KINEMATIC OBJECT TRACK STITCHER FOR POST TRACKING FRAGMENTATION DETECTION AND CORRECTION

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electrical Engineering

By

Alex Wunderlin Beigh

UNIVERSITY OF DAYTON

Dayton, Ohio

May, 2015
KINEMATIC OBJECT TRACK STITCHER FOR POST TRACKING
FRAGMENTATION DETECTION AND CORRECTION

Name: Beigh, Alex Wunderlin

APPROVED BY:

______________________________          ________________________________
Eric Balster, Ph.D.                  Vijayan K. Asari, Ph.D.
Advisory Committee Chairman          Committee Member
Associate Professor, Department of   Professor, Department of Electrical
Electrical and Computer Engineering   and Computer Engineering

______________________________         ________________________________
Juan Vasquez, Ph.D.                  Eddy M. Rojas, Ph.D., M.A., P.E.
Committee Member                    Dean
Senior Research Electronics Engineer,
                                     School of Engineering
ABSTRACT

KINEMATIC OBJECT TRACK STITCHER FOR POST TRACKING
FRAGMENTATION DETECTION AND CORRECTION

Name: Beigh, Alex Wunderlin
University of Dayton

Advisor: Dr. Eric Balster

A common goal of tracking objects over multiple image frames is to ensure that the object tracks are as accurate and error free as possible. One of the biggest impediments to accurate object tracking is an event called track fragmentation. Track fragmentation occurs when an object’s trajectory or track is reported by the tracker as two or more separate track instances. Fragmentation occurs because of features present in the imagery and the method at which the imagery is processed. Some causes of track fragmentation include: occlusions, object trajectory or velocity changes, proximity of objects to one another, parallax, image stitching, lighting changes, and image artifacts. Track fragmentation is tackled in multiple ways. The GATER tracker uses projected points to make predictions where tracks will be in future frames to compensate for events that cause fragmentation. Another solution is to use a process known as track stitching. A post-tracking kinematic track stitcher is proposed that is able to detect and correct track fragmentation, improving fragmentation from GATER’s projected points. The proposed track stitcher requires only a track’s centroid and frame, information any tracker can provide. The track stitcher will improve the accuracy
of the tracks by reducing track fragmentation. The track stitcher consists of a series of filters, including a temporal filter and multiple kinematic filters, designed to determine the likelihood and feasibility of a track being fragmented. The filters are designed to be easily enabled, disabled, and integrated depending on the needs of a given data set. These filters include sets of parameters that can be manually altered by the user for different tracking scenarios. A selection algorithm, in this case Munkres, is used to evaluate the track scores and make stitches accordingly. The track stitcher is run on multiple track sets acquired from a Wide Area Motion Imagery image set. For the first experiment, one of the image sets is selected for tuning, and then the track stitcher is run with varying parameters for the kinematic filter and scoring filter. The best performing parameters for the first image set are then used to test the track stitcher on the remaining image sets in order to prove that the same parameters can be universally applied to similar imagery. In every test case the track stitcher reduces the amount of track fragmentation and error (spuriousness) present in the tracks. In a majority of the tests, the purity of the tracks also increases. When compared to the point projected tracks GATER produces, the proposed track stitcher improves spuriousness by 26.67% and fragmentation by 17.67% on average.
# TABLE OF CONTENTS

ABSTRACT ................................................................. iii
LIST OF FIGURES ........................................................... vii
LIST OF TABLES ........................................................... ix

I. INTRODUCTION ......................................................... 1

II. BACKGROUND .......................................................... 4
   2.1 Image Registration .................................................. 4
   2.2 Object Tracking: Trackers .......................................... 6
   2.3 GATER Tracker ...................................................... 10
   2.4 Causes of Track Fragmentation .................................... 12

III. CREATING A KINEMATIC TRACK STITCHER ....................... 22
   3.1 Description of Tracks ............................................... 25
   3.2 Time Filter ......................................................... 29
   3.3 Kinematic Filters: Distance, Velocity, Heading ............... 32
   3.4 Scoring the Track Pairs .......................................... 37
   3.5 Selection Algorithm: Munkres ................................... 40

IV. RESULTS AND DISCUSSIONS ......................................... 45
   4.1 Establishing Truth ................................................ 46
   4.2 Metrics: Purity, Fragmentation, and Spuriousness ............ 47
   4.3 Projected Points .................................................. 49
   4.4 Testing: Using Metrics for Tuning the Track Stitcher ....... 51
      4.4.1 Controlled Image Set Parameter Optimization ............ 54
      4.4.2 Optimized Track Stitcher on Image Set 2 ............... 63
      4.4.3 Optimized Track Stitcher on Image Set 3 ............... 64
LIST OF FIGURES

2.1 Two unregistered images of the same scene. The left image is the reference image and the right image is the moving image ............................. 5

2.2 Example of an image created through image stitching from the CLIF data set [6] .......................................................... 6

2.3 Example of background subtraction in which the background is not aligned in the two frames [18] .................................................. 8

2.4 Example of GATER detections .......................................................... 13

2.5 Example of GATER tracks .......................................................... 14

2.6 Example of white car occluded by trees over four consecutive frames ................... 16

2.7 Example of white car slowing down and stopping over eight consecutive frames 16

2.8 Example of image stitching causing lighting change midframe .................... 18

2.9 Comparison of the same image frame with 0 compression versus 80 compression. Image artifacts induced by the compression ..................... 19

2.10 Example of cars in close proximity and the subsequent tracking failure ...... 19

2.11 Example of parallax using registered imagery ................................. 20

3.1 Frame A with white car as a POI .................................................. 23

3.2 Frame B with white car as a POI .................................................. 24

3.3 Architecture of the track stitcher .................................................. 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4 Architecture of the time filter</td>
<td>31</td>
</tr>
<tr>
<td>3.5 Architecture of the kinematic filters</td>
<td>33</td>
</tr>
<tr>
<td>3.6 Architecture of the scoring method</td>
<td>38</td>
</tr>
<tr>
<td>4.1 First frame of image set 1</td>
<td>55</td>
</tr>
<tr>
<td>4.2 First frame of car turning, image set 1</td>
<td>56</td>
</tr>
<tr>
<td>4.3 Second frame of car turning, image set 1</td>
<td>57</td>
</tr>
<tr>
<td>4.4 Purity graphs for image set 1</td>
<td>59</td>
</tr>
<tr>
<td>4.5 Fragmentation graphs for image set 1</td>
<td>60</td>
</tr>
<tr>
<td>4.6 Spuriousness graphs for image set 1</td>
<td>62</td>
</tr>
<tr>
<td>4.7 First frame of image set 2</td>
<td>65</td>
</tr>
<tr>
<td>4.8 First frame of image set 3</td>
<td>67</td>
</tr>
</tbody>
</table>
LIST OF TABLES

3.1 Example of selection matrix before Munkres assignment . . . . . . . . . . 41
3.2 Initial matrix for Munkres example . . . . . . . . . . . . . . . . . . . . . . 41
3.3 Step 1: Munkres example after row minima are removed. . . . . . . . . 42
3.4 Step 2: Munkres example matrix after column minima are removed. . . 42
3.5 Step 3: Munkres example with bold elements mark the optimal solution . 43
3.6 Optimal solution from step 3 being applied to the original matrix. Bold
elements mark the optimal solution . . . . . . . . . . . . . . . . . . . . . . . 43
3.7 Matrix without an optimal solution after step 3 of the Munkres algorithm.
Bold elements represent covered rows and columns . . . . . . . . . . . . . 44
3.8 Step 4: The matrix from Table 3.7 after undergoing step 4 of the Munkres
algorithm. Bold elements represent the optimal solution . . . . . . . . . 44
4.1 Table comparing average track length and spuriousness of image sets with
and without projected points . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
4.2 Table comparing average purity of image sets with and without projected
points . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.3 Table containing the purity scores for each combination of filter allowance
and kinematic maximum. Scores in blue indicate the best performing tests . 58
4.4 Table containing the fragmentation scores for each combination of filter allowance and kinematic maximum. Scores in blue indicate the best performing tests .......................................................... 60

4.5 Table containing the spuriousness scores for each combination of filter allowance and Kinematic maximum. Scores in red indicate the worst performing tests .......................................................... 61

4.6 Table comparing the purity, fragmentation, and spuriousness of image set 1’s tracks with the projected points, without the projected points, and after the track stitcher is run .......................................................... 63

4.7 Table comparing the purity, fragmentation, and spuriousness of image set 2’s tracks with the projected points, without the projected points, and after the track stitcher .......................................................... 64

4.8 Table comparing the purity, fragmentation, and spuriousness of image set 3’s tracks with the projected points, without the projected points, and after the track stitcher .......................................................... 66
CHAPTER I

INTRODUCTION

The use of computer software to track objects over multiple frames of imagery is becoming increasingly common. As a result, there has been a great deal of research dedicated to obtaining more accurate and comprehensive lists of object tracks. Object tracking performance can be improved across many steps of the tracking process. Sensor improvements provide a more informative image frames, while enhanced registration techniques allow more challenging image sets to be properly registered to one another. As tracking algorithms themselves improve, the accuracy of the track creation and assignment process has also increased. Despite the numerous improvements of trackers, there are still often problems that arise in the formation of tracks. Occlusions, poor frame registration, lighting changes, and proximity of tracks to one another can cause what should be a single track to instead be reported by trackers as multiple unique tracks. This results in fractured tracks, also known as track fragmentation. One method to fix fragmentation is to predict where a track will be in future frames using projected points. Using the projected points, a tracker can predict where future track assignments should be made, lowering the impact of track fragmentation. These projected values are estimations and not real, but can be used by the tracker to keep tracks from becoming fragmented when the projected point is sufficiently
accurate. However, projected points do not remove all fragmentation. Another method, *track stitching*, can be used to further reduce the amount of fragmentation in tracks.

Generally, when people think about ways to improve the quality of the track output from tracking systems, the focus is placed on early processes in the tracking chain. Improvements to the sensor array, registration method, and tracking suite or algorithm yield better images and as a result, better tracks, but there are limitations. Some sensors are better suited for a certain type of data acquisition, while registration methods vary in effectiveness depending on the features of the registered imagery. The tracker itself is limited by the quality of the image frames received, the detection algorithm used, how tracks are updated and scored, and how the tracker assigns track likelihood. A track stitcher is an additional tool that can be added without making any changes to any of the other steps of the tracking process. This thesis proposes a track stitcher that provides improved track fragmentation reduction over the point projection used by the GATER tracker. The proposed track stitcher reduces the amount of track fragmentation, while also increasing track purity and decreasing the amount of spuriousness in the tracks.

The thesis is broken down in the following construct. Chapter 2 provides an overview of registration, tracking, and common causes of track fragmentation. Chapter 3 presents a mathematical model for understanding detections and their subsequent formation into tracks, and presents a kinematic track stitching algorithm that can be used on any set of tracks that have kinematic and time data available. Chapter 4 discusses the process of preparing the tracks for stitching, the metrics that are used to analyze the performance of the track stitcher, a method for optimizing parameters for a given set of data, and the test methodology and results from the track stitcher across multiple image sets. Chapter 5
summarizes the conclusions and recommends future work. The sources used in this thesis are included in the bibliography.
CHAPTER II

BACKGROUND

This chapter provides background on the steps leading up to the formation of tracks. Section 2.1 discusses image registration and provides examples of various image registration techniques. Section 2.2 provides information regarding trackers and the various methods trackers employ to form tracks. Section 2.3 discusses the GATER tracker, the tracker used in the analysis of the proposed track stitcher. Section 2.4 is an overview of the different causes of track fragmentation.

2.1 Image Registration

Image registration is the first step in the tracking process. For a given set of image frames, image registration seeks to find the transformation that will align two or more images of the same scene with one another. Sensor movement, differing viewpoint angles, and time changes are just a few of the scenarios that can warrant image registration [1]. The best registration usually requires that an image be shifted, rotated, and/or scaled in order to create properly aligned image frames. An example of images before registration is shown in Figure 2.1. The first image, shown on the left, is known as the reference image and the second image, shown on the right, is the moving image. The moving image is shifted in both the x and y direction from the reference image. There are many different

Image stitching is another type of image registration. Unlike traditional image registration, which seeks to completely align images, image stitching partially aligns, blends, and overlaps different images of a scene to create a single larger image of the scene [4]. A tutorial of different image stitching techniques is published by Szeliski [5]. Figure 2.2 shows an example of an image created through image stitching. In this example, a series of images were taken simultaneously by multiple sensors looking at different parts of a landscape. When combined, the resulting image shows the entirety of the landscape. The advantage of using multiple sensors and image stitching is that the image in Figure 2.2 has a higher resolution than if a single sensor had been used to film the same area. It is important to
not confuse image stitching with track stitching, as they are performing completely separate functions from one another.

2.2 Object Tracking: Trackers

The primary purpose of object tracking is to obtain information about desired objects across multiple frames. There are many applications of trackers, including motion recognition, surveillance, traffic monitoring, medical imaging, human-computer interfaces, and
A comprehensive survey on trackers is provided by Yilmaz et al. [7].

The first task of a tracker is to determine how an object will be represented. An object can be represented by something as simple as the centroid location of the object [12] or by something more complex, such as the object’s probability density. For example, Comaniciu et al. propose a histogram-based representation of objects, with the reference model using the color pdf of the object [11]. Additionally, the covariance descriptor, proposed by Tuzel et al. [13] and further refined by Wu et al. to use Riemannian manifolds [9], can also be used to describe an object.

Next, the detection method for objects is determined. One of the most common techniques for object detection is background subtraction [7]. Piccardi reviews a variety of the current background subtraction techniques in a 2004 paper [14], but a basic overview is presented below. Background subtraction can be used to detect moving objects when there is a static background scene. Background subtraction methods subtract the current image from the reference image to obtain the foreground of the current image. When subtracted, the foreground appears as a silhouette, indicating moving objects worth tracking [15]. Pixel thresholds are also commonly used to filter noise between the two images and to allow for more leeway in color differences between frames. The downsides of traditional background subtraction is the heavy reliance on image registration and the susceptibility to changes in the illumination between images [15]. If the reference frame and current frame are not aligned, the resulting foreground may contain portions of the background. The image shown in Figure 2.3 is an example of a house that is brought to the foreground by background subtraction. This occurs because the background is not aligned in the two frames used to
conduct the background subtraction. Elgammal et al. in 2006 proposes a method of non-parametric background subtraction that is capable of handling small motion discrepancies between frames, such as swaying trees [16]. Illumination changes can result in wildly different color characteristics of the background between the reference and current image. The result is that large portions of the image are falsely flagged as the foreground. Javed et al. present a background subtraction approach that is able to handle brief illumination changes and shadows through the use of color gradients [17].
Once objects are detected in frames, the next step is to find the trajectory of the objects across each frame. In a system that has to track multiple targets, such as GATER, multiple hypothesis tracking (MHT) is a preferred method [19]. The first step of MHT requires that tracks be initialized, commonly performed by a Kalman filter.

The Kalman filter is prevalent in tracking algorithms. First proposed by Kalman in 1960 [20], the Kalman filter is a recursive optimal estimator which predicts variable states based off observations. One reason the Kalman filter is so common in tracking application is that it still functions even for inaccurate observations. The Kalman filter has two steps: prediction and update. In the prediction step, the Kalman filter estimates what a variable’s next state will be. Once the next state is known, the estimates are updated based on the certainty of the update. Weights are then given based on the certainty of the estimated state. In this way, the Kalman filter removes noise by assigning low estimations and confidence weights to noise. By using the Kalman filter, a tracker is able to estimate the next location of an object and, when a detection near the estimated region occurs, assign a confidence weight that it is the same object. Based on this information, a track can be initialized.

After track initialization, many MHTs continue to utilize the Kalman filter. Using the prediction covariances provided by the Kalman filter, the tracker will look to see what detections are within the track’s gate. In this context a gate is the region the tracker determines is valid for a track continuation to exist [21]. One of the most common types of gates is Euclidean distance. In MHT applications, there is often more than one detection that falls within a gate. In these scenarios, the MHT approach defers associating the detection to a track until after looking at detections in future frames [19]. Using information from the Kalman filter, scores are applied to each detection that can potentially be a part of a track. The MHT looks at all possible solutions and carries them forward through the data.
set, which can become computationally expensive. In a 2004 paper [19], Blackman outlines different scoring techniques that MHTs can use, such as the Likelihood Ratio and track score. These scores can be used to then compute the best track result, usually through the application of a selection algorithm [21]. Once the MHT processes all possible solutions for each detection, the tracker’s output is the list of tracks that have the optimal score computed by the MHT, based on the Kalman filter’s (or another similar filter) estimations and weights.

2.3 GATER Tracker

The Government Algorithms for Tracking Exploitation Research (GATER) is a tracking suite created by AFRL [21, 22] and improving the output of the GATER tracker though track stitching is the focus objective of this research. GATER contains tools for processing, target detection, tracking, visualization, metrics, and more. GATER is extremely robust and is not limited to a single type of imagery or sensor type.

GATER takes a set of images and generates detections for all the image frames. The types of objects detected can be easily changed, as GATER has many user controlled parameters that enable the detections to be individually tailored to an image set. Minimum and maximum pixel size, eccentricity, and standard deviation are just a few of the parameters that can be manipulated to fine tune the detections for a given detector. One of the most important aspects of GATER is its flexible design. If a user desires to use a different detection algorithm from three frame difference imagery, the desired algorithm can be easily added to GATER. This is true for each step of the process; virtually every algorithm can be substituted for a different algorithm to allow GATER to process different tracking
scenarios. This enables GATER to be an extremely powerful testing bed for algorithms that span all parts of the image processing problem.

GATER is a multiple hypothesis kinematic tracker which, by design, predicts multiple solutions for the continuation of each individual track. This prediction allows GATER to estimate where future detections may occur. Through the estimations, GATER has a better idea of how the detections relate to one another. When GATER is run, the MHT solutions help link the detections together to form the strongest tracks. The resulting tracks are a combination of the detections and the MHT, as each of the resulting track values are weighted based on the confidence score of the MHT solution.

The GATER tracker’s MHT algorithm allows it to make predictions of where tracks might be located. GATER uses these predictions to hone in on an area when making the track assignments for a POI. GATER also has a setting that allows a track to coast a certain number of times before the track is terminated. When a detection is made in the region that the MHT predicts a point should be at while the track is coasting, the track resumes from the detection and the coast counter is reset. Projected points are defined as a reported point in a track’s history that is not validated by a real detection. In the event that the track coasts past the set limit without a real detection, the track ends and the MHT algorithm no longer continues the overhead of tracking predictions. From that point forward, any detections near the end of the original track are given a new track identification number. Within a tracker, keeping a track alive, even if real detections are not made, gives it the capability to respond robustness to a variety of errors such as those caused by occlusions, image artifacts, or poorly registered imagers.

The built-in metrics program allows users to see how well GATER runs based on the specified algorithms and parameters, allowing for powerful tuning applications. When it
comes to analyzing and manipulating data from an image set, it's extremely useful to be able to see the formation and history of the tracks. GATER allows users to visualize what the detections of the tracks look like based on the imagery, as shown in Figure 2.4 and 2.5. In Figure 2.4, each colored circle is a potential detection. As can be seen, not all detections are centered on a car or other object of interest, such as the detections on the three roofs in the top left quadrant of the image. Many of the false positive detections are results of registration errors or artifacts within the images. Figure 2.5 shows how all the detection that are kept by GATER are linked together to form tracks. In Figure 2.5, it can be seen that not all of the tracks are on the road, indicating false positives, and that some of the tracks are not as long as they should be. These problems occur due to events that cause track fragmentation.

2.4 Causes of Track Fragmentation

Track fragmentation results from a myriad of different sources. A fragmentation event refers to an occurrence in the image set that causes track fragmentation to occur. Two different types of fragmentation events are analyzed. The first type of fragmentation event is caused by the features of the scene from which the imagery is taken. These events are called scene events. The second type of event is caused by the hardware, software, algorithms, and methods implemented to collect and analyze the image sets. These are called artificial events. Some fragmentation events, such as parallax and object proximity, cause fragmentation due to both a scene and artificial event.

There are many types of image features that can cause scene events. Occlusions are one of the leading causes of scene events [7, 23]. An occlusion refers to any time an object that is being tracked somehow becomes hidden by another object. There are two types
Figure 2.4: Example of GATER detections
Figure 2.5: Example of GATER tracks
of occlusions, moving and stationary. Moving occlusions refer to occlusions occurring in multiple scenes that appear to move and block the scene, such as people walking through a photo and obscuring the scene. These moving occlusions can be removed if sufficient frames of the desired scene are present. Herley presents an algorithm to remove moving occlusions that only needs one image frame of non-occluded imagery for each location [24]. However, in tracking, moving occlusions are usually the objects that the tracker is attempting to track. The other, and most problematic, type of occlusion is the stationary occlusion. Stationary occlusions are part of the image scene and are interposed between the camera and the object being tracked, causing the tracker to lose detections or, worse, discontinue the track completely [23, 25]. Overpasses and trees are two common types of stationary occlusions. Figure 2.6 shows four consecutive frames of the same stretch of road that has a tree canopy. A white car can be seen in the first frame, which proceeds to become occluded by the brush in the second frame. In the third frame, the white car blends in with the trees. It isn’t until the fourth and final frame that the white car starts to clearly emerge from underneath the brush. While the eye can track the car without too much difficulty, occlusions like this can easily lead to the tracker dropping or mischaracterizing the object, causing track fragmentation. Stationary occlusions cannot be removed in the same fashion as moving occlusions because the stationary occlusions are present in each frame.

Slowing or stopped objects are another common scene fragmentation event. When an object slows down or stops moving completely, the object will often be re-organized into being part of the background of the scene [26]. Figure 2.7 shows a white car approaching stopped traffic. There is also a black car driving alongside the white car at the start, providing a comparison of the speed difference over the frames. As the white car approaches traffic, it slows down, until eventually the car stops in the last two frames.
stopped for multiple frames, the white car will be considered part of the image background. When it starts moving again, the white car will be classified as a new track, as most trackers cease maintaining the overhead required to re-obtain the white car’s original track. This is a prime example of a fragmentation event that track stitching seeks to rectify.

Often times fragmentation is not caused by the images themselves, but rather by the method in which the images are acquired and registered. Image stitching, blurring, and the appearance of image artifacts, or noise, are almost always artifact events[7]. Figure 2.8 shows an image frame that is acquired by stitching together input from separate sensors. The line running diagonally across the frame is an indication of where the image from one sensor stops and the other begins, a sign of poor image stitching[4]. Trackers are prone to an increase in tracking errors when there are lightning changes or shadows present in imagery.
so, unless a tracker is equipped to handle such illumination changes, tracks will potentially be lost or terminated [23, 27]. Poor quality sensors and data compression of the imagery are additional examples of the ways that artifacts can occur in imagery[11]. Figure 2.9 compares the same image frame with 0 compression versus 80 compression. Artifacts can cause association errors, mimicking scene events such as close object proximity. Artifacts that are prevalent can also occlude the objects that are desired to be tracked, causing the tracker to lose them. An even more problematic scenario for artifacts is that they become false positive detections. The false detections can get built into tracks of real objects, creating erroneous predictions and leading to a higher rate of early track termination [19].

For multiple object tracking, the proximity of objects to one another can cause a fragmentation event that can be considered a scene or artificial fragmentation event. Tracking will often fail when the tracker attempts to track objects that are in close proximity to one another or if one of the tracked objects occludes another [28]. Figure 2.10 shows an example of two objects whose close proximity results in track fragmentation. The top row is the raw image and the bottom row has the tracks overlayed on the image. The purple track is the black car and the green track is the white car. Circles indicate an object is detected and an x indicates the object location is being completely estimated. In this example, the MHT used by GATER mixes the two cars up when they pass by. The misidentification of the cars in a single frame throws off the prediction so that in the subsequent frame, the predicted detection region for each car is skewed enough that neither car is picked up. This event is categorized as both a scene event and an artificial event because the two objects in close proximity is a feature of the image set, but the problem is further aggravated by a shortcoming in the tracker's association algorithm.
Figure 2.8: Example of image stitching causing lighting change midframe
Figure 2.9: Comparison of the same image frame with 0 compression versus 80 compression. Image artifacts induced by the compression.

Figure 2.10: Example of cars in close proximity and the subsequent tracking failure.
Motion parallax is another problem that results from both a scene and artificial event [7]. Parallax is the phenomenon that occurs when images contain objects that extend vertically from the ground plane. These vertical objects move differently than the background scene that comprises the ground plane. Parallax becomes particularly pronounced in aerial imagery where there is a significant distance between the sensor and the ground plane [29]. Figure 2.11 shows a set of registered images affected by parallax. The effects of the parallax can be noticed in the way that the trees and building appear to sway between the two frames. Image registration is primarily affected by parallax, as the post registration images tend to have more error than parallax free imagery. The registration errors impact the trackers as swaying objects that are sometimes classified as tracks or occlude real objects, leading to an increase in track fragmentation.
The best way to tackle track fragmentation events is to solve them in the sensor, registration, or tracking steps, but often times it is not cost effective, timely, or feasible to do so. This is where a track stitcher helps to reduce fragmentation in an image set.
CHAPTER III

CREATING A KINEMATIC TRACK STITCHER

In order to create the kinematic track stitcher, a more in-depth analysis of objects, detections, and the subsequent creation of tracks is required. Objects are defined as items of interest that exist in an image frame or across a series of image frames. An object tends to have many distinct features and characteristics that set it apart from other objects. Humans can usually easily look at a series of images and track objects between them but doing so with a computer system is far more complicated. For most computer systems, objects are detected on a frame-by-frame basis, usually through a combination of image registration and change detection. After a detection, a point of interest (POI) is defined in a frame associated with the detected object. An object that spans multiple frames will leave an equal number of POIs.

These POIs provide the foundation for establishing a relationship between image frames. This is where trackers are utilized. Trackers compare the points of interest across multiple frames of an image and determine if there is a relationship of a POI in frame A to a POI in frame B. An example of this can be seen in Figure 3.1 and Figure 3.2. Figure 3.1 shows a white car during an early frame of an image set and Figure 3.2 shows the same white car on a later frame of the image set. The tracker should be able to detect the motion of the white car and label the location of the white car in each image as POIs to be matched together.
Grouping POIs in this fashion creates tracks. Tracks are the system’s way of describing objects over multiple frames, and the ideal tracker outputs one track for one object.

As mentioned previously, trackers will not always create a single track for a single object. There are many scenarios that can cause this phenomenon, including occlusions, frame registration errors, and significant track trajectory changes. Having two or more tracks associated with single object is called track fragmentation. Track fragmentation
Figure 3.2: Frame B with white car as a POI
causes inaccuracies when analyzing object tracks. These inaccuracies manifest in tracks that end prematurely and have POIs detected after the drop labeled as new objects.

Fragmented tracks are classified as dropped tracks and candidate tracks. Dropped tracks are one or more tracks whose POIs are not found in the last frame associated with the object, whereas candidate tracks are one or more tracks whose POIs are not found in the first frame associated with the object. The goal of a track stitcher is to analyze and merge the appropriate dropped and candidate tracks together in order to form stitched tracks. This will reduce the amount of track fragmentation in the track results.

3.1 Description of Tracks

\( P_k \) is a vector whose components are the centroid of a POI and the corresponding frame the POI is found in, as determined by a track stitcher. \( P_k \) is defined as

\[
P_k = [x_k, y_k, n_k],
\]

(3.1)

where \( k \) is a single POI, \( x_k \) is the position of the centroid described by \( P_k \) in the x-axis, \( y_k \) is the position of the centroid \( P_k \) in the y-axis, and \( n_k \) is the frame in which the POI was detected.

A track, \( X_i \), is a combination of vectors determined by a tracker to belong to the same object across multiple frames of time and is defined as

\[
X_i = \begin{pmatrix}
P_{j_1} \\
\vdots \\
P_{j_m}
\end{pmatrix} = \begin{pmatrix}
x_{j_1} & y_{j_1} & n_{j_1} \\
\vdots & \vdots & \vdots \\
x_{j_m} & y_{j_m} & n_{j_m}
\end{pmatrix},
\]

(3.2)

where \( i \) is the object number for a given track and \( j \) is a vector or set of vectors associated with a given track number (\( j \) is subset of \( k \)). For example, object A is represented by track \( X_A \), object B by track \( X_B \).
Sometimes, not all points of interest of a given object are grouped into the same track. The result is track fragmentation, a situation where two or more tracks are associated with the same object. In terms of the tracker, this means that the tracker has identified what is in reality a single object as two or more individual objects and, as a result, has assigned a separate track for each object. For example, object A and object B are really the same object C, but because of occlusions or some other issue, they were each given a separate object number. The tracks $X_A$ and $X_B$ are really parts of the same object C and should be grouped together to form track $X_C$.

Because of track fragmentation, there is no guarantee that there is a one-to-one ratio of objects to tracks. When it is known which tracks are associated with the same object, the fragmentation can be reversed. We define $\tilde{X}_i$ as a dropped track, or a track which has ended prematurely. Also, we define $\hat{X}_i$ as a candidate track, or a track which is the continuation of a dropped track.

\begin{align}
\tilde{X}_i, \exists N_i > n_{jm} \\
\hat{X}_i, \exists N_i < n_{j1},
\end{align}

where $n_{jm}$ is the last frame a track $X_i$ is associated with. $N_i$ is a set of all the frames the object associated with $X_i$ appears. $N_i$ can only be defined with knowledge of the ground truth. In practice, for a pair of tracks $X_A$ and $X_B$ that both refer to object C, the equations result in the track that ends first being assigned the dropped track and the track that ends last being assigned the candidate track. For example, if track $X_A$ ended in frame 5 and track $X_B$ started in frame 7, $X_A$ is defined as $\tilde{X}_A$ and $X_B$ is defined as $\hat{X}_B$. The stitched track, $\bar{X}_i$, is the track which results from merging $\tilde{X}_i$ and $\hat{X}_i$ together.
with $\tilde{X}_A$ and $\hat{X}_B$, the stitch track will look as follows:

$$\tilde{X}_C = \begin{pmatrix} \tilde{X}_A \\ \hat{X}_B \end{pmatrix}.$$  \hfill (3.5)

The primary objective of the track stitcher is to form as many $\tilde{X}_i$ as possible. The definitions of $\tilde{X}_i$ and $\hat{X}_i$ require $N_i$ in order to be applied, which is only known when all the fragmented tracks are known and grouped together with the correct objects. Very rarely is this the case. The track stitcher looks at all the tracks that exist and determines whether or not they belong to a fragmented track. Then, the track stitcher combines the fragmented tracks together to form the strongest stitches.

The track stitcher is organized so that the stitcher analyzes every potential track pair in order to determine the likelihood of them belonging to the same object. Figure 3.3 shows the architecture of the kinematic track stitcher. In the first phase the time filter of the the track stitcher looks at the temporal information for each track pair. The time filter analyzes a pair of tracks solely on the tracks’ frame component to ensure that tracks are considered for stitching only when there is no common frame, that is they do not exist simultaneously at any given time. A pair that passes the time filter is advanced to the kinematic filter. The kinematic filter looks at the kinematic information regarding the track pair, with the goal of disallowing track pair combinations that are kinematically infeasible. Extremely large heading changes or position changes between the drop and candidate tracks within the span of a few frames are considered kinematically infeasible. Finally, if a pair passes both the time filter and kinematic filter, the track pair is scored. Once every pair has been compared, all the pairs that received a score are sent through a selection algorithm, known as the selection filter. In this stage, the track stitcher analyzes all the pairs and tries to combine pairs in such a fashion as to create the best scoring stitches.
Figure 3.3: Architecture of the track stitcher
### 3.2 Time Filter

The time filter is the first filter of the track stitcher. Figure 3.4 shows the architecture of the time filter and shows the decision making process associated with each pair passed from the unstitched track pool. The input to the time filter is a pair of tracks and the output of the filter is how the tracks are related to one another temporally, $\hat{X}$ and $\hat{X}$. When a pair of tracks is put into the time filter we define the pair of tracks as $X_A$ and $X_B$, where $X_A$ is the first track of a given pair and $X_B$ is the second track of a given pair. For each pair of tracks, the time filter specifies that

$$N_A = \forall n \in X_A$$

$$N_B = \forall n \in X_B$$

where $N_A$ represents all frames that $X_A$ exists in and $N_B$ represents all frames in which $X_B$ exists. The time filter now determines how the frames in $N_A$ and $N_B$ relate to one another. $N_C$ is the representation of all frames associated with $X_A$ and $X_B$ if the track pair could belong to the same object $C$. $N_C$ is defined as

$$N_C = \begin{cases} 
  \begin{pmatrix} N_A \\ N_B \end{pmatrix} & \text{if } N_A < N_B \\
  \begin{pmatrix} N_B \\ N_A \end{pmatrix} & \text{if } N_A > N_B \\
  \text{does not exist} & \text{otherwise}
\end{cases}$$

where $<$ returns true if each element of $N_A$ is less than each element of $N_B$ and $>$ returns true if each element of $N_A$ is greater than each element of $N_B$.

There are three potential outcomes for $N_C$. The first two cases order $N_C$ by frame from lowest to highest frame number, as long as $N_A$ and $N_B$ have no frames in common. The third case, in which $N_C$ does not exist, occurs when $N_A$ and $N_B$ share at least one similar
frame. The third case means that the two tracks in the pair exist simultaneously and cannot
be associated with the same object, so \( N_C \) is left undefined. In the event there no defined
\( N_C \) for a pair of tracks, the time filter rejects the current pair and proceeds to the next pair
. If \( N_C \) does exist, the time filter uses modified versions of Equations 3.3 and 3.4 to assign
\( X_A \) and \( X_B \) to the appropriate drop or candidate value.

\[
\tilde{X} = X_A, \exists N_C > n_{A_{jm}} \tag{3.9}
\]

\[
\hat{X} = X_A, \exists N_C < n_{A_{j1}} \tag{3.10}
\]

\[
\tilde{X} = X_B, \exists N_C > n_{B_{jm}} \tag{3.11}
\]

\[
\hat{X} = X_B, \exists N_C < n_{B_{j1}} \tag{3.12}
\]

where \( n_{A_{jm}} \) and \( n_{B_{jm}} \) are the last frames \( X_A \) and \( X_B \) are associated with and \( n_{A_{j1}} \) and \( n_{B_{j1}} \)
are the first frames \( X_A \) and \( X_B \) are associated with. The time filter outputs the original
inputs \( X_A \) and \( X_B \) in the form of \( \tilde{X} \) and \( \hat{X} \) to the kinematic filter.

Despite the tracks being in a form that could have Equation 3.5 applied to them, the
track pair has not yet been validated as belonging to the same fragmented object. It
is possible at this point that the tracks are actually separate distinct objects that just
happen to never share a common frame. The time filter has, however, made the important
distinction of which track existed before the other. This means that kinematic analysis can
be done with more accuracy on the pair.

The time filter can also have additional logic to reduce error, in the form of minimum
and maximum time differences. Minimum frame values require that there is a minimum
time difference between \( n_{j_m} \) and \( n_{j_1} \). For example, if a minimum value of three frames
is specified, \( n_{j_1} - n_{j_m} \geq 3 \) in order for the output to be passed to the kinematic filter.
The maximum frame value requires that the difference between \( n_{j_m} \) and \( n_{j_1} \) not exceed
Figure 3.4: Architecture of the time filter
the specified maximum. For example, if a maximum value of eleven frames is specified, 
\( n_{j_1} - n_{j_m} \leq 11 \) in order for the output to be passed to the kinematic filter.

### 3.3 Kinematic Filters: Distance, Velocity, Heading

Just because the track pair has passed the time filter, that does not mean it should be stitched. The goals of the kinematic filters are to determine if the candidate track, \( \hat{X} \), could feasibly be a continuation of the dropped track, \( \tilde{X} \). The input to the kinematic filter is the track pair \( \tilde{X} \) and \( \hat{X} \); output of the time filter when \( N_C \) is defined. The kinematic filter uses the x and y centroid locations of \( \tilde{X} \), \( \tilde{x}_j \) and \( \tilde{y}_j \), and the x and y centroid locations for \( \hat{X} \), \( \hat{x}_j \) and \( \hat{y}_j \).

The kinematic filter is divided into three parts: the distance filter, the velocity filter, and the heading filter. In order for a track pair to move past the kinematic filter, the pair must pass three individual filters. Figure 3.5 shows the architecture of the kinematic filters and the order that the track pairs pass through each of the filters. The distance filter is designed to ensure that it is possible that the object could travel from the ending location of \( \tilde{X} \) to the beginning location of \( \hat{X} \) given the frame difference. For example if there is a 1000 meter difference between the end of \( \tilde{X} \) and the start of \( \hat{X} \), and \( \tilde{X} \) was only moving an average of 10 meters per frame, the distance filter would reject the validity of the pair if there was only a five frame difference between \( \tilde{X} \) and \( \hat{X} \). If the tracks only have a 50 meter difference between them and \( \tilde{X} \) moved 10 meters per frame on average, the distance filter would pass the pair to the next kinematic filter, the velocity filter.

The velocity filter checks to see if the the object associated with \( \tilde{X} \) could have reasonably accelerated or deccelerated in the frame difference between \( \tilde{X} \) and \( \hat{X} \) for the drop track’s velocity to match the candidate track’s velocity. If a pair of tracks has a single frame
Figure 3.5: Architecture of the kinematic filters
difference and \( \hat{X} \) has a velocity of 50 meters per second and \( \hat{X} \) has a velocity of 100 meters per second, the velocity filter will reject the pair unless the acceleration of \( \check{X} \) was around 50 meters per frame\(^2\). When the acceleration supports the velocity difference, or similarity as the case may be, between \( \check{X} \) and \( \hat{X} \) the pair is passed to the third and last of the kinematic filters; the heading filter.

The heading filter is the final kinematic filter, and checks to see if the change in heading from the end of \( \check{X} \) is similar to the heading of \( \hat{X} \) at the start. This is to ensure that the track pair does not have a significant sudden change in heading, such as 180 degree direction change in a single frame.

All three kinematic filters rely heavily on the velocity. The velocity of \( \check{X} \) is defined as

\[
\check{V}_x = \check{x}_m - \check{x}_{m-1}
\]  

(3.13)

\[
\check{V}_y = \check{y}_m - \check{y}_{m-1}
\]  

(3.14)

where \( \check{V}_x \) is the drop track’s velocity in the x plane, \( \check{x}_m \) is the x centroid location of the last frame of the drop track, \( \check{x}_{m-1} \) is the x centroid location of the second to last frame of the drop track, \( \check{V}_y \) is the drop track’s velocity in the y plane, \( \check{y}_m \) is the y centroid location of the last frame of the drop track, \( \check{y}_{m-1} \) is the y centroid location of the second to last frame of the drop track. The velocity of \( \hat{X} \) is defined as

\[
\hat{V}_x = \hat{x}_2 - \hat{x}_1
\]  

(3.15)

\[
\hat{V}_y = \hat{y}_2 - \hat{y}_1
\]  

(3.16)

where \( \hat{V}_x \) is the candidate track’s velocity in the x plane, \( \hat{x}_2 \) is the x centroid location of the second frame of the candidate track, \( \hat{x}_1 \) is the centroid location of the first frame of the candidate track, \( \hat{V}_y \) is the candidate track’s velocity in the y plane, \( \hat{y}_2 \) is the y centroid location of the second frame of the candidate track, and \( \hat{y}_1 \) is the centroid location of the first frame of the candidate track.
location of the second frame of the candidate track, and \( \hat{y}_1 \) is the y centroid location of the first frame of the candidate track.

The difference in the position of the end of the drop track and the start of the candidate track is defined as

\[
\Delta x = \tilde{x}_m - \hat{x}_1; \quad (3.17)
\]

for the x plane, and defined as

\[
\Delta y = \tilde{y}_m - \hat{y}_1; \quad (3.18)
\]

for the y plane. The time difference, or difference of frames, is defined as

\[
\Delta t = \hat{n}_1 - \tilde{n}_m; \quad (3.19)
\]

where \( \hat{n}_1 \) is the first frame of the candidate track and \( \tilde{n}_m \) is the last frame of the drop track.

The last requirement of the distance filter is the filter allowance. As the distance and velocity equations stand now, two different filter allowances are required, one for the x axis and one for the y axis. Because the heading filter concerns itself entirely with direction, the magnitudes of the distances and velocities can be taken. This means only a single filter allowance value is required for the distance filter. The velocity of \( \tilde{X} \) can be used to estimate how much distance is changed in a single frame. Dividing the magnitude of the distance by the time results in the average distance traveled in a single frame. The comparison of the euclidean distance between the drop track and candidate track and the instantaneous velocity of the drop track is

\[
\left| \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\Delta t} - \sqrt{\tilde{V}_x^2 + \tilde{V}_y^2} \right| < D_e \quad (3.20)
\]

where \( D_e \) is distance filter allowance. If the track pair successfully passes the distance filter, the distance filter sends the pair to the velocity filter. Otherwise the track pair does not
proceed and is considered to be unmatchable, and a new track pair is loaded into the time filter. Tuning of filter allowance value is described in detail in Section 4.4.

Because each track has x and y velocity components, two velocity allowances are required, one for the x axis and one for the y axis. The heading filter specifically vets the validity of the tracks’ heading so, like in the distance filter, the magnitude of the velocities can be looked at. Now the filter only needs a single velocity allowance, $V_e$, in order to filter the track pair. The equation for passing the velocity filter is

$$\left| \sqrt{v_{x}^2 + v_{y}^2} - \sqrt{\hat{v}_{x}^2 + \hat{v}_{y}^2} \right| < V_e \tag{3.21}$$

The difference in the magnitude between the drop and candidate tracks’ velocities is divided by the difference in frames to obtain the average acceleration required per frame. If the average acceleration required is below the specified threshold, the pair is rejected and a new pair is loaded into the time filter. If the track pair passes the filter it instead moves to the final kinematic filter, the heading filter.

The heading filter is designed to determine if the change in heading from the drop track to the candidate track is feasible. The angle of difference, $\theta$, is between the tracks in the the track pair and is found using the following equation:

$$\theta = \cos^{-1} \left( \frac{\hat{v} - \hat{v}'}{\sqrt{\hat{v}_{x}^2 + \hat{v}_{y}^2} \sqrt{\hat{v}'_{x}^2 + \hat{v}'_{y}^2}} \right) \tag{3.22}$$

where the numerator is the dot product of the drop and candidate velocities and the denominator is the product of the drop and candidate velocities’ norms. The equation for passing the heading filter is

$$\theta \leq H_e \tag{3.23}$$

where $\theta$ is the difference in angle between the drop and candidate tracks and $H_e$ is the heading filter allowance which determines the amount of heading deviation that is allowable.
to pass the heading filter. If the track pair passes the filter, it is finished with the kinematic filter and moves on to scoring, otherwise the track pair is discarded and a new track pair is loaded into the time filter.

### 3.4 Scoring the Track Pairs

The scoring filter’s goal is to assign each track pair a kinematic score. Figure 3.6 shows an overview of how the track pairs are scored. The first step in the scoring process is to predict where $\tilde{X}$’s centroid would be during the first frame of $\hat{X}$ if $\tilde{X}$ kept its last known trajectory. The predicted track information is determined by first creating a state transition matrix $\phi$ which is

$$
\phi = \begin{pmatrix}
1 & 0 & \Delta t & 0 \\
0 & 1 & 0 & \Delta t \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
$$

(3.24)

where $\Delta t$ is the difference in time between the drop track, $\tilde{X}$, last frame and the candidate track, $\hat{X}$, first frame. With the state transition matrix $\phi$ created, $\phi$ can be used to predict what the position and velocity of $\tilde{X}$ are at the beginning frame of the candidate track, $\hat{X}$, using

$$
\hat{X}_p = \begin{bmatrix}
x_m \\
y_m \\
x_{\hat{V}} \\
y_{\hat{V}} \\
\end{bmatrix}
\phi
$$

(3.25)

where $\hat{X}_p$ is $\hat{X}$ predicted forward $\Delta t$ frames, $x_m$ and $y_m$ are the x and y centroid position in the last frame, and $\hat{V}_x$ and $\hat{V}_y$ are the x and y velocities for the final frame. In the predicted track, $X_p$, only the x and y centroid will be changed. The velocities in $X_p$ will be the same as they were in the last frame of $\tilde{X}$. With the predicted location of the drop track determined, a score can be generated relating the kinematic that a track pair should be stitched together.
Figure 3.6: Architecture of the scoring method

1. Predict the dropped track $\tilde{X}$ forward $\Delta t$ frames using the state transition matrix $\phi$. $\tilde{X}$ then becomes $\tilde{X}_p$.
2. Compute the likelihood score, $L_k$, of $\tilde{X}_p$ and $\tilde{X}$.
3. Is $L_k$ less than the likelihood maximum, $L_M$?
   - Yes: Go back to Scoring Process
   - No: Use Selection Algorithm
The kinematic score is generated from the kinematic information of the track pairs. The \( \tilde{x}_p, \tilde{y}_p, \tilde{V}_{px}, \) and \( \tilde{V}_{py} \) are the x and y centroid of \( \tilde{X}_p \) and the x and y component velocities of the predicted forward frame. The difference in the centroid coordinates of \( \tilde{X}_p \) and the first frame of \( \tilde{X} \) are

\[
\Delta x_p = \tilde{x}_p - \dot{x}_1 \quad (3.26)
\]

\[
\Delta y_p = \tilde{y}_p - \dot{y}_1 \quad (3.27)
\]

and the difference in the velocities of \( \tilde{X}_p \) and the first frame of \( \tilde{X} \) are

\[
\Delta V_{xp} = \tilde{V}_{xp} - \dot{V}_x \quad (3.28)
\]

\[
\Delta V_{yp} = \tilde{V}_{yp} - \dot{V}_y. \quad (3.29)
\]

With the difference in \( \tilde{X}_p \) and \( \tilde{X} \) determined, the kinematic score, \( L_k \), for a pair is obtained as

\[
L_k = \sqrt{\Delta x_p^2 + \Delta y_p^2} + \sqrt{\Delta V_{xp}^2 + \Delta V_{yp}^2} \times (1\, frame). \quad (3.30)
\]

The more closely related the centroids and velocities of the track pair are, the lower the score value. Conversely, greater disparity between centroids and velocities leads to an increase in the kinematic score value. For this implementation of track stitching, lower scores are desired in potential track pairs. If, for example, the centroids and velocities of \( \tilde{X}_p \) and \( \tilde{X} \) are identical, Equation 3.30 will result in a kinematic score of 0, the lowest score possible. On the other hand, a score of 1000 would mean that the track pair is less likely to belong to the same object.

With a kinematic score generated, the score can now be filtered. The scoring filter exists as the last method of removing track pairs that would be bad stitches. The kinematic score is compared against a specified kinematic maximum, \( L_M \), and the scoring filter is

\[
L_k \leq L_M. \quad (3.31)
\]
Passing the scoring filter means that the track pair will be sent to the selection algorithm, whereas track pairs that fail are not considered for stitching. Higher $L_M$ tend to allow more stitches to be made but also increases the chances that poor stitch pairs are also stitched. Finding the optimal $L_M$ is extremely important in order to maximize the performance of the track stitcher. Tuning of $L_M$ is described in detail in Section 4.4.

### 3.5 Selection Algorithm: Munkres

Once every potential track pair has been filtered, all of the track pairs that passed the filters are sent to the selection algorithm. The goal of the selection algorithm is to determine which pairs are stitched together based on the likelihood score, $L_k$, generated. The selection algorithm is particularly important because it is possible for multiple track pairs with a shared track to make it through all the previous filters. When this happens, the algorithm determines which pair in these cases is stitched together. For example, if two of the track pairs that pass all the filters are $\tilde{X}_A$, $\tilde{X}_B$ and $\tilde{X}_A$, $\tilde{X}_C$, only one of the track pairs is the best stitch pair. In the simple example presented, the pair that has the lowest kinematic score is the one that will be stitched together.

In this implementation of the track stitcher, the selection algorithm used is Munkres assignment algorithm. The Munkres algorithm is an optimization algorithm that looks to find the optimal solution through minimizing the cost of assigning stitch pairs. To use Munkres, all of the track pairs are converted into a large selection matrix, with the corresponding kinematic score, $L_k$, entered as the cost for the pair. Once the selection matrix is generated, Munkres algorithm can be used to obtain the optimal stitch list. Table 3.1 shows an example of a selection matrix that is inputted into Munkres algorithm. In this example, some notable decisions the algorithm will have to make are: whether track 18 should be
stitched to track 440 or 256, if track 45 should be stitched to 267 or 406, and if track 501 should be stitched to 108 or 368. In order to run Munkres on the selection matrix shown in table 3.1, all the elements that do not have a kinematic score are set to an arbitrarily high number. In this way, the selection matrix is in a form that can be used in the Munkres algorithm without the values impacting the optimal solution.

In order to better understand the results that Munkres outputs, an analysis of the Munkres algorithm is necessary. Munkres algorithm, or the Hungarian Method for the Assignment Problem, was first proposed by Kuhn in 1955 [30]. It consists of a four-step process to determine which assignments, or, in the case of the track stitcher, stitches to make. Munkres requires an \( nxm \) matrix with every element having an associated cost, such as the matrix in Table 3.2.
When provided a matrix of this form, step 1 is to find the smallest element of each row and subtract it from each element in its row. In the case of the matrix in Table 3.2, 7 is subtracted from the first row, 8 from the second row, 3 from the third row, and 10 from the last row. The resulting matrix is shown in Table 3.3.

Once the minima is subtracted from each row, step 2 requires that the smallest element of each column be subtracted from each element in its column. For the matrix in Table 3.3, 3 is subtracted from the first column, 0 from the second, 0 from the third, and 3 from the last column. The resulting matrix is shown in Table 3.4.

Step 3 is to cover all zeros with as few lines as possible. If the number of lines required is equal to the matrix’s size, the optimal solution has been determined. In this case, four line are the minimum required to cover all the zeroes. Therefore, the matrix can make the optimal assignment. The bold elements in Table 3.5 are the optimal solution for the matrix after Munkres is performed. When the optimal solution is applied to the original matrix
Table 3.5: Step 3: Munkres example with bold elements mark the optimal solution

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>37</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>56</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>4</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>42</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: Optimal solution from step 3 being applied to the original matrix. Bold elements mark the optimal solution

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>7</th>
<th>44</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>64</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>7</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>52</td>
<td>10</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

from Table 3.2, the result is shown in Table 3.6 where the optimal solution are the bolded elements.

In the event that executing the above 3 steps fails to provide an optimal solution, there is an additional step. When step 3 is completed and the condition for the optimal solution is not met, an additional step is added. Step 4 finds the minimum element out of all elements that are not covered by a line. The minimum element from all uncovered elements is subtracted from the uncovered elements, while at the same time the minimum element is added to all elements that are covered by two lines. Step 3 and, if necessary, step 4 are repeated until an optimal solution is reached. Table 3.7 shows a matrix that has failed to find an optimal solution in step 3. The bold elements in this matrix denote lines covering the rows and columns. Applying step 4 to this matrix, the matrix in Table 3.8 is obtained, where the bold elements represent the optimal solutions when analyzed with step 3 of Munkres algorithm.
Table 3.7: Matrix without an optimal solution after step 3 of the Munkres algorithm. Bold elements represent covered rows and columns

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>90</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.8: Step 4: The matrix from Table 3.7 after undergoing step 4 of the Munkres algorithm. Bold elements represent the optimal solution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>18</td>
<td>40</td>
</tr>
<tr>
<td>0</td>
<td>58</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>96</td>
<td>0</td>
</tr>
</tbody>
</table>

When the Munkres algorithm is applied to the selection matrix of the track stitcher, the resulting output is the best pairing between drop and candidate tracks. Based on the optimal track pairs listed from Munkres, the only step left is to merge the listed drop and candidate tracks together. Once this is accomplished, the track stitching process is complete.
The track stitcher should be able to detect and stitch all the appropriate track pairs together across any tracking scenario, but in practice a track stitcher must be tuned to respond optimally to a track set from a scenario. Tracking scenarios have numerous variables, such as the size of the image regions, the time between image frames, the number of frames in a set of images, the type of objects being tracked within an image, or even the background of the image. For example, using the same kinematic filter allowances $D_e$, $V_e$, or $H_e$ for data sets with different coordinate frames will likely result in poor track stitching. This brings up the important question of how to determine if the output of the track stitcher is actually better than the tracks from GATER. For example, if a dropped track, $\tilde{X}$, is erroneously stitched to a candidate track, $\hat{X}$, that only exists as a result of an image artifact, the resulting stitch track, $\bar{X}$, will not be representative of the real track. In a tracking or surveillance applications, this could result in the stitched tracks having poorer performance metrics than non-stitched tracks.

In order to test the track stitcher, three image sets are registered and put through the GATER tracker to obtain three unique track sets. Each of these track sets is preprocessed, explained in the preprocessing section below. Once the image sets have gone through the tracker and have been preprocessed, a single track set is selected to be the tuning test
case. This case is run through the track stitcher for various filter and scoring parameters. Once the stitched tracks for the tuning set have been formed, the results are put through a metric software suite called COMPASS Tracker Evaluation Software Suite (CTESS)[31]. The CTESS software provides the metrics for the tracks before preprocessing, after preprocessing, and after the track stitcher has been run. By comparing the CTESS results of the tuning set to the non-stitched tracks with and without preprocessing, a set of optimal filter parameters is obtained that are subsequently used to test the track stitcher on the other track sets. In order to use the CTESS software and determine if the track stitcher’s results are an improvement over the tracker’s tracks, CTESS requires a truth track set to be manually created.

4.1 Establishing Truth

A track is a computer system’s attempt at relating objects’ points of interest between frames. The tracks’ truth is created by manually selecting and grouping the points of interest of an object. Properly truthed tracks are the results that a tracker and track stitcher should seek to replicate completely. The truth set’s POIs for an object truth are $P_T$, which is defined as

$$P_T = [x_T, y_T, n_T],$$

(4.1)

where $T$ is a single point of interest, $x_T$ is x position of an object, $y_T$ is the y position of an object, and $n_T$ is the frame the object appears in. This should look extremely similar to Equation 3.1 that defines $P_k$, where the only difference is that $P_T$ is selected manually and $P_k$ is formed within the tracker. The truth track, or object truth, $O_i$, is defined as

$$O_i = \begin{pmatrix} P_{t_1} \\ \vdots \\ P_{t_m} \end{pmatrix} = \begin{pmatrix} x_{t_1} & y_{t_1} & n_{t_1} \\ \vdots & \vdots & \vdots \\ x_{t_m} & y_{t_m} & n_{t_m} \end{pmatrix},$$

(4.2)
where $i$ is the object number for a given object truth and $t$ is a vector or set of vectors associated with a given object number (subset of $T$). This mimics the form of a track, $X_i$, given in Equation 3.2, except that the points to be associated with a given object truth are manually picked instead of grouped by a tracker or track stitcher. With the truth for a data set constructed, it is possible to determine how well the tracker and track stitcher’s tracks compare against the ideal best case scenario, represented by the truthed set. The metrics that will be used to determine the performance are purity, fragmentation, and spuriousness, based on three of the six parameters used to evaluate the performance of tracking algorithms by Yin in 2007 [32].

4.2 Metrics: Purity, Fragmentation, and Spuriousness

Purity is the fraction of an object’s truth has a correct dominant track association. A dominant track is the track that is most closely associated with a given object truth. In the event that two tracks are closely associated with an object truth, the track with more matches will be considered the dominant track. If a single track has all the same points as the object truth, the track will have a purity of 1. A purity of .5 indicates the highest matching track only has half the matches to the object truth and a 0 purity means there are no tracks that have a single match to the object truth. The purity of a track, $M_{pure}$ is determined by

$$M_{pure} = \frac{\# \text{ of truth object's primary associations to dominant track}}{\# \text{ of truth object's primary associations}} \quad (4.3)$$

where the numerator is the number of times a dominant track exists for a point of interest in a object truth and the denominator is the number of points of interest in the object truth. When the purity of the tracks from the tracker are compared to that of the track stitcher, the track stitcher should always have purity equal to or greater than the tracker.
The second metric is fragmentation. Fragmentation is a fraction representing the number of tracks that are associated with an object truth, given by

\[ M_{frag} = \# \text{ of tracks associated with object truth}. \quad (4.4) \]

Unlike purity, which only concerned itself with the dominant track for a given truth, fragmentation looks for how many unique tracks are associated with a given object truth. The ideal fragmentation for a track is 1, which means that there exists only a single unique track that is closely associated with an object truth. If a fragmentation score of 2 appears for a truth, that means that there are two unique track IDs closely associated with the truth. The fragmentation score that will be looked at is the average fragmentation, \( A_{frag} \), which is calculated by

\[ A_{frag} = \frac{\sum M_{frag}}{N_{objects}}; \quad (4.5) \]

where the numerator is the summation of the individual fragmentation scores for each object truth and \( N_{objects} \) is the number of object truths that exist. The track stitcher’s average fragmentation score should always be less than the tracker’s fragmentation score, as each successful stitch eliminates a unique track that would have been associated to an object truth. If the fragmentation score remains unchanged, it means that either stitches did not occur, that all the stitches that did occur were wrong, or that there were no stitches that neccessary. These wrong stitches could take the form of a track that is closely associated with the object truth being stitched to track that was made up of false positives detections, or a pair of false positives being stitched together.

The third metric is spuriousness. A spurious point is a POI that is not spatially close to an object truth. POIs that result from image artifacts or from projected points (discussed in Section 3) are typically classified as spurious points. The spuriousness score, \( M_{spur} \), is
calculated by

\[ M_{\text{spur}} = \frac{\# \text{ of spurious points}}{N_{\text{frames}}} \]  

(4.6)

where \( N_{\text{frames}} \) is the number of frames included in the image set. An \( M_{\text{spur}} \) of 0 indicates that all points of interest that comprise the test track are associated to a point of interest in the truth, whereas a score of 10 means that for each frame there are 10 spurious points. The \( M_{\text{spur}} \) tends to increase for every incorrect stitch the tracker makes because of the presence of projected points, making spuriousness a good indicator of how much error running the track stitcher adds to the tracks.

### 4.3 Projected Points

Before the stitcher can be run on the tracks that result from GATER, preprocessing has to be done on the tracks. The way that the tracks from GATER are reported creates an issue for the proposed track stitcher. The projected points that are predicted by GATER for a given track are included in the track’s output. When GATER misses one or two detections for a track but then makes a real detection before reaching the maximum coast counter, the projected points do not cause an error for the stitcher. However, when a track dies and its ID number is retired, there are a number of projected points recorded as real tracks that are reported in the output. This results in the tracks from GATER having their length artificially increased and in track end times that do not reflect real detections.

Table 4.6 shows a comparison of track lengths and associated spuriousness, generated by CTESS, for image sets with the projected tracks included and with the projected tracks removed. Across the board, the image sets with the projected tracks included have nearly twice the track length as the non-projected track sets. The spuriousness is higher with the projected tracks as well, particularly in sets 1 and 3.
### Table 4.1: Table comparing average track length and spuriousness of image sets with and without projected points

<table>
<thead>
<tr>
<th>Image Set</th>
<th>Average Track Length in Frames</th>
<th>Spuriousness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With Projected</td>
<td>Without Projected</td>
</tr>
<tr>
<td>Image Set 1</td>
<td>6.7059</td>
<td>3.8824</td>
</tr>
<tr>
<td>Image Set 2</td>
<td>8.7059</td>
<td>4.2669</td>
</tr>
<tr>
<td>Image Set 3</td>
<td>9.8571</td>
<td>4.7273</td>
</tr>
</tbody>
</table>

In events where a track is dropped and GATER misidentifies a detection of the same object as the beginning of a new track within one or two frames, the time filter of the track stitcher would look at the data as reported and see that the tracks in the potential pair exist at the same point of time and the pair would not be considered for stitching any further. In order to circumvent this, the data from GATER must have all the projected tracks that do not occur between real detections removed. The preprocessing done on the GATER output removes the projected tracks so that the end point reported for each track is the last place a real detection of a POI occurred, instead of a projected point. All tests that did not include this preprocessing step resulted in the track stitcher breaking down and discarding numerous potential track pairs because of time overlaps. The only stitches that were made in this case were between tracks with more than four frames of difference between them.

One metric that the presence of projected tracks does greatly increase is the purity. Table 4.2 shows the difference in purity between tracks with the projected points included and tracks without the projected points. For each image set, the tracks with the projected points included have a higher purity than when the points are removed. The difference between tracks with and without the project points included is very low for image sets 1 and 3, so removing them for track stitching will not cause a significant loss of purity. Image set 2, however, has a significant discrepancy between the tracks with the projected points
<table>
<thead>
<tr>
<th></th>
<th>Purity With Projected</th>
<th>Purity Without Projected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Set 1</td>
<td>.8593</td>
<td>.855</td>
</tr>
<tr>
<td>Image Set 2</td>
<td>.941</td>
<td>.8675</td>
</tr>
<tr>
<td>Image Set 3</td>
<td>.8433</td>
<td>.8371</td>
</tr>
</tbody>
</table>

Table 4.2: Table comparing average purity of image sets with and without projected points

and without the projected points. This implies that for image set 1, the projected tracks are almost all located near real POIs but that the tracker, for whatever reason, could not make a proper detection.

Image set 2 shows how projected points can help increase the purity metric, and suggests that the useful characteristics of projected tracks should not be ignored. Therefore, the track stitcher should strive to make smart use of the projected points insofar that it doesn’t hinder the primary goal of reducing track fragmentation. The proposed method of utilizing projected points in the stitching process suggests that for every track pair that is part of the optimal stitching solution, the projected tracks that are removed from the end of the drop track that do not have a shared frame with the candidate track can be added back into the stitch track’s history. This will lead to an increase in purity, but also spuriousness. Adding small amounts of spuriousness can actually be of benefit to the track stitcher. By adding potentially spurious projected points into the newly stitched tracks, the spuriousness can be turned into an indicator of the error induced by false stitches. If spuriousness increases while purity and fragmentation remain constant, it means that false stitches are occurring.

4.4 Testing: Using Metrics for Tuning the Track Stitcher

With the evaluation metrics established the stitcher is tested using the three case image sets. The test cases’ imagery is from the Blue Devil Large Area Image Recorder (LAIR)
data set taken at Wright Patterson Air Force Base and surrounding areas. Each frame of the LAIR data is approximately 256 Mega-pixels. Three segments of the LAIR data is cropped into a 1024x1024 pixel area and registered. There are 20 frames of images for each cropped region and there is 0 compression in the imagery. The three sets of images are registered using GATER. GATER creates the tracks from the input images, using a pixel space reference. Therefore, all the centroids correspond to pixel locations. For each of the three data sets, the average pixel length of a car is determined to be 15 pixels, obtained using MATLAB’s ginput function.

Because the image sets are similar, a single set of parameters should be applied to the three image sets. The parameters that must be determined for the track stitcher to be optimized for a given set of images are: the minimum and maximum time gap for stitching, the allowance for the kinematic filters, and the kinematic maximum. The easiest parameters to determine are the minimum and maximum time gaps. The GATER object tracker has a user controlled parameter that determines the minimum number of frames in which an object must be detected in order for GATER to register the object as a track. In the case of these image sets, GATER requires three frames of confirmation for a track and then establishes the track’s starting location as the first frame the object was detected within. This means that if a track is dropped, it is possible that it could be detected by the tracker as a new object starting at the next frame. So, for this image set, the minimum time gap is selected as 1.

The maximum time gap can be determined in a variety of ways. The first method is to set it at the full number of frames, ensuring that all tracks that do not exist simultaneously are compared to one another. For large image sets, this can slow the track stitcher down as there are more comparisons that must be processed. The second method is to use
information known about the imagery. For example, consider a highway underpass with cars traveling at similar speeds. If it is known that a car upon entering the image frames leaves the image frames after about 10 frames have passed, having a maximum time gap of 30 frames could be considered too great a discrepancy. There are, of course, exceptions depending on the imagery but this is a good general rule. If metrics for the pre-stitching imagery are available, the maximum time gap can be determined by looking for the longest pre-existing track. For example, there is an average track length metric from CTESS that can be used to obtain the individual track lengths. For one of the image sets, there is a track that is 13 frames long after projected track points are removed and before any stitching occurs, implying that an accurate maximum time gap should be at least 13 frames. It is also known that there are 20 frames comprising each image set. Looking at the imagery of image set 1 (see appendix), it can be seen that many of the cars are not moving fast enough to exit the image frame region unless they start near the edges of the image frame. Therefore it is possible that a track at the end of the image sets’ frames contains an extension of an earlier frame. Using the aforementioned approach, a maximum time gap of 20 is selected for the track stitcher, ensuring that every track that ends is compared to all of the tracks that start later.

The last two parameters that the track stitcher requires, the kinematic filter allowance and kinematic maximum, are more complicated to optimize. In order to find optimal values for these two parameters, one image set, called the ”control image set”, will be selected to be optimized by analyzing metrics for differing filter allowances and kinematic maximums. After the optimal allowance and kinematic maximum are obtained for the control image set, the track stitcher is run on the other two image sets using the optimized parameters. The rationale for this is that once the optimized parameters are determined for a given type
of image frames, the track stitcher should be properly optimized for all similar subsequent image sets.

4.4.1 Controlled Image Set Parameter Optimization

Image set 1 is used to optimize the track stitchers parameters. Figure 4.1 shows the first frame of the image set that GATER obtains the tracks from. The kinematic allowance for the heading, $H_e$, must be able to accommodate turning cars. Looking at additional frames, Figure 4.2 and Figure 4.3, a car from stop makes nearly a 90 degree turn between the frames, which are three frames apart. The parameters that are held constant for the tuning are the minimum time gap at 1 frame, the maximum time gap at 20 frames, and

$$H_e = 45^\circ \ast \Delta t$$

which errs on the safe side and varies $H_e$ by 45 degrees per frame of difference between $\hat{X}$ and $\tilde{X}$. The velocity and distance allowances, $V_e$ and $D_e$, will be equalized so that $V_e$ and $D_e$ are equal to one another for each test. A $V_e$ and $D_e$ based on one car length, where a car length is 15 pixels for the test image sets, of 15 pixels, will pass track pairs that have a centroid difference of less than one car length and a velocity difference of less than a car length per frame. These two filter parameters will be varied based on the average car length, which in these image sets is determined to be 15 pixels in length. These two kinematic filter allowances will be tested as .5 car lengths (7.5 pixels), 1 car length (15 pixels), 2 car lengths (30 pixels), and 3 car lengths (45 pixels). Meanwhile, the kinematic maximum, $L_M$, will also be varied. $L_M$ will be tested as 25, 50, 100, and 250. Each combination of filter allowances and $L_M$ will be tested, for a total of 16 different track stitching scenarios.

The first metric analyzed is the purity. Table 4.3 shows the purity scores for image set 1’s track stitcher results. A graph of the results can be seen in Figure 4.4. The test cases with
Figure 4.1: First frame of image set 1
Figure 4.2: First frame of car turning, image set 1
Figure 4.3: Second frame of car turning, image set 1
the highest purity score, .9033, are filter allowances of 1, 2, and 3 car lengths when combined with a kinematic maximum of 50. Of the filter allowances, .5 car lengths performed worse than the other three for each associated kinematic maximum. The kinematic maximum of 50, when paired with a filter allowance of .5 car lengths, performs the worst of any case scenario with a purity of .8485. The fact that increasing the kinematic maximum from 50 to 100 or 250 causes a drop in purity indicates that, while a higher maximum can allow more potential stitches, the quality of the stitch can suffer as a result. Based on the results, the best $L_M$ appears to be 50 because, despite underperforming when paired with an allowance of .5 car lengths, all the other test cases using $L_M$ as 50 have the highest scores. The best filter allowance is not as obvious. The worst allowance, .5 car lengths, is evident, but no conclusions can be drawn regarding the best allowance before analyzing the other metric results.

The second metric analyzed is the fragmentation. Table 4.4 shows the fragmentation scores for image set 1’s track stitcher results. Figure 4.5 shows a graph of the fragmentation scores from Table 4.4. The worst fragmentation score, 1.875, occurs when the $L_M$ is 25. Of the kinematic filter allowances, .5 car lengths once again tests worse than the other three allowance values. The best fragmentation result, 1.375, is obtained for all combinations of allowances 1, 2, and 3 car lengths and $L_M$s of 50, 100, and 250. The fragmentation

<table>
<thead>
<tr>
<th>Kinematic Filter Allowance in Car Lengths</th>
<th>.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematic Maximum</td>
<td>25</td>
<td>0.8629</td>
<td>0.8629</td>
<td>0.8629</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.8485</td>
<td><strong>0.9033</strong></td>
<td><strong>0.9033</strong></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.8615</td>
<td>0.8903</td>
<td>0.8903</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>0.8615</td>
<td>0.8903</td>
<td>0.8903</td>
</tr>
</tbody>
</table>

Table 4.3: Table containing the purity scores for each combination of filter allowance and kinematic maximum. Scores in blue indicate the best performing tests.
scores reinforce that the best $L_M$ value is 50, but the scores still do not provide any further insight into what the best filter allowance is. It is also important to note that the changing fragmentation scores mean that the track stitcher is stitching track pairs together successfully.

The third metric analyzed is the spuriousness. Table 4.5 shows the spuriousness scores for the image set 1’s track stitcher results. A graph of the spuriousness scores is shown in Figure 4.6. The best spuriousness value, .9286, is obtained for all allowances at a maximum of 25. It is known from Figures 4.4 and Figure 4.5 that this is because the least amount of stitching occurred for these results, allowing for fewer projected tracks to be introduced.
Table 4.4: Table containing the fragmentation scores for each combination of filter allowance and kinematic maximum. Scores in blue indicate the best performing tests.

<table>
<thead>
<tr>
<th>Kinematic Filter Allowance in Car Lengths</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematic Maximum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>1.875</td>
<td>1.875</td>
<td>1.875</td>
<td>1.875</td>
</tr>
<tr>
<td>100</td>
<td>1.75</td>
<td>1.375</td>
<td>1.375</td>
<td>1.375</td>
</tr>
<tr>
<td>100</td>
<td>1.65</td>
<td>1.375</td>
<td>1.375</td>
<td>1.375</td>
</tr>
<tr>
<td>250</td>
<td>1.625</td>
<td>1.375</td>
<td>1.375</td>
<td>1.375</td>
</tr>
</tbody>
</table>

Figure 4.5: Fragmentation graphs for image set 1
Table 4.5: Table containing the spuriousness scores for each combination of filter allowance and Kinematic maximum. Scores in red indicate the worst performing tests.

<table>
<thead>
<tr>
<th>Kinematic Filter Allowance in Car Lengths</th>
<th>.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematic Maximum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.9286</td>
<td>0.9286</td>
<td>0.9286</td>
<td>0.9286</td>
</tr>
<tr>
<td>50</td>
<td>1.0714</td>
<td>1.5714</td>
<td>1.5714</td>
<td>1.5714</td>
</tr>
<tr>
<td>100</td>
<td>1.2143</td>
<td>1.7143</td>
<td>1.9286</td>
<td>1.9286</td>
</tr>
<tr>
<td>250</td>
<td>1.2143</td>
<td>1.7143</td>
<td>1.9286</td>
<td>1.9286</td>
</tr>
</tbody>
</table>

back into the track stitcher’s results. The best spuriousness score that coincides with the best fragmentation and purity scores is 1.5714. This score is achieved for a maximum of 50 and filter allowances of 1, 2, and 3 car lengths. When $L_M$ is increased to 100 and 250, the spuriousness scores for allowances 1, 2, and 3 car lengths all increase, therefore worsening. Because the spuriousness scores for these tests increase, coupled with a lack of fragmentation change and a lower purity score, it is evident that more false stitches occurred at the higher maximums. Of these three filter allowances, 1 car length has the smallest spuriousness score after the increase than filter allowances 2 and 3 car length. This means that an allowance of 1 car length makes less false stitches than the other two tracks. As a result, the optimal parameters for this image set are selected as a filter allowance of 1 and $L_M$ of 50.

The optimized track stitcher results of image set 1 are now compared the track output from GATER with projected points included and excluded. Table 4.6 shows the comparison of the three scenarios. The track stitcher has the highest purity score, .9033, and least amount of track fragmentation, 1.375. When the fragmentation and spuriousness scores of the tracks that have not been run through the track stitcher are compared to the results of Figure 4.5 and Figure 4.6, it can be seen that in every test with an $L_M$ of 25 no stitches occurred. The tracks from the track stitcher unsurprisingly has more spuriousness than the
Figure 4.6: Spuriousness graphs for image set 1
tracks without projected points, but the track stitcher’s results have half as many spurious points as GATER’s tracks with the projected points included. The track stitcher performs better in every category over the original output tracks, proving that the track stitcher can successfully reduce fragmentation and improve track quality. This test was done to tune the track stitcher using the control image set, so next the track stitcher is tested using the optimized parameters of image set 1 on the tracks of the other two image sets.

4.4.2 Optimized Track Stitcher on Image Set 2

Figure 4.7 shows the first frame of the image set that GATER obtains the tracks from. Table 4.7 shows the comparison between the tracks with projected points included, without projected points, and after the track stitcher is run. In the first category, purity, the original GATER tracks perform the best, with a score of .941. The track stitcher has a purity of .9298, a purity that is .0112 lower. The track stitcher’s fragmentation is less than the other two tracks, at 1.1875. The spuriousness is lowest in the track set without the projected points, with the track stitcher results having a slightly lower spuriousness score than the original tracks. When it comes to reducing fragmentation, the track stitcher completed its goal of reducing fragmentation. Interpreting the purity results can be slightly trickier.
Table 4.7: Table comparing the purity, fragmentation, and spuriousness of image set 2’s tracks with the projected points, without the projected points, and after the track stitcher

<table>
<thead>
<tr>
<th></th>
<th>GATER Tracks</th>
<th>GATER Tracks with Projected Points</th>
<th>Track Stitcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purity</td>
<td>.8675</td>
<td>.941</td>
<td>.9298</td>
</tr>
<tr>
<td>Fragmentation</td>
<td>1.3125</td>
<td>1.3125</td>
<td>1.1875</td>
</tr>
<tr>
<td>Spuriousness</td>
<td>1.3571</td>
<td>1.9286</td>
<td>1.8571</td>
</tr>
</tbody>
</table>

As it turns out, this test case actually supports the argument for re-introducing select projected tracks back into the track stitcher’s final output. Based on the purity and spuriousness results, the projected points actually account for a significant portion of the accurate tracks for the image set. This determination is made because there is a significant difference between the purity of the tracks with projected points and the tracks without projected points. While such a large purity discrepancy exists between these two tracks, the spuriousness difference is relatively low. This means that when all the projected points were removed, the error was only slightly reduced, while the purity was greatly reduced. The track stitcher, by adding select projected points back in, is able to increase purity by a large amount over the removed point case, while only adding slightly more error into the results. Coupled with the obvious decrease in track fragmentation from 1.3125 to 1.1875, the track stitcher is successful in improving the tracks of this image set.

4.4.3 Optimized Track Stitcher on Image Set 3

The first frame of image set 3 is shown in Figure 4.8. The comparison between the tracks with the projected points, without the projected points, and post stitching are shown in Table 4.8. The highest purity belongs to the track stitcher with a score of .9274. This score is significantly higher than the original track score of .8433. The track stitcher’s output also has the lowest fragmentaion score, 1.1329. Once again, the lowest spuriousness is the track
Figure 4.7: First frame of image set 2
Table 4.8: Table comparing the purity, fragmentation, and spuriousness of image set 3’s tracks with the projected points, without the projected points, and after the track stitcher with the projected points removed, with a score of 5.000. The track stitcher’s spuriousness is 7.5000, lower than the original track’s spuriousness of 9.8671.

Despite the track stitcher being optimized for the tracks in image set 1, the track stitcher provides an across-the-board improvement to the tracks of image set 3. These results reinforce the idea that the track stitcher can be optimized on one set of images and subsequently applied to great effect on a similar image set. The inclusion of projected points is proven to boost to the track stitcher’s performance when it comes to track purity, while still enabling the track stitcher to accomplish its primary task of track fragmentation reduction.

4.4.4 Results Overview

From the three tests, a number of conclusions can be made. The track stitcher’s primary goal is to have a lower fragmentation score than GATER’s tracks with projected points. The track stitcher succeeds at reducing fragmentation in each test scenario when compared to GATER’s output, both with and without the projected tracks. Based on the decrease in fragmentation, fragmented tracks were present and fixed in each image set. When it comes to increasing track purity, the track stitcher improves results in two out of the three cases, with the third case having only a small reduction in purity. Looking at the comparison of the GATER tracks with and without projected points, it can be seen that projected points play a large roll in how the track sets’ purity is scored, as the tracks with the extra points
Figure 4.8: First frame of image set 3
included have increased purity over the tracks with the points removed. However, based on the spuriousness the increased purity comes at a cost. In each of the test cases the spuriousness is lower in tracks with the projected points but higher than the track sets with the projected points removed. Based on these observations, it can be concluded that having projected points included in the tracks results in more spuriousness but also increases the purity. In the case in which the track stitcher results had less purity than the track with projected points included, the track stitcher still results in an improvement in purity over the tracks without projected points. The track stitcher is therefore proven to increase purity in all cases when only real detections are used to compute the metric results, with a slight drop in performance in one test case when projected points are included. The track stitcher is also proven to reduce the amount of error present in the tracks from the original results with projected points included. Overall, these results show the track stitcher improves the quality of the tracks that are analyzed.
CHAPTER V

CONCLUSION AND FUTURE WORK

The primary goal of this thesis is to demonstrate how the implementation and use of track stitching improves the tracks that are formed through the process of object tracking. I selected this topic in an effort to help further the industry goal of obtaining more accurate tracks in a multiple object tracking environment. Much of the research in this field has been dedicated to improving registration and tracking algorithms, but improvements in track quality can also be realized once tracks are established. In support of my thesis, I created a post tracking kinematic track stitcher that is able to reduce the fragmentation present in the tracks created from GATER’s projected points method.

My research into track stitcher development covered the early phases of the object tracking process, the events that can cause track fragmentation, the common elements of tracks from multiple trackers, and the process of making stitches to track pairs. By analyzing the early phases of the tracking process, including image registration techniques and multiple object tracking, the causes of track fragmentation are able to be more readily identified. Subsequently, the events that cause track fragmentation can be attributed to features of the image set, results of processing the images, or a combination of the two.

The track stitching process itself is predicated upon establishing how tracks relate to one another. The first component of the tracks relationship is the tracks’ temporal information.
A time filter is able to find the temporal relationship, or lack thereof, of track pairs. If there is a relationship, each member of the track pair is reclassified as either the drop track or the candidate track and is able to be analyzed further.

When the temporal relationship between tracks is established, the kinematic relationship of drop and candidate tracks is utilized. The kinematic information is first used to pare down the number of potential stitch pairs. The kinematic filters of the track stitcher are designed to remove track pairs that are kinematically unfeasible, a process commonly referred to as sanity checks. Once tracks with low kinematic similarity are removed, the remaining tracks are scored based on their similarity. The more similar the kinematic components of the tracks, the lower the score. Scores that fall below a user defined threshold are sent to the Munkres selection algorithm in order to finalize the list of track pairs to be stitched together.

My findings indicated that the performance of the track stitcher is dependent on the kinematic filter allowance and the kinematic score threshold. Finding the appropriate values for these parameters is crucial to track stitching. When the parameters are properly defined on a set of WAMI data sets, the track stitcher makes numerous stitches among the track pairs. The track stitcher has less fragmentation and spuriousness than GATER’s tracks with projected points. In one case, the track stitcher results in a slight decrease in purity from the projected points, but this is offset by the much greater decrease in fragmentation.

The design of the proposed track stitcher has room for expansion and additional future research. Based on my findings, I believe that implementing feature-based filters and scoring will significantly improve the results of the track stitcher by reducing the ambiguity of nearby objects with similar trajectories. Based on the track stitchers architecture, new filters can be seamlessly added in series, or in parallel, to the existing filters. Utilizing the object
features in this fashion would help mitigate any shortcomings in the kinematic analysis, such as accounting for car color or size of the objects being tracked. Further work should include exploration into the incorporation of the feature and kinematic information together for use in the scoring algorithm. For example, if the match ratio is used as a scoring metric for the features, what is the best way of combining the match ratio and kinematic score for better track results. Some experiments with integrating histogram feature information, aided by the kinematic lists, have shown promising results. Additionally, improvements could be made in determining which artificially added points should be included in the final track list, ideally leading to decreased error without sacrificing track purity in the process. Lastly, real-time applications are an area the track stitcher can develop and improve in, especially for surveillance applications that cannot afford to wait for results from a post-tracking algorithm. This will likely require finding ways of successfully integrating the track stitcher into the desired tracker suite.
BIBLIOGRAPHY


