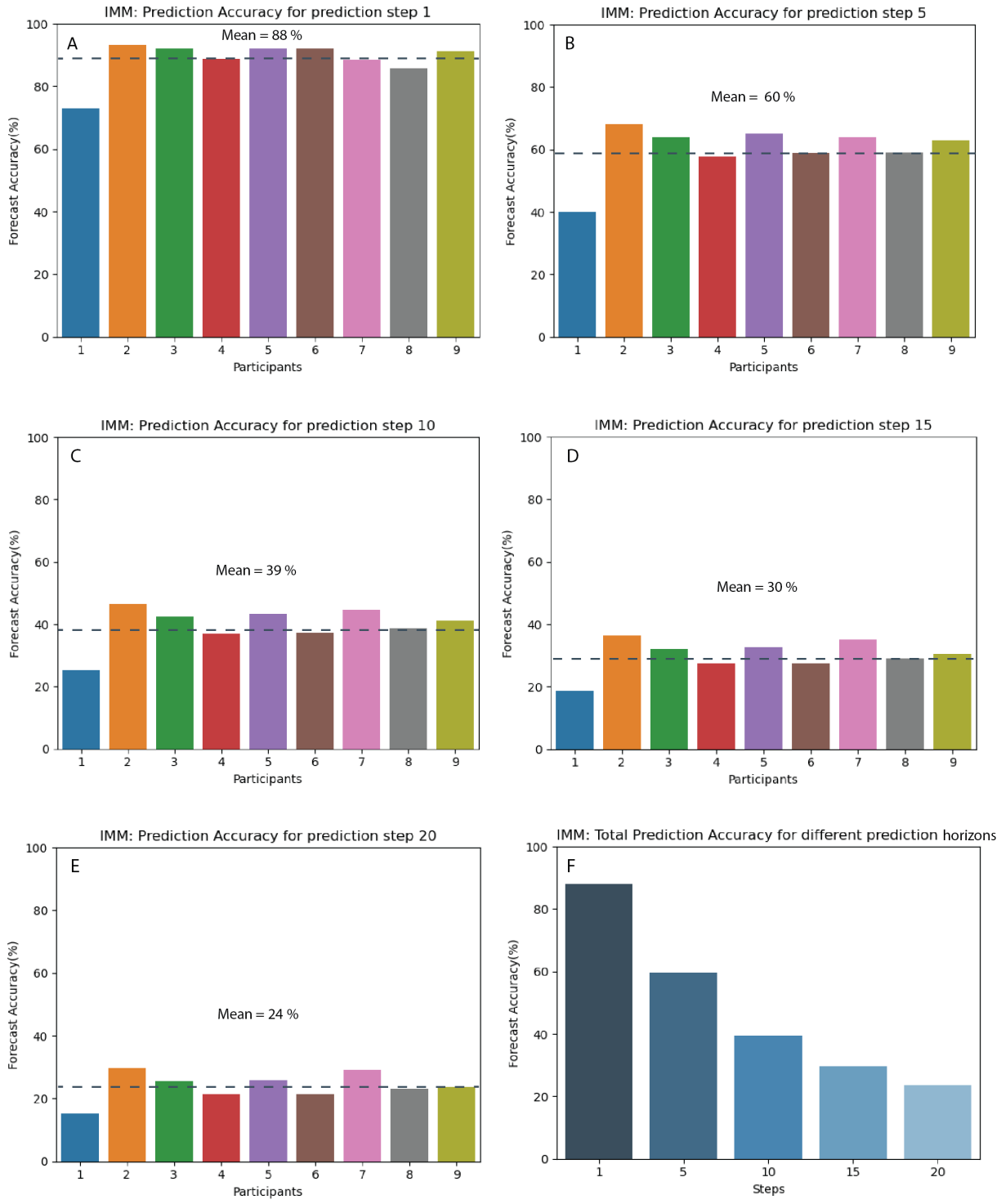


**Figure 4.23:** The plot illustrates the percentage forecast accuracy for the Interactive multiple model estimator algorithm conducted on different participants for the varying prediction steps: **A** Prediction step 1, **B** Prediction step 5, **C** Prediction step 10, **D** Prediction step 15, **E** Prediction step 20. Plot **F** illustrates the total forecast accuracy for the varying prediction steps

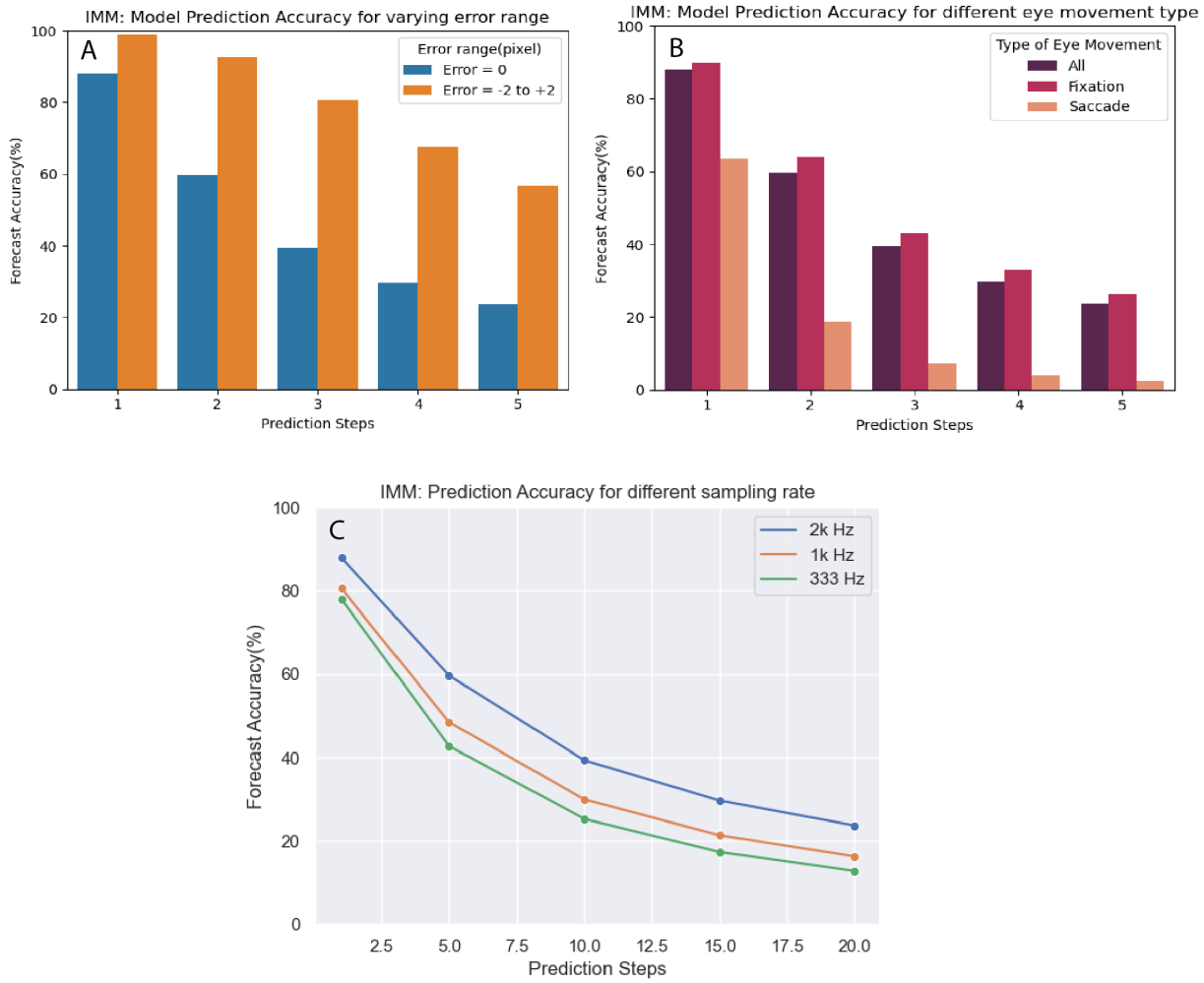


**Figure 4.24:** The plot compares different parameters and illustrates the percentage forecast accuracy with increasing prediction steps for the Interactive multiple model estimator algorithm. Parameter: **A** Different Error range (pixels), **B** Types of eye movements, **C** Sampling rate (Hz)

## 4.9 MODEL COMPARISON

The Figure 4.25 (A) illustrates the percentage forecast accuracy of the different algorithms at a sampling rate of 2 kHz. For a sampling rate of 2 kHz and error range of 0 pixel, the Polynomial Regression model has the highest forecast accuracy amongst the deterministic forecasting models, while the Interactive Multiple Model estimator model has the highest forecast accuracy overall. The Interactive Multiple Model gives us the maximum accuracy of 88% for the first prediction step. The forecast accuracy however drops to 24% for the twentieth prediction step.

In Table 4.2, for an error range between +2 and -2 pixels, the IMM model gives us a forecast accuracy of 99% for the first prediction step and a 58% accuracy for the twentieth prediction step.

In Figure 4.25 (A-C), we compare the percentage forecast accuracy of the different models with a lower sampling rate. The IMM estimator model again has the highest forecast accuracy, but the percentage accuracy falls down as the sampling rate is reduced.

**Table 4.1:** Percentage forecast accuracy for different algorithms evaluated for: **A** Sampling rate = 2 kHz, **B** Error range = 0 pixel

| Name of Model | Step=1 | Step=5 | Step=10 | Step=15 | Step=20 |
|---|---|---|---|---|---|
| Delta Coding | 69% | 44% | 30% | 23% | 19% |
| Linear Regression | 71% | 51% | 37% | 29% | 24% |
| Polynomial Regression | 71% | 51% | 37% | 29% | 24% |
| Kalman Constant Velocity Model | 46% | 41% | 35% | **31%** | 28% |
| Kalman Constant Acceleration Model | **87%** | 36% | 20% | 13% | 10% |
| Adaptive Filter | 48% | 43% | 37% | **33%** | **29%** |
| Multiple Model Adaptive Estimator | 72% | 51% | 38% | 30% | 25% |
| Interactive Multiple Model Estimator | **88%** | **60%** | **40%** | 30% | 24% |

**Table 4.2:** Percentage forecast accuracy for different algorithms evaluated for: **A** Sampling rate = 2 kHz, **B** Error range = +2 to -2 pixels

| Name of Model | Step=1 | Step=5 | Step=10 | Step=15 | Step=20 |
|---|---|---|---|---|---|
| Delta Coding | 89% | 84% | 77% | 70% | **66%** |
| Linear Regression | 89% | 80% | 77% | **72%** | **66%** |
| Polynomial Regression | 88% | 83% | 76% | 71% | 65% |
| Kalman Constant Velocity Model | 75% | 72% | 70% | 63% | 60% |
| Kalman Constant Acceleration Model | 98% | 77% | 44% | 29% | 21% |
| Adaptive Filter | 79% | 76% | 70% | 68% | 61% |
| Multiple Model Adaptive Estimator | 88% | 78% | 68% | 60% | 57% |
| Interactive Multiple Model Estimator | **99%** | **92%** | **83%** | 67% | 58% |

**Table 4.3:** Percentage forecast accuracy for different algorithms evaluated for: **A** Sampling rate = 1 kHz, **B** Error range = 0 pixel

| Name of Model | Step=1 | Step=5 | Step=10 | Step=15 | Step=20 |
|---|---|---|---|---|---|
| Delta Coding | 61% | 30% | 19% | 14% | 12% |
| Linear Regression | 63% | 38% | 26% | 20% | 17% |
| Polynomial Regression | 60% | 37% | 26% | 21% | 17% |
| Kalman Constant Velocity Model | 28% | 23% | 20% | 17% | 16% |
| Kalman Constant Acceleration Model | 79% | 28% | 12% | 8% | 5% |
| Adaptive Filter | 32% | 27% | 23% | 20% | 17% |
| Multiple Model Adaptive Estimator | 69% | 38% | 25% | 18% | 15% |
| Interactive Multiple Model Estimator | **81%** | **49%** | **31%** | **22%** | **20%** |

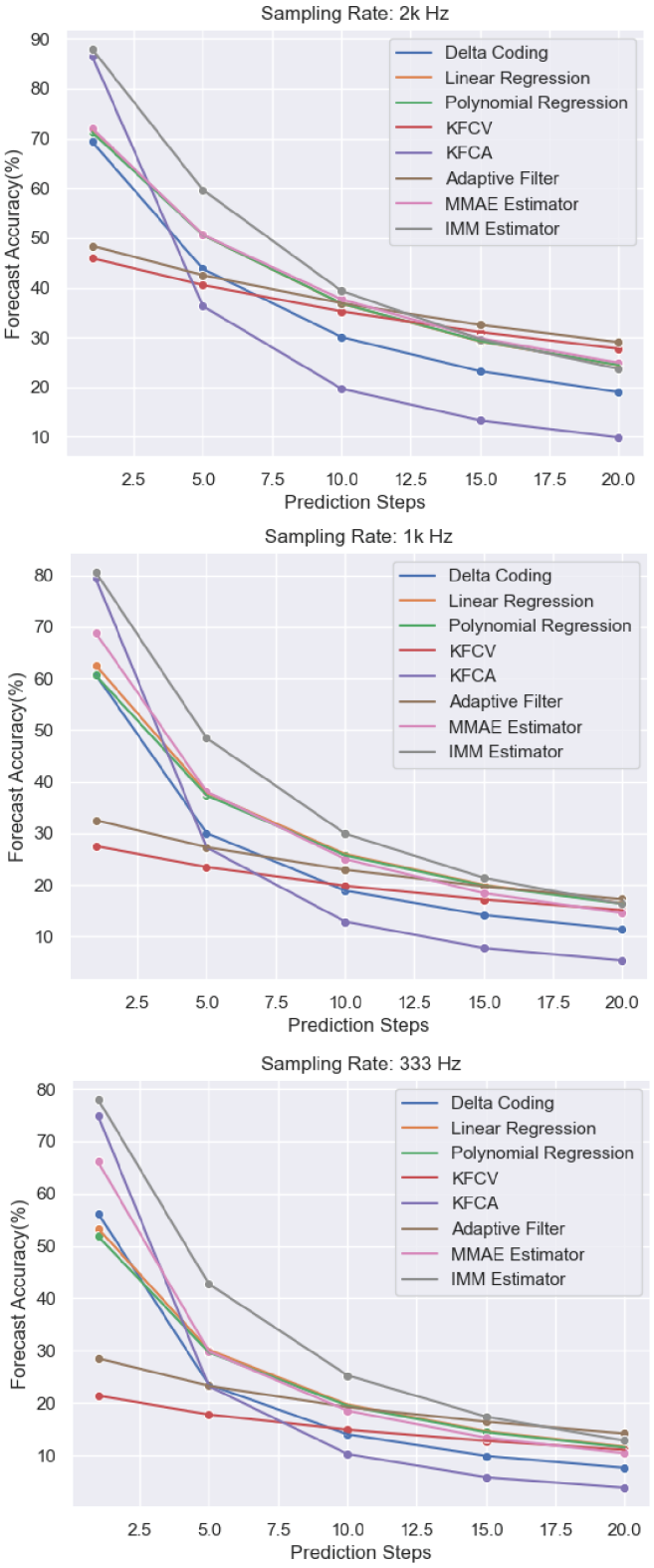


**Figure 4.25:** Comparing the model performance for the different algorithms for a sampling rate of: **A** 2 kHz, **B** 1 kHz, **C** 333 Hz

## 4.10 DISCUSSION

In this paper we have discussed the designing of real time prediction algorithms that have shown to predict the future eye position up to 10 ms (20 frames) in advance. We have created our evaluation metrics and plots for carrying out comparable studies for assessing the prediction algorithms. Although the offline evaluation of the IMM prediction model appear sufficiently accurate for the given range of horizons (the prediction error range from +2 to -2 pixels which is less or comparable to the system noise), we still need to integrate the algorithms into the FPGA firmware and evaluate the response latency for the generated image transformations.

Our results indicate that the prediction models have a higher forecast accuracy when it comes to predicting eye positions during fixation eye movements but the forecast accuracy falls down when it comes to predicting saccade eye movements. Since the saccade eye movements have higher and unpredictable variations in velocity when compared with the fixational eye movements, the predictive models fail to accurately predict the path of the eyes. Here we discuss the possible solutions for improving the forecasting accuracy of the predictive models. A solution could be to improve the IMM model by adding multiple banks of dynamic models that would cover wider range of system dynamics to generate accurate estimations. The bank of models may include Kalman constant velocity, constant acceleration, constant turn and also regression algorithms that are previously discussed. The IMM algorithm should be able to give us an improved prediction, with the banks of several additional models. A previous study on eye movement forecasting during saccadic eye movement has proposed the use of Taylor series that could reduce the latency between the eye position and the image transformation during eye movements [49].

With the development of faster digital processors, it would be possible to utilize more complex algorithms for predicting the eye movements. With the implementation of faster processors, we could make use of more complex and advanced prediction algorithms. Studies of time-series forecasting models using Nonlinear filtering theory such as the particle filters (sequential Monte Carlo methods) have shown to produce vastly superior estimates in comparison to the unscented Kalman filters as well as extended Kalman filters [50][51][52].

# CHAPTER 5

# CONCLUSIONS AND RECOMMENDATIONS

## 5.1 RESTATEMENT OF RESEARCH GOAL

As stated in Section 1.5, the goal of this study was to develop real-time eye position prediction models capable of forecasting the eye positions for up to 10 ms (20 frames) ahead. This study is a part of a bigger project "Visual Scrambler" that aims to develop a novel EMCD system to communicate visual information to people by exploiting the current knowledge of the human visual system and making use of state-of-the-art equipment. In order to reduce the perceived stimulation latency, we build an image generator on a fast FPGA platform that is capable of generating image transformations at the level of electronics. However, for conducting several planned experiments, a further reduction in the response latency is a crucial requirement. The prediction models once ready, will be integrated into the FPGA firmware and we hypothesis that with the dynamic inflow of the eye positions, the algorithm will generate future eye positions thereby producing faster image transformations and reducing the total response latency of the system.

## 5.2 SOLUTION

To achieve the goal parameters for the "Visual Scrambler" project, we have developed several eye movement prediction models, that could predict the subjects eye positions in real-time for up to 10 ms (20 time steps) in advance, thereby reducing the effective latency. We have created the evaluation tools and carried out comparable studies for choosing the prediction algorithms. Given the specific experimental requirements, one could select the optimal prediction algorithm based on the evaluations.

## 5.3 SUMMARY OF RESULTS

The prediction models proposed in this paper can be used to predict the future eye position up to 10 ms (20 frames) in advance. Each model discussed has a higher complexity with respect to the previous algorithm so, depending on the processor capability, one could chose the optimum algorithm.

We evaluate each model performance on the data collected from the nine participants. Results have shown that the Interactive Multiple Model (IMM) estimator is the best performing model, when it comes to prediction of eye movements. The IMM estimator which uses a bank of Kalman constant velocity and constant acceleration models, can predict the eye positions with an absolute accuracy of 88% for the first prediction step. As the prediction steps decrease, the accuracy of the models decrease, with the IMM model accuracy falling to 24% for the twentieth prediction step. For an error tolerance of +2 to -2 pixels, the IMM model has a forecast accuracy of 99% for the first prediction step and a forecast accuracy of 83% for predicting the eye positions 5 ms in advance (10 frames). For the twentieth frame, the model accuracy falls to 58%. Comparing the forecast accuracy during fixational and saccadic eye movements have shown that the algorithms have a higher performance while predicting fixational eye movements, but the accuracy drops, while predicting saccade movements. We also evaluate the model performance for a lower sampling rate of 1 kHz. Here again the IMM estimator has the highest percentage forecast accuracy of 81% for the first prediction step but the performance decreases to 20% for the twentieth prediction step.

The offline evaluation of the prediction models have shown to be promising for predicting the future eye positions. However we still need to integrate the prediction algorithm into our FPGA firmware and determine if we are able to reduce the perceived latency thereby generating faster gaze contingent stimuli.

## 5.4 OUTLOOK

The idea of utilizing machine learning and statistical forecasting models for predicting eye positions is a fairly modern approach that could play a significant role, when it comes to vision science research. The field is moving in the direction of further improving the existing EMCD control as well as development of faster novel EMCD systems. Following this trend, the field promises the development of novel experiments for further recording the neural responses that correlate with fixational eye movements. The generation of faster gaze contingent stimuli along with the development of new experiments could finally put an end to the vital debate on how fixational eye movements together with the neural activities they generate, helps in our visual perception.

# REFERENCES


[1] Andrew T Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002.

[2] Michele Rucci and Jonathan D Victor. The unsteady eye: an information-processing stage, not a bug. *Trends in neurosciences*, 38(4):195–206, 2015.

[3] Susana Martinez-Conde, Stephen L Macknik, and David H Hubel. The role of fixational eye movements in visual perception. *Nature reviews neuroscience*, 5(3):229–240, 2004.

[4] M Ezenman, PE Hallett, and RC Frecker. Power spectra for ocular drift and tremor. *Vision research*, 25(11):1635–1640, 1985.

[5] Lorrin A Riggs, John C Armington, and Floyd Ratliff. Motions of the retinal image during fixation. *JOSA*, 44(4):315–321, 1954.

[6] Mary P Lord. Measurement of binocular eye movements of subjects in the sitting position. *The British journal of ophthalmology*, 35(1):21, 1951.

[7] AA Skavenski, RM Hansen, Robert M Steinman, and Barbara J Winterson. Quality of retinal image stabilization during small natural and artificial body rotations in man. *Vision research*, 19(6):675–683, 1979.

[8] Tom N Cornsweet. Determination of the stimuli for involuntary drifts and saccadic eye movements. *JOSA*, 46(11):987–993, 1956.

[9] Jacob Nachmias. Determiners of the drift of the eye during monocular fixation. *Josa*, 51(7):761–766, 1961.

[10] RW Ditchburn. The function of small saccades. *Vision Research*, 1980.

[11] Gerald Westheimer. The spatial sense of the eye. proctor lecture. *Investigative Ophthalmology & Visual Science*, 18(9):893–912, 1979.

[12] Roger HS Carpenter. *Movements of the Eyes, 2nd Rev.* Pion Limited, 1988.

[13] Gregory D Horwitz and Thomas D Albright. Short-latency fixational saccades induced by luminance increments. *Journal of Neurophysiology*, 90(2):1333–1339, 2003.

[14] Fabrizio Santini, Gabriel Redner, Ramon Iovin, and Michele Rucci. Eyeris: a general-purpose system for eye-movement-contingent display control. *Behavior Research Methods*, 39(3):350–364, 2007.

[15] Jeffrey S Perry and Wilson S Geisler. Gaze-contingent real-time simulation of arbitrary visual fields. In *Human vision and electronic imaging VII*, volume 4662, pages 57–69. International Society for Optics and Photonics, 2002.

[16] Susana Martinez-Conde, Stephen L Macknik, and David H Hubel. Microsaccadic eye movements and firing of single cells in the striate cortex of macaque monkeys. *Nature neuroscience*, 3(3):251–258, 2000.

[17] Michael B McCamy, Jorge Otero-Millan, R John Leigh, Susan A King, Rosalyn M Schneider, Stephen L Macknik, and Susana Martinez-Conde. Simultaneous recordings of human microsaccades and drifts with a contemporary video eye tracker and the search coil technique. *PLoS One*, 10(6):e0128428, 2015.

[18] Ziad M Hafed. Mechanisms for generating and compensating for the smallest possible saccades. *European Journal of Neuroscience*, 33(11):2101–2113, 2011.

[19] D MAX SNODDERLY, Igor Kagan, and Moshe Gur. Selective activation of visual cortex neurons by fixational eye movements: implications for neural coding. *Visual neuroscience*, 18(2):259–277, 2001.

[20] Fabio Richlan, Martin Kronbichler, and Heinz Wimmer. Structural abnormalities in the dyslexic brain: A meta-analysis of voxel-based morphometry studies. *Human brain mapping*, 34(11):3055–3065, 2013.

[21] PIA Rämä and Thierry Baccino. Eye fixation–related potentials (efrps) during object identification. *Visual Neuroscience*, 27(5-6):187–192, 2010.

[22] Sven-Thomas Graupner, Sebastian Pannasch, and Boris M Velichkovsky. Saccadic context indicates information processing within visual fixations: evidence from event-related potentials and eye-movements analysis of the distractor effect. *International Journal of Psychophysiology*, 80(1):54–62, 2011.

[23] Fabio Richlan, Benjamin Gagl, Sarah Schuster, Stefan Hawelka, Josef Humenberger, and Florian Hutzler. A new high-speed visual stimulation method for gaze-contingent eye movement and brain activity studies. *Frontiers in systems neuroscience*, 7:24, 2013.

[24] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons, 2012.

[25] Kyung-Bin Song, Young-Sik Baek, Dug Hun Hong, and Gilsoo Jang. Short-term load forecasting for the holidays using fuzzy linear regression method. *IEEE transactions on power systems*, 20(1):96–101, 2005.

[26] Eva Ostertagová. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 2012.

[27] Howard J Seltman. Experimental design and analysis, 2012.

[28] David G Kleinbaum, LL Kupper, KE Muller, and A Nizam. Regression diagnostics. *Applied regression analysis and other multivariable methods*, 2, 1998.

[29] Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

[30] Tilmann Gneiting, Larissa I Stanberry, Eric P Grimit, Leonhard Held, and Nicholas A Johnson. Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictions of surface winds. *Test*, 17(2):211, 2008.

[31] V Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, 2(3):24–33, 2003.

[32] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[33] Nathan Funk. A study of the kalman filter applied to visual tracking. *University of Alberta, Project for CMPUT*, 652(6), 2003.

[34] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, 22(12):985–1003, 2003.

[35] Dan Simon. Kalman filtering. *Embedded systems programming*, 14(6):72–79, 2001.

[36] Roger Labbe. Kalman and bayesian filters in python. *Chap*, 7:246, 2014.

[37] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, 1995.

[38] Joana Barbosa Bastos Gomes. An overview on target tracking using multiple model methods. *Instituto Superior Tecnico*, 2008.

[39] Andrew Kiruluta, Moshe Eizenman, and Subbarayan Pasupathy. Predictive head movement tracking using a kalman filter. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):326–331, 1997.

[40] Ian Reid and Hilary Term. Estimation ii. *University of Oxford, Lecture Notes*, 2001.

[41] JD Parker and WD Blair. Use of target-oriented process noise in tracking maneuvering targets. Technical report, NAVAL SURFACE WARFARE CENTER DAHLGREN VA, 1992.

[42] Christopher Hide, Terry Moore, and Martin Smith. Adaptive kalman filtering for low cost ins/gps. In *Proceedings of the 15th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2002)*, pages 1143–1147, 2002.

[43] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[44] Jason L Williams. Gaussian mixture reduction for tracking multiple maneuvering targets in clutter. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND ..., 2003.

[45] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[46] Henk AP Blom. An efficient filter for abruptly changing systems. In *The 23rd IEEE Conference on Decision and Control*, pages 656–658. IEEE, 1984.

[47] Henk AP Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, 33(8):780–783, 1988.

[48] Thiagalingam Kirubarajan and Yaakov Bar-Shalom. Kalman filter versus imm estimator: when do we need the latter? *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1452–1457, 2003.

[49] Shuhang Wang, Russell L Woods, Francisco M Costela, and Gang Luo. Dynamic gaze-position prediction of saccadic eye movements using a taylor series. *Journal of vision*, 17(14):3–3, 2017.

[50] Jeffrey L Anderson and Stephen L Anderson. A monte carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, 127(12):2741–2758, 1999.

[51] Changqing Cheng, Akkarapol Sa-Ngasoongsong, Omer Beyca, Trung Le, Hui Yang, Zhenyu Kong, and Satish TS Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. *Iie Transactions*, 47(10):1053–1071, 2015.

[52] Fred Daum. Nonlinear filters: beyond the kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, 20(8):57–69, 2005.