2012-09-05

Automatic Content-Based Temporal Alignment of Image Sequences with Varying Spatio-Temporal Resolution

Samuel R. Ogden
Brigham Young University - Provo

Automatic Content-Based Temporal Alignment of Image Sequences with Varying Spatio-Temporal Resolution

Samuel R. Ogden

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2012

ABSTRACT

Automatic Content-Based Temporal Alignment of Image Sequences
with Varying Spatio-Temporal Resolution

Samuel R. Ogden
Department of Computer Science, BYU
Master of Science

Many applications use multiple cameras to simultaneously capture imagery of a scene from different vantage points on a rigid, moving camera system over time. Multiple cameras often provide unique viewing angles but also additional levels of detail of a scene at different spatio-temporal resolutions. However, in order to benefit from this added information the sources must be temporally aligned. As a result of cost and physical limitations it is often impractical to synchronize these sources via an external clock device. Most methods attempt synchronization through the recovery of a constant scale factor and offset with respect to time. This limits the generality of such alignment solutions. We present an unsupervised method that utilizes a content-based clustering mechanism in order to temporally align multiple non-synchronized image sequences of different and varying spatio-temporal resolutions. We show that the use of temporal constraints and dynamic programming adds robustness to changes in capture rates, field of view, and resolution.

Keywords: multi-resolution temporal alignment, video alignment, hierarchical clustering, dynamic programming

ACKNOWLEDGMENTS

Many thanks to my thesis adviser, Dr. Bryan Morse, for his time in providing valuable feedback and wisdom throughout this process, as well as to my committee members Dr. Michael Goodrich and Dr. Sean Warnick. Thanks to fellow Computer Vision students for assistance with general day to day implementation challenges.

Also, thanks to members of the collaborative research group, Wilderness Search and Rescue (WiSAR), for aiding in the acquisition of aerial footage for the purposes of experimentation.

Finally, a special thanks to my parents for their support and encouragement, and to my wonderful wife, Brittany, for her many sacrifices and countless hours spent offering support.

Without any of these individuals, certainly none of this would have been possible.

Table of Contents

List of Figures

Chapter 1

Introduction

With the growth of video usage, applications that use multiple imaging devices within a rigid configuration to simultaneously acquire content from different view points have become common. It is often the case where in such configurations, content from one source needs to be located in the alternate source as well. However, there needs to exist a temporal ordering or mapping between each image sequence, either partial or total, in order to identify such correspondences. To this end, we present a solution to the image sequence alignment problem that provides a mapping between two independently ordered image sequences. We also show that through the use of temporal constraints and dynamic programming, our method is robust to variations in temporal and geometric resolutions.

1.1 Motivation

Image sequence alignment, or video alignment, is the process of matching images from two or more independently ordered sequences (such as from a video stream or a series of images known to be in a particular order) that are most similar in content but in a fashion that maintains the original ordering of each sequence (Figure 1.1). We call this a temporal mapping or alignment. A somewhat trivial solution to this problem is to synchronize frames according to an external timing device. There are, however, several scenarios that limit the use of such hardware such as budget constraints, physical limitations of the system (e.g., distance between cameras), or situations where the data has already been recorded. A few applications that depend on the temporal alignment of sequences include stereoscopic

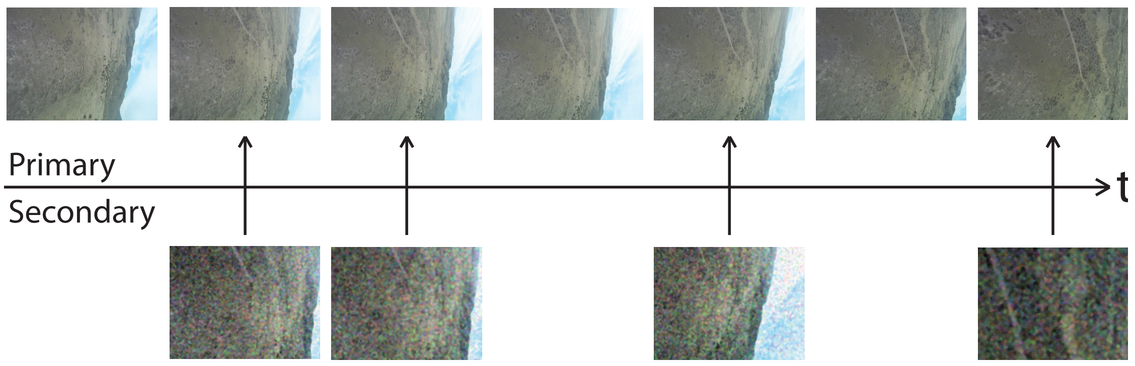

Figure 1.1: Temporal mapping between two image sources. The primary source is high-framerate, low-resolution, while the secondary source is low-framerate, high-resolution.

processing of multiple sequences, dynamic scene reconstruction [21], identification of content of interest in video surveillance, image mosaicking, and independent recordings of historical events. One particular application of our method is in video-aided search and rescue scenarios.

In Wilderness Search and Rescue (WiSAR) aided by unmanned aerial vehicles (UAV), the acquisition and streaming of video sequences provides critical imagery for ground searchers to be able to quickly identify areas of interest for directing further search (Figure 1.2). However, in many cases due to hardware constraints imposed by a relatively small UAV such as limited transmission bandwidth, image quality needs to be reduced in order to improve transmitted video playback quality (e.g. maintain a particular frame rate). Thus, in order to provide additional high-quality visual information the UAV is also mounted with a camera that can capture high-resolution imagery but that stores the images to a local memory card for later use. Due to hardware limitations only the low-resolution data is transmitted wirelessly back to the ground crew. Additionally, frames are dropped from the transmitted source in order to maintain playback quality, which further increases frame rate differences between sources. Finally, after the flight, offline searchers must be able to identify a valid mapping between both the low-resolution video sequence and the offline high-resolution images. By doing so, once content of interest is identified within one sequence its corresponding location in the alternate, aligned, source can be quickly determined.

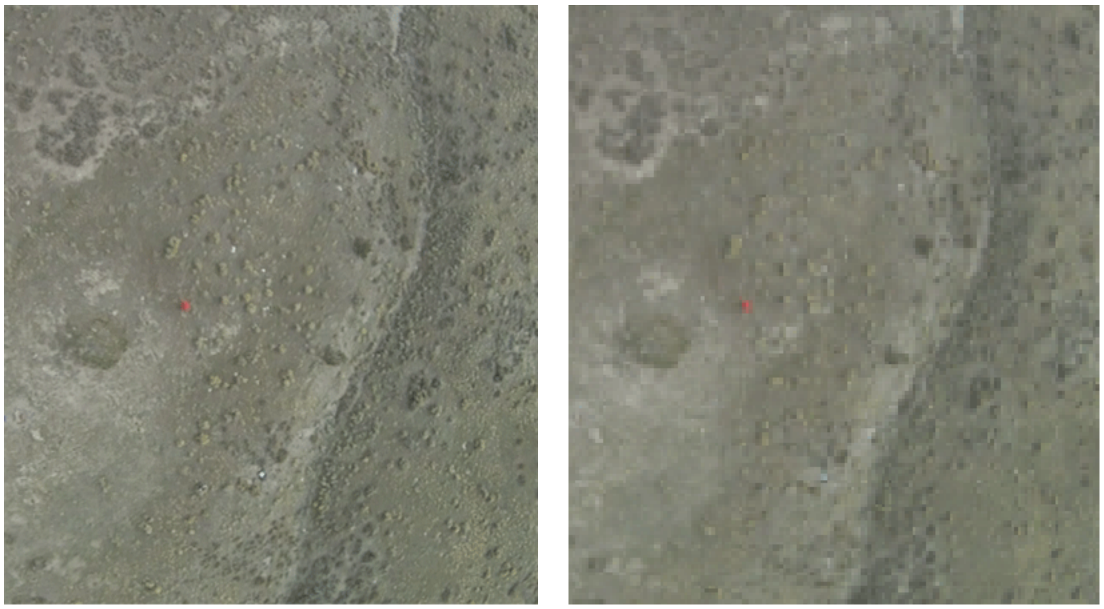
(a) High-Resolution (b) Low-Resolution

Figure 1.2: Example high- and low-resolution images transmitted from UAV to ground search crew.

1.2 Existing Work

Current literature categorizes solutions to the video alignment problem under two groups: feature-based [1, 3, 6, 9–11, 13, 15, 16, 18, 19, 21], and direct [2, 4, 14, 17]. Direct methods usually involve metrics for comparing images as a whole (such as in sequence-to-sequence alignment involving the minimization of differences in pixel intensities between images [2]). As a result, such methods suffer in accuracy from changes that affect an image as a whole such as differences in lighting or contrast between frames. Feature-based alignment addresses this problem by focusing on localized groupings of pixels (features) within an image and how they change (location, size, etc...) between images (possibly over time across multiple frames). An example of such is the use of a feature's path as it changes locations across multiple frames, its trajectory, to align different sequences [3, 10, 13]. Feature-based approaches generally work best when there is large variance between images as a result of a greater view baseline [9]. This is because many feature detection algorithms tend to be robust to modest changes in viewing angles. Current solutions in both categories make several assumptions that prevent the realization of a more general solution.

Most solutions assume that there is a constant time delay between sources [2–4, 10, 15–19]. This assumption means that once the start-time offset and the capture-rate scale factor between each source are recovered, corresponding frames between sources can be quickly computed. The problem with this however is that alignment calculations will be inaccurate for sources that have a variable frame rate, or intermittent footage (such as in a sequence of image still shots, or a system that drops frames to maintain playback quality).

Other solutions require that the camera system is stationary or rigid and that the subject is moving in order to accurately determine the path of a point or feature of interest across several frames, such as in [18]. In such a case, in order to recover a start-time offset and scale factor between sequences, trajectories from each sequence are matched in both time and space.

Perhaps most related is work by Fraundorfer [6] that uses a scalable image retrieval structure, proposed by Nister and Stewenius [8], as a means for robotic self-location. An offline database of images (of the area to be traveled) is queried with the robot's current view to find the most similar image match, and as a result the associated location. If the offline set was originally temporally ordered, this process over time could be used to define a temporal mapping.

Similarly, our solution utilizes a hierarchical structure as developed by [8] for efficient image comparison. However, we also impose temporal constraints through dynamic programming in order to explicitly determine a temporal mapping between sequences. Because we align sequences according to image content, there is no need to be concerned with consistency of frame rate. Thus we present a video alignment solution that is robust to changes in both spatial and temporal resolution.

1.3 Overview

1.3.1 Image Comparison

In order to find a temporal mapping between two image sequences, we must be able to quantify the similarity between two images. To this end, we use the image-based method described by Nister and Stewenius [8] which draws from a technique in Natural Language Processing referred to as "Bag of Words" for content-based text-document search.

This technique is used in traditional document retrieval for being able to rank documents according to how similar a document's content is to a given query. Specifics of this technique and the ranking process is further given in Appendix A. In order to use this text-based document comparison method with images, however, the notion of words is replaced by that of visual words or clustered image features.

Ideally, features are distinguishing regions of content (or collections of pixels) within an image such as corners, edges, or holes (Figure 1.3). Using a feature detection and descriptor algorithm, such as SIFT (scale invariant feature tracking), these features can be represented numerically, such as by a 128-dimensional vector or feature descriptor. SIFT is a valuable feature detection algorithm for its ability to identify features that are invariant to changes in scale, rotation, and modest amounts of warping between images. For further details on SIFT we defer to work by Lowe [7].

In a set of images, a visual word is simply a common feature, one that potentially exists across multiple images, or even multiple times within a single image and is found by clustering features from all images (Figure 1.4). [8] makes this process efficient by clustering features through an iterative, hierarchical process into a tree structure. The leaf nodes of this structure represents a dictionary of visual words, or a "Bag of Words", that represent the original set of images. Now, in the same manner as text-based document retrieval, the dictionary becomes a basis whereby images can be compared according to their use of common visual words from the dictionary.

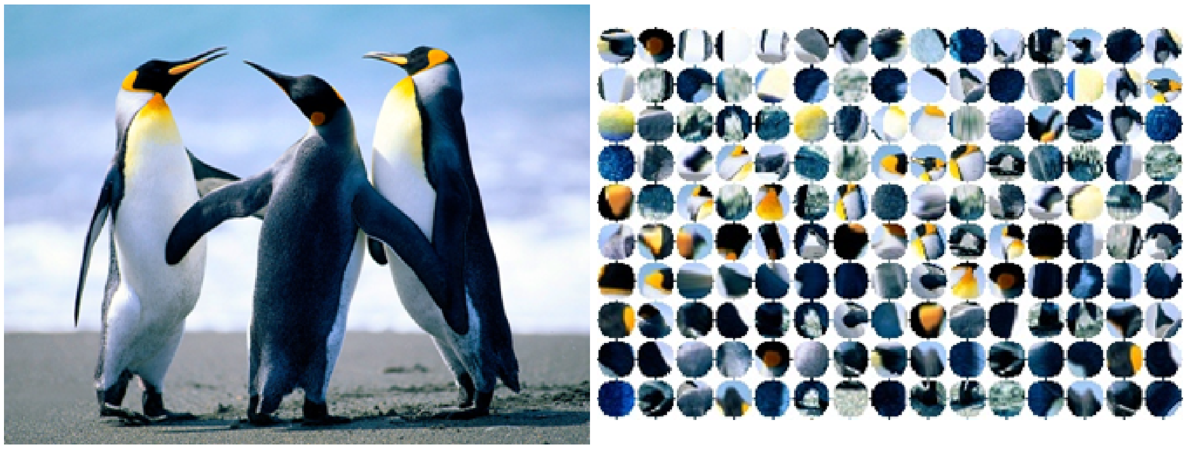

Figure 1.3: Example scale-invariant features extracted from image

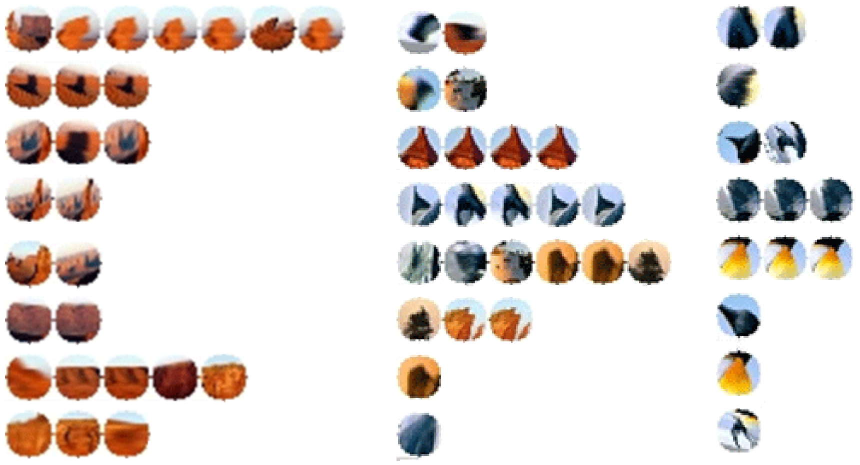

Figure 1.4: Visual words (clustered features) from multiple images

1.3.2 Alignment

Given two independently ordered image sequences, for the purposes of alignment, we designate the source with a higher framerate as the primary source and the other as the secondary source. The purpose of alignment is to find a mapping from each image of the secondary source to images from the primary source with respect to image similarity while maintaining the original ordering of each set. To do so, we represent all possible image pairs as nodes within a graph and impose temporal constraints via the links between nodes. Each link is assigned a cost according to the associated image pair's similarity. Thus, our method uses dynamic programming to find a least cost path through this graph, one that minimizes the accumulated pairwise cost according to the imposed temporal constraints.

Further details and problem formulation for these methods are given in Chapter 2, a self-contained research paper submitted for publication to the IEEE Workshop for Applications of Computer Vision, August 2012. This paper includes an evaluation of the robustness of our methods to changes in noise, capture rates, field of view, and resolution via experiments on both synthetic and real-world data.

Chapter 2

Research Paper

Submitted to *IEEE Workshop for Applications of Computer Vision*, August 2012.

Abstract

Many applications use multiple cameras to simultaneously capture imagery of a scene from a rigid, moving camera system over time. Multiple cameras often provide unique viewing angles but also additional levels of detail of a scene at different spatio-temporal resolutions. However, in order to benefit from this added information the sources must often be temporally aligned. As a result of cost and physical limitations it is often impractical to synchronize these sources via an external clock device. Most methods attempt synchronization through the recovery of a constant scale factor and offset with respect to time. This limits the generality of such solutions. We present an unsupervised method that utilizes a content-based clustering mechanism in order to temporally align multiple non-synchronized image sequences of different and varying spatio-temporal resolutions. We show that the use of temporal constraints and dynamic programming adds robustness to changes in capture rates, field of view, and resolution.

2.1 Introduction

Multi-camera configurations are used in many applications today such as stereoscopic processing of multiple sequences, dynamic scene reconstruction [21], video surveillance, image mosaicking, and independent recordings of historical events. In such configurations it is common that identified content from one source needs to be located in the alternate source as well. However, in order to identify such correspondences there needs to exist a temporal ordering or mapping between the image sequences (Figure 2.1).

In particular, during Wilderness Search and Rescue aided by unmanned aerial vehicles (UAV), the acquisition and streaming of video sequences provides critical imagery for ground searchers to be able to quickly identify areas of interest for directing further search. However, in order to transmit video using equipment that can be carried by a relatively small UAV, it is necessary that the video sequences be of low resolution. In order to provide high-quality visual information of the surrounding view, the UAV is mounted with a camera that supports low-resolution high-frame-rate video with intermittent high-resolution imagery. Due to hardware limitations only the low-resolution data is transmitted wirelessly back to the ground crew, and the high-resolution low-frame-rate imagery is stored locally for offline search. After the flight, searchers must be able to identify a mapping between the live video sequence and the offline high resolution images. By determining a mapping between these sequences, identified content of interest within one sequence can be easily located within the aligned sequence. To this end, we present a solution to the image sequence alignment problem that provides a mapping between two independently ordered image sequences. We also show that our method is robust to variations in capture rates, field of view, and resolution.

Solutions to the video-alignment problem typically fall into two groups: feature-based [1, 3, 6, 9–11, 13, 15, 16, 18, 19, 21], and direct [2, 4, 14, 17]. Direct methods rely on comparing images as a whole (such as in sequence-to-sequence alignment involving the optimization of inter-image sum of squared differences [2]) and as a result struggle with changes in lighting, scale, and/or contrast between frames. Feature-based alignment addresses

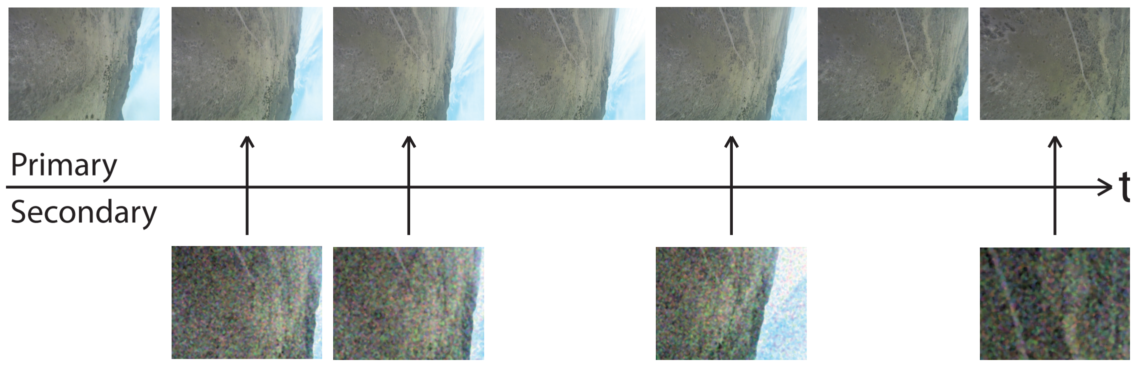

Figure 2.1: Temporal mapping between two image sources. The primary source is high-framerate, low-resolution, while the secondary source is low-framerate, high-resolution.

this problem by focusing on visual features and how they change spatially and/or over multiple frames. An example of such is the use of the path of a feature across multiple frames, its trajectory, to align different sequences [3, 10, 13]. Direct methods tend to perform best when intensities and lighting are similar between images, while feature-based approaches generally work best when there is large variance between images as a result of a greater view baseline [9]. Current solutions in both categories make several assumptions that prevent the realization of a more general solution.

Most solutions assume that there is a constant time delay between sources [2–4, 10, 15–19]. This allows for the recovery of a time offset and scale factor, which can then be used for quickly finding inter-sequence correspondences. The problem with this, however, is that some sources have a variable frame rate, or intermittent footage (such as in a sequence of image still shots, or a system that drops frames to maintain playback quality), which causes alignment calculations that depend on constant frame-rates to be inaccurate.

Other solutions require that the subject is moving and that the camera rig is stationary or rigid in order to accurately determine the path of a point or feature of interest across several frames, such as in [18]. With such a trajectory defined, the temporal alignment is found by determining which trajectories from each sequence match in time and space. These trajectories allow for a time offset and scale factor to be discovered.

Perhaps most related is work by Fraundorfer [6] that uses a scalable image lookup and comparison structure, proposed by Nister and Stewenius [8], as a means for robotic self-location. With an offline database of images covering an area to be traveled, a view image is queried against the data structure to find the most similar image match, and as a result the associated location. Through self-localization over time, an ordering for captured images is implicitly defined. This method does not however place temporal constraints on the source image set.

Our solution utilizes a hierarchical structure along with scale invariant features (SIFT) [7] as a mechanism for efficiently comparing images as well as for obtaining spatial invariance. In addition to the tree structure we explicitly determine an inter-sequence temporal mapping by imposing a temporal constraint through dynamic programming. Because we choose a temporal ordering that optimizes the image pair similarities, there is no need to be concerned with consistency of frame rate. Thus we present a solution to the video alignment problem that is robust to changes in both spatial and temporal resolution.

2.2 Image Comparison and Retrieval

In order to map two sequences, our method depends on the quantification of image similarities. To do this efficiently, we leverage SIFT features extracted from a training set of images and clustered into a hierarchical image-query scheme as developed by Nister [8]. The combination of these two concepts provides a spatially invariant mechanism for comparing images.

2.2.1 Features

SIFT is a common algorithm for locating and describing distinctive scale invariant visual elements, such as edges or corners, within an image [7]. Features are extracted from the image at different scales, and features that exist in multiple (or all) scale levels remain. Once extracted, these features are represented in 128-dimensional vectors, known as descriptors,

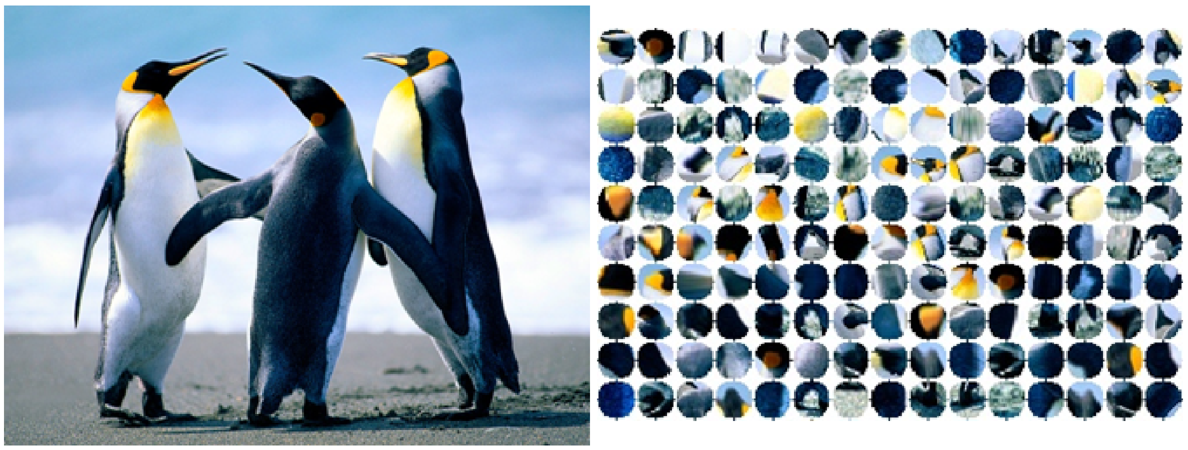
Figure 2.2: Example features extracted from image

according to the features' corresponding gradient orientations, thereby adding rotational invariance.

Another common feature-extraction method includes the identification of maximally stable extremal regions, or MSERs [5], which are located by finding regions of images that do not change size dramatically as the image is thresholded at changing thresholds. This is generally well suited for finding holes or blobs within an image and tends to work at small variations in scales (Figure 2.2). However, with variations in scale, the set of features produced may change due to the existence of smaller (or larger and thus more stable) extremal regions [5]. In order to reduce noise added by this inconsistency across scales, we use SIFT for our feature extraction method because of its invariance to such scale changes.

2.2.2 Image-Based Comparison

Nister's work uses a document retrieval technique known as "bag of words", adapted to images by [12] for efficient image retrieval. Features are extracted from a set of images and then clustered using hierarchical k-means. The set of leaf-clusters from this tree represent a trained dictionary that can be used as a metric for comparing images.

Once the dictionary has been trained, any image can be defined with respect to the dictionary. For a given image, features are extracted and then classified by descending the tree. An n-dimensional vector (where each element corresponds to a leaf cluster in the tree) is then computed according to the frequency with which the features of the current image arrive at each of the leaf nodes. We use these vectors as the basis for image comparison.

With these vectors, image similarity is simply computed via the comparison of these vectors using either the Mahalanobis distance or an L-norm distance metric. [8] shows that the L_1 norm tends to provide more accurate results than the L_2 norm because of its ability to better handle variance. For this reason, we use the L_1 norm for vector distance.

2.3 Content-Based Alignment

With a method for quantifying the similarity between images, the problem becomes one of optimization with respect to the inter-image similarity scores. Initially we assume that there are two sources to be aligned. Of the two, the image sequence with the highest frame rate is designated as the primary source. This set can be thought of as the sequence to which an alternate secondary source is aligned. This source is used as for training the visual dictionary as previously described. Each image in the secondary source is then defined in terms of the dictionary and subsequently compared against the primary set and aligned using dynamic programming.

2.3.1 Problem Formulation

Using the metrics defined in the previous section, let the similarity between two images s and p be the distance between their respective vectors, or $D(s, p)$. Given P and S as the set of images from the primary and secondary sources respectively, we find the mapping M from all images in S to images in P:

$$M : s \to p \mid s \in S, p \in P \tag{2.1}$$

that minimizes the overall image-matching score:

$$C(M) = \sum_{s \in S} D(s, M(s)) \tag{2.2}$$

subject to the following temporal ordering constraint:

$$\forall s \in S : s_i < s_j \Rightarrow M(s_i) \leq M(s_j) \tag{2.3}$$

This temporal monotonicity constraint ensures that the initial frame ordering of each source is maintained across the resultant mapping. Though not necessary for alignment, the monotonicity constraint can be made more restrictive (and implemented more effectively) by incorporating knowledge of how far apart frames of the secondary source are (in time) relative to the primary source:

$$\forall s \in S : 0 \leq M(s_{i+1}) - M(s_i) \leq k \tag{2.4}$$

This allows us to place an upper bound, k, on how many frames to look ahead for matches to a given frame and as a result reduce the search space. Keep in mind that k does not imply knowledge of actual frame rates but rather allows one to place a loose constraint on how far ahead to search.

2.3.2 Creation of Image Vectors

In order to align two image sources, we must first compute the tree-vectors for all images of each image source. These vectors allow us to quantitatively compare two images and ultimately determine the cost associated with each graph link. As described in Section 2.2.2, features are first extracted from the primary image source (Figure 2.2) and are used to train the vocabulary tree described by Nister [8]. Then, we take each image from both sources individually along with their corresponding features, and classify each feature by descending the vocabulary tree. Each element of this vector corresponds to a leaf node in the vocabulary tree and is computed according to the frequency of occurrences of that cluster within the associated image and weighted by the uniqueness of that particular cluster among all images of the training set, also known as the term-frequency inverse document-frequency (TF-IDF) score in natural language processing [12].

2.3.3 Cost Function

With TF-IDF vectors computed for both sources, we can now create the cost function, $D(s, p)$ for all $s \in S$ and $p \in P$, which is visualized in Figure 2.3. Within this image each pixel (x, y) corresponds to a pairing (s, p) of an image from the primary source to one from the secondary source. The brightness of each pixel is proportional to the distance, $D(s, p)$, between the corresponding pair's TF-IDF vectors. The darker the pixel, the more similar the two images of the pair are.

In the example shown in Figure 2.3, the ratio of primary to secondary images is 1:1 and the correct solution lies along the diagonal. $D(s, p)$ demonstrates strong dark regions that follow the diagonal of the image but manifest as a thicker band due to normal frame-to-frame overlap. The uniform banding pattern present in the background is a result of each vector having a different quantity of features. A vector with fewer features has fewer associated clusters, thus reducing its score when compared to other vectors. The low-cost regions off the diagonal correspond to areas where multiple images share similar content such as when the same portion of the scene is observed at different times.

For distance, we use the L_1-norm because of its ability to better handle image variance than the L_2-norm as described by [8] though any multi-dimensional metric, such as the Mahalanobis distance, is certainly viable.

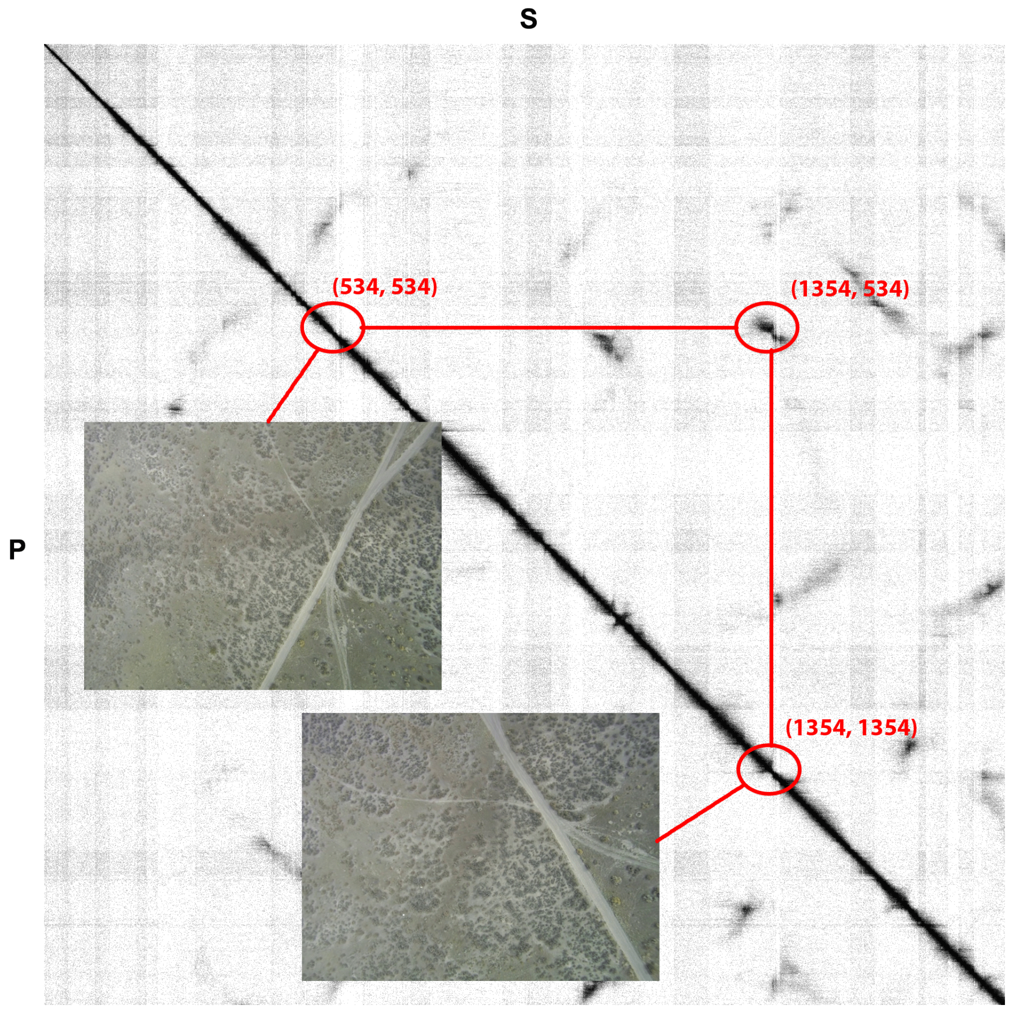

Figure 2.3: Visualization of the cost function $D(s,p)$ between two image sources S and P. Dark pixels correspond to image pairs that have a low cost (are a good match). The correct solution for this simple example lies along the diagonal manifested as a thicker band due to image overlap. Off-diagonal low-cost regions are a result of multiple occurrences of similar image content at different times. The uniform pattern in the background is a result of variance in the number of features between images.

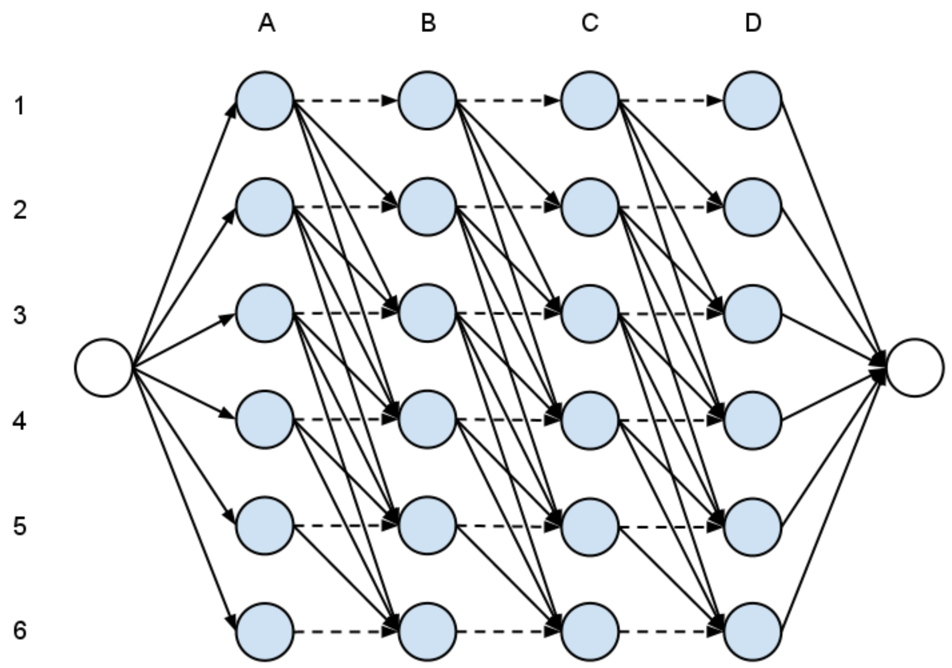

Figure 2.4: Monotonically increasing graph. Constructed with $k = 3$ look ahead, and from ordered sets $P = \{1, 2, 3, 4, 5, 6\}$, $S = \{A, B, C, D\}$.

2.3.4 Alignment through Dynamic Programming

Given the cost function, $D(s, p)$, Equation 2.2 can be trivially minimized by

$$M(s) = \underset{p}{\operatorname{argmin}} D(s, p) \tag{2.5}$$

in order to find a mapping between image sources. This method, which we refer to as "greedy", does not however meet the temporal constraints defined in Equation 2.3.

To implement these constraints we treat the problem as a least-cost path optimization. We construct a graph (Figure 2.4) with nodes that represent a pairing of two images, one from each source, and links arranged according to the defined temporal constraints. Horizontal links are added between nodes so as to allow the mapping of multiple secondary images to a single primary image such as when frames are missing from the primary source. Links leading to a given node are weighted according to the similarity score of the paired images at that node. The solution then is the least-cost path through the constructed graph, which we solve via dynamic programming (DP).

To find the least cost path, we define the minimum accumulated cost, q, of the pairing of a primary image, p, to a secondary image, s, as the cost, $D(s,p)$, of the pair plus the minimum accumulated cost of that pair's possible predecessor nodes. By accumulating the cost values we are able to determine the least-cost path leading to each node. This can be visualized as the construction of a topological surface in which to find a valley that connects the start and end nodes of the graph (Figure 2.5). The number of possible predecessor nodes, t, is limited by the temporal constraint, k, defined in Equation 2.4. Formulated as a typical dynamic programming problem,

$$q(s,p) = \begin{cases} \infty & \text{if } p < 1 \\ D(s,p) & \text{if } s = 1 \\ D(s,p) + \min_{t \in T}\{q(s-1, p-t)\} & \text{otherwise} \end{cases} \tag{2.6}$$

where

$$T = \{0, ..., k\} \tag{2.7}$$

Note that when $t = 0$, there is a small penalty added so as to discourage always following horizontal links and choosing a trivial low-cost solution, such as when many secondary images map to a single primary image in the absence of distinguishing features.

Once $q(s,p)$ has been computed for all pairs, the minimum cost path is found by selecting the pair, (s,p), with the lowest accumulated score that corresponds with the most recent secondary image (right-most column in Figure 2.4).

2.3.5 Complexity

Initially, the most general graph structure with H primary frames and L secondary frames has a DP solution complexity of $O(H^2L)$. By imposing monotonicity with forward or lateral

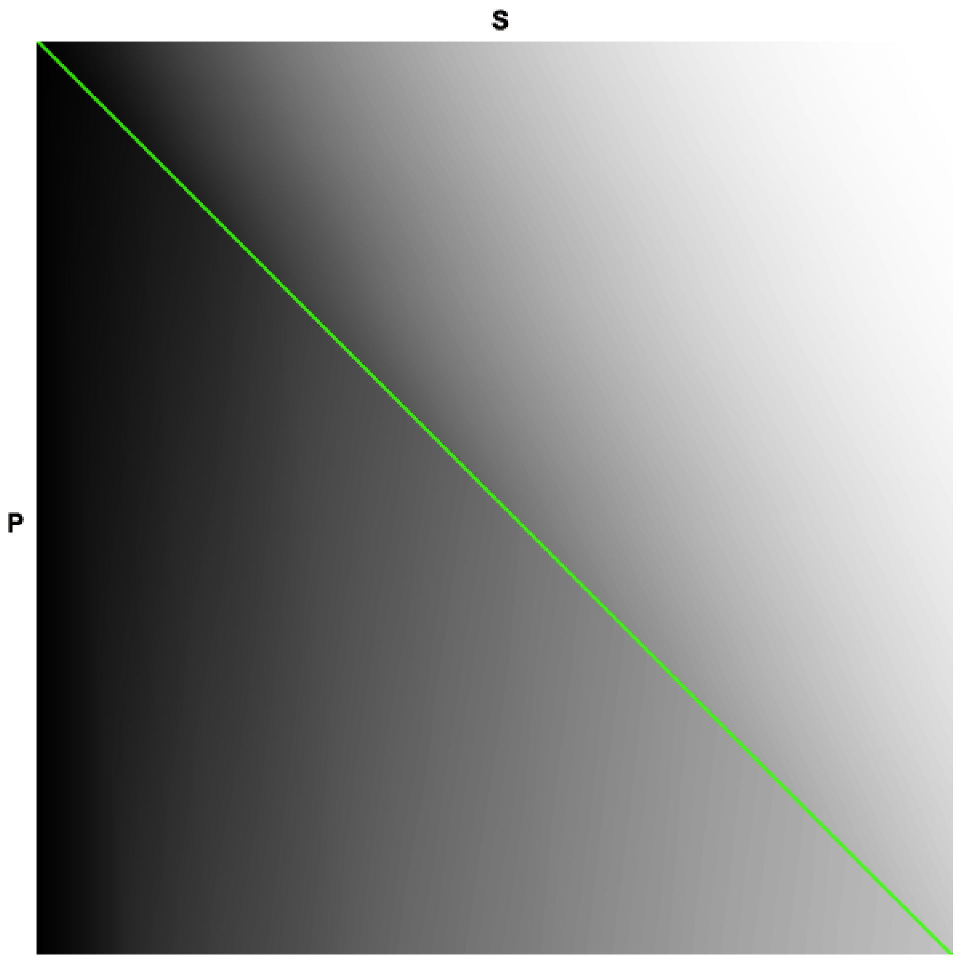

Figure 2.5: Visualization of the accumulated cost function, $q(s, p)$, from a graph with a k-lookahead of 3. Each pixel corresponds to a node in the constructed graph (Figure 2.4), the brightness of each is determined by the accumulate cost of the least-cost-path to the associated node. The least-cost-path is highlighted in green.

links, as well as a loose limit, k, on the number of frames to look ahead for matching, each node can only connect to at most k other nodes (Figure 2.4). Thus, the complexity is reduced to $O(HLk)$.

This still has a high-order complexity which can be reduced through a coarse-to-fine strategy. Because the primary source contains a denser set of frames, there is greater potential for frame overlap. As a result, we can create a subset of frames from the primary source such that the set is reduced in size but that the spatial overlap is maintained. Overlap is important so as to provide enough information for accurate alignment with the secondary source. This reduction factor r not only decreases the size of the primary source to $\frac{H}{r}$, but the frame look-ahead to $\frac{k}{r}$ as well, giving a new complexity of $O(\frac{HLk}{r^2})$. There is a cost associated with refinement after reduction occurs, $O(Lr^2)$, however it has lower complexity than the initial reduction, so it does not increase the overall complexity.

Furthermore, by recognizing that k is limited by how much larger H is than L, k can be considered to be proportional to the ratio of H to L:

$$k \propto \frac{H}{L} \tag{2.8}$$

As mentioned before, the reduction factor r is dependent on how dense the primary source is, so it is proportional to the frame rate f_p of the primary source:

$$r \propto f_p \tag{2.9}$$

Finally, H is a direct function of the frame rate of the primary source and its total duration, t:

$$H = f_p t \tag{2.10}$$

These final observations allow us to generalize our problem complexity to $O(t^2)$, which shows that complexity increases only with the total video duration, not its frame rate or that of the secondary source.

2.4 Evaluation

In this section we quantitatively evaluate the performance of the methods proposed in Sections 2.2 & 2.3. In order to do so effectively we use synthetic datasets, which allows us to more precisely determine ground-truth alignments for accuracy measurements as well as have more control over each dataset's parameters and allow more in-depth analysis. To address biases present in synthetic data, we also evaluate the accuracy of our methods on non-synthetic data.

Each of the synthetic datasets we analyze are generated from 1800 (approximately one minute) 1280×960 colored images extracted from aerial videos. To synthesize a primary set with low resolution and a reduced field of view, images from the original source are down-sampled by 40% and cropped to 640×480. Secondary images are left unscaled and are extracted from the original video using a Poisson distribution with a mean according to the needed primary-to-secondary image ratio. Unless specified, the primary-to-secondary ratio is roughly 8:1. We also corrupt the images from both datasets with Gaussian noise in order to reduce benefits of generating both from the same video source. We do not, however, explore changes in rotation or perspective between sources since SIFT is known to be rotationally invariant and robust to modest variations in warping [7].

Once primary and secondary sets are generated, the accuracy of a computed mapping is based on the number of image-pair mismatches with respect to ground truth data. In practice, however, due to differences in fields of view and overlapping content there may be ambiguous alignments such as when secondary images are of a higher resolution and share content from multiple primary images. Thus, for the purposes of this application, it is more useful to define accuracy as correct to within some number of frames. For our experiments, unless specified otherwise, a mapping is correct if it is to within 5 frames of ground truth. We further motivate this later by looking at the root-mean-square (RMS) of the disparity of chosen and ground truth primary image indices.

Additionally, to prevent DP from choosing trivial solutions (Section 2.3.4), we add a small cost of 10^{-4} to horizontal links during dynamic programming which we have empirically found to work well.

From initial observations of a cost function for a given dataset (Figure 2.3), it is apparent that the greedy method will work quite well in simple situations (such as with minimal noise). We show, however, that because of the lack of temporal constraints the greedy method is not robust to situations such as noisy images, lack of features, or locally ambiguous options. In each experiment we show the results of our dynamic programming and greedy methods, using greedy as a baseline for comparison.

2.4.1 Synthetic Data Limitations

The generation of the primary source in the described manner results in features from the primary source being a subset of the secondary source. To address this bias we ran experiments where we removed each image from the primary set that has an associated secondary match.

On the base-set previously mentioned, with removal the accuracy of DP was 99.481% with greedy at 97.403%, compared to 99.481% and 99.221% respectively without removal. Such similar results did not motivate further addressing this bias.

2.4.2 Affect of Noise on Accuracy

Despite the performance of the greedy approach in situations of low noise, lack of noise is not realistic to real-world data. As noise increases within images, the optimal solution with respect to image similarity becomes less pronounced (Figure 2.6) and the accuracy of both

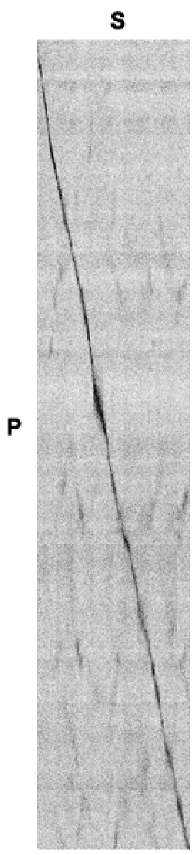

Figure 2.6: Visualization of cost function for a noisy dataset with a primary to secondary image ratio of 8:1. Datasets were generated using Gaussian noise with $\sigma = 16$ for the primary set and $\sigma = 64$ for the secondary set.

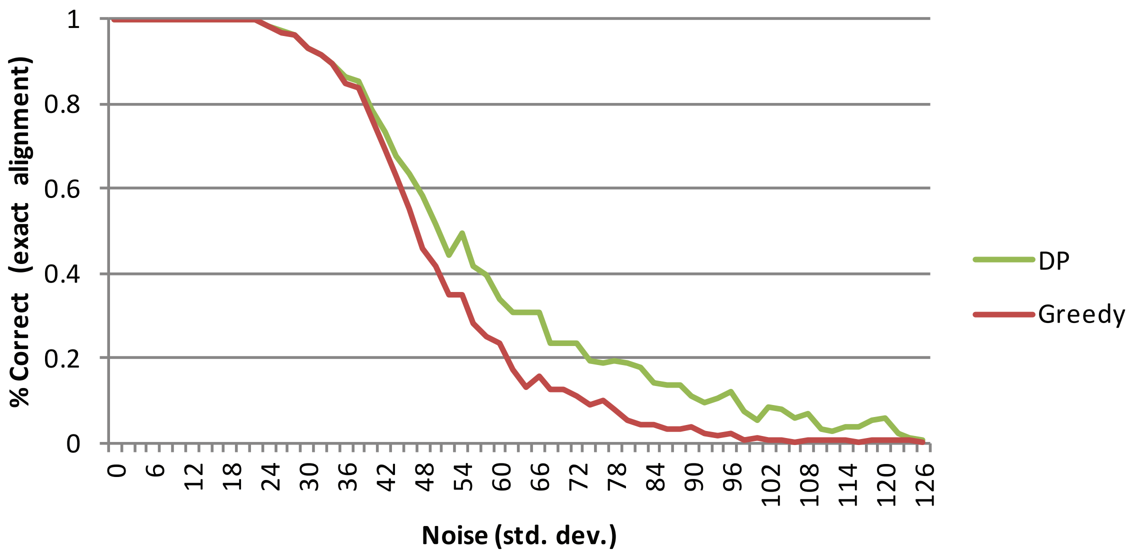

Figure 2.7: Accuracy of DP and greedy methods as image-noise variance increases. Primary to secondary image ratio is 8:1 with exact matches.

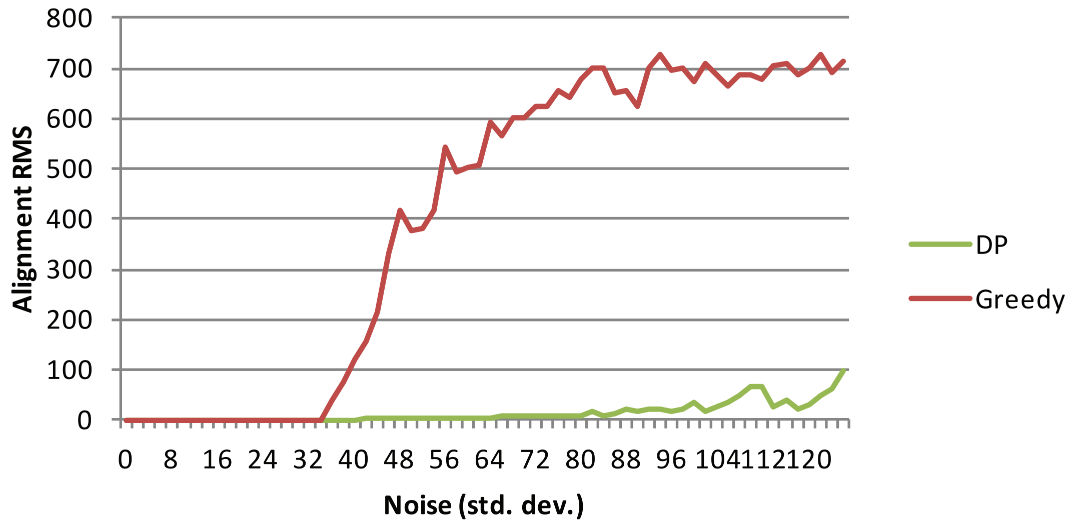

Figure 2.8: RMS alignment error of DP and greedy methods as image-noise increases.

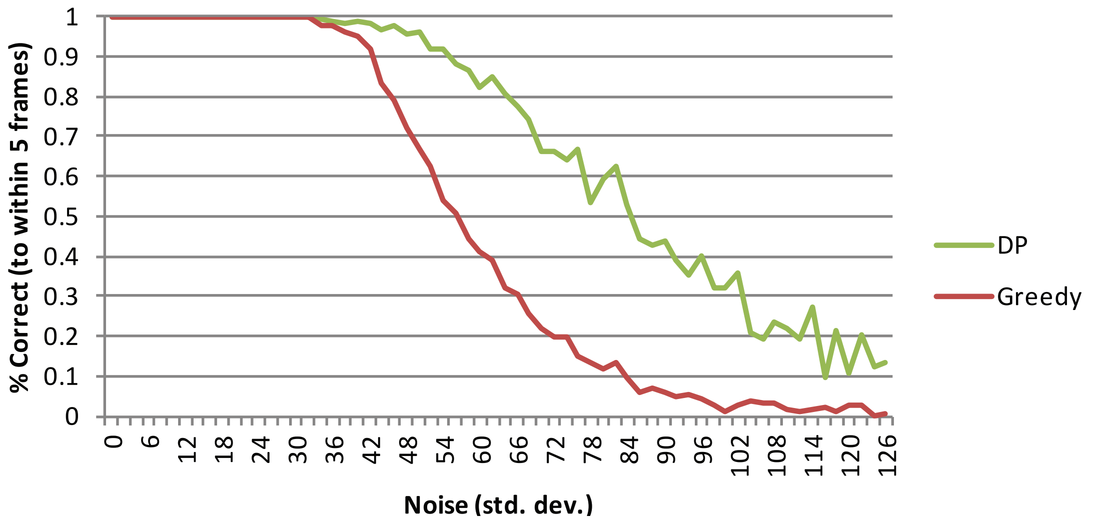

Figure 2.9: Accuracy of DP and greedy methods to within 5 frames.

methods decreases (Figure 2.7). Despite this decrease in accuracy, dynamic programming has a slower rate of decay because of temporal constraints.

Although the accuracy of each solution decays with noise, the RMS alignment error of each differs drastically (Figure 2.8). Because of this disparity in RMS error, and, as discussed previously, due to the fact that most alignment applications benefit from finding frames within a certain proximity to the exact match, we can mark pairs that are within five frames of ground truth to be correct. In doing so, the average accuracy of DP increases while that of greedy remains relatively unaffected (Figure 2.9).

2.4.3 Primary to Secondary Ratio

In order to show that our method is robust to differences in frame rates between sources, it is important to show how accuracy changes with an increase of the ratio of primary to secondary images. This is done by gradually increasing the mean of our Poisson sampling distribution. Not only does this increase the mean ratio of primary to secondary images, but the variance of sampling as well ($\sigma^2 = \mu$).

Our results show that although there is a slight decrease in accuracy, our method remains robust to increases in the ratio of primary to secondary images up to 60 primary frames for every secondary frame, or frame rates of 30 fps and 0.5 fps respectively (Figure 2.10).

2.4.4 Scale and Field of View Changes

It is not uncommon in a multi-camera configuration for there to be differences in scale and field of view between the cameras (Figure 2.11). Here we examine the affect of these differences on accuracy.

To simulate scale differences we gradually down-sample the primary source without cropping (e.g. down-sample by 30%). Field of view is simulated by simply cropping the primary images by the same scale factors (e.g. crop by 30%). We only affect the primary

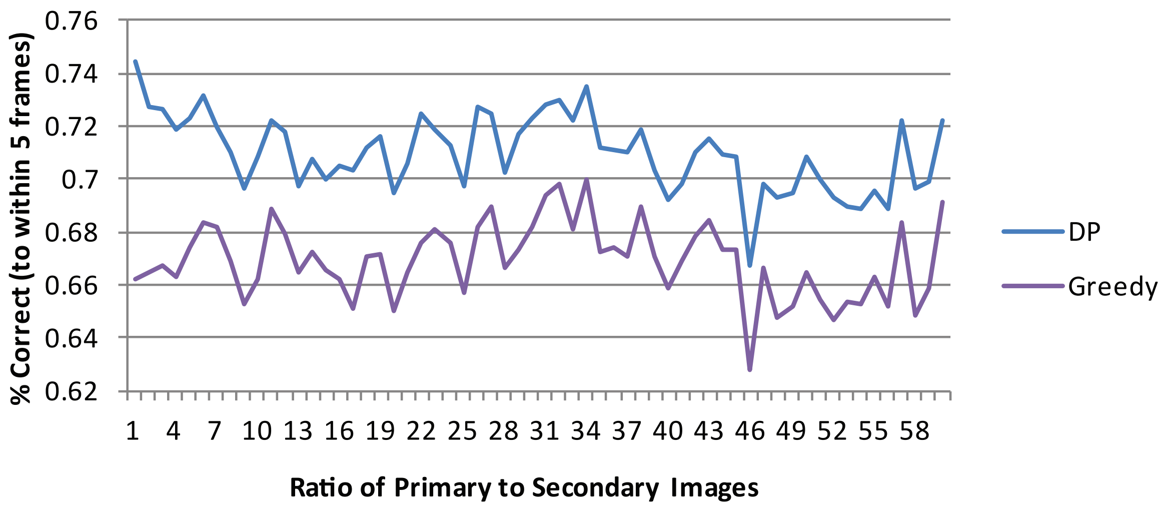

Figure 2.10: Accuracy vs. change in ratio of primary to secondary images. Datasets were generated using normal noise with $\sigma = 64$ and $k = 350$.

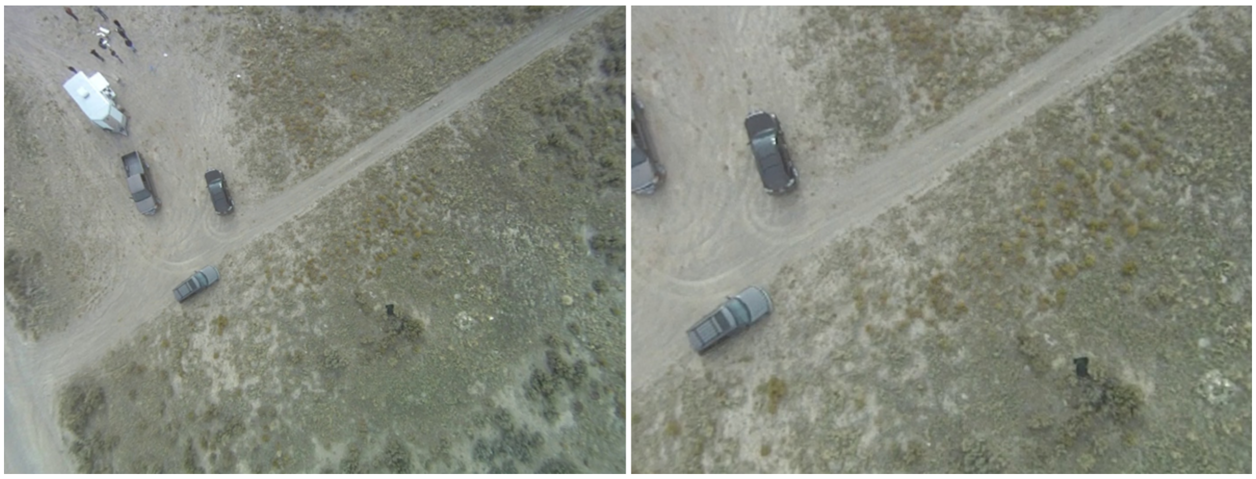

Figure 2.11: Example side-by-side of different fields of view.

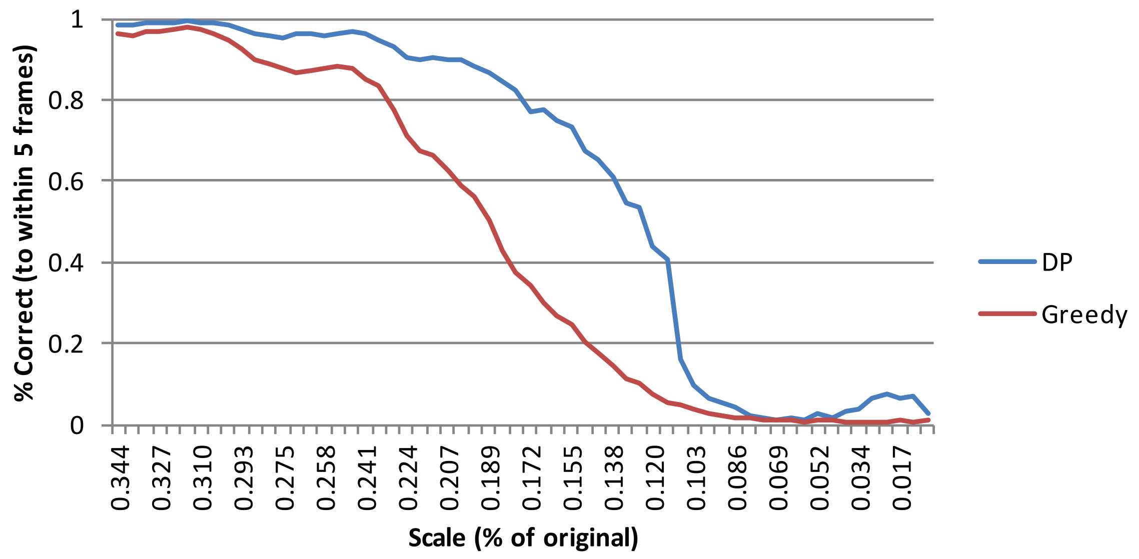

Figure 2.12: Accuracy of DP and greedy methods as the primary source scale decreases. Data set images were corrupted with Gaussian noise with $\sigma = 16$.

source in either of these changes since our initial assumption is that the secondary source is of a higher image resolution.

As images are reduced in size (whether by scale or by a change in field of view), there is an inherent reduction in the quantity or quality of features. This reduction yields falsely-similar, or low, comparison scores uniformly against all other images due to a lack of distinguishing features in general, thus diminishing accuracy. Beyond a particular downsample, however, scale actually changes features directly (in addition to reducing the feature set) while cropping the image only reduces the set. This explains the non-linear linear drop-off in accuracy due to changes in scale (Figure 2.12) and more gradual decay in accuracy with changes in field-of-view (Figure 2.13).

Without sufficient quantity or quality of features, DP may choose a trivial solution and underperform a greedy solution. However, in this situation both methods already suffer in accuracy due to a lack of features.

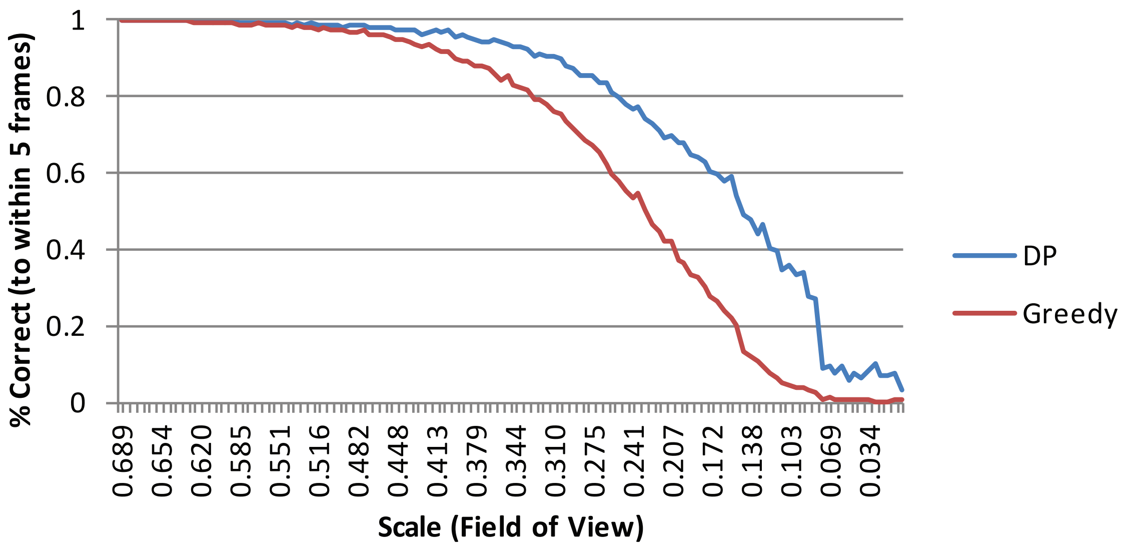

Figure 2.13: Accuracy of DP and greedy methods as the primary source field of view decreases. Data set images were corrupted with Gaussian noise with $\sigma = 16$.

2.4.5 Locally Ambiguous Solutions and Accuracy

Alignment ambiguity (and the need for temporal constraints) becomes most apparent when the same portion of the scene is viewed multiple times. We experiment with two scenarios that produce such ambiguities: loops (Figure 2.14), and sweeping patterns (Figure 2.15).

A loop attempts to simulate when the video covers content multiple times. A sweep, also known as a lawn-mower-pattern, is a means for uniform coverage of a particular area with overlapping sweeps. In either case, as the number of ambiguous options increases, the accuracy of the greedy solution is negatively affected in a linear fashion, while DP is relatively unaffected (Figure 2.16).

2.4.6 Performance

Each of the presented tests ran on a machine with an Intel i7 2.8 GHz quad-core, 24 GB of RAM, NVIDIA GeForce GTX 480 GPU, and solid state drive.

All computations were performed in memory and then subsequently written to disk. To do this we statically pre-allocate memory according to estimated needs. Due to the lack of memory management or disk I/O overhead this empirically improved runtime performance.

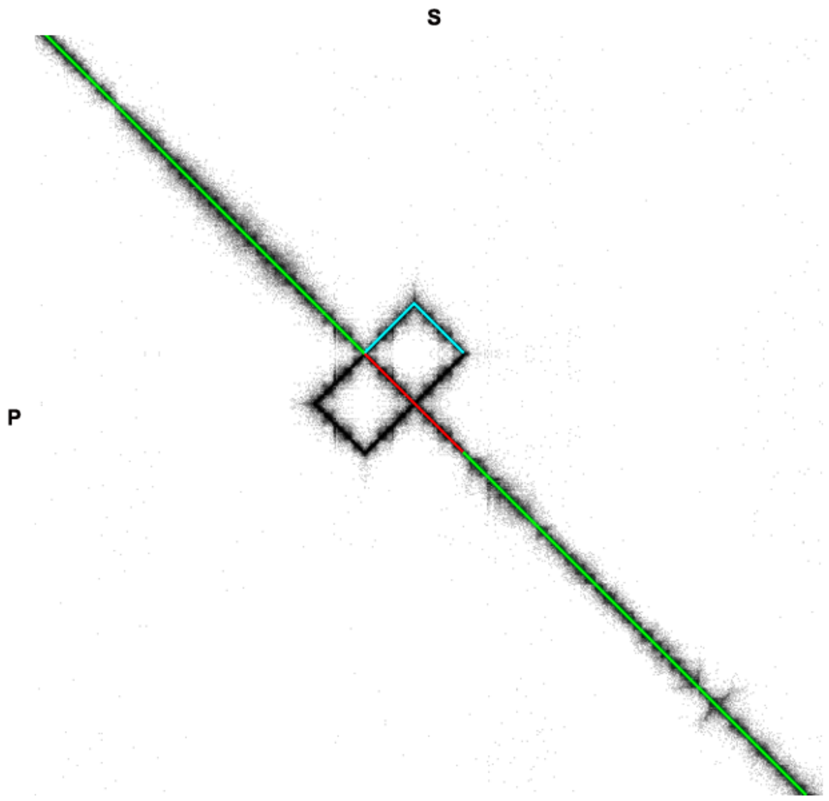

Figure 2.14: Example cost function with an 80 frame loop overlayed with DP path (red), greedy path (cyan), and both (green).

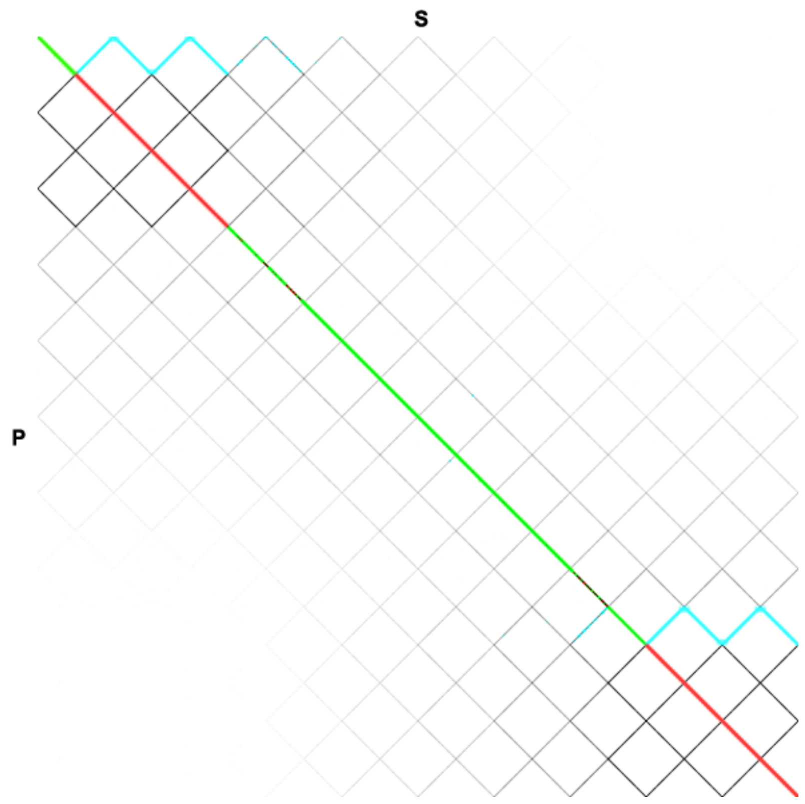

Figure 2.15: Example cost function with 20 overlapping sweeps overlayed with DP (red) and greedy (cyan) paths

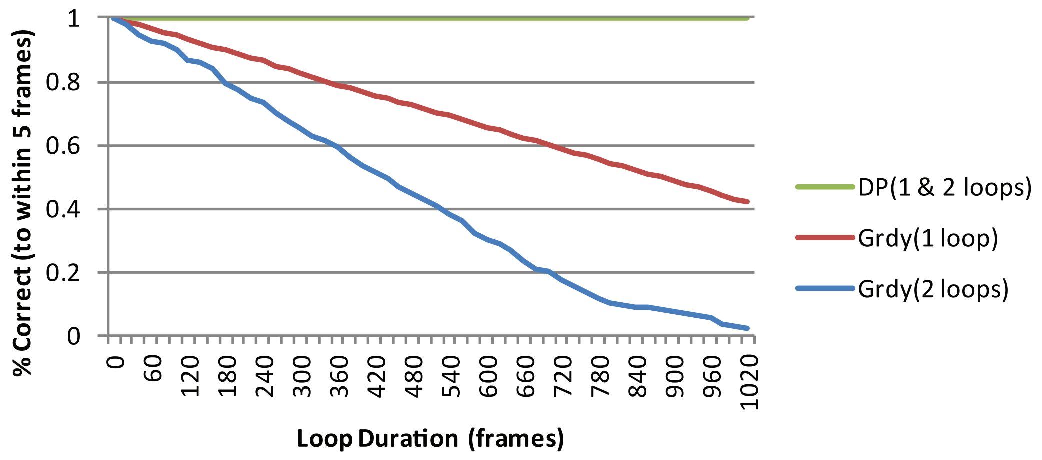

Figure 2.16: Accuracy of DP and greedy as loop duration increases. DP and Greedy are each shown with both one and two loops. Using k-lookahead of 3, primary to secondary ratio of 1:1, and minimal noise.

This also increases total memory consumption (around 6 GB for 3600, 640×480 images with approx. 3000 features per image). A more dynamic memory scheme may be used in order to achieve a smaller memory footprint.

The extraction of features from the primary and secondary sources are independent of each other. Thus, to save time, while the vocabulary tree is being from the primary source, we simultaneously extract features from the secondary. This gives a 23% speedup of the entire process on average.

Additionally, we implement various components of our method on the GPU. We use Wu's ([20]) SIFT implementation for the GPU and are able to achieve between 15–20 fps for feature extraction. Clustering is the most time consuming portion of our method running at 9,300 features per second, or 8 fps. The computation of TF-IDF vectors is usually at around 22 fps. The cost function computation and DP alignment takes about 11 seconds for 3600 $1 \times 7k$ vectors (totaling around 24M elements across all vectors). Finally, we run vector alignment on the GPU using OpenCL.

We can further improve run times (and memory performance) by reducing the number of features per image without much affect on accuracy. The reduction of features impacts the amount of data that needs to be clustered and, subsequently, reduces vector sizes. Experiments show that we can reduce from 3000 to between 100 and 200 features per image and still achieve comparable accuracy (Figures 2.17).

For small data sets, run time appears to achieve linear time-complexity with the number of images (Figure 2.18). This is because the $O(n^2)$ complexity of alignment is dwarfed by those of feature extraction and clustering for small sets. We would expect however that as the data set grows the alignment run time should eventually bypass and exceed the run-time of other components. If we isolate only the alignment component, we can see this quadratic behavior (Figure 2.19).

To reduce the run-time complexity of alignment, as described in Section 2.3.5, we can reduce the primary set by a particular reduction factor and perform a rough alignment. We

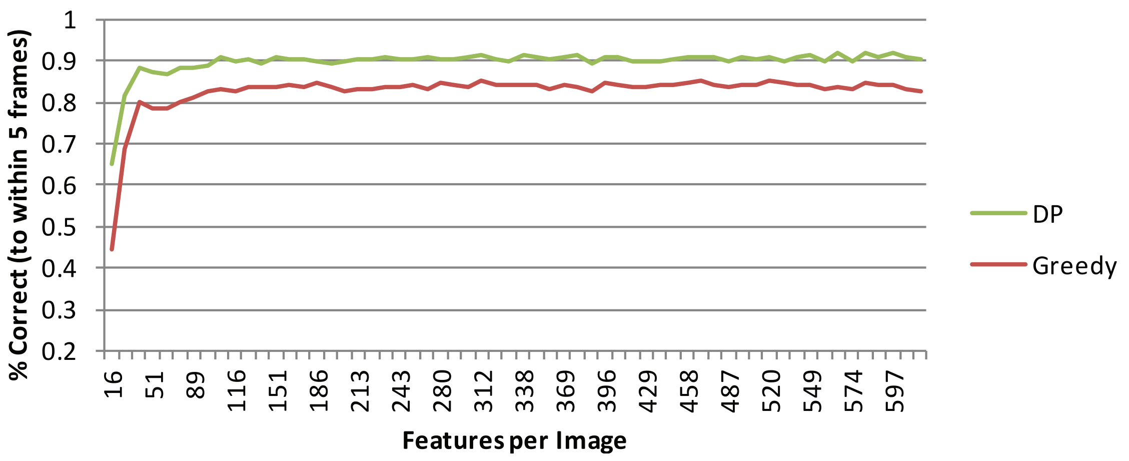

Figure 2.17: Accuracy increases with the number of features per image. Images corrupted with noise of $\sigma = 32$.

refine the initial alignment by aligning all neighboring primary images not included in the original alignment phase to local secondary images. In doing so, despite overhead perceptible in smaller data sets, the quadratic run-time complexity is delayed.

In order to determine the practicality of our method on real-world scenarios, we ran our method on larger data sets. With 5 minutes worth of video, and 2500 features per image, total run time was about 50 minutes. However, by reducing the features to approximately 150 features per image, and using a reduction factor of 5, total run-time was reduced to 12 minutes with comparable accuracy (93%). Memory consumption was likewise reduced from 9 GB to 800 MB. Similar results were achieved with a 15 minute dataset at 90% accuracy and a run time of 30 minutes (with roughly 150 features per image).

2.4.7 Real-World Data

In this experiment, we run our methods on non-synthetic data acquired by an HD GoPro Hero mounted to the bottom of an unmanned aerial vehicle (UAV).

The GoPro video feed was recorded to a local SD card for offline use as well as simultaneously transmitted live to a ground unit where the feed was also recorded. Offline images were captured at 30 fps and stored at a resolution of 1280×960 while the live signal

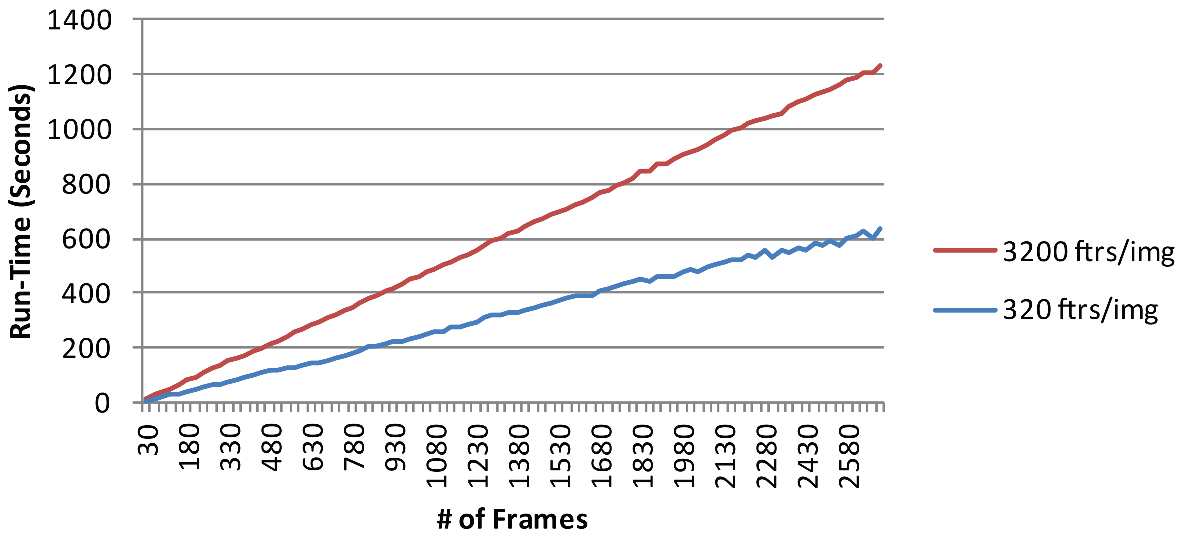

Figure 2.18: Overall run-time complexity growth with the number of images. Because of the data-set size, alignment run-time is dwarfed by the run-times of feature extraction and clustering. Thus, the overall run-times appear linear.

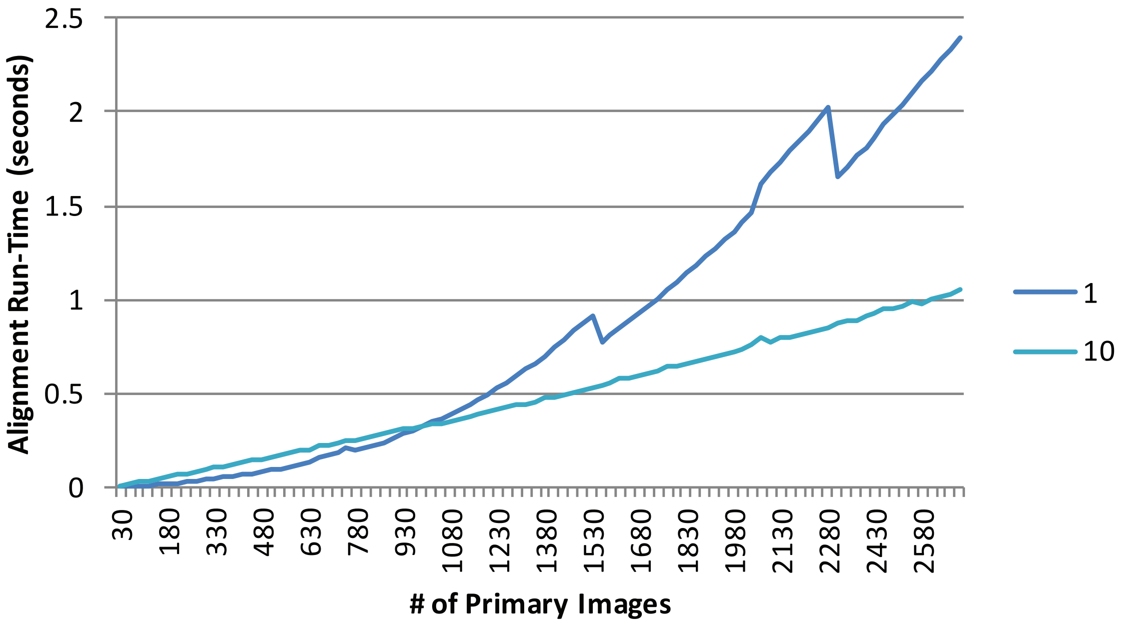

Figure 2.19: Alignment run-time complexity growth with the number of images using reduction factors of 1 and 10. As reduction factor increases the quadratic behavior of alignment run-time is delayed.

was recorded at 15 fps (interlaced) at 640×480. Both sources cover approximately 40 seconds of video, which yields 1300 and 595 frames between the offline and live sources respectively.

Due to the higher frame rate of the offline GoPro data, we designated the offline data as the primary source and the streamed transmission as the secondary source. Also, for quantitative analysis, we created a ground truth alignment between the sources by manually selecting the appropriate mapping from all secondary images to images from the primary source. Additionally, for alignment we used a k-look-ahead of 14, and a horizontal penalty of 10^{-5}.

With these parameters, the use of temporal constraints via DP resulted in 522 of 595 exact matches (with 100% to within 5 frames) and an RMS alignment error of 1.12, while a greedy approach matched 492 of 595 pairs (with 97.5% to within 5 frames) and an error of 89.20. A visualization of the associated cost function is shown in Figure 2.20.

2.5 Conclusion

We have presented a method for the temporal alignment of two image sequences that is robust to changes in imaging resolution, frame rate, scale, and field of view. Our solution is able to achieve spatial invariance through the utilization of a hierarchical SIFT-based feature clustering mechanism. The imposition of temporal constraints through dynamic programming provided a means for robustness to ambiguous alignment solutions created by image noise, high occurrence of content overlap, or differences in scale between sources.

We have also shown that the quantity of features can be reduced in order to achieve higher run-time and memory performance and still maintain comparable accuracy. Additionally, through a reduction and refinement process the computational requirements of our method can be reduced even further. Through these optimizations larger video segments can be aligned within a practical amount of time.

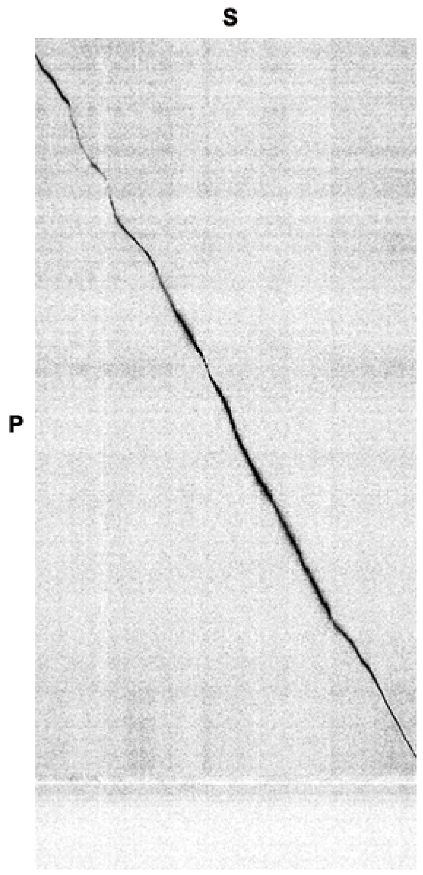

Figure 2.20: Visualization of cost-function of real-world data generated from HD GoPro Hero mounted to unmanned aerial vehicle.

2.6 Acknowledgments

This work was partially supported by the National Science Foundation under grant numbers 0534736 and 0812653. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Chapter 3

Conclusion and Future Work

We have demonstrated that through the imposition of temporal constraints, the accuracy of our method is robust to variations in spatial resolution, frame rate, field of view, and scale. Additionally, through the use of SIFT features, scale and rotational invariance is achieved in the presence of modest levels of noise.

We have also shown that higher run-time and memory performance can be achieved with little to no affect on accuracy by reducing the quantity of features extracted from each image. The computational requirements of our method can be further reduced through a reduction and refinement process. As a result of these improvements, it is possible to align larger video segments within a practical amount of time.

Although we presented a method for finding a direct temporal mapping between frames, further refinement is necessary in order to achieve a total ordering. One such refinement can be to search local neighbors of a primary image to find the optical flow between frames at the point of alignment. Using a computed homography that maps the secondary image to the primary image, where the secondary image lies relative to the center of the primary image with respect to the optical flow should indicate whether the secondary image comes before or after the primary image in time.

Additionally, though our method was able to improve upon the complexity of dynamic programming through an initial alignment with a reduced primary set, performance could be further improved through a reduction of the secondary source as well.

During the refinement phase of the reduction process, each sub-alignment is independent of other sub-alignments. Thus, a distributed solution could be considered in order to allow the processing of even larger datasets.

Appendix A

Text-Based Document Comparison

To begin, a dictionary of words is created from all documents of interest, called a "bag of words." Each word in the dictionary is assigned a weight, per document, known as the term-frequency inverse-document-frequency (TF-IDF) score, according to the word's commonality and uniqueness across the set of documents. The TF-IDF score for a single word, i, and document, d, given N documents, is given by:

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{N_i} \tag{A.1}$$

The first component, $\frac{n_{id}}{n_d}$, is the term-frequency weighting portion. Here, given the frequency that word i appears in document d, n_{id}, and the total number of words in document d, n_d, this simply gives weight to a particular word by how frequently it appears within a single document.

The second component, $\log \frac{N}{N_i}$, is the inverse-document-frequency. Given the total number of documents within the data set, N, and the total number of documents that contain word i, N_i, this portion diminishes the TF-IDF score as the number of documents that contain a particular word increases. As a result, this term gives more weight to words that appear less frequently in the entire data set.

Combining the two terms, the TF-IDF score, therefore, favors words that are unique among all documents while fairly prevalent within a subset of the documents. In contrast, words that occur a few times in a single document are given less weight ($\frac{\epsilon}{\alpha} \approx 0$ when $\epsilon \lll \alpha$) as well as those that occur in all documents ($\log 1 = 0$).

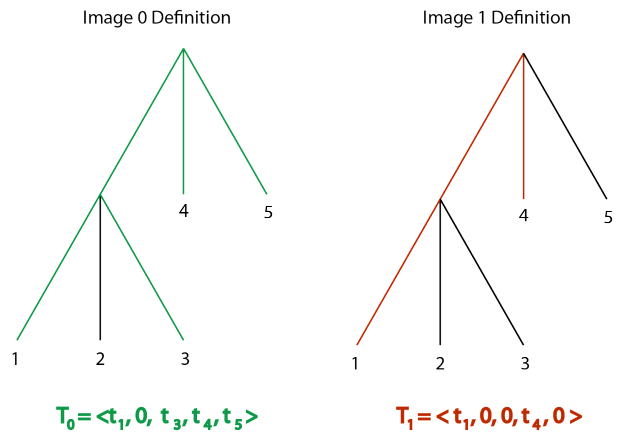

Figure A.1: Two TF-IDF vectors defined by visual descriptors in dictionary (tree)

A vector of TF-IDF scores, $T_d = (t_1, t_2, ..., t_n)$, is created for each document based on how many of each word a given document uses from the dictionary (Figure A.1). Finally, the content similarity between two documents is quantified by finding the distance between their respective TF-IDF vectors.

A common method for comparing two vectors is the Mahalanobis distance, which utilizes the inverse covariance matrix for normalization as used by [12]. Although other L-norm distance metrics can easily be substituted and have different impacts on accuracy, [8] shows that the L_1 norm tends to provide more accurate results than the L_2 norm because of its ability to better handle variance. Thus we use the L_1 norm for distance in our method.

References

[1] D. Brito, F. Padua, G. Pereira, and R. Carceroni. Temporal synchronization of non-overlapping videos using known object motion. *Pattern Recognition Letters*, 32(1):38–46, 2011.

[2] Y. Caspi and M. Irani. A step towards sequence-to-sequence alignment. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 682–689, 2000.

[3] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. *International Journal of Computer Vision*, 68:53–64, 2006.

[4] C. Dai, Y. Zheng, and X. Li. Accurate video alignment using phase correlation. *IEEE Signal Processing Letters*, 13(12):737–740, December 2006.

[5] P.-E. Forssen and D.G. Lowe. Shape descriptors for maximally stable extremal regions. In *Proceedings of the IEEE 11th International Conference on Computer Vision*, pages 1–8, October 2007.

[6] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, November 2007.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[8] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[9] F. Padua, R. Carceroni, G. Santos, and K. Kutulakos. Linear sequence-to-sequence alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:304–320, February 2010.

[10] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 939–945, October 2003.

[11] P. Sand and S. Teller. Video matching. *ACM Transactions on Graphics*, 23:592–599, August 2004.

[12] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, April 2009.

[13] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 1521, 1999.

[14] Z. Szlávik, T. Szirányi, and L. Havasi. Video camera registration using accumulated co-motion maps. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(5):298–306, 2007.

[15] P. Tresadern and I. Reid. Synchronizing image sequences of non-rigid objects. In *Proceedings of the 14th British Machine Vision Conference*, pages 629–638, 2003.

[16] T. Tuytelaars and L. Van Gool. Synchronizing video sequences. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I-762–I-768, June 2004.

[17] M. Ushizaki, T. Okatani, and K. Deguchi. Video synchronization based on co-occurrence of appearance changes in video sequences. In *Proceedings of the 18th International Conference on Pattern Recognition*, volume 3, pages 71–74, 2006.

[18] A. Whitehead, R. Laganiere, and P. Bose. Temporal synchronization of video sequences in theory and in practice. In *Proceedings of the 7th IEEE Workshop on Applications of Computer Vision*, volume 2, pages 132–137, January 2005.

[19] L. Wolf and A. Zomet. Sequence-to-sequence self calibration. In *European Conference on Computer Vision*, volume 2351 of *Lecture Notes in Computer Science*, pages 530–533. Springer Berlin, 2002.

[20] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[21] J. Yan and M. Pollefeys. Video synchronization via space-time interest point distribution. In *Proceedings of the 6th International Conference on Advanced Concepts for Intelligent Vision Systems*, 2004.