Computer Vision Based Interface Sensors for Oil Sands Primary Separation Vessel

by

Agustin Vicente

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Process Control

Department of Chemical and Materials Engineering

University of Alberta

Abstract

In the oil sands extraction process, bitumen (crude oil) is separated from the sands in the Primary Separation Vessel (PSV) through a water-based gravity separation process. The interface between froth (crude oil) and middlings (water and sand) is the most important control variable in the PSV operation. Bitumen recovery and downstream operations are critically dependent on interface level measurement and control. Most of the traditional PSV level instruments have deficient service factors, limiting the implementation of automatic control. Therefore, we proposed novel and robust computer vision based methods to estimate the froth-middlings interface level on video frames captured from a PSV's sight glasses camera.

The first chapter of the thesis discusses the computer vision as a knowledge basis for the proposed work. Typical image processing and analysis methods are described, and they provide the foundation for the subsequent chapters.

The subsequent chapters propose several approaches for the interface level detection. As the first approach, we present the froth-middlings interface level detection on single frames (static image processing). The level is detected based on edge detection performed on the frames, in which traditional filters are proposed to smooth the images through model-based image restoration.

Next, we develop and implement a robust computer vision based method to estimate the froth-middlings interface level in PSV, in which we additionally consider the dynamics of a set of consecutive frames to improve the level estimation.

The algorithm processes the online video frames of a camera mounted on PSV sight glasses, and the level is detected based on a combined operation of edges and motion detection in a set of consecutive frames (static and dynamic image processing). In addition, the algorithm uses reliability analysis to detect the environmental conditions that may limit the level estimation, and a time-based sliding window analysis of the level measurements is proposed. Industrial application results show that the proposed computer vision algorithm is more accurate and reliable when compared to other instruments, as well as more robust against the process and environmental abnormalities.

Finally, advanced filters with finite impulse response (FIR) structure are proposed and developed to improve the image restoration and object tracking process. We address the problem of smoother design for state estimation based on a finite number of measurements collected over a finite horizon. Three different FIR smoothing algorithms are proposed using the maximum likelihood FIR estimation, which is robust against uncertain noise statistics and modeling parameters. The FIR algorithms are applied to the image restoration and level tracking problem in PSV, and they show better robustness against modeling uncertainties than traditional IRR filtering approaches.

This thesis is dedicated to my wife, Fernanda Carozzi, my parents, Nestor Vicente and Ana Maria Esteban, and my brothers, Augusto Vicente, Federico Vicente, and Ariel Vicente, for their love, support and encouragement during my program.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Biao Huang, for the continuous support, advice, and encouragement during my M.Sc. program. He gave me the great opportunity the explore the interesting academic research in the Computer Process Control (CPC) group. With Dr. Huang guidance on my work direction and his patient to explain the research concepts, I was able to complete this thesis. Also, I would like to thank Dr. Huang for letting me get involved in various academic and industrial activities, which helped me to expand my background, and develop my professional and interpersonal skills.

It has been an honor to be part of the CPC group, working with such talent, grateful and professional people. Particularly, I would like to especially thank two great members: Dr. Shunyi Zhao and Rahul Raveendran. They helped me a lot during my research journey, providing ideas, discussions, and revisions to my research and industrial work presented in this thesis. A very special gratitude to Artin Afacan and Zheyuan Liu for setting up the PSV experimental equipment in the CPC lab. Also, I would like to thank Dr. Fadi Ibrahim, for his support and collaboration during my work in the CPC lab. I would like to thank all my present and previous colleagues for their help, listed but not limited as the follows: Seraphina Kwak, Alireza Kheradmand, Yanjun Ma, Lei Fan, Mengqi Fang, Anahita Sadeghian, Ming Ma, Hashem Alighardashi, Rui Nian, David Scott, Dr. Hariprasad Kodamana, Dr. Yousef Alipouri, Dr. Nabil Magbool Jan, Dr. Kirubakaran Velswamy, and many else.

I would like to thank Warren Mitchell, Hailei Jiang, Anuj Narang, and Shabnam Sedghi, for their support and help during my industrial development work at

Spartan Controls. The constant discussion helped me to improve the quality of the computer vision system that I described in the thesis. In my interaction with them, I also gained a substantial amount of knowledge about process control applications development and implementation for the oil sand processes.

A sincere gratitude to all my friends for their encouragement, inspiration and companion during my program, listed but not limited as the follows: Anuar Caldera, Daniel Moran, Vadim Kislitsin, Giselle Uzcategui, Linda Contreras, Hossein Shahandeh, Farshad Mohammad, Carlos Blanco, Abdulrahman Albeladi, Ramon Rosales, Tony Bruinje, Larissa Bruinje, Dante Gonzalez, Alejandro Hintze, Luciana Erregue, and many else. Also, an especial thank to professors Mauricio Sacchi and Patricio Mendez at the University of Alberta, for their inspiration and encouragement to follow my M.Sc. program.

Also, I thank the Natural Sciences and Engineering Research Council of Canada and Alberta Innovates for the financial support of the Industrial Research Chair program.

Last but not the least, I would like to give my special gratitude to my family for their constant love, support, and encouragement. Especially, a lovely and great appreciation to my wife, Fernanda Carozzi, who is a Ph.D. student in Geophysics at the University of Alberta. She was my principal support and motivation to pursue my academic research program when we came to Edmonton.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Acronyms

DCS	*Distributed Control System*
DP Cell	*Differential Pressure Cell*
FIR	*Finite Impulse Response*
IIR	*Infinite Impulse Response*
IP	*Internet Protocol*
ISS	*Internet Information Service*
KF	*Kalman Filter*
KS	*Kalman Smoother*
LSE	*Least Square Estimation*
MAP	*Maximum a Posteriori*
ML	*Maximum Likelihood*
MRF	*Markov Random Field*
ML-FIR	*Maximum Likelihood Finite Impulse Response*
OPC	*Open Protocol Communication*
PSV	*Primary Separation Vessel*
RMSE	*Root Mean Square Error*
ROI	*Region of Interest*
RTS	*Rauch-Tung-Striebel*
SG	*Sight Glasses*
SNR	*Signal to Noise Ration*
VPA	*Video Processing Application*
WF	*Wiener Filter*
w.r.t	*with respect to*

Chapter 1

Introduction

1.1 Background

Oil sands are a type of unconventional petroleum deposit that consists of a mixture of crude bitumen, silica sand, clay minerals, and water. Alberta's oil sands amount to 10 % of the world's total oil reserves with a production of more than 3 million barrels per day [1]. There are two predominant processes to extract the bitumen from the oil sands, 1) surface mining and 2) in situ thermal extraction. In surface mining, oil sands closer to the surface is mined and sent to the bitumen extraction facility. In the in situ thermal extraction, deposits below the surface are extracted by means of steam injection [2].

In the bitumen extraction facility for the surface-mined oil sands deposit, bitumen is separated from the sand using a water-based gravity separation process. The Primary Separation Vessel (PSV) or Cell (PSC) is the heart of the extraction process where up to 90% of the bitumen is recovered [2]. Figure 1.1 shows the cross-sectional view of a typical PSV. PSVs are large vessels with a conical bottom and have a steep side slope that helps the gravity separation process. A feedwell pipe feeds the oil sands slurry (mined and crushed oil sands feed mixed with hot water) into the vessel right below the froth layer where the separation process begins. The heavy solids sink to the bottom of the vessel, while the lighter bitumen floats to the top. Thus, three different segregated phases are generated in the separation process: 1) Overflow: a clean froth product that contains about 50-60% bitumen, 2) Middlings: a fine slurry composed of mostly water with about 2-4% bitumen, 3) Underflow: a coarse tailing consisting of at least 50% solids and residual bitumen (less than 1%)[2, 3]. The froth layer is the most important one where the majority of bitumen is recovered. The middlings layer contains a mixture of bitumen, water, and solids, and usually goes to further reprocessing to increase bitumen recovery. The tailing layer is composed of coarse solids and is pumped out to the tailings treatment plant.

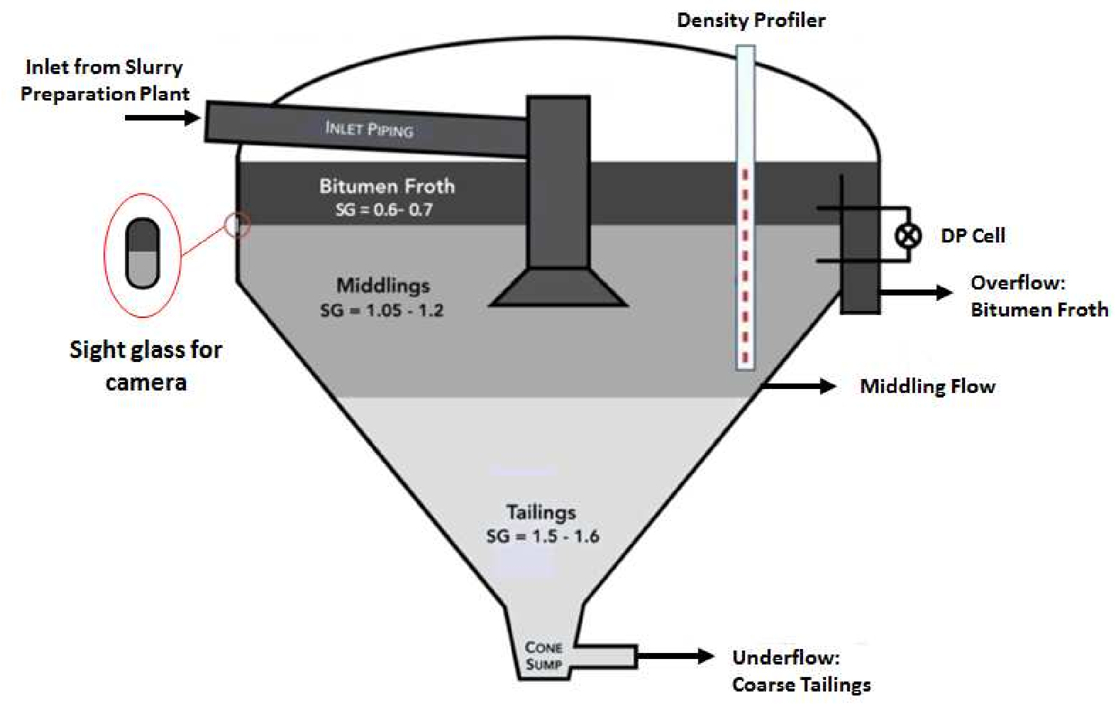


Figure 1.1: *Industrial Primary Separation Vessel(PSV) schematic. The sight glasses are installed within the froth-middlings zone.*

The interface between the froth and middlings layers is the most important control variable in the process. Bitumen recovery and downstream operations are critically dependent on the froth-middlings interface level measurement and control.

A high interface level increases the presence of solids in the froth layer deteriorating the bitumen quality. While a low interface level increases the presence of bitumen in the tailings, leading to bitumen loss and environmental consequences. Therefore, good and stable control of the froth-middlings interface level is necessary for the efficient bitumen recovery and reduction of process variations in the downstream operations [4–6].

Froth-middlings interface control can be achieved if it is measured accurately. Current best practices in the industry for the froth-middlings interface measurement include differential pressure (DP) cells, nucleonic density profilers, and capacitance probes. However, these instruments do not have a desirable service factor, limiting the application of automatic control in PSV operation [3, 7]. DP cells are more suitable for applications with uniform density fluid columns; however, they suffer when it comes to multiphase fluid columns such as the PSVs. Nucleonic profilers have a high installation and maintenance cost, prone to frequent errors caused by substance build-up on its surface, are subject to safety risk associated with the radiation loss, and need periodical inspection and approvals. Capacitances probes usually lose their sensitivity over time due to substance build-up on its surface [8]. Therefore, the use of traditional instruments to measure the PSV froth-middlings interface level accurately remains as a challenge. In most PSV applications, the froth-middlings interface is being monitored and controlled manually by the operators.

Typically, PSVs are equipped with sight glasses on the side of the vessel close to the froth-middlings zone, as shown in Figure 1.2, and they span a length of two meters from the top part of the froth launder. Sight glasses allow the froth-middlings interface level and the separation process dynamics to be visually monitored. The depth of the froth layer (interface level), the quality of the bitumen froth, and the clarity of the interface can be inferred through the sight glass visuals.

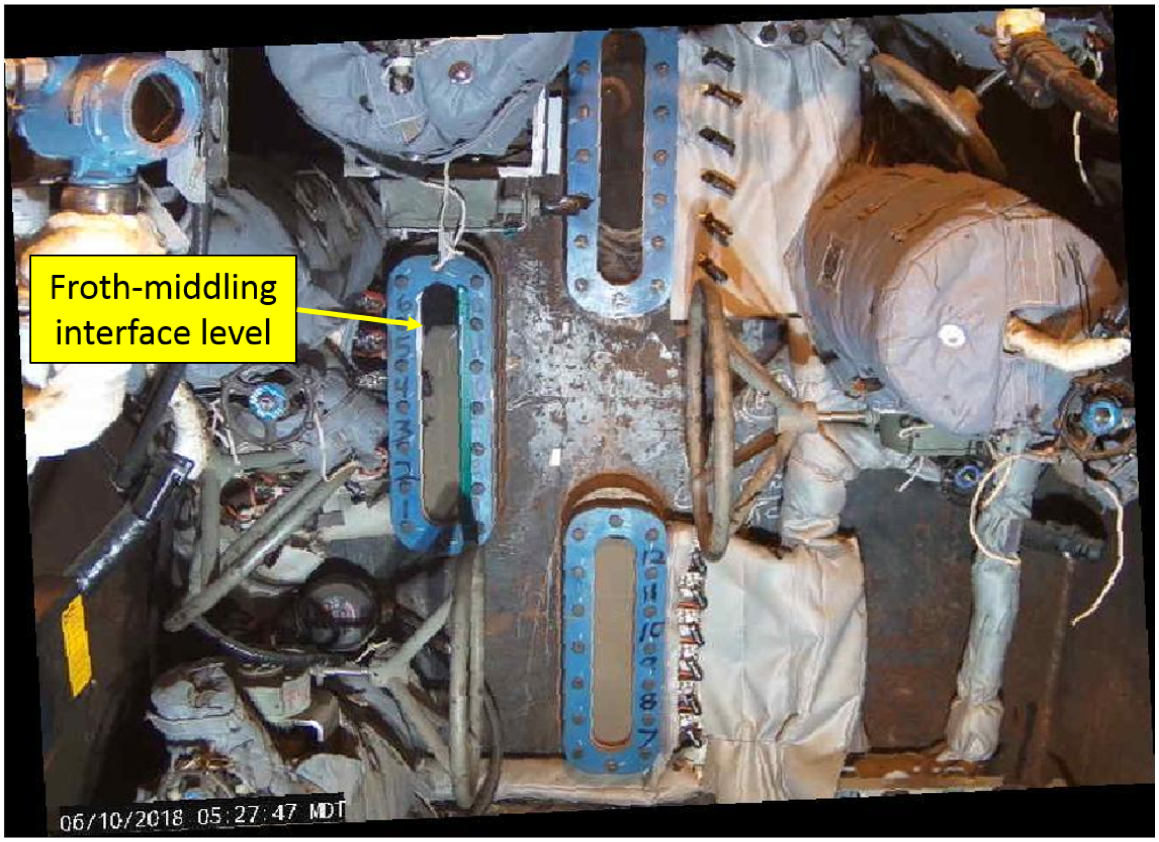


Figure 1.2: *PSV Sight glasses visuals. Acquired from a video camera mounted on PSV sight glasses.*

Therefore, the implementation of video cameras to capture the sight glass visuals and estimate the froth-middlings interface level from the video stream is a method that is gaining popularity during the recent years [3–5, 7, 9, 10]. In [7], a Markov Random field image segmentation based method was proposed to find the interface level by partitioning the image into oil and water regions. However, the segmentation based methods do not provide satisfactory results since they are highly affected by the lighting conditions, gradual variations of pixel intensities along the images and presence of shadows and glares. As lighting conditions may change, the algorithm needs to be robust to lighting changes. In addition, the method requires several image processing iterations to obtain a clean segmented image, which makes the method computationally expensive and undesirable for online implementation. In [3], an edge detection method, which is more robust to lightening condition changes or intensity variations on the image, is employed to detect the level interface. However, the use of just the edge detection method alone would produce an ambiguous output when sight glasses have significant stains and may require additional computationally expensive filtering techniques such as particle filtering [9]. For example, in [4], several different image processing filters were proposed to remove the noise and reduce the spurious edges generated by the method. In addition, an extended Kalman Filter is proposed to smooth the level measurements before they are sent to the level controllers. In [9], an image differentiation method was proposed to compute a confidence value for the level estimation. The method works well for ideal interface images which are relatively noise free. In the real world, the noise comes from different sources including the camera acquisition noise, fine sand particles moving in the separation process, stains that appear and disappear on the sight glass, and shadows/glares on the sight glasses.

1.2 Motivation

In this thesis, we are motivated to investigate and develop a new approach to improve the accuracy of the camera froth-middlings interface level detection. The objective is to exploit the information on the sight glasses visuals and use automatic computer vision techniques on the images obtained from the video camera.

Most of the previous methods were solely based on static image segmentation, such as region segmentation or edge segmentation, which search for the level location by processing the information in each frame separately. As a result, they had to deal with the problem of noisy detections (spurious edges in [3]) or difficulties to classify the oil and water regions ([3] and [7]). However, the level is a feature in the image that has dynamics in time and thus by considering methods that analyze the motion in subsequent frames we can extract additional information about the level location. Therefore, we are motivated to combine the information from static and dynamic image segmentation processing methods to obtain an image output that can provide a more accurate level location with less computational load. In addition, there are abnormal process scenarios that affect the level detection which have not been discussed and addressed in the previous works. Abnormal scenarios such as the stains on the sight glasses, blur or fuzzy level interface, froth and middlings phases with non-homogenous pixel intensities, and level switch from one sight glass to the other. These were in fact the main causes for the noisy detections when using most of the previously proposed algorithms.

Besides, the previous works do not provide the framework to address the industrial conditions that affect the reliability of the level detection when using a video camera sensor. For example, scenarios such as people/objects blocking the camera, camera motion or vibrations, lighting changes, or poor image quality.

Finally, we are motivated to investigate the state estimation theory and filtering design using finite impulse response (FIR) structures. FIR filters utilize finite measurements collected over the most recent interval, and thus have some advantages with respect to infinite impulse response (IRR) filters, such as robustness to modeling and noise uncertainties, guaranteed stability, and linear phase. We are motivated to implement the state estimation through FIR structures to improve the image restoration and level tracking problems in PSV.

The PSV laboratory experiment shown in Figure 1.3 is employed to study the froth-middlings interface behavior. The PSV experiment is located in the Computer Process Control Lab at the University of Alberta.

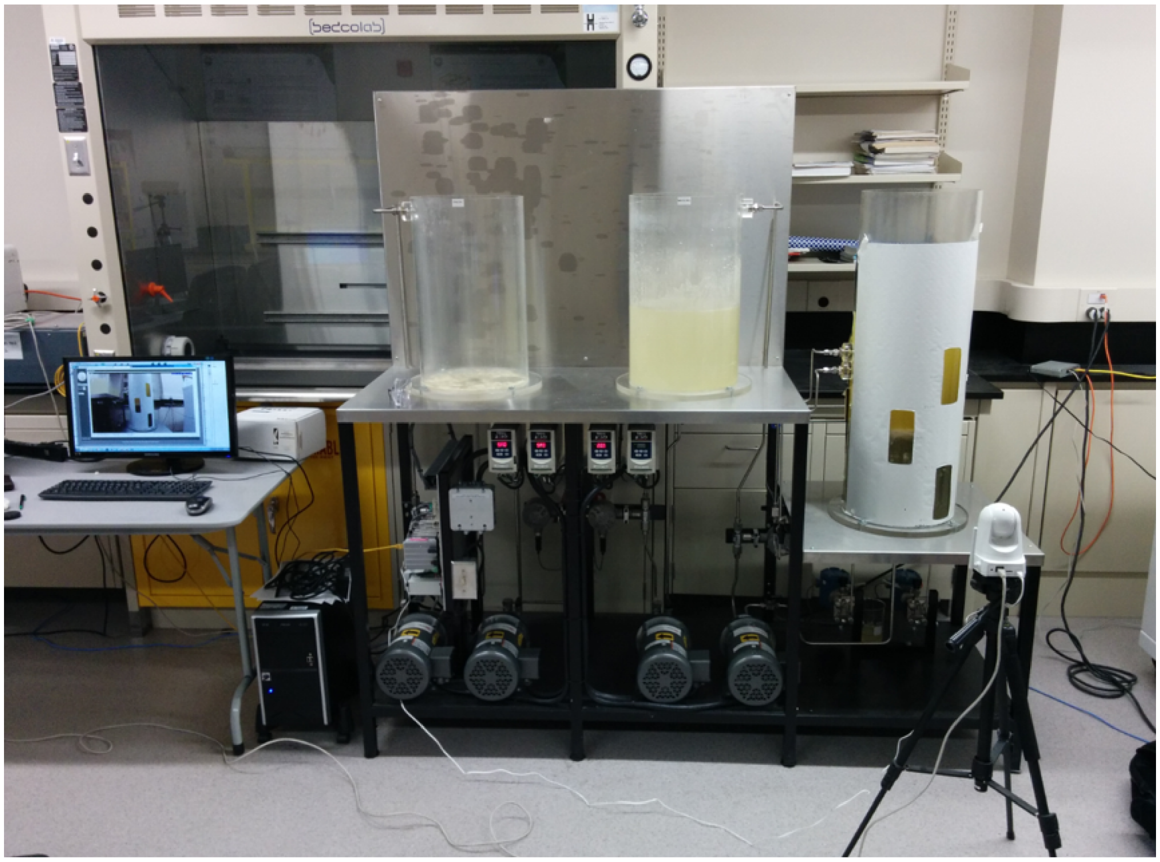


Figure 1.3: *PSV experimental setup in the Computer Process Control Lab at University of Alberta.*

An Internet Protocol (IP) video camera is used to acquire the images of the froth-middlings interface from the PSV experimental tank. Figure 1.4 displays the PSV experimental tank with and without sight glasses. The interface between canola oil and water mimics the real froth-middlings interface level in the PSV.

In addition to the PSV experimental setup, video data from industrial PSV sight glasses, as shown in Figure 1.2, are employed to test the video processing algorithms. The videos are recorded using a camera mounted on PSV sight glasses and then processed in the laboratory. MATLAB is employed to design and test the video processing algorithms.

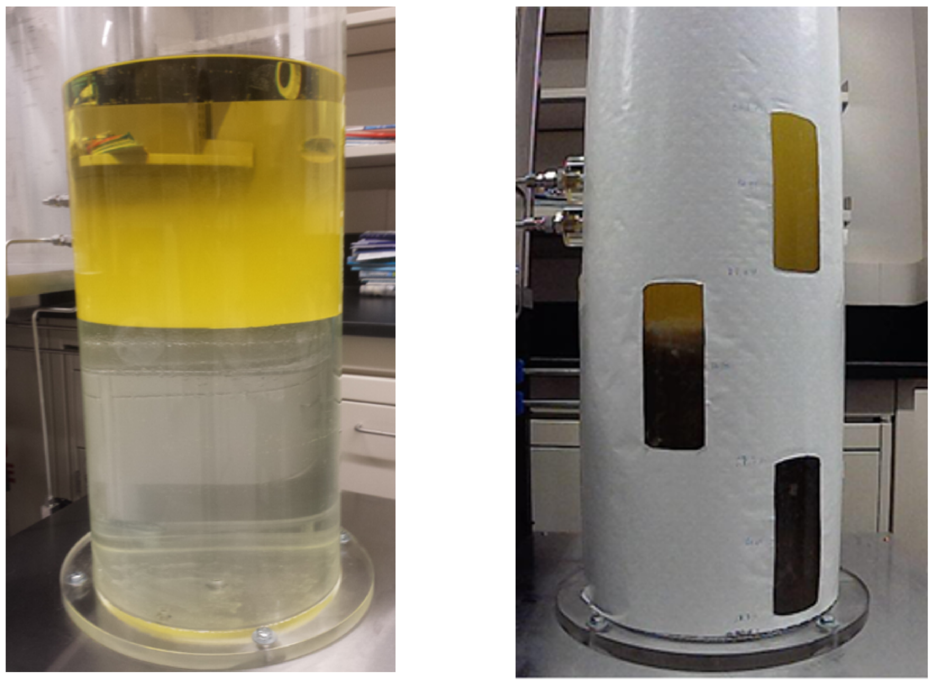


Figure 1.4: *PSV experimental tank. Left: Open PSV experimental tank. Right:PSV experimental tank with three sigh glasses.*

1.3 Thesis contributions

This thesis contributes mainly to the development of approaches for interface level detection using image processing and analysis techniques. Specific methods and algorithms are proposed to improve the detection of the interface level on the frames. The detailed contributions of this thesis are listed as follows:

1. Implemented an image segmentation based on edge detection to find the level interface in the frames (static image processing). The interface level is estimated based on the maximum value of the vertical profile of the segmented edge image.

2. Proposed image smoother algorithms with Infinite Impulse Response structure to remove the noise of the raw interface images. Image smoothing improves the performance of image segmentation based on edge detection to find the interface level.

3. Implemented an image segmentation based on motion detection to find the level interface in a set of consecutive frames (dynamic image processing). The objective is to capture spatial information (static image processing) about the level and observe its dynamics in consecutive temporal frames. The interface level is estimated based on the maximum value of the vertical profile of the segmented image.

4. Developed and implemented a computer vision method to estimate the forth-middlings interface online based on image and data processing techniques performed on video frames of the PSV sight glasses. The vision sensor is designed with an algorithm that consists of three main steps, in which the inputs are the video frames, and the outputs are the estimated level values. The video processing algorithm has three important steps: 1- Static and dynamic image processing step that detects the interface level in the frames, 2- Reliability analysis step that prevents inaccurate level estimations caused by the scenarios that affect the sight glass visuals , and 3- Time-based sliding window analysis of the level measurements to remove outliers, smooth and track the level location.

5. Developed forward-backward smoothing algorithms with Finite Impulse Response (FIR) structure to denoise the raw interface images. To achieve this objective, we exploit the iterative Maximum Likelihood Finite Impulse Response (ML-FIR) estimation to obtain estimates for each point inside the finite horizon. Then, we combine these estimates by employing the formulas for the optimal combination of two independent estimates, known as the forward-backward smoother equations. The proposed FIR smoothers have three important advantages over the traditional smoothers. First, the proposed methods are robust to uncertainties of modeling parameter and noise statistics. Second, the algorithms are independent of the initial states and covariance in each finite horizon because the initial estimates are obtained through the batch Maximum Likelihood (ML) calculation. Finally, the FIR

smoothers can have a faster and more accurate capture of the local dynamic changes in the data as they only consider the most recent measurements.

1.4 Thesis outline

The remainder of this thesis is organized as follows:

Chapter 2 is an introduction to image processing and analysis methods. It provides the mathematical background to understand the subsequent chapters. We particularly focus on the computer vision algorithms which typically include the following components: image acquisition, image prepossessing, image processing, image analysis, and decision making. Feature extraction and image segmentation are the most important steps during low-level image processing. They can be performed using the static information in one single frame, such as region segmentation or edge detection, or using dynamic information on several frames, such as motion detection.

Chapter 3 presents the problem of detecting the froth-middlings interface level on single frames (Static image processing). An image segmentation based on edge detection is proposed to detect the interface level. However, images are generally corrupted by the noise coming from different sources. Therefore, it is crucial to remove this noise to improve the performance of the level edge detector. Traditional model-based filters are investigated and applied to the interface level image to remove the noise and find the level through edge detection.

Chapter 4 describes a computer vision based method that uses image and data processing techniques to estimate the froth-middlings interface level in PSV. The algorithm process the online video frames of a camera mounted on PSV sight glasses, and it is based on a combination of static and dynamic image processing steps. The objective is to capture spatial information (Static image processing) about the level and observe its dynamics in a set of consecutive temporal frames. In addition, the algorithm uses a reliability analysis on the images to detect environmental conditions that limit the level estimation, and a time-based sliding window analysis of the level measurements to remove outliers, smooth and track the location of the level. The computer vision system was implemented in two industrial PSVs and it has been found more accurate and reliable when compared to other instruments, and more robust against process and environmental abnormalities.

In Chapter 5, filter and smoothers with FIR structure are studied and implemented on the image restoration and object tracking problems. We address the problem of smoother designs for state estimation based on a finite number of measurements collected in a finite estimation horizon. Three different finite impulse response (FIR) smoothing algorithms are proposed using the maximum likelihood FIR estimation, which is robust against uncertain noise statistics and modeling parameters, and also independent of the initial states of each finite horizon. Moreover, we provide equivalent but iterative Kalman-like structures of these algorithms for practical implementation. The applications of the proposed algorithms to the interface image restoration and level tracking problems are presented, and they show better robustness against modeling uncertainties than traditional filtering/smoothing approaches.

Chapter 6 draws the conclusions of the thesis and provides some future works directions.

Chapter 2

Introduction to Image Processing and Analysis

This chapter is an introduction to the field of image processing and analysis methods in the literature. It provides the foundations to comprehend the image operations of the subsequent chapters. Especially, it focuses on computer vision algorithms which are concerned with the automatic extraction, analysis, and understanding of useful information from a single image or a sequence of images. Finally, it describes different feature extraction and segmentation methods, which are the most important task to enhance the features of interest in the images. They can be performed by using information from one single static frame (region segmentation or edge detection), or by using information from several frames (motion detection).

2.1 Introduction

Image analysis is a computer-based process of extracting quantitative information from images. The process begins with the input of an image and ends with the output of numerical data. This process distinguishes image analysis from image processing where both input and output are in the form of an image. Image processing is the means by which the input image is modified by mathematical algorithms to generate an output image that is enhanced in some way. For example, by using an image processing operation, we can enhance the edges or reduce the noise on the image, while by using an image analysis operation, we can find the location (pixel coordinates x and y) of an object in the image. Image processing is often used to prepare the images before image analysis [11].

The field of image processing and analysis is diverse, and there are applications in multiple fields (Engineering, robotics, medical imaging, agriculture, food processing, video surveillance, object tracking, etc.) [12]. In this chapter, we introduce the mathematical methods and algorithms that are used to extract meaningful information from the images. It also provides the necessary mathematical basis and background about image processing and analysis for the subsequent chapters.

2.2 Digital image representation

A grayscale image could be defined as a two-dimensional function, $f(x, y)$, where x and y are the spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y and the intensity values of f are all finite, discrete quantities, we call the image a digital image. Thus, a digital image is composed of a finite number of elements called *pixels*, each of which has a particular location and value. The amplitude of the pixels is often quantized to 256 levels (which can be represented by eight bits). Each level is commonly denoted by an integer, with 0 corresponding to the darkest level and 255 to the brightest [13]. However, for image analysis simplicity it is convenient to rescale the gray intensity level from 0 to 1, where 0 correspond to the darkest level, 1 to the brightest level, and any number in between is a shade of gray. Figure 2.2 represents the PSV froth-middlings interface level in a bi-dimensional and three-dimensional map.

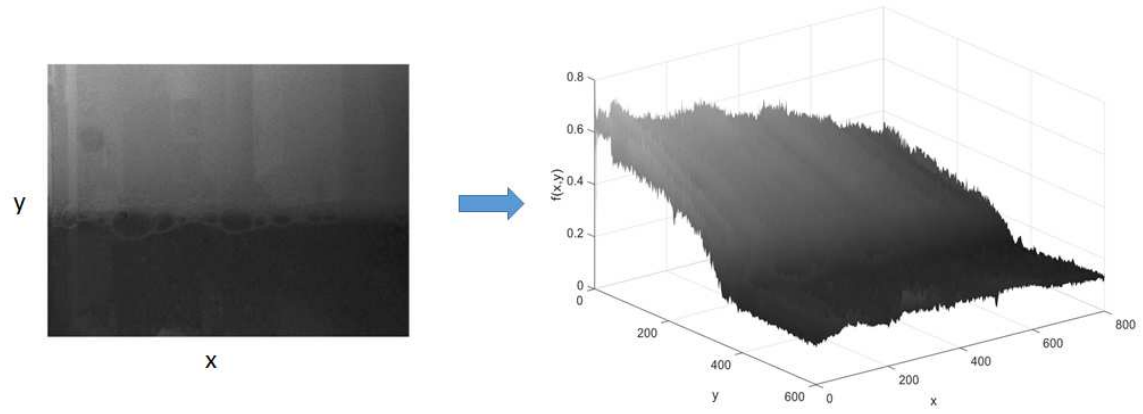


Figure 2.1: *Intensity or gray level representation of the froth-middlings interface level's image. Left: Grayscale bidimensional display. Right: Grayscale tridimensional display*

2.3 Type of image processing and analysis algorithms

There is no general agreement among researchers regarding where the scope of image processing ends, and how it distinguishes from other related areas, such as image analysis and computer vision [11–13]. However, we can organize and divide the image algorithms into four main categories [3, 14]:

- **Image Processing:** Comprises a broad variety of methods that operate on images to produce another image [11]. It is the process of enhancing the input image (colorizing, enhancing the contrast, image sharpening, etc.) in a manner that it is more suitable for image analysis. In most cases, the input to and the output of the algorithm is an image. For example, the edges of structures may be sharpened, or the image may be given an enhanced contrast or reduced noise.

- **Image Analysis:** It is the process of obtaining numerical data from images.

This is usually accomplished by a combination of measurements and processing operations. The data obtained by image analysis may subsequently be evaluated statically, or used to generate a graph or other visualization information.

- **Computer Vision:** It is a general class of algorithms which extract information from images. The input to the algorithm could be a single image or a sequence of images, or even images from multiple angles of a scene of interest. In Computer Vision the output of the algorithm does not need to be an image, as in Image Processing. Instead, the output could be a detected feature such us the location (x and y coordinates) of an object in an image.

- **Machine Vision:** It is the use of computer vision algorithms for industrial applications. Computer vision algorithms only extract information from an image, while Machine vision involves using this knowledge to manipulate an industrial element, such as a control valve, a pump, or a robotic arm.

Among all of them, computer vision algorithms are the ones that have a particular interest in the case of the PSV forth-middlings interface level detection. People utilize their eyes and brains to detect their general surroundings for different purposes. Computer vision is the science that intends to give a comparable ability to a machine or computer. For example, operators use their eyes to detect the interface level location through the sight glasses in the PSV and then manually control the separations process. However, the objective of this work is to replace this operator's eyes task by a computer vision system that can achieve the same results. Thus, in the following sections, we introduce the computer vision science and the typical method and algorithms utilized in the design.

2.4 Computer vision

Computer vision is concerned with the automatic extraction, analysis, and understanding of useful information from a single image or a sequence of images. From the perspective of engineering, it seeks to automate tasks that the human visual system can do [15]. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. For instance, a typical computer vision task is object detection. It is the process of finding instances of real-world objects such as faces, bicycles, and buildings in images or videos. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category.

Essentially, a computer vision system depends on its application. Some systems are a simple algorithm that solves a specific problem, while others constitute a larger design. However, there are typical functions which are found in many computer vision systems as Figure 2.2 depicts: Image Acquisition, Image Pre-processing, Image Processing, Image Analysis, and Decision making.

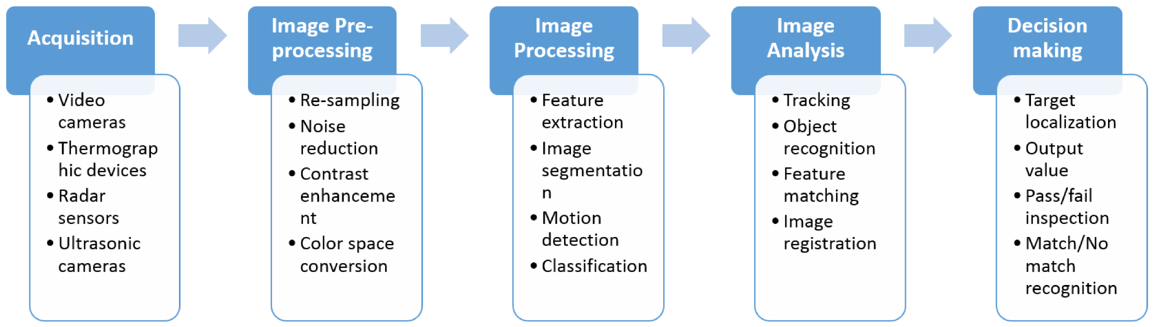


Figure 2.2: *Typical steps in a computer vision algorithm*

1. Image Acquisition: It is the process of translating the analog world around us into pixel data composed of numbers, interpreted as digital images. Different

tools have been created to build such datasets: Cameras, Webcams, Digital compact cameras, 3D cameras, laser, etc.

2. Image Pre-processing: It is the process of removing the variability on the pixel level without losing essential information about the image. Examples are data resampling, noise reduction, or contrast enhancement.

3. Image Processing: It is the process of using algorithms to infer low-level information on parts of the image. This type of information is characterized by image edges, point features or segments, for example. They are all the basic geometric elements that build objects in images. It usually involves advanced applied mathematics algorithms and techniques, such as edge detection, segmentation, classification, and feature detection and matching.

4. Image Analysis: It corresponds to the analysis and understanding of the data, which will allow the decision making. High-level algorithms are applied, using both the image data and the low-level information computed in previous steps. Examples of high-level image analysis are 3D scene mapping, object recognition, or object tracking.

5. Decision making: It provides the final decision output for the required application. For example, pass/fail on automatic inspection applications, match/no-match in recognition applications, or space coordinates (x, y) in tracking applications.

When developing computer vision algorithms, one has to face various issues and challenges, related to the very nature of the data or event for which the application has to be created and its context: 1. Noisy or incomplete data, 2. Real-time processing, or 3. Limited resources such as computational power and memory. Current research is focused on addressing these challenges to make the algorithms more robust and efficient under difficult conditions. In the following sections, we describe some of the most important methods that can be utilized in a computer vision algorithm for interface level detection. They provide a mathematical image processing basis for the following chapters of the thesis.

2.5 Feature extraction

Feature extraction is the process of detecting and isolating a desired portion or shapes (features) from a digitalized image or video. In general, a feature is defined as a region in an image that has some particular characteristic of interest. It refers to the process of extracting some quantitative information of interest from an image. When the input data to an algorithm is too large to be processed, and it is suspected to be redundant (e.g., the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data so that the desired task can be performed by using this reduced representation instead of the complete data.

In the froth-middlings image, the feature of interest corresponds to the froth-middlings interface region. The interface is a feature that lies along the vertical axis of the image. In other words, if we take one set of column data from the image we would have all the information that we need to study the system and extract the interface level feature.

Figure 2.3 has 3 images taken on different conditions. Image 1 seems to have the clearest conditions. However, Image 2 is blurred by dark lighting conditions, and Image 3 is occluded by sticky foam over the tank wall.

One set of column data from each image of Figure 2.3 was selected and plotted in

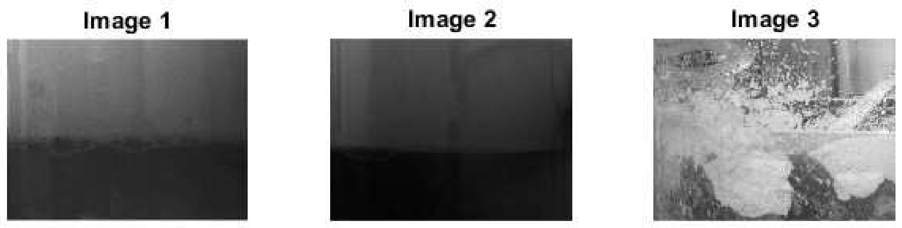


Figure 2.3: *Forth-middlings interface images with different environmental and process conditions.*

Figure 2.4. As we can observe, the vertical intensity profiles display the different types of conditions observed in Figure 2.3.

Moreover, if we focus on one of the images, for example, Image 1, we will observe

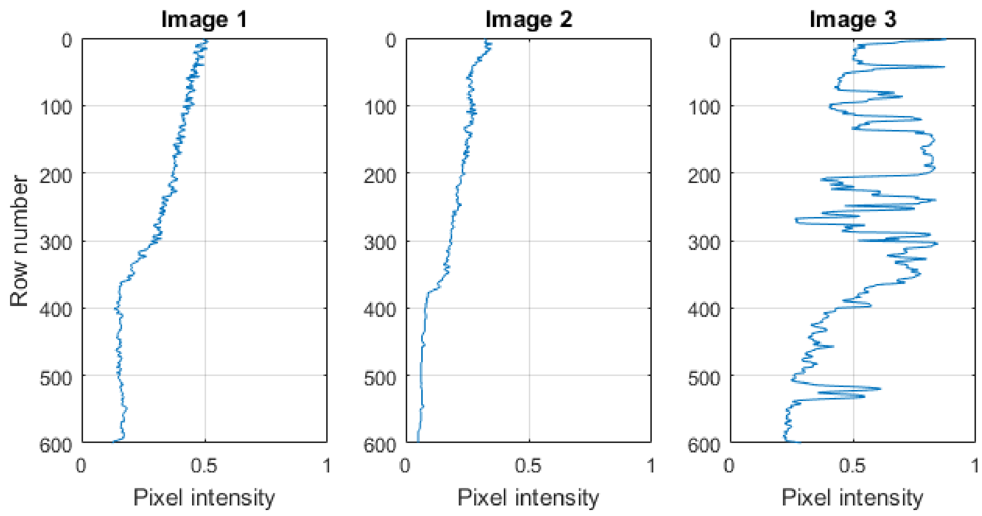


Figure 2.4: *Vertical intensity profiles of froth-middlings interface level images.*

that the vertical intensity profile may change along different longitudinal regions. For example, Figure 2.5 depicts the Image 1 vertical profile taken from different columns. As we can see, the vertical profile could vary by different conditions, such as noise, uncertainties or process properties in the image.

In summary, we can state that images are always corrupted with noise produced during the acquisition process (sensor, camera, and electronic circuits). This is a

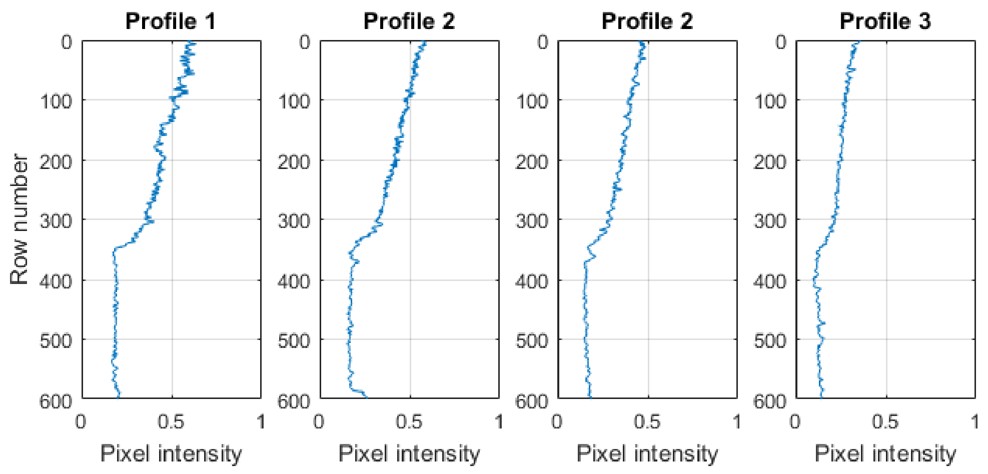


Figure 2.5: *Image 1 vertical intensity profile at different longitudinal regions.*

typical additive and independent noise at each pixel, and independent of the signal intensity. In addition, external environmental conditions, such as changing lighting conditions, shadows, and glares, can produce colored noise. This type of noise can appear on some specific regions of the image and has a neighborhood dependency. Finally, physical obstructions coming from the internal process (such as foam, bubbles or sticky oil) can generate unexpected disturbances on image properties.

2.6 Image segmentation

Image segmentation is one of the most critical tasks of image analysis. It has the objective of extracting information (represented by data) from an image via image segmentation, object representation and feature measurement [12]. Image segmentation is often described as the process that subdivides an image into its constituent parts and extracts those parts of interest (objects). The objective is to simplify or change the image into something easier to analyze.

Gray level image segmentation is generally based on one of two basic properties of gray level values in images: discontinuity and similarity. Thus, two categories of algorithms can be distinguished as in Figure 2.6: the boundary-based ones that detect object contours explicitly by using the discontinuity property and the region-based ones that locate object areas explicitly according to the similarity property.

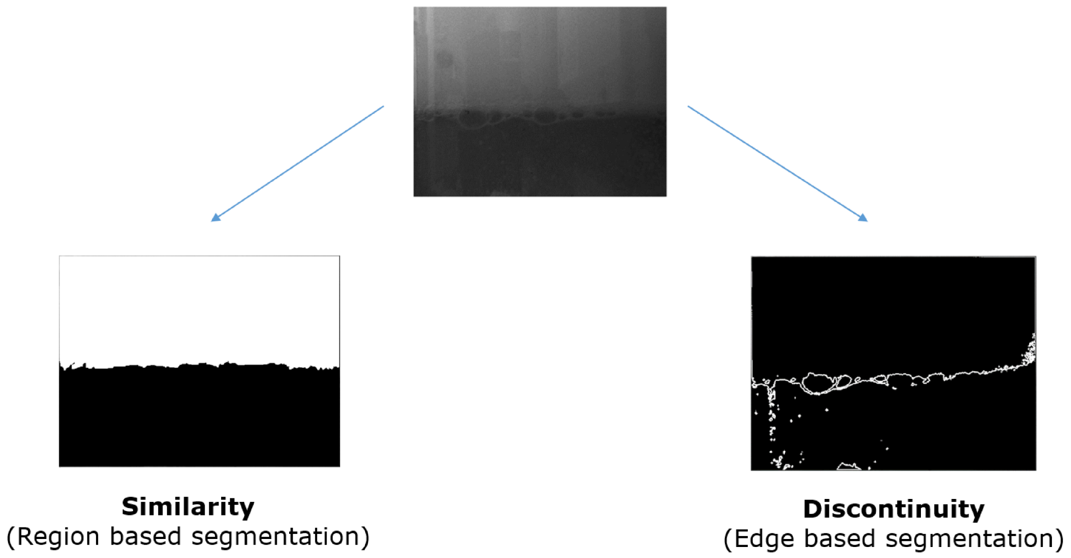


Figure 2.6: *Categories of gray level image segmentation process on a single image*

2.6.1 Region based segmentation

Region-based segmentation refers to the techniques for partitioning images directly into regions based on similar intensity values and properties of this values.

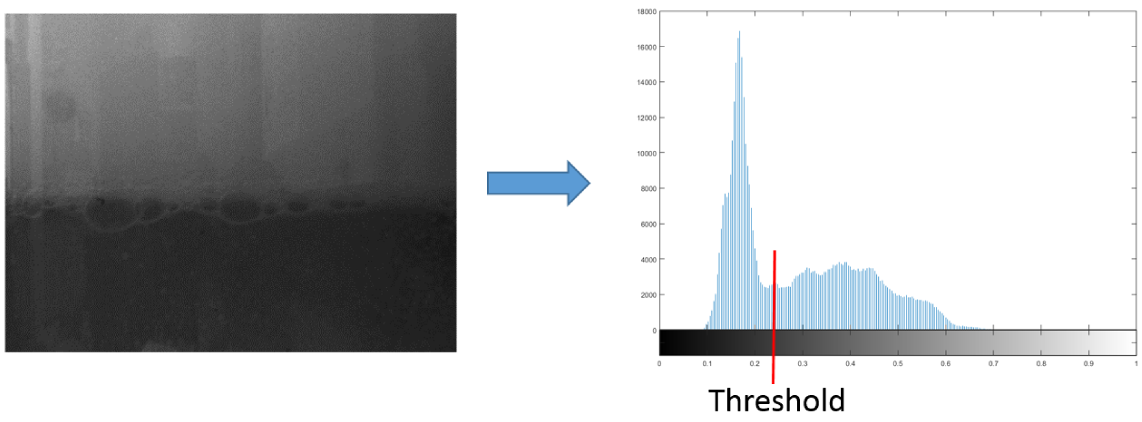


Figure 2.7: *Intensity histogram of froth-middlings interface level.*

In Figure 2.7 we have an image with its corresponding intensity histogram, where the pixel intensity values can be grouped into two dominant modes. One obvious way to separate the objects is to select a threshold T, that separates these modes. Then, any point (x, y) in the image at which $f(x, y) > T$ is classified into one region(white); otherwise is classified into the other region(black). Essentially, the segmented image output, $g(x, y)$, is given by Equation 2.1.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \leq T \end{cases} \tag{2.1}$$

When T is a constant value applicable over the entire image, the process is called *Global thresholding*. When the value of T changes over the image, it is referred as *Variable thresholding*. In addition, the term *local* or *regional thresholding* is used to denote a variable thresholding in which the value of T at any point (x, y) in an image depends on properties of its neighborhood points. Therefore, if the T depends on the spatial coordinates (x, y) themselves, then the *variable thresholding* is often referred to as *dynamic* or *adaptive thresholding* [12]. We explain and apply this method with more detail in Chapter 4.
Figure 2.8 illustrates the segmentation output of the interface level image based on an intensity threshold selection that separates the two dominant modes(oil and water). The success of image segmentation based on intensity thresholding is related

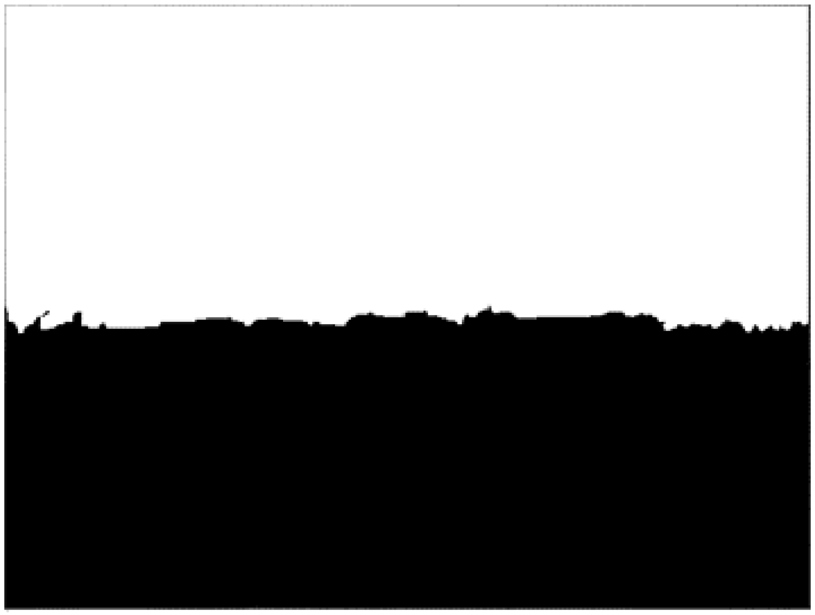


Figure 2.8: *Interface level segmentation based on histogram thresholding*

to the width and depth of the valleys separating the histogram modes. Thus, the main factors affecting the shape of the image modes are: 1) the separation between the peaks (the further the peaks are from each other, the better the chances to separate the modes); 2)the noise content in the image(the modes spread as noise increases); 3) the relative size of objects and background; 4) the uniformity of the illumination source; and 5) the uniformity of the reflectance properties of the image [12].

2.6.2 Boundary based segmentation (Edge Detection)

Changes or discontinuities in an image intensity amplitude are important characteristics of an image that carries information about objects borders. The detection methods based on image discontinuities are principal approaches to image segmentation and identification of objects in a frame.

Edge pixels are pixels at which the intensity of an image function changes abruptly, and edges (or edge segments) are sets of connected edge pixels. The local discontinuities in an image intensity map can fall into three categories: points, lines, and edges. The most common way to look for a particular image pattern(point, line, edge) is to convolve the image with an image mask or kernel. The mask could have a variable size of $M - by - N$ which depends on the type of the detected objects in the image. Figure 2.9 represents a 3x3 mask size.

The convolution between the mask and the image is performed by an inner vector

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figure 2.9: *Image Mask or Kernel*

product of the mask coefficients with the image intensity levels covered by the mask, as Equation 2.2 illustrates. The response of the mask (output) is given at the center point of the region.

$$R = w_1 * z_1 + w_2 * z_2 + \cdots + w_N * z_N = \sum_{i=1}^{N} w_i * z_i \tag{2.2}$$

Figures 2.10 and 2.11 are typical examples of point and lines detection masks respectively.

-1	**-1**	**-1**
-1	**8**	-1
-1	**-1**	**-1**

Figure 2.10: *Point detection Mask*

-1	-1	-1
2	2	2
-1	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

Figure 2.11: *Line detection Masks. a) Horizontal line detection mask ; b)Vertical line detection mask*

Edge detection is the most frequently used approach for segmenting images based on abrupt (local) changes of intensity. An edge is a boundary between two regions with distinct gray-level properties. Edges characterize the physical borders of objects, so their accurate detection plays a key role in image analysis and pattern recognition problems. Figure 2.12 illustrate two typical edge detection masks.
Edge models are classified according to their intensity profiles. Figure 2.13 displays

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 2.12: *Edge detection Masks. a) Horizontal edge detection mask ; b)Vertical edge detection mask*

the three different types of ideal edges (step edge, ramp edge, and roof edge) that we can find in an image map with their corresponding intensity profiles. However, in practice, digital images have edges that are blurred and noisy, with the degree of blurring determined by the focusing mechanism(i.e., Camera), and the noise level determined mainly by the electronic components of the imaging system.

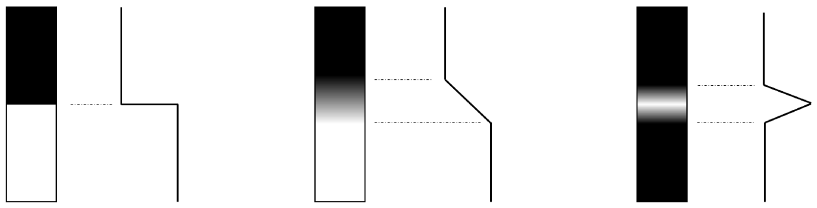


Figure 2.13: *Ideal representation of edge models and their corresponding intensity profiles. a) Step edge; b)Ramp edge; c) Roof edge*

Figure 2.7 depicts the segmentation output of the interface level image using the vertical edge detection mask of Figure 2.12a.

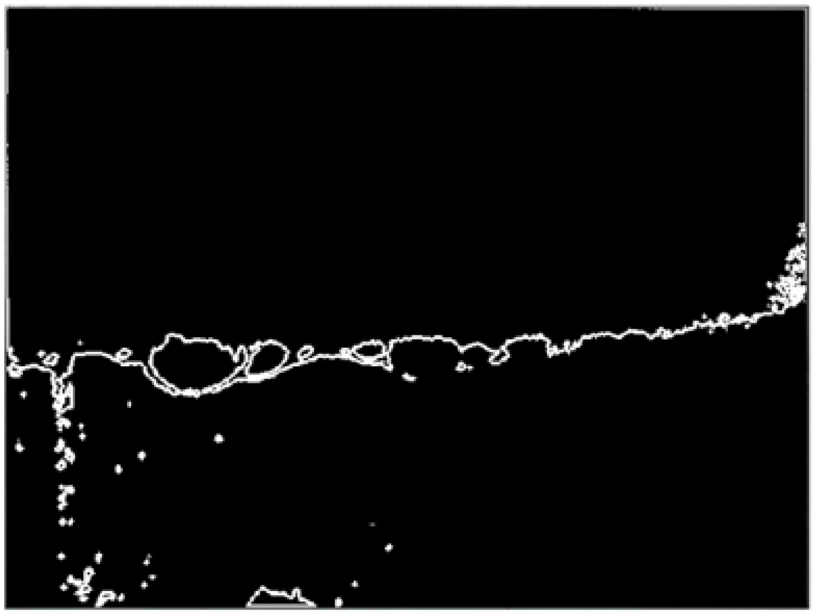


Figure 2.14: *Interface level segmentation based on edge detection.*

2.7 Motion detection

Motion is a powerful feature used by humans to extract objects of interest from a background of irrelevant detail. Motion detection is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. In the following section, we present two motion detection techniques used in image processing: Frame differences and Background Subtraction.

2.7.1 Frame differences

One approach for detecting changes between two image frames $f(x, y, t_i)$ and $f(x, y, t_j)$ taken at times t_i and t_j, respectively, is to compare the images pixel by pixel. One procedure to do this is to obtain a difference image. Suppose that we have a reference image that contains stationary components. Then, comparing this image against a subsequent image of the same scene, but including an object that has moved, results in a difference of the two images that cancels the stationary elements and keeps the non-stationary components [12].
A difference image between two images taken at times t_i and t_j can be defined by equation 2.4,

$$d_{ij}(x, y) = \begin{cases} 1 & \text{if } |f(x, y, t_i) - f(x, y, t_j)| > T \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

where T is a specified threshold. $d_{ij}(x, y)$ has a value of 1 at spatial coordinates (x, y) only if the intensity difference between two images is considerably different, as determined by the specified threshold T. Therefore, in dynamic image processing, all the pixel in $d_{ij}(x, y)$ with value 1 are considered the result of object motion.

Accumulative frame differences

Consider a sequence of image frames $f(x, y, t_1)$, $f(x, y, t_2)$,$\cdots$, $f(x, y, t_n)$, and let $f(x, y, t_1)$ be the reference image. An accumulative difference image (ADI) is obtained by comparing this reference image with every subsequent image in the sequence. A counter for each pixel location is incremented every time a difference occurs at that pixel location between the reference and an image in the sequence. Thus, when the k^th frame is compared with the reference, the entry in a given pixel of the accumulative image gives the number of times the intensity at that position was different from the corresponding pixel value in the reference image [12]. Equation 2.4 illustrates the accumulative difference calculation.

$$A_k(x,y) = \begin{cases} A_{k-1}(x,y) + 1 & \text{if } |f(x,y,t_1) - f(x,y,t_k)| > T \\ A_{k-1}(x,y) & \text{otherwise} \end{cases} \tag{2.4}$$

Figure 2.15 displays the accumulative difference image, $A_k(x,y)$, performed with a set of 5 consecutive frames. The brightness area corresponds to the main moving feature (interface level). However, we can also see that there are several white patches on various locations of the image, which are generated by the noise or moving elements/particles in each region (oil and water).

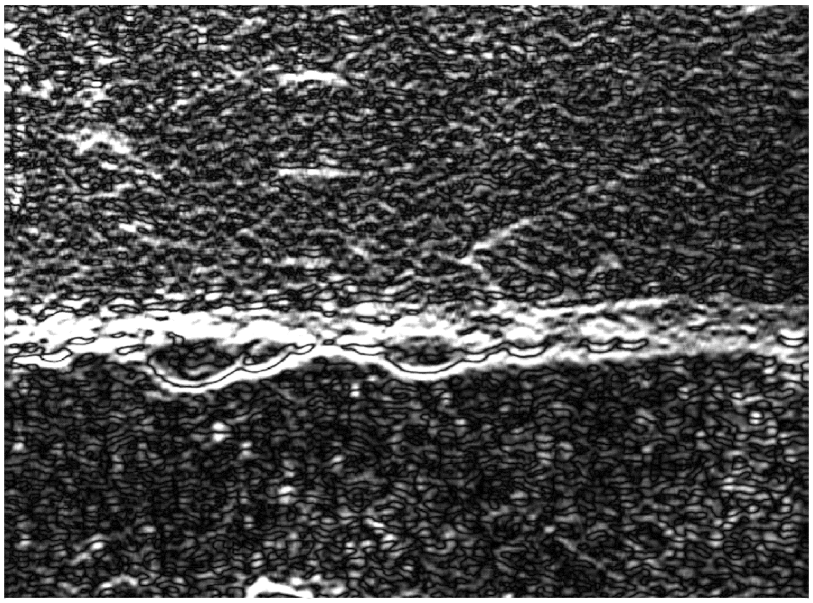


Figure 2.15: *Interface level segmentation based on motion detection*

2.7.2 Background substraction

Background subtraction is the process of separating out the foreground objects from the background in a sequence of video frames by considering a fixed reference "background image" $f(x,y,t_ref)$. Thus, the frame difference is performed between the current image,$f(x,y,t_i)$, and the reference background image, $f(x,y,t_ref)$. The key to this approach is to have an accurate "background image" or "background model", which has only information about the stationary components on the scene. In practice, obtaining a reference background image with only stationary elements is not always possible. Therefore, the background model has to be updated to the changes in the environment. In the case of indoor scenes, reflections, shadows, lighting changes or animated images on screens lead to background changes. Similarly, due to wind, rain or illumination changes brought by weather, static backgrounds methods have difficulties with outdoor scenes.

2.8 Conclusion

This chapter introduced digital image processing and analysis methods. Digital image processing is a transformation that takes an image into another image, while digital image analysis is the transformation of an image into numerical data. The digital image algorithms can be organized and classified into four types depending on its application: Image Processing, Image Analysis, Computer Vision and Machine Vision. Of particular interest are the computer vision algorithms that aim at giving computers a visual understanding of the world. A computer vision algorithm depends on its application, but there are typical functions that can be found in most architectures: Image Acquisition, Image Pre-processing, Image Processing, Image Analysis, and Decision making.
Feature extraction is the process of extracting some quantitative information of interest from an image. Usually, we are interested in particular features of the image that contains relevant information. Thus, we can restrict the image processing of the vision algorithm to those specific features in the images. In the case of the froth-middlings level detection problem, the feature of interest lays on the vertical axis of the image. By looking at a set of pixels data from the vertical profile (column data set)of the image we can infer the location of the level interface (the feature of interest).
Image segmentation is one of the most important and critical tasks in image

processing. It consists of subdividing the images into the parts of interest. Gray level segmentation is usually performed based on two basic image properties on the pixel level: discontinuity or continuity. Therefore, we described the region based segmentation algorithms that subdivide the image into regions that have similar pixel properties (continuity), and the edge based methods that subdivide the image into regions where there are pixels discontinuities (edge detection).

In addition, image segmentation can be performed based on the dynamical information of a set of consecutive frames. Motion detection finds the regions of an image where the position of an object has changed with respects to its surrounding among a set of different consecutive frames in time. Several approaches that perform image segmentation based on motion detection were described and analyzed with their advantages and disadvantages for each particular application: Frame differences, accumulative frame differences, and background subtraction.

Chapter 3

Interface Level Estimation Based on Image Restoration and Edge Detection

The objective of this chapter is to apply existing image processing techniques to interface level estimation in laboratory scale. The prior work conducted by Liu [7] has focused on Markov Random Fields based image segmentation to find the interface level by partitioning the image into oil and water regions. However, the segmentation-based methods may not provide satisfactory results since they are highly affected by the lighting conditions, gradual variations of pixel intensities along the images and presence of shadows and glares. Therefore, this chapter address the problem of detecting the interface level by looking at the intensity contrast change between the oil and water regions by applying edge detection operations on a single digital image, which is more robust to lighting conditions and extends the work of Liu [7] to other abnormal environmental scenarios. In addition, images are always corrupted by noise, and thus it is required before a good image preprocessing or restoration step to improve the efficiency of the level detection. The objective of image restoration is to remove the noise from the images without losing the features of interest (froth-middlings interface level). In this sense, we are motivated to use model-based image restoration methods which are more robust than regular restoration methods. Prior image models based on Bayesian Networks and Markov Random Fields are proposed for the restoration process in this chapter. The proposed methods are implemented on the forth-middlings interface images of the pilot-scale PSV setup.

3.1 Introduction

Edge detection is by far the most common approach for detecting significant discontinuities in gray level images [12]. An edge is a set of connected pixels that lie on the boundary between two regions. This means that if the edges in an image can be identified accurately, all of the objects can be located and some basic properties such as area, perimeter, and shape can be measured. The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries [16, 17]. In the interface detection problem, edge detection serves as a tool to identify the boundaries between the froth and middlings regions. In other words, by looking at the intensity change between froth and middlings, we can find a potential location of the interface level.

Efficient and accurate edge detection will improve the performance of subsequent image processing techniques, including image segmentation, object-based image coding, and image retrieval [18]. In practice, optics, sampling, and other image acquisition imperfections can generate edges that are blurred. The degree of blurring is determined by factors such as the quality of the image acquisition system, the sampling rate, and the illuminations conditions. As a result, edges are more closely modeled as having a "ramp-like" profile when the technique is applied to the raw image data [12]. Therefore, removing noise from the raw image will lead to more efficient and accurate edge detection of the level interface. In this sense, image restoration attempts to remove the noise and reconstruct the original image by using prior knowledge about degradation phenomenon.

In this chapter, we introduce the challenges of finding an interface by looking at the intensity contrast changes in a single image (vertical edge detector). Later, we describe the theory of image restoration problem that deals with the reconstruction of the original image data. Finally, we apply the state estimation theory to perform the image restoration process as a model-based optimization problem.

3.2 Problem Formulation

In general, if we look at the froth-middlings interface image, we can distinguish three regions on the vertical axis: the froth region (oil), the interface region, and the middlings region (water). Therefore, there are two boundaries along the vertical profile of the image. One boundary lies between the froth-interface region and the other lies between interface-middlings region. As we mentioned in Sectionv2.5, these interface-boundary features are located over the vertical direction of the image. Accordingly, an edge operator can be employed to detect the boundaries or transition regions on the images. Therefore, applying a vertical edge detector mask as in Figure 2.12b, we can detect the vertical properties changes in the froth-middlings interface image.

Figure 3.1 presents the froth-middlings interface image. On the left, there is a raw image acquired by the camera, while on the right there is an example of an ideal image. Figure 3.1 displays the region and boundary concepts of the forth-middlings interface. On the left, there is a raw image acquired by the camera, while on the right there is an ideal image. The real image has a vertical profile, 3.1.c, which is corrupted by noise and other factors, while the ideal image has a straight and clean vertical profile, 3.1.d. Thus, when running a vertical edge detector on each profile, we obtain the vertical change profile as Figures 3.1.e and 3.1.f. We notice that the real image vertical change profile is spiky and noisy, while the ideal image vertical change profile displays clearly the interface region.

Efficient and accurate detection of the interface can be achieved by removing the undesired noise in the image. As described previously, the objective of image processing is to enhance image properties in a manner that facilitates the following feature detection step. Thus, in the next section, we introduce the image restoration process on digital images. This method attempts to remove the corrupted/noisy image to get a clean and original image. An image restoration process will guarantee better outcomes on the interface level detection.

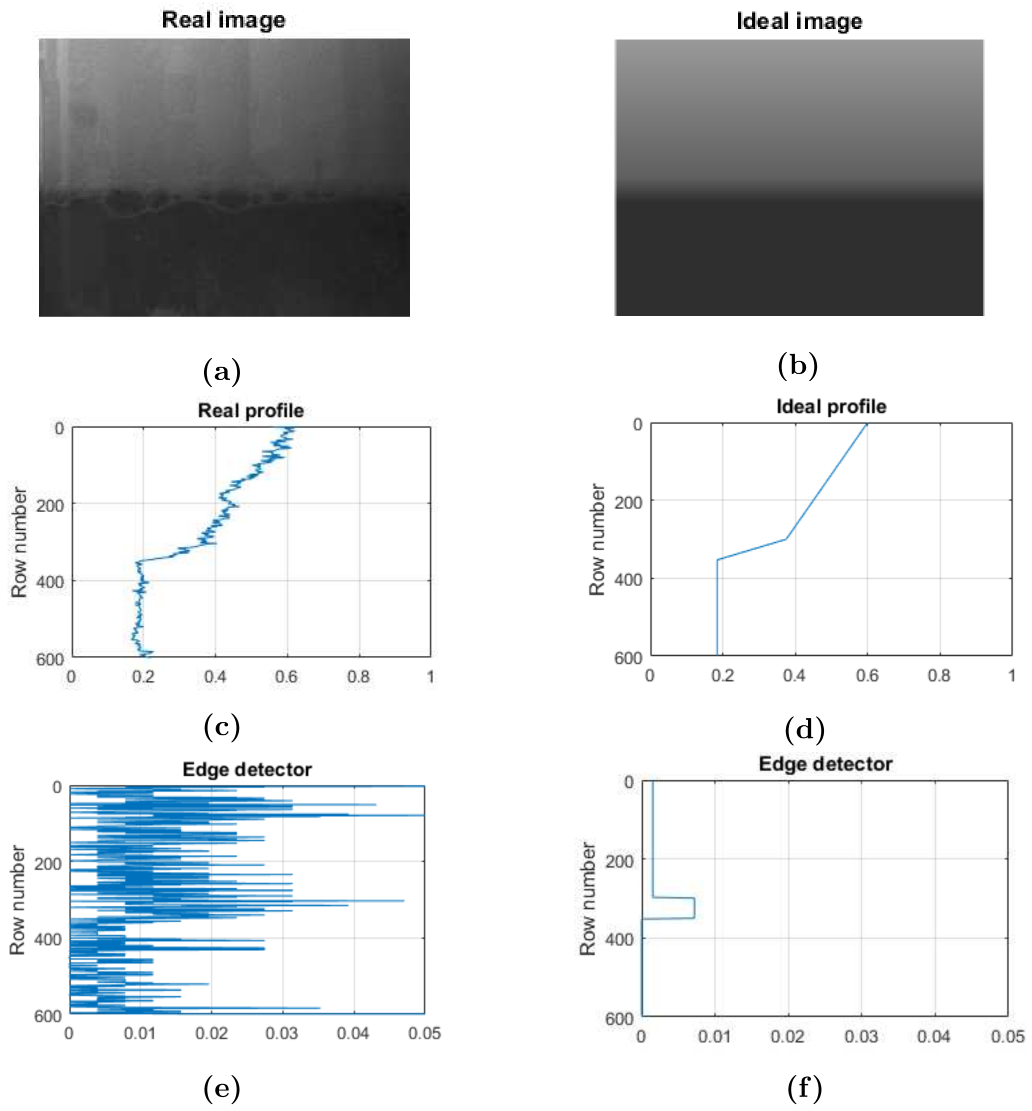


Figure 3.1: *Edge detection comparison between a real image -corrupted by noise and distortions- on the left-hand side, and an ideal image on the right-hand side.*

3.3 Image Restoration

The ultimate goal of image restoration techniques is to improve an image in some predefined senses. Restoration attempts to reconstruct or recover an image, which has been degraded by using prior knowledge of the degradation phenomenon [12, 19]. Thus, restoration techniques are oriented toward modeling the degradation and applying the inverse process to recover the original image. This approach usually involves formulation a criterion of goodness that will yield an optimal estimate of the desired result.
The degradation process, shown in Figure 3.2, is modeled as a degradation function that, together with an additive noise term, operates on an input image $f(x, y)$ to produce a degraded (observed) image $g(x, y)$.

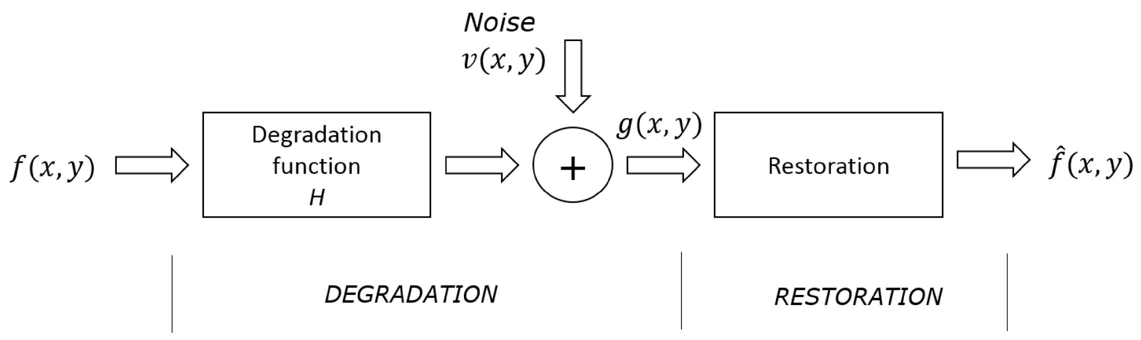


Figure 3.2: *Degradation and restoration process schematics.*

Given the observed image $g(x, y)$, some knowledge about the degradation function H, and some knowledge about the additive noise term $v(x, y)$, the objective of restoration is to obtain an estimate $\hat{f}(x, y)$ of the original image. We would like to make the estimate as close as possible to the original input image. And, the more we know about H and $v(x, y)$, the closer $\hat{f}(x, y)$ will be to $f(x, y)$.
The purpose of image restoration is to recover the true pixel values $f(x, y)$, from the observed (noisy) image pixel values $g(x, y)$ [12]. When the image $f(x, y)$ is considered as a noise-added surface, the problem now is to restore the underlying

surface $f(x, y)$ [20]. For the sake of notations simplicity, we represent the observation function $g(x, y)$ as observation values $\mathbf{y}_n$, the original image $f(x, y)$ as hidden true states $\mathbf{x}_n$, and the additive noise term $v(x, y)$ as the observation model noise $\mathbf{v}_n$

3.3.1 Observation and model-based image optimization

The image restoration performance can be improved by using a model-based optimization method. The main reason for the use of optimization is the existence of uncertainties in every vision process. Noise and other degradation factors, such as those caused by disturbances and quantization in sensing and signal processing are sources of uncertainties. Thus, the model-based optimization methods are more robust for an image restoration process than non-model-based optimization.
Figure 3.3 describes a typical framework of image optimization at a pixel level [21]. For the sake of simplicity, we represent the observation function $g(x, y)$ as observation values $\mathbf{y}_n$, and the original image $f(x, y)$ as hidden true states $\mathbf{x}_n$. Essentially, we have the sampled image in which each pixel is considered as an observation value $\mathbf{y}_n$ with noise and uncertainties. On the other hand, we have the hidden states $\mathbf{x}_n$, which represent the original image values that we would like to estimate. Then, based on these frameworks we can formulate two different functions to relate these graphical models nodes. One of the functions, φ, (called the Observation Model), relates the observations and the true states, and the other function, ψ, (called the Graphical or State Relation Model), relates the true states based on predefined criteria chosen by the user.

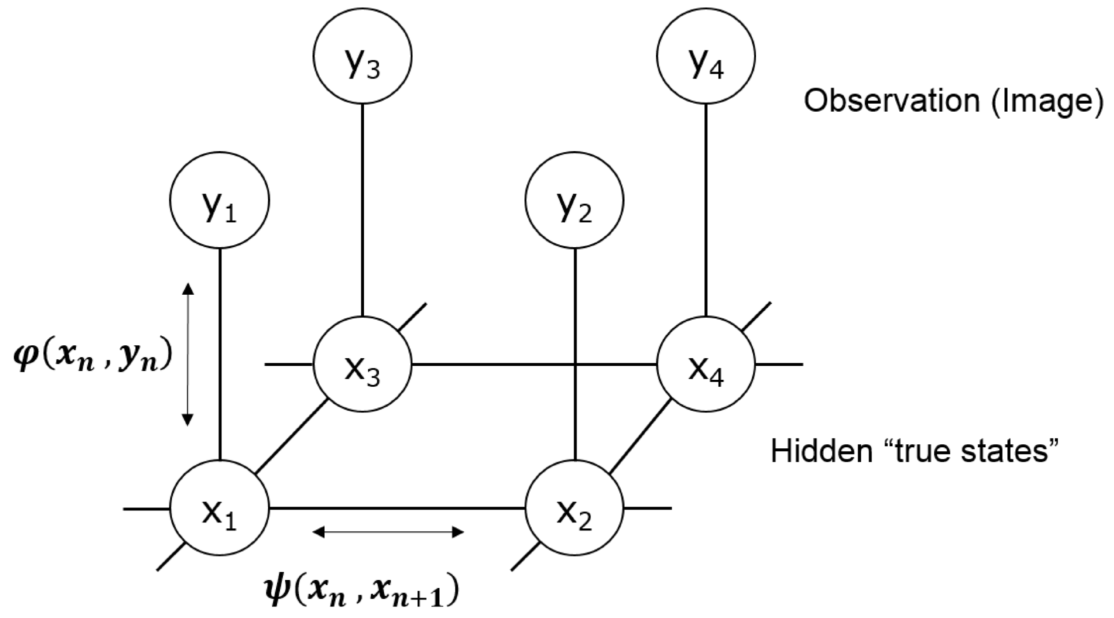


Figure 3.3: *Observation and model-based image optimization*

By defining the Observation and State/Graphical models, we can write an objective function as in Equation 3.1. The solution is explicitly defined as an optimum of an objective function by which the goodness of the solution is measured. Finally, employing an optimization algorithm, as in Equation 3.2 (minimization), we can obtain the optimal estimation of the true states and restore the original image.

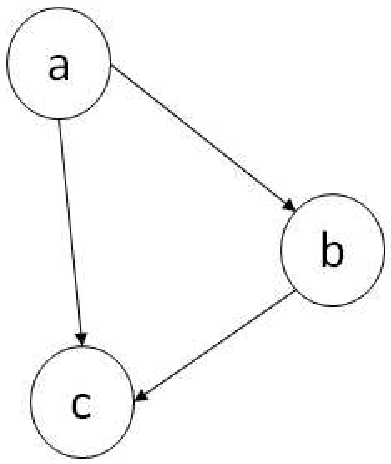
$$E(\mathbf{x}_n, \mathbf{y}_n) = \varphi(\mathbf{x}_n, \mathbf{y}_n) + \psi(\mathbf{x}_n, \mathbf{x}_{n+1}) \tag{3.1}$$

$$\min_{\mathbf{x}_n} E(x_n, y_n) \tag{3.2}$$

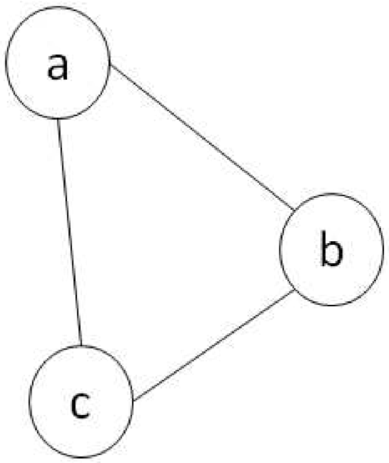
Equation 3.1 is generally called energy function in optimization-based vision problems. The role of the energy function is twofold: (1) as the quantitative measure of the global quality of the solution and (2) as a guide to the search for a minimal solution. Besides, the two most important aspects of an energy function are its structure and the parameters involved.

3.3.2 Graphical models

A graph is composed by nodes (also called vertices) connected by links (also known as edges or arcs). In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables [22]. One major class of graphical models are the Bayesian Networks(BN), also known as directed graphical models 3.4a, in which the links of graphs have a particular direction. The other major class of graphical models is Markov Random Fields (MRF), also known as undirected 3.4b graphical models, in which the links have no directional significance. Directed graphs are useful for expressing causal relationships between random variables, while undirected graphs are better for expressing soft constraints between random variables. Therefore, in the following sections, we introduce the optimal image restoration process through both image modeling approaches. We employ BN approaches in sections 3.4 and 3.5, and MRF in Section 3.6.



(a) *Directed graphical model (Bayesian Networks)*



(b) *Undirected graphical model (Markov Random Fields)*

Figure 3.4: *Graphical models representing the joint probability distribution over three variables a, b, and c.*

3.4 Image Restoration using Least Square Estimation (Wiener filter)

The first method developed to reduce additive random noise in images is based on Wiener Filtering (WF) [13] , which designs a linear, time-invariant filter to extract a signal from noise in the frequency domain [23]. The WF is an optimal minimum mean square error estimator. In other words, it minimizes the overall mean square error in the process of inverse filtering and noise smoothing.
The method consists of considering images and noise as random variables, and the objective is to find an estimate $\hat{\mathbf{x}}_n$ of the original image $\mathbf{x}_n$ such that the mean square error between them is minimized. This error measure is given by Equation 3.3.

$$\mathbf{e}^2 = E\{(\mathbf{x}_n - \hat{\mathbf{x}}_n)^2\} \tag{3.3}$$

It is assumed that the image, $\mathbf{x}_n$, and noise, $\mathbf{v}_n$ are uncorrelated and stationary linear stochastic process.
In addition, the WF can work as an adaptive method based on a statistical model estimated from a local neighborhood system around each pixel. The WF tailors itself to the local image variance. For example, the filter performs less smoothing when the variance is large, while performs more smoothing when he variance is small. This approach often produces better results than linear filtering. The adaptive filter is more selective than linear filtering, preserving edges, and other high-frequency parts(features) of an image.

Model Parameters Estimation

Adaptive WF estimates the local mean and variance around each pixel using the following Equations 3.4 and 3.5.

$$\mu = \frac{1}{NM}\sum \mathbf{y}_n \tag{3.4}$$

$$\sigma^2 = \frac{1}{NM}\sum \mathbf{y}_n^2 - \mu^2 \tag{3.5}$$

where $N \times M$ is the N by M local neighborhood system of each pixel in the image. Finally, adaptive WF estimate is based on the Equation 3.6.

$$\hat{\mathbf{x}}_n = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2}(\mathbf{y}_n - \mu) \tag{3.6}$$

where ν^2 is the average noise variance of the image.

The mean square error is not, however, the criterion used by a human observer in judging how close a restored image is to the original. Moreover, the WF is not very amenable to state estimation because of difficulty in extension to Multiple Input-Mulitple Output (MIMO) problems with state variable descriptions, and difficulty in the application to signals with time-varying statistical properties [23].

3.4.1 Simulations and results

The interface detection process is performed in two steps, image restoration, and vertical edge detection. In image restoration process, we use Equation 3.6 with parameters in Table3.1. After that we use the vertical edge operator described in 2.12 to find the level interface location.
Figure 3.5 illustrates the results of level interface estimation based on the WF restoration and edge detection. 3.5a corresponds to the raw interface image with its noisy vertical profile, 3.5b depicts the restored image and cleaned profile after smoothing with WF; and 3.5c shows the vertical change profile after using the vertical edge detector. The maximum of this vertical change profile corresponds to the interface level location.

Parameters	**Values**	**Description**
$\mathbf{x}_n$	[0,1]	Continuous interval set
$N \times M$	10x10	Local neighborhood system size
μ	Adaptive	Estimated from local neighborhood system
σ^2	Adaptive	Estimated from local neighborhood system

Table 3.1: *Wiener Filter model parameters*

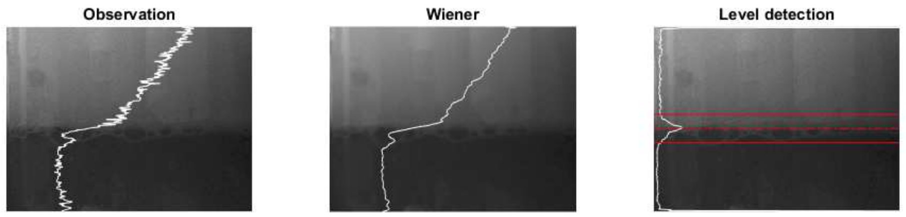


Figure 3.5: *WF restoration and level detection. (a)Raw image and vertical profile, (b) Wiener restored image and vertical profile, (c) vertical change and level detection on Wiener restored image*

3.5 Image Restoration using Kalman Filter

Many types of image degradation can be approximated by a linear, position-invariant process. The advantage of this approach is that the linear system theory becomes available for the solution of image restoration. The Kalman Filter(KF) and Kalman Smoother(KS) are well-known state estimation filters for linear systems with additive Gaussian noise. Therefore, we can formulate the image restoration process as a state estimation problem for a linear state-space model specified as Equations 3.7 and 3.8.

$$\mathbf{x}_n = \mathbf{F}_n \mathbf{x}_{n-1} + \mathbf{w}_n \tag{3.7}$$

$$\mathbf{y}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{v}_n \tag{3.8}$$

where $\mathbf{x}_n \in \mathcal{R}^K$ is the state vector, and $\mathbf{y}_n \in \mathcal{R}^L$ is the measurement vector. $\mathbf{F}_n \in \mathcal{R}^{K \times K}$ is the state transition matrix, $\mathbf{H}_n \in \mathcal{R}^{L \times K}$ is the measurement matrix, and $\mathbf{w}_n \in \mathcal{R}^P$ and $\mathbf{v}_n \in \mathcal{R}^L$ express the process and measurement noises that are zero-mean white Gaussian and mutually uncorrelated, i.e., $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ and $\mathbf{v}_n \sim \mathcal{N}(0, \mathbf{R}_k)$.

Kalman Filter

The optimal estimation tries to find the state estimate that minimizes the cost function (Minimum Mean Squared Error) as Equation 3.9:

$$\mathbf{J}_e = \int_0^{t_f} E[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T]dt \tag{3.9}$$

The optimal solution for a linear state-space model with Gaussian noise is provided by the KF specified as Equations 3.10-3.14.

$$\mathbf{P}_n^- = \mathbf{F}_n \mathbf{P}_{n-1}^+ \mathbf{F}_n^T + \mathbf{Q}_n \tag{3.10}$$

$$\mathbf{K}_n = \mathbf{P}_n^- \mathbf{H}_n^T (\mathbf{H}_n \mathbf{P}_n^- \mathbf{H}_n^T + \mathbf{R}_n)^{-1} \tag{3.11}$$

$$\hat{\mathbf{x}}_n^- = \mathbf{F}_n \hat{\mathbf{x}}_{n-1}^+ \tag{3.12}$$

$$\hat{\mathbf{x}}_n^+ = \hat{\mathbf{x}}_n^- + \mathbf{K}_n(\mathbf{y}_n - \mathbf{H}_n \hat{\mathbf{x}}_n^-) \tag{3.13}$$

$$\mathbf{P}_n^+ = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n)\mathbf{P}_n^- \tag{3.14}$$

3.5.1 Fixed-interval smoothing: Kalman Smoother

In fixed-interval smoothing, we seek an estimate of the state at some interior points of the interval. Several forms of the fixed-interval smoother have been proposed. One of the most common methods is the smoother by Rauch, Tung and Striebel, called the RTS smoother [23].

The RTS smoother is an efficient two-pass algorithm for solving the fixed-interval smoothing problem. The forward pass is the same as the standard KF algorithm. State estimates and covariances obtained during the forward pass are saved for use in the backward pass algorithm. The RTS smoother algorithm can be summarized as follows:

1. The system model is given as follow:

   $\mathbf{x}_n = \mathbf{F}_{n-1}\mathbf{x}_{n-1} + \mathbf{w}_{n-1}$

   $\mathbf{y}_n = \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n$

   $\mathbf{w}_n \sim (0, \mathbf{Q}_n)$

   $\mathbf{v}_n \sim (0, \mathbf{R}_n)$

2. Initialize the forward filter:

   $\hat{\mathbf{x}}_0 = E(\mathbf{x}_0)$

   $\mathbf{P}_0 = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$

3. For $n = 1, \cdots, N$ (where N is the final pixel), execute the standard forward KF:

   $\mathbf{P}_n^- = \mathbf{F}_{n-1}\mathbf{P}_{n-1}^+\mathbf{F}_{n-1}^T + \mathbf{Q}_{n-1}$

   $\mathbf{K}_{f,n} = \mathbf{P}_n^-\mathbf{H}_n^T(\mathbf{H}_n\mathbf{P}_n^-\mathbf{H}_n^T + \mathbf{R}_n)^{-1}$

   $\hat{\mathbf{x}}_{f,n}^- = \mathbf{F}_{n-1}\hat{\mathbf{x}}_{f,n-1}^+$

   $\hat{\mathbf{x}}_{f,n}^+ = \hat{\mathbf{x}}_{f,n}^- + \mathbf{K}_{f,n}(\mathbf{y}_n - \mathbf{H}_n\hat{\mathbf{x}}_{f,n}^-)$

   $\mathbf{P}_{f,n}^+ = (\mathbf{I} - \mathbf{K}_{f,n}\mathbf{H}_n)\mathbf{P}_n^-$

4. Initialize the RTS smoother as follows:

   $\hat{\mathbf{x}}_N = \hat{\mathbf{x}}_{f,N}$

   $\mathbf{P}_N = \mathbf{P}_{f,N}^+$

5. For $n = N-1, \cdots, 1, 0$ execute the following RTS smoother equations:

   $\mathbf{K}_n = \mathbf{P}_{f,n}^+\mathbf{F}_{n-1}^T(\mathbf{P}_{f,n+1}^-)^{-1}$

   $\mathbf{P}_n = \mathbf{P}_{f,n}^+ - \mathbf{K}_n(\mathbf{P}_{f,n+1}^- - \mathbf{P}_{n+1})\mathbf{K}_n^T$

   $\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_{f,n}^+ + \mathbf{K}_n(\hat{\mathbf{x}}_{n+1} - \hat{\mathbf{x}}_{f,n+1}^-)$

3.5.2 Simulations and results

As in the previous section, the simulation was performed in two steps: an image restoration using the KF and the KS, and interface level detection using a vertical edge mask. Recalling the interface detection problem, the interest is focused on the image vertical profile. Figure 3.6 displays a column data set from the image corresponding to the image vertical profile. The Figure compares the KF and the RTS, in which we can observe that RTS provides a smoother profile than the KF. Thus, we use the RTS algorithm to perform the image restoration process and level estimation.

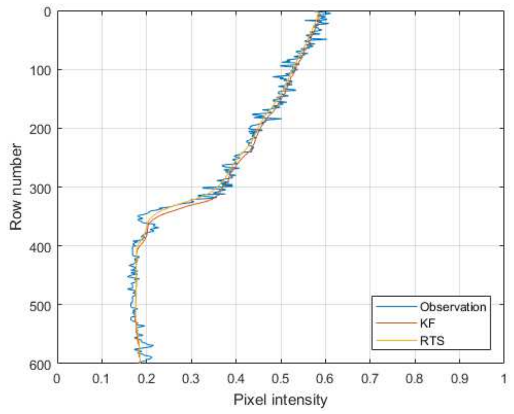


Figure 3.6: *The KF and the RTS restoration profiles comparison.*

The RTS algorithm is used on vertical (each column data-set) and horizontal (each row data-set) directions on the image. Also, it is used in both directions combined, by first running in a horizontal direction for every row data-set, and then in a vertical direction for every column data-set. Table 3.2 provides the parameters used in the RTS smoother simulation. We tuned $\mathbf{Q}_n$ and $\mathbf{R}_n$ to obtain a smoothed profile, under the consideration that system model uncertainties ($\mathbf{Q}_n$) are smaller than measurement model uncertainties ($\mathbf{R}_n$). Figure 3.7 has RTS results in each image direction, and the final combination in both directions.

Parameters	Values	Description
$\mathbf{F}_n$	1	
$\mathbf{H}_n$	1	
$\mathbf{R}_n$	1	Sensor noise
$\mathbf{Q}_n$	0.01	System noise
$\mathbf{x}(\mathbf{0})$	0.5076	1st row/column mean value
$\mathbf{P}(\mathbf{0})$	7.1577	1st row/column variance value

Table 3.2: *KF and RTS model and noise parameters*

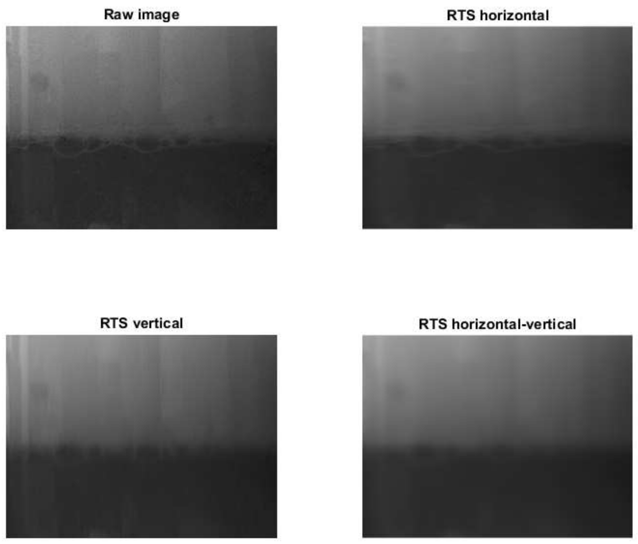


Figure 3.7: *Image restoration through RTS smoother. (a) Raw image, (b) RTS applied on image horizontal direction, (c) RTS applied on image vertical direction, (d) RTS applied on horizontal and vertical combined directions*

After obtaining the restored image by RTS smoother, we run the vertical edge detector to detect the level interface. Figure 3.8 describes the entire interface detection process. 3.8a shows the acquired interface image with its noisy vertical profile, Figure 3.8b depicts the restored image through RTS and its cleaned vertical profile, and Figure 3.8c illustrates the vertical change profile and the interface level location that correspond to the peak.

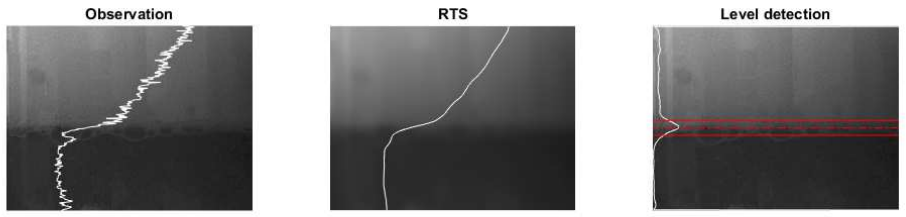


Figure 3.8: *Image restoration and level detection through RTS smoother. (a)Raw image and vertical profile, (b) RTS restored image and vertical profile, (c) vertical change profile and level detection on RTS restored image*

The WF approach is based on the frequency domain analyses, whereas the KF or the RTS that we derived later are based on the time domain analyses. Nevertheless, both filters are optimal under their own assumptions [23].

3.6 Image Restoration using Markov Random Models

When the observed image is considered as a noise-added surface, the problem is to restore the underlying surface. The purpose of image restoration is to recover the true pixel values $\mathbf{x}_n$, from the observed (noisy) image pixel values $\mathbf{y}_n$. In Section 3.3.2, we introduce the preliminaries of MRF graphical models. The next step is to develop the image optimization framework through MRF models. To do that, we first need to build the observation and prior models, which is given below.

3.6.1 MRF observation model for images

Each observed pixel value is assumed to be the sum of the true gray value and independent Gaussian noise:

$$\mathbf{y}_n = \mathbf{H}(\mathbf{x}_n) + \mathbf{v}_n \tag{3.15}$$

where $\mathbf{H}(\mathbf{x}_n)$ is a linear function and $\mathbf{v}_n \sim N(0, \sigma^2)$. The function $\mathbf{H}(\mathbf{x}_n)$ maps a label $\mathbf{x}_n$ to a real gray value, where $\mathbf{x}_n$ may be continuous or discrete. We can consider that there is a unique numerical value for a label $\mathbf{x}_n$ and write Equation 3.15 simply as $\mathbf{y}_n = \mathbf{x}_n + \mathbf{v}_n$, and the likelihood energy function then becomes:

$$U(y|x) = \sum_{n \in S} \frac{(\mathbf{x}_n - \mathbf{y}_n)^2}{2\sigma^2} \tag{3.16}$$

where S represents pixel sites.

3.6.2 MRF priors model for images

According to the Markov-Gibbs equivalence, the prior probability follows a Gibbs distribution

$$P(x) = \frac{1}{Z} e^{\frac{1}{T} U(x)} \tag{3.17}$$

where Z is a normalizing constant called partition function, T is a constant that is usually assumed as 1, and $U(x)$ is a prior energy function. The general expression for $U(x)$ is given as

$$U(x) = \sum_{n \in S} \sum_{n' \in N} g(\mathbf{x}_n - \mathbf{x}_{n'})^2 \ , \tag{3.18}$$

where N denotes the set of neighborhood pixels (cliques) defined in the MRF model. The underlying surface $\mathbf{x}_n$, from which the observation $\mathbf{y}_n$ is sampled, is a graph surface defined on a continuous domain. The first factor affecting the specification of the MRF prior distribution is whether $\mathbf{x}_n$ takes a continuous or discrete value [20]. Piece-wise constant surfaces $\mathbf{x}_n$ or homogeneos blob-like regions, can be properly characterized by the MML(multilevel logistic)[20] that uses cliques potentials(neighbor pixels) described by:

$$V_c(f) = \begin{cases} 0 & \text{if all sites have the same label} \\ \zeta_c & \text{otherwise} \end{cases} \tag{3.19}$$

If all sites have the same label, it means the entire smoothness of label f on sites c. Any violation of the entire smoothness incurs a penalty of the positive number $-\zeta_c$. The prior energy is the sum of all the clique (neighbors) potentials:

$$U(x) = \sum_{n \in S} \sum_{n' \in N} [1 - \delta(\mathbf{x}_n - \mathbf{x}_{n'})] \tag{3.20}$$

3.6.3 MAP-MRF framework

Bayesian framework helps to obtain statistical inferences, incorporating the prior information. The statistical image analysis problem based on MRF field can be solved using a MAP solution [7]. The objective of MAP solution is to maximize the posterior probability, which can be represented as

$$\max_x P(x|y) \tag{3.21}$$

According to the Baye's rule, the posterior probability can be computed by using the following equation:

$$P(x|y) = \frac{P(x,y)}{P(y)} = \frac{P(y|x)P(x)}{P(y)} \ , \tag{3.22}$$

where $P(x,y)$ is the joint probability distribution, $P(y|x)$ is the conditional probability density function (p.d.f) of the observation $\mathbf{y}$ (also called the likelihood function of $\mathbf{x}$ for $\mathbf{y}$), $P(x)$ is the prior probability of $\mathbf{x}$, and $P(y)$ is the density of observation $\mathbf{y}$. Therefore, the posterior probability is proportional to the product of likelihood function and the prior probability, which is provided as

$$P(x|y) \propto P(y|x)P(x) \tag{3.23}$$

In the image analysis problem based on MRF, $P(x|y)$ is indeed the posterior probability distribution of an MRF. According to the Markov-Gibbs equivalence, the prior probability follows a Gibbs distribution specified as

$$P(f) = \frac{1}{Z} e^{\frac{1}{T} U(x)} \ , \tag{3.24}$$

where Z is a normalizing constant called partition function, T is a constant that is usually assumed as 1, and $U(x)$ is the energy function. By substituting equation 3.24 into 3.23, we can obtain the expression for the posterior probability as,

$$P(x|y) \propto e^{U(x|y)} \propto e^{U(y|x)} e^{U(y)}, \tag{3.25}$$

from which we can determine that

$$U(x|y) \propto U(y|x) + U(y) \tag{3.26}$$

where $U(x|y)$ is defined as posterior energy, and $U(y|x)$ is called likelihood energy. Using the prior and likelihood models from Sections 3.6.1 and 3.6.2, we can write the general form of the posterior energy function as

$$U(x|y) = \sum_{n \in S} \frac{(\mathbf{x}_n - \mathbf{y}_n)^2}{2\sigma^2} + \sum_{n \in S} \sum_{n' \in N} g(\mathbf{x}_n - \mathbf{x}_{n'}) \tag{3.27}$$

Maximizing the posterior probability $P(x|y)$ is equivalent to minimizing the posterior energy function $U(x|y)$, which is specified as

$$\min_x U(x|y) \tag{3.28}$$

The only parameter that needs to be determined is the variance σ^2 of the noise distribution. Once σ^2 is determined, the minimum of posterior energy $U(x|y)$ based on MAP-MRF solution can be completely achieved.

3.7 MRF piece-wise constant restoration

In a piece-wise constant restoration, $\mathbf{x}_n$ consist of discrete values. It means that $\mathbf{x}_n$ can take values of 0 or 1. The task is to recover the true configuration $\hat{\mathbf{x}}_n$ from the observed image $\mathbf{y}_n$. In the MAP-MRF framework, the optimal $\hat{\mathbf{x}}_n$ is the one that minimizes the posterior energy function in Equation 3.28 [7].

Posterior Energy function

According to Li [20], the procedure of the MAP-MRF approach is summarized as the following steps:

1. Define a neighborhood system and set the cliques for it.

2. Define the prior cliques potential function in the Gibbs prior distribution.

3. Derive the likelihood energy function form the observation model.

4. Add the prior energy $U(x)$ and the likelihood energy $U(y|x)$ to yield the posterior energy.

$$U(x|y) = \sum_{n\in S} \frac{(\mathbf{x}_n - \mathbf{y}_n)^2}{2\sigma^2} + \sum_{n\in S}\sum_{n'\in N}[1 - \delta(\mathbf{x}_n - \mathbf{x}_{n'})] \tag{3.29}$$

Energy minimization

The minimization solution $\hat{\mathbf{x}}$ is the optimally restored image in the configuration space. The MAP solution is defined as $\hat{\mathbf{x}} = \min_x U(x|y)$. The simplest algorithm to find $\hat{\mathbf{x}}$ is the steepest local energy descent or the greedy method. The procedure is to start with an initial configuration $\mathbf{x}(0)$ and iterate until it reaches a minimum value. An example of such gradient descent algorithms is the iterative conditional modes(ICM)[20]. The steepest local energy descent algorithm finds a local energy minimum whose quality depends on the initial estimate $\mathbf{x}(0)$. Global minimization algorithms such as simulated annealing need to be used if global solutions are required.

3.7.1 Simulations and results

In a piece-wise constant restoration, $\mathbf{x}_n$ consist of discrete values/labels(i.e. 0,1,2,3, etc.). Thus, in this simulation we considered binary labels (0 or 1) for $\mathbf{x}_n$, which results into a binary segmentation problem. It means that we assign 1 to each pixel that corresponds to oil and 0 to each pixel that is water. An Iterative Conditional Mode (ICM) algorithm is used to minimize the Posterior Energy function. The model parameters, μ and σ^2, are estimated from the data. As we establish the problem as a binary discrete segmentation, we need to estimate a set of parameters for each

phase(μ, σ^2 for oil, and μ, σ^2 for water). One challenge of image processing is the image space-variant properties that we could find on different image regions. These space-variant properties can be caused by diverse factors such as noise, lightening conditions, shadows, reflections or obstructions. As a result, image statistical properties, μ, σ^2, could differ significantly from one region to the other on the same image. In [7], considers one set of parameter (μ, σ_2) for the entire image, which penalizes some regions during the image segmentation process and generates a poor segmentation. However, based on prior knowledge, the level is a property that lies on the image vertical direction. Therefore, we proposed to estimate the model parameter μ, σ^2 from each image column in order to improve the algorithm segmentation.
Once we obtain the segmented image, we run the vertical edge detector to find the level interface. Figure 3.9 has the original image, the binary segmented image through MRF, and the level detection results.

Parameters	**Description**
$\mathbf{x}_n$	Binary(0 for water and 1 for oil)
μ	Column adaptive, estimated from column data
σ^2	Column adaptive, estimated from column data
N Neighborhood system	1st order
Minimization algorithm	ICM (Iterative conditional mode)

Table 3.3: *MRF binary restoration parameters.*

3.8 Markov random field edge detection

Edges correspond to abrupt changes or discontinuities in certain image properties [20]. The image properties may be non-texture or texture. In this section, we are interested in non-texture edges due to changes in image intensity, such as jump edges and roof edges. Jump edges correspond to the discontinuities in the underlying surface $\mathbf{x}_n$.
Edge detection is an image restoration process that also involves discontinuities,

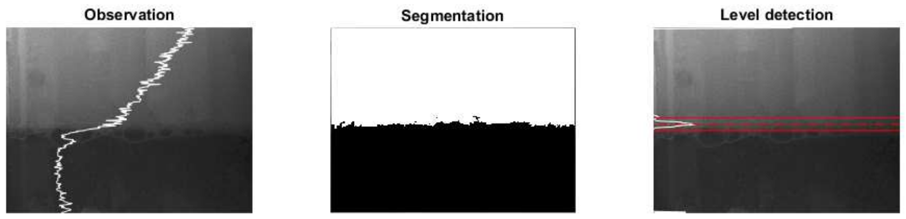


Figure 3.9: *Markov Random Field Restoration and level detection. (a)Raw image and vertical profile, (b) MRF binary segmented image, (c) vertical change profile and level detection on MRF binary segmented image*

therefore we are interested in removing the noise to obtain a smooth image, except on those regions where there are discontinuities. In the next section, we modify the previous piece-wise restoration graphical models described in Section 3.6.2 to obtain an edge labeling model.

3.8.1 Edge Labeling using line process

The Posterior Energy function presented in the previous section for a MRF restoration process can be written as:

$$E(x) = U(x|y) = \sum_{n \in S} (\mathbf{x}_n - \mathbf{y}_n)^2 + \lambda \sum_{n \in S} \sum_{n' \in N} g(\mathbf{x}_n - \mathbf{x}_{n'}) \tag{3.30}$$

For the edge labeling problem $g(\mathbf{x}_n - \mathbf{x}_{n'})$ should be modified for the purpose of explicitly marking edges. In addition to the existing MRF for pixel values, it is necessary to introduce another MRF, called Line Process(LP) in which each label takes a value of 0 or 1 related to the occurrence of edges. The line processes are located in between adjacent intensity variables and denote the presence (or absence) of a discontinuity . Therefore, $g(\mathbf{x}_n - \mathbf{x}_{n'})$ is a function based on two coupled MRFs which are illustrated in Figure 3.10. One set is the existing lattice for the intensity field (pixels), and the other is the dual lattice for the introduced edge field.

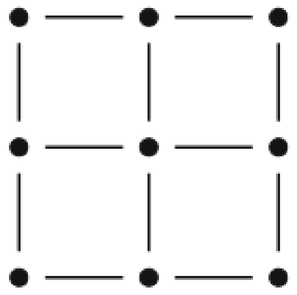


Figure 3.10: *Two-coupled MRF representation: A lattice of pixel sites(dots) and its dual edge sites(bars).*

Thus, we need to defined two MRFs: one for pixel sites (dots) represented by $\mathbf{x}_n^P$, and other for edges(bars) by $\mathbf{x}_{n,n'}^E$. The interaction between the two coupled MRF's is determined by the joint probability $P(x^P, x^E)$ or the prior energy $U(x^P, x^E)$. Then, we have to modify the prior energy into a function that depends on both MRF, intensity and edge variables as

$$U(x^P, x^E) = \sum_{n \in S} \sum_{n' \in N} (\mathbf{x}_n^P - \mathbf{x}_{n'}^P)^2 (1 - \mathbf{x}_{n,n'}^E) + \alpha \mathbf{x}_{n,n'}^E \tag{3.31}$$

Finally, adding the prior energy to the likelihood energy yields to the posterior energy for the edge labeling problem:

$$E(x) = \sum_{n' \in S} (\mathbf{x}_n - \mathbf{y}_n)^2 + \lambda \sum_{n \in S} \sum_{n' \in N} (\mathbf{x}_n^P - \mathbf{x}_{n'}^P)^2 (1 - \mathbf{x}_{n,n'}^E) + \alpha \mathbf{x}_{n,n'}^E \tag{3.32}$$

The first term on right hand side in Equation 3.32 attempts to keep the restoration of $\mathbf{x}_n$ close to the observed data $\mathbf{y}_n$ in a least square sense. While the second right hand term encodes the assumption that the data is smooth everywhere except at the discontinuities. The third term enforces a penalty for incorporating a discontonuity in the restoration. Also, minimization of Equation 3.32 has to be performed over all $\mathbf{x}_n^P$ and $\mathbf{x}_n^P$.

3.8.2 Simulations and results

Similarly to previous sections, the simulations were performed in two steps: an image restoration using MRF edge segmentation , and an interface level detection using a vertical edge operator. Table 3.4 has the parameters considered for the MRF edge segmentation. Figure 3.11 describes the process of detecting the interface level through MRF edge segmentation.

Parameters	**Description**
$\mathbf{x}_n^P$	Binary(0 for water and 1 for oil)
$\mathbf{x}_n^E$	Binary(0 for edge absence and 1 for edge presence)
μ	Column adaptive, estimated from column data
σ^2	Column adaptive, estimated from column data
α	0.042
N Neighborhood system	1st order
Minimization algorithm	ICM (Iterative conditional mode)

Table 3.4: *MRF binary restoration parameters*

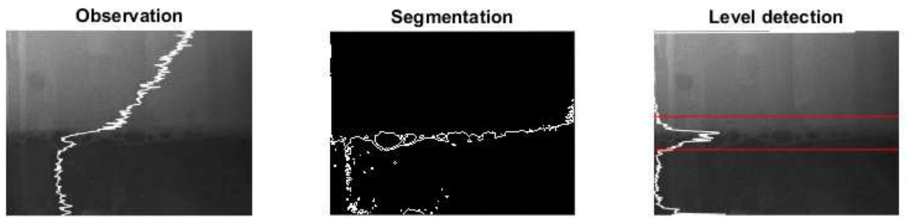


Figure 3.11: *Markov Random Field Edge detection. (a)Raw image and vertical profile, (b) MRF edge segmented image, (c) vertical change profile and level detection on MRF edge segmented image*

3.9 Conclusions

This chapter discusses the problem of detecting the level interface through image processing and analysis techniques on single static frames. The interface level detection problem can be divided into two main steps: a first step that consists of restoring the image and a second step to detect the feature of interest (interface). In the image restoration process, the objective is to take the corrupted and noisy image and estimate the clean and original image. Subsequently, a feature detection step based on vertical edge detection is performed on the restored image to detect the level interface.

In image restoration process, four model-based image optimization methods were investigated. The first two (Least Squares -WF-, KF and KS estimations) are based on the Bayesian Network graphical models. In each of them, we consider continuous states (from 0 to 1) for the restoration process, which provides a continuous smoothed image as output. The smoothed image output, $\hat{f}(x, y)$ or $\hat{\mathbf{x}}_n$, attempts to recover the pixel values of the original image $f(x, y)$ or $\mathbf{x}_n$. The last two methods (MRF segmentation and MRF edge segmentation) are based on MRF graphical models. In these cases, we consider discrete states (0 or 1) for the restoration process, which provides a binary segmented image as output. The image output, $\hat{f}(x, y)$ or $\hat{\mathbf{x}}_n$, is a segmented image of the original image $f(x, y)$ or $\hat{\mathbf{x}}_n$.

For the feature detection step, we utilize a vertical edge detector(mask) to find boundary changes along the vertical image profile. These boundaries delimit the three-phase regions in the image (oil, interface, and water) in the vertical change profile, and the maximum value in the profile corresponds to the interface level location.

In summary, the four image restoration methods achieved successful results for our enhancing objective, and the level interface was detected accurately by the vertical edge detector.

Chapter 4

Computer Vision Based Method for Froth-Middlings Interface Level Detection in the Primary Separation Vessels

[*] A robust computer vision based method to estimate the froth-middlings interface level on the primary separation vessel (PSV) is described in this chapter. Figure 4.1 describes the computer-vision system architecture for the froth-middlings interface level detection in the PSV, which is the most important control variable in the PSV operation. Bitumen recovery and downstream operations are critically dependent on the froth-middlings interface measurement and control. Typically, PSVs are equipped with sight glasses that allow the operations to track the level of the froth-middlings interface. The vision sensor consists of an algorithm that processes the online video frames of a camera mounted on the PSV sight glasses, and it is based on a combination of static and dynamic image processing techniques. Industrial results show that the computer vision algorithm is more accurate and reliable when compared to the other instruments, as well as more robust against the process and environmental abnormalities.

[*]A version of this chapter is submitted to the *Computers and Chemical Engineering*, **2018**

4.1 Introduction

In this chapter, we are motivated to investigate and develop a new approach to improve the accuracy of the camera/computer vision system-based froth-middlings interface level detection. Most of the previous approaches were solely based on the static image segmentation, which searches for the level location by processing the information in each frame separately. As a result, they had to deal with the problem of noisy detections (spurious edges [3]) or difficulties to classify the oil and water regions ([3] and [7]). However, the level is a feature in the frames that has dynamics in time. Thus, by considering methods that analyze the motion in subsequent frames, we can extract additional information about the level location. Therefore, we are motivated to combine the information from static and dynamic image processing methods to achieve an accurate interface level detection with less computational load. In addition, there are abnormal process scenarios that affect the level detection which have not been discussed and addressed in the previous works. Abnormal scenarios include such as the stains on the sight glasses, blur or fuzzy level interface, froth and middlings phases with non-homogenous pixel intensities, and level switch from one sight glass to the other. These were in fact the main causes for the noisy detections when using most of the previously proposed algorithms. Also, the previous works do not provide the framework to address the industrial conditions that affect the reliability of the level detection when using a video camera sensor. For example, scenarios such as people/objects blocking the camera, camera motion or vibrations, lighting changes, or poor image quality.

The main contribution of this chapter is the development of a robust computer vision-based method to estimate the forth-middlings interface online. The vision sensor consists of an algorithm that processes the video frames through three main steps and outputs the estimated level values. In the first step, the algorithm infers the level location in the frames based on a static and dynamic image processing technique. The static image processing method identifies the potential level locations with sharp pixel intensity changes (froth and middlings have different intensity colors), and the dynamic image processing picks the level location that has the maximum motion among the potential level locations detected by the static image processing method. In the second step, the algorithm uses a reliability analysis on the images to prevent inaccurate level estimations caused by the environmental scenarios that affect the sight glass visuals. This allows the algorithm to run online and with no supervision on the industrial site. In the third step, the algorithm performs a time-based sliding window analysis to remove outliers and smooth the level measurements.

Figure 4.1 shows the schematic of the computer-vision system design for the industrial PSVs. The setup has a video camera that captures and transmits the visuals of the sight glasses to the application PC, where the Video Processing Application runs in the background to infer the level from the video frames. Then, the estimated level value is communicated through Open Protocol Communication (OPC) to the Distributed Control System (DCS), and a video stream with a level indicator is transmitted to a webserver that can be visualized from the control room or any other machine on the network. The video processing algorithm was developed and tested in the Process Control Laboratory at the University of Alberta. Then, an application based on the algorithm was developed and deployed in two industrial PSVs, where it has been found more reliable and accurate than the other instruments.

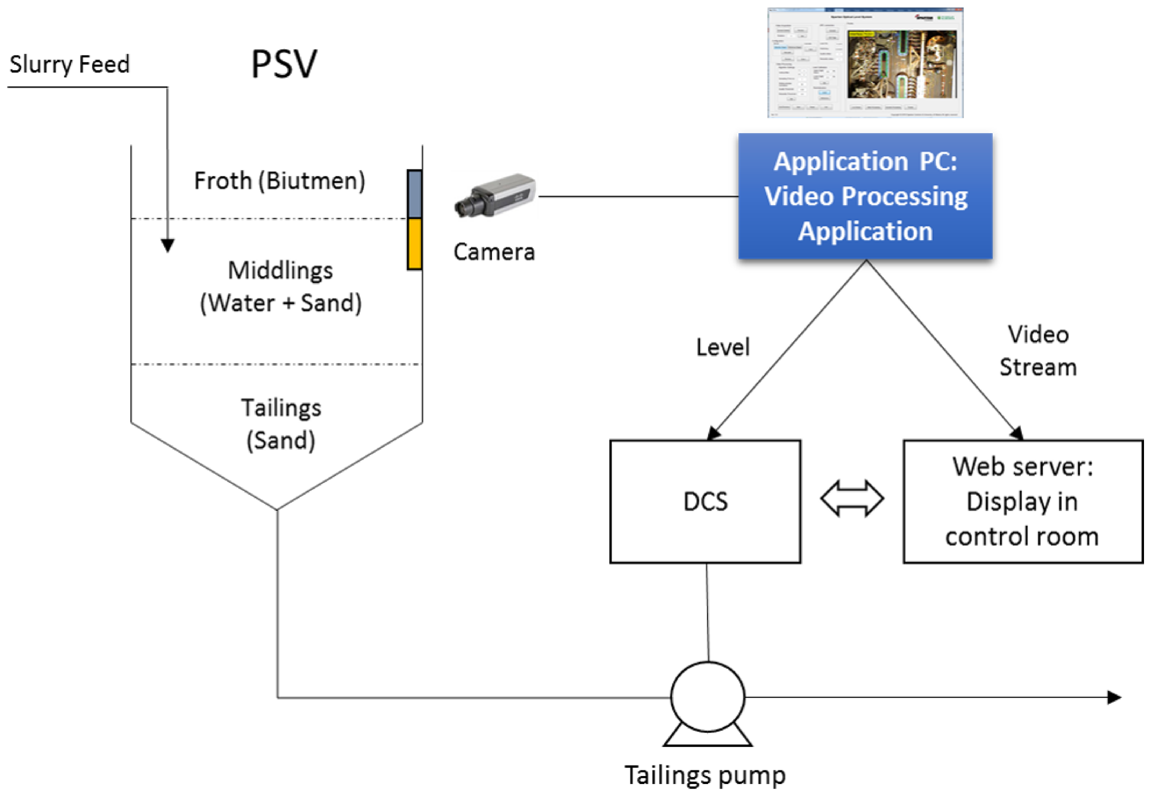


Figure 4.1: *Schematic of computer-vision system architecture for the industrial PSV. A camera installed on the sight glasses transmit the images to the Application PC, where the Video Processing Application (VPA) process the data and calculates the interface level. Then, the inferred level, among with other parameter, are communicated to the DCS though OPC. In addition, the VPA transmits the online stream with the level information to a web server, which can be visualized from the control room or any other machine.*

The rest of the chapter is organized as follows. Section 4.2 describes the video processing algorithm in detail. Section 4.3 describes the computer vision system architecture implemented on the industrial PSVs. Section 4.4 presents the results of the video level estimation on the industrial site, and Section 4.5 provides the concluding remarks.

4.2 Computer vision algorithm

Flowchart of the video processing application is shown in Figure 4.2. It illustrates all the steps and functions executed by the algorithm to infer the froth-middlings interface level. The algorithm takes the live video stream coming from the video device as the input and provides estimated level values with its timestamp as the output.

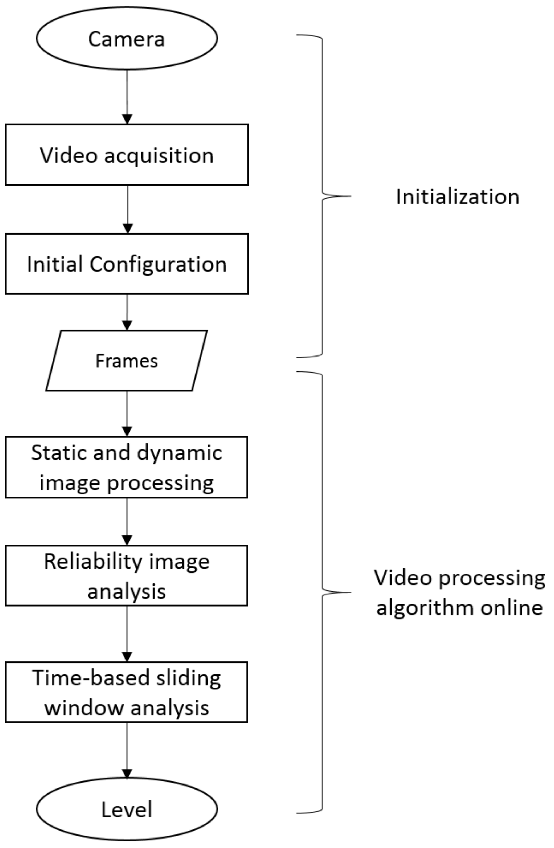

Camera
Video acquisition
Initial Configuration
Frames
Static and dynamic image processing
Reliability image analysis
Time-based sliding window analysis
Level
Initialization
Video processing algorithm online

Figure 4.2: *Flowchart of the video processing application. It corresponds to the sequence of main functions executed by the computer vision algorithm to estimate the level.*

The video processing application has an initialization part which consists of video acquisition settings and initial configurations, and the video processing algorithm part that runs online. The video processing algorithm has three important steps which are described later with more detail:

1- Static and dynamic image processing step

2- Reliability image analysis

3- Time-based sliding window analysis

4.2.1 Initial configurations

The video processing algorithm uses the following information, provided during the initialization, to perform the image processing calculations:

1. Sight glasses or the regions of interest (ROI) in the frames.
2. Reliability reference region in the frames.

These regions must be configured in the initial frames acquired from the video camera. Figure 5 shows an example of the configuration. The green rectangles correspond to the ROI where the algorithm executes the calculations to detect the level. The red rectangle corresponds to the reliability reference region where the algorithm executes the calculation to provide an alert during abnormal scenarios in the environment that affects the level estimation. Appendix A.2 provides more details about the initial configurations.

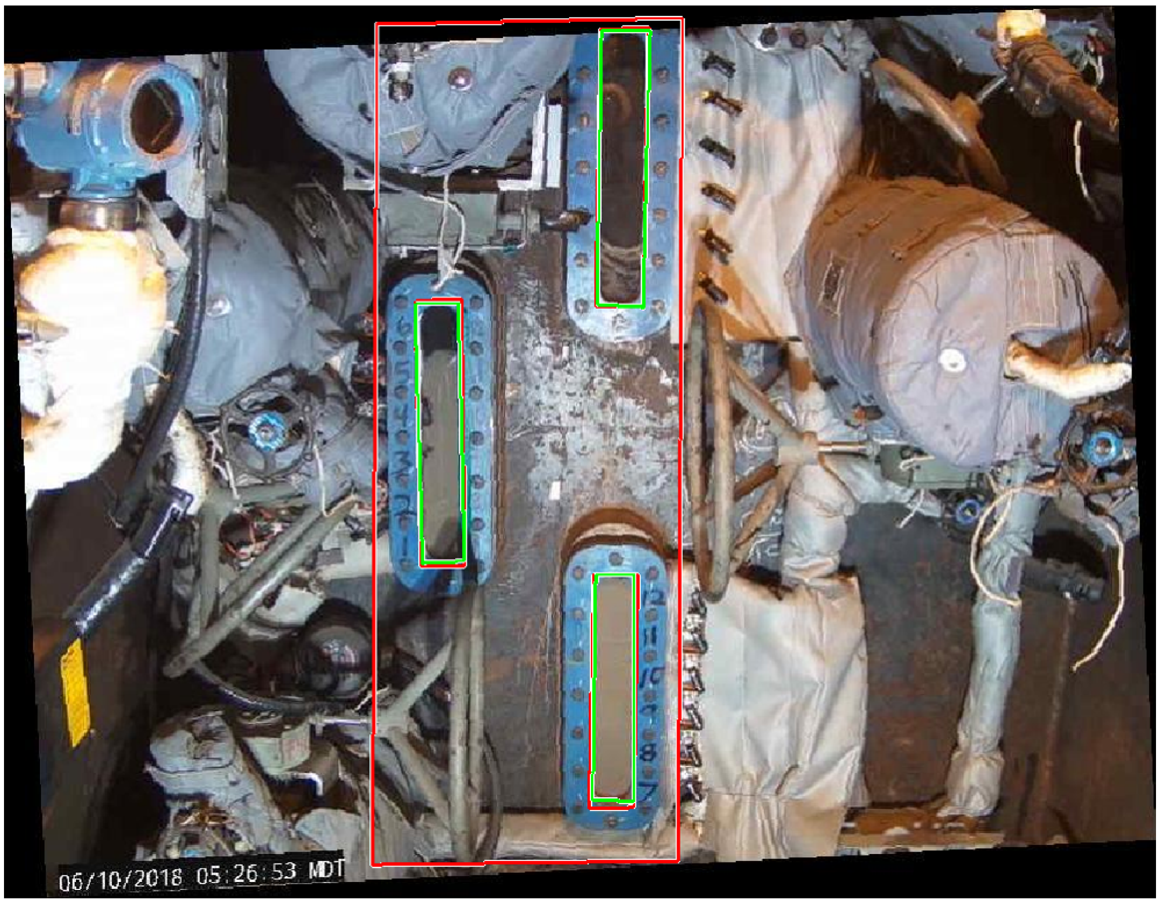


Figure 4.3: *Sight glasses or the regions of interest (green) and the reliability region (red). The green rectangles correspond to the sight glasses region where the algorithm executes the calculations to detect the level. The red rectangle corresponds to the region where the algorithm executes the calculation to alert during any abnormal scenarios in the environment that affects the level estimation.*

4.2.2 Static and dynamic image processing

The static and dynamic image processing step is essentially of edge and motion detection in a set of consecutive frames. Figure 4.4 shows the static and dynamic image processing algorithm flowchart, and Figure 4.5 illustrates the outputs of major steps in the algorithm. The calculations performed in each block of the algorithm flowchart are explained in detail in the following sections.

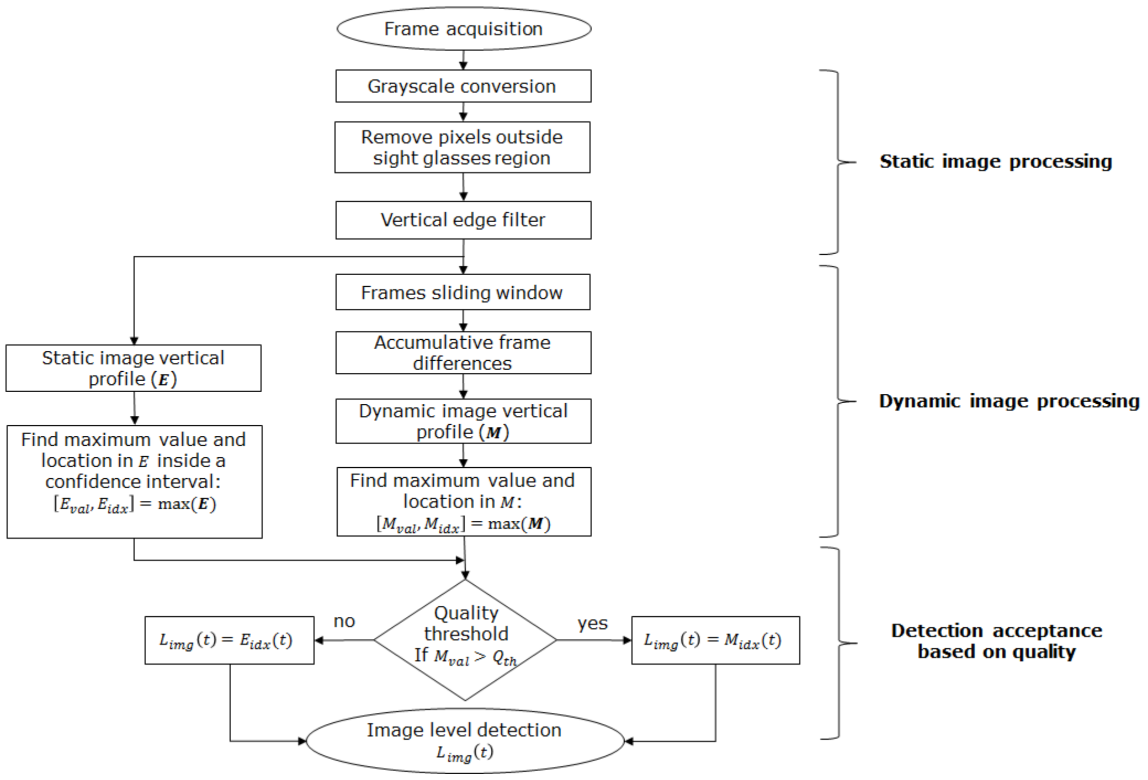

Frame acquisition
Grayscale conversion
Remove pixels outside sight glasses region
Vertical edge filter
Static image processing
Frames sliding window
Accumulative frame differences
Dynamic image vertical profile ($\boldsymbol{M}$)
Find maximum value and location in M: $[M_{val}, M_{idx}] = \max(\boldsymbol{M})$
Static image vertical profile ($\boldsymbol{E}$)
Find maximum value and location in E inside a confidence interval: $[E_{val}, E_{idx}] = \max(\boldsymbol{E})$
Dynamic image processing
Quality threshold If $M_{val} > Q_{th}$
no
yes
$L_{img}(t) = E_{idx}(t)$
$L_{img}(t) = M_{idx}(t)$
Detection acceptance based on quality
Image level detection $L_{img}(t)$

Figure 4.4: *Flowchart of the static and dynamic image processing step. The algorithm estimates the level based on static and dynamic image processing techniques performed on a set of temporal consecutive frames.*

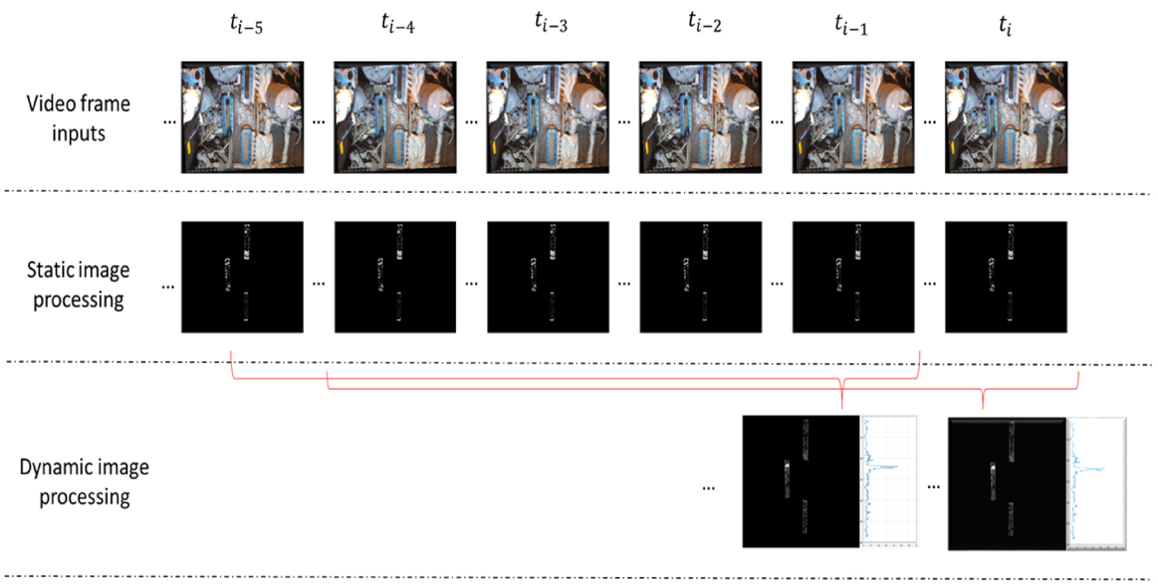


Figure 4.5: *Output of the static and dynamic image processing step. Top shows a set of 6 consecutive frames acquired from the camera, second row shows the static image processing output with edges highlighted in white pixels of the corresponding frames, and third row shows the output from the dynamic image processing step that utilizes five consecutive frames to generate one output with actual level highlighted in white pixel.*

Frame acquisition:

Figure 4.6, illustrates the frames, $\mathbf{X_{RGB}}(t_i)$, acquired by the algorithm from the video camera. $\mathbf{X_{RGB}}(t_i) \in \mathcal{R}^{M \times N \times 3}$ corresponds to a 3-dimensional RGB frame acquired at time instant t_i with frame length M and frame width N. In our study, we utilize a t_i of 1 frame/second.

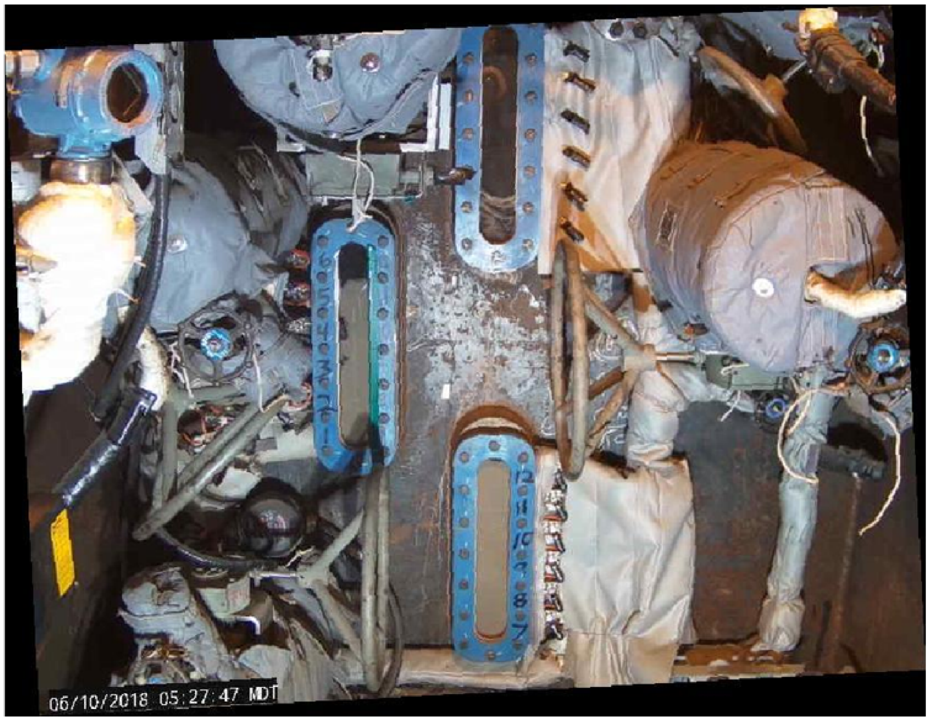


Figure 4.6: *Acquired image from video device mounted on PSV sight glasses.*

Grayscale and double conversion:

The acquired images are then converted from color (RGB: 3-D matrix) to grayscale (1-D matrix) to reduce the data size. Followed by that the Grayscale images are converted and rescaled to double precision data format, in which the pixels intensities take values that range from 0 to 1. After the conversion, we would have $\mathbf{X}_G(t_i)$. $\mathbf{X}_G(t_i) \in \mathcal{R}^{M \times N}$ correspond to the current grayscale intensity frame with length M in pixels and width N in pixels.

Region of interest (ROI):

The sight glasses regions are the target of the algorithm to detect the level. In the grayscale frames, all the pixels outside the sight glasses/ROI are removed by using a binary matrix ($\mathbf{BM_{SG}}$). $\mathbf{BM_{SG}} \in \mathcal{R}^{M \times N}$ is a matrix with the same size as the image frame that contains pixel values of 0 outside the ROI and 1 inside the ROI. Thus, by doing a point-wise multiplication of the image frames with the binary ROI matrix as shown Equation 4.1, all the pixels outside the ROI are converted into 0, while all the pixels inside the ROI keep the same value.

$$\mathbf{X_{ROI}}(t_i) = \mathbf{X}(t_i) * \mathbf{BM_{SG}} \tag{4.1}$$

Therefore, the subsequent image processing calculations will be limited only to the pixels constrained to the ROI in the frames ($\mathbf{X_{ROI}}(t_i)$).

Vertical edge filter:

In this step, a vertical edge filter runs over the ROI $\mathbf{X_{ROI}}(t_i)$. This filter strengthens the regions with sharp intensity changes, enhancing the potential level regions in the images [24]. In image processing, a mask (also called convolution matrix or kernel) is a small matrix that could be used for different purposes, such as blurring, sharpening, edge detection, or more [12]. The operation is accomplished by doing a convolution between the mask and the image. The vertical edge filter/mask used in this work is a matrix with positive and negative rows of ones stacked above and below a row of zeros as shown below,

$$\mathbf{VF} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

The above matrix is called the Prewitt vertical filter of size 3x3 in the literature [24–27]. The size of the vertical filter can be tuned to enhance the images in different ways. The size implies the cluster of pixel involved in the calculation. For instance, increasing the vertical size, by adding a row of ones to the top and negative ones to the bottom, the filter will consider more neighboring pixels vertically in the calculation. While increasing the column size, by adding a new column, [1; 0;-1], the filter will consider more neighboring pixels horizontally in the calculations. In addition, the element values in the filter determine the magnitude of the filter. The convolution mask calculation between the ROI and the vertical edge filter can be described by Equation (2).

$$\mathbf{X_S}(t_i) = \mathbf{X_{ROI}}(t_i) \otimes \mathbf{VF}\ , \tag{4.2}$$

where $\mathbf{X_S}(t_i) \in \mathcal{R}^{M \times N}$ correspond to the vertical edge filter output, shown in Figure 4.7. The white regions correspond to areas where there is a sharp intensity change in the sight glasses. These places correspond to potential static level locations on the image.

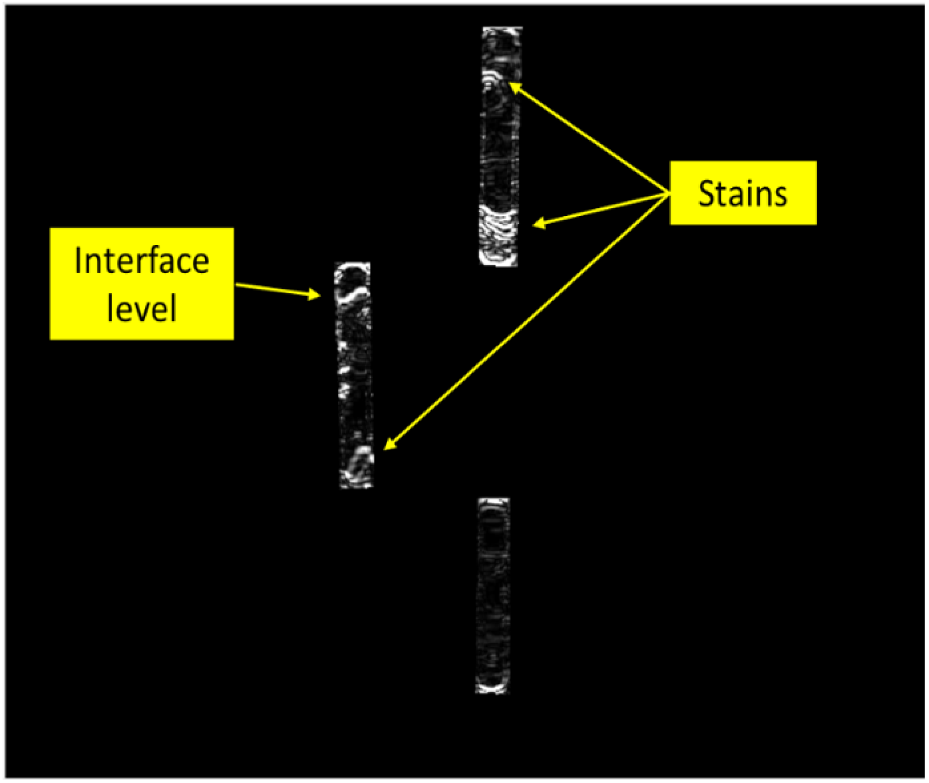


Figure 4.7: *Output of the vertical edge filtering with segmented horizontal lines/regions. White regions correspond to places where there is a vertical sharp contrast of pixel intensity (potential level locations).*

Frames sliding window:

The outputs of the vertical edge filter are stacked together in a sliding window which retains the current frame and discards the oldest frame in the window. We utilized a window size of five frames stacked with a sample time of T_s as shown below,

$$[\mathbf{X_S}(t_{i-4}), \mathbf{X_S}(t_{i-3}), \mathbf{X_S}(t_{i-2}), \mathbf{X_S}(t_{i-1}), \mathbf{X_S}(t_i)] \ . \tag{4.3}$$

where $\mathbf{X_S}(t_i) \in \mathcal{R}^{M\times N}$ correspond to the vertical edge filter outputs, and T_s is the distance between each frame. We provide more information about the sample time T_s in Appendix A.2.

Frame differentiating:

The objective of this operation is to find changes among the accumulated frames in the sliding window after the vertical edge filtering step. An absolute accumulative frame differences is performed as shown below,

$$\mathbf{D}(t_i) = |\mathbf{X_S}(t_i) - \mathbf{X_S}(t_{i-1})| + |\mathbf{X_S}(t_i) - \mathbf{X_S}(t_{i-2})| + |\mathbf{X_S}(t_i) - \mathbf{X_S}(t_{i-3})| + |\mathbf{X_S}(t_i) - \mathbf{X_S}(t_{i-4})| \,. \tag{4.4}$$

We call the resulting matrix, $\mathbf{D}(t_i) \in \mathcal{R}^{M\times N}$ as the dynamic image matrix. Figure 4.8 displays a sample of the dynamic image matrix $\mathbf{D}(t_i)$. The white region corresponds to the region with the most significant motion among all the potential level regions (white patches) detected in the previous static step, refer to Figure 4.7.

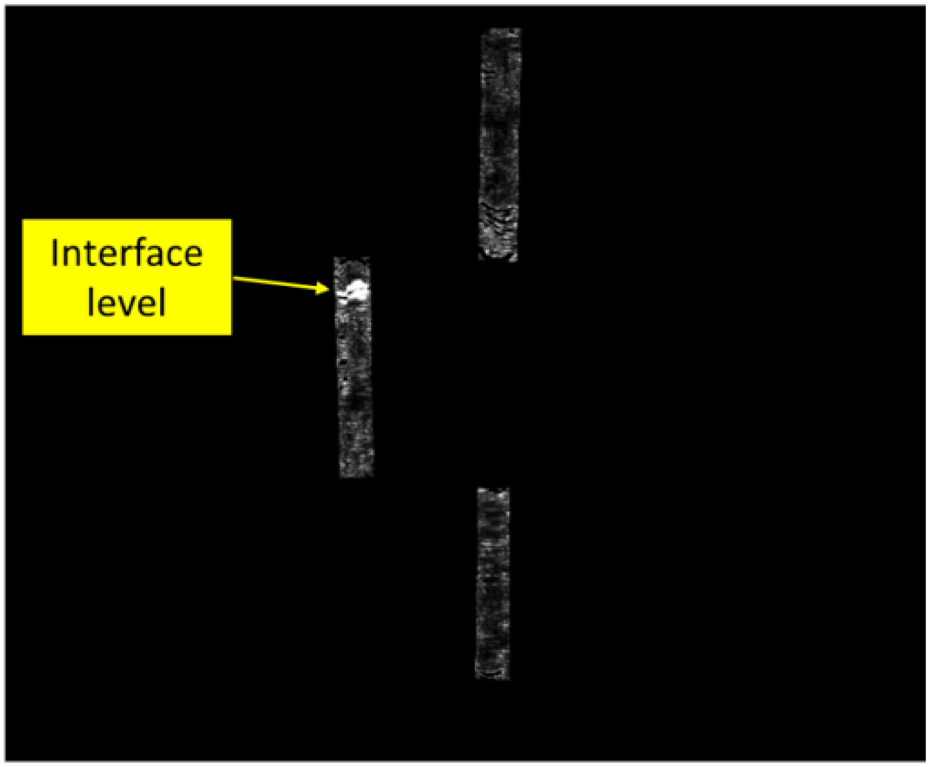


Figure 4.8: *Dynamic image processing output,* $\mathbf{D}(t_i)$*, after performing the absolute accumulative frame difference. White patch corresponds to the region where there is more significant motion.*

Frame differentiating increases the robustness of the algorithm. Increasing the number of frames for this operation (for example taking 6 consecutive frames instead of 5) will increase the motion detection robustness. However, this increases the computational load. Therefore, the selection of number of frames is a trade off between the algorithm accuracy and the computational load.

Level detection in the vertical profile:

This function determines the pixel index location corresponding to the current level. The algorithm computes vertical profile of the ROI and then looks for the maximum value in this profile which should correspond to the interface level. In the first step, the algorithm obtains the vertical profile of the dynamic matrix, $\mathbf{M}(m, t_i)$, by performing the horizontal mean over every row of the dynamic matrix $\mathbf{D}(t_i)$, as shown in Equation

4.5,

$$\mathbf{M}(m, t_i) = \frac{\sum_{n=1}^{N} \mathbf{D}(m, n, t_i)}{\sum_{n=1}^{N} \mathbf{BM_{SG}}(m, n)} \ . \tag{4.5}$$

Figure 4.9 depicts the vertical profile, $\mathbf{M}(m, t_i)$, to the right of the dynamic matrix $\mathbf{D}(t_i)$. This profile is a column vector. If the detection has been done correctly and the level has been moving, the peak of this profile corresponds to the location of the level. In the second step, the algorithm finds the pixel location ($M_{idx}(t_i)$: Motion location) of the maximum value in the profile and the maximum value ($M_{val}(t_i)$: Motion magnitude) in the vertical profile, $\mathbf{M}(i, t_i)$ as shown in Equations 4.6 and 4.7,

$$M_{idx}(t_i) = \max_{m}(\mathbf{M}(m, t_i)) \ , \tag{4.6}$$

$$M_{val}(t_i) = \mathbf{M}(M_{idx}(t_i), t_i)) \ . \tag{4.7}$$

$M_{idx}(t_i)$, would correspond to the location (pixel coordinates) of the level in the frame. The motion magnitude or the largest element value, $M_{val}(t_i)$, has information about the significance of the actual change. If the region is very bright, this number will be large, meaning that there is a significant change in the interface level location. Otherwise, the detection is considered to be weak, and the maximum of the profile will be smaller than what could be attributed as an actual motion.

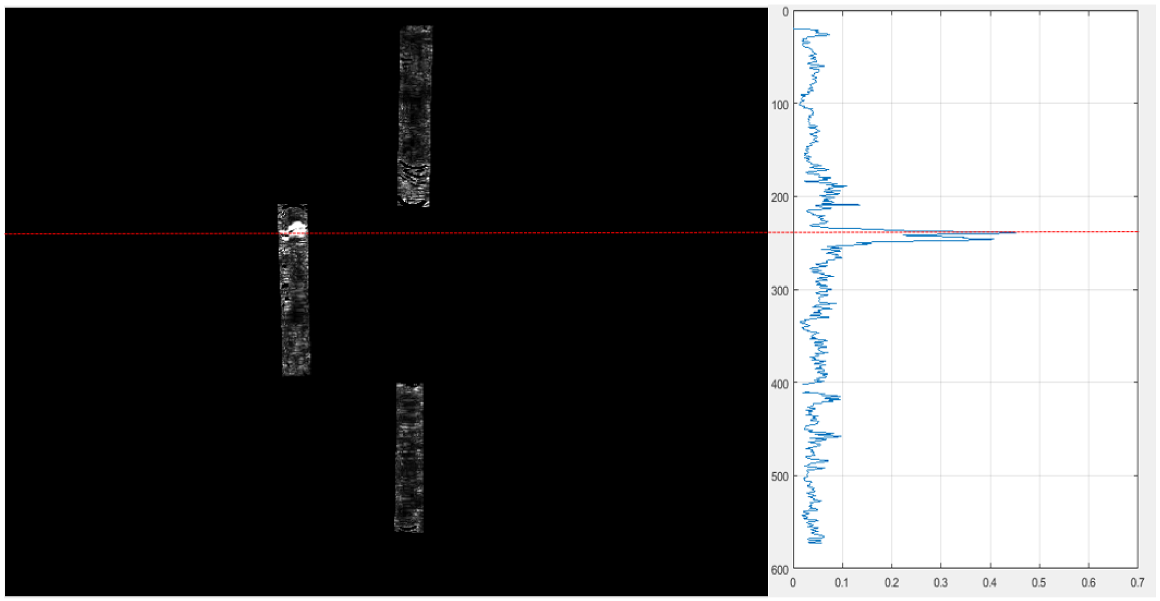


Figure 4.9: *Vertical mean profile,* $\mathbf{M}(i, t_i)$*, beside the accumulative frame-difference output,* $\mathbf{D}(t_i)$*. . The peak represents the region with more significant motion on the image. The algorithm searches the vertical profile maximum value to locate the level.*

When the detection is found to be weak or there is no motion, the algorithm fails over to the static image processing-based detection. It obtains the vertical profile of the static images with the edges , $\mathbf{E}(m, t_i)$. Equations 4.8, 4.9, and 4.10 depict the calculations.

$$\mathbf{E}(m, t_i) = \frac{\sum_{n=1}^{N} \mathbf{X_S}(m, n, t_i)}{\sum_{n=1}^{N} \mathbf{BM_{SG}}(m, m)} \ , \tag{4.8}$$

$$E_{idx}(t_i) = \max_{m}(\mathbf{E}(m, t_i)) \tag{4.9}$$

$$s.t. \quad L_s(t_{i-1}) - CI \leq \mathbf{E}(m, t_i) \leq L_s(t_{i-1}) + CI \ ,$$

$$E_{val}(t_i) = \mathbf{E}(E_{idx}(t_i), t_i) \ . \tag{4.10}$$

The algorithm looks for the maximum value ($E_{val}(t_i)$: Edge magnitude) and the index ($E_{idx}(t_i)$: location of the edge) in a confidence region (CI) constrained by the last effective level detection ($L_s(t_{i-1})$). In other words, the algorithm looks for the edge around the last detected region by the motion detection calculation.

Level detection based on the quality threshold:

Algorithm 1 presents the logic for level detection based on the quality threshold. It determines the level detection by comparing the maximum value in the dynamic vertical profile, $M_{val}(t_i)$,which we call as the quality index Q_{idx}, to a quality threshold, Q_{th}. For Q_{idx} above Q_{th} the algorithm estimates the level based on the vertical profile of the , $M_{idx}(t_i)$ and for Q_{idx} below Q_{th} the algorithm estimates the level based on the vertical profile of the static images $E_{idx}(t_i)$. Finally, the accepted level value on the image, $L_{img}(t_i)$, is sent to the reliability image analysis step described in the next section.

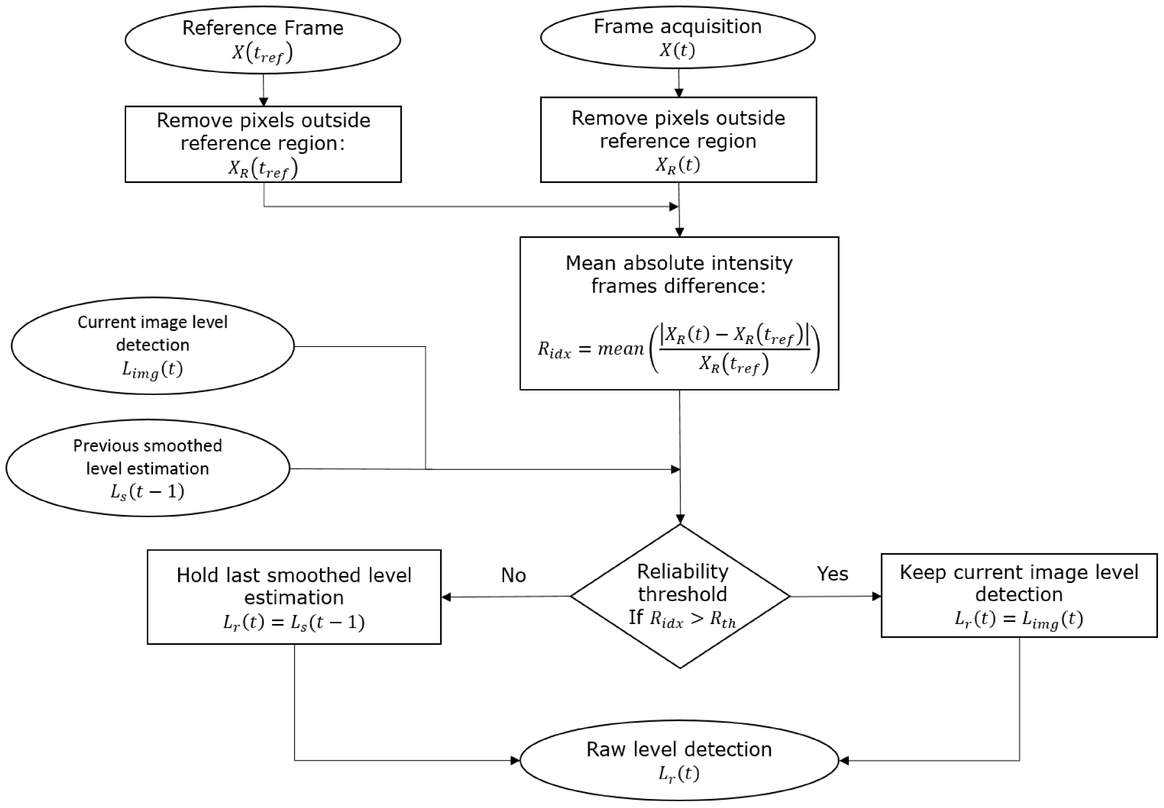
```
begin
    if $Q_{idx}(t_i) > Q_{th}$ then
        $L_{img}(t_i) = M_{idx}(t_i)$
    else
        $L_{img}(t_i) = E_{idx}(t_i)$
    end
end
```

Algorithm 1: Detection acceptance based on quality threshold

In addition, if Q_{idx} is smaller than Q_{th} for a sustainable period, the algorithm will send an uncertain quality alarm to the user. Typical uncertain level scenarios occur when the level is not visible, such as when it is behind a stain on the sight glass, is switching between sight glasses, or outside the sight glasses visual span. $M_{val}(t_i)$ and Q_{th} are displayed online in the figure trends of the VPA (Appendix A.1). The quality threshold value, Q_{th}, should be tuned to the conditions in the process (Appendix A.2).

4.2.3 Reliability image analysis

The purpose of the reliability image analysis is to prevent the level detection from abnormal scenarios in the environment, such as people/objects crossing in front of the camera, unintentional camera motions or vibrations, lighting changes, and conditions that affect the quality of video acquisition (mist, condensations, ice). The algorithm will hold the last reliably estimated level value if the image acquisition conditions have been changed, or eventually alert the user if the conditions have been changed drastically that prevents the accurate tracking of the level. Figure 4.10 describes the reliability analysis flowchart, and each block operations are explained in the following sections.



Reference Frame $X(t_{ref})$
Frame acquisition $X(t)$
Remove pixels outside reference region: $X_R(t_{ref})$
Remove pixels outside reference region $X_R(t)$
Mean absolute intensity frames difference: $R_{idx} = mean\left(\frac{|X_R(t) - X_R(t_{ref})|}{X_R(t_{ref})}\right)$
Current image level detection $L_{img}(t)$
Previous smoothed level estimation $L_s(t-1)$
Hold last smoothed level estimation $L_r(t) = L_s(t-1)$
No
Reliability threshold If $R_{idx} > R_{th}$
Yes
Keep current image level detection $L_r(t) = L_{img}(t)$
Raw level detection $L_r(t)$

Figure 4.10: *Reliability image analysis flowchart. The objective is to prevent any abnormal scenario in the environment, such as people/objects affecting the camera visuals, changes in lighting and environmental conditions.*

Reliability reference region(RRR):

As mentioned in the Initial Configurations section, the algorithm removes all pixels outside the selected RRR by employing a binary matrix ($\mathbf{BM_R}$) on the images, similar to the one described previously in Section 4.2.2,

$$\mathbf{X_R}(t_{ref}) = \mathbf{X}(t_{ref}) * \mathbf{BM_R} \ . \tag{4.11}$$

The output image, $\mathbf{X}(t_{ref})$, is the RRR, which consists of the selected red rectangle area excluding the ROI (Figure 4.3). The algorithm performs this operation during the initialization and considers the initial frame as the RRR.

Current reliability region:

When the algorithm runs online, takes the current image, $\mathbf{X}(t_i)$, and removes all pixels outside the reliability region using the same calculation in Equation 4.11. The output image, $\mathbf{X_R}(t_i)$, is the current reliability region.

Reliability regions comparison:

The current reliability region, $\mathbf{X_R}(t_i)$, and the RRR, $\mathbf{X_R}(t_{ref})$, are compared continuously by considering the mean absolute intensity difference in the reliability region as shown below,

$$R_{idx}(t_i) = mean\left(\frac{|\mathbf{X_R}(t_i) - \mathbf{X_R}(t_{ref})|}{\mathbf{X_R}(t_{ref})}\right) , \tag{4.12}$$

where $\mathbf{X_R}(t_{ref})$ represents pixels intensity from the RRR, and $\mathbf{X_R}(t_i)$ represents the pixels intensity from the current reliability region. The subtraction and division operation between $\mathbf{X_R}(t_i)$ and $\mathbf{X_R}(t_{ref})$ are performed element wise (pixel by pixel). The output, $R_{idx}(t_i)$, is a scalar value, and compared against a reliability threshold to determine if the environmental conditions are abnormal or not.

Detection acceptance based on reliability threshold:

The algorithm decides the reliability of the level detected by the image processing algorithm based on a reliability threshold R_{th}. Algorithm 2 presents the logic for the detection acceptance based on the reliability threshold.

```
begin
    if $R_{idx}(t_i) > R_{th}$ then
        $L_r(t_i) = L_{img}(t_i)$
    else
        $L_r(t_i) = L_s(t_(i-1))$
    end
end
```

Algorithm 2: Level detection acceptance based on reliability threshold

If the $R_{idx}(t_i)$ is greater than R_{th} , the algorithm will consider that the environmental conditions are normal and will rely on the level estimation from the image processing algorithm $L_r(t_i) = L_{img}(t_i)$. However, if $R_{idx}(t_i)$ is smaller than R_{th}, then the algorithm will assume that environmental conditions are abnormal, and hold the last reliable level value. Moreover, if $R_{idx}(t_i)$ is smaller than R_{th} for a sustainable period of time, then the algorithm will alarm the user about an abnormal condition in the environment of the room.

4.2.4 Level tracking over time-based sliding window analysis

The level is a physical quantity that has coherence and continuity in time, and thus we do not expect to observe a sharp raise or dip in the level measurements. The time-based sliding window analysis of the most recent detected level values is performed to smooth the level trends and remove the unrealistic jumps in the level measurements. Figure 4.11 summarizes the flowchart of the time sliding window calculations. The sliding window length, W, is defined a priori. A larger sliding window size implies stronger and smoother outlier filtering.

In the first step, the algorithm computes the local median (m) and standard deviation (σ) inside the window and replaces any outlier (outside the 3σ threshold) by the median value (m). The algorithm uses the Hampel filters, which replaces the outliers based on past and future neighbors of each sample inside the measurement window [28].

In the second step, the algorithm performs a moving average calculation inside the window to smooth the values based on past and future elements in the array.

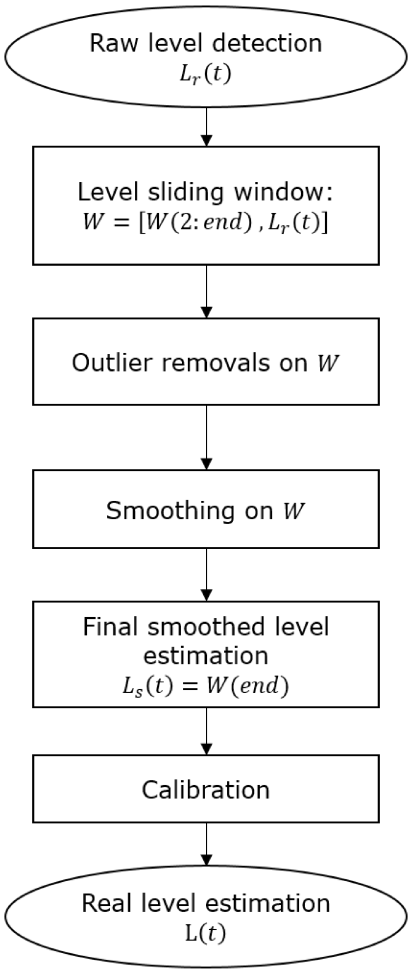


Figure 4.11: *Time-based level sliding window analysis and calibration flowchart. It is used to remove outliers, smooth and calibrate the level estimation based on the most recent measurement.*

Level calibration:

The output of the sliding window analysis is then calibrated to engineering units which is the final output of the algorithm. Figure 4.12 illustrates in a video frame the calculations involved in the algorithm to calibrate the level location. In this example, the calculations are provided for a vessel with three sight glasses, but it could be adapted to any other number of sight glasses.

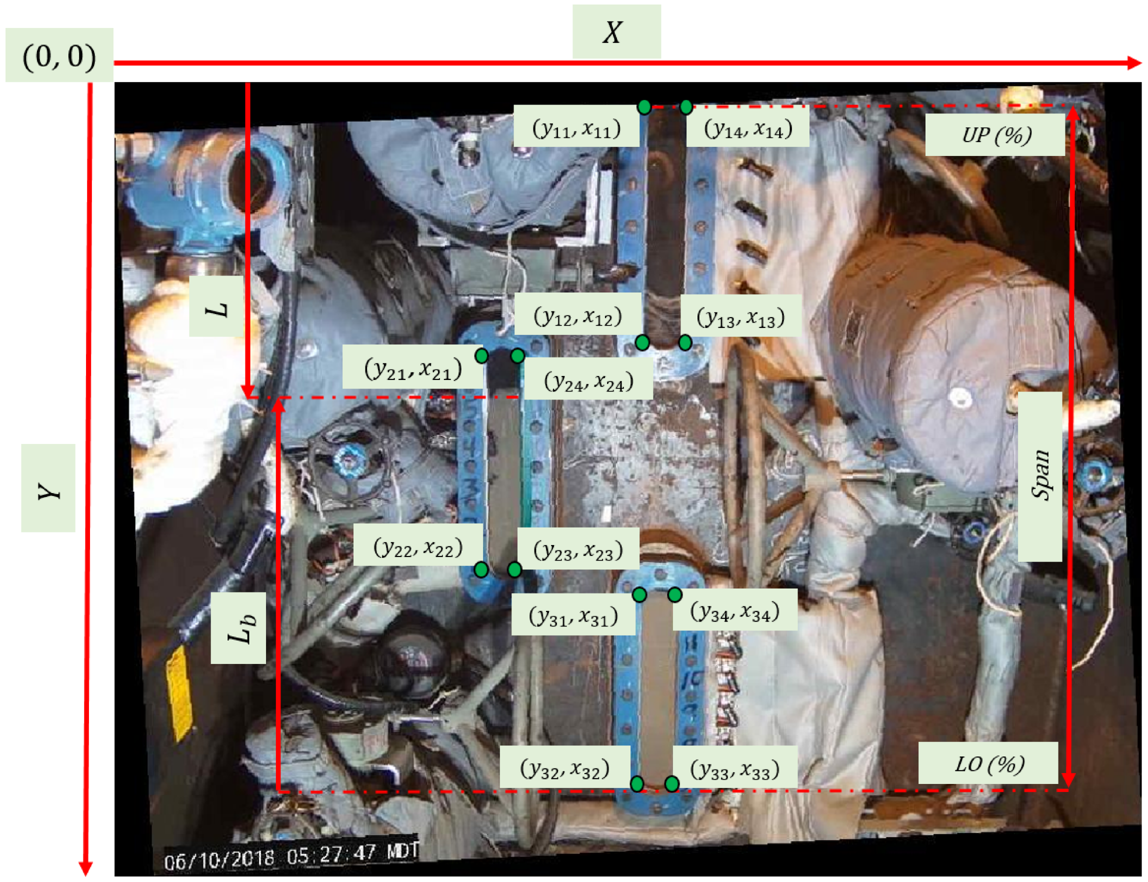


Figure 4.12: *Level calibration from pixels to engineering units performed on the image frame. The coordinates obtained from the ROI are used as reference to calibrate the level.*

The coordinates obtained from the ROI are employed by the algorithm to perform the calculations. The top left corner in the frame is the origin $(0, 0)$. Thus, vertical pixel distances are obtained with respect to the top border of the image, while horizontal distances are obtained with respect to the left border of the image. The Equations 4.13,4.14 and 4.15 are used for the level calibration considering the coordinates displayed in Figure 4.12.

$$Span = y_{3,2} - y_{1,1} \ , \tag{4.13}$$

$$L_b = (Y - L) + (Y - y_{3,2}) \ , \tag{4.14}$$

$$Level = \frac{UP - LO}{Span} * L_b + LO \ , \tag{4.15}$$

where $y_{3,2}$ is the bottommost coordinate of the lower sight glass , $y_{1,1}$ is the topmost coordinate of the upper sight glass in pixels, and $Span$ is the level span in pixels covered by the sight glasses. Y is the length of the entire frame in pixels, L is the level location in pixels measured from the top, and L_b is the level location in pixels measured from the bottom. UP and LO are the levels in engineering units corresponding to the top part of the upper sight glass, and the bottom part of the lower sight glass respectively in the vessel. $Level$ is the calibrated real level estimation in percentage, which is communicated to the DCS.

4.3 Computer vision system architecture

Figure 4.1 shows the schematic of the computer-vision system design for the industrial PSV. A video camera captures the visuals of the sight glasses, and transmits to a PC, called the application PC, where the video processing application(VPA) is installed and running. In the application PC, the video processing application runs in the background to infer the level from the video frames. Then, the value of the inferred level is communicated through Open Protocol Communication (OPC) to the DCS, and a video with an overlaying marker indicating the inferred level is transmitted to a web server that can be visualized from the control room or any other machine on the network.
Flowchart of the VPA architecture is shown in Figure 4.13. The VPA reads the camera feed through the network in the communication layer. Then, the video acquisition and settings layer allow the user to set the video acquisition settings and the tuning parameters of the video processing algorithm. Video processing algorithm layer process the video feed and executes the level estimation algorithm. Video processing algorithm provides the estimated level values and the video frames with level indicators as the outputs. The estimated level values are communicated to the DCS through OPC and the video frames with the level indicators are communicated through ISS communication to the webserver.

The VPA has a graphical user interface (shown in Appendix A.1) to execute,

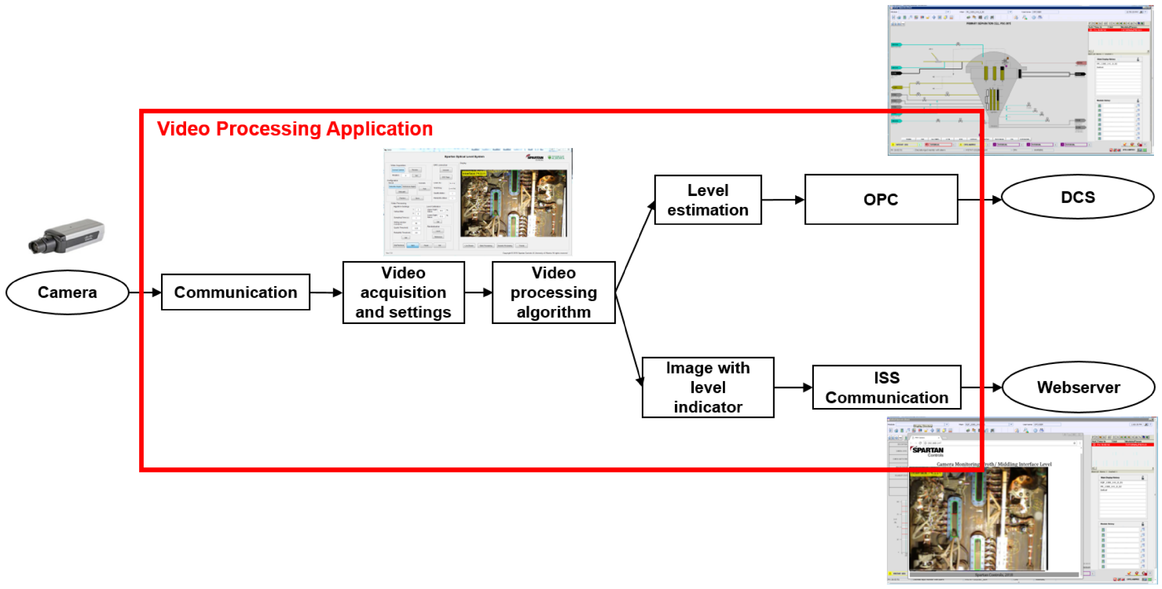

Video Processing Application
Camera
Communication
Video acquisition and settings
Video processing algorithm
Level estimation
OPC
DCS
Image with level indicator
ISS Communication
Webserver

Figure 4.13: *Video Processing Application architecture. Inside the red rectangle are the main function executed by the VPA. It communicates to the video camera, where the inputs are the video images, and transmit the level estimation and live stream to the DCS and webserver respectively.*

configure, monitor, and control the video processing algorithm. In addition, it establishes the communication settings for inputs (Camera) and outputs (DCS and webserver). The video processing algorithm and its GUI were developed using MATLAB Version 2015b.

4.3.1 Video algorithm configuration and tuning

The video processing algorithm has parameters that can be tuned to adjust the calculations to different operational conditions or to improve the accuracy of the level estimation. These parameters depend on the process behavior, the environmental conditions on the sight glass monitoring room, and the image quality provided by the acquisition device.

Appendix A.2 provides a detailed description of the configuration and tuning parameters listed as follow: The sight glasses or ROI, the RRR, the number of frames for static-dynamic image processing, the sample time of the dynamic image processing, the size of the vertical edge filter, the time-based sliding window length, the quality threshold and the reliability threshold.

4.4 Results

4.4.1 Computer vision system implementation

Initially, the video processing application was developed and tested in the Process Control Laboratory at the University of Alberta. A Primary Separation Vessel (PSV) experimental setup was used during the research and development (R&D) stage. A factory acceptance test (FAT) was conducted in the lab to test the algorithm performance under process and environmental conditions like the ones in the industry. In addition, recorded video files from the industrial PSV sight glasses were employed during the FAT. After the R&D stage, the vision system has been deployed and installed in two industrial PSVs. In both cases, a site acceptance test (SAT) was conducted to test the algorithm performance in the industrial environment, and it has shown that the sensor has a reliable and accurate operation. Figure 4.14 depicts an image output of the online stream display of the video processing algorithm with the level information overlay.

Industrial results show that the algorithm is robust to different process abnormalities, such as blurry interface, level switch from one sight glass to other, stains in the sight glasses and PSVs running on water. In addition, the reliability analysis can handle abnormalities in the environment, such as people/objects blocking the camera visuals, unintentional camera motions, lighting changes, and conditions that affect the quality of image acquisition (mist, condensation, ice).
The vision sensor provides a level estimation that is more reliable and accurate when compared to the other instruments in the process. Figures 4.15 and 4.16 compares the camera level estimation to the two other instruments in the PSV

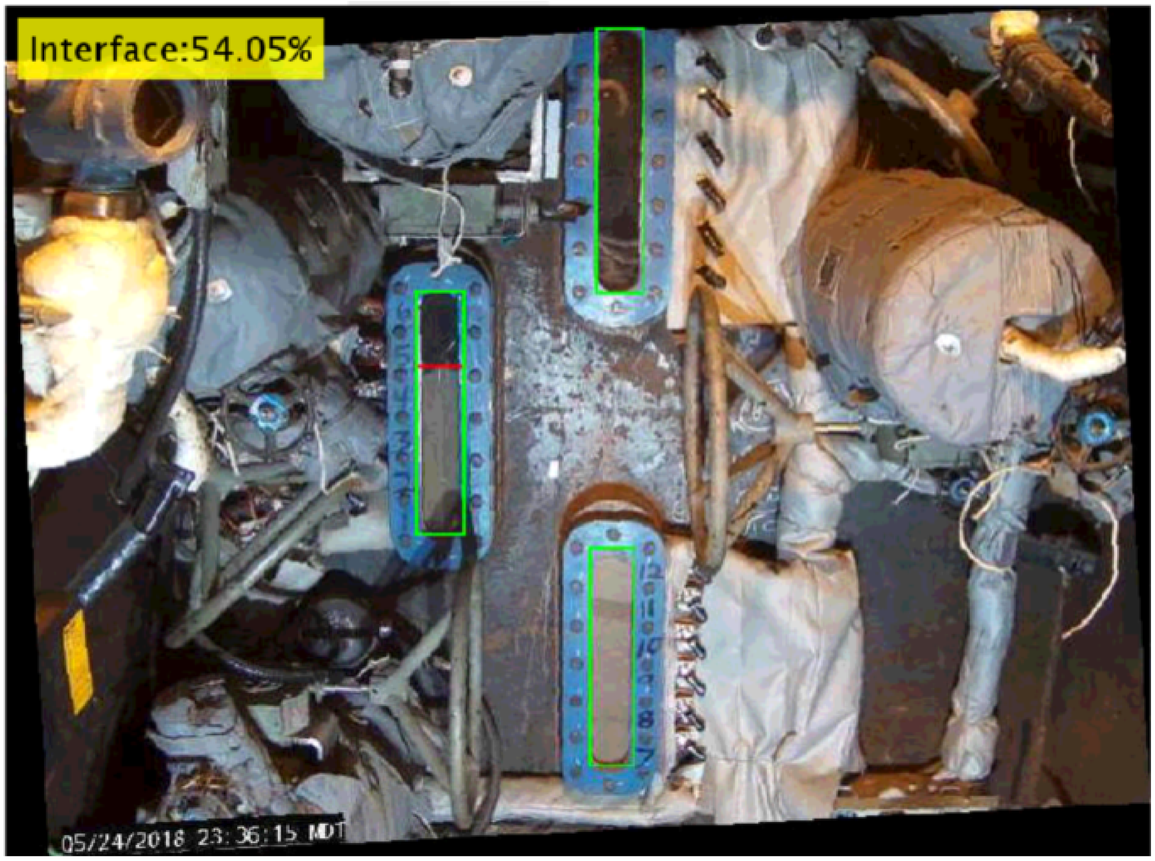


Figure 4.14: *Online stream display with the level information overlay. This output can be visualized on the VPA GUI display, or through the web browser in any other machine connected to the network.*

(nucleonic density profiler and differential pressure cells). Figures 4.15 corresponds to two hours of data trends, while Figure 4.16 is a three hours data trend. Essentially, both instruments had a bias with respect to the camera level estimation; Nucleonic profiler had a the larger Mean Squared Error (MSE). Table 4.1 has the comparison MSEs between the camera level estimation and the other two instruments in the process.

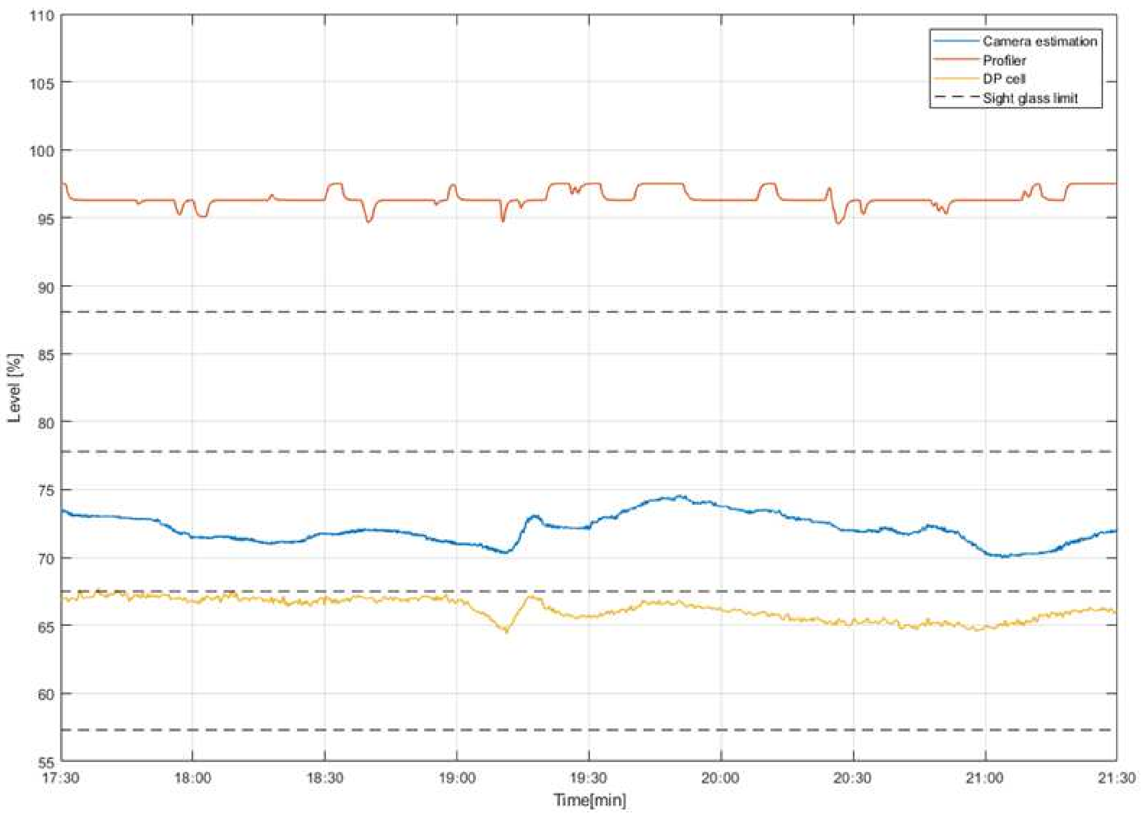


Figure 4.15: *Camera estimation scenario 1. Red line: Nucleonic profiler estimation; Blue line: Camera estimation; Yellow line: DP cell estimation; Black dashed line: Sight glasses borders.*

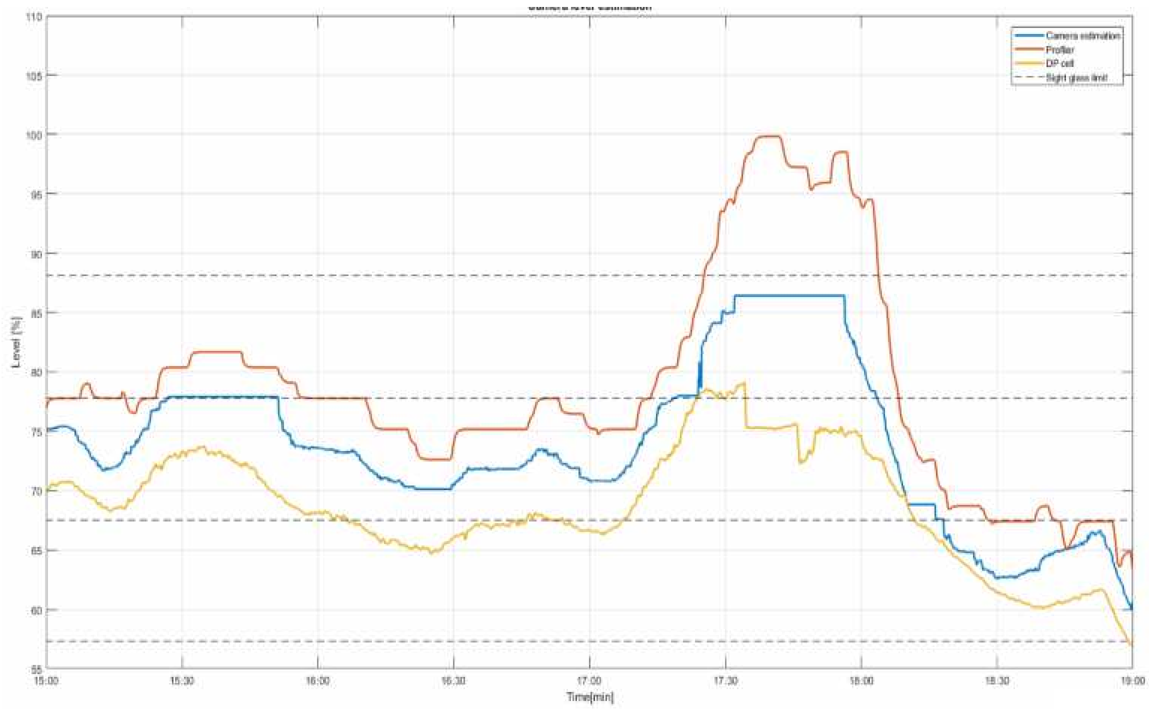


Figure 4.16: *Camera estimation scenario 2. Red line: Nucleonic profiler estimation; Blue line: Camera estimation; Yellow line: DP cell estimation; Black dashed line: Sight glasses borders.*

	Nucleonic Profiler MSE	**DP Cells MSE**
Scenario 1	595.36	36.02
Scenario 2	34.56	32.16

Table 4.1: *Mean Squared Error (MSE) comparison between camera level estimation vs nucleonic profiler and DP cells for each scenario.*

In the SAT, the video algorithm was monitored for a month and it showed a reliability service factor of 99.54% in PSV 1 and 99.92% in PSV 2. It means that during 0.46% and 0.08% of the time of each PSV operation, there was an unreliable environmental condition (reliability index went below the reliability threshold) affecting the level estimation where the algorithm held the last accurate value. These unreliable scenarios were generated by operators working in the room and blocking the camera visuals. In addition, the quality of the level estimation was monitored during the same month resulting in quality service factors of 62.21% for PSV 1 and 68.04% for PSV 2. It implies that 37.79% and 31.96% of the time the level presented a less reliable estimation, with a quality warning from the algorithm in which it had less dynamics (below the quality threshold). These scenarios are associated with moments when the level is outside the sigh glasses visuals, level switches from one sight glass to other or level occluded by the stains on the glasses. Finally, the accurate and reliable estimation of the vision sensor facilitates the closed-loop control of the froth-middlings interface. It helps to stabilize the extraction process in the PSV, increase the bitumen recovery, and reduce the variability in the downstream operations.

4.5 Conclusion

A computer vision system based on image and data analysis techniques is developed to process the online video stream of a camera mounted on PSV sigh glasses. It detects and communicates the inferred level to the distributed control system (DCS) facilitating the implementation of closed-loop control.
The computer vision algorithm is designed with three main steps, in which the inputs are the video frames, and the outputs are the estimated level values. In the first step, the algorithm detects the level in the frames based on edge and motion detection performed on a set of consecutive frames. In the second step, the algorithm uses a reliability analysis of the images to handle the abnormal environmental conditions that affect the sight glasses visuals. In the third step, the algorithm performs a time-based sliding window analysis to eliminate outliers and provide smooth level tracking.
The computer vision algorithm was developed in the Computer Process Control laboratory using a PSV experimental setup, and was later implemented in two industrial PSVs. The results under the industrial environment are presented, and they show that the algorithm is accurate, reliable and robust against the process

and environmental abnormalities.

The video algorithm has two other important advantages for the PSV operation. First, it is a non-intrusive sensor in the sense that it does not require a vessel shut down (with respective oil production losses) to do the installation and maintenance. Additionally, the level estimation of the vision sensor can be employed online as a reference value to recalibrate the other instruments (nucleonic density profiler and differential pressure cells).

Chapter 5

Image Restoration and Object Tracking using FIR Structures

[*] In this chapter, we address the problem of smoother design for state estimation based on a finite number of measurements collected in a finite estimation horizon. Three different finite impulse response (FIR) smoothing algorithms are proposed using the maximum likelihood FIR estimation, which is robust against uncertain noise statistics and modeling parameters, and also independent of the initial states of each finite horizon. Moreover, we provide equivalent but iterative Kalman-like structures of these algorithms for practical implementation. The applications of the proposed smoothing algorithms to an object tracking and image processing examples are demonstrated, and it shows that their have better robustness against modeling uncertainties than traditional smoothing approaches. Finally, we proposed the implementation of FIR algorithms to the image restoration process of PSV's interface level, and to the level tracking problem in the time-based sliding window.

5.1 Introduction

Digital filters can be classified in two types based on the duration of their impulse response: the infinite impulse response (IIR) filters, and the finite impulse response

[*]A version of this chapter was submitted to the *IEEE Transaction on Systems, Man and Cybernetics: Systems*, **2018**

(FIR) filters [29, 30]. IIR filters consider all the available information from a data-set, while FIR filters use the most recent measurements (finite horizon) within the data-set. FIR filters are also known as non-recursive filters as they do not have the feedback (a recursive part of a filter), even though a recursive algorithm can be used for FIR realization at each time step [31, 32].

As an IIR filter, the Kalman Filter (KF) [23, 33, 34] is the most popular estimator for linear systems with the Gaussian noise. It is well known that KF achieves the optimal estimation based on all previous measurements. However, it has been proved that KF only gives the optimal estimation under certain conditions [35, 36]. For example, to implement the KF optimally, a complete specification of dynamical and statistical model parameters must be given [37–39], which are not exactly known in most practical cases. In addition to KF, Kalman smoothers have been developed to achieve a linear optimal estimation of any point in the past. In [40], a forward-backward smoother is derived as a combination of two optimum filters based on the maximum likelihood principle. In [41], the Rauch-Tung-Striebel (RTS) smoother is presented as a more computationally efficient algorithm than the forwardbackward smoother, which computes the smoothed estimates with a backward pass. Nevertheless, all these smoothers suffer from the same weakness as the KF, which is a consequence of the IIR structure. In general, it is crucial to know the system modeling parameters and noise statistics before using the KF or smoothers [42–44].

Contrary to IIR, FIR filters utilize finite measurements collected over the most recent time interval, and thus have some advantages, such as guaranteed stability, linear phase and robustness to parameters changes, and round-off errors [43, 45–48]. The development of FIR filtering following the theory of receding horizon control was initiated by Kwon [43]. Later, Shmaliy developed a linear optimal FIR estimator to deal with the problems of filtering, smoothing and prediction in the state-space [49–52], which was further extended to non-linear models in [53]. Other work related to the development of FIR filtering can be seen in [54–57]. However, a critical drawback of the FIR filtering is its batch-form estimation, which makes matrices and vectors acquire large dimensions. As an alternative solution, iterative Kalman-like algorithms for FIR calculation within the finite horizon were introduced in [32, 52, 58, 59]. These iterative estimations are of higher computational efficiency than the batch-form estimators. In view of this, Zhao et al. applied the sequential Bayesian theory to the most recent measurements, and revealed the relationship between the FIR and the IIR filters with a maximum likelihood FIR (ML-FIR) estimator [58]. The proposed estimator has the advantage of being independent of the initial states and covariance within each finite horizon, and converges to the optimal Kalman estimation with the increase of the finite horizon length.

Compared with the enormous works done on the FIR filtering, the FIR smoothing problem remains open, although some related hints have been shown [49, 60, 61]. Hence, the motivation of this paper is to extend the theory of FIR filtering to that of the FIR smoothing. The main contributions of this work are to develop three smoothing algorithms with FIR structure. To achieve this objective, we exploit the iterative ML-FIR estimation to obtain estimates for each point inside the finite horizon. Then, we combine these estimates by employing the formulas for the optimal combination of two independent estimates, known as the forward-backward smoother equations. The proposed FIR smoothers have three important advantages over the traditional smoothers. First, the proposed methods are robust to uncertainties of modeling parameter and noise statistics. Second, the algorithms are independent of the initial states and covariance in each finite horizon because the initial estimates are obtained through the batch ML calculation. Finally, the FIR smoothers can have a faster and more accurate capture of the local dynamic changes in the data as they only consider the most recent measurements.

This chapter is organized as follows. In Section 5.2 we introduce and compare the FIR filtering structure to the IIR filtering structure. In Section 5.3, we describe the system model and formulate the problem. In Section 5.5, the ML-FIR estimator is presented and decomposed into a prior and posterior estimator to be employed recursively within the finite horizon in a forward-backward direction. In Section 5.6, we introduce the optimal smoothing combinations and three different types of FIR smoothers are developed. Finally, we apply the proposed FIR smoothing algorithms to the object tracking and image processing problems in Section 5.7. Conclusions are drawn in Section 5.8.

5.2 FIR filters

FIR filters are filters with finite impulse response. They are also known as non-recursive filters as they do not have the feedback (a recursive part of a filter), even though recursive algorithm can be used for FIR realization. Figure 5.1 depicts FIR and IRR general structures.

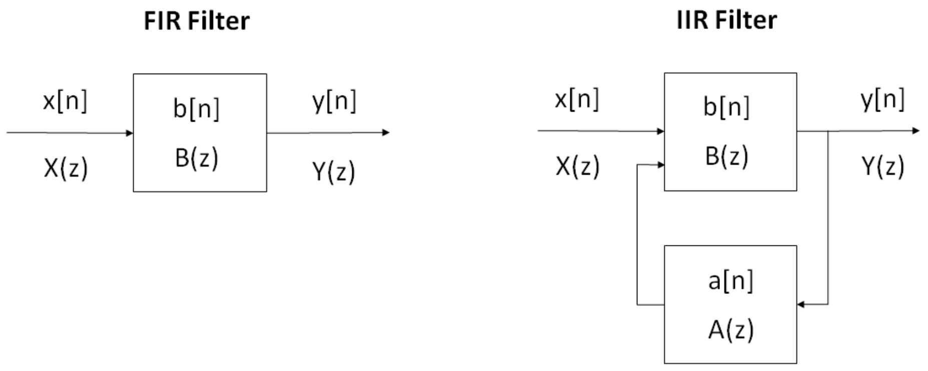


Figure 5.1: *FIR and IIR structures.*

A FIR filter takes an input signal $x[n]$, and produces an output signal $y[n]$ based on a weighted sum of the most recent input values $x[k-n]$. A typical form of the FIR equation is given as:

$$y[n] = \sum_{k=0}^{M} b_k x[k-n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_M x[n-M] \quad (5.1)$$

where b_k are called the FIR filter coefficients , and M is the order of the FIR filter. The set of FIR filter coefficients completely specifies an FIR filter. Different choices of the order and the coefficients leads to different kinds of filters (low-pass, high-pass and band-pass).

FIR filters have several desirable properties that make them attractive for a wide range of applications [43]. An exactly linear phase-response can be achieved with FIR filters, with the result that they can be used in the faithful reconstruction of signals without phase distortion. Consider, for instance, a filter that weights samples nearby more strongly than those that are far away. This performs a weaker smoothing, but it also introduces less distortion. Because of this, FIR filters are often more useful for random noise reduction.

In addition, FIR filters are inherently stable, and hence the question of stability does not arise either in the design or in the implementation of these filters. For example, images are very complex signals, which are very difficult to model accurately, and stability is thus a highly desirable feature for image processing filters.

The primary disadvantage of FIR filters is that they often require a much higher filter order than IIR filters to achieve a given level of performance. Other disadvantage of linear-phase FIR filters is the overall group delay in certain applications [29] .

5.2.1 Kalman Filter limitations

The KF works well but only under certain conditions [23], and the following are some of its drawbacks:

1- It is necessary to know the mean and correlation of the noise, $\mathbf{w}_n$ and $\mathbf{v}_n$ at each time instant.

2- It is necessary to know the covariances $\mathbf{Q}_n$ and $\mathbf{R}_n$ of the noise processes. The KF uses $\mathbf{Q}_n$ and $\mathbf{R}_n$ as design parameters, so if we do not know them, then it may be difficult to successfully use the KF.

3- The attractiveness of the KF lies in the fact that it is the one estimator that results in the smallest possible standard deviation of the estimation error. That is, the KF is the minimum variance estimator if the noise is Gaussian, and it is the linear minimum variance estimator if the noise is not Gaussian. If we desire to minimize a different cost function (such as the worst-case estimation error) then the KF may not accomplish our objectives.

4- It is necessary to know the system model matrices $\mathbf{F}_n$ and $\mathbf{H}_n$.

5.3 System Model and Problem Formulation

Consider a general discrete-time linear model represented in the state-space by:

$$\mathbf{x}_n = \mathbf{F}_n\mathbf{x}_{n-1} + \mathbf{G}_n\mathbf{w}_n \ , \tag{5.2}$$

$$\mathbf{y}_n = \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n \ , \tag{5.3}$$

where $\mathbf{x}_n \in \mathcal{R}^K$ is the state vector and $\mathbf{y}_n \in \mathcal{R}^L$ is the measurement vector. $\mathbf{F}_n \in \mathcal{R}^{K\times K}$ is the state transition matrix, $\mathbf{H}_n \in \mathcal{R}^{L\times K}$ is the measurement matrix, and $\mathbf{G}_n \in \mathcal{R}^{K\times P}$ is the noise matrix. $\mathbf{w}_n \in \mathcal{R}^P$ and $\mathbf{v}_n \in \mathcal{R}^L$ are the process and measurement noises that are zero-mean white Gaussian and mutually uncorrelated, i.e. $w_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ and $v_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_n)$.

The FIR filtering problem consists of estimating the current state $\mathbf{x}_n$ given the most recent measurements $\mathbf{y}_{m:n} \triangleq \{\mathbf{y}_m, \cdots, \mathbf{y}_n\}$, as Figure 5.2 illustrates. The interval that goes from m up to n is called the finite horizon.

Similarly, we can define the FIR smoothing estimation problem as follows. Given a linear Gaussian state-space model as in Equations 5.2 and 5.3, and letting $\mathbf{x}_n$ be the current state, the objective of the FIR smoothing is to estimate $\mathbf{x}_{n-q}$, given the most recent measurements $y_{m:n}$, where $1 < q \leq N$ is a positive integer. Figure 5.2 illustrates the FIR smoothing estimation problem.

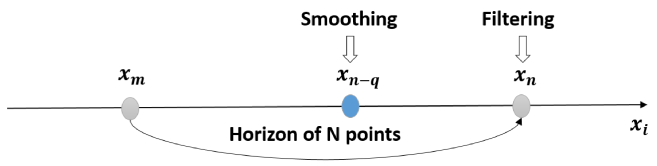


Figure 5.2: *FIR filtering and smoothing estimation problem.*

5.3.1 Extended State-Space model

In this section we introduce the extended state-space model that characterizes the batch form calculation of Equations 5.2 and 5.3 over the finite horizon $[m, n]$. Considering the forward-in-time solution and transforming all the state dynamics and measurements equations within the horizon $[m, n]$ with respect to x_{m-1}, we can find the batch form structure as:

$$\mathbf{X}_{n,m} = \mathbf{F}_{n,m}\mathbf{X}_{n,m} + \mathbf{G}_{n,m}\mathbf{W}_{n,m} \ , \tag{5.4}$$

$$\mathbf{Y}_{n,m} = \mathbf{H}_{n,m}\mathbf{X}_{n,m} + \mathbf{L}_{n,m}\mathbf{W}_{n,m} + \mathbf{V}_{n,m} \ , \tag{5.5}$$

where $\mathbf{X}_{n,m} \in \mathcal{R}^{NK\times NP}$,$\mathbf{W}_{n,m} \in \mathcal{R}^{NP\times 1}$, $\mathbf{Y}_{n,m} \in \mathcal{R}^{NL\times 1}$, $\mathbf{V}_{n,m} \in \mathcal{R}^{NL\times 1}$ are specified by, respectively,

$\mathbf{X}_{n,m} = [\mathbf{x}_n^T, \mathbf{x}_{n-1}^T, \cdots, \mathbf{x}_m^T]^T$,

$\mathbf{W}_{n,m} = [\mathbf{w}_n^T, \mathbf{w}_{n-1}^T, \cdots, \mathbf{w}_m^T]^T$,

$\mathbf{Y}_{n,m} = [\mathbf{y}_n^T, \mathbf{y}_{n-1}^T, \cdots, \mathbf{y}_m^T]^T$,

$\mathbf{V}_{n,m} = [\mathbf{v}_n^T, \mathbf{v}_{n-1}^T, \cdots, \mathbf{v}_m^T]^T$.

The extended system matrices $\mathbf{F}_{n,m} \in \mathcal{R}^{NK\times K}$, $\mathbf{G}_{n,m} \in \mathcal{R}^{NK\times NP}$, $\mathbf{H}_{n,m} \in \mathcal{R}^{NL\times K}$, and $\mathbf{L}_{n,m} \in \mathcal{R}^{NL\times NP}$ are

$$
\begin{aligned}
\mathbf{F}_{n,m} &= \left[(\mathcal{F}_n^m)^T, (\mathcal{F}_{n-1}^m)^T, \cdots, (\mathbf{F}_m^T)^T\right] \ , \\
\mathbf{G}_{n,m} &= \begin{bmatrix} \mathbf{G}_n & \mathbf{F}_n\mathbf{G}_{n-1} & \cdots & \mathcal{F}_n^{m+2}\mathbf{G}_{m+1} & \mathcal{F}_n^{m+1}\mathbf{G}_m \\ \mathbf{0} & \mathbf{G}_{n-1} & \cdots & \mathcal{F}_{n-1}^{m+2}\mathbf{G}_{m+1} & \mathcal{F}_{n-1}^{m+1}\mathbf{G}_m \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{G}_{m+1} & \mathbf{F}_{m+1}\mathbf{G}_m \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G}_m \end{bmatrix} \ , \\
\mathbf{H}_{n,m} &= \bar{\mathbf{H}}_{n,m}\mathbf{F}_{n,m} \ , \\
\mathbf{L}_{n,m} &= \bar{\mathbf{H}}_{n,m}\mathbf{G}_{n,m} \ ,
\end{aligned}
$$

where $\bar{\mathbf{H}}_{n,m} = \mathrm{diag}(\mathbf{H}_n, \mathbf{H}_{n-1}, \cdots, \mathbf{H}_m)$ is a diagonal matrix and $\mathcal{F}_i^j = \mathbf{F}_i\mathbf{F}_{i-1}\cdots\mathbf{F}_j$.

5.4 Optimal filtering on finite horizons

5.5 Maximum Likelihood FIR Estimation

A solution to the FIR filtering problem given a finite horizon of measurements $\mathbf{y}_{m:n}$, has been provided by the ML-FIR estimation $\hat{\mathbf{x}}_{n|N}^{ML}$, which is specified in [58] as

$$
\begin{aligned}
\hat{\mathbf{x}}_{n|N}^{ML} &= \mathbf{K}_{n,m}^{ML}\mathbf{Y}_{n,m} \\
&= (\tilde{\mathbf{H}}_{n,m}^T\mathbf{\Sigma}_{n,m}^{-1}\tilde{\mathbf{H}}_{n,m})^{-1}\tilde{\mathbf{H}}_{n,m}^T\mathbf{\Sigma}_{n,m}^{-1}\mathbf{Y}_{n,m} \ , \qquad (5.6)
\end{aligned}
$$

where $\mathbf{K}_{n,m}^{ML}$ is the ML-FIR gain, $\tilde{\mathbf{H}}_{n,m} \equiv \mathbf{H}_{n,m}(\mathcal{F}_n^m)^{-1}$, $\mathbf{\Sigma}_{n,m}$ is the finite horizon measurement variance,

$$
\begin{aligned}
\mathbf{\Sigma}_{n,m} &= Var[\mathbf{Y}_{n,m}|\mathbf{x}_n] \\
&= (\mathbf{L}_{n,m} - \tilde{\mathbf{H}}_{n,m}\bar{\mathbf{G}}_{n,m})\mathbf{Q}_{n,m}(\mathbf{L}_{n,m} - \tilde{\mathbf{H}}_{n,m}\bar{\mathbf{G}}_{n,m})^T + \mathbf{R}_{n,m} \ ,
\end{aligned}
$$

and $\bar{\mathbf{G}}_{n,m}$ corresponds to the first row of $\mathbf{G}_{n,m}$, which is specified as
$\bar{\mathbf{G}}_{n,m} = [\mathbf{G}_n \ \mathbf{F}_n\mathbf{G}_{n-1} \ \cdots \ \mathcal{F}_n^{m+2}\mathbf{G}_{m+1} \ \mathcal{F}_n^{m+1}\mathbf{G}_m]$.

As can be seen, Equation 5.6 has a batch form in calculation, where large dimension matrices have to be computed for each finite horizon. Therefore, the

ML-FIR estimator has also been presented in a Kalman-like iterative structure, in which the general estimation and covariance equations can be written as:

$$\hat{\mathbf{x}}_i = \mathbf{F}_i\hat{\mathbf{x}}_{i-1} + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)(\mathbf{y}_i - \mathbf{H}_i\mathbf{F}_i\hat{\mathbf{x}}_{i-1}) \ , \tag{5.7}$$

$$\begin{aligned}\mathbf{P}_i = {} & (\mathbf{I} - (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{H}_i)(\mathbf{F}_i\mathbf{P}_{i-1}\mathbf{F}_i^T + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T)(\mathbf{I} - (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{H}_i)^T \\ & + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{R}_i(\mathbf{K}_i + \tilde{\mathbf{K}}_i)^T. \end{aligned} \tag{5.8}$$

In Equation 5.7, the first term on the right-hand side, $\mathbf{F}_i\hat{x}_{i-1}$, is used to predict the estimate from $i-1$ to i, while the second term, $(\mathbf{K}_i+\tilde{\mathbf{K}}_i)(\mathbf{y}_i-\mathbf{H}_i\mathbf{F}_i\hat{\mathbf{x}}_{i-1})$ updates the predicted estimate at i based on the measurement $\mathbf{y}_i$. $(\mathbf{K}_i+\tilde{\mathbf{K}}_i)$ is the correction gain, where $\mathbf{K}_i$ is equivalent to the Kalman Gain, and $\tilde{\mathbf{K}}_i$ works as an additive correction gain to $\mathbf{K}_i$. Similarly, in Equation 5.8 the term $\mathbf{F}_i\mathbf{P}_{i-1}\mathbf{F}_i^T+\mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T$ is the prediction error variance, and $(\mathbf{I}-(\mathbf{K}_i+\tilde{\mathbf{K}}_i)\mathbf{H}_i)$ indicates the posterior estimation error variance. The complete description of iterative ML-FIR estimator can be found in Appendix B.1.

Base on these, Equations 5.7 and 5.8 can be decomposed into prior (prediction) and posterior (update) forms 5.9-5.12. In order to simplify the notations, we use the superscript "$-$" to indicate a prior estimation, and superscript "$+$" to indicate a posterior estimation.

1) *Prior estimation:*

$$\hat{\mathbf{x}}_i^- = \mathbf{F}_i\hat{\mathbf{x}}_{i-1}^+ \ , \tag{5.9}$$

$$\mathbf{P}_i^- = \mathbf{F}_i\mathbf{P}_{i-1}^+\mathbf{F}_i^T + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ . \tag{5.10}$$

2) *Posterior estimation:*

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i^-) \ , \tag{5.11}$$

$$\begin{aligned}\mathbf{P}_i^+ = {} & (\mathbf{I} - (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{H}_i)(\mathbf{P}_i^-)(\mathbf{I} - (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{H}_i)^T \\ & + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{R}_i(\mathbf{K}_i + \tilde{\mathbf{K}}_i)^T \ . \end{aligned} \tag{5.12}$$

Note that Equations 5.7 and 5.8 can be used either in a forward or backward direction inside each finite horizon $[m,n]$. That is, they can be used to estimate

$\hat{\mathbf{x}}_n$ starting from $\mathbf{x}_m$ in a forward direction; or to estimate $\hat{\mathbf{x}}_m$ starting from $\mathbf{x}_n$ in a backward direction. The slight modification is that for backward estimation the system model has to be inverted, that is, we should use $\mathbf{F}^{-1}$ instead of $\mathbf{F}$, by assuming $\mathbf{F}$ is invertible. In the same way, Equations 5.9-5.12 can be employed either in a forward or backward direction inside each finite horizon. The complete description of the iterative ML-FIR estimator can be found in Appendix B.

The forward and backward estimates obtained by Equations 5.9-5.12 are not optimal since they are calculated from the iterative ML-FIR estimator, which approaches to the optimal estimation only when the finite horizon $[m; n]$ extends to the full dataset [58]. Therefore, the length of the finite horizon plays an important role in the estimation performance.

5.6 Proposed FIR Smoothers

In this section, we derive three different FIR smoothing algorithms based on the combinations of Equations 5.9-5.12. Figures 5.3, 5.4 and 5.5 illustrates each of these FIR smoothing algorithms respectively. The first two smoothers (Figures 5.3 and 5.4: fixed-lag FIR smoothers) have more benefits for on-line applications where an estimate with a specific lag q is required. In this situation, we want to obtain an estimate of the state at $n - q$ given measurements from m up to n, where the horizon $[m, n]$ continually changes as we obtain new measurements, but the lag q is constant. The first algorithm, in Figure 5.3, considers a combination of estimates within the same finite horizon, while the second one, in Figure 5.4, combines estimates from different finite horizons. Finally, the third algorithm (fixed-interval FIR smoothing in Figure 5.5) is intended to be used for off-line application. In this situation, we have a fixed interval of measurements $(y_1, y_2, \cdots, y_T)$, and we want to obtain the state estimates for all the points in that interval.

To derive the FIR-type smoothers, we consider the forward-backward combination of estimates here, which reduces the complexity of the smoothing procedure [23]. However, other types of combinations can also be performed inside the horizon.

5.6.1 Optimal smoothing

Forward-backward smoothing

The well known formulas for the optimal combination of two independent estimates, $\mathbf{x}_1$ (covariance $\mathbf{P}_1$) and $\mathbf{x}_2$ (covariance $\mathbf{P}_2$), of $\mathbf{x}$ are [40],

$$\hat{\mathbf{x}} = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1}(\mathbf{P}_1^{-1}\hat{\mathbf{x}}_1 + \mathbf{P}_2^{-1}\hat{\mathbf{x}}_2) \ , \tag{5.13}$$

$$\mathbf{P} = cov(\hat{\mathbf{x}} - \mathbf{x}) = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1} \ , \tag{5.14}$$

where the statistics of noise are assumed to be Gaussian with zero mean. These estimates represent the minimum variance and maximum likelihood estimate of $\hat{\mathbf{x}}$. If we assume that $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are the estimates for a given point i obtained from two different optimal filters, one of which runs from the beginning of the data interval forward to i, and the other of which runs backward to the point i from the end of the data interval, then $\hat{\mathbf{x}}$ is the optimal smoother estimate for the point i.

Hence, applying some mathematical rearrangement on Equations 5.13 and (5.14 we can achieve the forward-backward smoothing Equations 5.15-(5.17. The forward-backward approach for smoothing combines two estimates to obtain the smoothed estimate. The subscript 'f' refers to forward estimates while the 'b' corresponds to backward estimates.

$$\mathbf{K}_f = \mathbf{P}_b^-(\mathbf{P}_f^+ + \mathbf{P}_b^-)^{-1} \ , \tag{5.15}$$

$$\hat{\mathbf{x}} = \mathbf{K}_f\hat{\mathbf{x}}_f^+ + (\mathbf{I} - \mathbf{K}_f)\hat{\mathbf{x}}_b^- \ , \tag{5.16}$$

$$\mathbf{P} = [(\mathbf{P}_f^+)^{-1} + (\mathbf{P}_b^-)^{-1}]^{-1} \ . \tag{5.17}$$

RTS smoothing

The RTS smoother is an efficient two-pass algorithm for fixed interval smoothing [41], which has two steps. A first step (forward path), where a filter runs forward to obtain estimates for each point in the fixed interval. And a second step (backward path), where we compute the smoothed state estimates $\hat{\mathbf{x}}_i$ and covariances P_i. In the forward path, the filtered apriori and aposteriori estimates $\hat{\mathbf{x}}^-_{f,i}$, $\hat{\mathbf{x}}^+_{f,i}$, and covariances $\mathbf{P}^-_{f,i}$, $\mathbf{P}^+_{f,i}$ are saved for use in the backward path, which start at the last point in the fixed interval and proceed backward considering the following recursive equations:

$$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}^+_{f,i} + \mathbf{K}_i(\hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}^-_{f,i+1}) \ , \tag{5.18}$$

$$\mathbf{P}_i = \mathbf{P}^+_{f,i} - \mathbf{K}_i(\mathbf{P}^-_{f,i+1} - \mathbf{P}_{i+1})\mathbf{K}^T_i \ , \tag{5.19}$$

where $\mathbf{K}_i$ is the RTS smoothing gain,

$$\mathbf{K}_i = \mathbf{P}^+_i \mathbf{F}^T_i (\mathbf{P}^-_{i+1})^{-1} \ . \tag{5.20}$$

5.6.2 Fixed-lag FIR smoothing (FIR-MLS1): Combination of estimates within the horizon

In fixed-lag smoothing, the objective is to obtain an estimate of the state at $\mathbf{x}_{n-q}$ given the measurements up to and including index n, where the index n continually changes as we obtain new measurements, but the lag q is a constant. Figure 5.3 illustrates the fixed-lag smoothing estimation $\hat{\mathbf{x}}_{n-q}$ in a finite horizon of measurements $\mathbf{y}_{m:n}$.

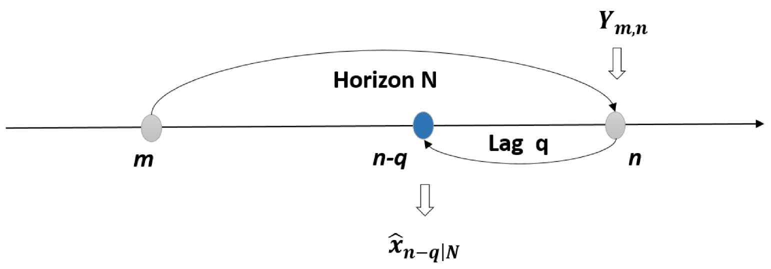


Figure 5.3: *Fixed-lag FIR smoother: combination of forward and backward estimates in the same finite horizon.*

The estimate $\hat{\mathbf{x}}_{n-q}$ inside the finite horizon of measurements $\mathbf{y}_{m:n}$ can be considered as a special case of fixed-interval smoothing in the finite horizon. Suppose that we want to estimate a state $\mathbf{x}_{n-q}$ within the finite horizon based on the measurements $\mathbf{y}_{m:n}$. Considering the forward-backward approach we can obtain two estimates for $\mathbf{x}_{n-q}$. The first estimate, $\hat{\mathbf{x}}_f$ is based on a forward filter that goes from $i = m$ to $i = n - q$. The second estimate $\mathbf{x}_b$, is based on a backward filter that runs backward from $i = n$ to $i = n - q$. In this sense, we can employ Equations 5.15, 5.16 and 5.17 to combine the ML-FIR estimation in a forward-backward direction for each finite horizon, and finally obtain an smoothed estimation of any point within the finite horizon. Thus, we are able to describe an iterative FIR smoother algorithm based on a forward-backward estimates combination within the finite horizon, which is shown in Algorithm 3. A detailed description of Algorithm 3 can be found in Appendix B.2.

```
Data: $\mathbf{y}_{m:n}$
begin
    for $i = m : 1 : n$ do
        Initialization: $\hat{\mathbf{x}}_\alpha$, $\mathbf{P}_\alpha$ ;
        Execute forward ML-FIR estimation: $\hat{\mathbf{x}}^-_{f,i}$, $\mathbf{P}^-_{f,i}$, $\hat{\mathbf{x}}^+_{f,i}$, $\mathbf{P}^+_{f,i}$ ;
    end
    for $i = n : -1 : n - q$ do
        Initialization: $\hat{\mathbf{x}}^+_{b,n} = \hat{\mathbf{x}}^+_{f,n}$, $\mathbf{P}^+_{b,n} = \mathbf{P}^+_{f,n}$ ;
        Execute backward ML-FIR estimation: $\hat{\mathbf{x}}^-_{b,i}$,$\mathbf{P}^-_{b,i}$,$\hat{\mathbf{x}}^+_{b,i}$,$\mathbf{P}^+_{b,i}$ ;
        Combine forward and backward estimations using (5.15), (5.16) and
        (5.17): $\hat{\mathbf{x}}_i$, $\mathbf{P}_i$ ;
    end
end
Result: $\hat{\mathbf{x}}_{n-q}$
```

Algorithm 3: Fixed-lag FIR smoother within the horizon

Through Algorithm 1 we can obtain the smoothed estimates, $\hat{\mathbf{x}}_i$, for each point within the finite horizon $[m, n]$. However, if we are interested in a particular smoothed estimation within the finite horizon with a given lag q, we only need to perform the forward-backward combination until that estimation point of interest $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{n-q}$.

5.6.3 Fixed-lag FIR smoothing (FIR-MLS2): combination of estimates from different finite horizons

In this section, we derive an extension of Algorithm 1, where we analyze a combination of ML-FIR estimates within the same finite horizon. In fact, we can consider a combination of ML-FIR estimates coming from different finite horizons. Figure 5.4

depicts the smoothing combination of estimates from different finite horizons. From

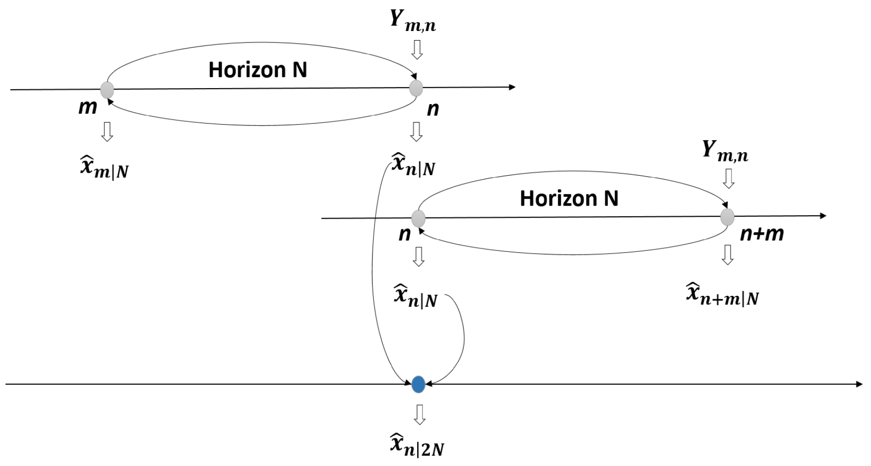


Figure 5.4: *Fixed-lag FIR smoother: combination of forward and backward estimates coming from different finite horizons.*

[58], we know that the most accurate estimation in a finite horizon $[m, n]$ is obtained at the last point $\hat{\mathbf{x}}_n$. Following a similar argument, the most accurate estimation in a backward ML-FIR estimation is given at the last point, which also corresponds to the beginning of the finite horizon $\hat{\mathbf{x}}_m$. Therefore, at any point $\mathbf{x}_i$ we propose a combination of a forward ML-FIR estimation $\mathbf{x}_{f,i}$ from one finite horizon and a backward ML-FIR estimation $\mathbf{x}_{b,i}$ from the other finite horizon.

Then, we are able to describe an iterative FIR smoother algorithm based on a forward-backward estimates combination from different finite horizons, which is shown in Algorithm 4. The detailed description of Algorithm 4 is in Appendix B.3.

```
Data: $\mathbf{y}_{m:n}$
begin
    for $i = 1 : n$ do
        for $i = m : 1 : n$ do
            Initialization: $\hat{\mathbf{x}}_\alpha$, $\mathbf{P}_\alpha$ ;
            Execute forward ML-FIR estimation ;
            Save estimates at the end of the finite horizon: $\hat{\mathbf{x}}^-_{f,n}$, $\mathbf{P}^-_{f,n}$, $\hat{\mathbf{x}}^+_{f,n}$, $\mathbf{P}^+_{f,n}$ ;
        end
        for $i = n : -1 : m$ do
            Initialization: $\hat{\mathbf{x}}^+_{b,n} = \hat{\mathbf{x}}^+_{f,n}$, $\mathbf{P}^+_{b,n} = \mathbf{P}^+_{f,n}$ ;
            Execute backward ML-FIR estimation ;
            Save estimates at the beginning of the finite horizon: $\hat{\mathbf{x}}^-_{b,m}$, $\mathbf{P}^-_{b,m}$,
            $\hat{\mathbf{x}}^+_{b,m}$, $\mathbf{P}^+_{b,m}$ ;
        end
        Combine forward and backward estimates from different finite horizons
        through (5.15), (5.16) and (5.17): $\hat{\mathbf{x}}_i$, $\mathbf{P}_i$
    end
end
Result: $\hat{\mathbf{x}}_{n-m}$
```

Algorithm 4: Fixed-lag FIR smoother (different finite horizons)

The best combination of estimates coming from different finite horizon is achieved when the largest number of measurement can be covered by the horizons. This occurs when the last point of the horizon, n, coincides with the first point of the next finite horizon, m. In this case, if n is the last point of the finite horizon we can use the forward ML-FIR estimator to obtain $\hat{\mathbf{x}}_{f,n}$. Then, if m is the first point of the next horizon we can employ the backward ML-FIR estimator to achieve $\hat{\mathbf{x}}_{b,m}$. Finally, the combination of those two estimates $\hat{\mathbf{x}}_{f,n}$ and $\hat{\mathbf{x}}_{b,m}$ is the best combination from different finite horizons.

In a real time application, if $[m, n]$ is the current finite horizon, the best fixed-lag FIR smoothing estimate $\hat{\mathbf{x}}_{n-q}$ combining different finite horizon estimates is achieved at the beginning of the current finite horizon $\hat{\mathbf{x}}_{n-q} = \hat{\mathbf{x}}_m = \mathit{combination}(\hat{\mathbf{x}}_{f,m} + \hat{\mathbf{x}}_{b,m})$.

5.6.4 Fixed-interval FIR smoothing (FIR-MLS3)

In this section, we propose a fixed-interval smoothing employing ML-FIR estimations from different finite horizons. In a fixed-interval smoother, we seek an estimate of state at some of the interior points of the fixed-interval. During the smoothing process, we do not obtain any new measurements.

If we consider a fixed-interval of measurements $\mathbf{y}_{1,\cdots,T}$ (where T is the final point in the fixed interval), we can use the ML-FIR estimation in a forward direction to obtain estimates $\hat{\mathbf{x}}_i$ for each point. Finally, we combine the ML-FIR estimates in a backward direction to obtain the fixed-interval smoothing estimation. Figure 5.5 illustrates the fixed-interval FIR smoothing approach. In this case, for the backward

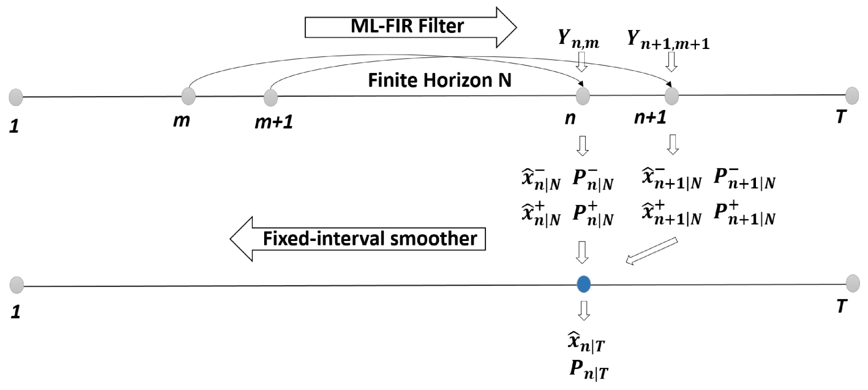

ML-FIR Filter
$Y_{n,m}$
$Y_{n+1,m+1}$
Finite Horizon N
1
m
m+1
n
n+1
T
$\hat{x}^-_{n|N}$ $P^-_{n|N}$
$\hat{x}^-_{n+1|N}$ $P^-_{n+1|N}$
$\hat{x}^+_{n|N}$ $P^+_{n|N}$
$\hat{x}^+_{n+1|N}$ $P^+_{n+1|N}$
Fixed-interval smoother
1
T
$\hat{x}_{n|T}$
$P_{n|T}$

Figure 5.5: *Fixed-interval FIR smoothing. Use a forward ML-FIR estimation to obtain estimates at each point, then execute a backward smoothing combination.*

smoothing combination of estimates, we use the RTS smoother equations described in Section 5.6.1 in order to simplify the calculations [23]. Therefore, the smoothing procedure has two steps. In a first step, the ML-FIR filter runs forward to obtain and keep the prior and posterior estimates, $\hat{\mathbf{x}}_i$, for each point in the fixed-interval. In a second step, Equations 5.18, 5.19, and 5.20 go backward to achieve the smoothed estimates. Hence, we are able to describe an iterative fixed-interval FIR smoother algorithm based on the RTS smoother equations, which is shown in Algorithm 5. The detailed description of Algorithm 5 can be read in Appendix B.4.

```
Data: $\mathbf{y}_{1:T}$
begin
    for $i = 1:1:T$ do
        Input: $\mathbf{y}_{m:n}$ ;
        for $i = m:1:n$ do
            Initialization ;
            Execute forward ML-FIR estimation ;
        end
    end
    Output: $\hat{\mathbf{x}}^{-}_{i,m:n}$; $\mathbf{P}^{-}_{i,m:n}$; $\hat{\mathbf{x}}^{+}_{i,m:n}$; $\mathbf{P}^{+}_{i,m:n}$ ;
    for $i = T:-1:1$ do
        Initialization: $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}^{+}_{f,T}$, $\mathbf{P}_T = \mathbf{P}^{+}_{f,T}$ ;
        Execute backward RTS smoother through (5.18), (5.19), and (5.20):
        $\mathbf{K}_i$; $\hat{\mathbf{x}}_i$; $\mathbf{P}_i$ ;
    end
end
Result: $\hat{\mathbf{x}}_{i|T}$
```

Algorithm 5: Fixed-interval FIR smoother

5.7 Simulations and Results

In this section, we describe two different applications of FIR smoothing. The first case is a tracking object simulation with approximate modeling parameters. While the second is an image restoration problem. In both cases, we compare FIR estimations to traditional IIR estimations.

5.7.1 Experimental Validation

The quantitative analysis for experimental validation is carried out by using two parameters widely use in signal processing, Signal-to-Noise Ratio (SNR) and Root

Mean Square Error (RMSE), which are defined in 2 dimensions as

$$SNR_{dB} = 10 * log_{10}\left(\frac{\sum_{i=1}^{r}\sum_{j=1}^{c}[\mathbf{x}(i,j)]^2}{\sum_{i=1}^{r}\sum_{j=1}^{c}[\hat{\mathbf{x}}(i,j) - \mathbf{x}(i,j))]^2}\right) \ , \tag{5.21}$$

$$RMSE = \sqrt{\frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}[\hat{\mathbf{x}}(i,j) - \mathbf{x}(i,j)]^2} \ , \tag{5.22}$$

where r corresponds to one of the signal dimension length and c to the other dimension length. In the case of 1 dimensional signal analysis we only need to consider the dimension r in Equations 5.21 and 5.22.

5.7.2 Object tracking simulation

In this section we simulate a moving object example. Data is generated with additive Gaussian noise, and the system is modeled with approximate parameters. The objective is to estimate the true states.

Data Generation:

In the simulation, the true states are generated by Equation 5.23 . The measurements, $\mathbf{y}(t_n)$, are generated from Equation 5.24 with sampling time $\mathbf{t}_n = 0.04 \times \pi \times n$, and contaminated with additive Gaussian noise $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \sigma^2 = 0.25)$. A total of 200 data points ($n = 1, 2, \cdots, 200$) are considered in the analysis [62].

$$\mathbf{x}(t) = [-cos(t) - sin(t)]^T \ , \tag{5.23}$$

$$\mathbf{y}(t_n) = \mathbf{x}(t_n) + \mathbf{v}_n \ . \tag{5.24}$$

Modelling:

We model the two components of the state in Equation 5.23 as the first and second integrals of white noise. Therefore, the dynamic and measurement equations of the system can be specified by Equations 5.2 and 5.3 with,

$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} \mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{Q} = \begin{bmatrix} T & T^2/2 \\ T^2/2 & T^3/2 \end{bmatrix}$

$\mathbf{H} = \begin{bmatrix} 0 & 1 \end{bmatrix}$ $\mathbf{R} = \sigma^2$,

where the sample time is $T = 0.04 \times \pi$ and $\sigma^2 = 0.25$.

Simulations:

Figure 5.6 compares the classical RTS smoother to the three FIR smoothers developed in this paper. It is observed that FIR smoothers estimate the true states with more accuracy than RTS smoother. As ML-FIR structure is more robust to noise statistics and modeling parameters, we expect to have better smoothing performance. The robustness can be observed particularly in the regions with more dynamics (signal bends in Figure 5.6).

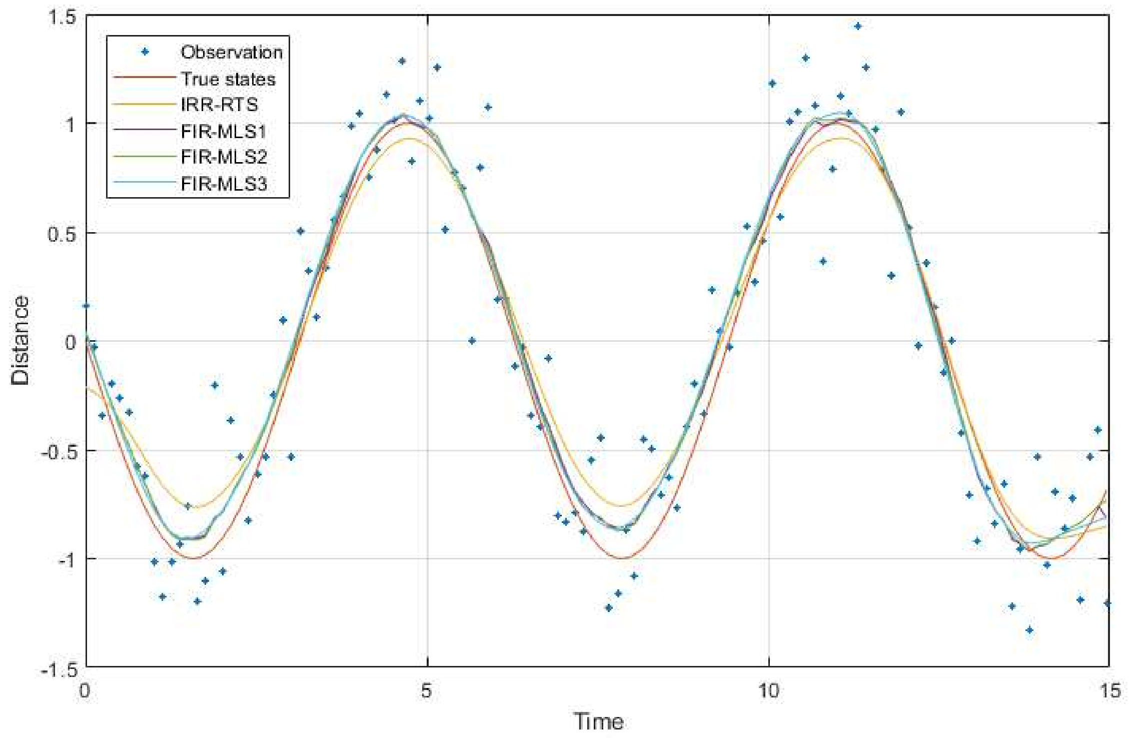


Figure 5.6: *Smoothing performance of FIR smoothers to a moving target with approximate modeling parameters. IRR-RTS: RTS Fixed-interval smoother. FIR-MLS1 (Algorithm 1): Fixed-lag smoother within the same finite horizon [N=25 q=10]; FIR-MLS2 (Algorithm 2): Fixed-lag smoother from different finite horizons [N=25]; FIR-MLS3 (Algorithm 3): Fixed-interval smoother [N=25]*

Table 5.1 illustrates the SNR and RMSE performance of each filter. The IIR-KF and ML-FIR were also included in the table to compare their performance to FIR smoothers. Essentially, we can observe that FIR filters provide a more accurate estimation than traditional IIR filters when the system has approximate modeling parameters.

Table 5.1: *Filters Performance and Experimental Validation*

Filter Structure	**SNR(dB)**	**RMSE**
IRR-KF	12.35	0.1726
IRR-RTS	16.18	0.1111
FIR-ML	13.30	0.1572
FIR-MLS1 Fixed-lag smoother(within horizon)	16.59	0.0960
FIR-MLS2 Fixed-lag smoother(different horizons)	17.59	0.0960
FIR-MLS3 Fixed-interval smoother	17.13	0.1000

Figure 5.7 illustrates the RMSE as a function of the finite estimation horizon length N. We can see that the RMSE decreases when we increase the finite horizon length. The reason is that ML-FIR achieves better estimates when more measurements are included in the horizon. As a reference, the RMSE for the IIR filters are included in the same figure.

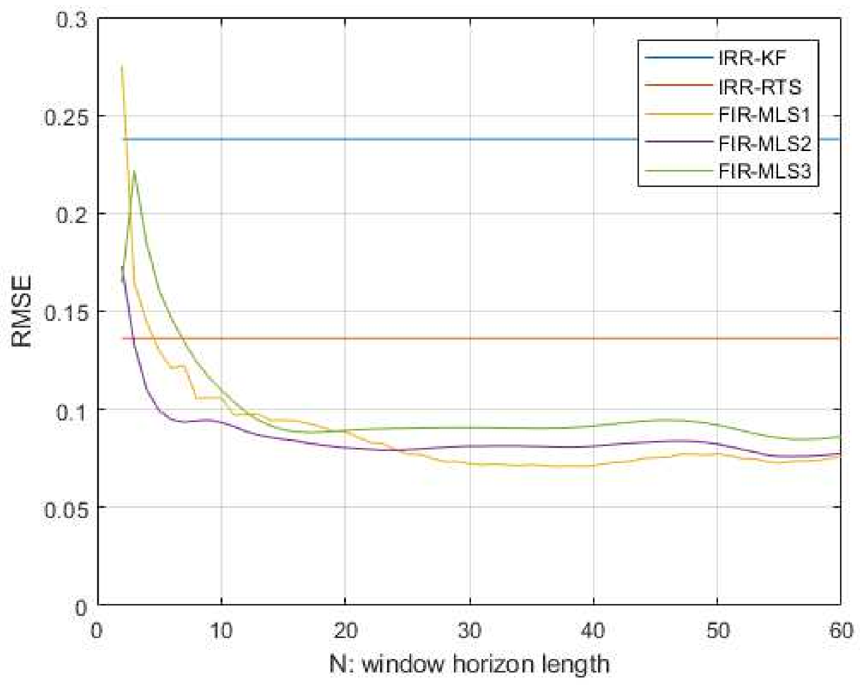


Figure 5.7: *Effect of finite estimation horizon length (N). IRR-KF: Kalman Filter. IRR-RTS: RTS Fixed-interval smoother. FIR-MLS1 (Algorithm 1): Fixed-lag smoother within the same finite horizon, FIR-MLS2 (Algorithm 2): Fixed-lag smoother from different finite horizons, FIR-MLS3 (Algorithm 3): Fixed-interval smoother*

5.7.3 Image Processing

In this section, we compare IIR and FIR smoothers performance on an image restoration problem. A benchmark image is selected and contaminated with noise. Then, it is restored through FIR smoothers developed in this paper, and the results compared with traditional IIR smoothers

Image benchmark:

The benchmark image 'Coins' is selected. Figure 5.8 displays the original image on the left, and the contaminated image with Gaussian noise ($\sigma^2 = 1$) on the right.

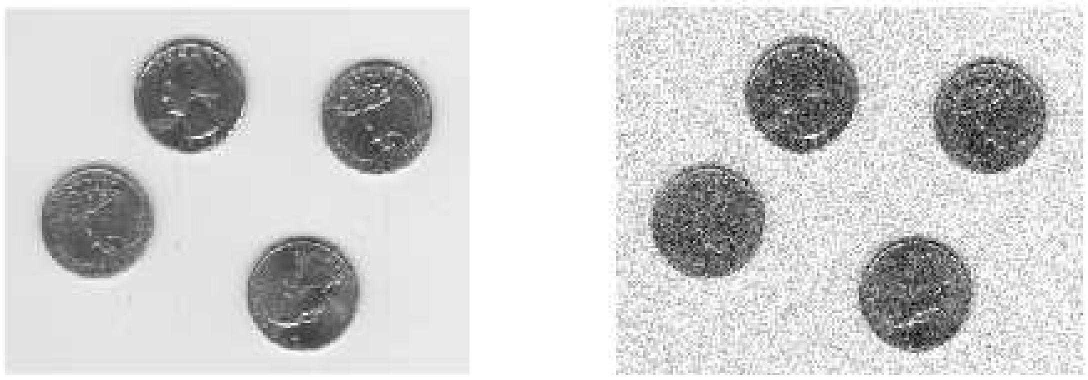


Figure 5.8: *Benchmark image: 'Coins'. (a) Original image, (b) Image contaminated with Gaussian noise, σ^2=1.*

Image Modeling:

A two-dimensional image is often represented as matrix $\mathbf{M} = x_{i,j}$ with r_i rows and c_i columns. The filtering procedure is then applied twice, first to each column and then to each row, or vice versa [63].

A 1-pixel model is considered for the restoration. Thus, dynamic and measurement equation of the image model can be specified by Equations 5.2 and 5.3 with

$\mathbf{F} = \begin{bmatrix}1\end{bmatrix}$ $\mathbf{G} = \begin{bmatrix}1\end{bmatrix}$ $\mathbf{Q} = \begin{bmatrix}\sigma_Q^2\end{bmatrix}$

$\mathbf{H} = \begin{bmatrix}1\end{bmatrix}$ $\mathbf{R} = \begin{bmatrix}\sigma_R^2\end{bmatrix}$

Image Restoration:

Two different scenarios were simulated in order to illustrate the robustness of FIR against IIR. In each scenario, we consider different values of **Q** and **R** (Scenario 1: **Q**=0.1 and **R**=1. Scenario 2: **Q**=0.01 and **R**=10). A finite horizon of N=20 is considered in all FIR smoothers; and for Algorithm 1 (fixed-lag smoother within the same finite horizon) a lag of q=10 is used. Figure 5.9 and 5.10 illustrates both scenarios with different noise model parameters.

When noise models parameters (**Q** and **R**) are known, both structures (IIR and

FIR smoothers) give a similar estimation (Scenario 1 in Figure 5.9). However, when $\mathbf{Q}$ and $\mathbf{R}$ are incorrect, FIR estimations are more robust than IIR estimations (Scenario 2 in Figure 5.10). In Figure 5.10, we can observe that IIR restoration is blur which makes it very hard to find the coins in the image. However, using the FIR restoration the coins can be distinguished clearly.

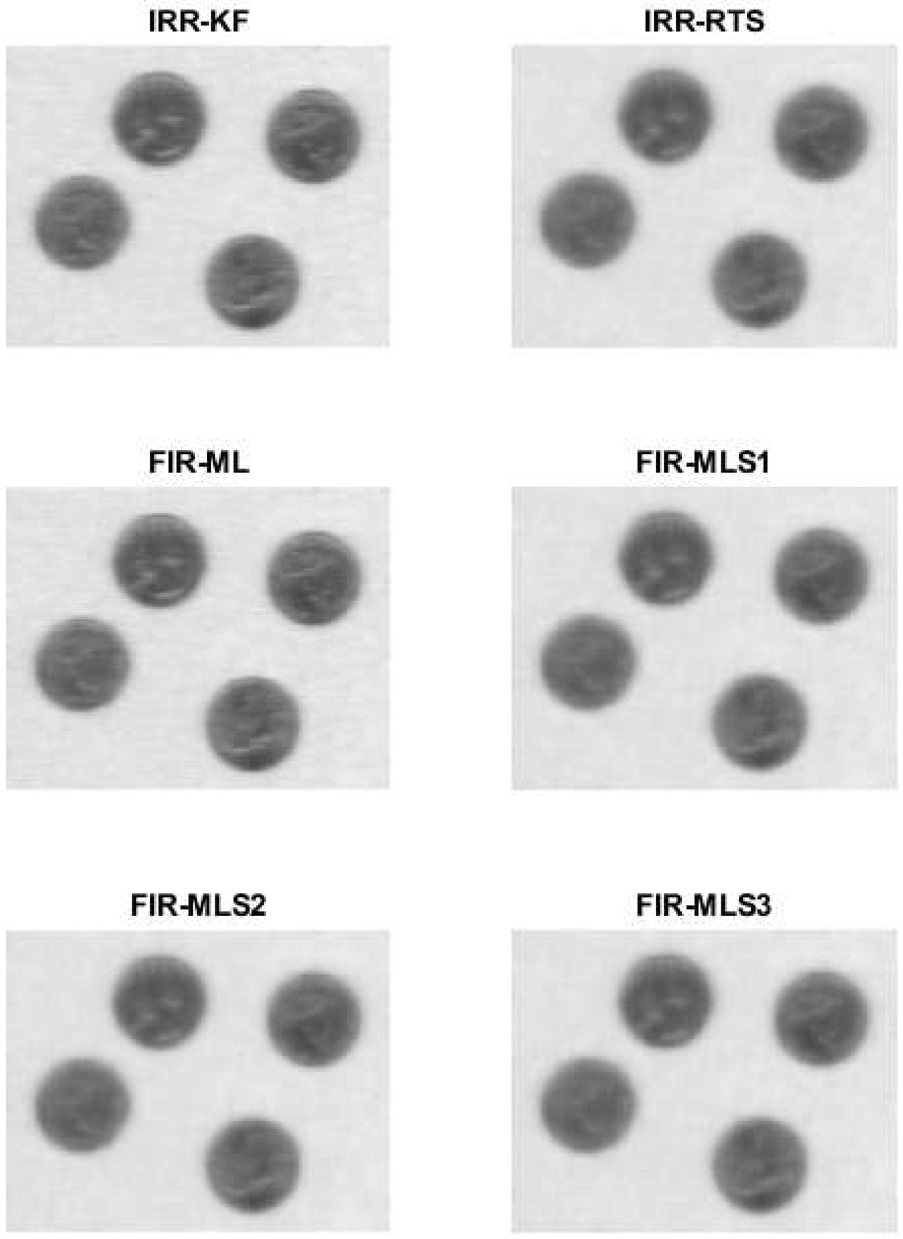


Figure 5.9: *Scenario 1: IIR and FIR restoration considering* $\mathbf{Q} = 0.1$ *and* $\mathbf{R} = 1$. *(a) Kalman Filter, (b) RTS Smoother, (c) Maximum Likelihood FIR, (d) FIR-MLS1 (Algorithm 1): Fixed-lag smoother within the same finite horizon, (e) FIR-MLS2 (Algorithm 2): Fixed-lag smoother from different finite horizons, (d) FIR-MLS3 (Algorithm 3): Fixed-interval smoother.*

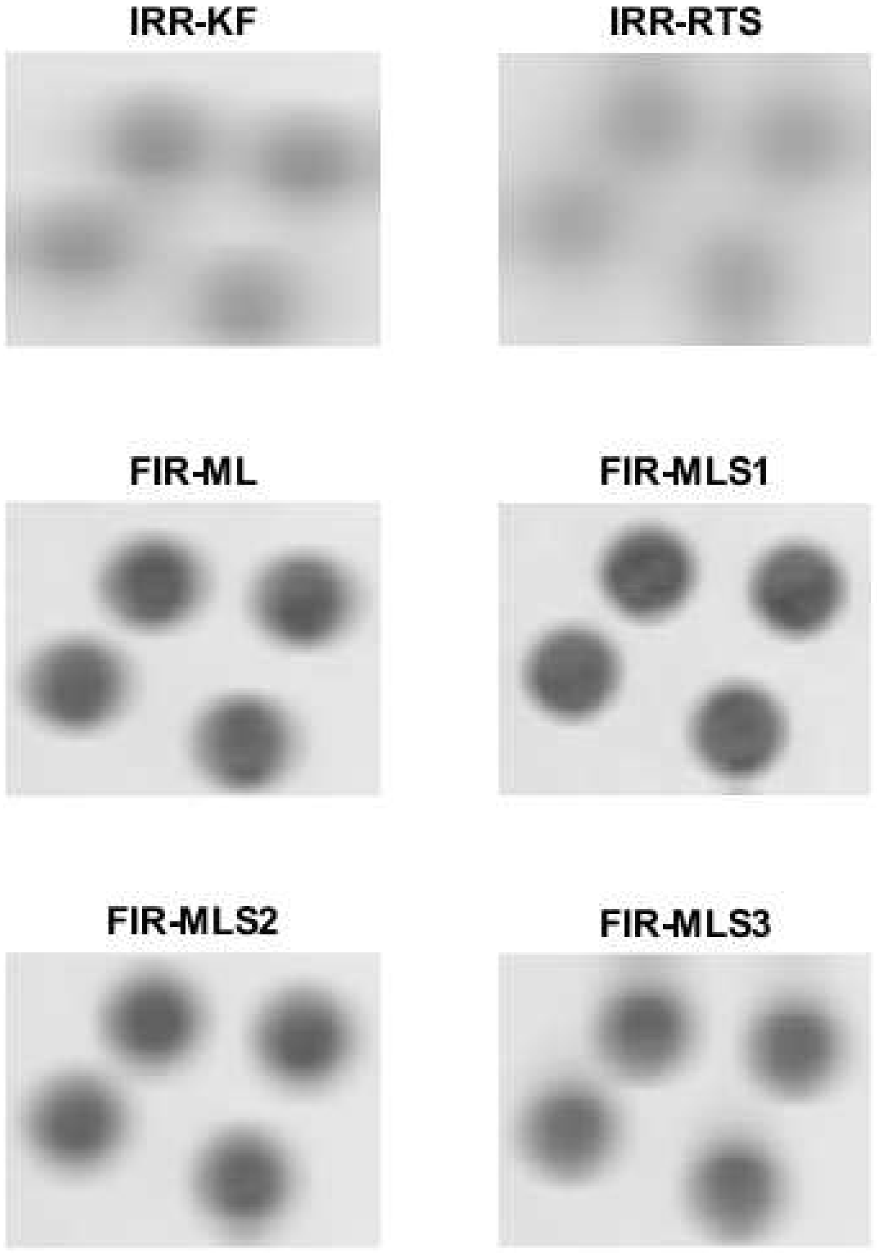


Figure 5.10: *Scenario 2: IIR and FIR restoration considering* $\mathbf{Q} = 0.01$ *and* $\mathbf{R} = 10$. *(a) Kalman Filter, (b) RTS Smoother, (c) Maximum Likelihood FIR, (d) Algorithm 1: FIR Maximum likelihood (within finite horizon), (e) FIR-MLS2 (Algorithm 2): Fixed-lag smoother from different finite horizons, (d) FIR-MLS3 (Algorithm 3): Fixed-interval smoother.*

Table 5.2 compares the RMSE for both Scenarios and confirms numerically what was observed in Figures 5.9 and 5.10. In Scenario 1, the RMSE in all the methods is around the same value, being a little bit smaller for FIR filters. However, in Scenario 2 the RMSE is considerably smaller in all FIR estimations.

Table 5.2: *Filters Performance and Experimental Validation*

Filter Structure	**Scenario 1 (RMSE)**	**Scenario 2 (RMSE)**
IRR-KF	0.0624	0.1837
IRR-RTS	0.0572	0.1743
FIR-ML	0.0624	0.1189
FIR-MLS1 Fixed-lag smoother	0.0561	0.0724
FIR-MLS2 Fixed-lag smoother	0.0569	0.0970
FIR-MLS3 Fixed-interval smoother	0.0571	0.1094

Figure 5.11 compares the IIR and FIR estimation of the same row data-set for different $\mathbf{Q}$ and $\mathbf{R}$ values. We can appreciate how $\mathbf{Q}$ and $\mathbf{R}$ values affect gradually the IIR estimation. However, the FIR smoother always provides a similar estimation no matter which $\mathbf{Q}$ and $\mathbf{R}$ values are considered.

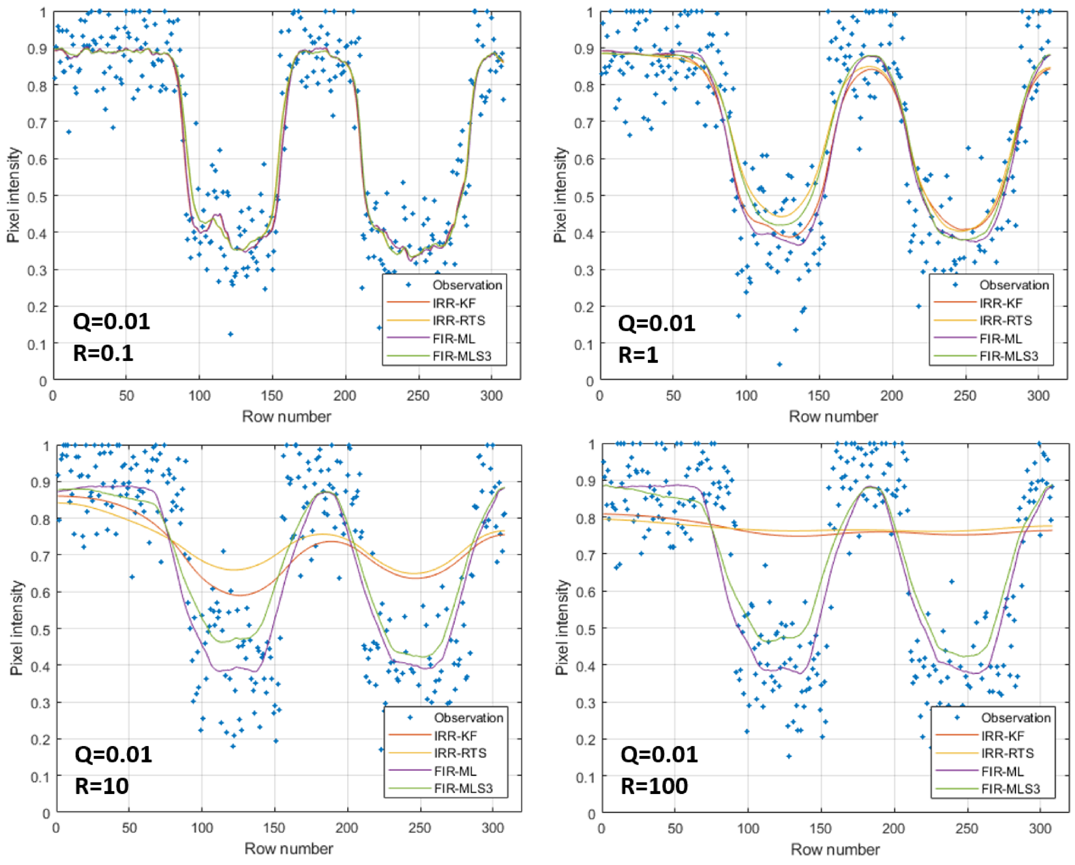


Figure 5.11: *IIR and FIR restoration with different noise model parameters* **Q** *and* **R**. *The trends correspond to one image row data-set. IRR-KF: Kalman Filter; IRR-RTS: RTS Smoother; ML-FIR Filter [N=20]; FIR-MLS3 (Algorithm 3): Fixed-interval smoother [N=20].*

In order to have a better understanding of FIR estimations under uncertain noise statistics, we introduce two different coefficient ρ_Q and ρ_R to vary $\mathbf{Q}$ and $\mathbf{R}$ as $\rho_Q * \mathbf{Q}$ and $\rho_R * \mathbf{R}$. Thus, the RMSE is obtained as a function of ρ_Q and ρ_R, and the results shown in Figure 5.12. As we can observe, FIR smoother has a better performance against the errors of noise statistics than the IRR smoother. The performance is even better when there is a large noise measurement error $\mathbf{Q}$.

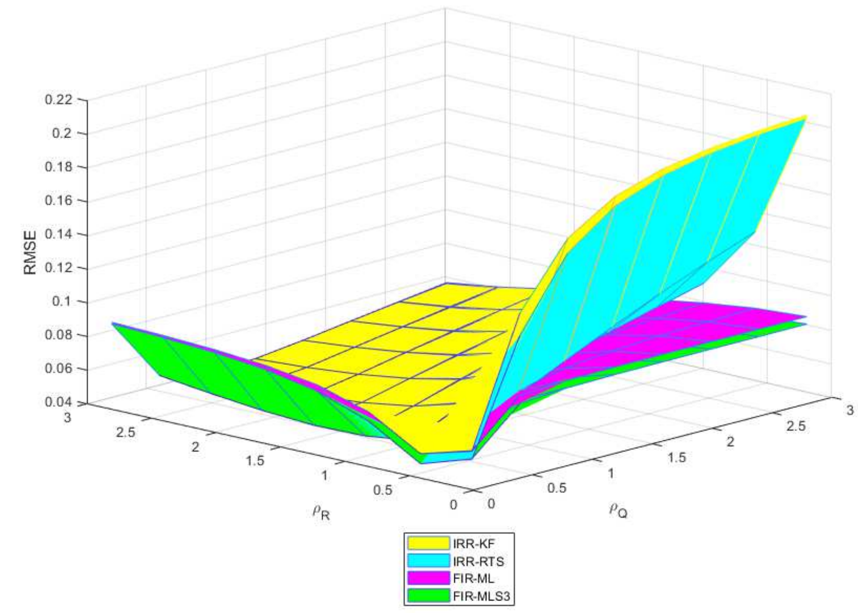


Figure 5.12: *RMSE as a function of* $\rho_{\mathbf{Q}}$ *and* $\rho_{\mathbf{R}}$*. IRR-KF: Kalman Filter; IRR-RTS: RTS Smoother; ML-FIR Filter [N=20]; FIR-MLS3 (Algorithm 3): Fixed-interval smoother [N=20].*

5.7.4 Interface level detection based on FIR-image restoration and edge detection

One of the biggest challenges during image restoration is to remove the noise on the image while preserving the features of interest. As it was described in Section 5.7.3, one of the advantages of FIR smoothing is that it can capture faster and with more accuracy the local dynamic changes in data as they only consider the closest measurement to the estimation point. Thus, FIR smothers preserve the edge-features of interest (level interface and remove) and remove the noisy pixels on the homogeneous regions(middlings and froth). Finally, the vertical edge detector can easily enhance the edge region of the real level.

The fixed-lag FIR smoother (FIR-MLS1) described in Section 5.6.2 is designed and implemented on the sight glasses regions before using the vertical edge detector on the images. The image restoration procedure is similar to the one described previously in Section 5.7.3. Figure 5.13 compares the level interface feature detection with and without considering FIR smoothing. It shows that FIR smoothing helps to remove the noise level on the homogeneous region (froth and middlings) while keeping the information on the interface change region.

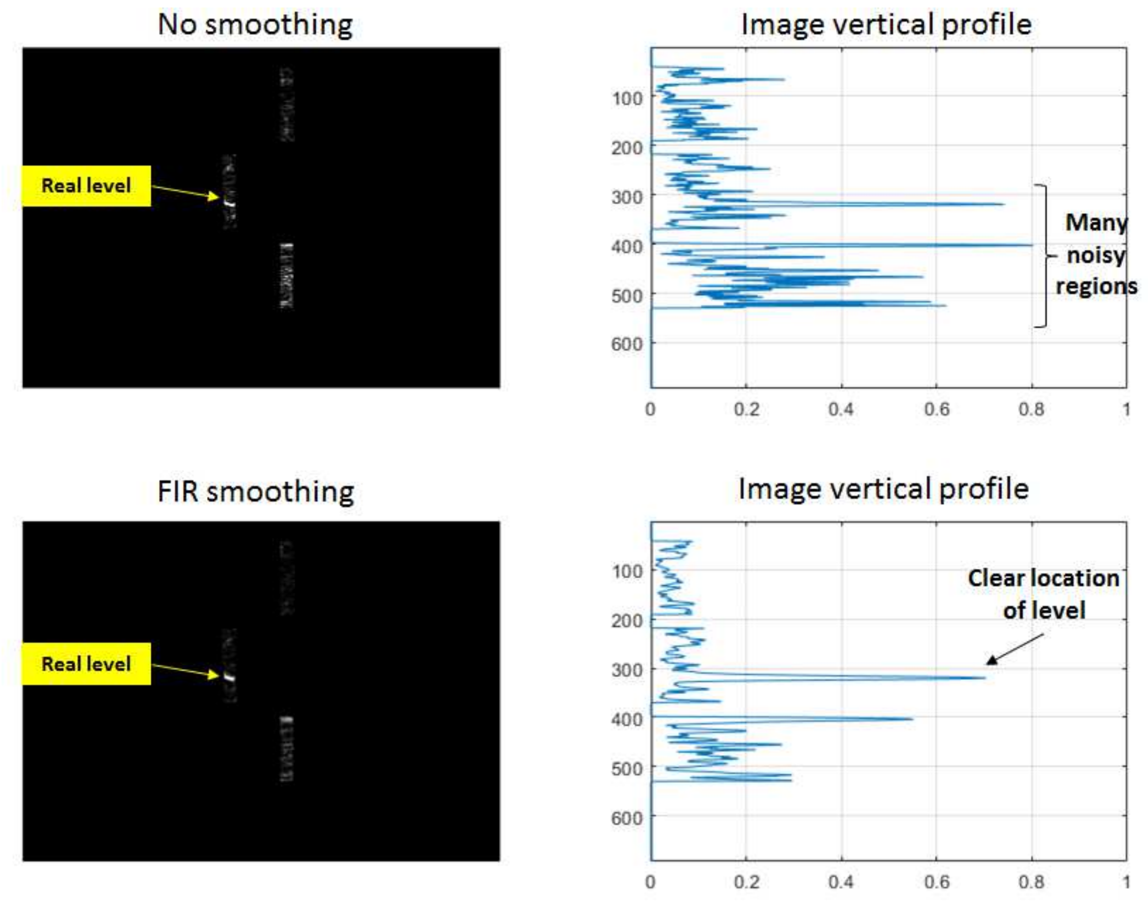


Figure 5.13: *Top: Interface level detection on the raw image. Bottom: Interface level detection with FIR smoothing*

In addition, FIR smoothing is robust to uncertainties to modeling parameters and noise statistics. Thus, **Q** and **R** are not necessary to be known to obtain a smoothing estimation of the image. Therefore, two different scenarios were simulated to illustrate the robustness of FIR against IIR. In each scenario, we consider different values of **Q** and **R** (Scenario 1: **Q**=0.01 and **R**=0.1. Scenario 2:

Q=0.01 and **R**=10). A finite horizon of **N**=10 is considered in the FIR smoothing algorithm. Figures 5.14 and 5.15 illustrates the edge level detection considering FIR smoother and RTS smoother with the two different noise model scenarios.

When noise models parameters (**Q** and **R**) are known, both structures (IIR-RTS and FIR smoothers) give a similar estimation (Scenario 1). However, when **Q** and **R** are incorrect, FIR estimations are more robust than IIR estimations (Scenario 2). In Figure 5.15, we can observe that IIR restoration blurs the image and makes very hard to find the level location. However, using the FIR restoration, the level can be distinguished clearly. The FIR smoother always provides a similar estimation no matter which **Q** and **R** values are considered.

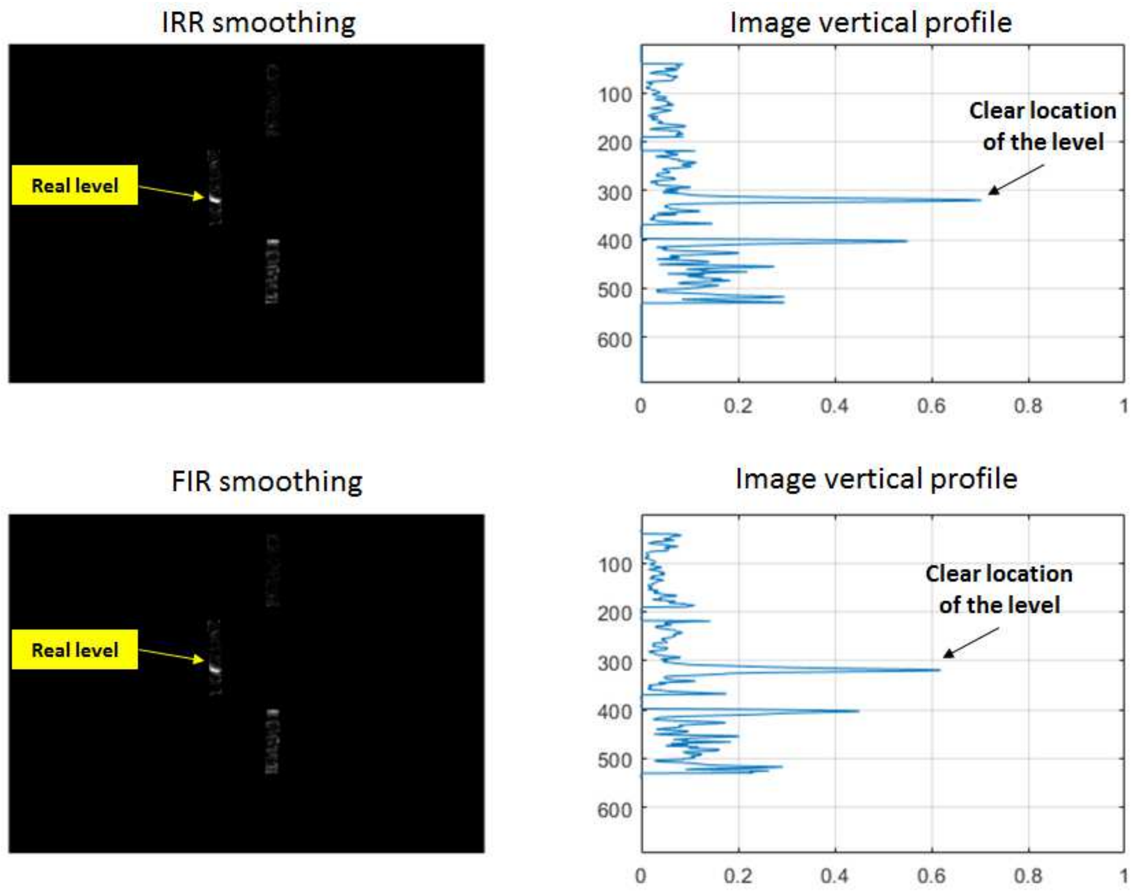


Figure 5.14: *Scenario 1: IIR and FIR edge detection with accurate noise parameters (Q=0.01 R=0.1). Top: Level interface detection with RTS smoothing. Bottom: Level interface detection with FIR smoothing.*

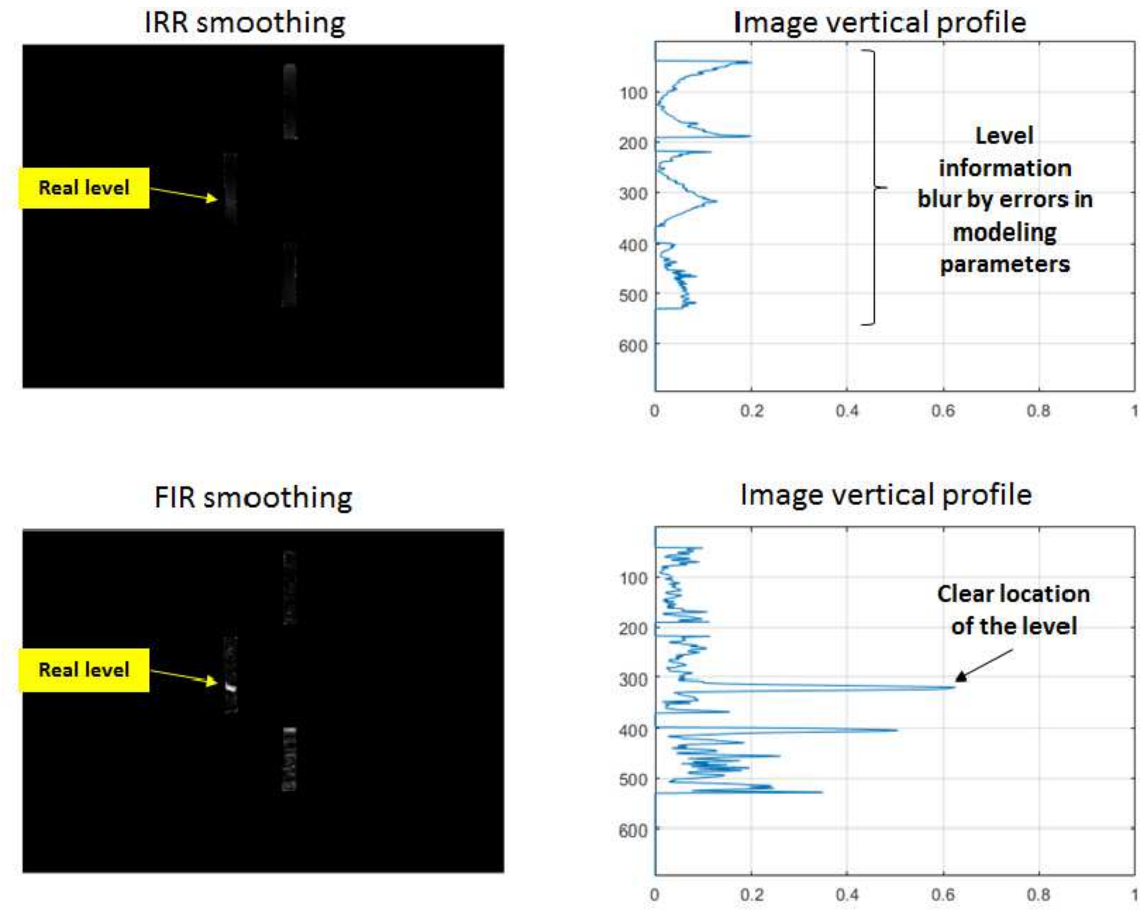


Figure 5.15: *Scenario 2: IIR and FIR edge detection with incorrect parameters (Q=0.01 R=10). Top: Level interface detection with RTS smoothing. Bottom: Level interface detection with FIR smoothing.*

5.7.5 FIR level tracking over time-based sliding windows

In computer vision, object detection is scanning and searching for an object in an image or a video. It consist of knowing the location of an object (possible with some attribute information). While tracking is the estimation of the state of a moving object based on remote measurements. It is maintaining the state and identity of an object over time despite detection errors(false negatives, false alarms), occlusions, and the presence of other objects [64].

The static and dynamic image processing step described in Section 4.2 works as an object detection algorithm (low-level image processing) that scans and finds the level location in the frames. While the time-based sliding window analysis in Section 4.2.4 works as an object tracking algorithm (high-level image processing) that estimates the level location based on the most recent measurements. The static and dynamic image processing algorithm provides the detected level measurements at every T_s seconds, but we know that these measurements could be imprecise by different factors, such as Gaussian noise, the level behind sight glasses, the level behind a stain, camera vibrations, occlusions by people or object in the environment. Therefore, we proposed to employ an FIR level tracking algorithm over the time-based sliding window to track the real level location given the noisy or uncertain level detections.

We know from the process behavior that the interface level has coherence in time. Thus, it can be model as an state-space tracking problem with constant velocity and random acceleration. In this sense, by implementing an FIR tracking model, we can estimate the level location given the noisy detection, or predict the level location when the measurements are missing or not available.

Assuming a fixed sampling interval T_s, we can derive a second order state-space level tracking model with constant velocity and random acceleration as:

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_t \ , \tag{5.25}$$

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \ , \tag{5.26}$$

with:

$$\mathbf{x_t} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{5.27}$$

$$\mathbf{F} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \tag{5.28}$$

$$\mathbf{Q} = \sigma_w^2 * \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \tag{5.29}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.30}$$

$$\mathbf{R} = \sigma_v^2 \tag{5.31}$$

where x is the level location, and $\dot{x}$ is the level velocity given by the derivative of the position (x) respect to time (T_s). $\mathbf{F}$ is the second order model of the level with constant velocity, $\mathbf{Q}$ is the model noise covariance matrix, $\mathbf{H}$ is the measurement model, and $\mathbf{R}$ is the measurement noise covariance.

Figure 5.16 illustrates the level tracking comparison between the IIR-KF and the FIR-ML with modeling errors. We assumed a 25% modeling error in $\mathbf{F}$ parameters, and we observe that IIR-KF deviates from the real level when there are errors in the modeling parameters, while FIR-ML level estimation is more accurate and converges faster to the actual location of the level.

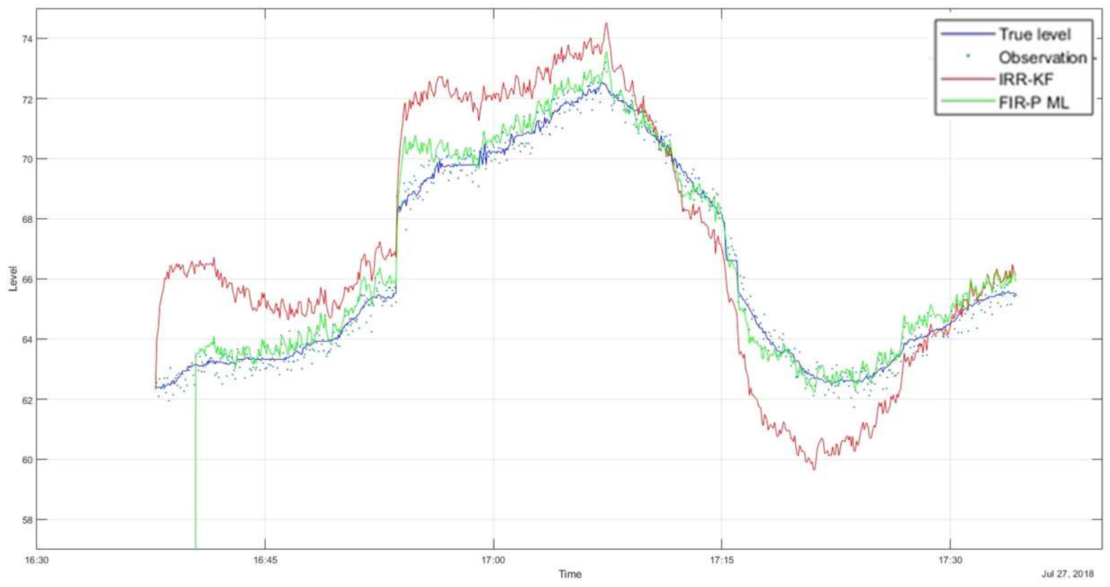


Figure 5.16: *Level tracking comparison between the IIR-KF and the FIR-ML with modeling errors.*

Figure 5.17 illustrates the level tracking comparison between the IIR-KF and the FIR-ML with presence of outlier measurements in the data. As FIR structure uses the most recent measurements it adapts faster to the real level trend than the KF.

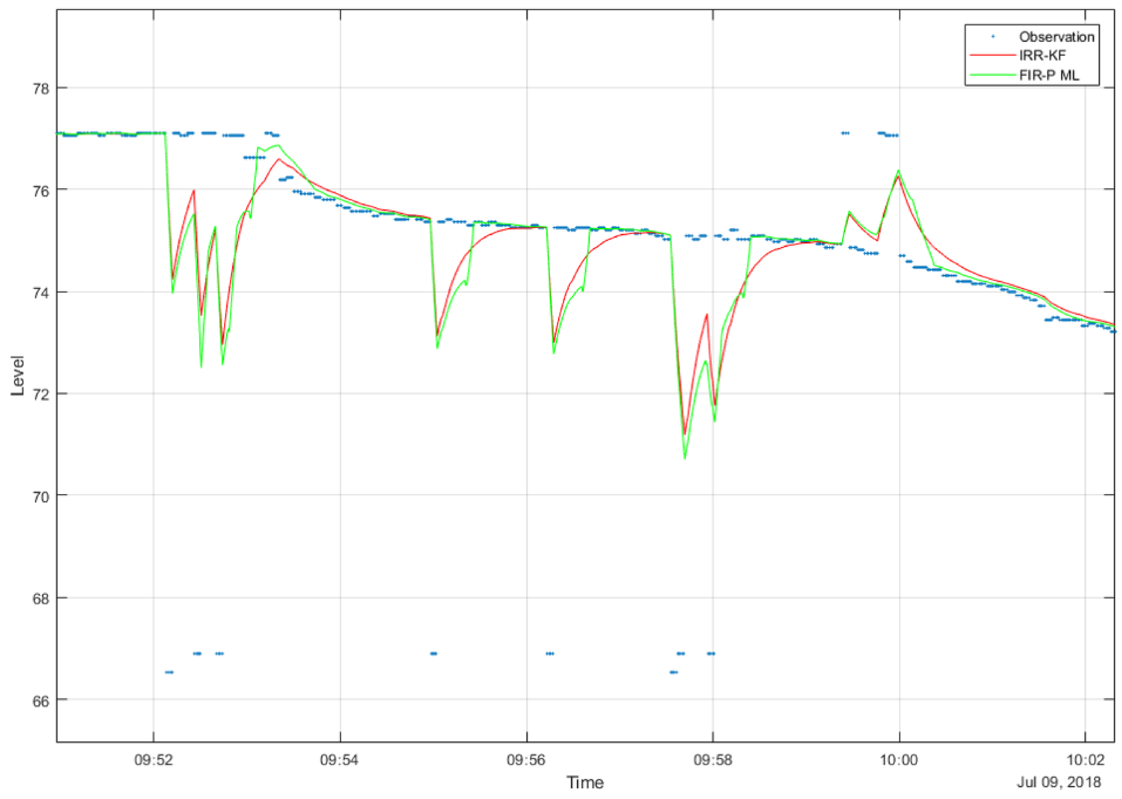


Figure 5.17: *Level tracking comparison between the IIR-KF and the FIR-ML with presence of outlier measurements in the data.*

5.8 Conclusion

In this chapter, three different FIR smoothing algorithms are developed based on a maximum likelihood estimation and a combination of estimates in the finite horizon. The Algorithm 1 (FIR-MLS1), fixed-lag smoother within the finite horizon, considers a forward-backward combination of estimates within the same finite horizon. While the Algorithm 2 (FIR-MLS2), fixed-lag smoother from different finite horizons, deals with a forward-backward combination of estimates coming from different finite horizons. Finally, the Algorithm 3 (FIR-MLS3), fixed-interval smoother, combines finite horizon estimates in one backward path for a given fixed interval of measurements. FIR-MLS1 and FIR-MLS2 are fixed-lag smoothers and

have more advantages for on-line applications where a smoothed estimate with a given lag q is required. FIR-MLS3 is a fixed-interval smoother which is more convenient for off-line applications where the interval of measurements is fixed.
Finally, two different types of applications, an image restoration and object tracking problems, are considered to compare the IIR and FIR filters performance. In both applications, it is shown that the FIR estimations are more robust against uncertainties in model parameters and noise statistics. When system model parameters are known, both structures IIR and FIR, achieve similar results. However, when there are errors in the model or noise parameters, FIR filters accomplish much more accurate estimations than traditional IIR filters. The computer vision algorithm described in Chapter 4 can be improved by implementing the FIR smoothing in the static image processing step, and the FIR tracking in the time-based sliding window analysis.

Chapter 6

Conclusions

6.1 Summary of Thesis Research

This thesis addressed the problem of froth-middlings interface level detection and estimation in Primary Separation Vessels utilizing image processing and analysis techniques. In addition to applying existing image processing techniques, we investigated and developed new approaches to improve the accuracy of the interface level detection. An experimental lab setup and video camera data from industrial PSV were employed to conduct the research and algorithms testings.

We introduced the field of image processing and analysis in Chapter 2. We particularly focused on the computer vision science, which attempts to provide machines and computers a comparable ability to the one that humans have in their eyes. The tasks of computer vision consist of methods to acquire, process, analyze and understand single or sequence of digital images. It automatically extracts high-dimensional data from the real world to produce information and helps in the decision-making process. Moreover, we provided the mathematical background to extract information from images by performing low-level operations. We described different feature extraction and segmentation methods, which are typical kickoff tasks to enhance the features of interest in the image (forth-middlings interface level).

In Chapter 3, we addressed the problem of detecting the froth-middlings interface level through a vertical edge detector mask on a single digital image. However, images are always corrupted by noise, and thus, it requires a good image restoration pre-processing step to increase the efficiency of the vertical edge detector. The aim of image restoration is to remove the noise from an image without destroying objects features(forth-middlings interface). We proposed a model-based image restoration process which is more robust than regular restorations methods. Bayesian Networks and Markov Random Field are utilized as prior graphical models.

In Chapter 4, we developed a computer vision system to process the online video stream of a camera mounted to the PSV sight glasses. The video processing algorithm is based on edge and motion detection performed on a sequence of digital video frames. In addition, it is designed with a reliability image analysis operation to handle the abnormal environmental conditions that affect the sight glass visuals. The computer vision system detects and communicates the estimated froth-middlings interface level to the distributed control system (DCS), facilitating the implementation of closed-loop control in PSV. The results under the industrial environment were presented, and they showed that the algorithm is more accurate and reliable when compared to the other instruments already installed in the vessel.

We investigated the implementation of finite impulse response (FIR) structures in image restoration and object tracking problems in Chapter 5. We addressed the problem of smoother design for state-space models based on a finite number of measurements collected in a finite horizon. Three finite impulse response (FIR) smoothing algorithms were developed using the maximum likelihood FIR estimation in a forward-backward structure along with combination. Besides, we provided equivalent iterative Kalman-like structures of these algorithms for practical implementation. Finally, we applied the FIR algorithms to the level detection problem in PSV, and the level tracking in the time-based sliding window. The results show that the FIR algorithms have better robustness against modeling and noise uncertainties than traditional filtering and smoothing approaches.

6.2 Directions for Future Work

In this thesis, we have proposed new approaches for froth-middlings interface level detection in PSV. The motivation was to developed an accurate and reliable method based on image processing techniques, and with a low computational load. To further investigate image processing and analysis method in the froth-middlings interface level problem, some directions for future work could be summarized as follows:

1. Perform automatic image detection of sight glasses: In the current computer vision solutions, the sight glasses regions or Region of Interest(ROI) have to be selected manually. The user has to select or crop the ROIs during the initialization step, and then the algorithm performs the level detection calculation inside the ROIs. Thus, a research direction is to investigate image processing techniques, such as template matching, to locate the sight glasses automatically in the frames. By having an automatic sight glasses region detection, the vision system would be able to process the image frames without any initialization step automatically. In addition, if the camera accidentally moves, then the algorithm would be able to recalibrate to the new location of the sight glasses automatically.

2. Other two properties that could be extracted using image analysis on the sigh glasses visuals, which are the color of the froth layer, and the clarity of the interface. The color of the froth layer is a very important indicator of bitumen grade. High-grade froth tends to be a rich, almost black color. Low-quality froth tends to be more brown, indicating a high clay content. Hence, an image based algorithm can be used to estimate and correlate the quality of the bitumen froth automatically. The clarity of the interface is the most important indicator of the "health" of the gravity separation vessel. A well-performing vessel has a crisp, clean interface, showing a sharp division between the froth and the middlings. This indicates very good bitumen recovery and typically occurs when processing oil sands with very little clay or fines. Conversely, a fuzzy interface indicates poor separation efficiency and warnings of low bitumen recovery. Thus, an image

based algorithm that measures the width of the interface can be developed to diagnose the separation process performance in the PSV.

3. Track and predict the level outside the sigh glasses visuals. The current computer vision system provides a level estimation in the regions where the level is visible. However, when the level is occluded or outside the sight glasses visuals, the algorithm described in Chapter 4 provides the most recent accurate measurement. Hence, by investigating high-level Visual Object Tracking (VOT) methods, it is feasible to track and predict the location of the level when not visible. Also, information about other process variables, such as PSV's inlet flow rate, can be included to make the dynamic model more accurate.

4. Make the computer vision method robust to camera vibrations. The current algorithm described in Chapter 4 can estimate the level when there are low vibrations on the camera. However, when the vibrations of the camera are high, the algorithm gets confused and holds the last accurate level measurement. Thus, the algorithm can be potentially improved by analyzing other image processing techniques which are robust to the camera vibrations.

5. Improve the initialization of FIR algorithms described in Chapter 5. The iterative maximum likelihood FIR filter must be initialized in a batch form for each finite horizon. In this sense, we have to compute a batch calculation with a few points at the beginning of the horizon to initialize the calculations, implying that the accuracy of the initialization affects on the final filtering and smoothing estimations. Hence, by substituting the batch initialization by another more accurate initialization algorithm, the finals filtering estimation would be improved.

Bibliography

[1] Goverment, A. Alberta Oil Sands Quarterly Spring 2018. 2018; `http://www.albertacanada.com/`.

[2] Alberta, Oil Sands Magazine. 2018; `http://www.oilsandsmagazine.com/`.

[3] Jampana, P.; Shah, S. L.; Kadali, R. *Control Engineering Practice* **2010**, *18*, 349–357.

[4] Li, B.; Xu, F.; Ren, Z.; Espejo, A. Primary separation vessel interface control. 2011.

[5] Narang, A.; Shah, S. L.; Chen, T.; Shukeir, E.; Kadali, R. Design of a model predictive controller for interface level regulation in oil sands separation cells. 2012.

[6] Narang, A.; Shah, S. L.; Chen, T.; Shukeir, E.; Kadali, R. *International Journal of Mineral Processing* **2015**, *145*, 94–107.

[7] Liu, Z. Camera based Primary Separation Vessel Interface Level Detection and Estimation Utilizing Markov Random Field based Image Processing. Ph.D. thesis, University of Alberta, 2016.

[8] Meribout, M.; Al Naamany, A.; Al Busaidi, K. *Expert Systems for Human, Materials and Automation*; InTech, 2011.

[9] Jampana, P.; Shah, S. *Machine Vision and Applications* **2012**, *23*, 283–298.

[10] Jampana, P.; Chitralekha, S.; Shah, S. L. *IFAC Proceedings Volumes* **2009**, *42*, 116–121.

[11] Russ, J. C.; Russ, J. C. *Introduction to image processing and analysis*; CRC press, 2007.

[12] Gonzalez, R. C.; Woods, R. E. *Upper Saddle River, NJ* **2002**,

[13] Lim, J. S. *Englewood Cliffs, NJ, Prentice Hall, 1990, 710 p.* **1990**,

[14] Szeliski, R. *Computer vision: algorithms and applications*; Springer Science & Business Media, 2010.

[15] Ballard, D. H. and Brown, C. Computer Vision. 1982.

[16] Dutta, S.; Chaudhuri, B. B. A color edge detection algorithm in RGB color space. 2009.

[17] Canny, J. *IEEE Transactions on pattern analysis and machine intelligence* **1986**, 679–698.

[18] Dutta, S.; Chaudhuri, B. B. A statistics and local homogeneity based color edge detection algorithm. 2009.

[19] Maru, M.; Parikh, M. *International Journal of Computer Applications* **2017**, *160*.

[20] Li, S. Z. *Markov random field modeling in image analysis*; Springer Science & Business Media, 2009.

[21] Bouman, C. A. *Purdue University* **2013**,

[22] Bishop, C. M. *Pattern recognition and machine learning*; springer, 2006.

[23] Simon, D. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*; John Wiley & Sons, 2006.

[24] Eppel, S.; Kachman, T. *arXiv preprint arXiv:1404.7174* **2014**,

[25] Chakravarthy, S.; Sharma, R.; Kasturi, R. *IEEE transactions on Instrumentation and measurement* **2002**, *51*, 353–361.

[26] Pithadiya, K. J.; Modi, C. K.; Chauhan, J. D. Comparison of optimal edge detection algorithms for liquid level inspection in bottles. 2009.

[27] Prewitt, J. M. *Picture processing and Psychopictorics* **1970**, *10*, 15–19.

[28] Liu, H.; Shah, S.; Jiang, W. *Computers & chemical engineering* **2004**, *28*, 1635–1647.

[29] Elliott, D. F. *Handbook of digital signal processing: engineering applications*; Academic press, 2013.

[30] Jazwinski, A. *Google Scholar* **1970**,

[31] Shmaliy, Y. S.; Zhao, S.; Ahn, C. K. *IEEE Control Systems* **2017**, *37*, 70–89.

[32] Zhao, S.; Shmaliy, Y. S.; Liu, F. *IEEE Transactions on Signal Processing* **2016**, *64*, 2284–2297.

[33] Kalman, R. E. *Journal of basic Engineering* **1960**, *82*, 35–45.

[34] Zhang, L.; Ning, Z.; Wang, Z. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2016**, *46*, 559–572.

[35] Fitzgerald, R. *IEEE Transactions on Automatic Control* **1971**, *16*, 736–747.

[36] Lin, H.; Sun, S. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2017**,

[37] Sangsuk-Iam, S.; Bullock, T. E. *IEEE Transactions on Automatic Control* **1990**, *35*, 1304–1309.

[38] Zhang, D.; Song, H.; Yu, L. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2017**, *47*, 1826–1838.

[39] Li, W.; Wei, G.; Ding, D.; Liu, Y.; Alsaadi, F. E. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2016**,

[40] Fraser, D.; Potter, J. *IEEE Transactions on automatic control* **1969**, *14*, 387–390.

[41] Rauch, H. E.; Striebel, C.; Tung, F. *AIAA journal* **1965**, *3*, 1445–1450.

[42] Li, H.; Fu, M. *IEEE Transactions on Signal Processing* **1997**, *45*, 2338–2350.

[43] Kwon, W. H.; Han, S. H. *Receding horizon control: model predictive control for state models*; Springer Science & Business Media, 2006.

[44] Pomárico-Franquiz, J. J.; Granados-Cruz, M.; Shmaliy, Y. S. *IEEE Journal of Selected Topics in Signal Processing* **2015**, *9*, 229–238.

[45] Shmaliy, Y. S.; Simon, D. *EURASIP Journal on Advances in Signal Processing* **2013**, *2013*, 113.

[46] Kwon, W. H.; Kim, P. S.; Han, S. H. *Automatica* **2002**, *38*, 545–551.

[47] Zhao, S.; Shmaliy, Y. S.; Liu, F. *IEEE Signal Processing Letters* **2015**, *22*, 718–722.

[48] Ahn, C. K.; Han, S.; Kwon, W. H. *IEEE Transactions on Circuits and Systems II: Express Briefs* **2007**, *54*, 97–101.

[49] Shmaliy, Y. S. *IEEE Transactions on Signal Processing* **2010**, *58*, 3086–3096.

[50] Shmaliy, Y. S.; Ibarra-Manzano, O. *International Journal of Adaptive Control and Signal Processing* **2012**, *26*, 95–104.

[51] Shmaliy, Y. S. *ieee transactions on ultrasonics, ferroelectrics, and frequency control* **2006**, *53*, 862–870.

[52] Shmaliy, Y. S. *IEEE Transactions on Signal Processing* **2011**, *59*, 2465–2473.

[53] Shmaliy, Y. S. *IEEE Transactions on Signal Processing* **2012**, *60*, 5519–5527.

[54] Ahn, C.; Han, S.; Kwon, W. H. *IEEE Signal Processing Letters* **2006**, *13*, 557–560.

[55] Yin, S.; Wang, J.; Liu, T. *Indonesian Journal of Electrical Engineering and Computer Science* **2016**, *2*, 344–350.

[56] Zhao, S.; Shmaliy, Y. S.; Shi, P.; Ahn, C. K. *IEEE Transactions on Industrial Electronics* **2017**, *64*, 3075–3083.

[57] Ahn, C. K.; Shi, P.; Basin, M. V. *IEEE Transactions on Circuits and Systems I: Regular Papers* **2016**, *63*, 1210–1221.

[58] Zhao, S.; Huang, B.; Shmaliy, Y. S. *Automatica* **2017**, *85*, 91–99.

[59] Zhao, S.; Shmaliy, Y. S. *IEEE signal processing letters* **2016**, *23*, 1848–1852.

[60] Kwon, W. H.; Lee, K. S.; Lee, J. H. *Automatica* **1994**, *30*, 489–492.

[61] Crouse, D. F.; Willett, P.; Bar-Shalom, Y. *IEEE Signal Processing Letters* **2010**, *17*, 177–180.

[62] Aravkin, A. Y.; Burke, J. V.; Pillonetto, G. *SIAM Journal on Control and Optimization* **2014**, *52*, 2891–2916.

[63] Morales-Mendoza, L.; Vázquez-Bautista, R.; Morales-Mendoza, E.; Shmaliy, Y.; Gamboa-Rosales, H. *Procedia Engineering* **2012**, *35*, 202–209.

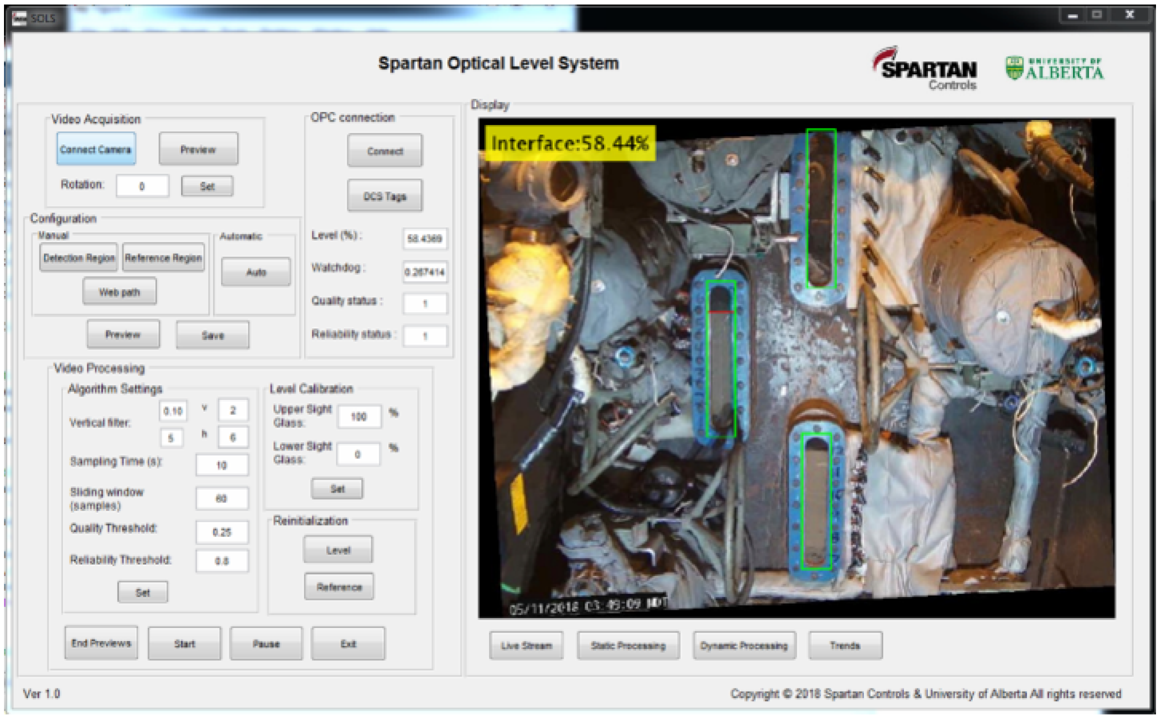
[64] Zhao, Y.; Shi, H.; Chen, X.; Li, X.; Wang, C. An overview of object detection and tracking. 2015.

Appendix A

Appendix to Chapter 4

A.1 Video Processing Application Graphical User Interface

The VPA is a standalone executable application that can be installed in any machine. The Graphical User Interface (GUI) allows the user to configure, execute, monitor and tune the video processing algorithm (Figure A1). On the left side, the GUI has banners to establish the input video camera communication, the image processing regions of interest, the tuning parameters values of the algorithm, the level calibration values and the communication outputs tags to DCS. On the right side, it has a video display to visualize the online video stream with the level information overlay. In addition, the display can be switch to the static image processing output or to the dynamic image processing output to visualize and tune the video algorithm calculations.
The GUI provides a visualization trend to monitor the algorithm performance online during the last hour. It is a figure with three charts, Figure A2. The first chart displays the level estimation trend, the second chart displays the quality index of the level estimation, and the third chart displays the reliability index of the estimation.



Rotation: 0

Level (%) : 58.4369
Watchdog : 0.267414
Quality status : 1
Reliability status : 1

Vertical filter: 0.10 v 2
5 h 6
Sampling Time (s): 10
Sliding window (samples) 60
Quality Threshold: 0.25
Reliability Threshold: 0.8

Upper Sight Glass: 100 %
Lower Sight Glass: 0 %

Ver 1.0

Copyright © 2018 Spartan Controls & University of Alberta All rights reserved

Figure A1: *Video Processing Application Graphical User Interface.*

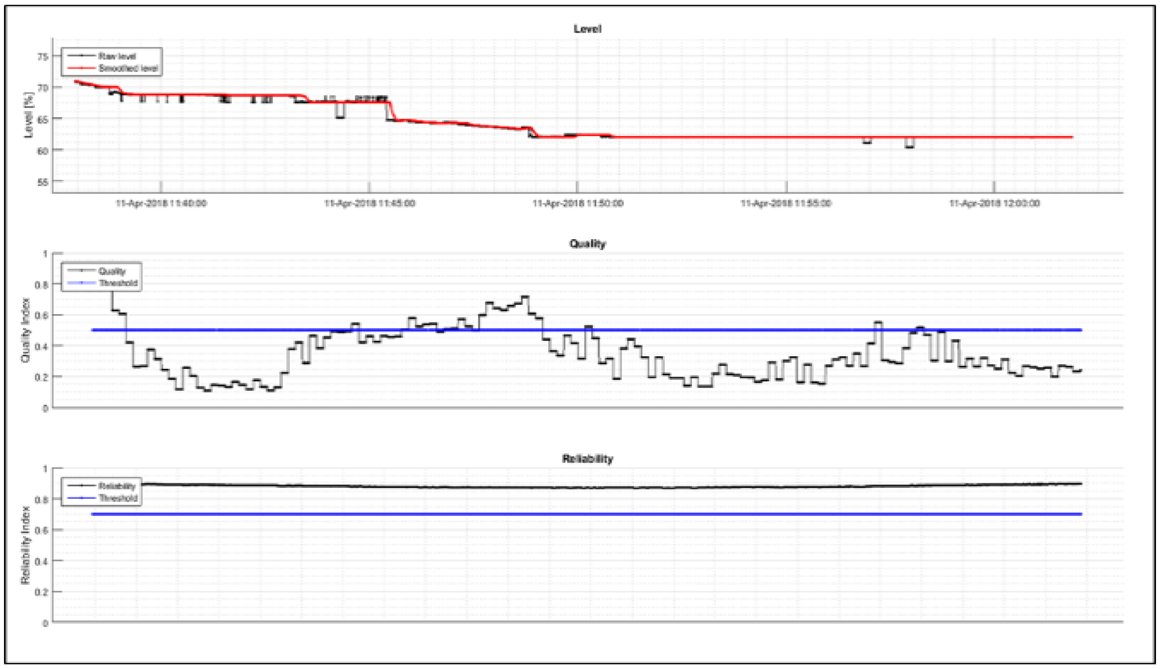


Figure A2: *Video Processing Application trends.*

A.2 Video processing application configuration and tuning

The video processing algorithm has parameters that can be tuned to improve the accuracy of the level estimation. These parameters depends on the process behavior, the environmental conditions on the sight glasses room, and the image quality provided by the acquisition device. The following sections describe the parameters that can be tuned in the video processing algorithm.

A.2.1 Initial Configurations

Before starting the VPA online, there are initial settings and configuration that has to be specified:

1. Sight glasses or regions of interest (ROI) in the frames.
2. Reliability reference region in the frames.
3. Web path for video output display in control room.
4. OPC settings to communicate the level output to the DCS.

Essentially, configurations 1 and 2 are related to the video-processing algorithm, while configurations 3 and 4 are related to the communication of the outputs. Figure 4.3 shows an example of an image preview with configurations 1 and 2 already established on the PSV sight glasses. The green rectangles correspond to the sight glasses region where the algorithm executes the calculations to detect the level. The red rectangle corresponds to the region where the algorithm executes the calculation to alert any abnormal condition in the environment that affects the level estimation.

A.2.2 Sight Glasses or ROI

The target of the algorithm is the visuals through the sight glasses (SG). The user has to define the regions where the algorithm looks for the interface level. The user must manually select 3 regions (each corresponding to the each of the sight glasses). This operation creates a mask that limits the algorithm calculations to within the ROI mask. For example, if the vessel has 3 sight glasses, the user should select 3 rectangles (one for each SG). The green rectangles in Figure 4.3 represent the sight glasses ROI selected by the user in this case. In section 4.2.2 we describe with more detail the image processing calculations performed inside the sight glasses ROI.

A.2.3 Reliability reference region (RRR)

The selection of a reliability reference region adds robustness to the video processing algorithm, especially during scenarios, such as people/objects crossing in front of the camera, unintentional camera motions or vibrations, lighting changes, and conditions that affect quality of video acquisition (mist, condensations, ice). The algorithm will hold the last reliably detected level if the image acquisition conditions have been changed, or eventually alarm the user if the conditions have been changed drastically that prevents the accurate tracking of the level. The user has to define the regions where the algorithm performs the reliability image analysis. This operation creates a binary matrix, $\mathbf{BM_R}$, that limits the analysis to within the reliability region. The red rectangle in Figure 4.3 represent the reliability region selected in this case.
The user can reset the reliability reference region in the VPA at any time. In this case, the algorithm will consider the image at that particular instant as the reliability reference region $\mathbf{X}_R(t_{ref}) = \mathbf{X}_R(t_i)$. It is important to select a good quality image (with no environmental obstructions) as the reliability reference region.

A.2.4 Web path for video output display

The user should specify a web folder in which the VPA stores the information of the live stream with the detected level as shown in Figure 17. The images from this web folder are used by an Internet Information Services (IIS) to stream the images online in a web browser. Therefore, any machine that has access to the network can use the web browser to monitor the video processing application results in real time. The main purpose of this web server is to have a visual monitoring of the level detection in the DCS station located in the operation control room.
An IIS is an extensible web server created by Microsoft for use with the Windows NT (New Technology) family. ISS supports HTTP, HTTP/2, HTTPS, FTP, FTPS, SMTP and NNTP communication protocols. An HTTP script uploads the images, saved by the VPA in the web folder, to the web browser. Then, the IIS extends the web visualization to any machine in the same network.

A.2.5 Open Protocol Communication (OPC) Settings

The VPA has the function to create an OPC client and send/receive data to the DCS server. The OPC operations are integrated into the video application algorithm.
OPC connection:
The VPA creates an OPC data access (OPCDA) client and connects that client to

the DCS server. First, an OPC data access object must be created for the host specified by the server IP address and ID (the host could be a local server or a remote server). Then, it connects the client to the DCS server.
DCS modules:
The user must specify the address of the DCS landing modules where the parameters will be written in the DCS. A total of 8 parameters are written by the VPA in the DCS. These parameters are the estimated level , watchdog , quality status, reliability status, quality index, reliability index, quality threshold and reliability threshold. The writing operations are performed inside the VPA once the parameters are calculated.
The watchdog is a parameter that monitors the VPA operation and the camera communication. Essentially, it is a random number that changes in every calculation loop of the algorithm. Therefore, if the application fails the watchdog value will keep frozen in a particular number alerting the user that the VPA is not working. In addition, the watchdog value turns into 0 when the communication to the camera is lost.

A.2.6 Number of frames for static-dynamic image processing

The algorithm needs a sequence of consecutive frames to analyze the dynamics of the images over a period of time. These number of frames that the video application accumulates into the image-sliding window can tuned. In this VPA example, the trigger is set to automatically accumulate 5 frames in the image sliding window. But, by increasing the number of frames, we can increase the robustness of the motion detection operation.

A.2.7 Sample time for dynamic image processing

The frame period acquisition of the algorithm is set to 1 frame per second, and the level estimation is calculated every 1 second. However, the user can specify the image sampling rate T_s for the dynamic image calculation. This number determines the interval between frame differences calculations. For example, if the user sets $T_s = 1$, it implies that the application performs the frame differences calculation between two consecutive frames, $\mathbf{X_s}(t_i)$ and $\mathbf{X_s}(t_{i-1})$, with an interval of 5 seconds. Increasing T_s increases the algorithms stability and robustness since there are higher chances to observer image changes (motion) in consecutive frames over time. Conversely, a very small T_s increases the influence of noise, which potentially leads to reduced accuracy in the level detection.
T_s has to be tuned based on the properties of the camera (fps, resolution) and the time constant of the process (slow/fast). PSV interface level does not change

rapidly, so the recommendation based on the process behavior is to use a dynamic frame period time no less than $T_s = 5$.

A.2.8 Vertical edge filter

The user can tune the vertical sharpening filtering size by changing two parameters, v and h, which are vertical and horizontal filter size, respectively. For instance, by increasing vertical size, v, increases the number of rows to the filter mask, which will consider more vertical neighbor pixels in the calculation. Increasing the horizontal size, h, means that more horizontal neighbors pixels are considered in the filtering calculation. In summary, by increasing the vertical size the filter will look for more contrast changes between upper and lower region on the vertical axis, while in increasing the horizontal size the filter will look for more similar pixel characteristic over the horizontal axis.

A.2.9 Sliding window length

The user can specify the length of the time-based sliding window,W, which gives stronger or weaker filtering effect to the level measurement in time. A larger time sliding window increases the outliers/smoothing effect, while a shorter time sliding windows decrease outliers/smoothing effect.

A.2.10 Quality index threshold

The quality index threshold Q_{th} is a threshold beyond which the changes are attributed to the real level motion in the frames(dynamic image processing). If Q_{th} is very small, the algorithm will be sensitive to any type of motion and also to noise, while with a large Q_{th}, the algorithm will be sensitive to only the significant changes in the images. However, if no motion is detected because of the current detection being below the threshold, the algorithm will keep the level at the closest edge (static image processing). Therefore, the quality index threshold Q_{th} can be considered as the threshold between dynamic and static image processing algorithms. Above Q_{th}, the algorithm estimates the level based on both static and dynamic image processing, while below Q_{th} the algorithm estimates the level based on the static image processing. The VPAs trends in Figure A2 displays the Q_{idx} and Q_{th} for monitoring and tuning purposes. The task is to set a threshold number in order to allow the algorithm to detect real level motion.

A.2.11 Reliability index threshold

The reliability index threshold R_{th} is a threshold beyond which the changes in the environment will be considered as abnormal for the image-processing algorithm. The user can change and tune R_{th} based on the environmental condition in the room where the video camera is installed. If R_{th} is very large, the algorithm will be sensitive to any type of environmental changes, while with a small R_{th}, the algorithm will be sensitive to only the significant changes in the environment. The VPAs trends in Figure 19 displays the R_{idx} and R_{th} for monitoring and tuning purposes.

Appendix B

Appendix to Chapter 5

B.1 Interative ML-FIR estimation

Given the system models (5.2) and (5.3) with zero mean and white Gaussian noises $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ and $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_n)$, the iterative ML-FIR estimate $\hat{\mathbf{x}}^{ML}_{n|N}$ and covariance $\mathbf{P}^{ML}_{n|N}$ based on a finite horizon of measurements $\mathbf{y}_{m:n}$ are

$$
\begin{aligned}
\hat{\mathbf{x}}_i &= \mathbf{F}_i\hat{\mathbf{x}}_{i-1} + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)(\mathbf{y}_i - \mathbf{H}_i\mathbf{F}_i\hat{\mathbf{x}}_{i-1}) \ , \\
\mathbf{P}_{i|N} &= (\mathbf{I} - (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{H}_i)(\mathbf{F}_i\mathbf{P}_{i-1|N}\mathbf{F}_i^T + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T)(\cdots)^T \\
&\quad + (\mathbf{K}_i + \tilde{\mathbf{K}}_i)\mathbf{R}_i(\cdots)^T \ ,
\end{aligned}
$$

with

$$
\begin{aligned}
\mathbf{P}_i &= \mathbf{F}_i(\mathbf{I} - \mathbf{K}_{i-1}\mathbf{H}_{i-1})\mathbf{P}_{i-1}\mathbf{F}_i^T + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ , \\
\mathbf{K}_i &= \mathbf{P}_i\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\bar{\mathbf{P}}_i &= \mathbf{F}_i(\mathbf{I} - \mathbf{K}_{i-1}\mathbf{H}_{i-1})\bar{\mathbf{P}}_{i-1} \ , \\
\bar{\mathbf{K}}_i &= \bar{\mathbf{P}}_i\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\mathbf{Z}_i &= (\mathbf{Z}_{i-1}^{-1} + \bar{\mathbf{K}}_i\mathbf{H}_i\bar{\mathbf{P}}_i)^{-1} \ , \\
\tilde{\mathbf{K}}_i &= (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\bar{\mathbf{P}}_i\mathbf{Z}_i\bar{\mathbf{K}}_i \ ,
\end{aligned}
$$

where i ranges from $\varepsilon = max(m + K, m + 2)$ to n, and the true outputs $\hat{\mathbf{x}}^{ML}_{n|N} = \hat{\mathbf{x}}_i$

and $\mathbf{P}^{ML}_{n|N} = \mathbf{P}_{i|N}$ are taken when $i = n$ with initial states computed at $\alpha = \varepsilon - 1$ as:

$$\begin{aligned}
\mathbf{P}_\alpha &= \bar{\mathbf{G}}_{\alpha,m}\mathbf{Q}_{\alpha,m}\bar{\mathbf{G}}^T_{\alpha,m} - \mathbf{F}_\alpha\mathbf{D}^T_{\alpha-1}\mathbf{F}^T_\alpha \ , \\
\mathbf{Z}_\alpha &= (\mathbf{H}^T_{\alpha,m}\mathbf{\Delta}^{-1}_{\alpha,m}\mathbf{H}_{\alpha,m})^{-1} \ , \\
\bar{\mathbf{P}}_\alpha &= \mathbf{F}^m_\alpha - \mathbf{F}_\alpha\mathbf{D}_{\alpha-1}\mathbf{\Delta}^{-1}_{\alpha-1,m}\mathbf{H}_{\alpha-1,m} \ , \\
\mathbf{K}^{ML}_{\alpha,m} &= (\mathbf{F}^m_\alpha\mathbf{Z}_\alpha\mathbf{H}^T_{\alpha,m} + \mathbf{D}_\alpha - \mathbf{D}_\alpha\mathbf{\Delta}^{-1}_{\alpha,m}\mathbf{H}_{\alpha,m}\mathbf{Z}_\alpha\mathbf{H}^T_{\alpha,m})\mathbf{\Delta}^{-1}_{\alpha,m} \ , \\
\hat{\mathbf{x}}_\alpha &= \mathbf{K}^{ML}_{\alpha,m}\mathbf{Y}_{\alpha,m} \ , \\
\mathbf{P}_{\alpha|N} &= (\mathbf{K}^{ML}_{\alpha,m}\mathbf{L}_{\alpha,m} - \bar{\mathbf{G}}_{\alpha,m})\mathbf{Q}_{\alpha,m}(\cdots)^T \\
&\quad + \mathbf{K}^{ML}_{\alpha,m}\mathbf{R}_{\alpha,m}\mathbf{K}^{ML^T}_{\alpha,m} \ .
\end{aligned}$$

B.2 Algorithm 1 (FIR-MLS1): Fixed-lag FIR smoothing within the finite horizon

Detail description of Algorithm 1 from Table 3 in Section 5.6.2. Perform the following steps:

1. Forward ML FIR estimation:
1.1. Initialize the forward FIR estimation in a batch form from m to α:

$$\begin{aligned}
\mathbf{P}_\alpha &= \bar{\mathbf{G}}_{\alpha,m}\mathbf{Q}_{\alpha,m}\bar{\mathbf{G}}^T_{\alpha,m} - \mathbf{F}_\alpha\mathbf{D}^T_{\alpha-1}\mathbf{F}^T_\alpha \ , \\
\mathbf{Z}_\alpha &= (\mathbf{H}^T_{\alpha,m}\mathbf{\Delta}^{-1}_{\alpha,m}\mathbf{H}_{\alpha,m})^{-1} \ , \\
\bar{\mathbf{P}}_\alpha &= \mathbf{F}^m_\alpha - \mathbf{F}_\alpha\mathbf{D}_{\alpha-1}\mathbf{\Delta}^{-1}_{\alpha-1,m}\mathbf{H}_{\alpha-1,m} \ , \\
\mathbf{K}^{ML}_{\alpha,m} &= (\mathbf{F}^m_\alpha\mathbf{Z}_\alpha\mathbf{H}^T_{\alpha,m} + \mathbf{D}_\alpha - \mathbf{D}_\alpha\mathbf{\Delta}^{-1}_{\alpha,m}\mathbf{H}_{\alpha,m}\mathbf{Z}_\alpha\mathbf{H}^T_{\alpha,m})\mathbf{\Delta}^{-1}_{\alpha,m} \ , \\
\hat{\mathbf{x}}_\alpha &= \mathbf{K}^{ML}_{\alpha,m}\mathbf{Y}_{\alpha,m} \ , \\
\mathbf{P}_{\alpha|N} &= (\mathbf{K}^{ML}_{\alpha,m}\mathbf{L}_{\alpha,m} - \bar{\mathbf{G}}_{\alpha,m})\mathbf{Q}_{\alpha,m}(\cdots)^T \\
&\quad + \mathbf{K}^{ML}_{\alpha,m}\mathbf{R}_{\alpha,m}\mathbf{K}^{ML^T}_{\alpha,m} \ .
\end{aligned}$$

1.2. For $i = \alpha, \cdots, n$ perform the forward iterative ML-FIR estimation, and keep the prior and posterior estimates.

$$\begin{aligned}
\mathbf{P}_{f,i} &= \mathbf{F}_i(\mathbf{I} - \mathbf{K}_{f,i-1}\mathbf{H}_{i-1})\mathbf{P}_{f,i-1}\mathbf{F}_i^T + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ , \\
\mathbf{K}_{f,i} &= \mathbf{P}_{f,i}\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_{f,i}\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\bar{\mathbf{P}}_{f,i} &= \mathbf{F}_i(\mathbf{I} - \mathbf{K}_{f,i-1}\mathbf{H}_{i-1})\bar{\mathbf{P}}_{f,i-1} \ , \\
\bar{\mathbf{K}}_{f,i} &= \bar{\mathbf{P}}_{f,i}\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_{f,i}\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\mathbf{Z}_{f,i} &= (\mathbf{Z}_{f,i-1}^{-1} + \bar{\mathbf{K}}_{f,i}\mathbf{H}_i\bar{\mathbf{P}}_{f,i})^{-1} \ , \\
\tilde{\mathbf{K}}_{f,i} &= (\mathbf{I} - \mathbf{K}_{f,i}\mathbf{H}_i)\bar{\mathbf{P}}_{f,i}\mathbf{Z}_{f,i}\bar{\mathbf{K}}_{f,i} \ , \\
\hat{\mathbf{x}}_{f,i}^- &= \mathbf{F}_i\hat{\mathbf{x}}_{f,i-1}^+ \ , \\
\mathbf{P}_{f,i}^- &= \mathbf{F}_i\mathbf{P}_{f,i-1}^+\mathbf{F}_i^T\mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ , \\
\hat{\mathbf{x}}_{f,i}^+ &= \hat{\mathbf{x}}_{f,i}^- + (\mathbf{K}_{f,i} + \tilde{\mathbf{K}}_{f,i})(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_{f,i}^-) \ , \\
\mathbf{P}_{f,i}^+ &= (\mathbf{I} - (\mathbf{K}_{f,i} + \tilde{\mathbf{K}}_{f,i})\mathbf{H}_i)(\mathbf{P}_{f,i}^-)(\cdots)^T \\
&\quad + (\mathbf{K}_{f,i} + \tilde{\mathbf{K}}_{f,i})\mathbf{R}_i(\cdots)^T \ .
\end{aligned}$$

2. Backward ML FIR estimation:

2.1. Initialize the backward ML FIR estimation $\hat{\mathbf{x}}_{b,n}$ using the forward ML FIR estimation $\hat{\mathbf{x}}_{f,n}$ at the last point of the finite horizon:

$$\begin{aligned}
\hat{\mathbf{x}}_{b,n}^+ &= \hat{\mathbf{x}}_{f,n}^- \ , \\
\mathbf{P}_{b,n}^+ &= \mathbf{P}_{f,n}^+ \ .
\end{aligned}$$

2.2. For $i = n-1, \cdots, m$ perform the backward iterative ML FIR estimation, and keep the prior and posterior estimates. Remember that for backward estimations we need to consider the inverse model parameters $\mathbf{F}_i^{-1}$:

$$\begin{aligned}
\mathbf{P}_{b,i} &= \mathbf{F}_i^{-1}(\mathbf{I}-\mathbf{K}_{b,i+1}\mathbf{H}_{i+1})\mathbf{P}_{b,i+1}\mathbf{F}_i^{-T} + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ , \\
\mathbf{K}_{b,i} &= \mathbf{P}_{b,i}\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_{b,i}\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\bar{\mathbf{P}}_{b,i} &= \mathbf{F}_i^{-1}(\mathbf{I}-\mathbf{K}_{b,i+1}\mathbf{H}_{i+1})\bar{\mathbf{P}}_{b,i+1} \ , \\
\bar{\mathbf{K}}_{b,i} &= \bar{\mathbf{P}}_{b,i}\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_{b,i}\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \ , \\
\mathbf{Z}_{b,i} &= (\mathbf{Z}_{b,i+1}^{-1} + \bar{\mathbf{K}}_{b,i}\mathbf{H}_i\bar{\mathbf{P}}_{b,i})^{-1} \ , \\
\tilde{\mathbf{K}}_{b,i} &= (\mathbf{I}-\mathbf{K}_{b,i}\mathbf{H}_i)\bar{\mathbf{P}}_{b,i}\mathbf{Z}_{b,i}\bar{\mathbf{K}}_{b,i} \ , \\
\hat{\mathbf{x}}_{b,i}^- &= \mathbf{F}_i^{-1}\hat{\mathbf{x}}_{b,i+1}^+ \ , \\
\mathbf{P}_{b,i}^- &= \mathbf{F}_i^{-1}\mathbf{P}_{b,i+1}^+\mathbf{F}_i^{-T}\mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i^T \ , \\
\hat{\mathbf{x}}_{b,i}^+ &= \hat{\mathbf{x}}_{b,i}^- + (\mathbf{K}_{b,i}+\tilde{\mathbf{K}}_{b,i})(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_{b,i}^-) \ , \\
\mathbf{P}_{b,i}^+ &= (\mathbf{I}-(\mathbf{K}_{b,i}+\tilde{\mathbf{K}}_{b,i})\mathbf{H}_i)(\mathbf{P}_{b,i}^-)(\cdots)^T \\
&\quad + (\mathbf{K}_{b,i}+\tilde{\mathbf{K}}_{b,i})\mathbf{R}_i(\cdots)^T.
\end{aligned}$$

3. Obtain ML FIR smoother estimates:
3.1. For $i = n-1, \cdots, m$ combine forward and backward ML FIR estimations to obtain ML FIR smoother estimates $\hat{\mathbf{x}}_{s,i}$. This calculation can be done synchronously with backward ML-FIR estimation.

$$\begin{aligned}
\mathbf{K}_{s,i} &= \mathbf{P}_{b,i}^-(\mathbf{P}_{f,i}^+ + \mathbf{P}_{b,i}^-)^{-1} \ , \\
\hat{\mathbf{x}}_{s,i} &= \mathbf{K}_{s,i}\hat{\mathbf{x}}_{f,i}^+ + (\mathbf{I}-\mathbf{K}_{s,i})\hat{\mathbf{x}}_{b,i}^- \ , \\
\mathbf{P}_{s,i} &= ((\mathbf{P}_{f,i}^+)^{-1} + (\mathbf{P}_{b,i}^-)^{-1})^{-1} \ .
\end{aligned}$$

B.3 Algorithm 2 (FIR-MLS2): Fixed-lag FIR smoother from different finite horizons

Detail description of Algorithm 2 from Table 4 in section 5.6.4. Perform the following steps:

1. Forward FIR estimation:
For $i = m, \cdots, n$ use forward ML-FIR estimation from Algorithm 1 to obtain prior and posterior estimates at the end of the finite horizon $\hat{x}_n$, and keep them:

$$\begin{aligned}
&\hat{\mathbf{x}}_{f,n}^-, \mathbf{P}_{f,n}^- \ , \\
&\hat{\mathbf{x}}_{f,n}^+, \mathbf{P}_{f,n}^+ \ .
\end{aligned}$$

2. Backward FIR estimation:
For $i = n, \cdots, m$ use the backward ML-FIR estimation from Algorithm 1 to obtain prior and posterior estimates at the beginning of the finite horizon $\hat{\mathbf{x}}_m$, and keep them:

$$\hat{\mathbf{x}}^-_{b,m}, \mathbf{P}^-_{b,m} \ ,$$
$$\hat{\mathbf{x}}^+_{b,m}, \mathbf{P}^+_{b,m} \ .$$

3. Forward-backward FIR combination:
For $i = n, \cdots, N-m$ combine forward, $\hat{\mathbf{x}}_n$, and backward, $\hat{\mathbf{x}}_m$, ML-FIR estimates from different finite horizons. Forward and backward estimates must coincide for the same point of interest:

$$\begin{aligned}\mathbf{K}_{s,i} &= \mathbf{P}^-_{b,i}(\mathbf{P}^+_{f,i} + \mathbf{P}^-_{b,i})^{-1} \ ,\\ \hat{\mathbf{x}}_{s,i} &= \mathbf{K}_{s,i}\hat{\mathbf{x}}^+_{f,i} + (\mathbf{I} - \mathbf{K}_{s,i})\hat{\mathbf{x}}^-_{b,i} \ ,\\ \mathbf{P}_{s,i} &= ((\mathbf{P}^+_{f,i})^{-1} + (\mathbf{P}^-_{b,i})^{-1})^{-1} \ .\end{aligned}$$

B.4 Algorithm 3 (FIR-MLS3): Fixed-interval FIR smoother

Detail description of Algorithm 3 from Table 5 in section ??. Perform the following steps:

1. Forward FIR estimation:
For $i = 1, \cdots, N$ execute the forward ML-FIR from Algorithm 1 to obtain estimates $\hat{\mathbf{x}}_{i|y_{m:n}}$ for each point in the fixed interval. Keep the prior and posterior estimates at the end of each finite horizon estimation:

$$\hat{\mathbf{x}}^-_{f,i}, \mathbf{P}^-_{f,i} \ ,$$
$$\hat{\mathbf{x}}^+_{f,i}, \mathbf{P}^+_{f,i} \ .$$

2. Backward FIR smoother estimation:
2.1. Initialize the smoother as follow:

$$\begin{aligned}\hat{\mathbf{x}}_N &= \hat{\mathbf{x}}^+_{f,N} \ ,\\ \mathbf{P}_N &= \mathbf{P}^+_{f,N} \ .\end{aligned}$$

2.2. For $i = N, N-1, \cdots, 1$ executes the smoother estimation:

$$\begin{aligned}
\mathbf{K}_i &= \mathbf{P}_i^+ \mathbf{F}_i^T (\mathbf{P}_{i+1}^-)^{-1} \ , \\
\hat{\mathbf{x}}_i &= \hat{\mathbf{x}}_{f,i}^+ + \mathbf{K}_i (\hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}_{f,i+1}^-) \ , \\
\mathbf{P}_i &= \mathbf{P}_{f,i}^+ - \mathbf{K}_i (\mathbf{P}_{f,i+1}^- - \mathbf{P}_{i+1}) \mathbf{K}_i^T \ .
\end{aligned}$$