An Environmental Change Detection and Analysis Tool Using Terrestrial Video

by

Javier Velez

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

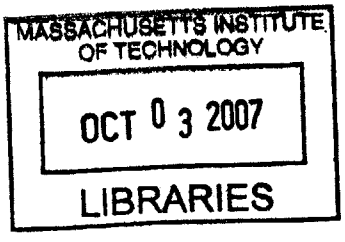
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[September 2006]
September 2007

Author .....
Department of Electrical Engineering and Computer Science
August 22, 2007

Certified by...
Seth Teller
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students



An Environmental Change Detection and Analysis Tool Using Terrestrial Video

by

Javier Velez

Submitted to the Department of Electrical Engineering and Computer Science
on August 22, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

We developed a prototype system to detect and flag changes between pairs of geo-tagged videos of the same scene with similar camera trajectories. The purpose of the system is to help human video analysts detect threats within a set of videos. While computers cannot differentiate threat from non-threat events, they can assist analysts by guiding their attention to sections of video where interesting events are more likely to appear. The system generates a single output video representing the difference between the input pair as well as a set of regions denoting sections of the world where changes occurred between the input videos. These regions represent segments of video where interesting events are likely to be seen. The difference video allows a video analyst to quickly see the differences between the two input videos and decide whether further analysis is required. The system is based on the video matching work by Seth Teller and Peter Sand [14].

Thesis Supervisor: Seth Teller
Title: Associate Professor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, video analyst must watch hours of video footage to find sections of video depicting a possible threat. A human analyst watches video sequences, looking for areas of interest within the video. Some of these videos represent the same scene at different times. For example, the analyst might be looking for improvised explosive device (IED) placement in a convoy route, an increasing threat to U.S. military forces [7] [1]. The analyst has access to several videos of the route, each hours long, and must watch them all and report any sections of possible threat. A video analyst's time and focus are valuable resources, yet each video sequence requires hours of watching. In scenarios like the example above, several video sequences can be nearly identical to each other. Watching the same clip of a truck driving thought an empty road several time is a waste of the video analysts time; nothing has changed since the last video's footage so no threat is present.

The system described in this paper is designed to help focus a video analyst's time and energy in scenarios were several videos of the same scene are to be analyzed for possible threat. Rather than having a human analyst watch the full set of videos one after the other, the system generates a single difference video which contains all changes between the set of input videos. Regions where changes occur between the videos are potential threat regions. By generating a single video of the changes, the system can assist video analysts by guiding human focus and time (both precious resources) to sections of video where interesting things are more likely to appear

(regions where changes occur between the videos).

The current system takes in a pair of videos, with GPS data, of the same scene where the camera's followed roughly the same trajectories. Similar frames from the two videos are matched together to form a dense coupling between frames in one video and frames in the other. Using the dense coupling, a difference video is generated where each frame contains a difference image representing the changes between the corresponding coupled frames from the input pair. Lastly, the system searches for regions of differences within the difference video generated and flags these regions for the analyst.

1.1 Preliminaries

Image processing has steadily improved in both technique and complexity, from the first tepid cycles used to extract centroids from black and white binary images to the slew of procedures used today for feature extraction, segmentation, and registration. Most techniques utilize one or a few images, processing the pixel values in order to extract relevant information for the application at hand. A single image can represent a set of objects in the world. Techniques such as shape from shading and photometric stereo try to extract physical characteristics of the objects seen within an image pair. Using the fundamental mathematical models of light reflection, an object's three-dimensional structure can be inferred (partially); modeling the camera as a pinhole allows depth information to be gained for objects within an image pair. Further processing can segment the image into a set of objects, estimate the motion of objects within a pair of images, and even recognize whether an object has been seen before by the system.

Images contain a lot of information; most image representations today contain a small amount of meta-data (perhaps a time-stamp and such) and include a dense two-dimensional array of values (pixels). Each of these values has d dimensions, or channels, to represent magnitudes of component basis within some color space (usually the RGB color space is used). For example, in the RGB color space each pixel is

represented by three magnitude value pertaining to the amount of Red, Green, and Blue pigmentation combined to get a particular color. The sheer amount of data contained within images precludes the use of techniques requiring many passes through all of the pixels in systems where speed and/or interactivity are major concerns. Feature extraction techniques reduce the amount of information to be processed by transforming the image data (the dense set of pixels) into a sparse feature representation of the pixels.

Features are regions of the image which are somehow judged by the system to be interesting or salient; ideal features should be stable, differentiable, and spatially compact. Stable features are not influenced much by noise in the images or by variations in the camera positioning, the environment of the scene (including lighting), or movement of objects within the scene, as long as the region is still visible within a particular image. The stability of features is important in a system using information gathered from multiple images. For example, say that a system takes two snapshots of a room and wants to see what objects changed in the time between the snapshots. If a particular table in the first snapshot becomes a feature, then in the second shot the table (if it is still within the image) should also become a feature and the two features should be the same (it is the same table, after all), regardless of whether someone turned the light on between images, or a small tremor caused the camera to shift to the left and the table to topple sideways. Features must also be easily distinguishable from each other and similar regions should result in similar features (such features are called differentiable above). Two features describing two different tables should be different; after all, the tables, while similar, are not the same. However, two tables that are similar should have features closer to each other than two widely different tables (or a table and a window). Features are used to reduce the information size of images and therefore must be represented in compact formats; it would not do to represent a small image with a sparse feature set requiring more memory bits than the original pixel array.

Videos are sequences of images strung together; as such, they inherently add time dimensionality to simple images. Video sequences also include small amount

of meta-data, most including the number of frames per second in hertz as well as any compression codecs used to actually store the sequence of images in memory. A particular image, or frame, of a video sequence is referred to by its time index in the video; such indexing is possible because the frames per second of a video are constant. Being a sequence of images also allows frames within videos to be indexed solely by their relative frame position in the sequence (for example. the first frame, the fifth frame, the second to last frame, etc.).

The information content of videos is radically different from that of single images. Whereas single images show a particular scene at a moment in time, videos encode a scene and the changes within the scene in time. Furthermore, video sequences supply a strong temporal constraint to neighboring frames; an object seen within a particular frame, say a red leather chair, must also be seen in the next frame, and the previous frame, unless the object's velocity (or the camera's velocity) is fast enough to force the object out of the camera view. Reasonable frames per second for digital video sequences today range from 15 - 30 hertz, fast enough to capture multiple frames of objects with a large range of velocities. Similar to images, video sequences contain massive amount of information, too much to be rapidly processed by systems, so ways of reducing the information meant to be processed are required (more on this in the Solution section below).

1.2 The Problem

Currently, video analysts must watch hours of video a day and try to pick out interesting or dangerous sections of the videos. The analysts look for sections of video where things "stand out", whether it is a car that has been parked where no car was there before, or maybe a small box appears between one day and the next. A video analysts looks through the hours of stored video for such events and flags them for further study. The car might be simply a homeowner denied their usual parking spot, the box may be a kid's ultimate fortress against water balloons. It is the analyst's job to ignore such events and focus on the spotty truck that pulls up one night, or

the odd-looking pile of debris that suddenly appears at a strategic location.

Humans can be trained to distinguish between such events in videos, computers have yet to reach a level of video, or image, processing capable of easily discerning threat from everyday occurrence. However, the sheer number of video files, and the length of each video sequence, taxes a human's focus. Staring at fifteen hours of video of the same forest is hard, especially if the scene does not change. Not only is it hard for the analyst, but it is a waste of precious resources. Say that there are two videos, each taken of the same road, each four hours long, taken on two different days. In addition let us say that only three events occurred between the videos: a garbage can was tipped over, a car changed parking spots on the roadside, and a shack appeared near the side of the road. These changes are recorded in the second video taken but everything else is the same as the first. In order to find these three changes, an analyst theoretically needs to only watch the three small patches of video in both the first and the second sequence (say this comes to a sum total of fifteen minutes of video). In today's world, however, the analyst will watch all eight hours of video, most of which (seven hours and forty-five minutes) encodes no relevant information about changes in a scene. Clearly, the human analyst's time is being wasted by watching hours of video where no interesting events occur.

While computers can not differentiate threat from non-threat events, they can assist video analysts by guiding human focus and time (both precious resources) to sections of video where interesting things are more likely to appear. The system described in this paper was created in order to improve the effectiveness of the human resource (an analyst's time and focus/attention) to analyze pairs of videos for interesting changes. If an analyst's attention can be guided towards those areas where changes have occurred, then little of their time is wasted watching sections of video pairs where no changes appear between the two sequences.

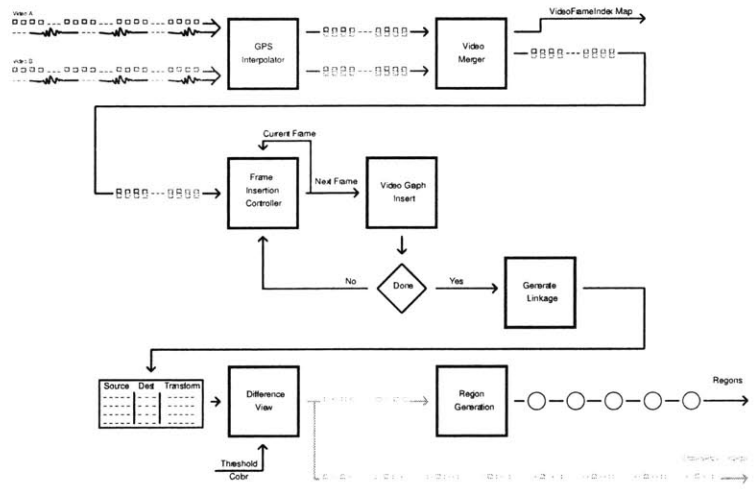


Figure 1-1: High-Level Input/Output Flow of the System. A Video Pair with GPS information is taken as input and a difference video and set of difference regions are generated as output.

1.3 The Solution

Figure 1-1 shows a high-level view of the current system. A pair of geo-coded videos (described below) are loaded into the software. The system processes the video and generates a difference view of the input videos as well as a list of regions where changes occurred between the input pair. The system is based upon the Video Matching [14] system. The difference view of the input pair is generated by first creating a video graph [14] representation of the inputs. Using the video graph, the system matches the video pair together and creates a video of the changes in the input pair.

First, the input video pair is merged into a single input stream with a linear mapping into the video frames for processing ease. An insertion controller takes care of adding all of the frames from the inputs into a video graph. Once the frames have all been added to the video graph, the system generates a linkage from frames in the first video to frame in the second video. This linkage is a dense set of links such that there is one and only one link from every frame of the first video to a frame in the second video. The linkage allows the system to create a difference frame for every input frame in order to generate a difference view of the input pair. Lastly, the processing extracts regions of video where changes occurred.

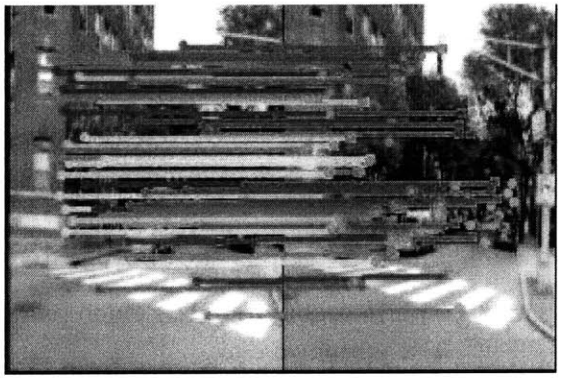
In order to increase the effectiveness of video analysts, several key concepts are utilized, including: geo-coding of videos, matching/registering of video sequences, and the terrestrial ground-plane assumption and its applications. The application is a user-centered proof of concept, and is meant to be a system that video analysts can effectively use to pinpoint areas of interests within pairs of videos. The visual graphical user interface emphasizes the most common tasks and sequencing of events useful to video analysts. Last of all, the system works on real data sets; video pairs encompassing several hours of video can be processed and analyzed with the current system. The sequences used for testing were taken using real-world conditions and realistic camera rigging setups to show the usability of the system.

A geo-coded video sequence is a video with a GPS signal densely associated with the sequence frames. For example, the test sequences have a GPS signal synchronized with the video frames having a two hertz update rate (the two hertz signal is interpolated to get a particular GPS coordinate for each frame in a video). Geo-coded video inputs allow the system to display the sequences in a spatial context for the user. Furthermore, geo-coding lets the system create a rough temporal to spatial mapping across and within video sequences, enabling rough estimates of camera velocity as well as general video sequence paths through the world. The system displays a map of the world to the user and can place the current video section on the map to give analysts greater context when looking at the video sequences; context such as the knowledge that the camera is turning in a large circle or currently moving through a dry river bed.

The most computationally intensive step in the above system diagram (Figure 1-1) is the generation of a video graph in order to match video pairs together. A matched video pair results in a dense linkage between frames from the first video to frames of the second video. Figure 1-2 shows a particular link (source and destination) of a matched video pair. As can be seen, the frames are very similar yet they are from different videos. For each link, or matched frame, the system generates a transform (Figure 1-2(c)) that warps the first frame to the second frame. Once a matching has been found, the two video sequences of the pair can be played in synchrony by the

(a) Source Frame

(b) Destination Frame



(c) Transform

Figure 1-2: A sample link from a matched video pair: source frame, destination frame, and a transform to warp from the source to the destination frame. The color in the transform lines signifies the distance away from paired similar feature after transform (gray lines have close features, red lines were transformed farther away from their paired feature).

analyst, allowing him to see the two matched frames and, most importantly, allowing him to see the differences between the matched frames. A video matching is not simply a linear linkage from one video to the next. Because of changes in the camera movement, position, and setup the linkage can become quiet complex. For example, say that we have two video sequences of a camera in a car driving down a road with three stoplights. In the first sequence, the second and third stoplights are green when the car arrives at them so there exists only a slight slowing of the car before crossing the intersections. In the second sequence, the first and third stoplights are green, the second is red and the car stops for the light before crossing the intersection. For such a video pair, the matching must be able to link the video frames in the first video to those frames most closely resembling them in the second video, even though the relative timing of the linked frames are widely different because of the different motion of the camera (the stoplights make the videos stop at different relative frame times).

The ideal video matching will correctly link every frame in the first video to the frame in the second video which is most like the first frame. The transforms within each frame link is used to negate perspective camera effects as well as small translational shifts between frames. Perspective effects come into play because cameras at slightly different positions will have slightly altered views of the same scene and the camera paths in the video pairs are, understandably, not exactly the same but follow roughly the same trajectories (for example, say both videos are of driving down the same road). The test sequences include video pairs where the camera trajectories follow a car driving down a road, driven by a real human, so these video pairs have similar but not exact trajectories. There is also no guarantee that a whole particular frame in the first video will be in the second video, but a large portion of the view seen in the first frame will be seen within a frame of the second video, hence the transform is required to warp the image seen by the first frame onto the second frame. In the current system, piecewise affine warps are used to approximate this transform between linked video frames in a video matching.

The terrestrial ground-plane assumption allows the system to map image coordi-

nates ((x, y) points of a particular frame) to real world coordinates ($(lat, long)$ points in the real world) by assuming that the image seen by the frame is of a particular plane in the world. This assumption is obviously not true everywhere, but for video sequences where the scene is very flat, the assumption is a reasonable one for mapping image coordinates to world coordinates. Using the ground-plane assumption, the system can extract the rough real-world position of change regions in video sequences. Analysts can then flag regions of the world, rather than regions of video, as being interesting. In order to get true image to world coordinate mappings, the locations of the frames in the real world (taken from the geo-coding) are used to localize a particular image in space, then the ground-plane assumption is used to map the image onto the "plane" of the world.

The graphical user interface allows a video analyst to load a pair of geo-coded videos. Once the system is done with processing, the analyst can view the video sequences together in a synchronized fashion. Along with the synchronized videos, the system also creates a "difference view" of the sequence pair, which highlights areas of the video where the linked frames have differences in them. Figure 1-3 shows a screen-shot of the GUI; note the three synchronized view near the top of the screen, the right-most of which displays the difference view of the other two images. The difference view guides the focus of the analysts to areas of the video where differences actually appear. Sections of video without changes will have very distinct difference views so that video analysts can fast-forward through the regions of inactivity in the video pairs. Furthermore, the system initially extracts regions of video where changes occur and flags them, creating a region in the real world where interesting things likely appear. Analysts can skip from region to region, without actually having to watch the entire video sequence, flagging each region as threat or not depending on what they see. All flags are based on real-world regions. Regions can also be created by the video analysts for sections of video, and sections of the real world, not initially flagged by the system.

A significant part of the system, as a proof of concept, is the ability for real data sets to be processed and analyzed. Real data sets included videos with several

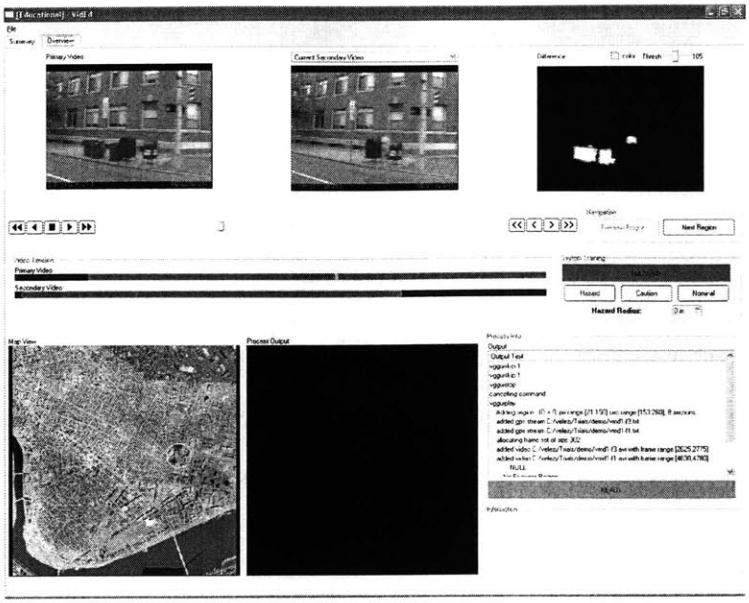


Figure 1-3: Screen-shot of GUI. The top three image displays are synchronized together, with the right-most display showing the differences between the left-most pair of displays. The map shows some flagged regions where changes occurred in the video input pair.

hours worth of data from real environments. The data used and tested was taken using reasonable assumptions; no effort was taken to "clean up" or massage the video sequences for the purpose of testing out the system. In order to function reliably with large data sets, the system employs several methods to reduce space complexity (in effect increasing time complexity) and tries to order its computations to use the inherent sequential nature of videos and maximize cache efficiency for both video data and computed values.

1.4 Applications

The system can be used for any sort of video surveillance. The requirements include: having cameras that follow similar paths (the farther away the paths followed, the worse the video matching and the worse the accuracy of the resulting output), video sequences where the ground-plane assumption is reasonable, and video inputs where the velocities of moving objects is small compared to the frame rate so that an object

appears in more that a single frame. Since the system is designed to flag changes, foreign/dangerous object detection can be easily accomplished using concepts and ideas of the system.

The ability to detect changes in matched videos can also be used by systems that need a way to recognize interesting areas. For example, explorer robots might want to make several passes by a certain structure and only examine (or re-examine) it if any changes occurred after the last time the robots studied the structure. A set of exploration robots can use a similar system to initially explore an area, then act dynamically to changes, only re-exploring sections where interesting things happen rather than re-exploring the entire area assigned for them to explore.

Chapter 2

Previous Work

This work builds upon the Video Matching paper by Peter Sand and Seth Teller [14]. In the paper, salient feature points (in particular a modified version of the Harris feature detector) were used to register frames from two video sequences together. The current system uses the concept of a video graph structure (referred to in [14]) to register frames. The structure represents sets of videos as a single graph where the nodes are individual frames and the links reflect a similarity between a pair of frames. Each link also contains information on how the linked images are registered together, including a confidence level and a transform to warp one image to the other. The original paper builds a video graph using a 2-phase algorithm were first forward links are established (links from the first video to the second) then later a second pass utilizes the forward links to generate links from the second video to the first. The current system uses the concept of a video graph but constructs the structure using a significantly different algorithm.

The idea of change detection is not new to the graphics and vision community. A large body of work is focused towards efficient archiving and searching of video databases. Towards this end, scene change detection is used to segment video sequences into clips which are then characterized and stored in the database for later retrieval (see [12] and [13]). A variety of methods are used to measure the similarity between two video frames, ranging from pixel-based correlation techniques to feature-based approaches more akin to the approach used in this paper.

In order for the system to detect changes, individual video frames representing the same scene must be registered together. Once registered, the differences in the images can be extracted. Image registration has been well-studied and many techniques have been developed; refer to [17] for a survey of registration techniques.

Chapter 3

The System Setup

The system consists of a code-base written in the C++ programming language along with several support libraries. In particular, the system uses the following third-party libraries: Qt for the graphical user interface, the IPL processing libraries for matrix and image computations, and the DirectX 9.0 environment for video access. The software system is split into eight major modules as shown in figure 3-1. The GUI module contains all code pertaining to the user interface and acts like an interface between the Qt library and the rest of the modules. The Common module includes simple utilities that all other modules can use such as simple commands to display message to the user and a set of string processing functions used throughout the system. Together, the Image and Video modules contain all the code used to represent images and video files in the rest of the modules; these modules function as abstractions between the low-level image and video representations and the object used by the high-level processing code. The Math module is full of useful mathematical utilities such as matrix and vector abstractions. The ImageMotion module contains code to represents the transforms between two images using piecewise affine warps. All low-level operating system code, such as performance timers, is abstracted away behind the System module. Lastly, the VideoGraph module contains code responsible for creating and querying a video graph and matching video input pairs together.

The general software architecture of the system follows a server-client approach to systems. The GUI acts like a client, allowing the user to send requests to a server

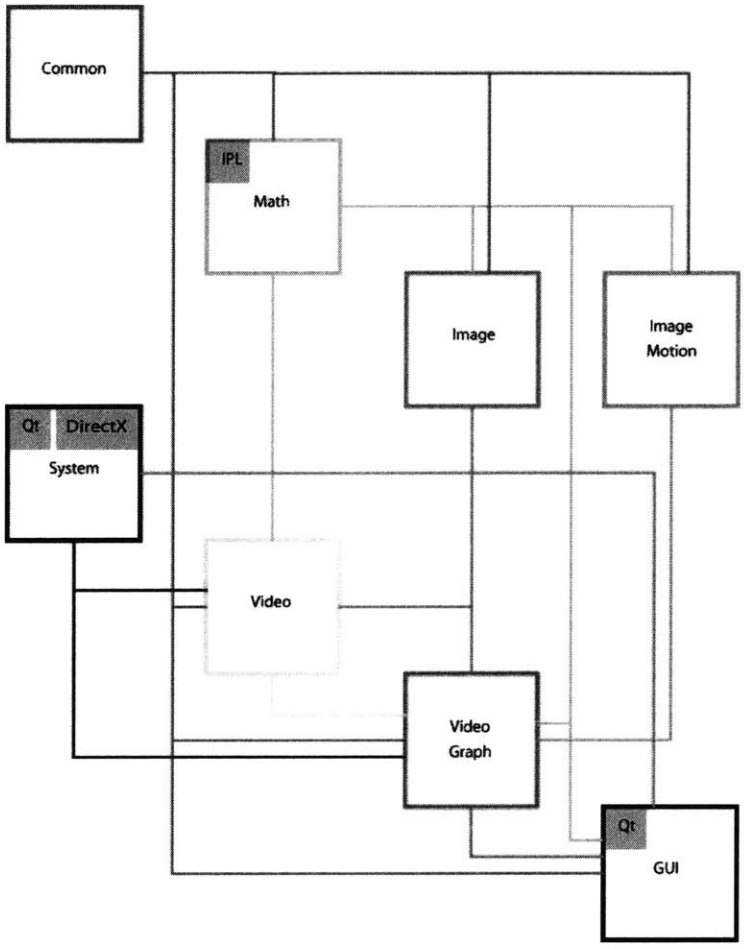


Figure 3-1: Graphical view of software modules and dependencies. No module may be dependant on a module underneath its position. The colored tracks denote the actual module dependencies.

and displaying any responses received from the server. The video graph generation and difference region code acts like a server, accepting requests from the GUI and sending replies back to the GUI. The state of the system is kept in the server part of the code, separate from the client. Such an architecture facilitates the change of interfaces from a GUI to a simple command line or even a networked distributed system where several servers are being accessed by a particular client.

The current system uses the Qt widget library for all graphical user interface code. The interface code using Qt is separate from the rest of the code and encapsulated in a set of small calls so that any widget library supporting windows and mouse/keyboard inputs can be used instead of Qt. The system model was created so that all of the processing runs separate from the GUI, a design decision intended to ease the use of other widget sets or even batch mode processing of data. While Qt contains more than just a widget library, the current system uses only the widgets; the threads, strings, utilities, and signal/slot mechanisms of Qt are purposefully not used in the processing code in order to modularize use of the code base.

Running on the Windows platform, the software uses the DirectX package to read video files. The codecs enabling read/write of video inputs files must also be installed on the machine running the system (the DirectX package takes care of finding any necessary installed codecs). In general, the more RAM a machine has the better for the system (test machines included 2GB of RAM). The current GUI expects a display resolution of 1600x1200 pixels. The machine used for all tests has the following specifications: dual Intel Xeon 3.20 GHz processors, 2.0 GB of RAM (133 MHz), running Windows XP Professional Service Pack 2, and a GeForce 6800 GPU.

In order to synchronize the GPS signal with the video signal, input videos must have the GPS signal recorded onto their audio tracks. Using a simple hardware component, the GPS signal can be serialized and then plugged straight into the microphone jack of the video camera. All data signals are therefore synchronized using the sampling of the video camera itself as a synchronization point, allowing the system to forgo complex multi-device clock synchronization schemes.

The camera used for experiments was a simple DV camera with microphone jack

which took frames of 720x640 resolution at 30 hertz. However, as long as the camera has a microphone jack for the GPS signal, any digital video camera should be able to work. Increasing frame resolution will increase overall processing time, while decreasing the sampling frequency will decrease accuracy and lower the maximum camera/object velocity handled by the system.

Chapter 4

The Basic System

The basic system computes a video graph from the input video pair, a dense matching of the input video pair's frames, and a set of flagged regions in world coordinates. First, the input video pair is processed and a video graph is created. This graph contains linkages from the frames of the first video to the frames of the second video, as well as inter-video linkages. Each link also includes a transform from the source to the destination frame. Once the video graph has been created, a dense video matching must be interpolated from the (possibly sparse) matching stored in the graph. The dense matching must also include transforms from source to destination frames along all links. This mapping is used to "glue" the video pair together and synchronize the two input videos into a single video entity.

The synchronized video inputs are then processed to find regions where interesting changes happen between the video pair. Using the video matching, a difference image is generated for each linked frame pair in the synchronized videos. A difference image representation of the video pair lets the system locate regions of change. Continuous regions of change (changes detected across multiple frame pairs) are all merged into a single flagged region created by the system. After all change regions have been identified, the video analyst can add extra regions, view the synchronized video pair and difference images, and modify any regions flagged in the system.

4.1 Video Graph

Several modifications were made to the original video graph, including: the use of SIFT [9] features rather than corner detectors, the addition of feature clustering, and the use of RANSAC [6] for edge transforms. The internal design of the video graph software also changed since the original version. The video graph was re-designed to be incrementally built, a single frame at a time. The structure contains a set of feature clusters, stored in a structure called the feature table, detailing all distinct features seen so far in any of the frames inserted into the graph. Whenever a new frame is inserted into the video graph, all of the features in the frame are extracted. These extracted features are matched with the feature clusters stored in the feature table and a set of feature clusters is found for the frame. Temporary links are created between the inserted frame and all frames that belong to any cluster within the set of found clusters. If enough temporary links are created between two particular frames, then a bi-directional link is formed between the frame pair, and a transform from source to destination is extracted. Lastly, the feature clusters are updated with any new members, growing if necessary. The basic system creates a video graph by inserting the frames of the video pair in alternating sequence from the first to the last frame of the videos.

4.1.1 SIFT Features

The current system uses SIFT features with 128 element descriptors to build the video graph of a video pair. SIFT features use a pyramidal representation of images. Each level of the pyramid is a sub-sampled Gaussian-blurred version of the previous lower level. The pyramid structure allows the system to extract features at multiple resolutions. Furthermore, SIFT features are somewhat scale invariant. The scale invariance is important in this application because the separate videos might see the same scene from slightly altered perspectives. A video that sees the scene from a slightly closer vantage point will be composed of frames with features that have different scales even though they represent the same locations in real-world coordinates. In addition to

being scale invariant, SIFT features are also rotationally invariant. This is achieved by the feature descriptor construction, which utilizes a histogram of gradients rather than a pixel patch or simple gradient array.

SIFT features were chosen because of their scale invariance, their rotational invariance, and the stability expressed by the feature points. During testing, edge effects became noticeable. Edge features included descriptors which changed greatly with single pixel variations. In addition, a single pixel shift in an image did not correspond to a pixel shift in the extracted SIFT feature when that feature occurred near the edges of the image. Because of these instabilities, the system ignores any features found near the edges of images.

Figure 4-1 shows a graph of the stability of SIFT features versus Harris corner detectors using gradient patch descriptors. The stability of the feature detectors was compared by comparing features extracted from a source image to those resulting from a warped version of the source image. Two parameters where varied: ϵ and δ. The ϵ parameter determines the maximum descriptor distance between two features which are treated as equal. The δ parameter determines the maximal (x,y) distance between two equal features for the features to be considered in the same spot. One thousand random warps were generate which translated the source image up to twenty pixel in either axis (or both). For each detector, the set of features extracted from the original image were compared to those extracted for the warped image. The percent of features which were equal and in the same location was computed for each of the random warps; this value is called the hit fraction. The graph shows three different ϵ levels for the two feature detectors tested. The δ parameter is reported in the x-axis of the figure while the y-axis contains the average hit percentile for the set of random warps generated. The graph clearly show that in both of the higher ϵ trials SIFT features have better average hit percentile than Harris corner features. For the lowest ϵ shown, both detector do reasonably the same (the lines are drawn on top of one another in the figure hence only four separate lines are discernible).

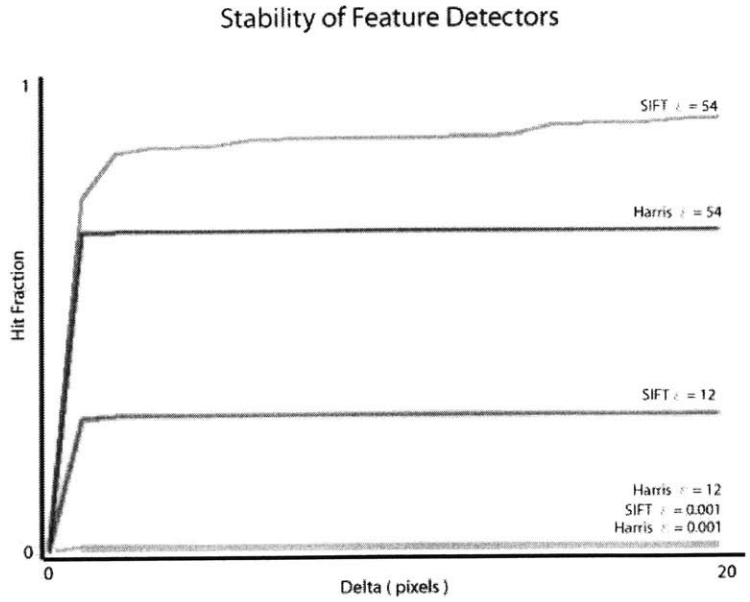


Figure 4-1: Graph of the stability of both SIFT and Harris feature to translational warps. The x-axis represents varying δ, the y-axis is the hit percentile in fraction form, and each line is labeled with the detector and ϵ used to generate the data.

4.1.2 Feature Table

The feature table stores all distinct features seen so far while building a video graph. The table stores a set of clusters rather than the entire feature set. These clusters are used to create temporary links between frames with potential for becoming permanent links. Also, the clusters allow the system to differentiate features which are the same from those which are different; same features belong to the same cluster, different features belong to different clusters. Clusters are identified by their mean descriptor (the mean of all the current descriptors belonging to the cluster), which is updated every time a new frame is inserted into the graph.

The system currently contains two ways to cluster the features: k nearest neighbor, and neighbors within ϵ distance. The distance of features is defined as the L_2 norm of the feature descriptors (the L_2 norm of a 128 dimensional vector) and is as follows: $||f_1 - f_2||_2 = |\mathbf{f_1} - \mathbf{f_2}|$, where f_1 and f_2 are two features, $\mathbf{f_1}$ and $\mathbf{f_2}$ are the corresponding 128-dimensional vectors representing the feature descriptors, and $|\mathbf{a}-\mathbf{b}|$ represents the normal vector distance measure ($\sqrt{\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{b}_i)^2}$). Similarly, the distance between

two feature clusters F_1 and F_2 is defined as: $||F_1 - F_2||_2 = |\bar{\mathbf{F}}_1 - \bar{\mathbf{F}}_2|$, where $\bar{\mathbf{F}}_\alpha$ denotes the 128-dimensional mean descriptor vector representing a feature cluster F_α. Using the same guidelines as above, the distance between a feature cluster and a feature is given as $||F_\alpha - f_\beta|| = |\bar{\mathbf{F}}_\alpha - \mathbf{f}_\beta|$.

The nearest neighbor approach to clustering requires an initialization step before the actual video graph can be built. During initialization, several key frames are inserted into the graph so that their features become part of the feature table; no clustering is done at this time. Once the set of key frames have been inserted, any features with exactly the same descriptors are merged together into a feature cluster; distinct features are upgraded to feature cluster with a single feature in the cluster. This completes the initialization phase. New features are added to the k nearest clusters within the feature table (k is a user defined parameter having to do with the range of feature descriptors expected to be seen within the video, see table 7.1). Because all new features are added to the k-nearest clusters, there is no way for the feature table to generate new clusters using this clustering scheme. Because of this, the selection of the key frames used for initialization is paramount to the performance of the nearest neighbor clustering scheme. The system currently splits the video sequences into k equal length sections of video and chooses a uniformly random frame from each section as a key frame (hence the system utilizes $2k$ key frames to initialize the nearest neighbor clustering). A value of $k = 10$ seemed to give reasonable results for short (less than 1000 frame) sequences.

In addition to the nearest neighbor clustering scheme, the system also allows for clustering based on neighbors within ϵ distance away. This scheme does not require an initialization step. The feature table starts out with an empty set of feature clusters. New features are added to all clusters which are at most ϵ distance from the feature being added (ϵ is a user defined value, see table 7.1). A value of $epsilon = 180$ was used for most test sequences ($epsilon = 220$ for low-contrast sequences). This value was determined by taking two frame which are known to match (by human inspection) and extracting features from both frames to see how far apart the descriptors for equal feature are in the image pair. After all features have been added to neighboring

clusters, any features left without a cluster get upgraded into feature clusters with themselves as the only members. This scheme allows the system to dynamically create new clusters in the feature table as new features are extracted from the frames being used to create the video graph. Notice, however, that clusters can never "shrink" in size, but generally are 128-dimensional spheres centered at the mean descriptor.

Both clustering schemes, nearest neighbor and ϵ distance, must deal with noise in the features extracted from video frames. The nearest neighbor approach allows cluster to grow arbitrarily but does not create new clusters; noisy features get clumped with the nearest clusters. The distance clustering approach does create new clusters, so noisy features will become singular clusters within the feature table. In order to remove such small clusters, the feature table is periodically re-built and re-clustered. Feature clusters with less than a certain number of members are discarded (along with the features that were contained within the clusters unless they are shared with another active cluster). There are two reasons why a cluster contains only a few members: either the cluster was created from noisy feature that are not persistent within the input, or the cluster represents a section of the input which has just begun to be processed so only a few processed frames have seen the particular features. In either case, small clusters are removed when rebuilding the feature table. If the reason for the small member count is because a particular scene was just starting to appear in the frames added to the video graph, then a new cluster will be (re)created with the features of this new scene, as new frames containing the scene are added to the graph. Clusters created by noisy features will not be created again since the true scene does not actually contain such features.

The system uses several data structures to implement the clustering schemes described above. Nearest neighbor clustering is done using a k-d tree [3]. There are two possible data structures used for the ϵ distance scheme: LSH Hash-table [4], and linear search of a cluster set. The LSH hash-table provides fast clustering of high-dimensional elements (our clustering elements are 128-dimensional vectors from the SIFT features). However, LSH hash-tables implement approximate nearest neighbors within ϵ distance queries. A linear search of a set of clusters implements true nearest

neighbors within ϵ distance queries, but is rather slow. $K-d$ trees implement fast k-nearest neighbors queries, but are slow when using high-dimensional elements (a rule of thumb seems to be that 10-20 dimensions is a maximum before the structure becomes slow).

4.1.3 KD-Tree

A kd-tree is a tree spatial data structure in which each node subdivides the total k-dimensional space into two regions using an axis-aligned plane [3]. The tree is initially built from a list of k-dimensional points and takes $O(n \log n)$ time to build. Generally, the nodes of the kd-tree each split using a cycling axis-aligned plane according to depth; the root node splits using an x-axis aligned plane, depth 1 nodes use y-axis aligned planes, depth 2 nodes use z-axis aligned planes, etc. The median in the current axis of the points used to create the kd-tree is usually selected to position the splitting plane for each node created in the tree. Searching for the neighbors within ϵ distance is done using the following procedure: traverse the kd-tree towards the direction of the query point (traverse as if looking for the query point), cull all subtrees that have split points further away than the k nearest neighbors found so far. The nearest neighbor search takes $O(k \log(n))$ where k is the number of nearest neighbors queried for and n is the number of data points in the kd-tree structure [2]. Figure 4-2 shows a two dimensional kd-tree structure constructed out of a set of points; the lines represent the internal division planes created by the tree. The spatial nature of the tree is clearly visible in the figure.

4.1.4 LSH Hash-table

The LSH hash-table data structure utilizes s-stable distributions [4] to compute a hashing scheme that takes advantage of locality to improve upon approximate, and exact, nearest neighbor queries. LSH hash-tables use a locality-sensitive hashing family; the family allows for efficient solutions to the approximate and exact randomized near neighbors problem and is defined where L_s distance norms are used, for any

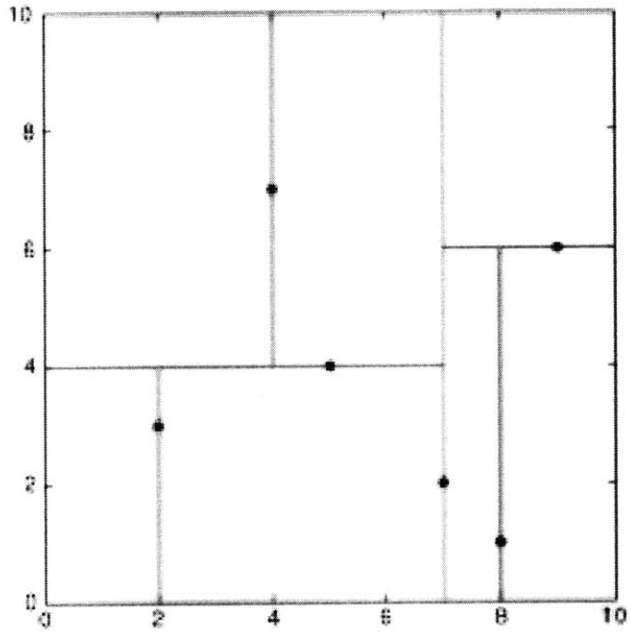


Figure 4-2: A 2-dimensional kd-tree structure partitioning the plane into regions based on the data points used to build it.

$s \in [0, 2]$ [4]. The general idea is to construct a hash family based upon s-stable distributions with the following constraint: the family should be locality sensitive, meaning that the probability of a collision between two points a,b should be inversely proportional to $||a - b||_s$. Formally we get hash functions $h_{a,b}(v) : R^d \rightarrow N$ mapping d-dimensional vectors v onto the integers where a is a d-dimensional vector whose entries are chosen independently from an s-stable distribution, b is uniformly chosen from $[0, w]$, and w is a user chosen parameter. The hash function for given a and b is $h_{a,b} = \lfloor \frac{a \cdot v + b}{w} \rfloor$ [4].

The system uses 2-stable distributions for the LSH hash-table since feature and feature cluster distances are measured using the L_2 norm as described above. LSH hash-tables do well when querying high-dimensional data sets for approximate and exact nearest neighbors, and additionally provide strong rigorous guarantees on any missing near neighbors when given approximate queries [4]. The current system implementation allows the user to select w through a series of separate parameters including key size and subtable count (see table 7.1).

4.1.5 PCA

The system uses SIFT features, each containing a 128-dimensional descriptor. It might be the case that not all dimensions of the feature descriptors are informative. For example, a certain pair of videos could always have the third dimension of the feature space be exactly alike; this dimension is useless and carries no information for either feature clustering or frame matching. A technique called principal component analysis [16], or PCA, allows the system to reduce the dimensionality of the feature space to the k dimensions with the most variance in the data set.

Given a set of d-dimensional vectors, PCA assigns a weight to each component, or dimension, according to how much variance that particular component exhibits in the data set. Principal component analysis is a linear transform which takes a high-dimension data set (d dimensional) and maps it into a k-dimensional space where the first component contains the dimension with highest variance in the data set, the second component is the dimension with the second highest variance, etc [16]. It can

be shown that PCA is very similar to singular value decomposition [15], or SVD.

A singular value decomposition allows the factorization of an $m \times m$ matrix M into:

$$M = U\Sigma V^*$$

where U is a unitary matrix [15], Σ is a diagonal matrix with real positive values, and V^* is the conjugate transpose of V, an $n \times n$ unitary matrix [15]. Intuitively, the singular value decomposition of a matrix constructs a matrix V with a set of "input" bases for M, a matrix U with a set of output bases, and a set of "gain" parameters from input to output in Σ. In the case where M is made up of real numbers, then:

$$V^* = V^T$$

In order to construct the PCA linear transform, we can first factor our data set using SVD to get:

$$X = U\Sigma V^T$$

with the $\Sigma_{i,i}$ ordered in non-increasing fashion. The reduced data set Y of PCA is obtained by projecting the data set matrix down using the first k rows of the singular value matrix.

$$Y = U_k X = \Sigma_k V_k^T$$

It is interesting to note that if X is a positive semi-definite Hermitian matrix, then the SVD produces:

$$X = V\Sigma V^T$$

where V are the eigenvectors of X, and Σ contains the eigenvalues for V. The ability to use SVD to compute the PCA transform stems from the fact that the covariance matrix of X has the exact same eigenvectors as the singular vectors of X, such that:

$$XX^T = U\Sigma^2 U^T$$

The eigenvectors with the largest eigenvalues correspond to the dimension with high-

est correlation in the data set [15] (in this case the highest vectors are the components with highest variances in the dataset since they are taken for the covariance matrix); therefore the SVD factorization is used to find the PCA linear transform.

The system uses PCA to reduce the dimensionality of the SIFT feature space, mostly to utilize the kd-tree clustering data structure. The user can select how many dimensions to keep when reducing the feature space. The reduced dimensionality also effects the LSH hash-table. Empirically, a value of $d = 20$ dimensions seems to work best for the test sequences. The computation of the factor matrix U_k found by the singular value decomposition is updated periodically, using the current feature cluster set as a sampling of the feature space. These updates coincide with the periodic structural re-building and re-clustering of either the kd-tree or the LSH hash-table used by the system to handle noisy features.

4.1.6 Vote Table

Whenever new features are added to the feature table, temporary links are made when clustering these new features with all the feature clusters stored in the table. In order to keep track of these temporary links between frames, the vote table keeps a mapping of all temporary links from the frame being inserted into the video graph to any frame already in the graph. Once all features have created any temporary links, or "voted" for a particular frame-to-frame permanent link, the system analyzes the vote table and builds permanent frame links in the video graph. A particular feature "votes" for all of the frames included in all of the feature clusters the particular feature belongs to. In essence, the votes for a particular frame tally the number of features the frame has that are similar to features in the inserted frame. There exists a unique vote table for every inserted frame. Currently, the system has two ways of creating permanent links: all frames within the vote table with greater than a certain number β of votes are linked with the currently inserted frame, or all frames with a greater than α percent of the votes are linked to the inserted frame; the link creation threshold is user selected. All links created are bi-directional (two links are created, source to destination and destination to source).

4.1.7 Link Transforms, Affine Warps

Once a permanent link is created in the video graph, a corresponding transform from the source of the link to the destination frame is found. In this step, the system defines two features as being similar if they belong to the same feature cluster in the feature table. Given two frames, the source and destination of the link, and a set of features for each frame, the system can create a correspondence between features in the source frame and features in the destination frame with similar features being "linked" together. Note that this does not guarantee a one-to-one correspondence set between source and destination.

The system finds the best transform from source to destination by generating the best affine warp given the correspondences between the two frames. An affine warp consists of six parameters; in matrix notation, affine warps are 3x3 matrices of the following form:

$$A = \begin{bmatrix} \alpha_x & \beta_x & c_x \\ \alpha_y & \beta_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$p' = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Trying to solve the least square minimization problem for the best affine warp using all correspondences takes up a lot of time. Furthermore, least squares minimization is highly affected by noise in the data. To solve both these problems, the system uses the RANSAC approach to find the best, or rather a good enough, affine warp between the source and destination frame.

4.1.8 RANSAC

RANSAC stands for RANdom SAmple Consensus [6], an algorithm designed to fit a specific model to a set of data in the presence of outliers. As opposed to least

square model fitting, RANSAC offers the ability for the model fit to reject certain points as outliers and can yield better results. To find an affine warp from the set of feature correspondences, the system first selects six random correspondences (the minimum number to uniquely solve for an affine warp). An affine warp is found for the chosen correspondences using a least squares error minimization technique. Once this warp has been generated, the system finds the distance of this warp from the theoretical best warp (having a distance of 0) by measuring how well it does on all of the correspondences. If the distance found is better than the distance to the last saved warp, save this warp as the current best transform found. The process repeats until either a certain distance threshold is reached, or a certain number of iterations are completed (both user-defined values, see table 7.1).

An important aspect of RANSAC is the distance metric used to evaluate a particular model, or affine warp in the system's case. Correspondences represent linkages of similar features' (x,y) positions on a pair of frames. The transform from a source to a destination frame represents a way to map (x,y) positions in one frame to positions in the other frame. As such, the basic system uses the average Euclidean distance between all of the correspondences to evaluate affine warps found during the RANSAC procedure. It computes the projected destination (x,y) of the source feature for every correspondence and stores the difference between this predicted position and the position of the destination feature in the correspondence; the average of these distances is used to evaluate a particular affine warp. Formally, the distance of a particular affine warp model A given a set of correspondences X can be written as:

$$dist(A, X) = \frac{\sum_{i=1}^{N} ||A(X_{i,src}) - X_{i,dest}||_2}{N}$$

where N is the number of correspondences, $A(X)$ means X warped using the affine warp A, and X is position (x, y) in image space.

4.1.9 Frame Distance Metric

Once the system has found a permanent link between a pair of frames along with a corresponding transform from the source to the destination frame, linked frames are assigned a "distance" based on how close they are to each other. A frame is represented as a matrix of pixel values. The system defines the magnitude of a pixel value as its gray-scale value. Given a pair of linked frames, S and D, with a transform A going from $S \rightarrow D$, the distance between S and D is:

$$|S - D| = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \sqrt{(|S_{A(i,j)}| - |D_{i,j}|)^2}}{mn}$$

where each image has $m \times n$ pixels. The distance is just the average pixel magnitude difference between the destination frame and the source frame warped onto the coordinates of the destination frame. Destination frame pixels which have no source frame pixels warped to their (x,y) coordinates assume a source pixel of 0 magnitude.

4.2 Interpolation for Dense, Smooth Matching

The video graph generated from a pair of videos is not guaranteed to be a dense matching between the videos. In order to generate such a dense matching, the graph is sampled and interpolated. In addition to interpolating the matching to be a one-to-one matching between frames from the first video to frames from the second video, it makes sense to try to generate as smooth a matching as possible. First, the system samples the links in the video graph to get a sparse set of frame linkages. Using simple linear regression generates a dense linkage set, but linear regression techniques are sensitive to global changes, so a particular dense interpolation of frame links near the beginning of the sequence is affected by sampled linkages near the end of the sequence. Polynomial interpolation also shows similar global sensitivity to the samples. To avoid such sensitivity, the system uses splines [8][11] to interpolate the sampled links. Once frame-to-frame linkages have been densely interpolated, the transforms from the sampled links must also be interpolated to create a dense set of

transforms for the set of links. Affine warp interpolation is also done using splines.

4.2.1 Sampling

The system provides several ways to sample the video graph links in order to interpolate a dense matching. Each link contains a certain weight pertaining to the frame distance of the linked frames. The first sampling approach simply uses the top N links to interpolate a dense matching. A second approach chooses the best link out of every m links of the video graph, and can be thought of as breaking up the graph into sections and choosing a single representative link for each section. The last approach is similar to the second but splits the graph into equal-sized sections (note that the second approach splits the possible links into sections, so gaps in the video graph are not taken into account). If a link does not appear within a section then no sample is chosen for that section. The user of the system can select which sampling method to use for a particular data set. In general, the first sampling approach is used, with N chosen to be very close to the total number of links, since it provides some filtering of outliers. The third sampling approach works better than the second approach, especially in the presence of gaps in the video graph.

4.2.2 Splines

Splines are a parametric interpolation technique. Parametric techniques map the intended interpolation range into a [0,1) range, where 0 is the first value in an interpolation range and 1 is the last value of the range. Splines, in particular, offer local sensitivity as opposed to global sensitivity, so samples near the end of a sequence do not affect the interpolation results near the beginning of the sequence. The system utilizes cubic b-splines, Catmull-Rom splines, and cubic Hermite splines to interpolate the frame linkages from sampled links. The three types of splines all generate a set of basis weight functions for a given set of sampled points. These weight functions are then used to create a mapping from [0,1) to a smooth curve interpolating the sampled points.

B-splines use polynomial functions to generate a smooth curve between the sampled points. The mapping from [0,1) returns the point along the curve at the given relative length; for example, the value 0 is mapped to the very beginning of the curve, the value 0.5 is mapped to the middle of the curve (in terms of distance travelled from beginning of curve to end of curve), and the value 1^- is mapped to the very end of the curve. A k-order uniform b-spline with n samples (or control points as they are called in the literature) is defined as:

$$P(t) = \sum_{i=0}^{n} W_{i,k}(t) P_i$$

where $W_{i,k}$ is the weight basis function for the spline and P_i is the i^{th} sample. The basis function for uniform b-splines are defined recursively as:

$$\begin{aligned} W_{i,k}(t) &= \frac{W_{i,k-1}(t)(t-i)}{i+k-1-t} + \frac{W_{i+1,k-1}(t)(i+k-t)}{i+k-1+1} \\ W_{i,1}(t) &= \begin{cases} 1, & \text{if } i \leq t \leq i+1 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Figure 4-3 shows the cubic b-spline basis functions with $W_{i,k}(t) = bsp_i(x)$. For an order n b-spline, n+1 samples are required. As can be seen from the definitions of the weight basis functions, a sample only affects k segments of the spline, where k is the order of the b-spline being used. Furthermore, every interpolation point is affected by k samples, hence the local sensitivity of b-splines. The system uses cubic b-spline for interpolation. To create a dense mapping using cubic b-splines, the application iterates through the sampled links and uses the current sample and the next three samples to create a local cubic b-spline. The local b-spline is evaluated at a number of uniformly spaced positions based on the first and last source frames of the samples to produce the dense mapping. Figure 4-4 shows a sample interpolation curve generated by the system while analyzing a input video pair. The red marks are sampled links, the green curve is the interpolated values. Note how the splines follow the data,

always maintaining a smooth matching.

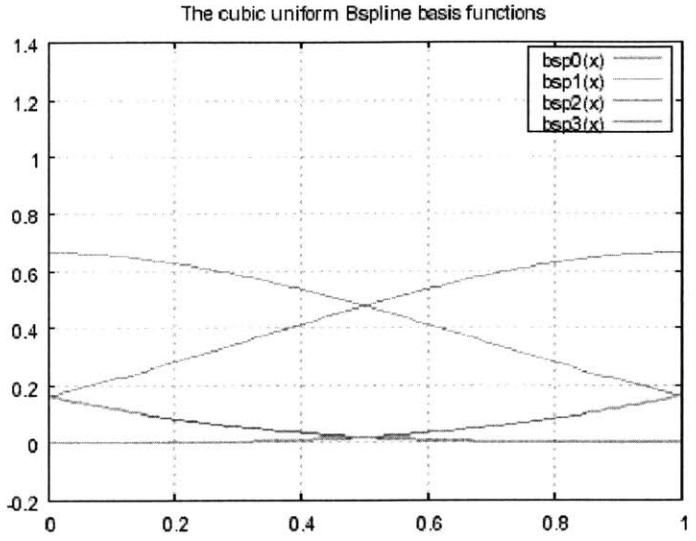


Figure 4-3: Basis functions for uniform cubic b-splines.

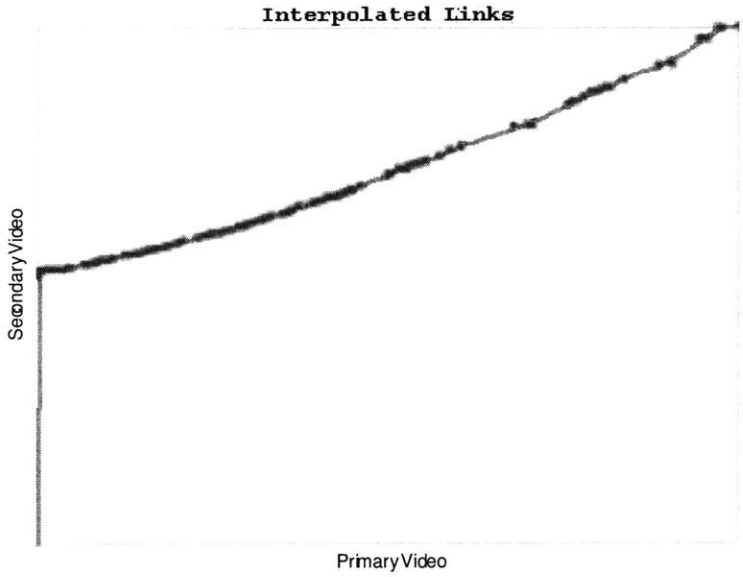


Figure 4-4: Interpolation of sample video graph links using cubic B-Splines.

General uniform b-splines do not guarantee that the sample points will be part of the interpolated curve. For some video pairs, it is useful to have the ability to create a dense, smooth mapping that goes through every single sampled link of the video

graph. Catmull-Rom splines generate interpolation curves which are guaranteed to go through the sample points used to generate the splines. Catmull-Rom splines also guarantee that the stitching of the local interpolation curve generates tangents that result in a smooth interpolation curve for the global set of samples. The splines are a subset of the more general cubic Hermite splines. Cubic Hermite splines use a pair of sample points, P_0 and P_1 along with a starting and ending tangents M_0 and M_1. The generated curve is of the form:

$$P(t) = (2t^3 - 3t^2 + 1)P_0 + (t^3 - 2t^2 + t)M_0 + (-2t^3 + 3t^2)P_1 + (t^3 - t^2)M_1$$

where $t \in [0, 1]$. Using the terminology of weight functions, cubic Hermite functions have the following weight basis:

$$\begin{aligned} W_{0,0}(t) &= 2t^3 - 3t^2 + 1 \\ W_{1,0}(t) &= t^3 - 2t^2 + t \\ W_{0,1}(t) &= -2t^3 + 3t^2 \\ W_{1,1}(t) &= t^3 - t^2 \end{aligned}$$

Figure 4-5 shows the cubic Hermite spline basis functions with $W_{x,y} = Hxy$. A particular set of cubic Hermite splines, called cardinal splines, have the tangents M_0 and M_1 defined using a "tension" parameter, such that:

$$M_i = \frac{1}{2}(1 - t)(P_{i+1} - P_{i-1})$$

with the first and last tangents user defined as well as the tension parameter t. Similarly, Kochnek-Bartels splines [8] are defined using tension, bias, and continuity parameters as follows:

$$S_i = \frac{(1 - t)(1 + b)(1 - c)}{2}(P_i - P_{i-1}) + \frac{(1 - t)(1 - b)(1 + c)}{2}(P_{i+1} - P_i)$$

$$D_i = \frac{(1-t)(1+b)(1+c)}{2}(P_i - P_{i-1}) + \frac{(1-t)(1-b)(1-c)}{2}(P_{i+1} - P_i)$$

where D_i are starting tangents, S_i are ending tangents, b is the bias, t is tension, and c is continuity. The current system allows the user to set the tension, bias, and continuity; note that setting all three to 0 results in a Catmull-Rom spline.

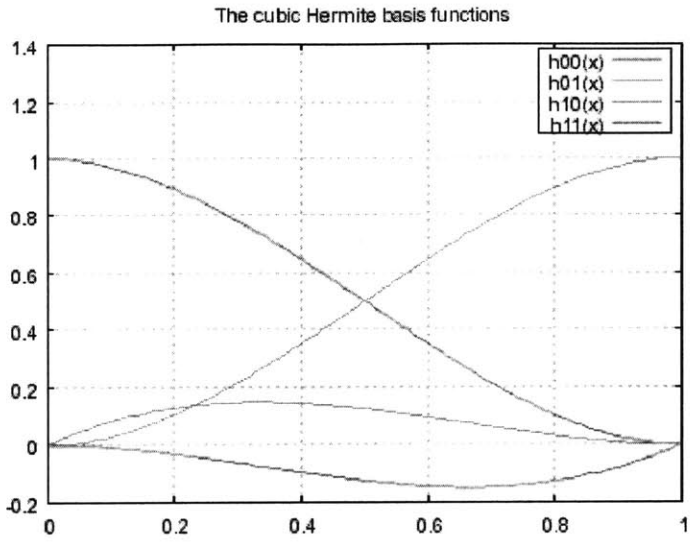


Figure 4-5: Cubic Hermite spline basis functions

4.2.3 NURBS

The above splines all use cubic polynomial curves to interpolate local samples. Another approach is to fit a set of ration polynomial curves. Furthermore, the above splines all assume a uniform placement of the sample points in order to create the mapping from [0,1) to the interpolating curve. Non-Uniform Rational B-Splines, or NURBS for short, utilize rational polynomial functions for curve fitting and allow the sample points to be mapped in a non-uniform matter to generate better, smoother dense matchings of the videos. In addition to the sample points, NURBS require a "knot" vector which represents the relative position of the sample points (this is the

non-uniform part of NURBS). NURBS are defined using the following formula:

$$
\begin{aligned}
P(u) &= \sum_{i=0}^{n} P_i R_{i,p}(u) \\
R_{i,p}(u) &= \frac{w_i N_{i,p}(u)}{\sum_{j=0}^{n} w_j N_{j,p}(u)} \\
N_{i,0}(u) &= \begin{cases} 1, & \text{if } u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases} \\
N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)
\end{aligned}
$$

where $U = u_0, ..., u_m$ is the knot vector, $R_{i,p}$ are the rational basis functions, $N_{i,p}$ are the normalized b-spline basis function of degree p, and w_i is the weight of the i^{th} sample point. The degree, knot vector size, and number of samples are related by the formula $m = n + p + 1$. It is interesting to note that Bezier and non-rational b-spline curves are special cases of NURBS. NURBS are only locally sensitive, so a change in either weight or sample point only effects $p + 1$ knot-spans of the interpolation curve.

Cubic B-Splines were used in all trials and experiments. The dense matching interpolated using the cubic b-splines proved adequate for the data without incurring the added complexity of a knot vector definition (NURBS) or extra parameters (tension, bias, and continuity). Given perfect data to interpolate, the Catmull-Rom splines generated a better curve; each perfect data-point lies on the interpolated curve. However, the effect of outliers in the generated interpolation curve is greater using Catmull-Rom splines than using uniform cubic b-splines, therefore uniform cubic b-splines were used for all of the trials.

4.2.4 Warps

All of the splines discussed above have basis functions defined in terms of sample points. These methods are all generalizable to n-dimensional points rather than simple (x,y) coordinates used to get dense video matchings. An affine warp can be

represented as a point in 6-space defined by the six parameters of the affine warp. Using such a representation, affine warps are interpolated in a similar manner than links. The system uses uniform cubic b-spline curves with 6-dimensional sample points in order to generate a dense set of affine warps, one for each link.

4.3 Difference Image

The system uses the dense matching to create a difference image for each video frame of the video pair. For each link in the matching, the source frame is transformed onto the destination's frame coordinate system using the affine warp. Once on the same coordinate system, create a new gray-scale image G with the pixel-by-pixel magnitude differences of the two frames using $G_{x,y} = |(|S_{A(x,y)}| - |D_{x,y}|)|$, where S is the source frame, D is the destination frame, and A is the affine warp between the source and the destination. Figure 4-6(c) shows a sample difference image created from the pair of frames seen in figures 4-6(a) and 4-6(b). Differences between the two images appear in white, while sections of the images which are similar have darker shades of gray. Furthermore, the difference image encodes the information from both frames for the areas where the frames differ. Figure 4-6(c) clearly shows objects in the first frame and objects in the second frame in the regions of change (the white regions) encoded so that the human eye can pick out general shape information about both images used to construct the difference image. The system uses the difference image representation of a video pair to focus the attention of video analysts towards regions of the videos where change occurs. Videos with no change will have dark gray difference images; small changes in a scene, for example a box appearing between the videos, will have dark difference images with the change easily distinguishable highlighted bright white.

The above difference image in generated by taking the difference in magnitude in a pixel-by-pixel basis. The magnitude of a pixel is greatly affected by the illumination of a scene. It follows that the difference image will also be greatly affected by scene illumination. The current difference image generation procedure does not

normalize the scene's brightness, nor takes illumination into account when generating a difference image. Large area illumination effects will produce incorrect difference images. For example, say that we have two images of the exact same object from the exact same viewpoint. However, in one scene, the sun is directly overhead whereas in the other scene the sun has rotated around the object 45 degrees. The large-scale illumination changes between the two scenes will cause the generated difference image to show a large difference between the two scenes, even though they are of the exact same object and viewpoint. Illumination directly affects the magnitude of pixels in an image. The difference images are generated using the difference between pixel magnitudes, and are therefore also affected by global and local illumination changes in a scene. In section 5.15, an improvement to the difference images is discussed which help deal with global illumination changes between images. Intelligently dealing with local illumination effects is still an open problem in the field.

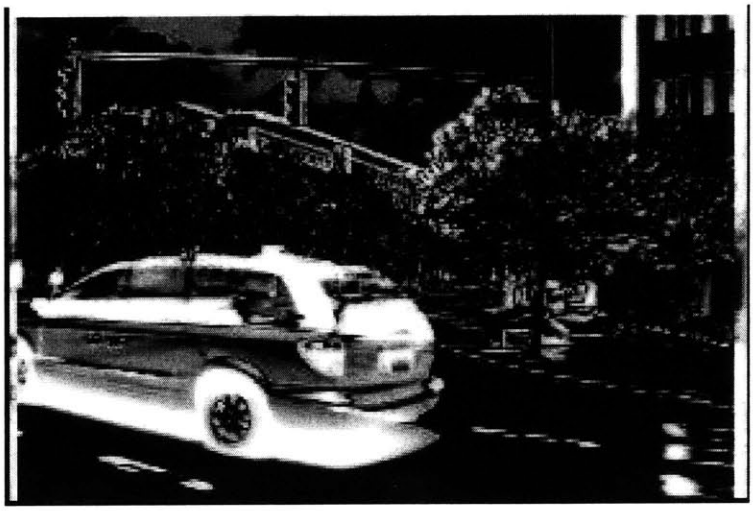
4.4 Difference Region Within a Frame Pair

The system uses the difference images to create regions of change for each frame pair in the dense video matching. A quad-tree is used to recursively split the difference image into regions with change and regions without change. These regions represent areas of the difference image where change occurs and could denotes interesting sections of the images for further analysis. As of this writing, the system does not use these difference regions for any further processing.

4.5 Coordinate Systems

The current system utilizes the GPS signal stored in the audio channel of the input video pair to create a mapping between image coordinates and real-world coordinates. The world coordinates are encoded in UTM [5] format, a triplet consisting of an (x,y) point in meters and a single letter code denoting the origin of the coordinate system in the world map. The application can switch from decimal Latitude/Longitude

(a) Source Frame (b) Destination Frame



(c) Difference View

Figure 4-6: A sample difference view generated from the shown source and destination frames.

coordinates to UTM and vice-versa at will, but internally uses UTM coordinates to synchronize all stored locations. Providing transforms from image coordinates to world coordinates requires the use of the terrestrial ground-plane assumption.

4.5.1 UTM Coordinates

Decimal Latitude/Longitude readings from the GPS are translated into NAD83 UTM coordinates using a fairly complicated model of the earth. The UTM system divides the surface of the earth into regions, each with their very own origin and metric Cartesian coordinate system. The earth's shape is modeled as a non-ideal ellipse with a stretch factor proportional to the longitude of a particular section. The parameters for the elliptical model and stretch factor are estimated from extensive surveys of the land. See [5] for further details on the UTM coordinate system.

4.5.2 Ground-plane Assumption

The pinhole camera model allows the system to transform world coordinates into (x,y) positions in the image plane. The perspective transform, a transform from world to image coordinates, is represented by a simple equation in the pinhole model.

$$\begin{bmatrix} x_{image} \\ y_{image} \end{bmatrix} = \begin{bmatrix} x_{world} \\ y_{world} \end{bmatrix} \frac{f}{z_{world}}$$

where f is the camera's focal length. The ground-plane assumption allows the system to generate a transform from image coordinates back to world coordinates by assuming that the world seen by the image is really a plane rather than a three-dimensional scene. Using the assumption, rays are extended from the camera position through the coordinates in the image plane to be transformed into world coordinates. The rays eventually intersect the assumed plane of the world and result in a particular (x,y) coordinate in planar world coordinates. Figure 4-7 shows the pinhole camera model with a point p in the ground plane, and a ray from the camera through the image plane to p used to transform the image coordinates to world coordinates on

the plane. The system approximates real world coordinates with these planar world coordinates.

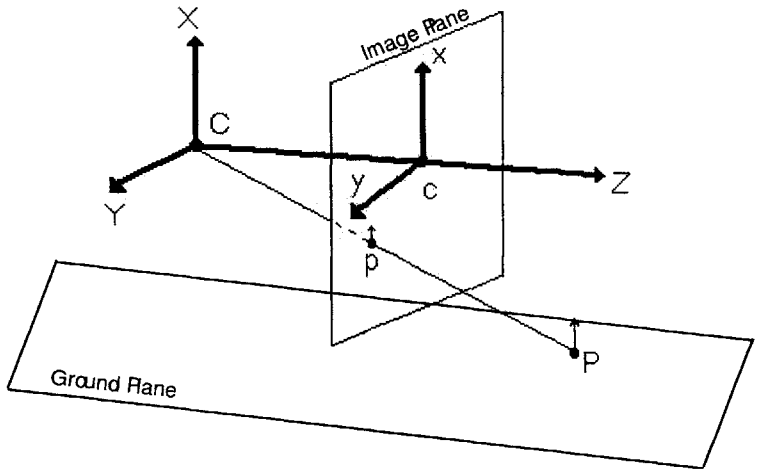


Figure 4-7: The pinhole camera model and ground-plane assumption. The ray from the camera to the point p in the image plane is used to generate the world coordinates of p in the ground plane.

4.6 Region Flags

The system stores a set of flags marking regions of the world. Once a pair of videos have been processed and a dense matching found, regions of high difference between the videos are clustered and flagged as interesting sites. The system iterates through the matchings checking for high change regions in the difference images of the linked frames. When a difference image is found containing a large difference, the system searches forward in the linked video pair until the change disappears and creates a new circular region in the world encompassing the positions of the camera pertaining to the difference seen. Such region flags shift the video analyst's attention towards sections of the world where changes occur in the input videos.

4.7 User Interface

The system's graphical user interface is composed of two major display tabs: a map view tab and an overview tab. The map view tab allows video analysts to select the video pair to process as well as displaying the GPS trace of the videos selected onto a satellite map. The user can zoom in and see detailed traces of the video sequences as well as the region of view of the current frame. Along with the map view, the map tab also provides a small display of the current frame. Figure 4-8 shows a screen-shot of the map tab with the gps trace of the currently loaded input pair. The overview tab is where the video analysts start the processing. The tab shows a synchronized view of the matching between the input video pair along with the difference view. A video time-line is included in the interface with the current frame of the sequence selected, along with the matched frame of the other video. Figure 4-9 shows a screen-shot of the overview tab with the synchronized views and video time-lines. The system tags the first video of the input pair as the primary video, used as ground truth, while the other video is termed the secondary video. To provide real world context, the overview tab also includes a small map view with the current frame position marked. Both maps also display flag icons denoting all of the stored region flags in the system. The entire user interface is synchronized so that a change in the map tab, say that the current frame was moved forward three seconds, will propagate to the overview tab; synchronizing the changes makes the interface consistent and easier to understand since the different tabs and views are simply viewpoints into the current, underlying, system state.

4.7.1 Region Flag Interface

The system creates a set of region flags when a video pair is processed. However, a video analyst might want to create new regions or change the properties of the regions found by the system. The overview tab allows the user to create a new region centered at the current frame's location and extending for a given radius in the world. Regions are displayed as small flag icons on the maps; flags with black borders have been

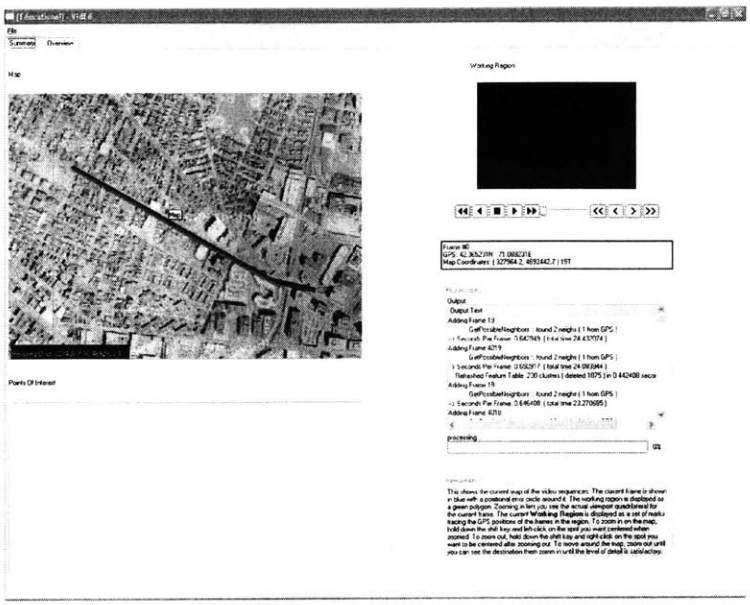


Figure 4-8: Screen-shot of the Map tab in the Graphical User Interface. A trace of input pair is shown drawn onto the map.

Figure 4-9: Screen-shot of the Overview tab in the Graphical User Interface. The synchronized views show a matching and corresponding difference image. The video time-lines show the positions of the displayed frames in the input video pair.

edited or created by the video analyst while flags without the border are unchanged system-generated regions. Clicking on a flag selects that region and resets the view to show the first frame within the region. A selected region can have its properties, such as threat rating, changed by the user. In addition, a video analyst can choose to iterate over the regions rather than watching the entire video pair; the analyst skips all sections of the input pair where no changes were registered by the system, focusing on only those sections of video where processing detected significant differences in the frames. Figure 4-10 shows the map tab with a set of regions displayed.

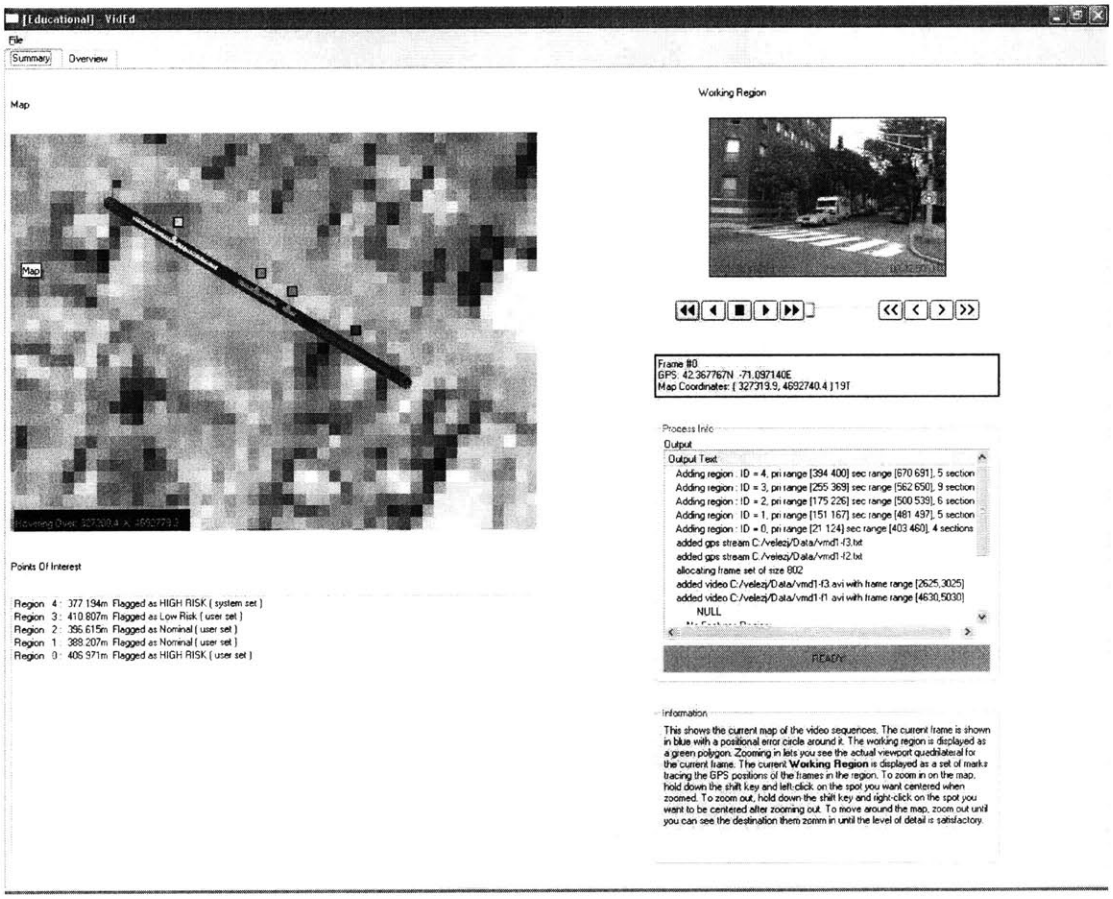


Figure 4-10: The map tab with a set of regions flagged. The color of the flag represents the threat level and the circle around the flag shows the region' radius in the world.

4.7.2 Synchronized Frame Views

The overview tab shows a synchronized view of the current primary frame, the matched secondary frame, and the difference view of the two frames (see Figure 4-9). The views are synchronized at all times, so zooming in a particular display causes all three views to zoom to the particular spot. The system uses the primary video as a basis for all the user interactions, allowing analysts to play forward and backwards the primary video; the three views remain synchronized, so playing forward the primary video also displays the linked secondary video frames and the difference images of primary and secondary videos.

4.8 System State

The entire state of the system can be saved, including any partial matchings found along with built video graphs and dense interpolations generated. The system can also be restored from the saved state. Since the saved state represents the entire system at a particular moment in time, the size of the state file in the hard-disk is proportional to the current memory being used by the system. The save file format is user-readable (including comments) and can be used to view all of the currently loaded structures in memory. Although system state can be stored and loaded, the system does utilize randomness in certain functions, therefore two runs of the exact same inputs will not necessarily result in the exact same output. In particular, the video graph uses the RANSAC approach, so the graph structure may differ slightly from run to run.

Chapter 5

Extensions to the Basic System

This section describes many extensions to the basic system detailed in the previous section. Some of the extensions do not in fact "improve" the system but are nonetheless useful to understand or think about and seemed to be improvements upon conception.

5.1 Video Graph Frame Insertion Order

The order in which frames are added to the incrementally built video graph has a large effect on the resulting graph created. The basic system interleaves the two videos and adds their frames in alternating order starting from the first frame of the primary video. Two extensions utilize different strategies to try to get "better" video graphs out of the process. The extensions were tested by trying to build a video graph out of a pair of the same video.

The first extension to the insertion order follows a control-theoretic approach. The two videos are treated as two signals coming into a control system. The goal of the control system is to add the frames in such a way that a secondary frame linked to a primary frame should be inserted right after the primary frame. In essence, the control system tries to track the current matching of the video and inserts the frames so that a prospective secondary linked frame is inserted right after the possibly linked primary frame. The control strategy follows a simple PID controller, where the goal

state is to add the frames in order for each video while keeping the newly inserted frames linked to the previously inserted frames. The system tries to avoid the state where multiple frames are added to the video graph without forming any new links between the primary and secondary videos. In the basic system, for example, if an input pair consists of the first one hundred frames of video A and frames 50 - 100 of the same video A, then adding the frames in interleaving order makes no sense since the first 50 frames of the primary video ought not link well with the first 50 frames of the secondary video (frames 50 - 100 of video A in this case).

The PID controller assumes that the videos are linearly linked so that the next possible link for a particular primary video frame is an offset into the secondary video. The controller uses a sliding window of the previous k links created in the video graph to estimate this offset by choosing the mean offset between primary and secondary video frame of the k links. The system adds frames from both the primary and secondary videos, making sure neither "falls behind" by trying to add frames such that the next secondary frame is exactly the estimated offset of the next primary frame. The PID controller uses user-tuned gains to keep the insertion order synchronized with the estimated offset. The following gains were used by the author for all trials: $K_p = 1.5$, $K_d = 0.25$, and $K_i = 0.2$. If a video sequence falls behind, say that the estimated offset is 10 but the next video frames are 30 and 70 for primary and secondary videos respectively, the controller "corrects" the order by repeatedly choosing to insert frames from the lagged video until finally the next video frame pair is synchronized with the estimated offset (60, 70 for the example).

Figure 5-1 shows a diagram of the PID controller. The current primary and secondary frames are used as inputs to the controller along with the mean offset and previous error (initialized to 0). The controller sets the current error, $error(t)$, to be the difference between the mean, or target, offset and the current offset. The controller calculates

$$output(t) = K_p(error) + \sum_{i=1} t(K_i * error(i)) + K_d * (error(t) - error(t-1))$$

A negative output results in the controller choosing to insert the next frame from the primary video whereas a positive output results in the next secondary video frame being inserted into the video graph.

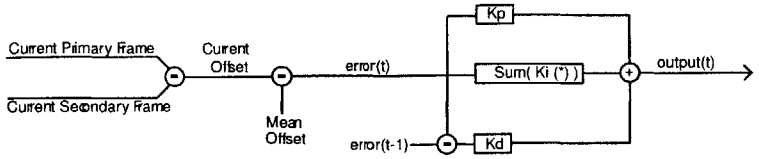


Figure 5-1: PID controller used to insert frames. A negative output results in primary frames being inserted, a positive output means that the next secondary video frame is inserted into the video graph.

This extension allows the system to handle changes in the velocity of the cameras between video input pairs. For example, the PID controller allows the insertion order to slowly keep up with an input pair sequence where the first video was generated from a car moving faster than the car used to generate the second video. The basic system's insertion controller will eventually add consecutive non-matching frames since the two cameras are moving through the same scene at different speeds. The PID controller updates the mean offset used to add frames to the system, keeping track of the next logical matching frames to add from each video input.

The second extension to the basic system uses an insertion queue to dictate which frame to insert next. Similar to the PID controller, the extension assumes a linear offset for linkages; this approach simply tries to make sure a primary video frame has been inserted into the graph before the secondary frame to which it links is inserted. The system keeps a queue of frames to insert next. Every time a particular frame is added to the graph, the system guesses which frame will link to it by estimating the offset as above. The system adds a region around the estimated linked frame to the insertion queue, then adds any frames these newly queued frames are estimated to link with, until k levels of linkages have been added to the queue. The first element in the queue is popped and inserted, repeating the above process. The depth k and size of the region around estimates used to enqueue frames are chosen by the user. Values of $k = 1$ and a region size of 5 were used and seemed to work well. If no elements are in the insertion queue, the system falls back to the strategy employed

by the basic system.

This extension allows the system to skip certain frames while inserting the frames from the video inputs. Rather than inserting the frames from a particular video in order (frame 1 before frame 2, frame 2 before frame 3, etc.), the extension allows gaps in the insertion order based upon the running offset computed. Furthermore, the insertion controller actively tries to make sure a particular primary video frame is already part of the video graph before adding the estimated linked secondary frame. The depth and region size parameters determine the controller's response time for adapting to camera velocity changes in the video input pair. In general, this insertion strategy worked best, enabling the system to cope with camera velocity changes between the video input pair and allowing the system to insert video frames from a particular video out or order.

5.2 Persistent Feature Filtering

The basic system uses all features extracted for a given frame to build permanent links in the video graph. However, noise in the image and errors from the actual feature detector can cause features to appear in an image which are really not good for building the graph. If a feature is truly generated because of image noise or detector noise, the feature most likely does not appear in all the frames around the current frame. Using a feature's "persistence", the system can filter out noisy, non-stable features that should not be used to generate the video graph. When a set of features is extracted for a given frame, the features are clustered with all of the feature from the past n frames in the video. All features within a cluster are regarded as equal. Features which do not appear in sufficiently many past frames are discarded as being unstable. Furthermore, equal features should be in similar positions within an image set in order for them to be counted as persistent. The system generates a bounding box centered on every feature of the newly inserted frame. Following the links backwards between the previous n frames, the box is transformed from the current frame's coordinate system into the previous frame's coordinate system. Any

features of the previous frame tagged as equal to the current feature but lies outside of the now warped bounding box are not counted towards the persistence of the current feature. The box is back-chained and warped through all of the previous n frames using the links (and the transforms attached to them). The features are clustered using their distance and a user defined threshold, this means that a pair of features could both be equal to a third feature but not equal to each other. Refer to table 7.1 for values used for the parameters.

For video input sequences where lots of noisy features are detected, the persistence filtering of features improved the performance of the system. For example, in low-contrast videos, the features detected tend to include more erroneous features. In such low-contrast settings, this extension improved the generated video graph for the input pair, provided that there were enough feature points after the filtering. However, persistence filtering can also lead to worse video graph generation due to the lack of feature points with which to match frames. In feature-rich scenes, the filter also improved the video graph created, but the improvement is small compared to the overall quality of the video graph generated without the filtering. Persistent feature filtering was used for trials with low-contrast video pair sequences, with the persistence box size of 30 pixels and $n = 7$.

5.3 Multi-Scope Frame Search for Clustering

The clustering done within the feature table takes a lot of time relative to other parts of the input processing. In order to avoid having to cluster every single feature, the system looks at the previous frame and all of the feature clusters containing members belonging to the previous frame. This hopefully small set of clusters are updated with the features of the new frame. Clustering stops for any features which become part of a cluster from this select set. Features that do not fall into any of the previous frame's clusters are inserted into the feature table to be fully clustered as in the basic system.

This extension works well, improving the overall system's processing performance

in terms of frames per seconds processed. Since video sequences are temporally consistent, the features of a frame should be similar to the features of the previous frame. This extension makes use of the previous observation to narrow the search space for clustering. Because not all features are clustered against the entire feature table, the ability for the system to close loops in the video graph is diminished. A closed loop in the video graph represents a case where the video input pair records a particular area more than once (say that a truck circles a particular street). In such cases, the features for the area are the same and should be clustered together using the feature table. However, the extension limits the features that are clustered using the feature table to those which are dissimilar from feature in the previous frame, reducing the chances of the system being able to link frames from the first view of the area to frames from the second view (likely far apart in time).

5.4 Vote Table Percentile Drop-off Threshold

Rather than having the vote table create permanent links in the video graph if enough temporary links are detected, this extension creates permanent links for all frame pairs with more than a user-defined percentile of the total temporary links. The system iterates through the frame pairs, beginning with the pairings with the most temporary links. If the percentile of temporary links belonging to the current frame pair is higher than a certain threshold (10 percent was used in the system), the system decides to generate a permanent link in the video graph. This is simply another way to distinguish which frame links should become permanent and which should not. For cases where the temporary links are concentrated in a set of frames, this threshold performs better than a simple numerical cutoff, since it does not depend on the number of features extracted from the video frames. As will be reported in the Results section, the feature density for different frames varies between video sequences.

The extension improved the speed of the system by stopping the search for links without having to process each possible frame pair. However, the extension forces

only the top voted frames to be matched, which usually resulted in only frames from within the same video being matched. The vote count for other-video frames is predominantly lower than the vote count for local frames, hence this extension reduced the quality of the video graph produced from an input pair. When used, a vote percentile threshold of 10 worked best.

5.5 Warp Search and Interpolation for Dense Matching

The basic system interpolates the sampled warps to generate a dense set of transforms for the matching. However, non-sampled links around the interpolated points can supply information on the correct transform to use for a particular interpolated link. For example, the dense matching may contain an interpolated link that is also a link in the video graph that, because of the sampling, was not used to generate the interpolation. The extension to the system allows warps to be sampled separately from the links before generating the interpolation curve for all transforms. Furthermore, nearby links are used to find warps to include in the interpolation, so if the system is searching for a warp between frames 7 and 23 it can use a warp found for frames 7 and 24. Because the video sequences are temporally tied together, warps between close frames are similar to each other since the frames are similar to each other. This extension takes advantage of the tight coupling between nearby frames.

This extension improved the interpolated transforms for the dense matching. Neighboring links around an interpolated linkage often had good transforms. These transforms, when included in the transform interpolation, generated a better overall dense matching. However, the extension is heavily affected by the number of erroneous links in the video graph. In the presence of many erroneous links, the extension generates a dense matching with inconsistent and incorrect transforms for the linkages interpolated, resulting in a poor difference video.

5.6 Warp "Goodness" Metric

The RANSAC procedure used to generate frame link transforms, affine warps in particular, can sometimes lead to erroneous warps. The sampled correspondences, usually six in all, are used to find an affine warp using least-squares. However, least-squares may result in bogus warps when the sampled points are co-linear, or very nearly so (numerical restrictions start applying when points are nearly co-linear). The least-squares algorithm for finding the affine warp given a set of correspondences is not stable in such cases and returns erroneous warps. To correct for such problems, the system defines a set of constraints on warps termed "good". If a warp fails to meet the constraints, the system rejects the warp rather than using it. The constraints include a maximum total possible translation of 100 pixels (it makes no sense to have a warp where the coordinate transform from source to destination shifts half the points outside of the image region). Scaling is also limited to a 20 percent change, as well as the total amount of rotation exhibited by the warp (less than 20 degrees).

This extension significantly reduced the amount of erroneous warps found by the system. In general, erroneous warps were found when the linked frames were not very similar or when the frames were in fact incorrectly linked. The limiting of total translation along with imposed limits on scaling and rotation filtered out warps that were truly inconsistent with the model and assumptions for the inputs sequences; it makes no sense to find a warp which requires a particular video to suddenly have an upside-down camera angle on a particular scene.

5.7 RANSAC Hit Metric and Thresholds

The basic system uses a simple (x,y) Euclidean distance metric for the RANSAC procedure. However, the best warp might not be the one with the least distance but rather the warp which is most consistent with the correspondences. This extension allows warps which transform the source of a correspondence within a small radius of the correspondence's destination to assume a perfect matching of zero distance.

The system then looks at the percent of correspondences that are consistent with the warp (those which have zero distance) versus those that are not consistent in order to decide whether to keep the warp in the RANSAC iteration or not. This evaluation method for warps is called the hit metric.

There exist several ways of evaluating a warp. The basic system measures warps by the mean distance of the correspondences. As seen above, RANSAC is extended to also use a percentile of consist correspondences as the metric for a particular warp. Lastly, the minimal maximal distance for any correspondence can be used to assign a weight to a particular affine warp.

A very important case where the hit metric outperforms the distance metric is when there are one-to-many mappings between features of one frame and features of the other. The correspondences are not guaranteed to be one-to-one, and in fact trials show that they almost never are. The Euclidean distance metric penalizes features which map to several similar features in the linked frame, and assigns better scores to transforms which warp features onto the centroid of all the linked correspondences than to transforms which closely match a single one of the many correspondences. A transform which warps to the centroid of the correspondence set of a particular feature fails to represent any one of the correspondences; in reality, a particular feature can only be in one place on the linked frame, so in fact only a single one of the correspondences from a source feature to multiple destination features is correct. The hit metric allows warps to represent a single one of the one-to-many correspondences without being penalized for not correctly warping the other sibling correspondences. In effect, the metric updates the system's model to reflect the idea that only a single correspondence can be correct when dealing with one-t-many feature pairings.

This extension works particularly well in cases where many similar feature exists within a given frame (this can occur when the frame includes repetitive textures such as windows on a sky-scraper). Without using the hit metric, RANSAC incorrectly assigns higher weights to transforms where features are warped near the centroid of similar features. This extension alleviates the incorrect weighting by gauging a transform's consistency based on the other feature pairings rather than just using

a simple distance measure. Using such a consistency metric, the extension does not penalize a transform which warps a feature onto a single similar feature on the opposite frame even though many similar features exists in the frame.

5.8 Probabilistic "Good Choice" RANSAC

The basic RANSAC chooses the sample correspondences at random. However, each correspondence is assigned a weight inversely proportional to the distance between the feature pair. Using this weighting, this extension makes the RANSAC algorithm biased towards choosing correspondences with high weights (or low distances). The bias is achieved by having the algorithm discard samples with a certain probability tightly coupled to the sample's weight. In addition, the probability of discarding a sample decreases proportionally to the cumulative weight of those samples already discarded; too much time should not be spent discarding samples simply because they have low weights assigned to them. This generates sample sets which are more likely to include correspondences for similar portions of the image pair rather than correspondences with features that are within a certain distance from each other, and hence a correspondence was created between them, but do not pertain to similar portions of the image pairs.

This extension did not produce significant improvement in the linkages created by the system. Most high-weight sampled were clustered around a particularly interesting and feature-rich object that appears in both frames. The extension frequently chose to use most of the samples from said object, forcing the sample points to be close together. In general, the more spread out the samples in the frame, the better a transform since a larger portion of the frame's image is used to generate the warping. The forced locality of samples decreased the quality of the transforms found by the system.

5.9 RANSAC Spread Trail Sampler

To avoid having nearly co-linear RANSAC samples, this extension to the basic system subdivides an image into six sections, three across and two down. Each section has a border around it. RANSAC samples one point from each of the six sections to get the minimum six samples required for the least-squares affine warp. Since the samples are now spread out among the sections, one each, and each section has a border around it, there is no possibility for choosing co-linear points. Unfortunately, there is no guarantee that at least one correspondence falls within each section. If a particular section does not have any correspondences inside, RANSAC simply chooses a random sample from any section containing points. As long as sample a chosen from at least four sections, the samples cannot be co-linear since the sections spread both the x and y axis. However, if three or less sections contain correspondences, co-linear sample sets can occur, in which case the system generates a bad transform for the link.

This extension worked extremely well. The extension forced the RANSAC procedure to generate warps using a wider sample of the image (the sampled points are spatially spread out, covering more of the image than if they were clustered together around a particular point). In most cases, enough section of the image contained samples that co-linear sample sets were avoided. This extension improved the system's performance without incurring a high cost in speed or complexity; it is one of the most useful extensions to the basic system.

5.10 RANSAC Clustered Feature Filter

The basic system samples from all correspondences within the RANSAC procedure. Some of the correspondences might be outliers caused by noise or bad clustering thresholds. This extension adds a new constraint to correspondences and filters out those which do not satisfy the constraint. A correspondence is tagged as good if neighboring correspondences in the source image are mostly the same as neighboring correspondences in the destination image. This adds a consistency constraint, essen-

tially stating that groups of correspondences should be consistent with each other. Those that fail to meet the constraint are assumed to be erroneous and discarded before running RANSAC on the remaining correspondence set.

The constraint added by this extension decreased the number of erroneous sample points used by RANSAC. It seems that neighboring samples are consistent in frames so that if an object appears in two frames, generally the features around the object are similar between the two frames. In frames with very few features, the extension further reduces the number of samples used by RANSAC because few of the features have neighbors. For video inputs that are not feature rich, this extension should not be used since it tends to limit the number of samples too much.

5.11 Feature-Based Difference Regions in Images

The basic system extracts regions of difference within particular images using a pixel based approach on the difference view. However, difference regions can also be extracted using a feature based approach. A quad-tree is created to recursively find regions of change using the feature set of the image pairs. A feature which appears in one image but does not appear in the second image or appears in a place inconsistent with the affine warp between the images is considered a difference. The quad-tree data structure subdivides the image space into regions of similar features and regions of different features. In general, the system does not utilize the regions of difference within a particular frame and so this extension was not used. The regions found did seem to encompass interesting changes between the frames.

5.12 Multi-Resolution Affine Warps

Objects within a scene can move at different velocities. Also, the perspective transform of the camera results in objects close to the lens appearing to move faster, in image coordinates, than objects moving farther away from the camera. A single affine warp cannot capture sections of the image with different velocities, yet large enough

moving objects can cause significant sections of images to move at different speeds relative to one another. For example, if a particular moving truck takes up half of the image, then the truck half will have a different affine warp than the non-truck half. The basic system tries to find a single affine warp as a transform between images and likely finds a warp which is somewhere in between the two.

The first attempt to implement multiple resolution affine warps for the transforms, meaning that images were composed of several sections each with their own particular affine warp, used a recursive threshold on the percentile of correspondences which agreed with an affine warp. First sort the correspondences according to agreement with the warp. The later half of the correspondences are separated and a warp found for just that half. If the warp for the later half is a good warp and consistent with the correspondences, the system chooses to store both warps (and the regions for each). This process recurses until the latter half of the correspondences does not generate a good, consistent affine warp. The approach proved to not work. The various noisy correspondences almost always fell in the latter half of the sorted list and clouded the true secondary warp, causing the process to stop recursion early or generating affine warps that fit the noise not the true feature pairs.

The working version of multi-resolution warps uses a slightly different approach. Rather than splitting the sorted correspondences and recursing, the image itself is split into four regions and a warp is found for each separate region. Regions without a good, consistent warp stay with the previous warp found at the previous depth (note that at first, a single warp is found for the entire image). If a good and consistent warp is found, the system adds it to the set of warps for the image along with the region it belongs to and applies the procedure again on the region to see if any more warp resolutions can be found within the region.

The procedure to find the multi-resolution warps takes longer and is more computationally intensive than the basic system's procedure for finding a single warp. In most cases, the multi-resolution warps found for a frame link end up being the same as a single warp. The frame images did not contain large enough areas with different enough transforms for the multi-resolution algorithm to decide to use more than a

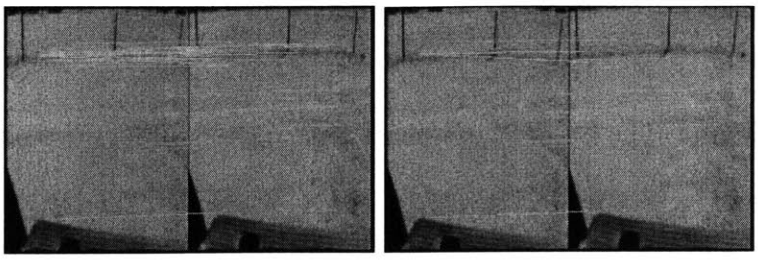
single transform for the entire image.

5.13 RANSAC Reduced-Linkage Form

As mentioned earlier, the correspondence set input to RANSAC is not an ideal one-to-one mapping. In order for the distance metric used in the basic system to work, the system tries to reduce the one-to-many mappings and generate an ideal set of correspondences. The first approach uses a simple greedy strategy: sort the correspondences by their weight, iterate choosing the highest weight, removing all other tied correspondences from the list of possible correspondences. Unfortunately, this greedy strategy can remove many correspondences that do not necessarily have to be removed. Figure 5-2(b) shows a particular setup where the greedy algorithm ends up removing most of the correspondences. What the system needs is to find the maximal matching given the correspondences. This problem, though solvable (often called the maximal bipartite matching problem), requires time to solve even when using the best of algorithms. Rather than solve the problem exactly, the greedy strategy combined with a better weight definition can lead to good results. The effective weight of a correspondence (a,b) is defined as:

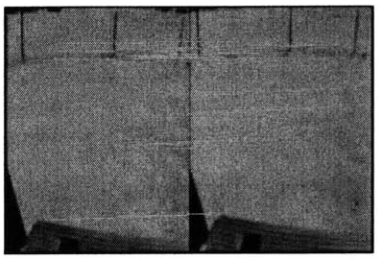
$$\begin{aligned} W_{eff}(a,b) &= w(a,b) - E(a,b) \\ E(a,b) &= max_{x \neq a,b}\{w(a,x), w(x,b)\} \end{aligned}$$

where $w(a,b)$ is the normal weight of the correspondence. In essence the effective weight is extra greedy, looking at both the correspondence's weight and the weight of the best correspondence that will be removed because of choosing it. Figure 5-2(c) shows the effective weight greedy algorithm on a sample set of correspondences. The effective weight prevents the greedy strategy from removing as many links and retains many more correspondences for a denser set that is still one-to-one.



(a) Original Correspondence Set

(b) After Greedy Filtering



(c) After Greedy Filtering with Effective Weight

Figure 5-2: Comparison of greedy filtering with and without the effective weight. The simple greedy filtering removed many more links than the effective weight filtering, links which are consistent with the general warp from one frame to the other. Filtering with the effective weight maintained many more warp-consistent links while still removing low-weight links that are not consistent with the general warp between the two frames.

5.14 Frame Match Verification Methods

As an extension to the basic system, before a permanent link is added to the video table, the frame match is verified using several possible methods. The user decides which methods to apply for verification and the order to apply them in. All chosen methods must verify the match for a permanent link to be added to the graph. Below are all of the different methods as well as a description of their verification approach.

RANSAC Affine: The frame match must have a good affine warp as a transform

RANSAC Multi-scope: The match must have a good multi-resolution affine warp generated using the first method described in the section (the failed method)

RANSAC Adaptive Warp Field: The match must have a good multi-resolution warp generated by the second, effective method described in the section above.

RANSAC Fundamental Matrix: The image pair must generate a consistent and plausible fundamental matrix [10]. The fundamental matrix encodes a measure of how consistent two images of a scene are with each other. A good fundamental matrix means that the view of the two images make sense with each other. The RANSAC procedure is used to find the matrix model.

Region Tagging: Regions of change are extracted using feature-based techniques (described in the extension to the basic system). If the regions of change are small enough (having a total area of less than 500 pixels) the match is verified.

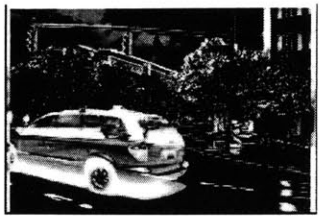
Simple Feature Descriptor Distance: Verify that the average descriptor distance, max descriptor distance, or median descriptor distance of the correspondences between the two frames is below a certain threshold (a distance of 180 worked well for mean and median distance, 240 for max).

5.15 Difference View

The difference view of the basic system is extended in several ways. First, the view is blurred to smooth out noise speckle in the image. Furthermore, the pixel magnitudes are thresholded (a slider in the GUI controls this number) and a binary image is created. The region is thinned using the standard kernel image processing technique. The thinning reduces the regions of change to their backbone shape. The image is then swollen using the standard kernel image processing technique to fill out the backbones of the regions. Regions that are close together are merged by bridging the space between with white pixels. This creates a difference view where the regions of change are filled out and areas where the images are similar are blacked out. Last of all, the blacked out areas can be filled with data from the original images to provide a color context, with regions of change being bordered and containing a gray-scale representation of the difference. Figure 5-3 shows a pair of images, the original difference view, the intermediate steps of the extended difference view, and the result of the new difference view with and without the color context.

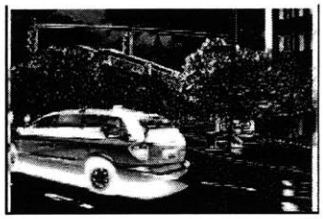
(a) Source Frame

(b) Destination Frame

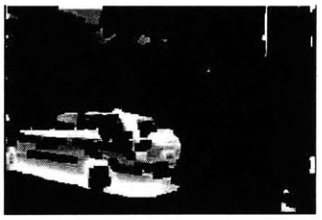


(c) Normal Difference View

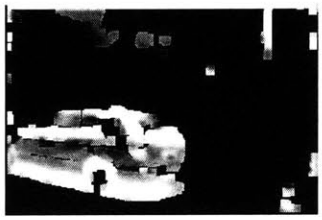
(d) Blurring



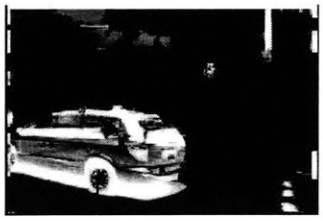
(e) Threshold



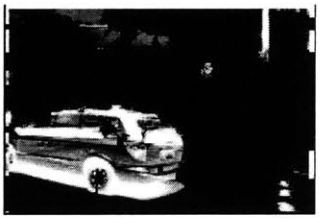
(f) Thinning



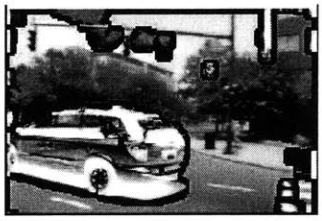
(g) Swelling



(h) Merging



(i) Extended Difference View



(j) Color Context

Figure 5-3: Difference view and intermediate steps.

Chapter 6

Usability Testing

The interface used by the system has gone through several revisions. The final interface includes the results of a usability test. An ex-marine experienced in the video surveillance and field training was given full use of the system. During the two-hour-long session, he tested the system and interface, discussing broken assumptions, ungainly interface choices, and missing system functionality. The session allowed the mental model idealized by the system to be compared to the model used by experienced military personnel in the real world (our intended video analysts and users of the system).

6.1 Improvement to GUI

The results of the usability session included a few changes to the user interface. Previously, the system allowed the video analyst to view primary and secondary videos uncoupled. The system presented the analyst with a view-port for each video and allowed the user to move forwards and backwards within each view. Furthermore, the user could chose to move the sequence in time, consecutive frames shown in order, or in space, moving the video forward a certain number of meters in the real world. The difference between space and time is significant; as an example, say that the user is looking at a video sequence of a car driving down the road. The sequence contains several minutes where the car is motionless waiting for a stop-light to turn

green. Moving the sequence forward in time shows the user several minutes of the motionless car. Moving forward in space, however, allows the stoplight to be shown in a couple of frames followed by the view after the car starts moving again. When showed this functionality, the usability tester concluded that the uncoupling of the videos along with the complexity of the different movements created an un-intuitive and hard-to-use interface. Following his suggestions, the final interface to the system maintains the primary and secondary videos synchronized according to the dense video matching generated by the system. All video motion is relative to time and no space movements are allowed in the final system.

A second aspect of the user interface that underwent revision is the ability to zoom in and out when looking at a particular frame of a video sequence. The previous interface allowed the user to select a particular zoom level and use the mouse to pan the image around. However, the tester repeatedly tried to "select" areas of the image which he wanted to view in more detail by trying to drag the mouse around the area. Following his example, the new system enables the video analyst to click and drag a selection box around a particular image spot. This spot is instantly displayed at a magnified size. Given that all of the views are synchronized, the equivalent spot in the other video and the difference view for it are also magnified in their respective views. Right-clicking on a view resets the zoom level of all the views to the default level. To further improve the zooming capabilities of the system, the final interface allows the user to retain a particular region of a view magnified as the video analyst moves along the video sequence. For example, say that a particular pair of videos have the top half of the frames always be sky. The analyst can select the regions of the frames where interesting things are likely to appear, the bottom half in this case, and instruct the system to keep the views magnified to the selected region as they watch the videos for changes. The video analyst can also set a particular region in the real world which they want to have centered in the views as they scan the video inputs. The system centers the given world region if the current frame can actually see the selected region. Objects in the world, such as a sign, can essentially be tracked by the analysts. The final interface is less cluttered, forcing the views to be synchronized

at all times and using the mouse rather than sliders to magnify interesting regions of the images.

Chapter 7

Results

The implemented system allows a video analyst to focus his or her precious attention to sections of video where changes occur in the input pair. All of the ideas in the previous sections have been implemented and merged into a working prototype. Three different input pairs were analyzed: a side view of an urban environment, a desert side view, and a forward facing view of a country road. The next several sections describe the most important results concerning the video matching and video graph, mainly how the system deals with partial occlusions, along with the input sequences analyzed.

7.1 Dealing With Occlusions

The fundamental structure used by the system is the video graph created out of the input pair. From this graph a tight coupling between the videos is generated. The coupling, or matching, also includes a set of transforms, modeled as affine warps, to change between the coordinate system of one video to the coordinates of the other in a per-frame basis. In order to build the video graph, the system uses SIFT features extracted from the frames of the videos. These features are clustered and similar frames are linked together in the graph. If the system is to be used to detect regions of change, however, it must be able to link frames that are similar but may contain significant sections that differ.

Tunable Parameters

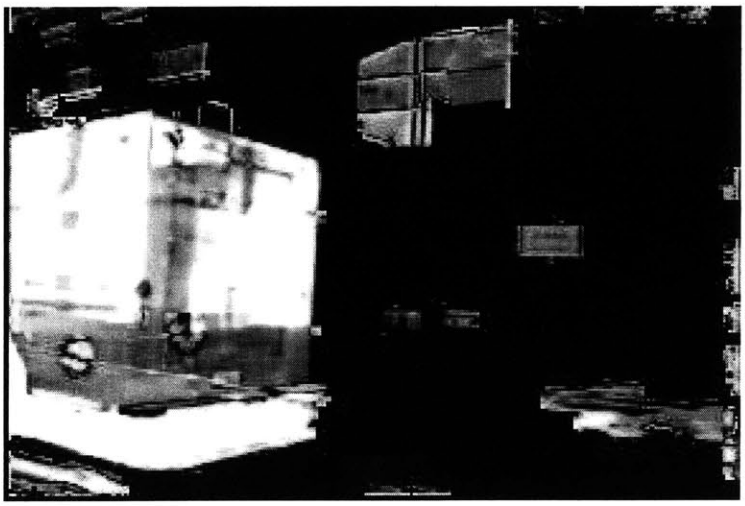
Param	Value	Description	Section
k	50	number of nearest neighbors to cluster in Feature Table	4.1.2
ϵ	180 for feature-rich environments, 220 for low-contrast environments	distance of 128-dimensional used to cluster features in Feature Table	4.1.2
LSH key size	20 bits	bit size of LSH hashtable keys	4.1.4
LSH subtable count	50	number of subtables in LSH structure	4.1.4
PCA dimensionality d	20	number of dimensions to reduce to using PCA	4.1.5
Vote Table β	50 for feature-rich videos, 10 for low-contrast videos	threshold of vote count before frames are linked	4.1.6
Vote Table α	0.15 (15 percent)	percentile of vote fraction for link	4.1.6
RANSAC distance threshold	0.2	"good enough" for RANSAC termination	4.1.8
RANSAC max iterations	40	the max RANSAC iterations	4.1.8
PID K_p gain	1.5	proportional gain for PID	5.1
PID K_d gain	0.25	difference gain for PID	5.1
PID K_i gain	0.2	integration gain for PID	5.1
Insertion queue depth k	1	the recursive depth when using the queued insertion order controller	5.1
Insertion queue region size	5	the number of neighboring nodes to add to queue at every level	5.1
Persistence filtering frame window size n	5	number of frames to check for persistence	5.2
Persistence filtering box size	20 pixels	window, in pixels, of acceptable locations of equal features	5.2
Persistence filtering cluster threshold	160	max distance between equal SIFT features	5.2
Vote percentile drop-off threshold	0.1 (10 percent)	percent of features needed for link in extension	5.4
Warp "Goodness" metric	20 degree max rotation, 100 pixel max translation, 20 percent max relative change in scale	constraints on warps defined as good	5.6
match verification region tag threshold	500 pixel area	area of change tolerated to verify using region tagging method	5.14
match verification feature distance	180 mean/median, 240 max	max distance of features tolerated to verify using descriptor distance	5.14

Table 7.1: Parameters and values used in the system.

The system is "locked" when the video graph produces a correct matching between frames from the primary and the secondary video sequences. The system loses the lock when the video graph contains no links between the videos for certain sections of the input. Sections of video with too few features or drastic changes can cause the system to lose its lock. There are two main questions concerning the state of the lock: how well does the system retain the lock with respect to large changes in the videos? If the system loses the lock for some reason, how efficient is it at regaining the match lock once the disruption in the input has passed? Figure 7-1 shows the system retaining the lock even when a significant portion of a frame is different between the two video inputs. The truck takes up a significant portion of the frame, yet the system still matches the two frames together and generates an informative difference image highlighting the truck as a major change. Figure 7-2, however, shows a situation where the system has not retained the lock for the dense matching. In this particular case, a correct matching would have managed to overlay the background building in the frames on top of each other in the difference image. Clearly the difference image shows that the system failed to find the correct affine warps and failed to detect the true change (just the truck) but rather found several false changes. Partial occlusions, such as that shown in figure 7-1 are effectively dealt with by the current system, yet disruptive cases such as figure 7-2 throw the system off. Currently, the system is intended to be run as a batch process, so there is no way for the user to force certain links to be used by the system. In the worst case, a lost lock will never be regained by the system.

If the disruptive section of video is short (a value relating to the rebuild rate of the feature table), the current system can regain the lock after the section and interpolate the matching within the section of lost frames. Figure 7-3 shows the system recovering from a major disruption in the input video. Figure 7-3(a) shows the links of the video graph between the primary and secondary video frames for an input pair with no disruptions. The x-axis denotes the frames of the primary video, the y-axis represents the frames of the secondary video. In 7-3(b), a section of one of the videos was replaced by a completely different sequence of frames from

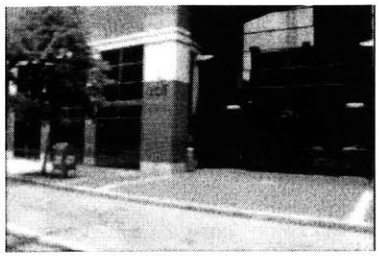
(a) Source Frame (b) Destination Frame



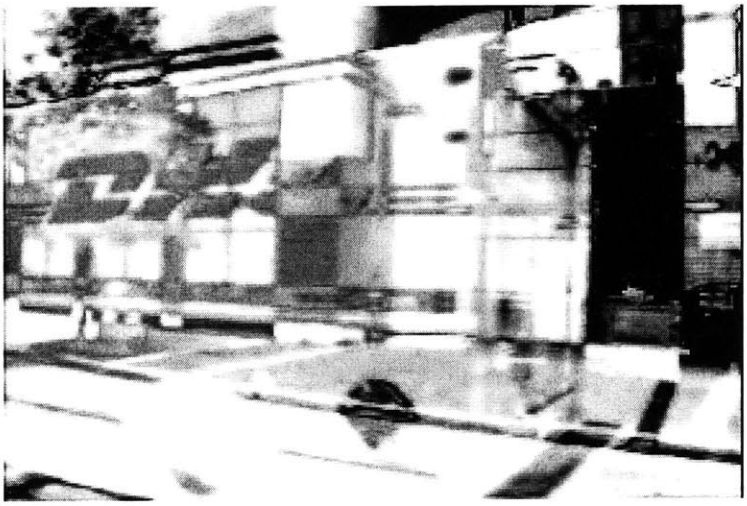
(c) Difference View

Figure 7-1: Partial occlusion example where the system retains lock.

(a) Source Frame



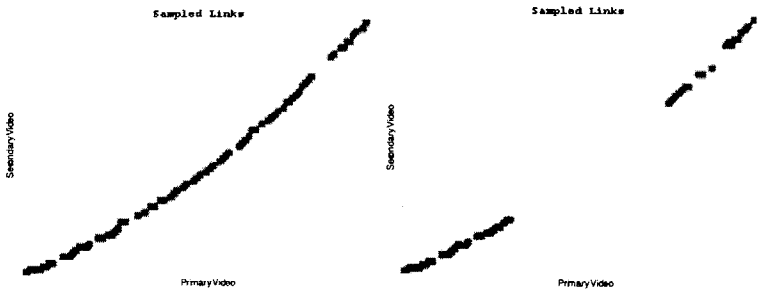
(b) Destination Frame



(c) Difference View

Figure 7-2: Near-Total occlusion example where lock is lost.

a third video source. The graph shows a clear gap in links around the region where the disruption occurs. However, once the original sequence begins again, the system regains the lock and keeps matching the rest of the sequence.



(a) Links For Original Video Pair

(b) Links For Video Pair With Section Replaced

Figure 7-3: Comparison of links when the lock is lost versus not lost while processing a video pair.

The density of features extracted per frame greatly affects the system's ability to deal with partial occlusions and large changes. The higher the density, the more points the system can use to retain the ability to link frames together. For inputs with low contrast, where features are scarce, the system's ability to deal with occlusions is diminished.

7.2 Region Flags

The system allows the user to create regions in real world coordinates where interesting or threatening objects appear in the video input. By combining the GPS data in the audio channel of the video sequences with the ground-plane assumption, the system can map image coordinates to locations on the earth. Figure 7-4 shows a view of the system with five regions. Currently, the user can flag a particular region as either a high-threat region (red), a low threat region (yellow), or a nominal section of video (green). The regions with the small yellow circle around its flag is the currently selected region. The trace of all the frames associated with the region is plotted on

the map along with the size of the region. Flags with borders around them were created and/or edited by the user whereas the single flag without a border is a system generated flag tagging a section of video where the system detected major changes in the input. A video analyst can opt to visit the current flagged regions in order rather than having to scan the entire video sequence. The system generates regions encompassing major changes in the input so that the analyst can skip video sections which are alike and concentrate on those areas where differences occur between the two video inputs.

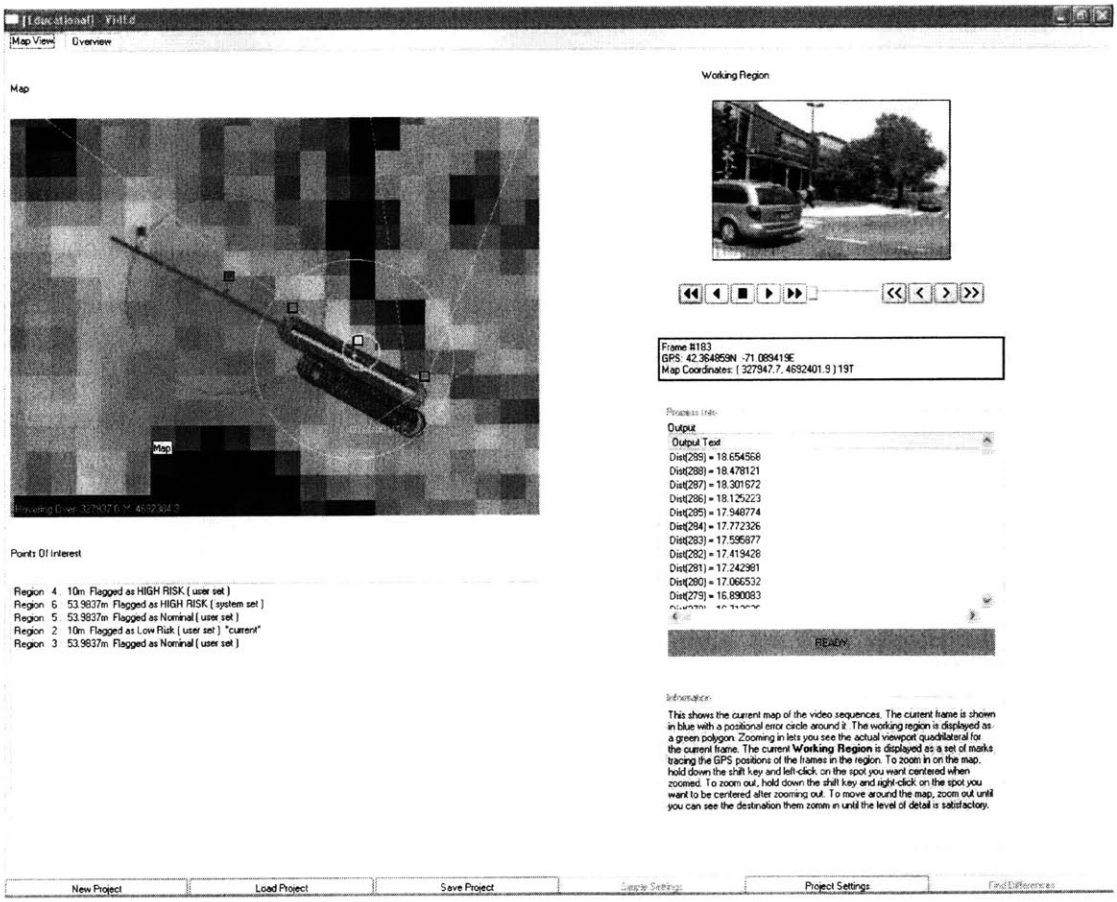


Figure 7-4: Five regions and their GPS traces.

7.3 Video

Along with the screen-shots included in this paper, a pair of demo videos show the system being used to process several video sequences. The videos includes extensive shots of the difference images generated by the system. They also demonstrates the usual way of using the system, from loading the video pair, processing the input to create a video graph, generating a dense matching, and using the interface to analyze the videos for changes, creating regions of interest for further analysis.

7.4 City Sequence

A camera was mounted on the side of a van looking to the right of the vehicle. The van drove down a set of streets in the Cambridge Massachusetts area. The same route was taken four days in a row at different times of the day around a four hour period. The GPS signal measured position at roughly 0.5 hertz. The driver tried to drive as close to the same route as possible (the same roads were taken, and care was given to driving in the same position within the lane). However, all traffic laws were obeyed so street-lights and other cars generated very different input sequences, particularly in respect to the time of certain events.

The urban setting is a feature-rich environment so the SIFT feature detector's thresholds were raised to extract only those features with highest stability and confidence (threshold changed from the default 200 to 180 for feature-rich environment). The feature density was generally around 150-250 features per frame. Since the detector's thresholds were set high, fewer erroneous or noisy features were generated, providing a better basis for the system to construct the dense matching between input videos. However, the urban video sequences all contained sections were most if not all of a particular scene was occluded because of a truck or large vehicle. As such, these input sequences provide good tests for the system's ability to retain a matching in the face of partial occlusions and to regain match lock after disruptive sections of video have been processed.

The matchings generated from the urban input sequences were by far the best out of the three environments tested. The feature-rich environment allowed the feature detector to reject non-optimal features. Any erroneous features were discarded as noise because of the sheer amount of good features in a particular video frame. Even though the urban sequences contained many partial occlusion sections, the system was able to deal with a large majority of them and quickly regained the match lock after failure cases (such as the example in section 7.1). Figures 7-5 and 7-6 show sample difference views generated from the urban sequences. The system clearly identifies sections of change but maintains a good matching using the surrounding areas. Trees provided a challenge to the system because of the huge changes in the features extracted from a tree object with respect to the lighting of a scene. Furthermore, a tree's shadow is commonly mistaken by the system as a difference between input videos; understandable since the system has no notion of a "shadow" as of yet. The system processed a 800 frame input pair (400 frames per video) at roughly 0.43 seconds per frame or 2.33 hertz.

7.5 Desert Sequence

A set of videos were recorded at Ft. Irwin military base to test the system. The sequences contained a GPS signal in the audio tracks updating approximately every two seconds. A video camera taking uncompressed DV video was harnessed in the back of a truck facing the right side. The camera viewpoint was angled downwards towards the regions of interest when trying to find objects near or on the ground. The harness was designed to filter out high-frequency noise, such as the vibration of the engine, by suspending the camera on a platform surrounded by springs and weighted below. Because of the weighting, low-frequency noise was amplified as the damping provided by the springs must fight against the torque of the weight. Several hours of video were taken of the jeep driving around a designated route. Sequences of the route were recorded at various times of the day on different days to try to sample the lighting conditions (ambient light, glare, cast shadows, sun position) for

Figure 7-5: Snapshot of analysis of a pair of video from the city input sequences.

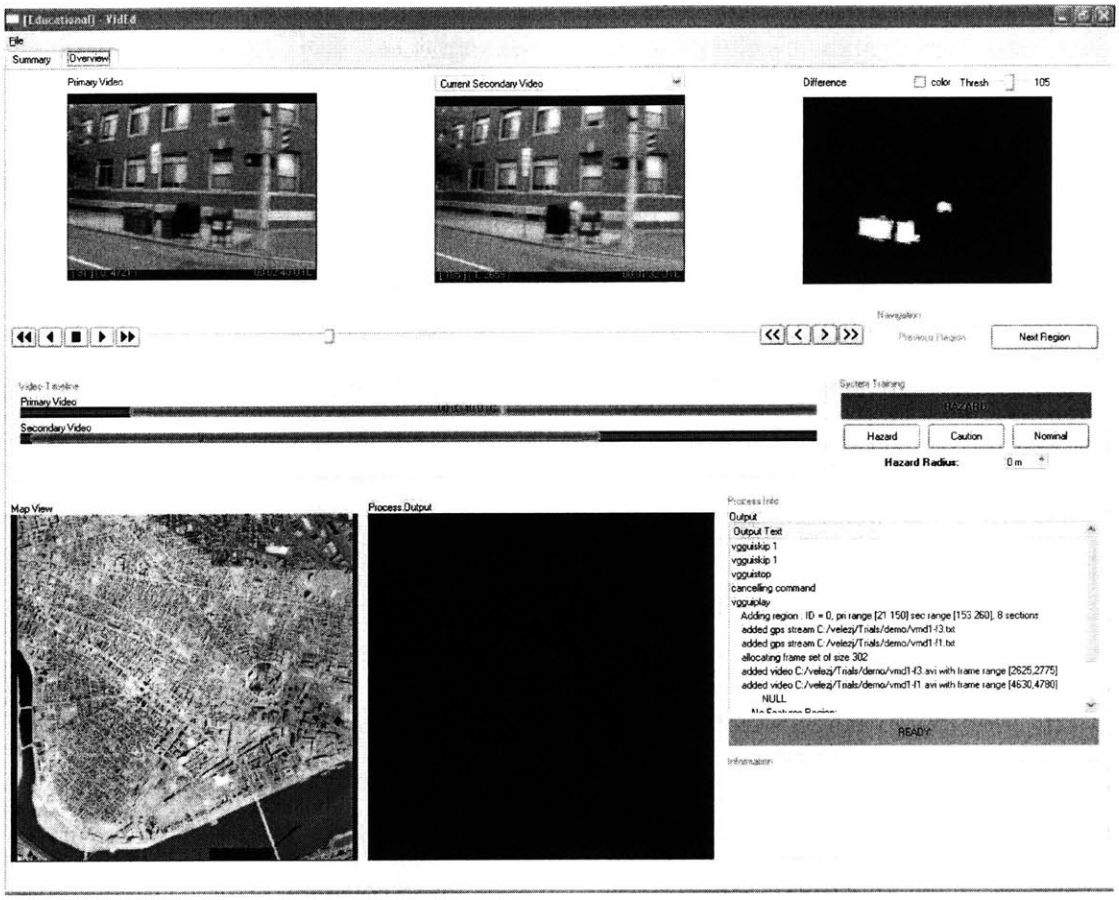


Figure 7-6: Snapshot of analysis of a pair of video from the city input sequences. The garbage cans are clearly shown as differences along with the shirt of the man behind one of the cans.

the environment.

The desert videos provide input sequences with very low contrast. The video sequences contained mostly sand and sky with varying amounts of dry brush in particular areas. The sand in particular supplies very few SIFT features that are easily distinguishable from each other. Tire-tracks in the sand provide nice features but are unlikely to last because of wind erosion and traffic. The reflectivity of the sand also causes problems with glare in some of the sequences; sections of the video have washed-out colors because of the intense light reflecting off the sand.

Because of the low-contrast conditions, the feature detector's thresholds were lowered when processing the desert input set (normally 200, threshold changed to 220). As such, the density of features found increased since the feature detector let through feature with less stability; an average density between 500 - 700 features per frame was recorded while processing the input videos. The large per-frame feature count directly affects the time required to process each frame. The more features, the more time the system must spend clustering the features, the larger the feature table and the slower the queries to the data structure. Lowering the thresholds for the feature detector also increased the number of erroneous or noisy features returned by the detector.

In addition to being low-contrast, the desert sequences also contained the most low-frequency noise between input pairs. The jeep's suspension system as well as the fact that the designated driving route was a dirt road rather than a paved street generated a set of input sequences with lots of low-frequency jitter. Furthermore, bumpy sections of the dirt road resulted in videos where the exact same scene was recorded from several different camera poses rotated in all three axes.

The desert input sequences generated the worst matchings of all of the inputs tested. Figure 7-7 shows a sample of the matching found by the system. The large area of sand contains many similar features, patches of sand which all look alike. The posts are registered as features in the system but are too small for the system to lock onto; the posts each result in 5-10 features whereas the sand generates roughly 400 features. However, the small section of disturbed earth can clearly be recognized as a

difference in the difference view. A human easily dismisses the doubled posts but the disturbed patch of sand clearly is a true change between the input video pair. For an input pair sequence of 800 frames (400 each video), using the lower thresholds for the feature detector and therefore generating 500-700 features per frame, the processing time per frame was roughly 16 seconds; initially the system processed the frames at 2 hertz, but the growth of the feature table and the sheer number of correspondences clustered and queried per frame decreased this to 0.06 hertz.

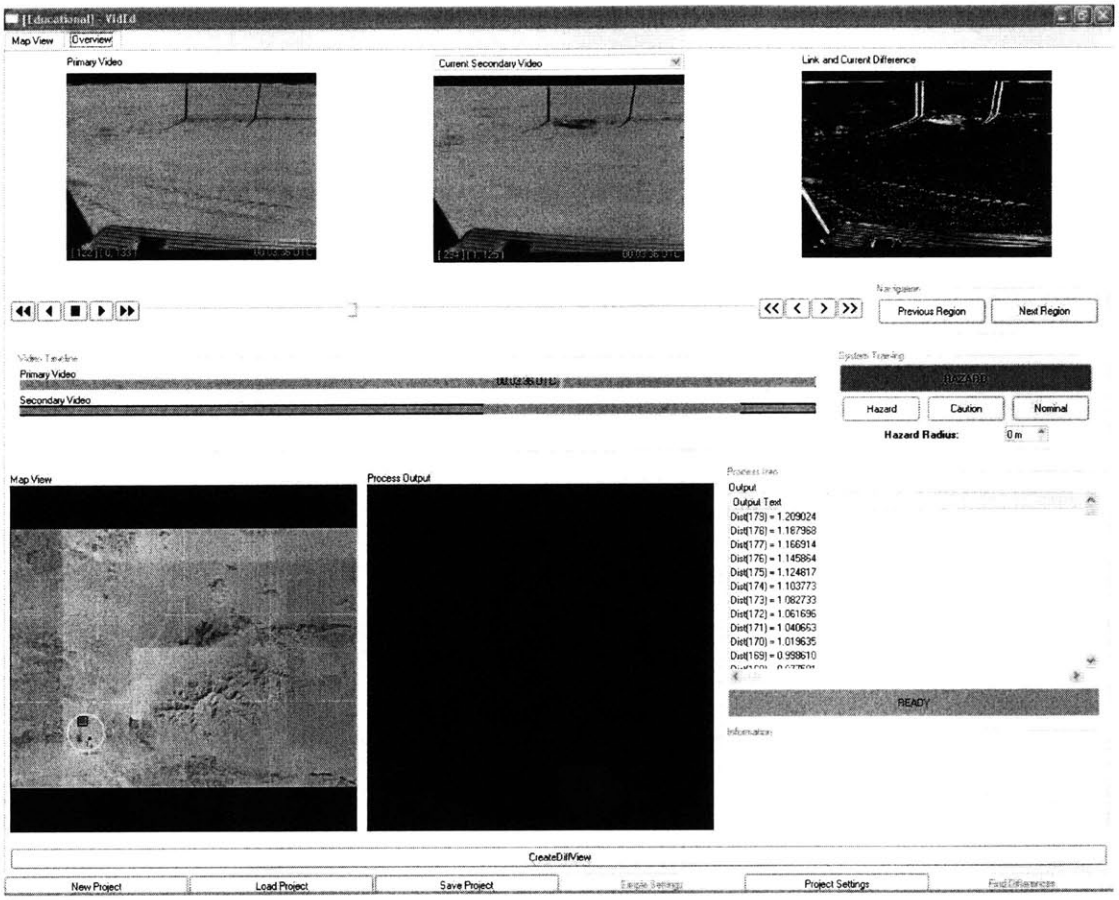


Figure 7-7: Snapshot of desert input sequence analysis

7.6 Forward View Sequence

The last set of inputs tested were of a country road. The camera is mounted on the inside of a car looking out the windshield. The motion for these sequences is

towards the camera, a very different motion path than the other sequences which had sideways left to right video motion. Two short videos were taken of the car driving down the road, around 300 frames each. The inputs are also similar, there are no major differences between the video pair. A particular section of each frame was taken out before processing because of the glare on the windshield; a 30x30 pixel square of each frame contained no data of the scene.

The country scene , while not as feature rich as the urban environment, is not a low-contrast environment. However, the camera used for these trials stored images of roughly half the resolution as the other two inputs sets. 50-70 features were generated per frame. With such a low feature density, the system's processing time decreased compared to the other sequences. However, such low feature count means that occlusions are harder to deal with and could cause problems; the particular video pair tested included no partial occlusions.

The forward facing input sequences generated very consistent matchings. Figure 7-8 shows a screen-shot while processing the forward facing input. As can be seen, the only differences shown are those caused by changes in illumination. Since there existed no real differences between the videos, the system should generate very nice difference views and video frame matching. The most challenging aspect is the high tree count. As stated previously, trees are a challenge because of their shadows and the effects lighting has on the feature representation. Given the reduced feature density, it is not surprising that the forward facing video inputs took the least mount of processing time per frame, averaging at 4 hertz.

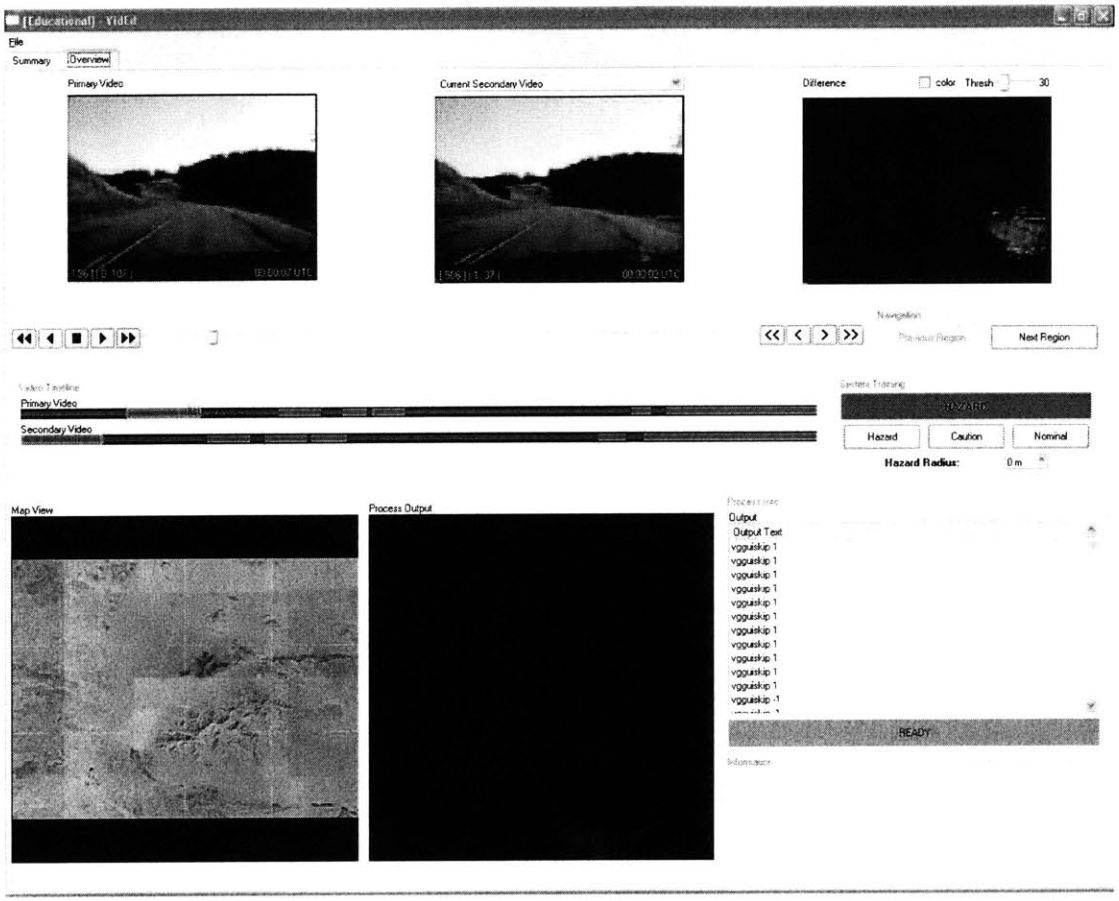


Figure 7-8: Snapshot of analysis of a pair of video from the forward input sequences. Differences shown are caused by illumination changes between videos.

Chapter 8

Failure Modes

The current system has trouble recognizing when an input video has looped, especially when extension 5.3 is used. Shadows pose a large problem for the system, most are treated as differences since no concept of a "shadow" is modeled in the current implementation. Furthermore, large illumination changes between the inputs cause severe problems. Large enough illumination changes disrupt the feature extraction (SIFT) and exacerbate the problems with shadows. Local illumination changes, such as sun glare from metal objects, are handled poorly by the current system (the glare is most often treated as a difference). Lastly, the system has trouble dealing with section of video that have large repetitive textures (for example, a brick wall). The repetitive textures disrupt the transforms found between frames since the texture allows for multiple ways to match two images to each other. In the brick example, it is difficult for the system to tell whether to shift one image to the left or the right in order to have the bricks line up, therefore the transforms generated for linked frames can differ greatly between similar frame pairs.

Chapter 9

Future Work

The current system can be improved by using the difference regions found within each particular frame. The system should be able to pinpoint actual region in the frames which are different and allow the video analyst to closely examine these regions. Furthermore, it would be beneficial if the system was changed from a batch mode to a more interactive unit, where a video analyst can specify known links and give hints to the system as it is processing the video input pair. The system was always meant to work with more than just a pair of video inputs. While dealing with illumination changes and shadows are both open problems, the current implementation does little to model the effects of light on a particular video. The integration of a good illumination model into the system will improve the output quality (difference video and regions) considerably.

Chapter 10

Conclusion

The current system allows video analysts to effectively focus their attention to sections of video where changes occur and interesting objects are likely to appear. A single difference view is generated from a pair of videos using a dense matching between the videos. Because a single video is generated as output, the analyst's time is at least cut in half; rather than scanning two hour-long video sequences , the system allows the user to simply scan a single hour-long sequence containing the information from both video inputs. The difference view highlights changes between the videos in an easy-to-detect manner to further help the user of the system. In addition, the system generates a set of flagged regions in real-world coordinates pertaining to sections of the input where major changes were detected while processing. The video analyst can simply iterate through the regions rather than having to waste time watching countless hours of video with no interesting differences. If the user chooses to watch the entire video, the system synchronizes the input pair so that the user can easily check for changes in the scene. Sections of video with interesting features can be flagged by the user for further analysis; such sections create regions in the world where the analyst believes an interesting event has occurred.

The three input sets tested show that the system generates correct dense matchings of the video pairs in most cases, even in the presence of partial or full occlusions. The viewpoint of the camera and video motion do not seem to greatly affect the system's capabilities as long as the video pairs are continuous and the assumption

that only small changes happen between contiguous frames of the same video holds. Further study is required to see the effect of different viewpoints in the system's video matching ability. The difference view resulting from an input pair highlights changes in the video pairs, guiding the user's focus to areas of interest.

Bibliography

[1] Sgt. 1st Class Doug Sample. Ied conference looks for solutions to save lives. *American Forces Information Service*, 4 May 2005.

[2] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Wu. An optimal algorithm for approximate nearest neighbor searching. In *SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 573–582, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[3] Jon Louis Bentley. K-d trees for semidynamic point sets. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, New York, NY, USA, 1990. ACM Press.

[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.

[5] Defense Mapping Agency, Fairfax, VA. *The Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS)*, 1st edition, September 1989.

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[7] Steve Fainaru John Ward Anderson and Jonoathan Finer. Bigger, stronger home-made bombs now to blame for half of u.s. deaths. *The Washington Post*, 26 October 2005.

[8] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 33–41, New York, NY, USA, 1984. ACM Press.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[10] Q. Luong and O. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *Internaltional Journal of Computer Vision*, 17(1):43–76, 1994.

[11] Gregory M. Nielson. Multivariate smoothing and interpolating splines. *SIAM Journal on Numerical Analysis*, 11(2):435–446, April 1974.

[12] Sarah Victoria Porter. *Video Segmentation and Indexing Using Motion Estimation.* PhD thesis, University of Bristol, February 2004.

[13] W. Ren, M. Singh, and S. Singh. Automated video segmentation, 2001.

[14] Peter Sand and Seth Teller. Video matching. *ACM Trans. Graph.*, 23(3):592–599, 2004.

[15] Gilbert Strang. *Introduccion to Linear Algebra.* Wellesley-Cambridge Press, 2nd edition, 1998.

[16] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. Singular value decomposition and principal component analysis, 2003.

[17] B. Zitova and J. Flusser. Image registration methods: A survey. *Image and Vision Computing*, 11(21):977-1000, 2003.