Material state tracking by nondestructive evaluation data

Elizabeth Dimmitt Gregory

Iowa State University
Material state tracking by nondestructive evaluation data

by

Elizabeth Dimmitt Gregory

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Program of Study Committee:
Stephen D. Holland, Major Professor
Ran Dai
Phillip Jones
Ping Lu
William Meeker

Iowa State University
Ames, Iowa
2016

Copyright © Elizabeth Dimmitt Gregory, 2016. All rights reserved.
DEDICATION

I dedicate this dissertation to my parents
Jack and Marilyn Gregory
for teaching what is good and right.
# TABLE OF CONTENTS

LIST OF TABLES ................................................................. v
LIST OF FIGURES .............................................................. vi
ACKNOWLEDGMENTS ............................................................ xiii
ABSTRACT ................................................................. xiv

CHAPTER 1. INTRODUCTION ...................................................... 1

CHAPTER 2. TRACKING SPATIO-TEMPORAL DATA ................. 6
  2.1 Introduction ................................................................. 6
  2.2 Bayes’ theorem as part of state tracking ......................... 6
  2.3 The Kalman filter and state tracking .............................. 8
  2.4 Markov random fields .................................................... 10
  2.5 Spatio-temporal variables and filtering .......................... 13
  2.6 Material state tracking in the context of spatio-temporal variables .... 17
    2.6.1 Prediction step details ........................................... 20
    2.6.2 Update step details .............................................. 22
  2.7 Simulation results ....................................................... 23
    2.7.1 Estimator Convergence .......................................... 26

CHAPTER 3. APPLYING SPATIO-TEMPORAL DATA TRACKING
  TO FLASH THERMOGRAPHY DATA OF COMPOSITE STRUCTURES ... 27
  3.1 Introduction ............................................................... 27
3.2 Damage initiation and growth ........................................... 29
3.3 Flash thermography measurement ..................................... 32
3.4 Computed tomography as ‘truth’ ........................................ 41
3.5 Specific algorithm implementation ..................................... 45
3.6 Experiment results and conclusions .................................... 45

CHAPTER 4. UNCERTAINTY IN IMAGE REGISTRATION ........... 55
4.1 Introduction ................................................................. 55
4.2 Spatial awareness and registration ..................................... 55
  4.2.1 Registering two dimensional data to a three dimensional object, a
        probabilistic approach ................................................ 56
  4.2.2 Evaluating mappings .................................................. 60
  4.2.3 Probabilistic method .................................................. 60
  4.2.4 Machine vision solution .............................................. 69
4.3 Tracking variables with more complicated posterior distributions .... 71
  4.3.1 Explanation of system ............................................... 72
  4.3.2 Variable two dimensional coordinates ............................ 72
  4.3.3 Variable three dimensional coordinates .......................... 76
  4.3.4 Variable two dimensional and three dimensional coordinates . . 80
4.4 Approximating complex distributions with Gaussian mixtures ....... 88
  4.4.1 Comparing to the Kalman filter equations ....................... 91
  4.4.2 Approximating Gaussian mixtures ................................ 91
4.5 Summary .................................................................. 100

CHAPTER 5. CONCLUSIONS .................................................. 101

BIBLIOGRAPHY ................................................................. 104
LIST OF TABLES

Table 2.1  Probabilities associated with the general sensor. . . . . . . . . . . 22
Table 2.2  Probabilities associated with the sensor for the simulation. . . 23
## LIST OF FIGURES

Figure 2.1  The Baysian filtering cycle as a flowchart, showing the steps of prediction and updating. .................................................. 7

Figure 2.2  An example of a material Markov random field used in the Ising model where binary pixels states are represented as ‘+’ and ‘-’. 12

Figure 2.3  An example of a sensor measurement of Markov random field where ‘+’ is an undamaged pixel, ‘-’ is a damaged pixel and ‘o’ is missing data. ............................................................... 13

Figure 2.4  Initial truth, estimate and sensor measurements for the simulation. Damaged pixels are pink and undamaged pixels are blue. 24

Figure 2.5  Final truth, estimate and sensor measurements for the simulation. Damaged pixels are pink and undamaged pixels are blue. 24

Figure 2.6  A comparison of percent error in the sensor and the estimate, both with respect to the truth. ........................................... 25

Figure 2.7  Error in sensor minus error in the estimate for 100 simulations. A positive result means that the sensor error is larger than the estimate error. ......................................................... 26

Figure 3.1  One of the composite specimens used in this experiment. .... 28

Figure 3.2  The ASTM D7136 impact apparatus located at the Center for Nondestructive Evaluation at Iowa State University. .... 29

Figure 3.3  The ASTM D7137 Edgewise compression frame used to grow delaminations. ......................................................... 30
Figure 3.4  The transverse compression setup with acoustic emissions probe used to grow delaminations. ................................. 31
Figure 3.5  The four-point bending setup with acoustic emissions probe used to grow delaminations. ................................. 31
Figure 3.6  The setup of the flash thermography system where two flash lamps are pointed at the specimen located in the center and the thermal camera, which collected temperature data as the specimen cooled, is located at the top of the image. ................................. 32
Figure 3.7  A comparison of the $\log_{10}$ temperature curve of a single damaged pixel and a single undamaged pixel. ................................. 34
Figure 3.8  A comparison of the second derivative of the $\log_{10}$ temperature curve with respect to time of a single damaged pixel and a single undamaged pixel. ................................. 35
Figure 3.9  The second derivative of $\log_{10}$ temperature - $\log_{10}$ time curves at a single time point for all pixels. Areas of damage are clearly visible. The image has been undistorted and cropped to include only the specimen. ................................. 36
Figure 3.10  The second derivative of $\log_{10}$ temperature - $\log_{10}$ time curve indicating areas of damage. The image is neither cropped nor undistorted. ................................. 37
Figure 3.11  The sum squared error with respect to the 'truth' for the top of specimen E-5 at time step $k = 3$. ................................. 39
Figure 3.12  The sum squared error with respect to the 'truth' for the bottom of specimen E-5 at time step $k = 3$. ................................. 40
Figure 3.13  The computed tomography setup at the Center for Nondestructive Evaluation, with the X-ray source, detector, and rotating platform. ................................. 42
| Figure 3.14 | A single slice of computed tomography reconstruction. | 43 |
| Figure 3.15 | The minimums of a few slices of the CT reconstruction of E-5 converted to space or composite. | 44 |
| Figure 3.16 | Results of resampling of the CT truth from Figure 3.15. | 45 |
| Figure 3.17 | Comparing error in the sensor and the estimate for specimen E-1 results. | 46 |
| Figure 3.18 | Comparing error in the sensor and the estimate for specimen E-3 results. | 46 |
| Figure 3.19 | Comparing error in the sensor and the estimate for specimen E-4 results. | 47 |
| Figure 3.20 | Comparing error in the sensor and the estimate for specimen E-5 results. | 48 |
| Figure 3.21 | Comparing error in the sensor and the estimate for specimen E-6 results. | 48 |
| Figure 3.22 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-1 at time step $k = 2$. | 49 |
| Figure 3.23 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-1 at time step $k = 6$. | 49 |
| Figure 3.24 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-3 at time step $k = 2$. | 50 |
| Figure 3.25 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-3 at time step $k = 6$. | 50 |
| Figure 3.26 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-4 at time step $k = 2$. | 51 |
| Figure 3.27 | Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-4 at time step $k = 6$. | 51 |
Figure 3.28 Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-5 at time step $k = 2$.

Figure 3.29 Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-5 at time step $k = 6$.

Figure 3.30 Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-6 at time step $k = 2$.

Figure 3.31 Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-6 at time step $k = 6$.

Figure 4.1 Illustration of image, object, and visible point definitions.

Figure 4.2 The camera frame, image frame and object frame for a single scene.

Figure 4.3 Object as observed by Camera GEV0824.

Figure 4.4 Object as observed by Camera GEV0828.

Figure 4.5 Camera GEV0824 with image points.

Figure 4.6 Camera GEV0828 with image points.

Figure 4.7 Edges identified in GEV0824.

Figure 4.8 Edges identified in GEV0828.

Figure 4.9 Mapping weights for GEV0824.

Figure 4.10 Mapping weights for GEV0828.

Figure 4.11 Results of point mapping using the new algorithm for image registration.

Figure 4.12 OpenCV stereo calibration for image registration.

Figure 4.13 Example of Radkowski’s program to automatically detect the composite specimens.

Figure 4.14 The specimen corners as detected by the Radkowski software.

Figure 4.15 The undistorted image, cropped to include only the specimen.

Figure 4.16 Rotation results with $N=1000$ and $\sigma_{2D} = 1$, isometric view.

Figure 4.17 Rotation results with $N=1000$ and $\sigma_{2D} = 1$, X-Y plane.
Figure 4.18  Translation results with $N=1000$ and $\sigma_{2D} = 1$, isometric view... 74
Figure 4.19  Translation results with $N=1000$ and $\sigma_{2D} = 1$, X-Y plane. ... 74
Figure 4.20  Rotation results with $N=1000$ and $\sigma_{2D} = 3$, isometric view. ... 75
Figure 4.21  Rotation results with $N=1000$ and $\sigma_{2D} = 3$, X-Y plane. ... 75
Figure 4.22  Translation results with $N=1000$ and $\sigma_{2D} = 3$, isometric view... 76
Figure 4.23  Translation results with $N=1000$ and $\sigma_{2D} = 3$, X-Y plane. ... 76
Figure 4.24  Rotation results with $N=1000$ and $\sigma_{3D} = 1$, isometric view. ... 77
Figure 4.25  Rotation results with $N=1000$ and $\sigma_{3D} = 1$, X-Y plane. ... 77
Figure 4.26  Translation results with $N=1000$ and $\sigma_{3D} = 1$, isometric view. ... 78
Figure 4.27  Translation results with $N=1000$ and $\sigma_{3D} = 1$, X-Y plane. ... 78
Figure 4.28  Rotation results with $N=1000$ and $\sigma_{3D} = 3$, isometric view. ... 79
Figure 4.29  Rotation results with $N=1000$ and $\sigma_{3D} = 3$, X-Y plane. ... 79
Figure 4.30  Translation results with $N=1000$ and $\sigma_{3D} = 3$, isometric view... 80
Figure 4.31  Translation results with $N=1000$ and $\sigma_{3D} = 3$, X-Y plane. ... 80
Figure 4.32  Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, isometric view. ... 81
Figure 4.33  Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, X-Y plane. ... 81
Figure 4.34  Translation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, isometric view. ... 82
Figure 4.35  Translation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, X-Y plane. ... 82
Figure 4.36  Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, isometric view. ... 83
Figure 4.37  Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, X-Y plane. ... 83
Figure 4.38  Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, isometric view. ......................................................... 84

Figure 4.39  Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, X-Y plane. ......................................................... 84

Figure 4.40  Rotation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, isometric view. ......................................................... 85

Figure 4.41  Rotation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, X-Y plane. ......................................................... 85

Figure 4.42  Translation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, isometric view. ......................................................... 86

Figure 4.43  Translation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, X-Y plane. ......................................................... 86

Figure 4.44  Rotation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, isometric view. ......................................................... 87

Figure 4.45  Rotation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, X-Y plane. ......................................................... 87

Figure 4.46  Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, isometric view. ......................................................... 88

Figure 4.47  Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, X-Y plane. ......................................................... 88

Figure 4.48  Compare a Gaussian mixture with 8 distributions with varying weights and grouped means to fitted Gaussian mixture with 4 distributions. ......................................................... 97

Figure 4.49  Compare a Gaussian mixture with 8 distributions with even weights and grouped means to a fitted Gaussian mixture with 4 distributions. ......................................................... 98
Figure 4.50  Comparing a Gaussian mixture with 8 distributions with varying weights and ungrouped means to a fitted Gaussian mixture with 4 distributions. .......................... 99

Figure 4.51  Comparing a Gaussian mixture with 8 distributions with even weights and ungrouped means to fitted Gaussian mixture with 4 distributions. .......................... 100
ACKNOWLEDGMENTS

I would like to thank Dr. Steve Holland for advising me through this entire process and for being my most enthusiastic advocate. I would also like to thank my committee, Dr. Ran Dai, Dr. Phillip Jones, Dr. Ping Lu, and Dr. William Meeker. Thank you to Dr. Joe Gray and Scott Wendt at CNDE for helping me collect data and answering questions. The entire staff at CNDE have been very supportive and helpful during my time at Iowa State. Tyler Lesthaeghe and Jyani Vaddi were crucial to my success. I was lucky to have many amazing mentors at Iowa State. Specifically, I would like to thank Dr. Leonard Bond and Dr. Joseph Schaefer.

Finally, I would like to thank my friends in the GradSWE community for their wonderful support.
ABSTRACT

This thesis tackles the challenge of using nondestructive evaluation data as a sensor measurement input for a state estimation scheme in order to estimate the current state a part that is changing over time. This is made particularly challenging because of the multidisciplinary nature of the problem. The estimation solution incorporates work from statistics, computer vision, and the physics of nondestructive evaluation.

This thesis discusses the basis for spatio-temporal Kalman filtering and uses a simple version of spatio-temporal filtering to simulate material state tracking on composite specimens. The simulation illustrates that by using the algorithm presented here, even with very naive inputs, it is possible to track a dynamic material state and provide estimates that better reflect the true state of the part as compared with the most recent sensor measurement alone.

Additionally, this thesis demonstrates the algorithm on laboratory test data. Composite panels were manufactured and then intentionally impacted to induce subsurface delaminations. The composite panels were then loaded multiple times in four-point bending to induce incremental damage growth. After each damage event (initiation and loading) flash thermography and computed tomography data was collected. The flash thermography data was used as a sensor measurement in the spatio-temporal Kalman filter and the computed tomography data was used as a ‘truth’ value for comparison. For four out of five data sets, at every time step, the spatio-temporal Kalman filter estimate matched the computed tomography ‘truth’ better than the most recent single sensor measurement. For the fifth data set, the estimate better matched the truth at most time steps.
Finally, we present a method that uses a probabilistic approach to identifying the location and orientation, or pose, of a specimen or part within an image. This process found the most likely transformation from the object coordinate frame to the camera coordinate frame but also ranked less probable transformations by likelihood. The discussion is continued with an exploration of different, non-Gaussian uncertainty distributions that result from the process of registering two dimensional images to three dimensional part models. A method for approximating non-Gaussian distributions using Gaussian mixtures is presented and discussed.

The work presented in this thesis successfully demonstrates that Bayesian estimation with nondestructive evaluation data will provide superior and more meaningful state estimates while discussing the issues that must be considered in doing this estimation properly.
CHAPTER 1. INTRODUCTION

At this time, nondestructive evaluation (NDE) in the aerospace industry is largely a manual process. Technicians acquire nondestructive evaluation data from a variety of modalities such as thermography, x-ray, fluorescent penetrant, and ultrasound. Each modality provides some information about the state of the part, but a full picture of the state of the part is often difficult to assess based on the disparate data provided by each modality. Additionally, data from previous inspections is often discarded and assessments are made based solely on the most recent data available. New abilities to handle large amounts of data and the need for life extending measures for flight critical parts means the aerospace industry must develop new techniques to methodically collect, fuse, and utilize nondestructive evaluation data.

The challenge of material condition tracking is not merely a data fusion problem nor a mechanics of material problem. In tackling this challenge, one must also incorporate machine learning, the physics of each nondestructive evaluation technique, and a statistical estimation. The goal of this work is not to provide a perfect template for obtaining perfect knowledge. That is an unrealistic goal; the pursuit of which is a waste of a good engineer’s time. The best solution is one that acknowledges the imperfections in data, process and material and creates a framework to overcome the uncertainty associated with real-world actions and decisions.

The multidisciplinary challenges that must be solved to better utilize the large amounts of data from nondestructive evaluation range from registering all nondestructive evaluation measurements into a physical frame [Brierley et al. (2012, 2014)] to properly au-
tomating what has been, historically, a manual process [Summan et al. (2010)]. There have also been significant efforts to create cohesive nondestructive evaluation models that utilize the widely disparate data that different nondestructive evaluation modalities provide [Chen (2003); Liu and Forsyth (2007)].

Efforts being made in industry to build holistic models center around a specific individual part that incorporate a variety of data about that part. The work done by the United States Air Force uses the term ‘digital twin’ or ‘digital thread’ to describe the effort of combining the lifetime data of a specific part in order to provide feedback to designers and manufacturers [Tuegel et al. (2011), Air Force Material Command (2013)]. The ‘digital twin’ would be a model that reflects the physical state of the part, such as material properties and physical dimensions. The ‘digital thread’ is more of a data tracking plan that allows the results of testing at any point during the lifetime of a part to be used as input to re-evaluate the design and manufacturing processes.

This holistic view of part evaluation is very attractive. It is enticing to think that we could know everything about a part, if we just paid enough attention and collected enough data. Obviously, this is an over simplification; but what is clear is that a large amount of data is not utilized or under-utilized. When nondestructive evaluation data is collected, it is often used to answer only one question: ”Is there anything observable in this data that may result in failure prior to the next inspection?” Once that question is addressed, the data is discarded. This leaves more interesting questions unanswered, such as:

- What is this data telling me about how the part is changing?

- What is the difference between this data set and a set that was taken previously?

- Is there anything in this data set that indicates a failure at any point in the future, even after the next inspection?
The inverse problem refers to determining what an inspection is actually communicating about a part beyond that the part falls outside a standard. The inverse problem usually does not have one solution. However, it is the single most important question. Addressing these problems will allow industry to make better decisions about planning inspections, retiring a part, and designing future structures.

Building a model that, in a sense, lives with the actual part means that we must track the true state of the part. The only methods available to do this are estimation tools. Estimating a dynamic state and tracking a state are fundamentally the same action. An estimate of a state is the best conclusion of the state of the part at a specific time point. This estimate must include a probabilistic model associated with it. For estimates that are treated like random variables, as they are in this document, this means there is an estimate and a probability distribution associated with that estimate. If the estimate is a Gaussian random variable then the estimate is the mean of a normal distribution and the shape of the distribution is fully described by the variance.

More specifically, we are interested in estimating the current state of a part. Is it damaged or undamaged? We have a sensor measurements that can be used in this estimation. The sensor measurement might measure thermal conductivity which changes if the part is damaged, but the measurement also has some uncertainty and variability associated with it. Therefore, the sensor measurement is a random variable. The state is a random variable, the sensor measurement is a random variable, and the estimate is a realization of the estimation process.

The Bayesian method for state tracking is a cycle of using a dynamic model to predict the state at the next time step based on current information and then updating that prediction using Bayes’ theorem when new sensor data is made available [Chen (2003), Varshney (1996)]. This methodological framework provides the benefit of an estimate and a probabilistic model that can be re-evaluated when any new data becomes available, meaning that both forward and backward smoothing in time can be applied.
The most widely used Bayesian estimation scheme for a dynamic state is the Kalman filter. In this context, the ‘filter’ refers to the process of ‘filtering’ the sensor data over a time period to track a state. The Kalman filter uses the Kalman gain to update a Gaussian probability distribution for the state estimate based on the prior Gaussian probability of that state estimate and the Gaussian distribution of sensor measurements [Kalman (1960)]. The Kalman filter assumes linear dynamic systems (meaning the state at the next time step is a linear function of the state at the current time step, input, and white process noise) and linear sensor models. The extended Kalman filter and the unscented Kalman filter have been developed to apply the Kalman filter tools to non-linear and non-Gaussian systems. These filters are in wide use but are not optimal in the way that the Kalman filter is for linear systems and Gaussian variables [Simon (2006)].

The Kalman filter is an appealing tool but the pure Kalman filter would be impossible to implement with nondestructive evaluation data. The state estimate vector would be too large. Nondestructive evaluation data is, by its nature, spatial data as it is always representing the physical state of part. Cressi and Wikle [Cressie et al. (2010); Cressie and Wikle (2002); Cressie (1991); Kang and Cressie (2011)] applied the Kalman filter to address the specific nature of spatio-temporal data. The applications for this work include weather prediction, disease infection tracking, and brain lesion growth tracking. This framework is ideal for tracking material state using multivariate nondestructive evaluation data.

To use the spatio-temporal tools, it is necessary to place all the nondestructive evaluation data into a spatial context, the context of the actual physical volume of the specimen or part being scanned. There have been efforts to automate systems that align or register spatial data from nondestructive evaluation measurements [Brierley et al. (2012), Brierley et al. (2014)]. This document describes some of the work that has been done in that area.
There is a critical need to make all data more connected, not just spatially, but also across time, operator, and modality. There has been significant work by Dr. Holland and Tyler Lesthaeghe to develop software tools to automate the NDE process, not merely the inspection process of collecting data, which in some cases is already very automated, but also the process of data analysis, storage, and visualization [Gregory et al. (2015), Holland and Lesthaeghe (2016)]. By recording all the steps taken to reach a certain conclusion, we can be sure that the steps can be repeated if data is altered in some way or, perhaps more importantly, the steps can be altered in a methodical way to compare different conclusions. The tools Lesthaeghe has developed for this purpose are Datacollect, a software tool to track data collection and automate the data storage and analysis process, and Databrowse, a database and data visualization tool used to analyze and organize data. These tools and those similar are essential for the next stage of big data analysis in nondestructive evaluation.

This thesis consists of three parts. The first part describes the theoretical framework used to develop the process of material state tracking. The second part presents some results from a demonstration study of the material state tracking process. The third part discusses some of the challenges that were encountered during the work and explains their significance. The fundamental goal of the work presented here is to demonstrate that a more holistic view of material state tracking using Bayesian inference and nondestructive evaluation data is achievable and results in improved outcomes, even in the case of complex sources of uncertainty.
CHAPTER 2. TRACKING SPATIO-TEMPORAL DATA

2.1 Introduction

The goal of this work is to develop a theoretical framework for tracking material state over time using nondestructive evaluation data as a sensor measurement. First, we describe the Kalman filter approach to state tracking, specifically how it uses the Bayes’ theorem as a method for updating a state estimate when new data becomes available and then how it predicts the state at the next time step. That process is state tracking. Then, we discuss the use of Markov random fields as a mathematical model for describing the spatially dependent nature of the material state. Next, we discuss the application of the Kalman filter to a Markov random field that is changing over time, the spatio-temporal Kalman filter. Finally, we present our algorithm for the spatio-temporal filter for material state tracking and some simulation results.

2.2 Bayes’ theorem as part of state tracking

The Bayesian approach to state estimation is the application of Bayes’ theorem to update an a priori estimate of a state variable with information from a sensor to determine a posterior estimate of that state. This is done specifically by treating the state as a random variable and performing the update on that random variable. The ‘estimate’ is the most probable value of the random variable. It can be read as the probability of a state, $X$, given a measurement, $Z$, is equal to the probability of the measurement given the state multiplied by the probability of the state divided by the probability of
the measurement, stated as

\[ P(X|Z) = \frac{P(Z|X) P(X)}{P(Z)}. \]  \hspace{1cm} (2.1)

However, filtering is not merely applying Bayes’ theorem. Filtering is a process of predicting (making an a priori estimate) and then updating the estimate based on any new information, such as a sensor measurement. Figure 2.1 shows the process. The posterior estimate at one time step evolves into the aprior estimate at the next step using a dynamic model. Filtering is specifically performing this estimation cycle on a dynamic state variable, meaning a state variable that is changing over time.

![Figure 2.1: The Baysian filtering cycle as a flowchart, showing the steps of prediction and updating.](image)

In this chapter, I will present the basis for the proposed spatio-temporal filtering algorithm to be used with nondestructive evaluation (NDE) data. First, I will discuss some of the key principals that were used in developing the algorithm, and finally, I will present the algorithm and simulation results.
2.3 The Kalman filter and state tracking

The simplest Bayesian filter is the Kalman filter. Proposed in 1960 by Rudolf Kalman [Kalman (1960)], the Kalman filter provides the optimal estimate and covariance matrix (thereby defining the distribution for the Gaussian state variable) for a state vector with linear dynamic and sensor models with additive Gaussian white noise. Equation 2.2 shows the dynamic system for the state variable $x$.

For example, if we are interested in tracking the location of a projectile in the real world, we have the simple dynamic equations of the projection. However, the simple dynamic equations do not account for random wind gusts. We could generalize the random gusts as random noise. Now let us assume that we have a sensor that measures location at each second, but the sensor also has some random error associated with it. To track the position of the projectile, we would start with our initial sensor measurement as an estimate and the variance of the sensor measurement would be the variance of the estimate. Then, we would predict the position at one second in the future using the simple dynamic equation, but the variance of the new predicted estimate would increase because we also know there are random gusts of wind. When the sensor measurement for the next second is available, we can update our predicted estimate and the estimate’s variance. That is the filtering procedure for state tracking.

All the general Kalman filter equations use the form provided by Dan Simon [Simon (2006)]. The dynamic state equation is

$$x_t = Fx_{t-1} + m + \omega,$$

where:

- $x$ is the state variable.
- $t$ is time.
• $F$ is the state transition matrix from time $t - 1$ to time $t$.

• $m$ is input.

• $\omega$ is additive Gaussian white noise with covariance $Q$, stated as

$$\omega \sim \mathcal{N}(0, Q). \quad (2.3)$$

Equation 2.4 shows the linear sensor model for measurement $z$,

$$z_t = Hx_t + \nu, \quad (2.4)$$

where $\nu$ is additive Gaussian white noise with covariance $R$. Stated as

$$\nu \sim \mathcal{N}(0, R). \quad (2.5)$$

The estimate of the mean, $\hat{x}$, and the covariance, $P$, are predicted by Equations 2.6 and 2.7. The superscript denotes that it is an a-priori estimate. It is important to understand the difference between the state, $x$, and the state estimate, $\hat{x}$. The state exists, is concrete, but is unknown. The estimate is a random variable created to approximate the state to the best of our ability. The state estimate in no way affects the state. In fact, the only reason we include the state at all in the model is for comparison purposes in a simulation setting. In the real world the state can never been known.

As previously stated the two important steps in the filtering procedure are the prediction step and update step. For the Kalman filter, the prediction step is

$$\hat{x}_t^- = F\hat{x}_{t-1} + m, \quad (2.6)$$

where

$$P_t^- = FP_{t-1}F^T + Q. \quad (2.7)$$
Then, the mean and covariance of the state estimate are updated by Equations 2.8 and 2.9. These describe the posterior estimate:

\[
\hat{x}_t = \hat{x}_{t-1} + K_t (z_t - H \hat{x}_{t-1}). \tag{2.8}
\]

The covariance is then updated as

\[
P_t = (I + K_t H) P^-_t, \tag{2.9}
\]

where \( K_t \) is the Kalman gain defined by as

\[
K_t = P^-_t H^T (H P^-_t H^T + R)^{-1}. \tag{2.10}
\]

The Kalman filter, and its derivatives (extended Kalman filter, unscented Kalman filter, etc), are easily the most widely used estimate schemes today. The Kalman filter does have its drawbacks. When state vectors become very large, the computational cost of inverting the covariance matrix becomes limiting. Also, noise that cannot be approximated as Gaussian cannot be processed at all.

### 2.4 Markov random fields

Markov random fields are spatially distributed random variables where each spatial variable is somewhat dependent on all the other random variables. This dependency is related to their relative locations. This results in Markov behavior for the entire graph model, meaning that points are dependent on their neighbors, which are then dependent on their neighbors.

In 1925, Ising proposed a model for determining the probability of a specific random field based on all possible random fields [Ising (1925)]. Ising developed his model to analyze the spins of electrons in a material. However, we will use this model in the context of material state damage. Let us first define a structured grid with dimensions \( x \)
and \( y \) with discrete pixels. Each pixel’s state is denoted as \( \omega_{ij} \) where \( i \in X \) and \( j \in Y \).

Let us also say that there exists a function such that \( \sigma (\omega_{ij}) = 1 \) if the electron has positive spin and \( \sigma (\omega_{ij}) = -1 \) if the electron has negative spin. We can simplify the the nomenclature as \( \sigma (\omega_{ij}) = \sigma_{ij} \).

Kindermann and Snell provide a simple nomenclature for discussing the Ising model [Kindermann and Snell (1980)] by defining the energy of any specific Markov random fields containing electrons as described

\[
U(\omega) = -J \sum_{p,q} \sigma_p \sigma_q - mH \sum_{p} \sigma_p, \tag{2.11}
\]

where:

- \( p \) and \( q \) refer to specific locations in the field that are exactly one unit apart. \( p \) and \( q \) will always refer to adjacent pairs. The Ising model only considered the interactions of adjacent pairs of electrons.

- \( J \) is a property of the field. If \( J > 0 \), then the model is attractive and neighboring spins will match. If \( J < 0 \), then the model is repulsive and the neighboring spins will not match. Thus, the first term in Equation 2.11 calculates the energy of the interactions of adjacent elections.

- \( H \) is an external magnetic field (or an input, if you will) that is driving the spins to be positive or negative.

- \( m \) is a material property.

The second term in Equation 2.11 is minimized when all the elements have the same spin.

We are not interested in electron spins, we are concerned with material state tracking. The model presented above is still valid, though. In the case of material damage, the two states are not positive or negative spins of electrons at point \( ij \), but damaged
or undamaged pixels at point $ij$. Energy Equation 2.11 would not contain the same constants but a similar energy equation could be determined.

Figure 2.2 shows an example of a $5 \times 6$ material Markov random field where the ‘+’ refers to an undamaged pixel and the ‘−’ indicates damaged pixel.

Figure 2.2: An example of a material Markov random field used in the Ising model where binary pixels states are represented as ‘+’ and ‘−’.

The probability of any particular field is defined by

$$
P(\omega) = \frac{\exp\left(-\frac{1}{kT}U(\omega)\right)}{\sum_{\Omega} \exp\left(-\frac{1}{kT}U(\omega)\right)},
$$

(2.12)

where:

- The denominator is the sum over all possible fields.
- $T$ is temperature;
- $k$ is a universal constant (recalling that the Ising model is for the spin of electrons so in the case of material state it would be different or, for simplicity, $k = 1$).

Using this probability model, it is easy to assign posterior probabilities to specific Markov random field configurations based on an incomplete sample. Figure 2.3 shows
just such an incomplete sensor measurement. The best estimate for the complete state can then be determined to be the most probable (according to Equation 2.12) field that matches the partial measurement. Or, if there is a prior probability for the field, then a Bayesian update step can be performed using this data.

Figure 2.3: An example of a sensor measurement of Markov random field where ‘+’ is an undamaged pixel, ‘−’ is a damaged pixel and ‘◦’ is missing data.

This type of model has been expanded to be used in all kinds of applications, including machine vision and image processing [Blake and Kohli (2011)] where the goal is to determine if pixels belong to an object in the foreground or in the background. This model is also used to differentiate between multiple sclerosis lesions and normal brain tissue in brain MRIs [Karimaghaloo et al. (2012)]. The binary nature of the state variable is ideal for determining damaged and undamaged areas of composite materials in nondestructive evaluation data.

2.5 Spatio-temporal variables and filtering

The Markov random field alone is not a filtering algorithm (meaning it is not a method to track a state); it is merely a tool for placing spatial data into a probabilistic
state-space model.\footnote{It is possible to treat time as just another dimension and solve for the posteriors of the entire state (including all time points) at once using the entire sensor data set. This could be done over and over as new sensor data becomes available. Unfortunately, in that case, the problem can quickly become massive [Riccio et al. (2006); Rue and Held (2005)]. Time is a unique dimension and knowledge about how the state will evolve over time is useful in state estimation.}

The generalized dynamic equation for a spatio-temporal state is

\[ x(s, t) = \int_D f(x(u, t - 1), m) \, du + \omega, \quad (2.13) \]

where:

- \( s \) is a specific location in two dimensional or three dimensional space. All the locations make up the surface or volume \( D \).
- \( u \) is also a spacial variable that is defined over the same space as \( s \). The integral means that \( s \) is dependent all of the other spacial points.
- \( t \) is time.
- \( m \) is some input.
- \( x(\cdot) \) is the state variable we are trying to estimate.
- \( \omega \) is Guassian white noise.

The state is defined at each location and time step. Equation 2.13 means that the current state at each location is a function of the state at every location from the previous time step, some input and some randomness.

The sensor model is then

\[ z(s, t) = h(x(s, t), \nu). \quad (2.14) \]

where:

- \( z \) is the sensor measurement for the space \( s \) at time \( t \).
• $h$ is a function of space, time and random noise $\nu$.

Cressie and Wikle proposed a reduced dimensional approach to applying a Kalman filter to these spatio-temporal variables [Cressie (1991); Cressie and Wikle (2002); Wikle and Cressie (1999)]. Cressie and Wikle provide a method to perform state tracking on Markov Random fields that are changing over time using the Kalman filter structure. The spatio-temporal Kalman filter allows us to model the material state damage as a Markov random field and use the very elegant Kalman filter equations. In this approach, they assume that the dynamic equation is specifically defined as the sum of a mean variable and some dynamic noise

$$x(s,t) = \bar{x}(s,t) + \nu, \quad (2.15)$$

where $\nu$ is Gaussian white noise with covariance $V$.

Now, the mean of the state vector is written similar to Equation 2.13 as

$$\bar{x}(s,t) = \int_D w_s(u) \bar{x}(u, t - 1) du + \eta, \quad (2.16)$$

where $\eta$ is noise that is white in spatial dimensions and colored in time. This is the term that included input, $m$. It has covariance $Q$ and $w_s(u)$ is an interaction function connecting the state mean variable, $\bar{x}$, to the spatial location $u$.

Cressie and Wikle proposed approximating the strongest spatial connections by applying the assumption that the mean state vector is only dynamically dependent on a certain subset of locations. For example, the subset could be assumed to be certain areas in the path of weather patterns (such as the area upwind) or, in the case of contamination, immediately adjacent areas. This is the reduced dimensional approach

$$\bar{x}(s,t) = \sum_{p=1}^{K_p} \phi_p(s) a_p(t) = \phi^T(s) a(t), \quad (2.17)$$

where
• $p$ is referring to a particular spatial location in the subset $K_p$. $K_p$ should not be confused with the Kalman gain $K$.

• $\phi_p(s)$ is defined for each location in the subset and together make an orthonormal sequence of basis functions.

• $\phi$ is the array containing all $K_p$ basis functions $\phi_p$.

• $a_p(t)$ is random time variable that is defined for each location.

• $a$ is the array of all $K_p$ values of $a_p$.

Equation 2.17 nicely segregates the problem into spatial and temporal components. The final component of the system is the dynamic transition of $a(t)$ from time step $t - 1$ to $t$:

$$a(t) = H a(t - 1) + J \eta,$$

where $H$ and $J$ are functions of the basis functions chosen and make up for the information loss between the state and the purely temporal variable.

Equations 2.15 and 2.18 are a simple linear sensor model like that shown in Equation 2.4 and lend themselves nicely to the Kalman filter equations, the results of which are given as

$$\hat{a}^- (t) = H \hat{a} (t - 1),$$

with covariance

$$P^- (t) = HP (t - 1) H^T + JQJ^T.$$

Equations 2.19 and 2.20 show the a priori estimates of the variable $a$ and its covariance matrix.
The estimate and the covariance are then updated based on the sensor information

\[
\hat{a} (t) = \hat{a}^{-} (t) + K (t) \left[ z (t) - \Phi \hat{a}^{-} (t) \right],
\tag{2.21}
\]

and

\[
P (t) = P^{-} (t) + K (t) \Phi P^{-} (t).
\tag{2.22}
\]

Equations 2.21 and 2.22 show the updated estimates of the variable \( a \) and its covariance matrix. The Kalman gain \( K (t) \) is found by

\[
K (t) = P^{-} (t) \Phi^T \left[ R + V + \Phi P^{-} (t) \Phi^T \right]^{-1},
\tag{2.23}
\]

where \( \Phi \) is the matrix containing all the \( \phi \) arrays for all the spatial locations \( s \). Recall that \( R \) and \( V \) are sensor and process noise respectively.

Our interest is in estimates for the state mean variable

\[
\hat{x} (s, t) = \phi (s)^T \hat{a} (t).
\tag{2.24}
\]

### 2.6 Material state tracking in the context of spatio-temporal variables

The specific approach taken to track the material state starts with the basic Bayesian filtering process. The process consists of two steps, predicting a state and then updating that prediction when a new sensor measurement becomes available. The process for estimating material state additionally incorporates the spatial association of the spatio-temporal Kalman filter but utilizes a simpler method than used by Cressie and Wikle. This approach is applicable to any type of nondestructive evaluation data and does not require detailed knowledge of the damage evolution system.

In this application, the state will be whether an area of our part is damaged or undamaged. In most real-world situations, we are concerned with the location and type
of flaw or damage within a part. For this study, both the simulation and the experiment will consider a specimen surface which has been divided into pixels or pixel groups (averaged over square groups of pixels).

The state is binary, where each pixel is either undamaged or damaged. $X_{ij} = 0$ means that a pixel in row $i$ and column $j$ does not contain damage. $X_{ij} = 1$ means that a pixel in row $i$ and column $j$ does contain damage. For each pixel or pixel group, the most likely state estimate is $\hat{X}_{ij}$ and the sensor measurement is $Z_{ij}$. For example, $\hat{X}_{ij}[t_k] = 1$ means that the most likely state estimate at time step $t_k$ for a pixel in row $i$ and column $j$ is that the pixel is damaged. By estimate, we mean our best guess as to the state of that pixel at that moment in time. The most likely state estimate, like the state, is binary and can be either 1 or 0. Recall, that the state and the state estimate are not the same thing. The state estimate is a random variable with a probability distribution based on prior knowledge and measured data. The most likely estimate of the state is determined by the probability of the pixel being damage. For each pixel or pixel group, $P(\hat{X}_{ij}[t_k] = 1)$ is the probability that a pixel in row $i$ and column $j$ is estimated to be damaged at time $t_k$. This probability is the critical value that we will tracking. The probability of the pixel being undamaged, $P(\hat{X}_{ij}[t_k] = 0)$, is equal to $1 - P(\hat{X}_{ij}[t_k] = 1)$. We determine the state estimate based on the probability which is changing at each time step.

Just as with the filtering process presented at the beginning of this chapter we are going through a process of initialization and then a cycle of prediction and updating for each time step. The process to track the part’s condition over time is described by Algorithm 1.
Algorithm 1 The general algorithm used for tracking material state.

1: Initialize Probability of damage of each pixel, \( P(\hat{X}_{ij}[t_0] = 1) \), with any prior knowledge or the first sensor measurement.

2: while \( t_k < T \) do

3: Prediction of State Estimate Probability (Equation 2.26)

   Let \( \mu_p \) be the mean probabilities for the neighboring pixels and \( m_{ij} \) be an adjustable input value for damage growth.

4: if \( \mu_p > P(\hat{X}_{ij}[t_{k-1}] = 1) \)

5: \( P^- (\hat{X}_{ij}[t_k] = 1) = \mu_p + m_{ij} \)

6: else

7: \( P^- (\hat{X}_{ij}[t_k] = 1) = P(\hat{X}_{ij}[t_{k-1}] = 1) + m_{ij} \)

8: end if

9: \( Z_{ij}[t_k] \) is the measurement of damage that occurs at time \( t_k \) and is either 1 or 0.

10: Update the probability of damage using the sensor measurement (Equation 2.27).

   Let the probabilities of the sensor measurements, \( P(Z_{ij}[t_k]) \), be determined by the sensor (see Table 2.2).

11: \( P(\hat{X}_{ij}[t_k] = 1|Z_{ij}[t_k]) = \frac{P(Z_{ij}[t_k]|X_{ij}[t_k]=1) \cdot P^- (\hat{X}_{ij}[t_k]=1)}{P(Z_{ij}[t_k])} \)

12: The updated probability will be used in the next time step to predict the new probability. \( P(\hat{X}_{ij}[t_k] = 1) = P(\hat{X}_{ij}[t_k] = 1|Z_{ij}[t_k]) \)

13: \( k = k + 1 \)

14: end while

Line 1 of the algorithm initializes the the probability of damage for each pixel. For the simulation presented at the end of this chapter and the application to flash thermography in the next, initialization is done with an initial measurement but can also be done with any prior knowledge about the probability of damage.
Lines 3-8 are the prediction step which provides the a priori estimate of pixel’s damage state based on the estimate from the previous step. This step uses the reduced dimensional approach where the predicted probability is only based on the mean probability of damage of the neighbors, $\mu_p$.

Lines 10-11 are the update step. This is not an update that uses the Kalman Gain, instead this step is a direct implementation of Bayes’ theorem. From the sensor measurement we can directly define the total probability of the measurement, $Z$, and the conditional probability of the measurement.

### 2.6.1 Prediction step details

In lines 3-8 in Algorithm 1, the probability is propagated between time steps to an a priori probability,

$$P^{-} \left( \hat{X}_{ij} [t_k] = 1 \right) = f \left( P \left( \hat{X}_{ij} [t_{k-1}] = 1 \right) \right), \quad (2.25)$$

where $f(\cdot)$ is a damage evolution function, predicting the changed in probability of damage from time $t_k$ to time $t_{k+1}$. For our case, this is a naive function that increases the probability of damage if adjacent pixels have higher probabilities of damage. It does not decrease the probability of damage, however, because damage would never decrease. The function used in this study is an average of the surrounding pixels’ probability of damage. If a certain pixel is surrounded by pixels with a higher probability of damage then it can be assumed that this pixel will be more likely to become damaged as the part experiences normal loading. This naive growth model is similar to that used to estimate multiple sclerosis lesion growth in brain MRI scans over time [Karimaghooloo et al. (2012)]. Subsurface delaminations in composite materials are unlikely to arise spontaneously under compression but are much more likely to grow from previously existing damage. For a specific part, for which more details are known about its in-service loads, there may be better, more detailed, damage evolution models available.
Here is the naive model that is being used, where \( f(\cdot) \) from Equation 2.25 now defined as:

\[
P^-(\hat{X}_{ij}[t_k] = 1) = \begin{cases} 
\mu_p + m_{ij}, & \mu_p > P(\hat{X}_{ij}[t_{k-1}] = 1) \\
= & P(\hat{X}_{ij}[t_{k-1}] = 1) + m_{ij}, & \text{otherwise}
\end{cases},
\]

(2.26)\] where \( \mu_p \) is average probability of damage of all 8 neighboring pixels to pixel \( ij \). By using the average of the surrounding pixels, the model is assuring that areas with low probability of damage that are adjacent to areas with high probability of damage are more likely to become damaged. The input, \( m_{ij} \), is a positive value that slightly increases the probability at each time step, as we will always assume that the probability of damage will slightly increase when loading is applied. The input may vary with location or be a constant slight increase in probability for all pixels. Damaged areas in composite materials arise in high-stress areas and then tend to grow, meaning that areas next to damage are likely to become damaged.

Much like Cressie and Wikle’s approach presented above, Equation 2.26 is a reduced dimensional approach where \( \mu_p \) is limited to the eight pixels that are directly adjacent to the the pixel of interest. In this naive dynamic model, instead of using the exact approach of the spatio-temporal Kalman filter in Equation 2.17, where the basis functions, \( \phi \), and then new hidden random processes, \( a \), are found based on a subset of locations; the variable being tracked is the probability and the subset is the neighbors. This dynamic model is only operating on the probability, as the state is binary. The two most important aspects to the dynamic model are that probability of damage increases with loading by \( m_{ij} \) and the the probability of damage increases if neighboring pixels have a higher probability of damage, based on the mean of the neighbors, \( \mu_p \).\footnote{In this case, the basis functions are just time spatial variables that result in the averaging of the adjacent pixels and the temporal variable is the input.} \footnote{For our purposes, we are assuming the each sensor measurement is independent, although in some cases where measurements are taken at a rate where the change in the state in insignificant they cannot be assumed to be independent. In this case, the measurements are taken after enough change in the state has occurred.}
2.6.2 Update step details

Lines 10-11 in Algorithm 1 are the update step. This is not an update that uses the Kalman gain; instead, this step is a direct implementation of Bayes’ theorem as

\[
P \left( \hat{X}_{ij} [t_k] = 1 | Z_{ij} [t_k] \right) \\
= \frac{P (Z_{ij} [t_k] | X_{ij} [t_k] = 1) P^{-} \left( \hat{X}_{ij} [t_k] = 1 \right)}{P (Z_{ij} [t_k])},
\]  

(2.27)

where the total probability of the measurement is

\[
P (Z_{ij} [t_k]) = \\
P (Z_{ij} [t_k] | X_{ij} [t_k] = 1) P^{-} \left( \hat{X}_{ij} [t_k] = 1 \right) \\
+ P (Z_{ij} [t_k] | X_{ij} [t_k] = 0) P^{-} \left( \hat{X}_{ij} [t_k] = 0 \right).
\]  

(2.28)

We can define the total probability of the measurement, \( P (Z_{ij} [t_k]) \), and the conditional probabilities of the measurement based on a statistical analysis of the specific sensor capabilities. These have to do with the probability of detection and the accuracy of the sensor. Table 2.1 shows the definitions of the conditional sensor probabilities.

Table 2.1: Probabilities associated with the general sensor.

| Probability of Detection | \( P (Z_{ij} [t_k] = 1 | X_{ij} [t_k] = 1) \) |
|---------------------------|------------------------------------------------|
| Probability of Miss       | \( P (Z_{ij} [t_k] = 0 | X_{ij} [t_k] = 1) \) |
| Probability of Accurate No Damage | \( P (Z_{ij} [t_k] = 0 | X_{ij} [t_k] = 0) \) |
| Probability of False Alarm | \( P (Z_{ij} [t_k] = 1 | X_{ij} [t_k] = 0) \) |

state that they can be assumed to be independent.
2.7 Simulation results

We developed a simulation to test the response of Algorithm 1. The model had three elements: $\hat{X}_{ij}$, the state estimate of the part at each pixel, $X_{ij}$, the true state of the part at each pixel; and $Z_{ij}$, the sensor measurements at each pixel. The structure of the part consisted of a matrix of 40 by 60 pixels.

The truth was initialized with a small area of damage that was randomly generated in a five pixel by five pixel area at the center of the part. At each time step the damage grew randomly to adjacent pixels. The sensor measurement at each time step was equal to the truth corrupted by random noise.

We designed our model to have a very low rate of false alarms and a moderately good probability of detection in order to be similar to the actual nondestructive evaluation measurements that were used in the experiment. The sensor probabilities (shown in Table 2.2) could be adjusted easily.

| Probability of Detection | $P(Z = 1|X = 1) = 0.95$ |
|--------------------------|--------------------------|
| Probability of Miss      | $P(Z = 0|X = 1) = 0.05$   |
| Probability of Accurate No Damage | $P(Z = 0|X = 0) = 0.9999$ |
| Probability of False Alarm | $P(Z = 1|X = 0) = 0.0001$ |

The state estimate, $\hat{X}$, went through two steps at each time increment. The state estimate was found using Algorithm 1. At the very beginning, it was initialized with the first sensor measurement and the associated probability, $P\left(\hat{X}_{ij}[t_0] = 1\right) = P\left(Z_{ij}[t_0]|X_{ij}[t_0] = 1\right)$. Equation 2.26 was used to predict the increase, between time steps, in a pixel’s probability of being designated as damaged. When a new sensor measurement was available, we used Bayes’ theorem, as stated above, to update the probability. If the $P\left(\hat{X}_{ij}[t_k] = 1\right) > 0.5$ for any pixel then $\hat{X}_{ij}[t_k] = 1$, the pixel was damaged.
Figures 2.4 and 2.5 show the initial and final truths, estimates, and sensor measurements for the simulation. This particular simulation had a fairly fast rate of growth, meaning that significant growth occurred during the simulation.

As expected, Figure 2.6 shows that the state estimate out-performed the stand-alone sensor measurements. The performance was evaluated by the percent of correctly identi-
fied (damaged as damaged and undamaged as undamaged) pixels by either the estimate or the sensor.

Figure 2.6: A comparison of percent error in the sensor and the estimate, both with respect to the truth.

Figure 2.7 shows error in $Z$, the sensor measurement, minus error in $\hat{X}$, the state estimate, for 100 simulations. A positive number means that error in stand-alone sensor measurement is greater than error in state estimate, so the estimate is performing better than the stand-alone sensor measurement. In the majority of time steps in the simulations the estimate outperforms the stand alone measurement. Notice that the difference in the error is also moving in a positive direction, meaning that not only is the state estimate performing better then the stand alone sensor but that as time goes by the amount by which it is doing better is increasing.
2.7.1 Estimator Convergence

Damage evolution is an unbounded process, proving convergence may not be possible. Specifically, the error, $\epsilon_X = X_{ij}[t_k] - \hat{X}_{ij}[t_k]$, may not be driven to zero in all cases. We can see, from the Monte Carlo simulation, it is clear that the error for the estimate is smaller than the error for the stand alone sensor, $\epsilon_Z = X_{ij}[t_k] - Z_{ij}[t_k]$. Not only is the estimate error smaller but the amount by which it is smaller is increasing over time. This is not convergence per se, but a demonstration that there is improvement in the estimate with respect to the sensor as the estimation process continues. Additionally, if we bound the input, meaning that growth will eventually stop, then the estimate must converge with the truth because we will eventually reach a steady state.
CHAPTER 3. APPLYING SPATIO-TEMPORAL DATA TRACKING TO FLASH THERMOGRAPHY DATA OF COMPOSITE STRUCTURES

3.1 Introduction

Composite materials are fast becoming a staple in aerospace design due to their high strength to weight ratio. However, with new innovations comes new danger, as these materials, which are fabricated from layers of fiber in a cured matrix, may sustain damage that cannot be identified from surface inspection. Subsurface delaminations in composite materials are difficult to detect inexpensively. Failure to detect such delaminations may result in loss of mission or loss of life. Subsurface delaminations caused by impacts may result in a significant loss of load carrying capacity, but leave little to no indication of damage at the point of impact or in surrounding areas [Maierhofer et al. (2014)]. Subsurface delaminations, along with flaws present from fabrication, may also grow over the lifetime of the part due to normal loads [Kim and Lee (1997); Ogasawara et al. (2013); Greenhalgh (1993); Lafayette (1989)].

To demonstrate that the process proposed can be successful in a real-world application, we designed an experiment where composite panels were made and impacted. The resulting subsurface delaminations were grown using compression loading. The test specimens were manufactured in the Aerospace Engineering Composites Lab at Iowa State University. The specimens were constructed using Toray CA unidirectional prepreg (T800SC fiber and 3900-2B Resin) [Van Ee and Poursartip (2009)]. Each specimen was
four inches by six inches with the layup $[[0/45/90/−45]_s]$, where the $s$ means that the pattern was symmetric. The total number of plies was 16.

Figure 3.1 shows one of the specimens used in the experiment. The small area of gray in the center of the specimen is wax residue from where the acoustic emissions probe was adhered during compression loading. It is not a visible indication of damage.

![Figure 3.1: One of the composite specimens used in this experiment.](image)

Prior to impacting, after impacting, and after each observed growth in delamination, flash thermography data was collected to be used as the sensor measurement in the state tracking process. Computed tomography (CT) data was also collected at each time step to be used as ‘ground truth’ in order to evaluate the error in the state estimate and the sensor measurement at each time step and assess the quality of both. CT is not required for the Bayesian filter. Error is the percentage of total pixels that were misidentified.

The process that was used for the state tracking was the algorithm present in Algorithm 1 in Chapter 2. This process uses the adjacent pixels’ probability of being damaged to infer the probability of a pixel becoming damaged. After predicting the damaged state of a pixel, that value is then updated by using sensor data from flash thermography and Bayes’ theorem. The cycle is repeated at each time step.
3.2 Damage initiation and growth

Subsurface delaminations were initially induced by impacting the specimens per ASTM D7136 [American Society of Testing and Measurement (2014a)] using the apparatus shown in Figure 3.2 [Davies and Zhang (1995)].

Figure 3.2: The ASTM D7136 impact apparatus located at the Center for Nondestructive Evaluation at Iowa State University.

The goal was to load the specimens in a way to cause observable but incremental damage growth; the damage growth should be significant enough that it is observable by flash thermography but not so extensive that the process cannot be repeated (i.e. catastrophic failure). An acoustic emissions probe was attached to the specimen during loading to monitor delamination growth [Grandin (2014)]. It was assumed that damage had grown after a flurry of acoustic emissions followed by silence. This is known as the felicity ratio [Kim and Lee (1997)].
Initially, the specimens were placed in an edgewise compression frame (Figure 3.3) and loaded in compression in the long dimension as per ASTM D7137 [American Society of Testing and Measurement (2014b)]. This loading method was selected because there was evidence in the literature that this method would result in the desired damage growth [Kardomateas et al. (1995); Davidson (1990)]. The edgewise compression did not produce noticeable, incremental damage growth at the impact location; it did, however, result in some damage growth at the edges, seen in Figure 3.12. The damage area on the bottom edge and the triangle shape near the bottom right side are not a result of the impacting but instead were caused during the compressions after impact loading. It was necessary to look at other loading configuration to obtain the desired damage growth.

![Figure 3.3: The ASTM D7137 Edgewise compression frame used to grow delaminations.](image)

The specimens were then loaded in transverse compression [Doxsee et al. (1993)] and then in four-point bending, as shown in Figures 3.4 and 3.5 respectively.
Figure 3.4: The transverse compression setup with acoustic emissions probe used to grow delaminations.

Figure 3.5: The four-point bending setup with acoustic emissions probe used to grow delaminations.
Both new loading configurations showed observable and incremental damage growth. The four-point bending loading configuration was selected for all the following time points for all the specimens because the acoustic emissions probe could be placed over the impact point and there was no additional surface damage.

### 3.3 Flash thermography measurement

Flash thermography is a relatively simple and cost effective NDE technique where a part is heated by flash lamps and a thermal camera collects temperature data as the part cools. Flash thermography is easy to use in the field as it only requires a heat source and a thermal camera. The data can also be analyzed fairly quickly. The flash thermography system used in this experiment consisted of two Speedotron 206VF flash lamps with 7 inch reflectors and a Flir A35sc thermal camera with a resolution of 320x256 pixels. The setup can be seen in Figure 3.6.

![Figure 3.6: The setup of the flash thermography system where two flash lamps are pointed at the specimen located in the center and the thermal camera, which collected temperature data as the specimen cooled, is located at the lot of the image.](image-url)
Flash thermography indications are areas where the cooling curves of adjacent pixels are different which make it ideal for observing delaminations which are discontinuities in interlaminar bonds.

A part is evaluated by comparing the cooling curves of each pixel and identifying discontinuities. Discontinuities are most apparent when looking at the concavity of the cooling curves (i.e. the second derivative of log temperature versus log time) [Shepard (2002, 2007)]. For homogeneous, isotropic materials the time at which the slope of the log temperature versus log time curve transitions from -0.5 to 0 correlates to the thickness of the material. Identifying thickness variations in these materials is fairly simple. In composite materials (non-homogeneous, non-isotropic), the correlation is not as strong [Maierhofer et al. (2014); Avdelidis et al. (2004, 2003)]. In this study we are not looking for thickness variation in the traditional sense. We are looking for subsurface delaminations which have a very complicated structure in three dimensions.

Figure 3.7 compares the cooling curves ($\log_{10}$ temperature versus $\log_{10}$ time) of a damaged pixel and an undamaged pixel, and Figure 3.8 does the same for the second derivatives of the same pixels.
Figure 3.7: A comparison of the $\log_{10}$ temperature curve of a single damaged pixel and a single undamaged pixel.
Figure 3.8: A comparison of the second derivative of the $\log_{10}$ temperature curve with respect to time of a single damaged pixel and a single undamaged pixel.

An example of how damage is visible in the second derivative of the cooling curves can be seen in Figure 3.9. Figure 3.9 shows an image of the second derivative of the cooling curve at a time point where the damage is visible. This data is typical of that collected during the study. From Figure 3.8, we see that there are many changes in concavity and the most noticeable variation from an undamaged pixel occurs early in the cooling process (i.e. the delamination is close to the surface).
Figure 3.9: The second derivative of $\log_{10}$ temperature - $\log_{10}$ time curves at a single time point for all pixels. Areas of damage are clearly visible. The image has been undistorted and cropped to include only the specimen.

For the algorithm proposed above, the flash thermography data was converted into a simple detector. First, the thermal image was corrected to remove the fish-eye distortion and the specimen was isolated within the scene. This was done using OpenCV, an open source computer vision software package [OpenCV Dev Team (2013b)]. After each individual measurement was placed into the unique coordinate frame of the specimen (Figure 3.10), correcting for any differences in placement in the original image, each pixel was determined to be damaged or undamaged.
In order to obtain a smooth second derivative, the original cooling curve was fitted with b-splines. Then, the second derivative was taken of that fitted function. For this application, the second derivative was defined at 200 points ($k = 0 - 199$). The time, $k$, was cooling time and should not be confused with time steps in the filtering process which correspond to times during the life-time of the part.

Pixels were designated as damaged by first selecting a pixel that was far away from the impact damage or any edge damage to be considered ‘undamaged’. For each pixel, the sum squared error for that pixel as compared to the undamaged pixel was calculated. This compressed the $x$, $y$, and time data into just $x$ and $y$. The sum squared error was
calculated by:

\[ sse_{i,j} = \sum_{k=0}^{199} \left( \frac{d^2 T_{\text{undamaged}}}{dt^2} (k) - \frac{d^2 T_{i,j}}{dt^2} (k) \right)^2, \]  

(3.1)

where \( \frac{d^2 T_{\text{undamaged}}}{dt^2} (k) \) is the second derivative of the temperature of an undamaged pixel at cooling time \( k \), \( \frac{d^2 T_{i,j}}{dt^2} (k) \) is the second derivative of the temperature of pixel \( i,j \) at cooling time \( k \), and \( k \) is defined for time steps 0-199.

The sum squared error data was then used in two ways. We set a threshold for sum squared error past which a pixel was considered damaged, but we also used this varying value as our probability of the sensor reading, \( P(Z = 1 | X = 1) \). Although a pixel may have fallen below a threshold, if it was closer to the threshold then it had a higher probability of being damaged than a pixel that was far below the threshold. Figures 3.11 and 3.12 show the sum squared error for both sides of a specimen at the same time step in the study.
Figure 3.11: The sum squared error with respect to the ‘truth’ for the top of specimen E-5 at time step $k = 3$. 
Figure 3.12: The sum squared error with respect to the ‘truth’ for the bottom of specimen E-5 at time step $k = 3$.

By selecting a threshold of $sse_{i,j} = 10$, then every pixel with $sse_{i,j} > 10$ was considered damaged and every pixel with $sse_{i,j} < 10$ was considered undamaged, but each pixel did not then have the same probability of damage. The probability of damage is defined as a value between 0 and 1 given by

$$P(Z = 1|X) = \frac{sse_{i,j}}{threshold}. \quad (3.2)$$

Thresholds were set for each specimen due to the specimen to specimen variability.

For this study, a total of seven sensor measurements were collected for each specimen over seven time steps. The first sensor measurement collected after impacting the specimen was $Z[t_0]$ and was also the initial state estimate, $X[t_0]$. The other six sensor measurements were each collected after there was additional delamination growth due to
compression with the final measurement being $Z[t_6]$. These sensor measurements were used to update the state estimate in the same manner as the simulation.

### 3.4 Computed tomography as ‘truth’

To evaluate the quality of the state estimates and the sensor measurements, we required a ‘truth’. In the real world, the truth is unknowable. We must instead utilize a much higher quality NDE measurement. Computed tomography (CT) uses X-ray data collected from as many angles as possible and then uses back projection reconstruction to create a three dimensional voxel model of the scanned part.

Although CT can yield high resolution models of the entire structure of a part, it is not a NDE method that is easily applicable to parts in service, as it requires an x-ray source and detector and a method to precisely rotate the part to collect the radiographs required. CT is also time consuming, as many images need to be taken and the reconstruction process requires significant computational resources. The CT data used in this study was collected at CNDE using the setup seen in Figure 3.13.
Figure 3.13: The computed tomography setup at the Center for Nondestructive Evaluation, with the X-ray source, detector, and rotating platform.

Figure 3.14 is an example of what a single slice from a CT reconstruction looks like.
Figure 3.14: A single slice of computed tomography reconstruction.

The through-thickness minimum (i.e. the minimum for each surface pixel) of a few adjacent in-plane slices from the CT reconstruction were used to determine the truth. The flash thermography measurement corresponds to the first few layers of the composite specimen, therefore, we could not use a single slice of the CT reconstruction; nor could we use the entire reconstruction because, as seen in Figures 3.11 and 3.12, the flash thermography data for the top and bottom of a specimen were very different. It was necessary to compress the information from a few slices of the CT reconstruction into a single two-dimensional data set. We were primarily concerned with delaminations, or empty space, in the specimen, so by first taking the minimum density for each pixel through the thickness of the first few slices we compressed the data. Then, by setting a threshold to distinguish between material and space, we were able to observe the
delaminations in any of those slices. This data was taken for the top of the specimens so it included the surface indentation made from impacting. This empty space was not reflected in the flash thermography data. Figure 3.15 shows the results of this process.

Figure 3.15: The minimums of a few slices of the CT reconstruction of E-5 converted to space or composite.

In order to compare this truth with the estimate and sensor measurement, we had to resample the data to make it the same size. This was done by binning pixels, where blocks of pixels were averaged. If the average was greater than 0.5 then the new pixel that represents the pixel block was assigned to be damaged. The size of the pixel blocks varied slightly from scan to scan because the CT reconstruction for each scan never had exactly the same magnification as another scan.

The results of the binning are shown in Figure 3.16. This process also served to reduced the noise in the data set. The ‘truth’ was found for each specimen at each time increment.
3.5 Specific algorithm implementation

In Algorithm 1, the sensor measurement is processed to be the sum squared error of each pixel compared to an undamaged pixel, as described in Equation 3.1. The decisions that $Z_{ij}[t_k] = 1$ or $Z_{ij}[t_k] = 0$ is based on some threshold for the error. Now let us assume that a specific pixel does not quite meet the threshold and is therefore determined to be undamaged, but it is much closer to the threshold than some other pixels. It would then make sense that $P(Z_{ij} = 1)$ is higher. Specifically, the probability of a miss (meaning $P(Z_{ij} = 0|X_{ij}[t_k] = 1)$) is determined by the the sum squared error as is the probability of correctly determining that the pixel is damaged (meaning $P(Z_{ij} = 1|X_{ij}[t_k] = 1)$).

3.6 Experiment results and conclusions

Figure 3.17 shows the percent of incorrectly identified pixels, as compared with the CT truth, for both the estimate and the sensor for the E-1 Specimen.
Figure 3.17: Comparing error in the sensor and the estimate for specimen E-1 results.

Figures 3.18 - 3.21 shows the same error metric for specimens E-3, E-4, E-5, and E-6.

Figure 3.18: Comparing error in the sensor and the estimate for specimen E-3 results.
For the five specimens where results could be used, only for one specimen (E-3) did the individual sensor measurement provide a more accurate estimate of the true state than the Bayesian estimate. This occurred at some time steps but not all, as seen in Figure 3.18.

![Graph comparing sensor and estimate errors for specimen E-4 results.](image)

Figure 3.19: Comparing error in the sensor and the estimate for specimen E-4 results.
Figure 3.20: Comparing error in the sensor and the estimate for specimen E-5 results.

Figure 3.21: Comparing error in the sensor and the estimate for specimen E-6 results.
Figures 3.22 and 3.23 show the truth, the sensor measurement, and the estimate for specimen E-1 at time increments 2 and 6, respectively. From these figures, we see that the estimate has filtered out some of the error associated with the sensor measurement.

Figure 3.22: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-1 at time step \( k = 2 \).

Figure 3.23: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-1 at time step \( k = 6 \).

Figures 3.24 - 3.31 show the same results for each specimen at an early and a late time step. The number of damaged pixels is growing over time and for most of the specimens that estimate is better tracking the truth than the stand-alone sensor measurement.
Figure 3.24: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-3 at time step $k = 2$.

Figure 3.25: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-3 at time step $k = 6$. 
Figure 3.26: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-4 at time step $k = 2$.

Figure 3.27: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-4 at time step $k = 6$. 
Figure 3.28: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-5 at time step $k = 2$.

Figure 3.29: Comparing the truth, the sensor measurement, and the Bayesian estimate for specimen E-5 at time step $k = 6$. 
This study demonstrated that utilizing Bayesian filtering techniques with NDE data yields better state estimates than the most recent NDE measurement, as is currently the practice in the aerospace industry. Figures 3.17 - 3.21 shows that for 4 out of 5 specimens, the estimate outperforms the stand alone sensor at all time steps and for specimen E-3, the estimate outperforms the stand alone sensor for most time steps.

This was accomplished by using a naive damage evolution model. Results using part specific damage evolution models would yield an even more improved estimate. This type
of data tracking will also lend itself to building and improving those damage evolution models. This study is clear evidence for the further development of state tracking systems in aerospace non-destructive evaluation.
CHAPTER 4. UNCERTAINTY IN IMAGE REGISTRATION

4.1 Introduction

In this chapter, we will discuss the error or uncertainty that is associated with estimating and tracking a material state based on nondestructive evaluation (NDE) data. The data used for the study in Chapter 3 was image data. The discussion presented in this chapter will be shaped by the unique distributions associated with using image data and registering that image data with a three dimensional model of a specific part. We will discuss ways to minimize and facilitate the non-Gaussian error distributions.

4.2 Spatial awareness and registration

In order for nondestructive evaluation data to be used to update a three dimensional model of a part, we need a way to register the nondestructive evaluation image data with the model automatically. When doing this we also need to understand what kinds of uncertainty, variability and error we are introducing into the data as a result of the registration. In this context “automatic” means that the task should be accomplished with no additional, user generated, information beyond the image, the camera’s intrinsic model, and the CAD model.

The goal is to automatically determine the most likely transformation from the specimen coordinate frame to the image coordinate frame and the associated probability distribution model for this transformation. In more general terms, we need a method to
quickly identify the specimen in the image. This will then make it possible to describe this uncertainty in a context useful to the state tracking algorithm proposed in Chapter 3. We began by assuming no knowledge at all about the orientation of a specimen (only that it is in front of the camera) or the number of object points that are visible in the image. The problem of automatic registration is not new in the field of computer vision and solutions do exist [Rusinkiewicz and Levoy (2001); Du (2011); Besl and McKay (1992)]. The most common solution is iterative closest point [Gao et al. (2003); Li et al. (2008)].

### 4.2.1 Registering two dimensional data to a three dimensional object, a probabilistic approach

In this section we describe one of the early probabilistic approaches to solving the registration problem. For the state tracking process discussed in chapter 3 we had to register each image to the specimen space. This was done by connecting the points in the thermal image to the the corners of the face of the specimen. That was a manual process, but ideally it would be an automatic process. The fundamental process would still be the same; points in the image would be mapped to points on the object. This process is the same for any image and any object.

First, let us start with some definitions: “Object points” are easily identifiable points associated with the specimen (object) and likely to be detected by the computer vision software in the image. Object points are usually corners, but could also by fiducial marks. These points are likely to be high contrast points in an image and serve as texture mapping points for the three dimensional CAD model. The number of object points is a known value $N$ (where $N \geq 4$). The set of object points is defined by the object and does not change.

“Image points” are high contrast points that are identified in the image. These may correspond to the object points, but they may also correspond to other locations on the
object or something else in the frame. When they do not correspond to the object points we will refer to them as “noise points”. The number of image points is \( k \) (where \( k \geq 4 \)). Image points are the complete set of points found by the computer vision software. For this study, the computer vision software used was OpenCV [OpenCV Dev Team (2013a)], an open source computer vision software package.

“Visible points” are the subset of image points that correspond to object points. The number of these points, \( n \), is initially unknown. In practice, the algorithm requires at least four visible points to correctly determine a transformation from the object frame to the image frame.

Figure 4.1 shows a VENN diagram of how these points are related. Image points are from the image and may or may not be associated with object points. When image points are associated with object points, they are visible points.

![Venn diagram of image, object, and visible point definitions.](image)

Figure 4.1: Illustration of image, object, and visible point definitions.

The mathematical model that connects points in the image to points on the object is a mapping that transforms a point from the object coordinate frame into the camera coordinate frame and then projects it on to the image plane,
\[
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}
= 
\begin{bmatrix}
    f_x & 0 & c_x \\
    0 & f_y & c_y \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_1 \\
    r_{21} & r_{22} & r_{23} & t_2 \\
    r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix},
\] (4.1)

Equation 4.1 shows how a point in three dimensional space, \([X, Y, Z, 1]^T\), is transformed to image space, \([u, v, 1]^T\). Where the projection matrix,

\[
A = \begin{bmatrix}
    f_x & 0 & c_x \\
    0 & f_y & c_y \\
    0 & 0 & 1
\end{bmatrix},
\] (4.2)

consists of the \(x\) and \(y\) focal lengths \(f_x\) and \(f_y\) and the \(x\) and \(y\) coordinates of the principal point \([c_x, c_y]\). The transformation matrix,

\[
T = \begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_1 \\
    r_{21} & r_{22} & r_{23} & t_2 \\
    r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix},
\] (4.3)

consists of a rotation and a translation from the object frame, \([X, Y, Z, 1]^T\), to the camera frame, \([x, y, z, 1]^T\).

The total number of possible mappings for \(k\) visible points, \(m_k\), is the number of combinations for the visible object points multiplied by the number of permutations of visible image points,

\[
m_k = \frac{N!}{k!(N-k)!} \cdot \frac{n!}{(n-k)!}.
\] (4.4)

The total number of mappings possible, \(M\), is the sum of all the mappings for all the possible number of visible points and can be written as,

\[
M = \sum_{k=1}^{\min[N,n]} m_k = \sum_{k=1}^{\min[N,n]} \frac{N!}{k!(N-k)!} \cdot \frac{n!}{(n-k)!}.
\] (4.5)
For a trivial example, let us assume we have an object with four points ($N = 4$) and image of that object with three identifiable high contrast points ($k = 3$). We shall call the object points $P_1$, $P_2$, $P_3$, and $P_4$, and the image points $p_a$, $p_b$, and $p_c$. Now if we have $n = 3$ visible points, meaning all the image points correspond to an object point, there are four combinations of object points and six permutations of image points. The combinations of object points are:

- $[P_1, P_2, P_3]$
- $[P_1, P_2, P_4]$
- $[P_1, P_3, P_4]$
- $[P_2, P_3, P_4]$

The six permutations of the image points are:

- $[p_a, p_b, p_c]$
- $[p_a, p_c, p_b]$
- $[p_b, p_a, p_c]$
- $[p_b, p_c, p_a]$
- $[p_c, p_a, p_b]$
- $[p_c, p_b, p_a]$

Each of the four combinations could map to each of the six permutations, resulting in $m_3 = 24$ possible mappings. For $n = 2$, $m_2 = 36$. For $n = 1$, $m_1 = 12$. There is also the trivial mapping of no points to no points, $n = 0$, that we will omit. This sums to $M = 72$ possible mappings.
These questions must be answered:

- How do we compare the possible mappings and choose the ‘correct’ one?
- How does a mapping translate into the alignment of an image to a three-dimensional model?
- Under what circumstances does the process provide a good solution and under what circumstances does it provide a bad solution?

4.2.2 Evaluating mappings

Once we have a set of mappings, we need to evaluate which mapping is the most likely. To do this, we first need to find the transformation from the object coordinate frame to the image coordinate frame as described by Equation 4.3.

We assume that we know the projection matrix for the camera (also called the intrinsic matrix), as it was determined during calibration, so we need only solve for the transformation matrix. This problem is known as the perspective n-point camera pose problem, or PnP, where \( n \) is the number of points for which there is data. In our case this would be the number of visible points.

Once we have a transformation matrix associated with each mapping, then we use the transformation matrix and the camera matrix to find the projections of the object points from the mapping, from Equation 4.1. Then we compare those projections to the original image points from the mapping. The mean squared error between the projected points and the image points is used to determine a weight or probability of a particular mapping. We assume that the mapping with the highest probability is the correct mapping.

4.2.3 Probabilistic method

The OpenCV function called SolvePnP [OpenCV Dev Team (2013b,a)] finds the transformation from object to camera frame that minimizes the projection error of
the points. The algorithm iterates over the transformation space using a Levenberg-Marquardt optimization algorithm [Marquardt (1963); Triggs (1999); Horaud et al. (1989)]. SolvePnp is used to find a transformation for each possible mapping.

Figure 4.2 shows how the transformation and projection process works. Points that exist in the object frame (or it can also be considered the world frame) are transformed into the camera frame and then projected onto the image plane.

![Figure 4.2: The camera frame, image frame and object frame for a single scene.](image)

Solving the PnP problem for every possible mapping is time consuming. For \(N = 8\) object points, \(n = 10\) image points, and the visible points, \(4 \leq k \leq 8\), the total number of mappings, as defined by Equations 4.5 is:

\[
M = \sum_{k=4}^{8} \frac{8!}{k!(8-k)!} \cdot \frac{10!}{(10-k)!} = 12,932,640,
\]

a significant number of mappings.

This number can be greatly reduced by using any apriori knowledge, such as:

1. Instead of \(4 \leq k \leq 8\) determine that \(k\) is a specific number, such as \(k = 7\) or \(k = 5\).

2. Place some restriction on the way points must fit together in the mappings. Instead
of every combination being tested with every permutation we might establish some rules about which combinations can be associated with which permutations.

Both of these tasks can be accomplished by including some information about the edges of the specimen in the problem definition. For example, in addition to object points we can include which of these object points are connected by straight edges.

Let us go back to the simple example used above, but instead of four object points there are eight object points and instead of three image points let us have ten. We will call the object points \( P_1 \) through \( P_8 \), and the image points \( p_a \) through \( p_{10} \). If we can add some information such that there are edges connecting \( P_1 \) and \( P_2 \), \( P_2 \) and \( P_3 \), \( P_3 \) and \( P_4 \), \( P_4 \) and \( P_1 \), and so on; and if, additionally, we can also determine from the image that there are probably edges connecting \( p_a \) and \( p_b \), then we can determine if \( P_1 \) maps to \( p_a \) then \( p_b \) should map to a point that shares an edge with \( P_1 \). That will allow us to discard a number of unlikely mappings. It will also allow us to better estimate the number of visible points. When all the object points are associated with at least one edge, then we can assume that an image point that has no associated edges is less likely to be a visible point.

This methodology falls short when the algorithm fails to detect an edge or accidentally detects an edge between two points where there is none. That is why this should be used as means of providing probabilities not merely eliminating all mappings that do not correspond to the rules.

These assumptions allowed us to reduce a 46 minute run time for one camera to a 40 second run time for two cameras. This can be further reduced as, right now, it is only using the best guess of visible points from edge detection and not the full edge description, that is, it is not discarding mappings based on edge relationships.

After we reduced the pool of possible mappings, we compared them by using the means squared error of the projected object points with respect to the image points. The mean squared error of the mapping is a random variable with a Gaussian distribution,
the mean of the distribution is 0 but the variance is determined by the error that we perceive in both the image (for example one or two pixels) and the object points (a few mm). The probability for the given mean squared error is the weight.

In summary, the order of the process is:

1. Define object points in three dimensions and defined edge connections.

2. Use OpenCV to find image points and edges in the image.

3. Create a pool of possible mappings (in order of probability).

4. Solve PnP for each mapping in order to obtain rotation and translation.

5. Use the rotation and translation (and intrinsic values) to project object points to the image plane.

6. Record the mean squared error of the projected points compared to the image points in order to weight the mapping.

7. The highest weighted mapping is the best estimate, and the maximum likelihood estimate.

We took two images with two different cameras in a fixed configuration. We had already completed both individual calibrations and stereo calibration for the two cameras (meaning there already existed a good estimate of the transformation from 'Camera 24' to 'Camera 28'). The images that were acquired are shown in Figures 4.3 and 4.4.
The image points found by OpenCV in both images are shown in Figures 4.5 and 4.6,
Figure 4.5: Camera GEV0824 with image points.

Figure 4.6: Camera GEV0828 with image points.

Edges that were observed are shown in Figures 4.7 and 4.8.

Figure 4.7: Edges identified in GEV0824.
The mappings that were identified were the highest weighted, by far. The weights for each mapping are shown in Figures 4.9 and 4.10.
Using the algorithm, we found the transformations from the object’s frame to each of the cameras. Then we found the transformation from ‘Camera 24’ to ‘Camera 28’ as follows:

\[ R_{24\rightarrow 28} = R_{28} R_{24}^{-1} \]
\[ T_{24\rightarrow 28} = T_{28} - R_{24\rightarrow 28} T_{24}. \]

When we projected the object points onto the images using the transformation to ‘Camera 28’ we found that the results compare favorably with a transformation we get from the results of the stereo calibration:

\[ R_{28} = R_{24} R_{24\rightarrow 28}^{(Stereo)} \]
\[ T_{28} = T_{24\rightarrow 28}^{(Stereo)} + R_{24\rightarrow 28}^{(Stereo)} T_{24}. \]
Figure 4.11 shows the results from the mapping weight solution and Figure 4.12 shows the results using the stereo calibration results only.

The weights give us a useful method of determining which of the poses is the mostly likely. In reality each of the poses has some error associated with it.
4.2.4 Machine vision solution

The solution for automatic detection that was eventually used was an program developed by Rafael Radkowski [Radkowski et al. (2016)] that uses iterative closest point algorithm to detect the specimen in a visible camera image [Besl and McKay (1992), Du (2011), Rusinkiewicz and Levoy (2001)]. Figure 4.13 shows an example of the detection software identifying a specimen with a visible camera.

![Figure 4.13: Example of Radkowski’s program to automatically detect the composite specimens.](image)

The visible camera is calibrated with the thermal camera, meaning that we know the transformation from the visible camera frame to the thermal camera frame so that once the corner points are found in the visible frame, they can be transformed to the thermal image. Figure 4.14 shows the points found by the software, as projected in the
undistributed thermal image and Figure 4.15 shows that image cropped to only include the specimen.

Figure 4.14: The specimen corners as detected by the Radkowski software.
4.3 Tracking variables with more complicated posterior distributions

The majority of the data that was collected for the study in Chapter 3 was collected by manually identifying the specimen corners in the undistorted image. The solution that was developed by Radkowski significantly reduced the error associated with identifying
the object in the image, but the error still exists. This section shows the type of error and uncertainty distributions that arise from both the manual and the Radkowski automatic process.

4.3.1 Explanation of system

The transformation between the camera coordinate frame and the specimen coordinate frame is found with the OpenCV function SolvePnP. This function takes as arguments a set of three dimensional points in the specimen frame, a set of two dimensional points in the image coordinate frame, the camera projection matrix, and any distortion coefficients. For our purposes, all the distortion coefficients were zero because the two dimensional points were gathered from an image that was already corrected for lens distortion.

The three dimensional points were from the specimen database and were the measured locations of the four fiducial marks. The two dimensional points are provided by manually clicking the corners of a specimen in the image. Both actions involve a great deal of human error.

4.3.2 Variable two dimensional coordinates

The first study was complete by only varying the two dimensional image coordinates. A population of 1000 coordinate sets were generated from a Gaussian distribution, using the value actually collected as the mean and a standard deviation of one pixel. The results are shown in Figures 4.16 through 4.19.
Figure 4.16: Rotation results with $N=1000$ and $\sigma_{2D} = 1$, isometric view.

Figure 4.17: Rotation results with $N=1000$ and $\sigma_{2D} = 1$, X-Y plane.
The second study was also completed by varying only the two dimensional image coordinates and using a population of 1000 coordinate sets generated from a Gaussian distribution. However, in this study a standard deviation of three pixels was used. The results are shown in Figures 4.20 through 4.23. This shows a similar, but broader pattern.
Figure 4.20: Rotation results with $N=1000$ and $\sigma_{2D} = 3$, isometric view.

Figure 4.21: Rotation results with $N=1000$ and $\sigma_{2D} = 3$, X-Y plane.
4.3.3 Variable three dimensional coordinates

The third study was completed by varying only the three dimensional object coordinates and using a population of 1000 coordinate sets generated from a Gaussian distribution. Not all of the coordinates were varied. In this particular instance all of the points lie on a the face that defines the x-y plane of the object coordinate frame so the z-coordinate is kept equal to 0. The results are shown in Figures 4.24 through 4.27. The standard deviation of the coordinates is one millimeter.
Figure 4.24: Rotation results with $N=1000$ and $\sigma_{3D} = 1$, isometric view.

Figure 4.25: Rotation results with $N=1000$ and $\sigma_{3D} = 1$, X-Y plane.
Figures 4.26 through 4.31 show the results using a standard deviation of the coordinates equal to three millimeters. The change in standard deviation in three dimensional coordinates has much more of an effect than the change in two dimensional coordinates. The resulting ring in the z axis variability is much more prominent. Although the ring is still not “even” in the sense that the density of distributions is still greater in one part of the ring.
Figure 4.28: Rotation results with N=1000 and $\sigma_{3D} = 3$, isometric view.

Figure 4.29: Rotation results with N=1000 and $\sigma_{3D} = 3$, X-Y plane.
Figure 4.30: Translation results with N=1000 and $\sigma_{3D} = 3$, isometric view.

Figure 4.31: Translation results with N=1000 and $\sigma_{3D} = 3$, X-Y plane

4.3.4 Variable two dimensional and three dimensional coordinates

After varying the object coordinates and the image coordinates independently, they were both varied together. It is comforting that the shapes of the distributions in the first two studies were not drastically different.

Figures 4.32 through 4.35 show the results when $\sigma_{2D} = 1$ pixel and $\sigma_{3D} = 1$mm.
Figure 4.32: Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, isometric view.

Figure 4.33: Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, X-Y plane.
Figure 4.34: Translation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, isometric view.

Figure 4.35: Translation results with N=1000 with $\sigma_{2D} = 1$ and $\sigma_{3D} = 1$, X-Y plane.

Figures 4.36 through 4.39 show the results when $\sigma_{2D} = 3$ pixels and $\sigma_{3D} = 1$mm.
Figure 4.36: Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, isometric view.

Figure 4.37: Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, X-Y plane.
Figure 4.38: Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, isometric view.

Figure 4.39: Translation results with N=1000 with $\sigma_{2D} = 3$ and $\sigma_{3D} = 1$, X-Y plane.

Figures 4.40 through 4.43 show the results when $\sigma_{2D} = 1$ pixel and $\sigma_{3D} = 3$mm.
Figure 4.40: Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, isometric view.

Figure 4.41: Rotation results with $N=1000$ with $\sigma_{2D} = 1$ and $\sigma_{3D} = 3$, X-Y plane.
Figure 4.42: Translation results with \( N=1000 \) with \( \sigma_{2D} = 1 \) and \( \sigma_{3D} = 3 \), isometric view.

Figure 4.43: Translation results with \( N=1000 \) with \( \sigma_{2D} = 1 \) and \( \sigma_{3D} = 3 \), X-Y plane.

Figures 4.44 through 4.47 show the results when \( \sigma_{2D} = 3 \) pixel and \( \sigma_{3D} = 3 \)mm.
Figure 4.44: Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, isometric view.

Figure 4.45: Rotation results with $N=1000$ with $\sigma_{2D} = 3$ and $\sigma_{3D} = 3$, X-Y plane.
4.4 Approximating complex distributions with Gaussian mixtures

As shown in the previous section, it is quite easy to get a very complex error distribution that is in no way Gaussian when looking at the error associated with image registration. In this section, we propose a method for approximating non-Gaussian distributions with Gaussian mixtures.
Let us look at a system with $n_x$ state variables with the state vector $x$ and a sensor vector $z$ with $n_z$ readings:

$$x \sim \sum_{m=1}^{M} p_m \mathcal{N}(\hat{x}_m, P_m^-) \quad (4.6)$$

$$z = Hx + v \quad (4.7)$$

where:

$$f(v) \sim \mathcal{N}(0, R_j)$$

There is a prior estimate of $x$ where the mean is $\hat{x}^-$, which is a mixture of Gaussian distributions, where each mixture has a weight, $p_m$. The nomenclature is taken from Simon [Simon (2006)]. Notice that the prior estimate is denoted by a $-$ superscript. In the paradigm of filtering we would call the mean(s) of the prior our estimate(s). It is not immediately clear why there is more than one Gaussian distribution right now, so we can assume that $M = 1$ and $p_m = 1$.

We also have a sensor. If used properly then it has one Gaussian distribution with the mean $Hx$, but the sensor can also be read incorrectly in $M - 1$ possible ways. Each of those readings has a different (Gaussian) distribution, each with a probability $p_j$. The distributions could also have unique transformations ($H_j$ instead of a single $H$) without changing the derivations.

Now let us define our prior as:

$$f_{X}(x) = \sum_{m=1}^{M} p_m (2\pi)^{-\frac{nx}{2}} |P_m^-|^{-\frac{1}{2}} \exp \left[ -\frac{1}{2} (x - \hat{x}_m)^T (P_m^-)^{-1} (x - \hat{x}_m) \right], \quad (4.8)$$

where

$$\sum_{m=1}^{M} p_m = 1.$$
where

\[ \sum_{j=1}^{J} p_j = 1. \]

So to find the probability density function of the posterior distribution, we can use Bayes’ rule which states:

\[ f_{X|Z}(x|z) = \frac{f_{Z,X}(z,x)}{\int f_{Z,X}(z,\chi) \, d\chi}, \quad (4.10) \]

where \( \int f_{Z,X}(z,\chi) \, d\chi \) acts as the normalizing factor and is not a function of the parameter \( x \). So we can re-write Equation 4.10 as:

\[ f_{X|Z}(x|z) \propto f_{Z,X}(z,x), \quad (4.11) \]

where

\[ f_{Z,X}(z,x) = f_{Z}(z|x) f_{X}(x). \]

So then

\[ f_{X|Z}(x|z) \propto \sum_{j=1}^{J} \sum_{m+1}^{M} p_j p_m \left( |P_m^+| |R_j| \right)^{-\frac{i}{2}} \exp \left[ -\frac{1}{2} B_{jm} \right], \quad (4.12) \]

where

\[ B_{jm} = (z - Hx)^T R_j^{-1} (z - Hx) + (x - \hat{x}_m^-)^T (P_m^-)^{-1} (x - \hat{x}_m^-). \]

Then with a little bit of algebraic manipulation and completing the square of the coefficient we can put it in the form of a new distribution:

\[ f_{X|Z}(x|z) = \sum_{j=1}^{J} \sum_{m+1}^{M} \hat{p}_{jm} (2\pi)^{-\frac{i}{2}} |P_{jm}|^{-\frac{i}{2}} \exp \left[ -\frac{1}{2} (x - \hat{x}_{jm})^T P_{jm}^{-1} (x - \hat{x}_{jm}) \right], \quad (4.13) \]

where

\[ P_{jm} = \left( H^T R_j^{-1} H + (P_m^-)^{-1} \right)^{-1}, \]

\[ \hat{x}_{jm} = P_{jm} \left( H^T R_j^{-1} z + (P_m^-)^{-1} \hat{x}_m^- \right), \]

\[ \hat{p}_{jm} = M p_j p_m \left( |R_j| |P_m^-| \right)^{-\frac{i}{2}} e^{c_i}, \]
\[ c_{jm} = -\frac{1}{2} (z^T R_j^{-1} z + (\hat{x}_m^-)^T (P_m^-)^{-1} \hat{x}_m^- - \hat{x}_i^T \hat{\Sigma}_i^{-1} \hat{x}_{jm}) , \]

and

\[ M = \left( \sum_{i=1}^{NM} p_m p_j (|R_j| |P_m|)^{-\frac{1}{2}} e^{c_i} \right)^{-1} . \]

From this we get new estimates \( \hat{\mu}_i \) and new weights \( \hat{p}_{jm} \), but the number of Gaussians in our mixture went from \( M \) to \( JM \).

4.4.1 Comparing to the Kalman filter equations

Each of the prior distributions in the mixture was updated with each of the sensor distributions according to the Kalman filter equations.

The Kalman filter equations yield:

\[ \hat{x}_k = \hat{x}_k^- + P_k H^T R^{-1} (z_k - H \hat{x}_k^-) P_k = \left( (P_k^-)^{-1} + H^T R^{-1} H \right)^{-1} , \quad (4.14) \]

where \( m \) refers to the distribution and \( k \) is time.

It is easy to see that the covariance matrices are the same but the estimates need a little finagling:

\[ \hat{x}_k = (I - P_k H^T R^{-1} H) \hat{x}_k^- + P_k H^T R_k^{-1} z_k = P_k \left( ((P_k^-)^{-1} - H^T R_k^{-1} H) \hat{x}_k^- + H^T R_k^{-1} z_k \right) . \quad (4.15) \]

Then from the covariance definitions, \((P_k^+)^{-1} = (P_k^-)^{-1} + H^T R^{-1} H\) so:

\[ \hat{x}_k = P_k \left( (P_k^-)^{-1} + H^T R_k^{-1} H - H^T R_k^{-1} H \right) \hat{x}_k^- + H^T R_k^{-1} z_k \right] = P_k \left( (P_k^-)^{-1} \hat{x}_k^- + H^T R_k^{-1} z_k \right) . \quad (4.16) \]

The filtering equations match those presented for the non-mixture case.

4.4.2 Approximating Gaussian mixtures

We have a Gaussian mixture that grows in number of elements at every step. This quickly becomes unfeasible. However, it is likely that some of the elements are weighted
so low that they may not be contributing very much. One way to reduce the number of elements in the mixture is to take the top $N$ highest weighted distributions and then re-weight them proportionally. Another way to reduce the number of elements, is to approximate the new mixture with another mixture with fewer distributions.

Let us define the probability density function of a multi-variant Gaussian distribution as:

$$f(x) = (2\pi)^{-\frac{d}{2}} \det(P)^{-\frac{1}{2}} \exp\left[ -\frac{1}{2} (x - \mu)^T P^{-1} (x - \mu) \right], \quad (4.17)$$

where:

- $d$ is the dimension of the distribution.
- $P$ is the covariance matrix.
- $\mu$ is the mean vector.

A mixture of Gaussian distributions is then:

$$f(x) = \sum_{n=1}^{N} p_n (2\pi)^{-\frac{d}{2}} \det(P_n)^{-\frac{1}{2}} \exp\left[ -\frac{1}{2} (x - \mu_n)^T P_n^{-1} (x - \mu_n) \right], \quad (4.18)$$

where $p_n$ is the weight of the $n$th distribution and $\sum_{n=1}^{N} p_n = 1$.

Let us say we have a mixed distribution made up of $N$ Gaussian distributions, as defined above and we wish to approximate it with a mixed distribution made up of $M$ Gaussian distributions, where $M < N$. The error metric, i.e. the value to be minimized, can be the integral of the squared error over the entire space, $\mathbb{R}^k$:

$$e = \int_{\mathbb{R}^d} (f_N(x) - f_M(x))^2 \, dx. \quad (4.19)$$

Now let us expand:

$$e = \int_{\mathbb{R}^d} \left( \sum_{n=1}^{N} w_n (2\pi)^{-\frac{d}{2}} \det(P_n)^{-\frac{1}{2}} \exp\left[ -\frac{1}{2} (x - \mu_n)^T P_n^{-1} (x - \mu_n) \right] - \right. \quad (4.20)$$
\[ \sum_{m=1}^{M} p_m (2\pi)^{-\frac{d}{2}} \det (P_m)^{-\frac{1}{2}} \exp \left[ -\frac{1}{2} (x - \mu_m)^T P_m^{-1} (x - \mu_m) \right]^2 dx. \]

For simplicity let us say:

\[ A_n = p_n (2\pi)^{-\frac{d}{2}} \det (P_n)^{-\frac{1}{2}} \]

and

\[ A_m = -p_m (2\pi)^{-\frac{d}{2}} \det (P_m)^{-\frac{1}{2}}. \]

Neither \( A_n \) or \( A_m \) is of a function of \( x \), so then:

\[ e = \int_{\mathbb{R}^d} \left( \sum_{n=1}^{N} A_n \exp \left[ -\frac{1}{2} (x - \mu_n)^T P_n^{-1} (x - \mu_n) \right] \right. \]

\[ + \sum_{m=1}^{M} A_m \exp \left[ -\frac{1}{2} (x - \mu_m)^T P_m^{-1} (x - \mu_m) \right] \]  
\[ \left. \right)^2 dx. \]  
\[ (4.21) \]

Now let us define two new indices:

\( i = n \) for \( i \leq N \) and \( i = m \) for \( N < i \leq M \).

\( j = n \) for \( j \leq N \) and \( j = m \) for \( N < j \leq M \).

So then:

\[ e = \int_{\mathbb{R}^d} \sum_{i=1}^{N} \sum_{j=1}^{M} A_i \exp \left[ -\frac{1}{2} (x - \mu_i)^T P_i^{-1} (x - \mu_i) \right] A_j \exp \left[ -\frac{1}{2} (x - \mu_j)^T P_j^{-1} (x - \mu_j) \right] dx. \]

Since this is just an integral of sums we can make it a sum of integrals and move the constants outside the integral by doing the following:

\[ e = \sum_{i=1}^{N} \sum_{j=1}^{M} A_i A_j \int_{\mathbb{R}^d} \exp \left[ -\frac{1}{2} (x - \mu_i)^T P_i^{-1} (x - \mu_i) \right] \exp \left[ -\frac{1}{2} (x - \mu_j)^T P_j^{-1} (x - \mu_j) \right] dx \]

\[ e = \sum_{i=1}^{N} \sum_{j=1}^{M} A_i A_j \int_{\mathbb{R}^d} \exp \left[ -\frac{1}{2} (x - \mu_i)^T P_i^{-1} (x - \mu_i) \right] \exp \left[ -\frac{1}{2} (x - \mu_j)^T P_j^{-1} (x - \mu_j) \right] dx \]

\[ e = \sum_{i=1}^{N} \sum_{j=1}^{M} A_i A_j \int_{\mathbb{R}^d} \exp \left[ -\frac{1}{2} \left[ x^T (P_i^{-1} + P_j^{-1}) x - 2 (\mu_i^T P_i^{-1} + \mu_j^T P_j^{-1}) \right] \right] dx \]

\[ + \mu_i^T P_i^{-1} \mu_i + \mu_j^T P_j^{-1} \mu_j \]  
\[ dx. \]
Let us make some further definitions:

\[ P^{-1}_{ij} = (P^{-1}_i + P^{-1}_j) \] is a new covariance matrix.

\[ \mu^T_{ij} = (\mu^T_i P^{-1}_i + \mu^T_j P^{-1}_j) P_{ij} \] is a new mean vector.

Then the error becomes:

\[ e = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} A_i A_j \int_{\mathbb{R}^d} \exp \left[ -\frac{1}{2} \left[ x^T P^{-1}_{ij} x - 2 \mu_{ij} P^{-1}_{ij} x + \mu^T_{ij} P^{-1}_{ij} \mu_{ij} \right] \right] dx. \]

Let us also say that:

\[ \mu^T_i P^{-1}_i \mu_i + \mu^T_j P^{-1}_j \mu_j + c_{ij} = \mu^T_{ij} P^{-1}_{ij} \mu_{ij} \]

and that \( \exp(c_{ij}) = C_{ij} \), which is not a function of \( x \).

So now the error is:

\[ e = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} A_i A_j \int_{\mathbb{R}^d} \exp \left[ -\frac{1}{2} \left[ x - \mu_{ij} \right]^T P^{-1}_{ij} \left[ x - \mu_{ij} \right] \right] dx \]

Now we have a new Gaussian mixture. Since the integral of a probability density function over its entire space is always equal to one then:

\[ e = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} A_i A_j C_{ij} \left( 2\pi \right)^\frac{d}{2} \det \left( P_{ij} \right)^\frac{1}{2}. \]

Because \( P_{ij} = P_{ji} \) and \( \mu_{ij} = \mu_{ji} \) and therefore \( C_{ij} = C_{ji} \).

We can see that:

\[ e = \sum_{i=1}^{N+M} A_i \left( 2\pi \right)^\frac{d}{2} \left[ \sum_{n=1}^{N} A_n C_{in} \det \left( P_{in} \right)^\frac{1}{2} + \sum_{m=1}^{M} A_m C_{im} \det \left( P_{im} \right)^\frac{1}{2} \right] \]

\[ e = \left( 2\pi \right)^\frac{d}{2} \left[ \sum_{i=1}^{N} \sum_{n=1}^{N} A_i A_n C_{in} \det \left( P_{in} \right)^\frac{1}{2} + \sum_{n=1}^{N} \sum_{m=1}^{M} A_n A_m C_{nm} \det \left( P_{nm} \right)^\frac{1}{2} + \sum_{q=1}^{M} \sum_{m=1}^{M} A_q A_m C_{qm} \det \left( P_{qm} \right)^\frac{1}{2} \right]. \]

Then, by substituting back in for \( A_n \) and \( A_m \):
The goal is to drive the error function to zero. The total number of parameters are:

\[
e = \sum_{i=1}^{N+M} A_i \left(2\pi\right)^{\frac{d}{2}} \left[ \sum_{n=1}^{N} A_n C_{in} \det \left(P_{in}\right)^{-\frac{1}{2}} + \sum_{m=1}^{M} A_m C_{im} \det \left(P_{im}\right)^{-\frac{1}{2}} \right]
\]

\[
e = \left(2\pi\right)^{\frac{d}{2}} \left[ \sum_{l=1}^{N} \sum_{n=1}^{N} w_l w_n \left(2\pi\right)^{-d} \det \left(P_l\right)^{-\frac{1}{2}} \det \left(P_n\right)^{-\frac{1}{2}} \det \left(C_{ln}\right) \right]
\]

\[
- 2 \sum_{n=1}^{N} \sum_{m=1}^{M} p_n p_m \left(2\pi\right)^{-d} \det \left(P_n\right)^{-\frac{1}{2}} \det \left(P_m\right)^{-\frac{1}{2}} \det \left(C_{nm}\right) \right]
\]

\[
+ \sum_{q=1}^{M} \sum_{m=1}^{M} p_q p_m \left(2\pi\right)^{-d} \det \left(P_q\right)^{-\frac{1}{2}} \det \left(P_m\right)^{-\frac{1}{2}} \det \left(C_{qm}\right) \right]
\]

Since \(P_n\) and \(P_m\) are both square, symmetric, and the same size, we can further simplify:

\[
e = \left(2\pi\right)^{\frac{d}{2}} \left[ \sum_{l=1}^{N} \sum_{n=1}^{N} p_l p_n C_{ln} \det \left(P_l P_n^{-1} P_l^{-1}\right)^{-\frac{1}{2}} \right]
\]

\[
- 2 \sum_{n=1}^{N} \sum_{m=1}^{M} p_n p_m C_{nm} \det \left(P_n P_m^{-1} P_m^{-1}\right)^{-\frac{1}{2}}
\]

\[
+ \sum_{q=1}^{M} \sum_{m=1}^{M} p_q p_m C_{qm} \det \left(P_q P_m^{-1} P_m^{-1}\right)^{-\frac{1}{2}} \right]
\]

Let us be very specific:

\[
P_{ij}^{-1} = \left(P_i^{-1} + P_j^{-1}\right)
\]

\[
\mu_{ij}^T = \left(\mu_i^T P_i^{-1} + \mu_j^T P_j^{-1}\right) \left(P_i^{-1} + P_j^{-1}\right)^{-1}
\]

so:

\[
C_{ij} = \exp \left[ \mu_{ij}^T P_{ij}^{-1} \mu_{ij} - \mu_i^T P_i^{-1} \mu_i - \mu_j^T P_j^{-1} \mu_j \right]
\]

\[
= \exp \left[ \left(\mu_i^T P_i^{-1} + \mu_j^T P_j^{-1}\right) \left(P_i^{-1} + P_j^{-1}\right)^{-1} \left(P_i^{-1} + P_j^{-1}\right) \left(P_i^{-1} + P_j^{-1}\right)^{-1} \left(\mu_i^T P_i^{-1} + \mu_j^T P_j^{-1}\right)^T \right]
\]

\[
- \mu_i^T P_i^{-1} \mu_i - \mu_j^T P_j^{-1} \mu_j \right]
\]

\[
= \exp \left[ \left(\mu_i^T P_i^{-1} + \mu_j^T P_j^{-1}\right) \left(P_i^{-1} + P_j^{-1}\right)^{-1} \left(P_i^{-1} \mu_i + P_j^{-1} \mu_j\right) - \mu_i^T P_i^{-1} \mu_i - \mu_j^T P_j^{-1} \mu_j \right]
\]

The goal is to drive the error function to zero. The total number of parameters are:
the $M$ means, each with $k$ elements;
the $\sum_{i=1}^{d} i$ entries in the covariance matrix that we can reduce down to $d$ if we require variables that are aligned with the principal dimensions;
and the $M$ weights.

So the total number of parameters to optimize is $M (2d + 1)$.

There already exists a Python toolbox that will perform this process. The Sci-kit learning tool box can perform the fitting process [Pedregosa and Varoquaux (2011)]. This tool lends itself very nicely itself to fitting distributions that are defined by populations (particle filter types). There are some limitations.

From Figures 4.48 - 4.51 we can see that the algorithm is able to fit to a distribution with means that are somewhat grouped whether the distributions are weighted evenly or unevenly. However, if the means are widely dispersed, then fitting a reduced number of distributions is very difficult.
Figure 4.48: Compare a Gaussian mixture with 8 distributions with varying weights and grouped means to fitted Gaussian mixture with 4 distributions.
Figure 4.49: Compare a Gaussian mixture with 8 distributions with even weights and grouped means to a fitted Gaussian mixture with 4 distributions.
Figure 4.50: Comparing a Gaussian mixture with 8 distributions with varying weights and ungrouped means to a fitted Gaussian mixture with 4 distributions.
4.5 Summary

Although for the specific study that was used in chapter 3, solutions for dealing with complex uncertainty models resulting from image registration to a three dimensional model were found, and those solutions were sufficient; there is still a great deal of work to be done. Properly defining the uncertainty models arising from variability of either image registration or object position in space are highly non-Gaussian and will require complex tools to approximate them.
CHAPTER 5. CONCLUSIONS

Nondestructive evaluation is a critical component for reducing failures in the aerospace industry, as well as in many other industries. Beyond just reducing cost and loss of mission, nondestructive evaluation is also critical for human safety, the primary concern. The best decisions are the ones that result in the safest outcome, however the safest course of action may also prove to be too conservative and prohibit mission goals. The airplane will always be safe in the hanger. Nondestructive evaluation must balance acceptable risk, current knowledge and probabilistic prediction of failure in order for humans to make choices about repair and retirement for cause. Historically, decisions were made based on anything that was observable in data that provided a single snapshot of the state or the part.

The goal of the work presented here is to provide a foundation for the next step in automated nondestructive evaluation data fusion and analysis. The case for using all data, over the lifetime of the part, to make predictions about failure and performance was made by using the established estimation framework for material state tracking in Chapter 2, demonstrating the process using actual data in Chapter 3, and discussing some of the issues that arose while trying to register image data to a spatial model of a specific part in Chapter 4.

Addressing the challenges to improve nondestructive evaluation require multi-disciplinary problem solving. Solutions draw from machine learning, state estimation, machine vision, human computer interaction, data fusions, as well as all the physics of each nondestructive evaluation modality being used. Attempting to solve these issues from a single
vantage point will inevitably yield solutions that are built more around the individual application rather than the overarching challenge. The only viable conclusions is to incorporate techniques from each field to provide the best solutions.

Chapter 2 discussed the basis for spatio-temporal Kalman filtering and used a simple version of this to simulate material state tracking on composite specimens. The simulation illustrated that by using the algorithm presented in that chapter, even with very naive inputs, it is possible to track a dynamic material state and provide estimates that better reflect the true state of the part as compared with the most recent sensor measurement alone.

Chapter 3 demonstrated the algorithm presented in chapter 2 on laboratory test data. Composite panels were manufactured and then intentionally impacted to induce subsurface delaminations. Then the composite panels were loaded multiple times in four-point bending to induce incremental damage growth. After each damage event (initiation and loading) flash thermography and computed tomography data was collected. The flash thermography data was used as a sensor measurement in the spatio-temporal Kalman filter and the computed tomography data was used as a ‘truth’ value for comparison. For four out of five data sets, at every time step, the spatio-temporal Kalman filter estimate matched the computed tomography ‘truth’ better than the most recent single sensor measurement. For the fifth data set, the estimate better matched the truth at most time steps. This was accomplished using an extremely naive damage evolution model. The outcomes could only improve with better dynamic models and sensor models.

Finally, Chapter 4 presented a method that uses a probabilistic approach to identifying the location and orientation, or pose, of a specimen or part within an image. This process found the most likely transformation from the object coordinate frame to the camera coordinate frame but also ranked less probable transformations by likelihood. This chapter also discussed the solution found by Dr. Rafael Radkowski to automatically identify parts in the flash thermography data by first identifying the part using
a visible light camera that is calibrated with the thermal camera and then transforming the part coordinates into the thermal camera frame. Chapter 4 also discussed the different, non-Gaussian uncertainty distributions that result from the process of registering two dimensional images to three dimensional part models. Finally, a method for approximating non-Gaussian distributions using Gaussian mixtures was presented and discussed.

It is critical that the aerospace industry begin to treat all data, collected over the lifetime of the part, as valuable. By discarding some data instead of incorporating it into holistic models, the industry is shortchanging itself. Integrating these data, as demonstrated in this thesis, leads to better estimates about the state of a part or structure. Improved estimates lead to better decisions, improving safety and reducing cost. The Bayesian filtering process for state estimate is the mathematically correct manner for incorporating life-time data into dynamic estimates. In addition it is a convenient and elegant method. The individual tools exist for solving the individual problems; all that is needed is the will to implement new procedures in the field.


OpenCV Dev Team (2013a). Camera calibration and 3D reconstruction.

OpenCV Dev Team (2013b). Camera calibration With OpenCV.


