COMPUTATIONALLY EFFICIENT VIDEO RESTORATION FOR NYQUIST SAMPLED IMAGING SENSORS COMBINING AN AFFINE-MOTION BASED TEMPORAL KALMAN FILTER AND ADAPTIVE WIENER FILTER

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electrical Engineering

By

Michael Armand Rucci

UNIVERSITY OF DAYTON

Dayton, Ohio

May, 2014
COMPUTATIONALLY EFFICIENT VIDEO RESTORATION FOR NYQUIST SAMPLED IMAGING SENSORS COMBINING AN AFFINE-MOTION BASED TEMPORAL KALMAN FILTER AND ADAPTIVE WIENER FILTER

Name: Rucci, Michael Armand

APPROVED BY:

Russell C. Hardie, Ph.D.
Advisory Committee Chairman
Professor, Electrical and Computer Engineering

Kenneth Barnard, Ph.D.
Committee Member
Principal Electronics Engineer, Air Force Research Laboratory

Eric Balster, Ph.D.
Committee Member
Assistant Professor, Electrical and Computer Engineering

John G. Weber, Ph.D.
Associate Dean
School of Engineering

Tony E. Saliba, Ph.D.
Dean, School of Engineering
& Wilke Distinguished Professor
ABSTRACT

COMPUTATIONALLY EFFICIENT VIDEO RESTORATION FOR NYQUIST SAMPLED IMAGING SENSORS COMBINING AN AFFINE-MOTION BASED TEMPORAL KALMAN FILTER AND ADAPTIVE WIENER FILTER

Name: Rucci, Michael Armand
University of Dayton

Advisor: Dr. Russell C. Hardie

In this thesis we present a computationally efficient video restoration algorithm to address both blur and noise for a Nyquist sampled imaging system. The proposed method utilizes a temporal Kalman filter followed by a correlation-model based spatial adaptive wiener filter (AWF). The Kalman filter employs an affine background motion model and novel process-noise variance estimate. We also propose and demonstrate a new multi-delay temporal Kalman filter designed to more robustly treat local motion. The AWF is a spatial operation that performs deconvolution and adapts to the spatially-varying residual noise left in the Kalman filter stage. In image areas where the temporal Kalman filter is able to provide significant noise reduction, the AWF can be aggressive in its deconvolution. In other areas, where less noise reduction is achieved with the Kalman filter, the AWF balances the deconvolution with spatial noise reduction. In this way, the Kalman filter and AWF work together effectively, but without the computational burden of full joint spatio-temporal processing. We also propose a novel hybrid system that combines a temporal Kalman filter and
BM3D processing. To illustrate the efficacy of the proposed methods, we test the algorithms on both simulated imagery and video collected with a visible camera.
To the memory of my cousin, Nicholas Rucci
ACKNOWLEDGMENTS

First and foremost, I would like to thank my family for their continuing love and support as I worked on my degree.

I would also like to thank Dr. Russell Hardie for all his enthusiasm and willingness to explain concepts and guide me through graduate school. His mentorship was unflagging throughout the time I spent under his tutelage. Finally, I am appreciative to have the chance to work with Dr. Russell Hardie and for the many engaging conversations about algorithm development and MATLAB coding.

Another person I would like to thank is Dr. Kenneth Barnard for always making himself available to answer any of my questions. He also played a key role in helping me set up and perform proper data collections at Wright Patterson Air Force Base. I am truly thankful for the exposure he gave me to many different techniques and systems, and giving me the opportunity to work alongside him.

A big thank you also goes out to all the individuals at AFRL/RYMT for always being there to answer any questions. The time I spent there performing my research was a very enjoyable experience.

Finally, I would to thank my committee for taking time out of their schedules to review my thesis and be part of my defense.
# TABLE OF CONTENTS

ABSTRACT  ................................................................. iii
DEDICATION  ............................................................... v
ACKNOWLEDGMENTS ....................................................... vi
LIST OF FIGURES ........................................................... ix
LIST OF TABLES ............................................................ xi

I. INTRODUCTION ......................................................... 1
   1.1 Object of the Study .............................................. 1
   1.2 Super Resolution ............................................... 2
   1.3 Image Restoration Techniques ................................. 3
   1.4 Significance of the Study ..................................... 3
   1.5 Organization of the Thesis ................................... 5

II. OBSERVATION MODEL .................................................. 6
   2.1 Kalman Filter Model ............................................ 6
   2.2 Noise Model ..................................................... 8
   2.3 PSF Model ....................................................... 9

III. ALGORITHM DESCRIPTION ............................................. 11
   3.1 Kalman Filter Algorithm Description ....................... 12
   3.2 Multi-delay Kalman Filter Algorithm Description ........ 15
   3.3 Adaptive Wiener Filter ....................................... 17
   3.4 Computational Complexity .................................... 20

IV. CAMERA CHARACTERIZATION .......................................... 22
   4.1 Camera Parameters .............................................. 22
4.2 Integrating Sphere ............................................................. 23
4.3 Mean versus Variance ......................................................... 25
4.4 Target Illumination ............................................................. 26
4.5 Radially Symmetric Linear Chirp Test Pattern ......................... 28
   4.5.1 Image Space Calculations ............................................. 28
   4.5.2 Object Space Calculations ............................................ 29
V. EXPERIMENTAL RESULTS ...................................................... 33
   5.1 Simulated Video Data ....................................................... 33
   5.2 Real Video Data ............................................................. 40
VI. CONCLUSIONS ........................................................................ 46
BIBLIOGRAPHY ............................................................................ 47
APPENDICES:
A. Camera Software .............................................................. 50
B. MATLAB Code ................................................................. 51
LIST OF FIGURES

2.1 Observation model used for the Kalman filter development. .......................... 7

2.2 Continuous PSF model of the camera. .......................................................... 10

3.1 Proposed video restoration using affine motion model based temporal Kalman filter followed by AWF. The AWF performs deconvolution and adapts to the spatially varying residual noise variance from the Kalman filter. .................................................. 11

3.2 Detailed block diagram for the temporal Kalman filter. ............................... 14

3.3 Point spread functions for camera and AWF. (a) Impulse invariant PSF of the camera; (b) AWF impulse response with SNR = 0.1; (c) SNR = 36.45; (d) SNR = 400. 19

4.1 Diagram of setup with Integration Sphere. ..................................................... 24

4.2 Camera calculations with varying sphere light levels. .................................. 25

4.3 Diagram of setup with Illumination Source. ................................................ 27

4.4 Diagram outlining variables pertinent to similar triangles. .......................... 30

4.5 Test Target with Ψ = 88.9 mm and ω_{ob} = 0.9986 cyc/mm used in collected imagery. 31

5.1 Imagery generated to test the various algorithms. (a) The original frame; (b) Poisson-Gaussian noise added to the blurred frame with α = 5.90, σ^2_{η_i} = 3222.99, and β = 0. 34

5.2 PSNR results on the filtered simulated imagery. .......................................... 35

5.3 Final frame of the filtered simulated imagery. (a) BM3D-Wiener; (b) VBM3D-Wiener; (c) Kalman-BM3D-Wiener; (d) Kalman-Wiener; (e) Kalman-AWF; and (f) MDKF-AWF. ................................................ 36
5.4 ROI of Kalman-AWF and BM3D-Wiener restored image. BM3D-Wiener results listed as ($\phi$, PSNR). (a) Kalman-AWF result with PSNR = 28.07; (b) (0.25, 24.34); (c) (0.50, 24.51); (d) (0.75, 24.88); (e) (1.05, 25.08); (f) (1.25, 24.87). 39

5.5 ROI of (a) original noisy frame and (b) MDKF-AWF filtering result with fast local motion. A ROI of the local motion object for the (c) Kalman-AWF, (d) and MDKF-AWF. 41

5.6 ROI of (a) original noisy frame and (b) MDKF-AWF filtering result with fast local motion. A ROI of the local motion object for the (c) Kalman-AWF, (d) and MDKF-AWF. 42

5.7 Final frame of the processed outdoor data imagery. (a) Original Frame; (b) BM3D-Wiener; (c) VBM3D-Wiener; (d) Kalman - BM3D-Wiener; (e) Kalman-Wiener; and (f) Kalman-AWF. 44

5.8 Final frame ROI of the processed outdoor data imagery. (a) Original Frame; (b) BM3D-Wiener, $\phi = 1$; (c) BM3D-Wiener, $\phi = 0.6$; (d) Kalman-AWF. 45
LIST OF TABLES

4.1 Camera Parameters during Data Collections ........................................... 23

5.1 Run Times for the 6 Test Algorithms over 100 Frames. .......................... 34
CHAPTER I

INTRODUCTION

1.1 Object of the Study

Video data acquired by imaging sensors are subject to a multitude of degradations affecting both sensitivity and resolution. Invariably, video frames are corrupted by diffraction blur from the optical system as well as noise [1]. In addition, significant atmospheric turbulence may be present which can introduce geometric warping and an additional spatially varying point spread function (PSF). Motion blur can also degrade images. Note that motion blur can be reduced, at the expense of signal-to-noise ratio (SNR), by decreasing the exposure time. Optical aberrations and defocus are an additional potential sources of burring. Noise in imaging sensors comes from a variety of sources. Detector nonuniformity on the focal plane array (FPA) leads to fixed pattern noise. Shot noise and noise from the readout electronics are the main sources of temporally varying noise. Finally, there is the issue of aliasing that may occur if the detector pitch on the FPA is not sufficiently small to meet the Nyquist sampling criterion for the sensor.

The design of imaging sensors for video acquisition involves a complex trade space where the factors above, and others, come into play. One of the main design issues relates to the selection of the optics for a given FPA. The f-number of the optics is defined as the focal length divided by the aperture diameter, and the optical cut-off frequency is inversely proportional to the f-number [2].
Thus, to meet the Nyquist sampling criterion for a given focal length, a sufficiently small aperture is required. In this way, the diffraction PSF serves as an anti-aliasing prefilter before the FPA sampling due to the optical blurring. However, choosing a smaller aperture comes at the expense of SNR, as it limits the light to the FPA. The low SNR problem is exacerbated in low light imaging applications, and/or when employing short exposure times. In many cases, imaging systems are intentionally designed with undersampling in order to reduce the blurring and increase the SNR to balance these factors [3]. However, aliasing can be a performance limiting factor with such systems. Also, with undersampled data, it is not theoretically valid to apply simple deconvolution to the imagery to compensate for the PSF.

1.2 Super Resolution

Multi-frame super-resolution (SR) processing [4] has proven to be effective for restoring imagery from such detector-limited systems. However, SR may require computationally demanding processing with precise subpixel registration. An alternative approach is to design an imaging system that samples at, or very near, the Nyquist rate and then use video restoration to attempt to address the noise and blur to maximize system performance. Here we can exploit the properties of Nyquist sampled imagery to simplify the processing requirements compared with SR that must contest with aliasing. For example, we can theoretically interpolate the imagery for registration and we can employ simple deconvolution at the native sampling grid of the observed data. Furthermore, a small aperture optical system may be desirable as it can be less costly, lighter, and more compact, than a larger aperture system. This is especially true in long focal length systems. However, the noise and diffraction blur are critical factors to address with such systems.
1.3 Image Restoration Techniques

There is a vast wealth of published work on the problem of general image and video denoising. Recently, patch-based image denoising has been a popular approach and this has been explored extensively. The first, and perhaps simplest, patch-based denoising method is nonlocal means (NLM) \cite{5, 6}. Current state of the art image denoising performance can be obtained with more recent patch methods that include the BM3D, V-BM3D, and V-BM4D \cite{7, 8, 9}. These algorithms combine similar 2D image patches into 3D arrays and denoise these 3D arrays through a multistage transform-domain shrinkage process. These methods perform well, but do not exploit any specific scene motion model and they may be computationally demanding.

Since many video sequences exhibit a strong temporal correlation, a variety of other video denoising methods focus specifically on exploiting this. A review of the early video denoising methods like this can be found in \cite{10}. A related approach to video restoration utilizes Kalman filters \cite{11, 12, 13, 14, 15, 16}. Simplified versions of the temporal Kalman filter for denoising are computationally efficient and suitable for real-time field-programmable gate array (FPGA) implementation, as described in \cite{14}. A temporal Kalman filter is combined in parallel with a simple edge-adaptive spatial filter in \cite{13} for denoising. Another two-stage video denoising method uses a motion compensated temporal average followed in series by spatial wavelet denoising \cite{16}.

1.4 Significance of the Study

In this thesis we present a computationally efficient video restoration algorithm to address both blur and noise for a Nyquist sampled imaging system. The proposed method utilizes a temporal Kalman filter followed by a correlation-model based adaptive wiener filter (AWF) \cite{17}. The temporal Kalman filter component is similar to that in \cite{13} and \cite{14}, but with an affine background
motion model and novel process-noise variance estimate. We also propose a new multi-delay temporal Kalman filter designed to more robustly treat local motion. The temporal Kalman filter not only provides an estimate of the noise free current frame, but it also provides an estimate of the residual noise variance remaining in the filtered frame. The AWF we employ is a spatially varying finite impulse response (FIR) Wiener filter [17]. The AWF performs deconvolution and adapts to the spatially-varying residual noise left after the Kalman filter step. In image areas where the temporal Kalman filter is able to provide significant noise reduction, the AWF can be aggressive in its deconvolution. In other areas, where less noise reduction is achieved with the Kalman filter, the AWF balances the deconvolution with spatial noise reduction. A finite number of distinct filter weights are precomputed and used to keep the computational complexity low.

We have elected to perform the restoration in two steps, temporal and then spatial. While joint spatio-temporal approaches are possible, the computational complexity would be greatly increased. We believe that by smartly utilizing the Kalman filter’s residual noise variance estimate to inform the AWF stage, we can approach the performance of a joint spatio-temporal processing, with just a fraction of the computational load. We believe that pairing the Kalman filter an AWF in this fashion is one key novelty of this work. It should be noted that our approach is somewhat related to that in [13] and [16]. However, there are many important differences. For example, note that the spatial filter employed in [13] does not provide deconvolution and is used in parallel with the Kalman filter, not in series. There, the residual noise of the Kalman filter is not used to inform the spatial filter as we do here. The method in [16] is also only for denoising. However, it does inform a spatial denoising filter with a residual noise estimate from a temporal denoising step.

Another key novelty of the current work is that we employ an affine background motion model with our temporal Kalman filter. This global model is powerful in certain applications where camera motion is the dominant source of temporal activity. Since this model has relatively few parameters,
registration can be done accurately, maximizing exploitation of temporal correlation. The affine background motion model used has been shown to be highly effective for multiframe SR in airborne imaging applications [18, 19]. By employing this specific motion model, the resulting algorithm will not be well suited to all video restoration applications. However, in cases where the motion model is appropriate, we expect to be able to outperform more general methods while maintaining a low computational complexity. In addition to the Kalman-AWF method, we also propose and compare a novel hybrid restoration method that combines Kalman filtering, BM3D, and Wiener filtering.

1.5 Organization of the Thesis

The remainder of this thesis is divided into the following chapters. The observation model used to motivate the proposed restoration algorithm is shown in Chapter II. The Kalman-AWF algorithm itself is presented in Chapter III. Camera characterization explaining the set up of the camera itself, the supporting equipment, and test target used in tandem with the camera can be found in Chapter IV. Experimental results for simulated and real video are located in Chapter V. Finally, conclusions are provided in Chapter VI.
CHAPTER II

OBSERVATION MODEL

We present the observation model in this chapter. We begin with the overall model in Chapter 2.1 and then address the noise and PSF models in more depth in Chapters 2.2 and 2.3, respectively.

2.1 Kalman Filter Model

A block diagram of the observation model for the proposed algorithm development is shown in Fig. 2.1. The model begins with the “ideal” frame $n$ image, represented in lexicographical notation as $x_n = [x_{n,1}, x_{n,2}, \ldots, x_{n,N}]^T$. The element $x_{n,i}$ represents ideal pixel $i$ in frame $n$. The PSF blurring is represented by the $N \times N$ matrix $H$ operating on $x_n$ to give $y_n = Hx_n$.

For the purposes of our Kalman filter development, we propose a dynamic process model for the blurred image $y_n$ [20]. The process model is shown in Fig. 2.1 and is defined by

$$y_n = F_n y_{n-1} + q_n,$$  \hspace{1cm} (2.1)

where the $N \times N$ matrix $F_n$ operates on $y_{n-1}$ to perform a warping operation. Here we assume an affine warping, which can be effective at modeling the pixel motion resulting from camera movement [18, 19]. Motion not accounted for by the affine model is treated as additive process “noise”,

6
denoted \( \mathbf{q}_n = [q_{n,1}, q_{n,2}, \ldots, q_{n,N}]^T \). For the Kalman filter, we shall assume that \( \mathbf{q}_n \) is a normal random vector with zero mean and covariance matrix \( \mathbf{Q}_n \) (i.e., \( \mathbf{q}_n \sim N(0, \mathbf{Q}_n) \)). Assuming independent pixels leads to a diagonal covariance matrix: \( \mathbf{Q}_n = \text{diag}(\sigma^2_{q_{n,1}}, \sigma^2_{q_{n,2}}, \ldots, \sigma^2_{q_{n,N}}) \).

The Kalman measurement model, also shown in Fig. 2.1, incorporates the measurement noise. In particular, the Kalman measurement model assumes that the blurred image is corrupted by additive Gaussian noise, so that

\[
\mathbf{z}_n = \mathbf{y}_n + \mathbf{v}_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n,
\]

where \( \mathbf{v}_n \) is an \( N \times 1 \) vector of noise samples. For the purposes of the Kalman filter development, we assume \( \mathbf{v}_n \sim N(0, \mathbf{V}_n) \), where the covariance \( \mathbf{V}_n = \text{diag}(\sigma^2_{v_{n,1}}, \sigma^2_{v_{n,2}}, \ldots, \sigma^2_{v_{n,N}}) \). In this case, the mean of \( \mathbf{z}_n \) is \( \mathbf{y}_n \), and the covariance is \( \mathbf{V}_n \). This simple additive noise model allows us to use the use a linear Kalman filter. However, camera noise may be more accurately described with a
Poisson-Gaussian model [21]. This model and its relationship to the additive measurement noise in Eqn. (2.2) is provided in Chapter 2.2.

The goal of the Kalman filter is to estimate the blurred image, \( y_n \), from \( z_n \). The AWF subsequently performs a deconvolution of the PSF and the estimate \( x_n \). Note that in a typical Kalman filter development, the process model would be used to represent the ideal image and the PSF blurring would be incorporated into the measurement model. However, our model allows us to decouple the spatial deconvolution of the PSF from temporal Kalman filter operation and develop a fast overall restoration algorithm. A similar approach is employed in [22] for SR processing.

2.2 Noise Model

Consider a Poisson-Gaussian noise model that accounts for the photon arrival distribution as well as noise in the electronics [21]. Applying this model, the measurement model from Eqn. (2.2) would be expressed as

\[
\tilde{z}_n = \alpha p_n + \beta 1 + \eta,
\]

where \( p_n \sim P((y_n - \beta 1)/\alpha) \) is an iid Poisson random vector with mean of \( (y_n - \beta 1)/\alpha \). This models the observed signal in the presence of shot noise. The parameter \( \alpha \) is a camera gain, \( \beta \) is a camera offset, and \( 1 \) is an \( N \times 1 \) vector of ones. Since the variance of a Poisson random variable equals its mean, the covariance matrix of \( p_n \) is given by \( P_n = \text{diag} \left( \frac{y_n,1-\beta}{\alpha}, \frac{y_n,2-\beta}{\alpha}, \ldots, \frac{y_n,N-\beta}{\alpha} \right) \).

The vector \( \eta \sim N(0, N) \) is an iid Gaussian random vector modeling the electronics noise terms, where \( N = \text{diag} (\sigma_{\eta,1}^2, \sigma_{\eta,2}^2, \ldots, \sigma_{\eta,N}^2) \). Note that the mean of \( \tilde{z}_n \) in Eqn. (2.3) is \( y_n \), and the covariance is given by \( \tilde{Z}_n = \alpha^2 P_n + N \). Also, note that \( \tilde{Z}_n \) is diagonal and the \( i \)’th diagonal element is given by \( \alpha (y_{n,i} - \beta) + \sigma_{\eta,i}^2 \).

For high mean values, a Poisson distribution is known to be well approximated by a Gaussian. In this case, we can approximate \( \tilde{z}_n \) as a heteroskedastic Gaussian random vector, such that \( \tilde{z}_n \sim \ldots \)
This is equivalent to using (2.2) with $V_n = \tilde{Z}_n$. This means that $\sigma^2_{v_{n,i}} = \alpha(y_{n,i} - \beta) + \sigma^2_{\eta_i}$, for $i = 1, 2, \ldots, N$. Since in practice we do not know $y_{n,i}$, we use the estimate $\sigma^2_{v_{n,i}} = \alpha(z_{n,i} - \beta) + \sigma^2_{\eta_i}$. For the purposes of generating realistic simulated data, we shall employ the model in Eqn. (2.3).

### 2.3 PSF Model

Following the approach in [17, 18], the AWF uses a PSF model that includes diffraction and detector integration. In this case, the optical transfer function (OTF) is given by

$$H(u, v) = H_{\text{diff}}(u, v)H_{\text{det}}(u, v),$$

where $u$ and $v$ are the horizontal and vertical spatial frequencies in cycles per millimeter, $H_{\text{diff}}(u, v)$ is from the diffraction-limited optics, and detector integration is modeled with $H_{\text{det}}(u, v)$. For a circular pupil function we have [2]

$$H_{\text{diff}}(u, v) = \left\{ \begin{array}{ll}
\frac{2}{\pi} \left[ \cos^{-1} \left( \omega \right) - \omega \sqrt{1 - \omega^2} \right] & \omega < 1 \\
0 & \text{else}
\end{array} \right.,$$

where $\omega = \sqrt{u^2 + v^2}/\omega_c$, the optical cut-off frequency is $\omega_c = 1/(\lambda F/#)$, and $F/#$ is the f-number of the optics. The detector OTF, $H_{\text{det}}(u, v)$, is given by the Fourier transform of the active area of a single detector on the FPA. Assuming a rectangular active area, $H_{\text{det}}(u, v)$ will be a sinc function. Note that the continuous PSF is given by the inverse Fourier transform of Eqn. (2.3).

If the detector pitch is given by $p$, then the sampling frequency is $1/p$. Note that the Nyquist sampling theorem requires $1/p > 2\omega_c = 2/(\lambda F/#)$ to guarantee no aliasing with the imaging system. In this case, a discrete impulse invariant version of the PSF can be obtained by sampling the continuous PSF with a sampling interval equal to the detector pitch [23]. This discrete version of the PSF can then be used for deconvolution of the video frames without SR. Let the discrete impulse invariant PSF be denoted $h(n_1, n_2)$. 


The camera utilized for the real video data capture in Chapter V is a Point Grey monochromatic camera, model number FL2-08S2M-C. The visible camera operates within a bandwidth of $\lambda = .4 - .7 \mu m$. The modeling of the PSF uses $\lambda = .55 \mu m$. The camera uses f-16 optics, while $p = 4.65 \mu m$.

It is assumed in the $H_{det}(u, v)$ model for the detectors of the camera to have a 100% fill factor. A slice through the continuous PSF and contributing transfer functions modeled for this camera can be seen in Fig. 2.2.

![Continuous PSF model of the camera.](image)

It is important to note the overall MTF falls within the folding frequency, calculated as $1/p$. Thus the camera system is set up successfully to ensure there is no aliasing in the imagery.
The proposed video restoration algorithm is shown in Fig. 3.1. The temporal Kalman filter operates on the raw frames to produce a reduced noise image estimate for each frame along with an estimate of the residual noise variance. Next, the AWF completes the restoration with an adaptive deconvolution.

Figure 3.1: Proposed video restoration using affine motion model based temporal Kalman filter followed by AWF. The AWF performs deconvolution and adapts to the spatially varying residual noise variance from the Kalman filter.
deconvolution operation that is informed by the residual noise estimate. Let us first address the Kalman filter in detail in Chapters 3.1 and 3.2. The AWF component is addressed in Chapter 3.3. Finally, computational complexity is analyzed in Chapter 3.4.

3.1 Kalman Filter Algorithm Description

The Kalman filter is a recursive filter designed to combine the previous estimate and current observation to provide a minimum mean squared error (MSE) estimate of the underlying process, $y_n$ [20]. The optimality properties of the Kalman filter require that the observation model as given by Eqs. (2.1) and (2.2) are valid and that the process noise, $q_n$, and measurement noise, $v_n$, are Gaussian and independent [20]. Even if these conditions are not strictly met, the Kalman filter is often still a very useful estimator.

Based on the dynamic system model defined by Eqs. (2.1) and (2.2) in Chapter 2.1, the well known linear Kalman filter can be obtained [20]. The Kalman filter process can be thought of as having two steps, the “predict” step and the “correct” step. In the predict step, the current true image is predicted from the last estimate. This is given by

$$\bar{y}_n = F_n \hat{y}_{n-1},$$

where $\bar{y}_n = [\bar{y}_{n,1}, \bar{y}_{n,2}, \ldots, \bar{y}_{n,N}]^T$ is the predicted noise-free current frame, and $\hat{y}_{n-1} = [\hat{y}_{n-1,1}, \hat{y}_{n-1,2}, \ldots, \hat{y}_{n-1,N}]^T$ is the last frame Kalman estimate. Note that $F_n$ performs the affine warping of the last estimate to align it with the current observed frame to form the prediction.

We estimate the affine warping parameters using the robust affine registration method in [19]. The robust method is able to perform background registration with a limited amount of local motion (e.g., 10-20% of the scene). We initialize the prediction image with the current frame at the start of processing. Also, if a sufficiently large registration error is detected, we reset the prediction to be the current frame. The Kalman predict step also involves updating the prediction error covariance
matrix, $\bar{E}_n = \mathbb{E}\{\bar{e}_n\bar{e}_n^T\}$, where $\bar{e}_n = y_n - \bar{y}_n$. This update is given by

$$
\bar{E}_n = F_n E_n^{-1} F_n^T + Q_n,
$$

(3.2)

where $E_n = \mathbb{E}\{e_n e_n^T\}$ and $e_n = y_n - \hat{y}_n$. The prediction error covariance is initialized (and reset), by setting it to $Q_n$.

The correct step of the Kalman filter can be defined with the following three equations. First, we get the Kalman gain matrix

$$
K_n = \bar{E}_n (\bar{E}_n + V_n)^{-1}.
$$

(3.3)

Next, the Kalman estimate of the current frame, combining the prediction and current frame, is given by

$$
\hat{y}_n = (I - K_n) \bar{y}_n + K_n z_n.
$$

(3.4)

Finally, the error covariance is updated as

$$
E_n = (I - K_n) \bar{E}_n = \bar{E}_n - K_n \bar{E}_n.
$$

(3.5)

Treating all of the covariance matrices as diagonal, yields a simplified pixel-wise form of the Kalman filter equations. This allows for a computationally efficient implementation. Let the diagonal elements of $\bar{E}_n$ be denoted $\text{diag}(\bar{E}_n) = \bar{\epsilon}_n = [\bar{\epsilon}_{n,1}, \bar{\epsilon}_{n,2}, \ldots, \bar{\epsilon}_{n,N}]^T$ and the diagonal elements of $E_n$ be $\text{diag}(E_n) = \epsilon_n = [\epsilon_{n,1}, \epsilon_{n,2}, \ldots, \epsilon_{n,N}]^T$. Now after implementing the affine warping of Eqn. (3.1), the prediction error diagonal covariance elements are updated as follows

$$
\bar{\epsilon}_n = F_n \epsilon_{n-1} + \tilde{q}_n,
$$

(3.6)

where $\tilde{q}_n = [\sigma_{q_n,1}^2, \sigma_{q_n,2}^2, \ldots, \sigma_{q_n,N}^2]^T$. Note that this amounts to an affine warping of the last frame Kalman error variances, to align with the current frame, and then adding the current frame process noise variances.
The simplified pixel-wise correct step equations are given by

\[ k_{n,i} = \frac{\bar{e}_{n,i}}{\bar{e}_{n,i} + \sigma^2 v_{n,i}}, \]  

(3.7)

\[ \hat{y}_{n,i} = (1 - k_{n,i}) \bar{y}_{n,i} + k_{n,i} z_{n,i}, \]  

(3.8)

and

\[ e_{n,i} = (1 - k_{n,i}) \bar{e}_{n,i}, \]  

(3.9)

for \( i = 1, 2, \ldots, N \). Note that the Kalman gain in Eqn. (3.7) balances the uncertainty in the prediction and the uncertainty in the current measurement. The Kalman estimate in Eqn. (3.8) uses that information to form a weighted sum of the observed current frame pixels and the corresponding predicted values. The details of the predict and the correct steps are shown in the block diagram in Fig. 3.2.

Figure 3.2: Detailed block diagram for the temporal Kalman filter.
Lastly, let us consider the estimation of the process noise variance. In [13] and [24], the process noise variance is estimated by the differences of the matched block average values between the current frame and previous frame. Here we use the difference between the current frame and the motion-compensated prior estimate. We employ a Gaussian low pass filter on this difference to reduce the impact of noise. This provides an estimate of the process noise at each pixel. This is given by

\[ \hat{q}_n = G (z_n - \bar{y}_n), \]

where \( G \) performs a Gaussian low pass filter operation and \( \hat{q}_n = [\hat{q}_{n,1}, \hat{q}_{n,2}, \ldots, \hat{q}_{n,N}]^T \). The variance of the process noise is estimated as follows

\[ \hat{\sigma}^2_{q_n,i} = \kappa \left( \frac{1}{|W_i|} \sum_{j \in W_i} \hat{q}_{n,j}^2 \right)^\gamma. \]

The set \( W_i \) defines a rectangular window about pixel \( i \) and the parameters \( \kappa \) and \( \gamma \) are tuning parameters that allow us to adjust the process noise variance estimate to help balance moving object detail preservation and background noise suppression. In our experimental results \( \kappa \) ranges from 1 to 2, and \( \gamma \) varies from 1.5 to 4. The Gaussian kernel has standard deviation of 1.5, while \( W_i \) represents a \( 3 \times 3 \) window about pixel \( i \).

### 3.2 Multi-delay Kalman Filter Algorithm Description

In the Kalman filter, when the process noise is large due to the presence of a moving object, the Kalman estimate gives more weight to the observed frame. Thus, we have increased residual noise in those areas. This increased residual noise forms a “noise trail” behind some fast moving objects that dissipates over time. The goal of the mutli-delay Kalman filter (MDKF) is to reduce these noise trails by incorporating more than one previous estimate into the current Kalman estimate.
To define the MDKF, let us define $\mathbf{m}_J = [m_1, m_2, \ldots, m_J]$ to be an array of $J$ delay values. The Kalman prediction step now uses prior estimates for all of these delays as follows

$$\bar{y}_n(m_j) = F_{n,m_j} \hat{y}_{n-m_j}, \quad (3.12)$$

where $F_{n,m_j}$ is the affine transformation mapping each estimated delayed image to the current frame. The prediction error is updated for each delay as

$$\bar{\epsilon}_n(m_j) = F_{n,m_j} \epsilon_{n-m_j} + \tilde{q}_n. \quad (3.13)$$

A Kalman gain is defined for each delay yielding

$$k_{n,i}(m_j) = \frac{\bar{\epsilon}_{n,i}(m_j)}{\bar{\epsilon}_{n,i}(m_j) + \sigma^2_{\epsilon_{n,i}}}. \quad (3.14)$$

Multiple estimates are formed, using each delay, yielding

$$\hat{y}_{n,i}(m_j) = (1 - k_{n,i}(m_j)) \bar{y}_{n,i}(m_j) + k_{n,i}(m_j) z_{n,i}. \quad (3.15)$$

The error variance for each estimate is

$$\epsilon_{n,i}(m_j) = (1 - k_{n,i}(m_j)) \bar{\epsilon}_{n,i}(m_j). \quad (3.16)$$

The final estimate frame is created with the help of all the $\epsilon_{n,i}(m_j)$ values. Let $m_{n,i}$ be the delay value for frame $n$ pixel $i$ with the lowest error variance. This is given by

$$m_{n,i} = \arg\min_{m \in \mathbf{m}_J} [\epsilon_{n,i}(m)]. \quad (3.17)$$

The final image estimate and noise variance estimate are then computed as follows

$$\hat{y}_{n,i} = \hat{y}_{n,i}(m_{n,i}), \quad (3.18)$$

and

$$\epsilon_{n,i} = \epsilon_{n,i}(m_{n,i}). \quad (3.19)$$
Note that the MDKF requires more processing and memory than the standard Kalman filter. The number of delays, \( J \), is a factor that governs both speed and memory requirements for the MDKF. The noise reduction on the trail of local motion objects is generally improved with the multi-delay Kalman filter using appropriate delay values.

### 3.3 Adaptive Wiener Filter

With the noise reduced Kalman estimate in hand, the AWF now addresses the PSF blur in the presence of the residual noise. The AWF is designed assuming the input is given by

\[
\hat{y}_n = Hx_n + e_n. \tag{3.20}
\]

The variance of the residual Kalman filter error, \( e_n \), is spatially varying. The Kalman filter provides an estimate of those variances with \( \varepsilon_n \). The impulse invariant PSF, \( h(n_1, n_2) \), is assumed to be known and this is represented in Eqn. (3.20) as the matrix \( H \). The correlation-model based AWF we employ is similar to that in [17], but the filter design and implementation here is simpler because we are treating only one frame at a time and are not upsampling the imagery.

The AWF uses a 2D finite moving observation window. Each output pixel is formed as a weighted sum of pixels in the observation window centered about the pixel. Let the \( M \) samples in the observation vector centered about pixel \( i \) in frame \( n \) be denoted by the \( M \times 1 \) vector \( \tilde{y}_n[i] \).

The AWF estimate is then given by

\[
\hat{x}_{n,i} = w_{n,i}^T \tilde{y}_n[i], \tag{3.21}
\]

where \( w_{n,i} = [w_{n,i,1}, w_{n,i,2}, \ldots, w_{n,i,M}]^T \) contains the filter weights for frame \( n \) pixel \( i \). The minimum MSE weights are given by

\[
w_{n,i} = R_{n,i}^{-1} p_{n,i}, \tag{3.22}
\]

where

\[
R_{n,i} = E\{\hat{y}_n[i]\hat{y}_n[i]^T\} \tag{3.23}
\]
and

\[ p_{n,i} = E\{x_{n,i} \hat{y}_n[i]\}. \] (3.24)

The correlation values in Eqs. (3.23) and (3.24) are populated using a correlation model in a manner similar to that described in [17]. The underlying 2D discrete correlation model for the ideal image, \( x_n \) is given by

\[ r_{xx}(n_1, n_2) = \sigma^2_x \rho \sqrt{n_1^2 + n_2^2}, \] (3.25)

where \( \sigma^2_x \) is the assumed signal variance and \( \rho \) is the one step correlation coefficient [17]. From this and the PSF, the 2D cross correlation between \( x_n \) and \( \hat{y}_n \) can be obtained [17]. If the residual noise variance, \( \epsilon_n \) is also given, the 2D autocorrelation for \( \hat{y}_n \) can be readily obtained [17]. These 2D correlation functions are sampled according to the spatial positions of the samples in \( \hat{y}_n[i] \) to populate Eqs. (3.23) and (3.24).

According to this correlation model, the only thing that changes from one pixel to the next is the residual noise variances that change the assumed local signal-to-noise ratio (SNR). However, this still requires that the computation of distinct weights in Eqn. (3.22) for each pixel. In order to dramatically reduce the computational burden, we precompute \( K \) sets of filter weights to accommodate the expected range of local SNRs before processing frames. A sample of these precomputed weights, along with the PSF of the camera used in the experimental results chapter, can be found in Fig. 3.3. Note that with a low SNR, Fig. 3.3(b), the AWF appears similar to a Gaussian low pass filter. For a high SNR, Fig. 3.3(d), the AWF becomes an aggressive deconvolution filter. When processing frames, we average the residual noise variance values over the span of the filter window and use a constant signal variance value. The resulting SNR is computed and quantized to the nearest SNR of the precomputed weights. The precomputed weights are then applied to form the AWF estimate for the given pixel. Since the weights are precomputed, this has approximately the same computational complexity as an \( M \) sample fixed FIR filter.
Figure 3.3: Point spread functions for camera and AWF. (a) Impulse invariant PSF of the camera; (b) AWF impulse response with SNR = 0.1; (c) SNR = 36.45; (d) SNR = 400.
3.4 Computational Complexity

In this chapter we consider some of the implementation details and provide an approximate computational complexity analysis by counting the floating point operations (flops) required to compute one complete processed frame. We define one flop as a combination of a floating point multiply and add operation.

For registration, we employ the robust registration method described in [19]. It is a gradient based least squares method applied using multiple Gaussian pyramid scales and is iterative at each scale. The accuracy of the method comes from the fact that it uses the vast majority of pixels to obtain the background affine motion estimates, not just a limited number of landmark pixels. Also, local motion is detected at each pyramid level and these pixels are excluded in the subsequent scale. This gives the method essential robustness to limited local motion. Here we employ a Gaussian pyramid with 3 levels. For speed enhancement, we make the highest resolution one half the scale of the native imagery. We use 5 iterations at each level. The main computational burden comes from the warping operations, of which there are a total of 15. However, these occur at reduced image scales. We use simple bilinear warping during registration.

Let us now consider the Kalman filter. In practice, the affine warping of the prior estimate in Eqn. (3.1) is accomplished in a three step procedure. Note that the matrix $F_n$ is sparse, and it would be highly inefficient to directly compute the matrix multiplications using $F_n$. To perform the warping, we first upsample the prior estimate image by a factor of 3 in each dimension using a Lanczos interpolation kernel. This requires 14 flops per output sample and there are $9N$ output samples. Next, the affine warped coordinates must be computed which requires $6N$ flops. Finally, this is followed by bilinear interpolation requiring approximately $8N$ flops. When we warp the error variance in Equation (3.6), preserving resolution is less critical and we do not use the Lanczos upsampling. Since the affine coordinates are already computed, this step takes only $8N$ flops. In
order to estimate the process noise variance, a Gaussian low pass filter and moving average filter are employed. The Gaussian has a standard deviation of 1.5 pixels. Using a separable $7 \times 7$ kernel, this requires $14N$ flops. In addition, a separable $3 \times 3$ moving average filter is used, adding $6N$ flops. The Kalman “correct” step requires approximately $4N$ flops, as can be seen in Fig. 3.2.

The first step in the AWF is to estimate the local SNR. As mentioned in Chapter 3.3, we assume a constant signal variance, and the local noise variance is obtained by applying a Gaussian low pass filter to the Kalman residual error variance. We employ a separable Gaussian filter with a standard deviation of 3 pixels for this purpose. This requires approximately $26N$ flops using a $13 \times 13$ window. The actual AWF can be implemented with a spatially varying finite impulse response (FIR) filter, by directly employing Eqn. (3.21). It has been shown small kernels with properly chosen weights can achieve desired restoration results [25]. We have found that using $9 \times 9$ filters are effective for our purposes. This requires $81N$ flops, plus a considerable amount of memory accesses, since the filters vary from pixel to pixel. We have found that memory access tends to be the limiting factor in processing speed in our MATLAB implementation. For this reason, we employ a fast Fourier transform (FFT) implementation of the AWF. Here we use $K = 8$ filters and generate $K$ complete outputs and then select among these according to the local SNR to form the final output. To do so we must perform a 2-D FFT on the Kalman filter processed frame, $KN$ products to compute the filtered images, and then $K$ inverse 2-D FFTs. Using row column decomposition, we must compute twice as many 1-D FFTs. Using the flop count analysis from [26] this amounts to

$$\frac{34(K+1)}{9}2N \log_2 \sqrt{N}$$

(3.26) flops. While the FFT approach generally has more flops than the FIR approach, we found it about 4 times faster. We believe this is due to memory access delays with the FIR version. Using an optimized implementation, the FIR version may prove to be the fastest, as the raw flop count suggests.

21
CHAPTER IV

CAMERA CHARACTERIZATION

4.1 Camera Parameters

The camera used in the data collects was connected to a Linux machine via a fireware. FlyCap2 was the software originally supporting the camera and provided by Point Grey. However, the software was not sufficient to grab more than one frame at a time. Additionally multiple camera settings were unable to be modified. The FlyCap2 software was then replaced with a freeware program named Coriander. The link to the website can be located in the APPENDIX. Coriander allowes full control over the camera parameters, along with being able to capture sequences of frames. All the data collects consists of many parameters remaining the same throughout all the data collections. A table has been generated below listing all the parameters and values.

The Frame Rate and Shutter time in Table 4.1 varied within the data collects. The images captured with the camera were saved as 16 bit raw format. It is interesting to note the camera itself has a 12 bit analog to digital converter. A look into the histogram of a captured image shows the 12 bit data is multiplied by a scalar value of 16 to create 16 bit data.
Table 4.1: Camera Parameters during Data Collections

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brightness</td>
<td>0</td>
</tr>
<tr>
<td>Auto Exposure</td>
<td>Off</td>
</tr>
<tr>
<td>Shutter</td>
<td>1 or 10 ms</td>
</tr>
<tr>
<td>Gain</td>
<td>0</td>
</tr>
<tr>
<td>Gamma</td>
<td>1</td>
</tr>
<tr>
<td>Trigger Delay</td>
<td>Off</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>15 or 30 fps</td>
</tr>
</tbody>
</table>

4.2 Integrating Sphere

In order for the Kalman filter to properly perform the denoising of the imagery, the variance for each pixel of the imagery must be known. As this algorithm is to be implemented in real time the variance for each pixel must be calculated for each incoming frame. We can use a poisson distribution model for the process of obtaining a pixel value. Thus with the assumed model there is a direct linear equation between the mean and variance value for each pixel. An Integrating Sphere is used to find the mean value and variance for varying light levels. The Integrating Sphere used in these experiments is a 12 inch diameter Labsphere Integrating Sphere, model number US-120-SF. The light source accompanying the sphere is a Quartz Tungsten Halogen (QTH) bulb. The amount of light entering the sphere is controlled by a motorized aperture. A diagram showcasing how the camera is positioned within the sphere shroud is shown below in Fig. 4.1.

A fabric sheet is also wrapped around the shroud entrance and camera tripod to ensure no outside light could enter the sphere. The aperture is opened in 2% increments from 0% until the camera was saturated. One hundred frames are captured and stored for further processing within Python.
Figure 4.1: Diagram of setup with Integration Sphere.
4.3 Mean versus Variance

The camera is set up with the parameters outlined in Table 4.1 with 12.5 mm lens and a f-number of 16. The frame rate was set to 15, while the shutter time was 10 ms. Each set of 100 frames per the varying light level in the sphere are flattened out into an array. The mean and variance are then computed on this array. The results from this test are shown below in Fig. (4.2). The camera data above contains points that were not used in the calculation for the best fit line. The

![Figure 4.2: Camera calculations with varying sphere light levels.](image)

four highest mean points contain lower variances due to the fact the camera was saturated at the
higher light levels. It is also important to note data points on the left hand side of the plot were also
not used since they began to move from signal dependent noise to electronic noise. The best fit line
yielded the line equation,

$$\sigma^2 = 3.9609\mu + 7907.8.$$  \hspace{2cm} (4.1)

This equation is used later on in Chapter 5.2 to produce the results shown in Fig. (5.5) and Fig.
(5.6).

The other variance calculation based on the mean equation used to produce the results in Fig.
(5.7) is calculated differently. In this data set the noise model did not match the results calculated
using the integrating sphere. Thus a temporal average through a static data set was taken. The mean
values were sorted in ascending order. Then a best fit line was applied to the sorted means and their
respective variances. This yielded the equation,

$$\sigma^2 = 5.90\mu + 3222.99.$$  \hspace{2cm} (4.2)

The integrating sphere results did in fact vary from the scene calculated noise model. A reason
for this may be due to the electromagnetic interface persistent at the elevated height position of the
camera.

4.4 Target Illumination

In the data collection utilizing the RSLC test pattern, Fig. 4.5, an additional light source is
needed to illuminate the target. The light source used during testing is a 60100 Series Newport/Oriel
Monochromator Illuminator with a QTH 100 watt bulb. The amount illumination is controlled via
a power supply regulating the voltage to the bulb. In the imagery show in Fig. 5.5 and Fig. 5.6
the voltage to the lamp was set at 8 volts. A diagram showing the set up between the camera,
light source, and target are illustrated in Fig. 4.3. The optical axis for both the camera lenses and
Illuminator lenses are on the same vertical axis. This is done to provided uniform illumination of the
Figure 4.3: Diagram of setup with Illumination Source.
target. The additional light added from the Illuminator did cause some specular reflection within the paper and ink the test pattern was printed on. Thus different quality paper and ink are used that diffuse the light and reduce the specular reflection.

4.5 Radially Symmetric Linear Chirp Test Pattern

This chapter discusses a radially symmetric linear chirp (RSLC) test target imaged by the visible camera in multiple data collects. A RSLC is a test target that has a linearly increasing sweep in the spatial frequency. The spatial frequency moves from a DC value to a predetermined cut-off spatial frequency. This is ideal with a Nyquist sampled imaging sensor. In order to fully test the camera system, we push the cut-off spatial frequency of the test target past the optical cut-off frequency. In doing so, it enables the restoration algorithm to try and improve all the spatial frequencies that could be present in an image.

4.5.1 Image Space Calculations

The calculations for the RSLC will be done in the image space as the optical cut-off frequency of the optics dictates the highest spatial frequency seen by the focal plane array. The equation used to generate the RSLC can be found in Eqn.(4.3)

\[ f(\psi) = \cos(\pi k \psi^2), \]  

(4.3)

where \( k \) is a scalar with its equation shown later and \( \psi \) is the radial coordinate calculated as

\[ \psi^2 = x^2 + y^2. \]  

(4.4)

where

\[ x, y \in [-\Psi, \Psi]. \]  

(4.5)

It is important to note that \( \Psi \) controls the length, typically in millimeters, the RSLC pattern has to go from DC to the spatial cut-off frequency. The instantaneous frequency of Eqn.(4.3) is calculated
by taking a derivative of the terms within the cosine function with respect to $\psi$, yielding

$$\frac{d}{d\psi}(\pi k\psi^2) = 2\pi k\psi. \quad (4.6)$$

While, the frequency at the maximum radius of the RSLC is

$$k\psi = \omega_{max} \quad (4.7)$$

with

$$\omega_{max} = 1.25\omega_c \quad (4.8)$$

and

$$\omega_c = 1/(\lambda F/\#). \quad (4.9)$$

Please note $\omega_c$ is the optical cut-off frequency and $\omega_{max}$ is a scaled quantity of the cut-off frequency. The multiplication of 1.25 is done to ensure the spatial frequencies on the RSLC exceed the optical cut-off frequency. Also, $k$ enables the RSLC to reach the specified spatial cut-off frequency in the length, $\Psi$, allowed.

### 4.5.2 Object Space Calculations

The cut-off spatial frequency between the physical RSLC pattern in object space and the focal plane in image space vary due to the distance between them. In order to ensure the proper spatial frequencies are seen by the focal plane, we will use a simple ratio called similar triangles. The setup used with similar triangles is shown below in Fig. 4.4 The actual equation of similar triangles is given as

$$\frac{\Delta x_{ob}}{l} = \frac{\Delta x_{im}}{l'}, \quad (4.10)$$

having $l$ represent the distance between the object and the lens and $l'$ being the focal length of the lens. The $\Delta x_{im}$ can represent either a single detector or the entire focal plane array along one dimension. Finally, the footprint of $\Delta x_{im}$ in the object space is $\Delta x_{ob}$. Since we can calculate the
frequency at any point on the RSLC, the period of the frequency is also known to us. The period of any frequency, \( \omega_{ob} \), in the object space, \( \Delta x_{ob} \), is

\[
\Delta x_{ob} = \frac{1}{\omega_{ob}}. \tag{4.11}
\]

Similarly the period for the same frequency in the image space, \( \omega_{im} \), is

\[
\Delta x_{im} = \frac{1}{\omega_{im}}. \tag{4.12}
\]

The direct relation between \( \omega_{ob} \) and \( \omega_{im} \) can be found by plugging Eqn.(4.11) and Eqn.(4.12) into Eqn.(4.10), resulting in

\[
\omega_{ob} = \frac{l'}{l} \omega_{im}. \tag{4.13}
\]

Now the desired spatial cut-off frequency in the image space was calculated in Eqn.(4.8). Instinctively this allows

\[
\omega_{ob} = \frac{l'}{l} \frac{1.25}{\lambda F/\#} \tag{4.14}
\]

and now all that is left is to solve for the scalar value of \( k \). The equation is

\[
k = \frac{\omega_{ob}}{\Psi}, \tag{4.15}
\]

allowing the RLSC to reach the required spatial cut-off frequency in the designated target radius size. The printed RLSC target must have a diameter equal to two times \( \Psi \). An image of the actual
Figure 4.5: Test Target with $\Psi = 88.9$ mm and $\omega_{ob} = 0.9986$ cyc/mm used in collected imagery.
test pattern used in the imagery shown in Chapter 5.2, is shown below in Fig.(4.5). The black border
around the letters was inserted to help with the registration of the collected frames. Letters were
also added to aid with registration and allow a variation in grayscale. A calculation of the RSLC
test pattern used in the captured image is available in APPENDIX.
CHAPTER V

EXPERIMENTAL RESULTS

In this chapter, we present results using simulated data for quantitative performance analysis. We also present results using real data collected from a tower using a visible camera for subjective analysis.

5.1 Simulated Video Data

The first simulated data set is created using a single aerial image shown in Fig. 5.1(a). The image shows a theme park with a parking lot in the top center and trees and shrubs along the bottom. Each simulated frame contains global affine motion with translation and rotation to simulate camera motion. Local motion is introduced in the form of a synthetic moving rectangular patch containing a chirp pattern. The chirp pattern moves horizontally back and fourth through the video sequence. Blur is introduced using the PSF shown in Fig. 3.3(a). Finally, Poisson-Gaussian noise is introduced. The progression between original and blurred/noisy can be seen in Fig. 5.1. The simulated data set is designed to approximate the application of airborne imaging where affine motion dominates due to camera motion, and some local motion is also present [18, 19]. Noise characteristics, along with the signal dynamic range of the imagery, are designed to match the video sensor data used for the real image results. For the simulation results, 100 frames are generated and processed in MATLAB on a PC with a Core i7 CPU @3.07 GHz, with 8GB of RAM. The size of
Figure 5.1: Imagery generated to test the various algorithms. (a) The original frame; (b) Poisson-Gaussian noise added to the blurred frame with $\alpha = 5.90$, $\sigma^2_i = 3222.99$, and $\beta = 0$.

the images is $380 \times 380$ pixels. In total, 6 restoration algorithms are tested here. The peak signal-to-noise-ratio (PSNR) for each method over the 100 frame sequence is plotted in Fig. 5.2. The final image for each of the methods is shown in Fig. 5.3 and the processing times are listed in Table 5.1.

Table 5.1: Run Times for the 6 Test Algorithms over 100 Frames.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Denoising</td>
</tr>
<tr>
<td>BM3D-Wiener</td>
<td>252.53</td>
</tr>
<tr>
<td>VBM3D-Wiener</td>
<td>52.73</td>
</tr>
<tr>
<td>Kalman-BM3D-Wiener</td>
<td>263.90</td>
</tr>
<tr>
<td>Kalman-Wiener</td>
<td>15.52</td>
</tr>
<tr>
<td>Kalman-AWF</td>
<td>15.71</td>
</tr>
<tr>
<td>MDKF-AWF</td>
<td>32.41</td>
</tr>
</tbody>
</table>
Figure 5.2: PSNR results on the filtered simulated imagery.
Figure 5.3: Final frame of the filtered simulated imagery. (a) BM3D-Wiener; (b) VBM3D-Wiener; (c) Kalman-BM3D-Wiener; (d) Kalman-Wiener; (e) Kalman-AWF; and (f) MDKF-AWF.
The benchmark algorithms are based on BM3D and VBM3D, as these provide state-of-the-art performance for general image/video denoising [7, 8]. Since we have signal dependent noise, we employ both a generalized and an exact unbiased inverse Anscombe variance stabilization transformation as described in [27] for only the VBM3D and BM3D methods. Both filters also have their theoretical noise standard deviation, $\sigma_T$, modified by a scalar, $\phi$, to control their operating point. The result yields what we call the assumed noise standard deviation, calculated as

$$\sigma_A = \phi \sigma_T.$$  \hspace{1cm} (5.1)

Thus when $\phi$ equals one the theoretical noise standard deviation is used in the filters. However if $\phi$ is less than one then the filter does less noise suppression, while greater than one increases the noise reduction. This is done to help facilitate comparable noise suppression to that of the Kalman filter. To provide deconvolution, the benchmark methods are followed by a Wiener filter. Unless otherwise specified, the $\phi$ and Wiener noise-to-signal ratio (NSR) that produce the highest PSNR are used in the simulated data for each of the benchmark methods. Note that BM3D operates on each frame independently. VBM3D operates on the video, but does not use knowledge of any particular background motion model. It is also important to note that VBM3D operates on the frames in a non-causal manner. These benchmark methods are designated BM3D-Wiener and VBM3D-Wiener.

Another of the methods tested is a novel hybrid method that combines the temporal Kalman filter from Chapter 3.1 with BM3D. The hybrid algorithm processes the imagery with both BM3D and the temporal Kalman filter. These two resulting images are alpha-blended based on the residual error variance from the Kalman filter. The local pixel blending parameter gives a weight of 1 to the Kalman filter output and 0 to the BM3D output where the Kalman residual error variance is zero (i.e., complete noise reduction). If the residual error variance equals the noise variance (i.e., no Kalman noise reduction), the blending weight is 1 for the BM3D output and 0 for the Kalman filter output. The blending weight is linearly mapped in between these two extremes. The blended
image is then processed by a constant NSR Wiener filter to provide the final output. Note that this approach combines the best of Kalman filter and BM3D. The Kalman filter generally does well on the background and the gives a small residual error variance there. For areas with local motion, less temporal correlation is present. Here, the BM3D can exploit the spatial correlations. This method is designated Kalman-BM3D-Wiener.

The method designated as Kalman-Wiener uses the temporal Kalman filter from Chapter 3.1 followed by a constant NSR Wiener filter for simplicity. The global Wiener filter is once again optimized with regards to the NSR value. The method denoted Kalman-AWF pairs the temporal Kalman filter from Chapter 3.1 with the AWF in Chapter 3.3. Finally, the method denoted MDKF-AWF uses the multi-delay Kalman filter from Chapter 3.2 and the AWF in Chapter 3.3.

As shown in Fig. 5.2, all of the methods significantly increase the PSNR of the observed noisy data. The restoration method with the highest PSNR is Kalman-BM3D-Wiener. However, it has the highest processing time, as shown in Table 5.1. The benchmark methods, BM3D-Wiener and VBM3D-Wiener, do not perform as well as the Kalman filter based methods on this dataset. We attribute this to the fact that the benchmark methods do not take advantage of a global motion model, which is dominant in this dataset. The fastest algorithm tested is the Kalman-Wiener, as shown in Table 5.1. However, this method is outperformed by the Kalman-AWF and MDKF-AWF, since the AWF is able to adapt to the spatially varying residual error from the Kalman filter stage. Note that here the MDKF-AWF uses \( m_2 = [1, 13]^T \) and has a correspondingly higher processing time than the Kalman-AWF.

Examining the image results in Fig. 5.3 shows the qualitative differences among the methods. In Figs. 5.3(a) and (b) we see that BM3D-Wiener and VBM3D-Wiener provide strong noise reduction, but background detail is lost. For example, note the loss in texture in the parking lot and the trees and shrubs. The Kalman-BM3D-Wiener output, shown in Fig. 5.3(c) has much better background
detail as a result of blending with the Kalman filter output. The moving object shows good noise suppression, thanks in large part to the BM3D processing. The Kalman-Wiener output in Fig. 5.3(d), also appears do well on the background. However, the fixed Wiener filter amplifies the noise on the moving object. The Kalman-AWF output in Fig. 5.3(e) has better performance on the moving object. The MDKF-AWF output appears very similar to the Kalman-AWF in this dataset. The benefits of the MDKF tend to be observed with fast local motion. This will be illustrated in the real video results in Chapter 5.2. Figure 5.4 compares the Kalman-AWF output to the BM3D-

![Image](a) (b) (c) (d) (e) (f)

Figure 5.4: ROI of Kalman-AWF and BM3D-Wiener restored image. BM3D-Wiener results listed as ($\phi$, PSNR). (a) Kalman-AWF result with PSNR = 28.07; (b) (0.25, 24.34); (c) (0.50, 24.51); (d) (0.75, 24.88); (e) (1.05, 25.08); (f) (1.25, 24.87).
Wiener method using a variety of assumed noise standard deviations. Note that the noise scaling parameter that yields the highest PSNR for BM3D-Wiener is $\phi = 1.05$ and has a PSNR of 25.08. The theoretical noise standard deviation produces nearly identical results with a PSNR of 25.07. Note in Fig. 5.4 that as $\phi$ is increased, we see more noise reduction as well as loss of detail. Even for $\phi = 0.25$ in Fig. 5.4(b), it appears that we have less detail preservation in BM3D-Wiener than in Kalman-AWF. Those loss of detail is very evident for the optimum PSNR $\phi$ in Fig. 5.4(e).

### 5.2 Real Video Data

The camera utilized for the real video data capture operates at 15 or 30 frames per second. We have selected F/16 optics to allow for Nyquist sampling.

Two data sets are used here to illustrate different aspects of the restoration algorithms. The first is an indoor data set of a 2D chirp pattern with a moving pendulum. The scene is illuminated with a light source to ensure enough signal is present, while the camera has a 12.5 mm lens. The second dataset shows a more realistic application where the camera is placed outdoors in a tower looking down at the target area, with a 25 mm lens.

The results from the indoor data sequence, at 30 fps, in Fig. 5.5 show the noise and blur reduction for a large portion of the image, along with the Kalman-AWF and MDKF-AWF results. A raw frame is shown in Fig. 5.5(a), and the denoised MDKF-AWF frame in Fig. 5.5(b), using $m_2 = [1, 5]^T$. Note that using more delays demonstrates the ability to greatly reduce the noise and blur within the imagery. A specific point of focus is the increase in rings visible within the restored image. The other feature of interest is the pendulum with ruler markings on it, clearly visible in all four images of Fig. 5.5. The Kalman-AWF struggles to reduce the trail of noise following the left edge of the pendulum, as shown in Fig. 5.5(c). On the other hand, MDKF-AWF is able to better resolve the underlying background pixel values presented in Fig. 5.5(d).
Figure 5.5: ROI of (a) original noisy frame and (b) MDKF-AWF filtering result with fast local motion. A ROI of the local motion object for the (c) Kalman-AWF, (d) and MDKF-AWF.
A similar test pattern was also used with faster local motion better demonstrate the ability of the MDKF-AWF. Here, the camera was equipped with a 12.5 mm lens and was ran at 15 fps. The Kalman-AWF struggles to remove the noise trail left behind by the fast motion where the pendulum seems to jump, rather than slowly sweep across, the collected imagery. Thus it is clear to see where the pendulum was in the previous frames due to the residual echoes in Fig. 5.6(a). The MDKF-AWF, Fig. 5.6(b), using \( \mathbf{m}_2 = [1, 5]^T \) is clearly able to remove the residual echoes in the restored imagery.

The image results for the second dataset, with the camera operating at 30 fps, are shown in Fig. 5.7. A raw noisy frame is shown in Fig. 5.7(a). The BM3D-Wiener and VBM3D-Wiener results are shown in Fig. 5.7(b) and Fig. 5.7(c), respectively. These two techniques have their NSR chosen based on aesthetics of the final denoised/deblurred imagery. Note that the small detail within the brickwork is smoothed out despite its redundant pattern within the collected imagery. It is
important to note however, that the moving object within the final restored frame appears relatively well restored.

The Kalman-BM3D-Wiener output is shown in Fig. 5.7(d). As expected, the detail in the denoised frame is improved due to the noise reduction and AWF. The local motion within the frame still retains its information with the aid of BM3D.

Lastly, the Kalman-Wiener and Kalman AWF outputs are shown in Figs. 5.7(e) and (f), respectively. The moving object is not as nicely restored as the Kalman-BM3D-Wiener, but the detail in the background of the imagery is still there. The AWF is relied upon entirely to provide noise reduction on the moving object and the trail behind it as seen in Fig. 5.7(f). Whereas, the fixed Wiener filter does no smoothing to the moving object, as shown in 5.7(e). Overall, temporal redundancy is well exploited by the Kalman filter to allow fine details in the background to be well preserved.

Specific points of focus are the brickwork in the bottom left hand corner of the images shown in Fig. 5.7(d), (e), and (f), along with the chain link fence visible after restoration located in the upper left hand corner of the imagery.

ROI’s of some of the results in Fig. 5.7 are shown in Fig. 5.8. The raw noisy frame is shown in Fig. 5.8(a), while the BM3D-Wiener result is shown in Fig. 5.8(b). BM3D-Wiener with a different operating point is shown below in Fig. 5.8(c). The noise reduction is evident compared to Fig. 5.8(a), along with a greater amount of detail present than in Fig. 5.8(b). The Kalman-AWF result, Fig. 5.8(d), still provides better noise reduction along with greater detail preservation.
Figure 5.7: Final frame of the processed outdoor data imagery. (a) Original Frame; (b) BM3D-Wiener; (c) VBM3D-Wiener; (d) Kalman - BM3D-Wiener; (e) Kalman-Wiener; and (f) Kalman-AWF.
Figure 5.8: Final frame ROI of the processed outdoor data imagery. (a) Original Frame; (b) BM3D-Wiener, $\phi = 1$; (c) BM3D-Wiener, $\phi = 0.6$; (d) Kalman-AWF.
CHAPTER VI

CONCLUSIONS

The Kalman-AWF method is a computationally efficient video restoration algorithm. The Kalman filter exploits temporal correlation in video sequences. The AWF provides spatially adaptive deconvolution to mitigate the effects of the system PSF in the presence of the spatially varying residual noise from the Kalman filter step. The proposed Kalman-filter based restoration methods use a robust affine background motion model. We believe this makes them suitable for applications where the motion is dominated by camera motion with limited local scene motion. Airborne imaging and tower based surveillance are possible applications for this kind of restoration. The MDKF-AWF can extend the use of temporal filtering to treat fast moving local motion. Furthermore, the ability to track residual noise after the denoising process allows the AWF to properly be applied to the denoised image. The proposed hybrid filter combining the Kalman filter and BM3D has been shown to combine the best of each method. Background restoration benefits from the temporal Kalman filter, and the local motion restoration benefits from BM3D. We believe that the proposed Kalman-AWF provides an excellent compromise between performance and computational complexity. We believe this method is well suited for real-time implementation. The Kalman-BM3D-Wiener provides the best performance, but has additional computational complexity. The MDKF can be used instead of the standard Kalman filter and provides benefits in the case of fast local scene motion.


APPENDIX A

Camera Software

The website to containing information about the Coriander software:

http://damien.douxchamps.net/ieee1394/coriander/

The website that has a download of the Coriander software:

http://sourceforge.net/projects/coriander/
MATLAB Code

MATLAB code written to generate the RSLC.

```matlab
clear all
close all
clc

%% The known parameters of the camera and environment
lambda = 550 * 10^-6; % Wavelength of light in millimeters
F_num = 16; % The F number of the optics
Psi = 88.9; % Desired radius of the printed RSLC pattern in millimeters
l = 1778; % Distance from lens to target in millimeters
l_prime = 12.5; % Focal length of the optics in millimeters

%% Object Space calculations
w_ob = (l_prime/l)*(1.25/(lambda*F_num)); % Eqn.(3.12)
k = w_ob/Psi; % Eqn.(3.13)

%% Create the RSLC pattern
% Generate the array values for x and y
[x,y] = meshgrid(linspace(-Psi,Psi,5000),linspace(-Psi,Psi,5000));

% Calculate psi, Eqn.(3.2)
psi = sqrt(x.^2 + y.^2);

% Calculate the RSLC pattern, Eqn.(3.1)
RSLC = 255*cos(pi*k*psi.^2);
```