2007-03-08

Obstacle Annotation by Demonstration

Michael David Clement
Brigham Young University - Provo

OBSTACLE ANNOTATION BY DEMONSTRATION

by

Michael D. Clement

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Masters of Science

Department of Computer Science

Brigham Young University

April 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Michael D. Clement.

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Dan R. Olsen, Chair
Date	Michael A. Goodrich
Date	Michael D. Jones

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Michael D. Clement in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Dan R. Olsen Chair, Graduate Committee

Accepted for the Department

Parris Egbert
Graduate Coordinator

Accepted for the College

Tom Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

OBSTACLE ANNOTATION BY DEMONSTRATION

Michael D. Clement

Department of Computer Science

Masters of Science

By observing human driving with a "digital head" (combined video camera and accelerometers) and taking a few hand annotations, we can automatically annotate regions in a robot's field of view that should be interpreted as obstacles to be avoided. This is accomplished by detecting the movement for a given frame in a video. Some hand annotations of video frames are necessary and they are used to create Probability Grids. Using the movement data and the Probability Grids, it is possible to annotate large amounts of video data quickly in an automated system.

ACKNOWLEDGEMENTS

Thanks to my wife who kept pushing me to get this done, and the two sons who have come since this journey began.

Table of Contents

Table of Figures

Chapter 1 : Introduction

Obstacle detection can be difficult for robots, especially in unfamiliar environments. Robots are being deployed in more and more settings where it is either impractical or dangerous for a human to enter. As the need for teleoperated robots and completely autonomous robots becomes greater, the need for those robots to have increasingly greater awareness of their environments increases as well. To allow a robot teleoperator to become more effective, even these remotely controlled machines must be able to perform semi-autonomous behaviors. These behaviors might include stopping when an obstacle is detected, autonomously navigating around that obstacle, or freely exploring an environment without detailed instructions. This frees the robot teleoperator to focus on high level strategic tasks while the robot handles low level tactical decisions. In addition, a team of robots can allow one person to "be" in many places at once when performing search and rescue or information-gathering missions.

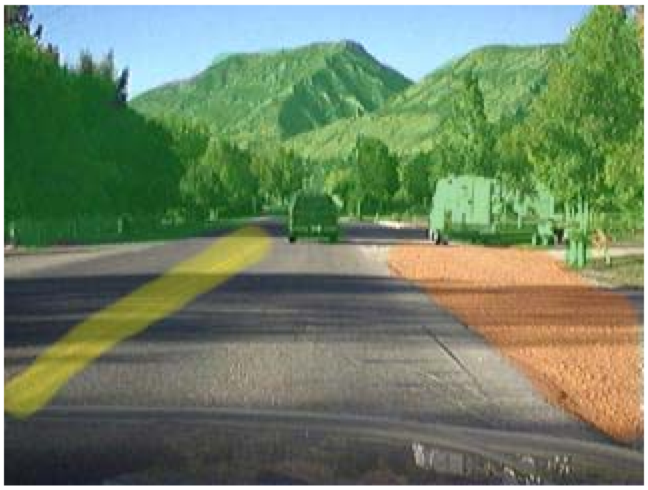


Figure 1 - Different Obstacle Types
Flat (Yellow) – Lane markers; Negative (Orange) – Lower Shoulder; Positive (Green) – Trees, Other Vehicles, Homes

In order to perform these tasks, the robot must first be able to distinguish between obstacles and safe ground. These obstacles, as seen in Figure 1, can be "positive" (they rise above the ground), "negative" (holes or lower ground) or "flat" (painted lane lines on a road or areas off to the side of the road where there is no curb). To cover these three situations adequately, the obstacle detection must include visual obstacle detection although may not be limited to it. This type of vision can be relatively difficult.

Humans have a visual obstacle detection system that works very well even in unfamiliar situations. As we get older we gather years and years of experience that enable us to detect what things around us are obstacles (chairs and tables) and which are ok (a change in flooring material). Using this immense prior knowledge, we are able to determine what areas of our field of view are safe to walk on or need to be avoided. For humans, this knowledge is then applied to driving. All of this experience enables us to know what to look for in our field of view that might be dangerous.

Unfortunately in many current systems, robot teleoperation (control from a remote location) can be unwieldy and require the driver to perform repetitive, attention-hogging tasks. In addition to safeguarding human life, robots should enable a person to focus on high-level tactical decisions to accomplish mission goals while relying on the robots to make low level decisions, such as obstacle detection and avoidance [ARKI91]. Several obstacle detection methods exist including the use of sonar and lasers. By using visual detection of obstacles, the robot not only avoids "positive" obstacles such as boxes and walls, it can safely avoid flat obstacles such as painted lines as well as "negative" obstacles such as holes.

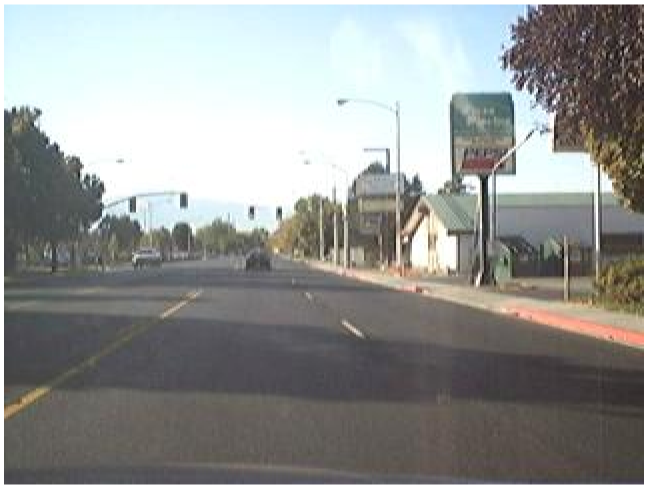


Figure 2 - Example Driving Situation

For example, in Figure 2, it is clear to a human observer that the road ahead is safe to drive on while the shops and trees to either side as well as the other cars on the road present unsafe obstacles. These facts can be expressed by a person in the form of annotations. Annotations map regions of the image to classifications. In Figure 3, the image from Figure 2 has the safe regions annotated in blue and the unsafe regions annotated in red. Since a robot can learn from input data as well as experience, we can take several annotated frames and use them as input into a machine learning algorithm in order to enable a robot to detect safe and unsafe areas on its own.

Unfortunately hand annotating video can be very time consuming. To train a robust classifier that works in a variety of situations, in addition to a good machine learning algorithm, a lot of training data is needed which translates into a need for a lot of annotation. Traditionally, video frames need to be individually annotated by hand in order to create a classifier. Because there may potentially be hours of video, if not tens or hundreds of hours, it is undesirable to annotate every frame (30 frames/sec x 60 sec/min x 60 min/hour = 108,000 frames/hour) or even every few frames. If each frame took only 5 seconds to annotate, it would take 150 hours to hand annotate every frame of an hour of video.

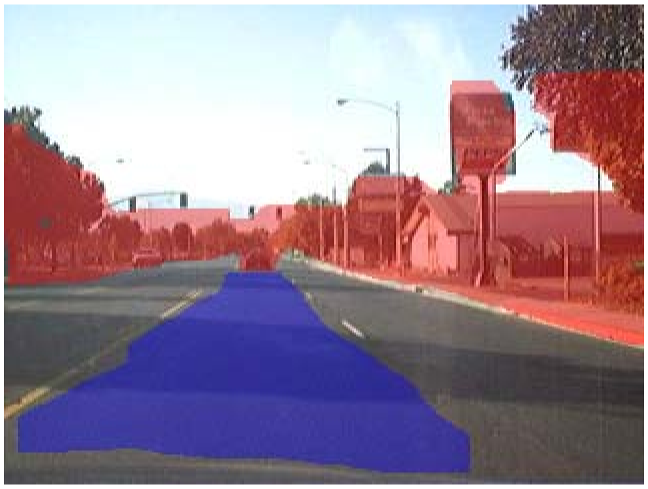


Figure 3 - Hand Annotated Driving Situation
Obstacles are annotated in Red; Safe areas are annotated in Blue; Unknown or unclear areas are left blank (such as other lanes)

The obstacle annotation system developed in this thesis, called DigiHead, allows large amounts of annotated video to be generated from a few hand annotations and the movement (or driving) data. Human drivers typically turn either to navigate or to avoid an obstacle. By observing how a human driver reacts, we can infer which parts of the field of view are obstacles. So instead of taking the data that we get and outputting what the driver should do, we observe what the driver does and infer what in the field of view caused the driver to react that way. By building a system that observes a human's driving habits, a robot has the opportunity to build its own knowledge base that it can use to identify and therefore avoid obstacles.

The DigiHead system provides a method for a human driver to demonstrate to a robot what should be annotated as an obstacle, allowing large amounts of video to be annotated using the actions of the human driver. It classifies regions of the video frames as safe, unsafe or possibly safe. This gives the robot hundreds, if not thousands of hours of prior knowledge by simply driving it in various environments, giving the robot the information that it needs to generalize its obstacle detection ability to a new environment.

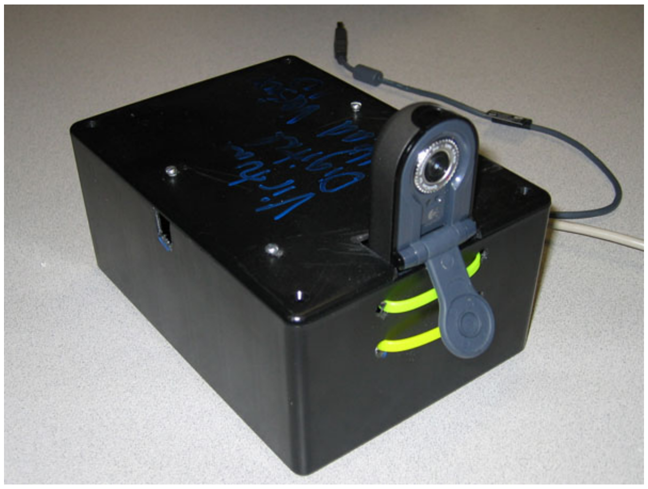


Figure 4 - The Digital Head

The digital head contains a synchronized camera and dual axis accelerometer as seen in Figure 4. These help simulate the human visual and orientation sensors (eyes and inner ears). The digital head records the video frames and accelerometer stream to be analyzed offline. Once we have this data, there are two main processes that take place. First, gathered data is processed to detect the motion of the vehicle and second, the motion information is used to annotate the video.

The captured data enables the DigiHead system to know how the driver reacted at certain points in the recorded data by detecting motion. Looking at Figure 5, it is obvious to a human observer that between the top frame and the bottom frame that the vehicle is turning to the left. The turn in Figure 5 has been hand annotated by using an aqua colored dot in the bottom frame. The solid vertical black line in the earlier frame (top) is used as reference for the center of the frame. The aqua dot in the later frame (bottom) indicates where the center of the earlier frame has moved. As the center has moved to the right, this indicates a left hand turn by the vehicle. But in order to automatically annotate the frames, the system must be able to automatically detect this turning movement.

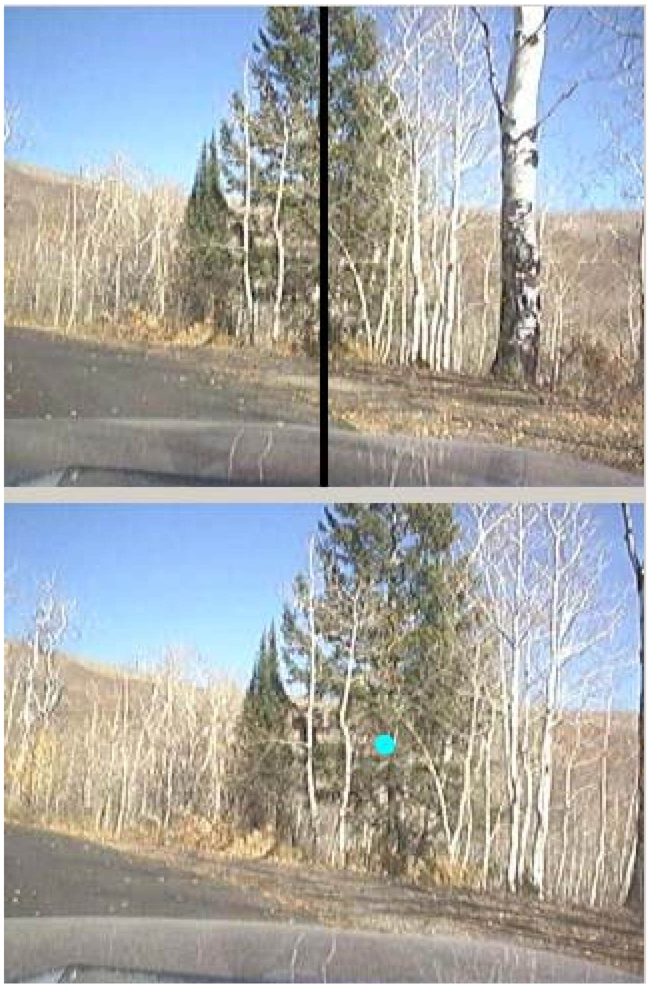


Figure 5 – Turning
Two images 10 frames apart (approx. 1/3 second); The center line on the top frame is used as a reference point to annotate the bottom frame. The bottom frame is annotated with the Aqua dot showing approximately how far the area marked by the center line in the top frame has moved.

With the movement accurately detected, the system must be able to determine which areas should be marked as safe and unsafe based on that movement. To accomplish this, a limited number of frames are hand annotated, as shown in Figure 3, with regions marked as safe (the lane the car is currently in), unsafe (obstacles such as sidewalks, buildings, trees, other cars and pedestrians) or possibly unsafe (other lanes of traffic, especially lanes of oncoming traffic). These annotated frames can be used to probabilistically predict where obstacles might be located in a video frame given what we know about the vehicle's movement using our movement detector. These predictions along with the movement detector can then be used to automatically annotate the remainder of the video. This simple system allows large amounts of driving video to be annotated with safe and unsafe regions quickly and reliably.

Chapter 2 : Related Work

There are several techniques currently used for automatic obstacle detection and avoidance in robot systems. The most common has been to use active sensing tools such as sonar and laser range finders. These sensors emit sound or light to determine the distance to obstacles. There are different methods of interpreting this data including Vector Field histograms [BORE91], [KWON95], Certainty Grids [BORE89], Nearness Diagrams [MING00], and Coastal Navigation [ROY99].

These methods generally focus on allowing the vehicle to build up an internal map-like representation of its environment that enables it to localize itself and then navigate. Some methods even build 3D polygonal representations of their environment [THRU00], [THRU04]. Other methods use multiple robots and merge their maps together [KO03], [KONO03].

One disadvantage of the active sensing techniques is that they emit sound or light waves (possibly invisible to the naked eye but still easily detectable). This poses a problem for covert operations where stealth is absolutely necessary. Another is the fact that generally only "positive" obstacles (obstacles that rise above the level of the ground) are detectable. "Flat" obstacles such as painted lane markers on roads or flat areas that are not roads are generally undetectable to active sensors. "Negative" obstacles such as holes are not detected by most systems although some notable exceptions do so by creating a 3D representation of the environment using lasers [THRU00], [THRU04]. Robots that use active sensors as their primary object detection technique are generally not capable of navigation at high speeds. This is currently due either to limited radius (sonar) or scan speed (laser). These limitations are beginning to be overcome as more advanced sensors are designed.

The alternatives to active sensing techniques are passive ones that include the use of computer vision. The availability of better, cheaper digital cameras and increased processing power has made it practical to do vision and image processing on mobile robots. Vision (sometimes in combination with other sensors) is also attractive because of its ability to sense "negative" and "flat" obstacles. Many visual techniques use multiple cameras and stereo vision techniques to determine distance. Use of stereo vision is inconsistent with the way that humans navigate environments. Given a single flat still image with one eye closed, a person can give a classification of safe areas and obstacles just as well as they could with both eyes in the real world.

Navigating in the real world is not an ability that we are born with. It takes years of experience before we are able to accurately identify obstacles and navigate around them successfully. All these experiences together form our knowledge of the world which then allows us to successfully detect obstacles in our surroundings. Therefore the real challenge to detecting objects more like humans do is to build up an extensive prior knowledge.

Most vision-based approaches rely on pre-programmed knowledge of the environment in order to match the current view to a previously constructed map of the environment [DAVI95], [DAVI98], [LANG99], [LI03], [MURR96], [OHYA98], [ZHAN94]. These generally take advantage of visual "features" in the environment that their respective algorithms can easily extract. The problems with these approaches are that oft times in a new environment a map is not available or the landscape is dynamic. They are also generally limited to indoor environments with well-defined geometries.

An interesting approach by Wu and He involves taking advantage of the geometry of the environment [WU02]. It matches points in its two cameras and assumes a flat driving surface to determine if a point is either closer (a “positive” obstacle) or further (a “negative” obstacle) than what flat should be. Unfortunately this technique ignores “flat” obstacles and it also requires the cameras to be perfectly calibrated, which is not always possible or practical.

Several methods use a combination of visual and sonar sensors, using each for their respective strengths [CHO00], [FOX97], [KATO02], [LI03], [OHYA98]. Some of these “sensor fusion” (using data from multiple sensors) approaches use vision to follow markings (lane lines or other painted guidelines) and use sonar to detect obstacles in their path [CHO00], [KATO02]. Taking advantage of multiple sensors is useful and DigiHead proposed to take advantage of sensor fusion by combining the visual and accelerometer data. Also, recognizing the limits of vision, this project could, in the future, be augmented by other sensors (including active sensors such as sonar and lasers).

2.1 - Foundational Work

The DigiHead project uses algorithms and techniques developed by past and present researchers in the BYU ICE lab as a foundation.

Image Processing with Crayons (Crayons) [FAIL03] allows a user to interactively build vision based object classifiers. Crayons accomplished classifier creation by enabling the user to interactively make annotations on still images and iterate until the classifier did a good enough job. This idea of easy visual annotation using a “crayon” is used by DigiHead for the pieces that require some hand annotation of images.

Turner’s work [TURN04] expanded the Crayons concept for use on video streams and, at the same time, focused its application to obstacle specification in shared control robot systems. Turner’s system allows the user to paint on the video stream to specify safe or unsafe regions within the robot's field of view. Figure 6 shows Turner's system and a sample classification. The middle image is where the user annotates and receives feedback. The right image is for driving commands and the left image is a two dimensional overhead view of the unsafe areas.

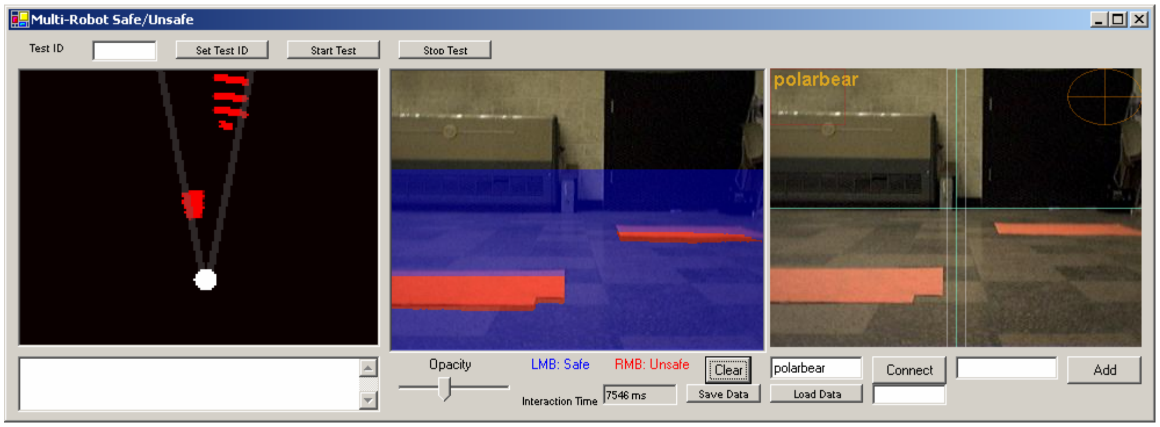


Figure 6 - Turner's System
Allows users to annotate the middle video for safe or unsafe areas (blue safe, red unsafe) and see the resulting classification

The weaknesses of Turner's system include that the robot's prior knowledge is not cumulative and therefore not very broad. The system allows for rapid building of a classifier in a new environment, but each time the environment changes beyond what the classifier can generalize, it becomes disoriented. While this is not a problem unique to Turner's system, it does highlight the need for not only a deep but a broad range of annotated data from which a robust classifier may be created. Amassing a large amount of annotated data will improve generalization because the more knowledge the robot has, the more likely a new situation will be familiar enough that the robot will be able to successfully navigate this new situation without significant user assistance. While Turner can gather and accumulate knowledge, all of the knowledge comes directly from time consuming user annotations.

Arthur developed an algorithm for matching interesting points from one image to the next using Integral Images [ARTH04]. Using these matching points, an overall movement from frame to frame (or every five or ten frames) can be detected. This is done by taking all the points that match in the two frames that is being analyzed. The vector between each point pair is calculated. These vectors can cumulatively give an approximation of the amount of movement, especially turn movement, that has taken place during the two frames.

Chapter 3 : Overview of the DigiHead System

In order to automatically annotate video by demonstration there are three necessary components. First, it is necessary to capture data about vehicle movement and the world around the vehicle. Next the data must be interpreted into usable information about the movement of the vehicle. Finally the movement information needs to be able to determine how a given frame should be annotated. The digital head is used to capture the data, the movement detector interprets the data and the video region annotator annotates the frame based on the other information. The three elements work together to enable the user to annotate large amounts of data with limited hand annotations.

3.1 - Data Collection using the Digital Head

The Digital Head, as seen in Figure 4 and Figure 7, is primarily a recording device. It captures information about the world around it and about its movement through a camera and a dual-axis accelerometer. An application was written so that when connected to a laptop, the video frames and accelerometer stream are synchronized and saved in order to be analyzed later. The video frames that are saved serve a dual purpose in that they help to detect the movement of the digital head in a vehicle and also are the data that is to be annotated.

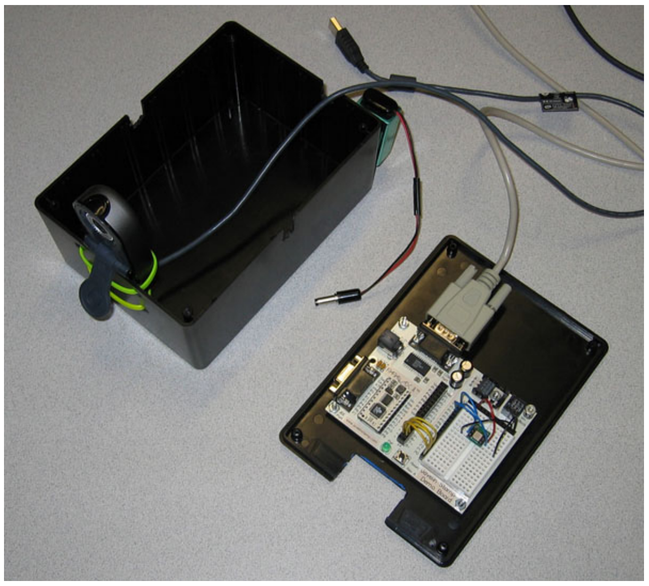


Figure 7 - Interior View of the Digital Head

Video frames are captured from a Logitech QuickCam Pro which costs less than $100. The frames are captured at approximately 30 frames per second and at a resolution of 640 x 480. Accelerometer data is gathered using a small dual axis accelerometer attached to a Javelin Board (developed by Parallax, Inc.). This rapid prototype board runs an embedded version of Java and allows for the data to be captured and transmitted via a serial cable to the laptop.

The data collected consists of a series of video frames stored as JPG images and a file listing the captured accelerometer data and the associated frame. Because of timing issues, sometimes multiple accelerometer data points were gathered for a single frame and occasionally a frame would be missed. To fill those gaps in the data, the data point prior to the current frame is used. This is used instead of an average of surrounding points because in real time at any given point in time the next frame's data is not known.

In order to facilitate gathering data, a removable tripod mount was attached to the bottom of the digital head. The tripod was then set up in the front seat of a 1994 Chevrolet

Cavalier for data collection. Two main types of data were gathered and used in testing. One type is road driving in the Provo, Orem, Lindon and Pleasant Grove areas of Utah County, Utah. This primarily consists of right angle turns on roads that often require stopping before turning or turns that are primarily navigational in nature instead of for obstacle avoidance. The other type is data collected along the Alpine Loop Scenic Highway in Utah County, Utah. This second class of data was collected in order to acquire data with turns that were non-navigational. As seen in Figure 8, this road has many curves in it which allowed data to be gathered from turns forced by a bend in the road instead of navigational necessity. Additionally turns along the scenic highway don't usually require a stop before proceeding.

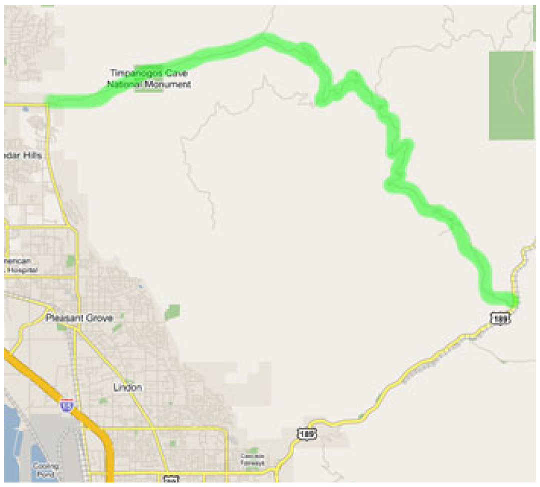


Figure 8 - Alpine Loop Scenic Highway Highlighted in Green (Map from Google Maps http://maps.google.com)

3.2 - Movement Detector

In order to analyze and interpret the data, a second application was written in order to view, annotate and automatically interpret the data. This was initially developed to analyze the collected data for movement. It was later augmented to include other tools and annotation abilities which are discussed later.

Several techniques were tested to detect the movement. The first method uses the accelerometer stream. The accelerometer data stream is simply a series of numbers where each is associated with a video frame. These numbers are the values that the accelerometer outputs which represent the amount of acceleration detected to the right or left. To detect the movement with the accelerometer stream various time-based features were computed from the stream. These time-based features are then fed into two different machine learning algorithms with associated hand annotations. The output of these algorithms is the amount of movement that the algorithm infers from the data for a given frame.

A vision based approach was also tested to implement movement detection. Arthur's algorithm for point matching is fed two images and returns a series of matching points. Using these point pairs, an overall movement from frame to frame can be determined as shown in Figure 9. With this method no training data is needed which eliminates the need for any user input for movement detection.

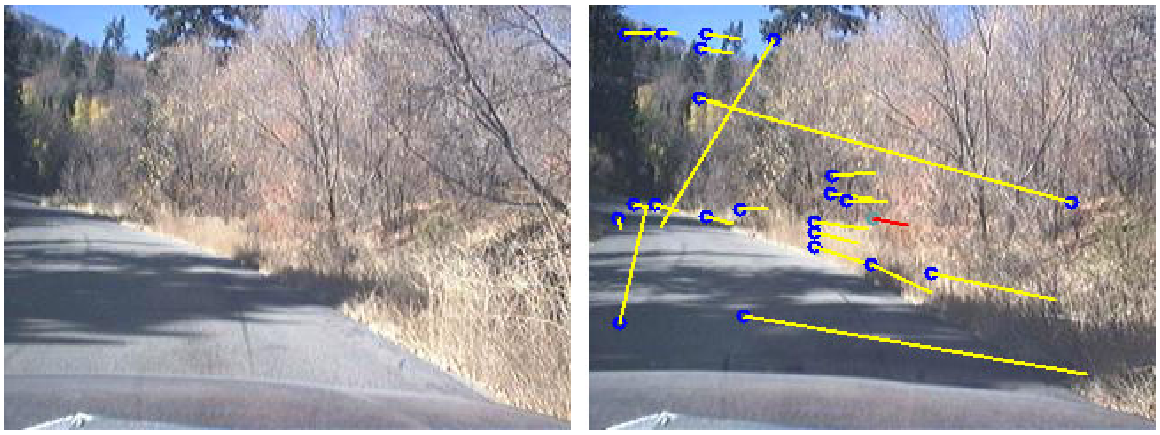


Figure 9 - Arthur's Point Matching with Overall Movement
Blue circles are the points found on the image, the yellow vectors point towards and terminate where the corresponding point is on the first image (left); The green circle is the center of the image with the red line indicating the overall movement

To account for natural inaccuracies in the data as well as to accommodate the discrete nature of the Decision Tree algorithm, pixel counts are converted into discrete ranges of pixel movement. For example, any movement that translates into less than 10 pixels to either side of the center line could be interpreted as "Center" or no turning (Figure 10). This allows for some noise in the data as well as for the natural bumps and small corrections that are made during road driving.

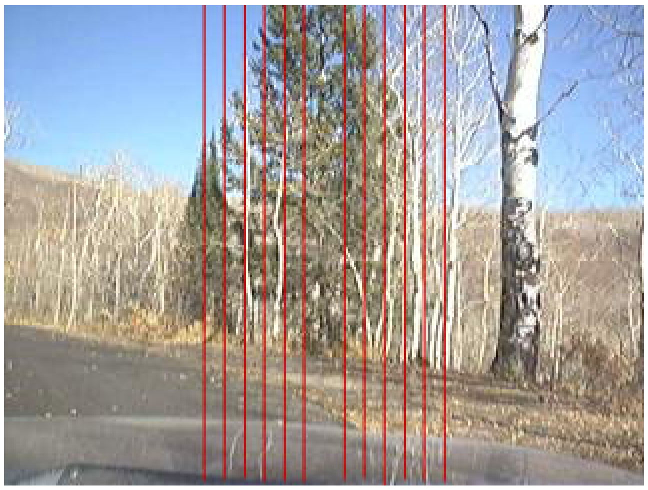


Figure 10 - Discrete Ranges of Pixels
These provide a discrete measurement and help reduce the effect of noise

3.3 - Video Region Annotator

Once a reliable movement detector was created, the next step is to determine which areas of the frames are safe or unsafe based on the detected movement. Unfortunately the only way to determine this for a particular frame is to annotate them by hand as shown earlier in Figure 3. The annotation application was augmented in order to allow for Crayon-style hand annotation of the video. The annotations are "averaged" together depending on how much movement is detected, taking into account the parts annotated as safe and unsafe as well as the areas that are not marked. This allows the DigiHead system to probabilistically predict where obstacles might be located in a video frame given what we know about the vehicle's movement using our movement detector.

The best movement detector along with the probabilistic predictions can then be used to annotate the entire video. This is accomplished by detecting the movement at each frame (or regular intervals) and using the turn type that was detected to look up which prediction should be applied to the frame.

Chapter 4 : Movement Detector

One of the key elements of the DigiHead Annotation System is the Movement Detector. This is used to determine what action a human driver takes during the course of a recorded video so that the proper probabilistic prediction can be applied to the video frames. Determining what a human driver does is important because, in general, human drivers intentionally turn for one of two reasons: navigation or obstacle avoidance.

In order to accomplish movement detection, the idea of movement must be defined precisely. This definition provides a common movement metric across all tests and includes the use of discrete ranges of pixel movement called Discrete Movement Classes to quantize the movement. Some detection methods require discrete training data and/or produce discrete results while others use continuous data. The Discrete Movement Classes allow different algorithms with discrete and continuous output to be compared for accuracy. They also filter out a certain amount of noise as we are primarily concerned with intentional turns taken by the human driver.

Two classes of methods were implemented and compared in order to find an accurate movement detection algorithm. One class of methods uses accelerometer data in order to determine when movement occurred. The other class uses vision techniques. This experimentation was facilitated by the creation of an application to view and hand annotate the combined video and accelerometer data.

4.1 - Movement Defined

In order to be consistent across tests, the amount of movement that has occurred for a given frame needs to be defined precisely. Movement for a particular frame is defined as the number of pixels moved between the current frame and 10 frames previous (approximately 1/3 second). Enough change was noticeable in 10 frames to be useful in detecting movement while also short enough such that the entire view did not change during a typical turn. It is important to have a sizeable enough change so that noise (small course corrections on a straight road or shaking caused by bumps in the road) in the data does not interfere with the ability to detect intentional turns. It is equally important to have landmarks in both the current frame and the reference frame (10 frames earlier) in order to detect movement visually. Additionally, the shadows between 10 frames don't usually alter significantly, therefore minimizing lighting as an issue. Shadows can, in some cases, be used as references for both automated and manual annotation.

Having a sufficient but not overwhelming frame difference is important to being able to detect the movement visually using an automated system. It also makes it easy to identify the turns manually for hand annotation. The hand annotated video is used as training data for the Movement Detector and as a standard for evaluation of the machine movement detection. An example of two images that are 10 frames apart during a left turn is shown in Figure 11. A hand annotation is shown by the aqua dot in the right frame which indicates where in that frame the area in the middle of the previous frame (left) can now be seen.

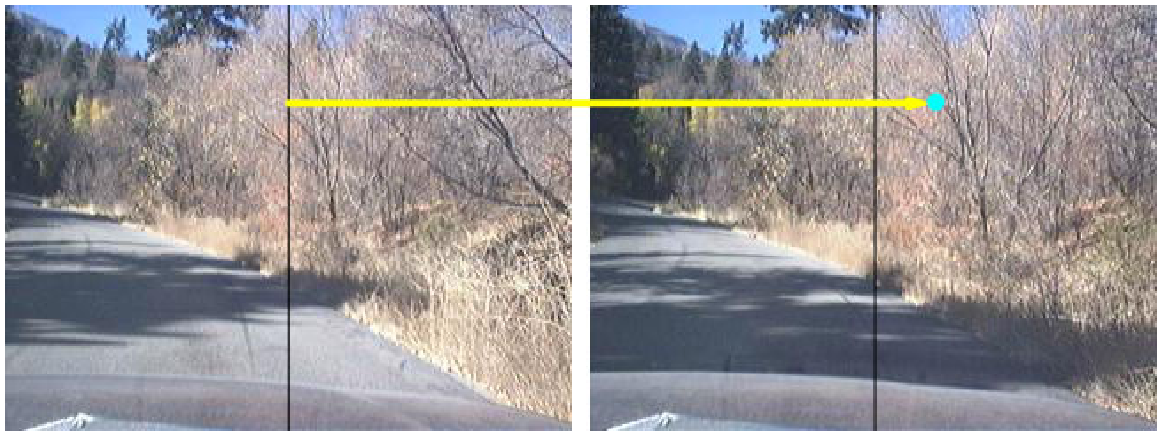


Figure 11 - Hand Annotation of two images 10 frames apart
Black lines are the middle of the image; The area in the middle of the left frame (10 frames previous) is marked with an Aqua dot in the right frame (current frame); The yellow arrow indicates the movement of corresponding area from frame to frame

4.2 - Discrete Movement Classes

In order to facilitate both the Decision Tree algorithm used and the other algorithms, pixel movement values were mapped to Discrete Movement Classes. These are basically discrete ranges of pixel movement. This also allows us to filter out a certain amount of noise in the data due to bumps in the road and small corrections made during road driving. An example of how a frame might be divided into these movement classes is shown in Figure 12.

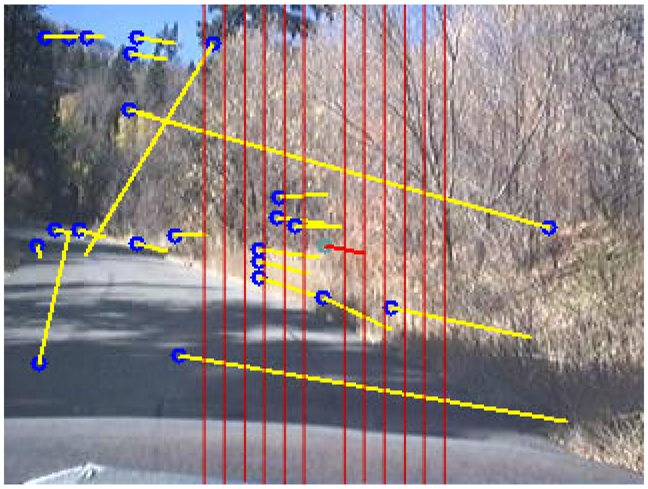


Figure 12 – Discrete Movement Classes Example Division
The band down the middle would map to "Center" with each of the side bands mapping to Right *X* or Left *X* (where *X* is the number of bands in either direction)
This particular movement would be classified as Left 1

In the example shown in Figure 12, the short bright red horizontal line in the center of the image shows the movement calculated for this image (based on a comparison with the image 10 frames previous). Since the green circle indicates the current center of the image and the other end of the red line indicates the center of the image 10 frames previous, the movement detected is actually about 19 pixels to the left. The dark red vertical lines show possible divisions for Discrete Movement Classes. In this case, the center class is 20 pixels wide and each of the side classes is 10 pixels wide (except for Left 6 and Right 6 which include all the area to the far right and far left).

Using the Discrete Movement Classes for movement detection and evaluation of detection techniques allows for algorithms with discrete and continuous results to be compared. By discarding slight variations in horizontal position, this also filters out most unintentional movement and therefore focuses the results on intentional movement by the human driver.

4.3 - Movement Detection using Accelerometer Data and Machine Learning

Detecting movement with accelerometer data and machine learning requires training data. This training data composed of hand annotations (classes) and accelerometer data (features), train a machine learning algorithm to detect movement based on an arbitrary accelerometer stream. Various time-based features must be computed on the accelerometer stream. The hand annotations of the video provide the machine learning class based on how much movement is detected visually by a human annotator. With an adequately trained machine learning approach, a given frame and the preceding accelerometer stream can be processed by the machine learning algorithm to calculate the movement between the current frame and 10 frames previous.

4.3.1 - Movement Hand Annotation

In order to define movement at a given frame, a hand annotation must be captured and recorded. An application was built to facilitate hand annotating the video. The application shown in Figure 13 displays the current video frame and the image 10 frames previous with a line down the vertical center of the image. The current frame can then be annotated with a point showing where the center of the previous image has moved to in the current image. These hand annotations are used as a raw pixel count from the center or are converted to Discrete Movement Classes depending on the machine learning algorithm used.

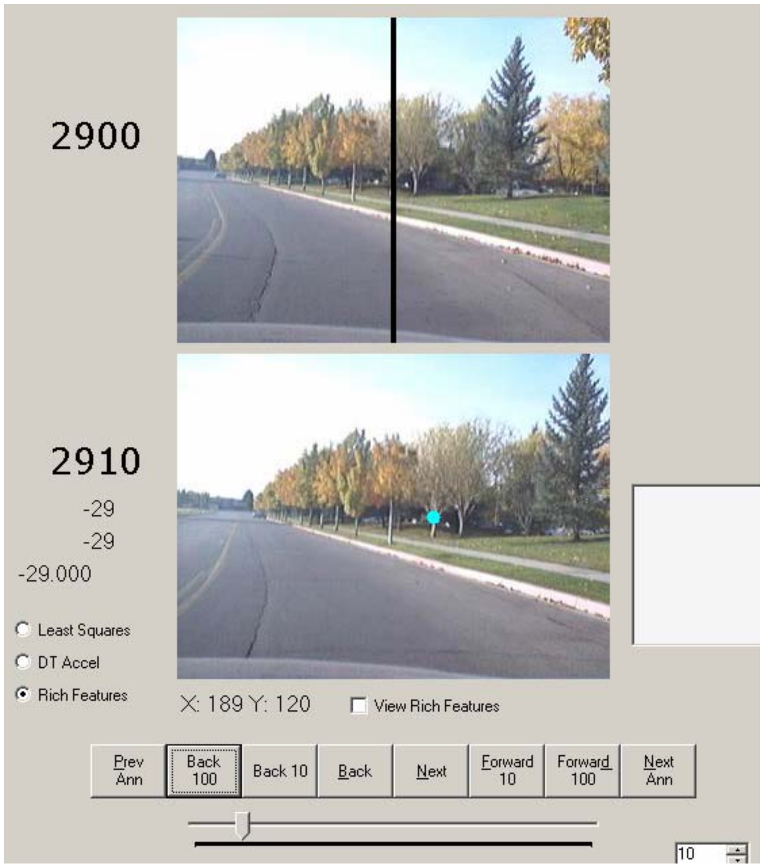


Figure 13 - Screenshot of the movement annotation application
Bottom frame is annotated indicating where the center line moved to during the 10 frames between the frames.

4.3.2 - Acceleration

Acceleration, defined as any change in an object's velocity, should be useful in creation of a movement detector. Since velocity includes both speed and direction, any change in either component indicates acceleration which can be detected by an accelerometer. Centripetal acceleration, the "center-seeking" acceleration found in turns, can be detected using an accelerometer that is oriented perpendicular to the vehicle's movement and parallel to the plane the vehicle is traveling on. An example of an accelerometer stream captured while in the middle of a turn is shown in Figure 14. As can be seen, the accelerometer detected acceleration during the turn. Note that the green data points are above the center "0 acceleration" line. Points below the 0 line would indicate a turn in the opposite direction.

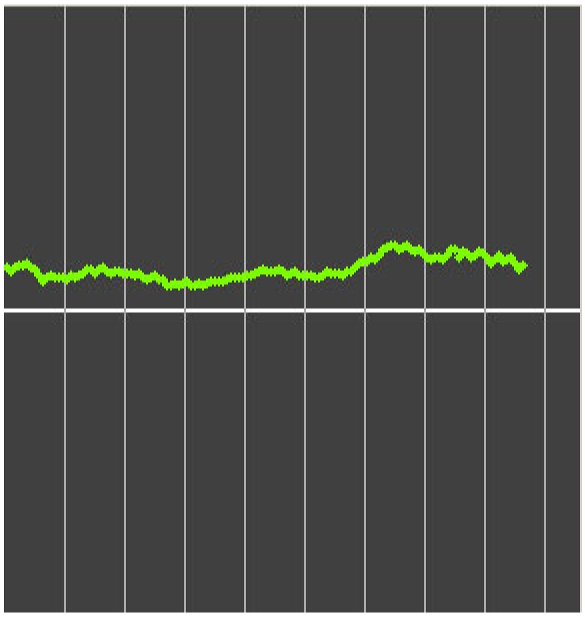


Figure 14 - Accelerometer Data recorded during the turn shown in Figure 11
Accelerometer data is green and the baseline (no acceleration) is white

4.4 - Using Machine Learning

The movement hand annotations and calculated feature set are fed into two machine learning algorithms and tested against other hand annotated frames. These two methods were selected for their ability to naturally select the most distinguishing features and use those to classify new data.

The first algorithm tested is Least Squares Approximation. This method takes a mathematical approach, creating an *n* degree polynomial, where *n* is the number of features, that best approximates the data given to it. This approach requires a lot of annotated frames. Decision Tree is the other machine learning technique used to compute what the discrete movement class should be based on the features of the accelerometer stream.

4.4.1 - Feature Extraction using Temporal Integral Data Streams (TIDS)

In order to efficiently extract features from the accelerometer stream, the data is first inserted into a Temporal Integral Data Stream (TIDS). TIDS mimics the functionality of an Integral Image [VIOL01] but on a stream of data over time. The example found in Figure 15 shows how a Temporal Integral Data Stream would be constructed from the fictitious data stream in the "Data" row. Each entry in the TIDS is the sum of the current value in the data stream and all the previous data in that block (Figure 16). The data is chunked into blocks (usually much larger than shown in the example) to prevent overflow in the TIDS entries which is possible with long data streams or streams with large values. The block size is configurable to allow for various types and sizes of data streams.

Time	0	1	2	3	4	5	6	7	8	9	10	11	...
Data	10	12	9	5	11	15	20	23	20	18	19	18	...
TIDS	10	22	31	36	47	62	82	105	20	38	57	75	...
								First Block				Second Block	

Figure 15 - Temporal Integral Data Stream Example
The TIDS row is the sum of the current data in the Data row and all the previous data in that block

$$TIDS_a = \sum_{i=j}^{a} x_i \text{ where } j = \operatorname{int}\left(\frac{a}{n}\right) \times n$$

a=element index and *n*=block size

Example:

$$TIDS_5 = \sum_{i=1}^{5} x_i = x_1 + x_2 + x_3 + x_4 + x_5$$
$$47 = 10 + 12 + 9 + 5 + 11$$

Figure 16 - TIDS Value Calculation
Example from the entry at time 4 in Figure 15

The TIDS allows for constant time extraction of summation features from the data stream. Taking the difference of any two values in the TIDS returns the sum of all the values between those two data points. Figure 17 and Figure 18 show examples of how to use a TIDS using the sample in Figure 15. Some examples of features that are extracted are the sum of the past 30 data points and the sum of the data from 30 frames prior until 15 frames prior. Both of these features can be extracted from the TIDS in constant time. Additional features were used including the square of the sum of the past 30 data points and difference between the average of the past 20 data points and the average of the last 10 data points. Figure 19 lists all 74 features extracted.

$$\sum_{i=j}^{a} x_i = TIDS_a - TIDS_{j-1}$$

$$\sum_{i=4}^{6} x_i = TIDS_6 - TIDS_3 = 82 - 36 = 46$$

and

$$\sum_{i=4}^{6} x_i = x_4 + x_5 + x_6 = 11 + 15 + 20 = 46$$

Figure 17 – Simple usage of TIDS
A simple usage from Figure 15 where the start and end values are within the same block

$$\sum_{i=j}^{a} x_i = TIDS_a + TIDS_b - TIDS_{j-1}$$

where *b* is the last entry in the block containing *a*

$$\sum_{i=6}^{10} x_i = TIDS_{10} + TIDS_7 - TIDS_5 = 57 + 105 - 62 = 100$$

and

$$\sum_{i=6}^{10} x_i = x_6 + x_7 + x_8 + x_9 + x_{10} = 20 + 23 + 20 + 18 + 19 = 100$$

Figure 18 – Cross-block usage of TIDS
A single cross-block usage from Figure 15 where the start and end values are in different consecutive blocks

back 45, length 45	back 20, length 5
back 10, length 5	back 18, length 5
back 15, length 5	back 15, length 15
(back 5, length 5) squared	back 18, length 18
(back 10, length 5) squared	back 14, length 14
(back 15, length 5) squared	(back 15, length 15) squared
(back 5, length 5) cubed	back 42, length 42
(back 10, length 5) cubed	back 44, length 44
(back 15, length 5) cubed	back 25, length 10
back 20, length 20	back 22, length 22
back 25, length 25	(back 8, length 8) squared
back 30, length 30	(back 12, length 12) squared
back 15, length 13	(back 20, length 10) squared
back 20, length 10	(back 4, length 4) squared
back 30, length 15	(back 10, length 8) squared
(back 45, length 45) /5	(back 15, length 10) squared
(back 10, length 5) /10	(back 20, length 5) squared
back 48, length 48	back 30, length 25
back 43, length 43	back 50, length 50
back 17, length 17	back 40, length 40
back 12, length 12	back 10, length 10
back 12, length 7	(back 30, length 25) squared
back 15, length 10	(back 20, length 20) squared
back 46, length 46	(back 30, length 30) squared
back 48, length 44	(back 30, length 20)/20 - (back 25, length 15)/15
(back 20, length 20)/20 - (back 10, length 10)/10	(back 30, length 15)/15 - (back 25, length 10)/10
(back 30, length 30)/30 - (back 20, length 20)/20	(back 30, length 20)/20 - (back 20, length 10)/10
(back 30, length 30)/30 - (back 10, length 10)/10	(back 50, length 50)/50 - (back 25, length 25)/25
(back 35, length 15)/15 - (back 25, length 5)/5	(back 50, length 50)/50 - (back 25, length 20)/20
(back 15, length 15)/15 - (back 10, length 10)/10	(back 50, length 25)/25 - (back 25, length 25)/25
(back 25, length 25)/25 - (back 20, length 20)/20	(back 40, length 20)/20 - (back 20, length 20)/20
(back 25, length 25)/25 - (back 15, length 15)/15	(back 50, length 10)/10 - (back 40, length 40)/40
(back 25, length 25)/25 - (back 10, length 10)/10	(back 50, length 30)/30 - (back 30, length 30)/30
(back 45, length 25)/25 - (back 25, length 5)/5	(back 50, length 40)/40 - (back 40, length 40)/40
(back 15, length 10)/10 - (back 10, length 5)/5	(back 50, length 20)/20 - (back 20, length 20)/20
(back 20, length 15)/15 - (back 15, length 10)/10	(back 40, length 30)/30 - (back 30, length 30)/30
(back 20, length 10)/10 - (back 15, length 5)/5	(back 30, length 20)/20 - (back 20, length 20)/20

Figure 19 - List of all 74 Accelerometer Features Used

4.5 - Enhancing the Data Stream

It can be seen in Figure 14 that the accelerometer stream was very shaky, jumping along the curve that represents the centripetal acceleration due to the turn. In order to mitigate the effect of this noise, several methods were tested. First, "wider" features were added such as “back 100, length 50” and “back 200, length 200”. The width of the features level out noise when they are calculated as they sum over larger time intervals. If these features are more useful in differentiating classes, they should be favored by either or both of the machine learning algorithms. This is sometimes the case. In some cases the longer features actually negatively impact the accuracy of the algorithms and were therefore removed from the feature set.

$$smoothedData_m = \sum_{i=m}^{m-5} data_i$$

Figure 20 - Smoothing

Local smoothing was used as another attempt to reduce the impact of noise on the machine learning algorithm. Each accelerometer data point was replaced with the average of that data point and the 10 previous data points as indicated in Figure 20. This resulted in the data stream smoothing out significantly as seen in Figure 21. This resulted in incremental but consistent improvements of the accuracy of the movement detectors that used the accelerometer stream. This enhancement is reflected in the results reported on later.

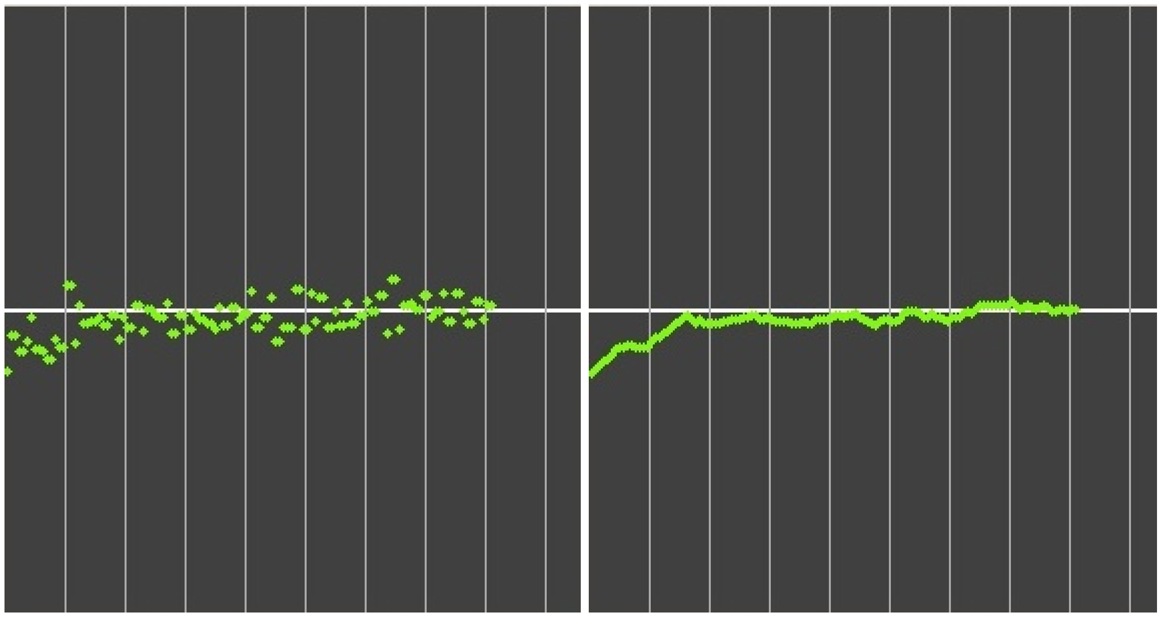


Figure 21 - Unsmoothed and Smoothed Accelerometer Streams
Left is unsmoothed; right is smoothed using algorithm in Figure 21

Unfortunately the accelerometer data did not work out as well as anticipated. Turns are clearly visible in the accelerometer stream due to centripetal acceleration as the vehicle turns. Because of inconsistent acceleration and velocity during turning, all turns that "look" the same did not always "feel" the same. This is demonstrated in Figure 22 where two accelerometer streams are shown that visually represent the Discrete Movement Class Left 5 (same "look") yet the graphs of the streams are significantly dissimilar ("different feel").

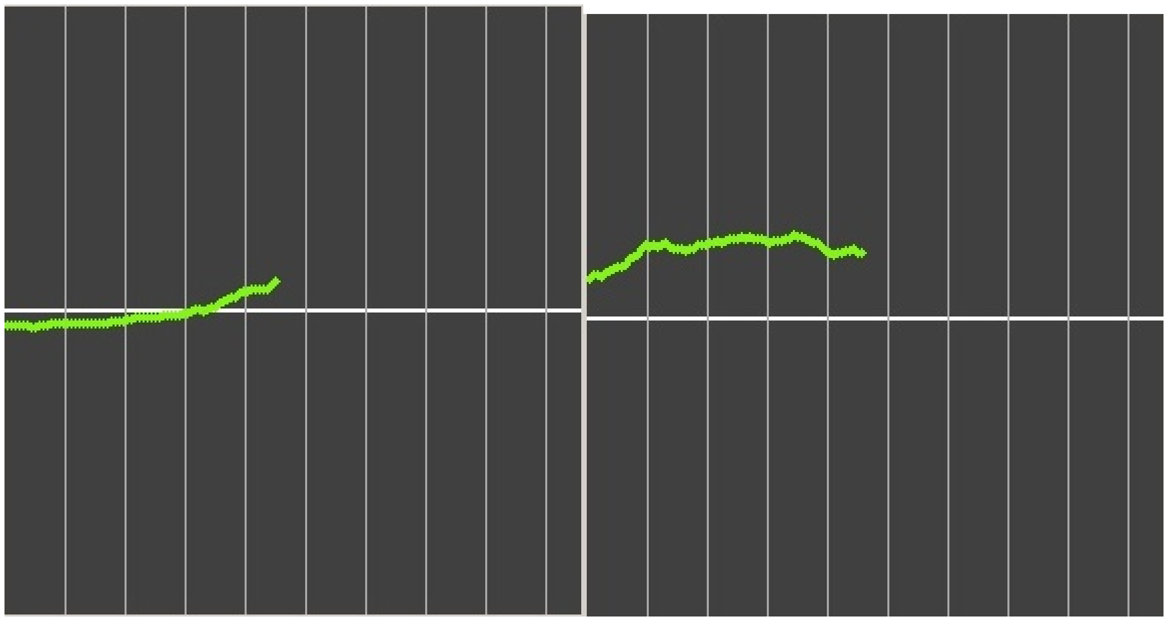


Figure 22 - Look is Different than Feel
Both of the above accelerometer streams are DMC Left 5 (“look”) yet are significantly different in shape and magnitude (“feel”)

4.6 - Movement Detection using Vision

Since the final goal is to annotate an image, it seems to be appropriate to detect turning using a visual approach. This allows for a more direct mapping between the movement detected and the image annotation to be applied later. The other major advantage of this approach is that no training is necessary. Using any two images the movement can be determined without the need for any human annotation or interaction.

To detect the movement visually, DigiHead uses Arthur’s Interesting Point Matching algorithm [ARTH04]. Arthur’s algorithm works by finding “interesting” points based on visually contrasting areas in each of the images. The algorithm mostly finds points along edges, especially corners, as these generally offer high contrast. It takes advantage of Integral Images to efficiently average various sizes of blocks of the image based on color values. Figure 23 shows some examples of interesting points (at the end of the yellow line without the blue circle) found in the right image of Figure 9. Note how the contrast between the sky and the tree (Figure 23 bottom left magnification) creates an area where interesting points can be easily identified.

The points, once identified, are matched by comparing the same features that were used to determine the interestingness of the points. For example, the bottom left magnification shown in Figure 23 shows interesting points with significant amounts of blue to the right and darker colors to the left. Points "matching" that description are then found in the corresponding image from 10 frames prior to the current frame creating point pairs. DigiHead calculates a vector (direction and magnitude) for each point pair which represents how much and in which direction the algorithm guesses that the given interesting point moved.

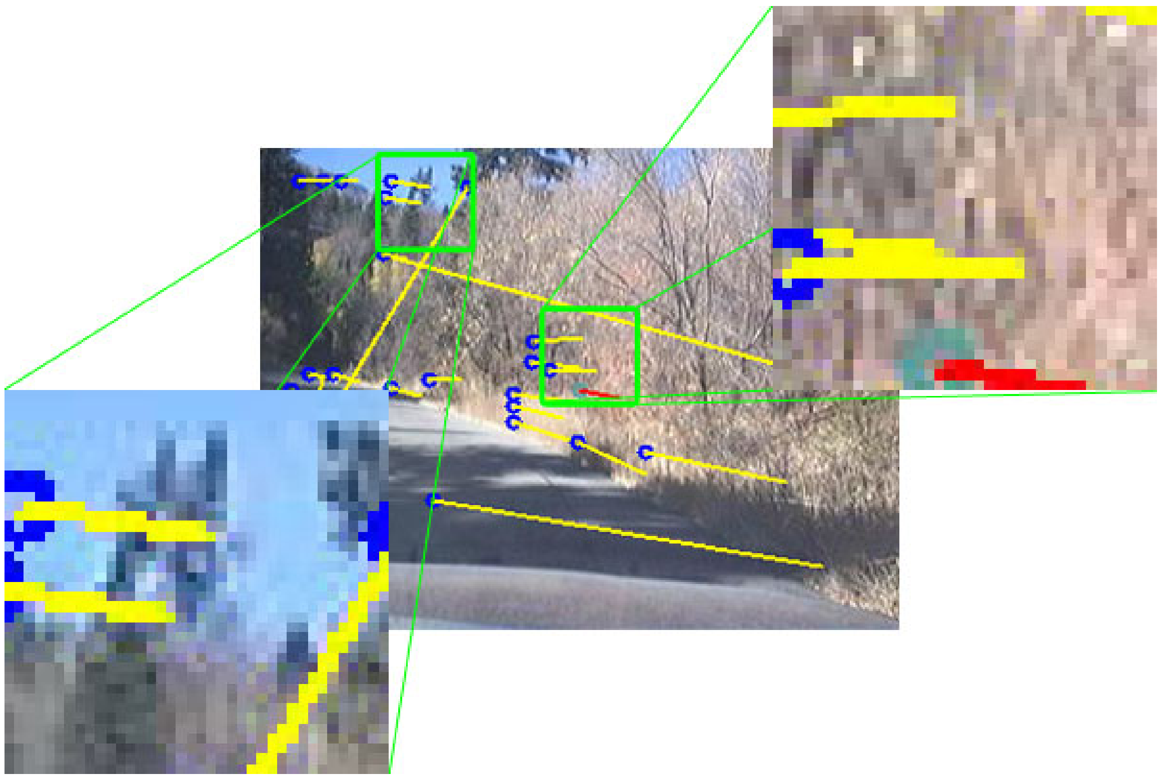


Figure 23 – Magnified examples of "interesting" points from Figure 9
The interesting points are at the end of the yellow lines without the blue circles.

These vectors cumulatively give an approximation of the amount of turn movement that has taken place between the two frames. First the average and standard deviation of both the horizontal and vertical components of the vectors is calculated. In order to eliminate noise caused by mismatched points and other objects moving within the field of view (such as other vehicles), any vectors with either horizontal or vertical components greater than one standard deviation away from the mean are thrown out. The mean of the remaining vectors is taken to determine the direction and magnitude of overall vehicle movement. The horizontal component of the resulting vector is used to guess the amount of turning that has taken place between the two frames. An example of this is found in Figure 24.

<10,1>,<10,-1>,<8,2>,<9,1>,<5,5>,<10,0>,<12,1>,<11,-2>,<10,-1>,<-7,3>,<12,-1>
$\text{mean}_x=8.27$, $\text{mean}_y=.73$
$\text{stdev}_x=5.44$, $\text{stdev}_y=2.05$
x range within one standard deviation of the mean 2.38 to 13.71
y range within one standard deviation of the mean -1.32 to 2.78
<5,5>, <11,-2> and <-7,3> are each more than one standard deviation from the mean in the x or y direction.
These are removed from the list.
<10,1>,<10,-1>,<8,2>,<9,1>,<10,0>,<12,1>,<10,-1>,<12,-1>
$\text{mean}_x=10.25$, $\text{mean}_y=0.25$
Overall Movement Vector <10.25, 0.25>

Figure 24 - Vector Composition

In the example shown in Figure 24, the overall movement vector is <10.25, 0.25>. This is over 2 units longer in the x direction although almost identical in the y direction when the outliers are removed. This method removes outliers such as <5,5> and especially <-7,3>. Figure 25 and Figure 26 show video frames where there are some clear outliers. In most of these cases the outlier indicates movement in the direction opposite the rest or significantly more movement in the correct direction than the rest. These outliers are disregarded based on the method described in Figure 24 and movement is accurately detected in spite of the anomalous vectors.

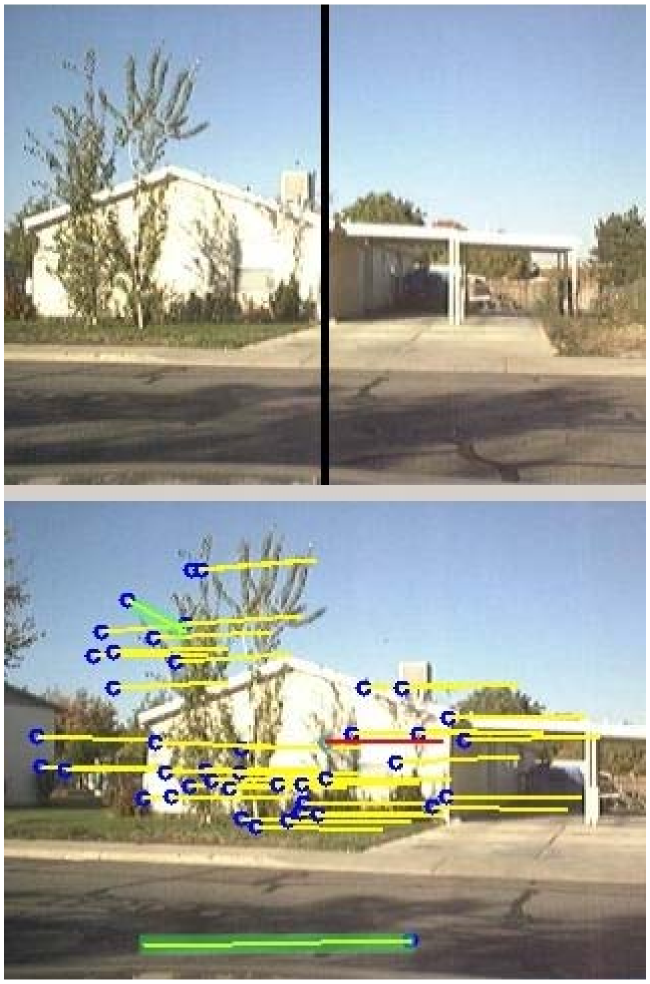


Figure 25 – Interesting Point Outliers
Outliers highlighted in green

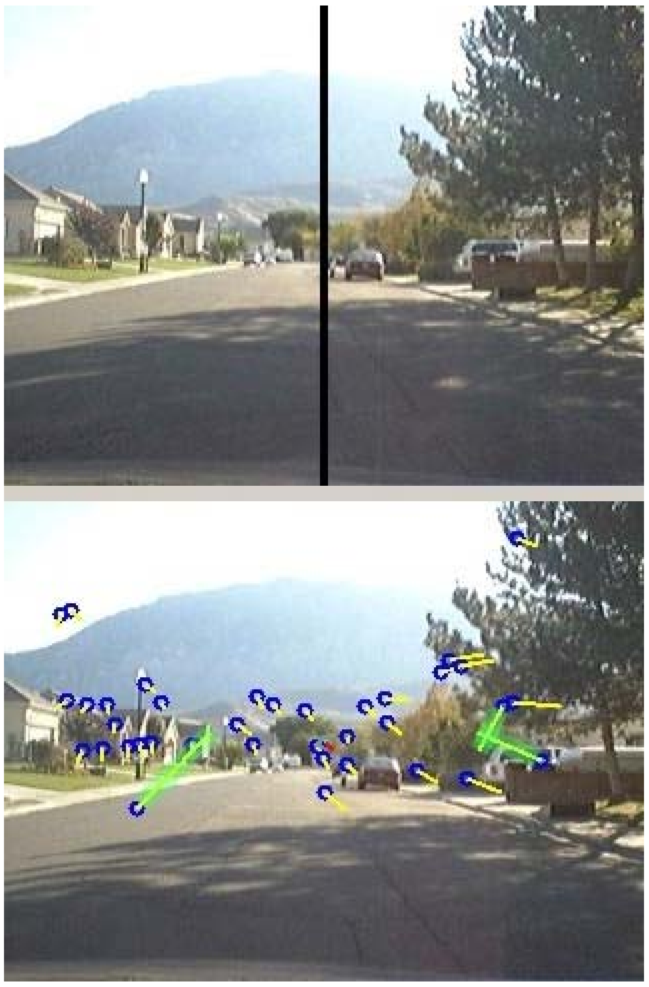


Figure 26 – More Interesting Point Outliers
Outliers are highlighted in green

In Figure 27 there is some difference between how long the movement vectors are on the left hand of the image compared with the right side. Also some of the vectors tilt up or down slightly. Despite that, using the method described above and demonstrated in Figure 24, the overall movement vector shown is as it would be if it were hand annotated. This follows from the observation that the human hand annotator probably uses the same types of contrasting areas (corners and edges) to determine how much movement has occurred.

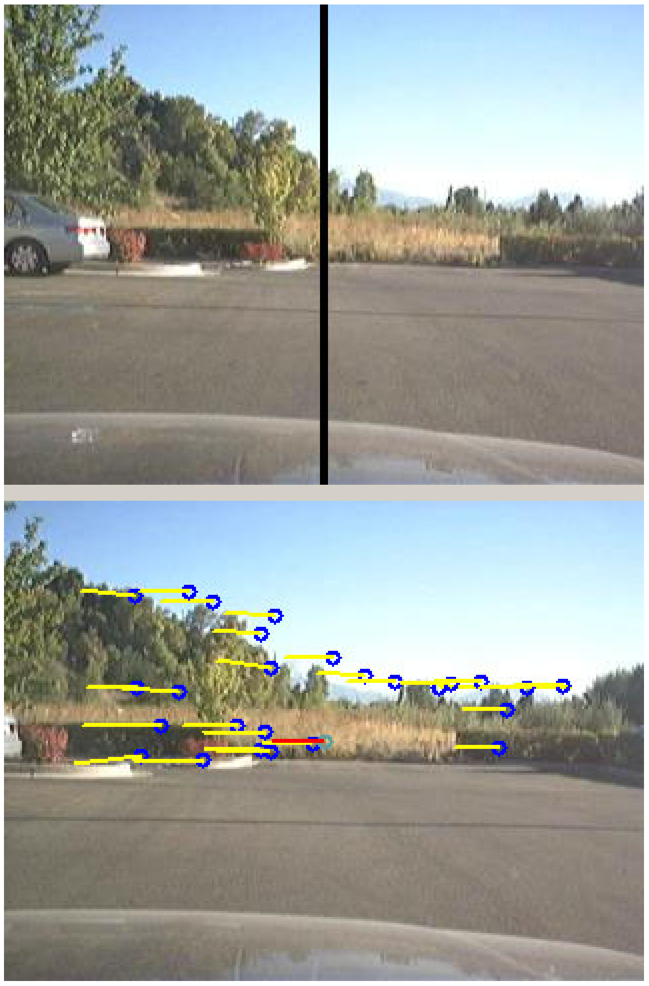


Figure 27 - Example of using interesting points to track movement
Point match movement vectors are shown with blue circles (interesting point from previous image) with yellow lines (movement since previous image). The overall movement that was calculated is shown with a green circle and a red line.

4.7 - Movement Detection Evaluation

The two core software modules (Movement Detection and Video Annotation) of the DigiHead System were evaluated separately. Since several Movement Detection algorithms were examined, each needed to be evaluated against a gold standard. Those results could then be compared. It was proposed that DigiHead would achieve an average of 90% correct annotation. In order to achieve this goal, at least 90% correct movement detection was also proposed as inaccurate movement detection leads to incorrect annotation. Accurate movement detection was achieved.

As previously defined, movement for a particular frame is the number of pixels and direction moved between 10 frames (approximately 1/3 second) previous and the current frame. In order to evaluate the movement detectors, a movement gold standard was created from hand annotations. Each was then compared against the gold standard by comparing the Discrete Movement Class (DMC) that resulted from each movement detector (the guessed DMC) with the hand annotated standard (annotated DMC). In order to account for boundary conditions when classifying the resulting movement, if the inferred DMC was adjacent to the annotated DMC, success was reported.

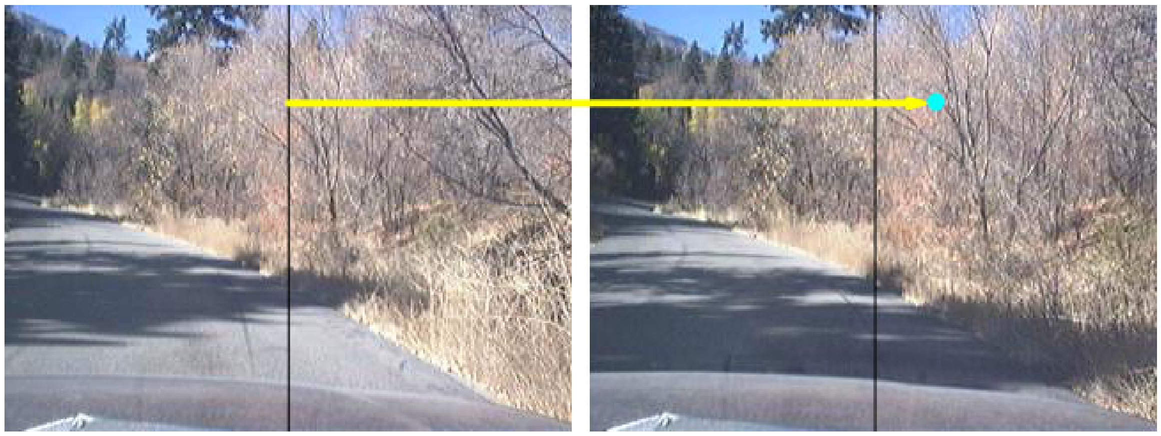


Figure 28 - Hand Annotation of two images 10 frames apart
Black lines are the middle of the image; The area in the middle of the left frame (10 frames previous) is marked with an Aqua dot in the right frame (current frame); The yellow arrow indicates the movement of corresponding area from frame to frame

4.7.1 - Movement Gold Standard

In order to create a gold standard and provide training data for the machine learning approaches to Movement Detection, video gathered was annotated by hand for movement. This was done as shown in Figure 28. Every hundredth frame was annotated. Additionally the more "interesting" parts of the video (i.e. turning) were more heavily annotated (every 10 frames) in order to ensure that all Discrete Movement Classes were represented both for training and testing purposes.

As discussed in Chapter 3, two main classes of video were used. One is data from road driving in the Provo/Orem area. The other is driving along the Alpine Scenic Highway. This provides two contrasting types of driving (navigational versus obstacle avoidance) to test against and compare results.

4.7.2 - Accelerometer Data and Machine Learning using TIDS

In order to test the TIDS approach, training data is necessary to feed into the two machine learning approaches. Training data was created from the feature set from the accelerometer stream data and the annotated Discrete Movement Class. Error rate is calculated as the number of "misses" divided by the total frames tested. This was calculated for each of the three detection types on several different videos. Figure 29 shows the results of these tests.

For Least Squares Approximation, all of the training data was fed to the algorithm. Each accelerometer feature set associated with an annotated frame was then fed into the resulting function and compared with the annotation. Least Squares Approximation is the least accurate method of movement detection that was tested. As can be seen in Figure 29, accuracy never exceeds 86%. Since testing with the same data that was used to train does not result in a good fit, it is clear that this method does not yield sufficiently good results.

For Decision Tree, cross-fold validation was used with 10 folds. Decision Tree is a more accurate movement detection algorithm using the accelerometer features. It classifies the accelerometer features better than the Least Squares method but is still not the best method tested. As can be seen in Figure 29, there was only one example where the accuracy exceeded 90%. In this instance visual point tracking exceeded the decision tree and performed at 97% accuracy.

	Least Squares	Decision Tree	Visual Point Tracking
Pleasant Grove	24.11%	21.43%	0.00%
BYU to UPS Store	28.97%	13.08%	5.61%
Canyon Tech Park to BYU 2	14.65%	11.11%	1.01%
Canyon Tech Park to BYU 1	37.16%	15.87%	3.13%
Canyon Tech Park to USPS in Orem	26.61%	16.13%	1.21%
Canyon Tech Park to UPS Store	13.28%	5.39%	2.90%
Alpine Loop Up	64.32%	34.86%	14.86%
Alpine Loop Down	54.26%	27.35%	15.70%
Max	64.32%	34.86%	15.70%
Min	13.28%	5.39%	0.00%
Mean	32.92%	18.15%	5.55%
Standard Deviation	18.18%	9.43%	6.24%

Figure 29 – Movement Error Rate for all videos
The percentage of images that were not correctly identified

Additional analysis indicates that the speed and acceleration while turning impacts the accelerometer readings more than anticipated (see Figure 22 in "4.5 - Enhancing the Data Stream"). Therefore using the accelerometer data, while interesting, does not provide the best movement detection.

4.7.3 - Visual Point Tracking

To test the Visual Point Tracking approach, no training is necessary. The algorithm was fed the annotated frame and the frame ten frames previous in order to determine the DMC representing the movement between the two frames. These results were then compared to the gold standard and are shown in Figure 29 alongside the results from the accelerometer data approach.

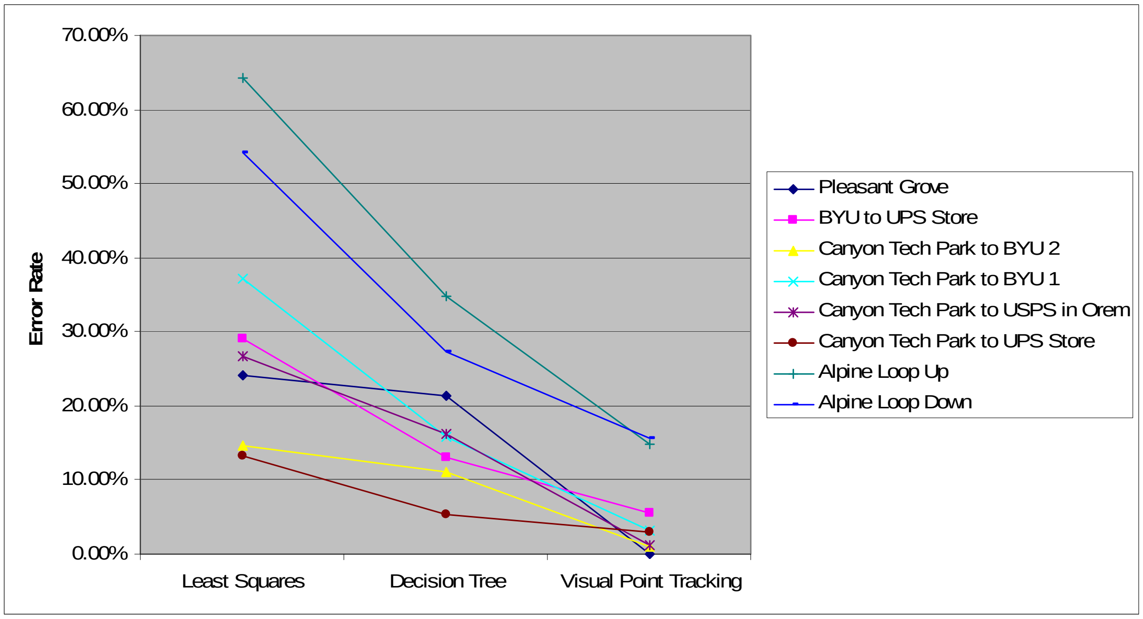


Figure 30 - Movement Results Chart

In each of the eight videos evaluated, the visual point tracking approach clearly outperformed the Decision Tree and Least Squares approaches. In six of the eight videos the accuracy exceeded 90% and in five of those the accuracy exceeded 95%. The two other videos were those taken on the Alpine Loop drive. The overall average error rate for visual detection is 5.55% with standard deviation of 6.24% compared with 18.15%/9.43% for Decision Tree and 32.92%/18.18% for Least Squares.

City Driving/Navigational

	Least Squares	Decision Tree	Visual Point Tracking
Mean	24.13%	13.84%	2.31%
Standard Deviation	9.02%	5.41%	2.01%

Mountain Driving/Obstacle Avoidance

	Least Squares	Decision Tree	Visual Point Tracking
Mean	59.29%	31.11%	15.28%
Standard Deviation	7.12%	5.31%	0.59%

Figure 31 - Aggregate Error Rate Statistics for different driving types

It is interesting to note the difference in error rates between videos of city driving vs those of mountain driving as seen in Figure 31. City driving (which excludes the Alpine Loop videos) performed even better than the overall average with an average of 2.31% with a standard deviation of 2.01%. Mountain Driving (which includes the two Alpine loop videos) does significantly worse with an average of 15.28% with a standard deviation of 0.59%. Despite this, the Visual Point Tracking algorithm outperforms the other two methods by a wide margin with Decision Tree averaging 31.11% and Least Squares inferring less than half correctly with an error rate of 59.29%.

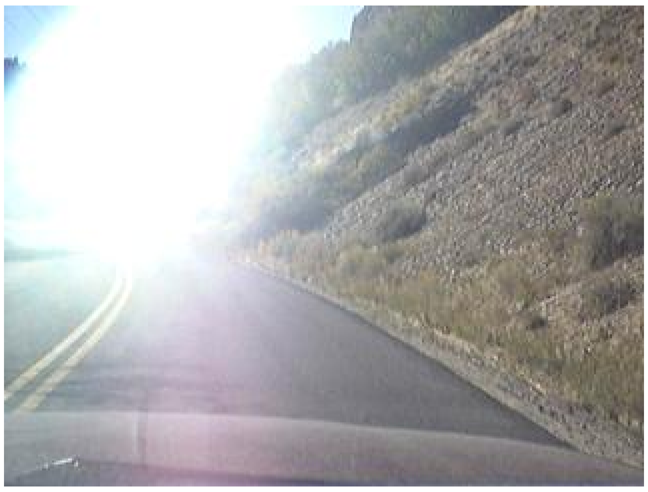


Figure 32 – Field of view is obscured by glare

While the visual point tracking algorithm does quite well in most circumstances, in examining the videos there are some reoccurring reasons that the visual point tracking fails to perform accurately. Figure 32 shows video frames where for certain a portion of the drive, part of the field of view is obscured by glare from the sun. In the Alpine Loop videos there were several spots where trees or tall grass are close along the side of the road creating walls that have few interesting points. Some frames show that while turning a wall is all that is visible or a tunnel of trees blocks any view of distant objects. Figure 33 shows some tall grass and trees that have some distinguishing features, but are similar enough that the point matching algorithm is confused and therefore gives an inaccurate movement.

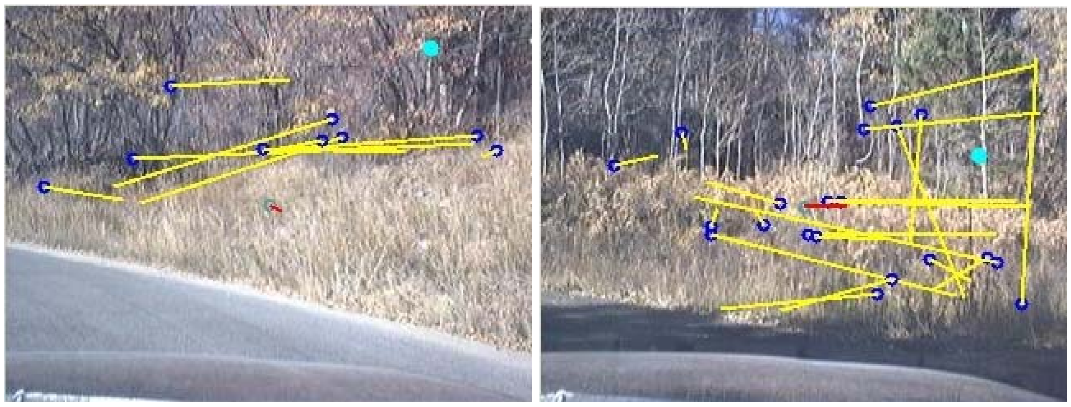


Figure 33 - Close Grass confuses Interesting Point matching
These are two examples of the Interesting Point matching algorithm failing with close uniform objects

Distant objects clearly are more favorable in order to get an accurate movement reading. Figure 34 shows how the points identified far away were more accurate while those that are closer (the upper right) tended to be incorrect. In the case shown the incorrect vectors are outliers and therefore ignored enabling accurate movement detection.

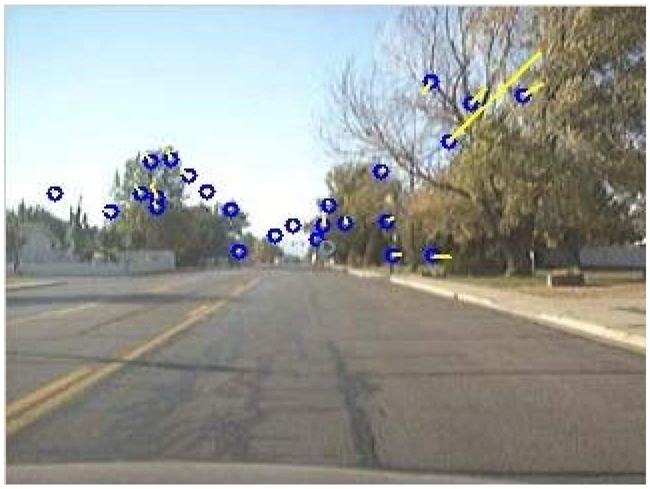


Figure 34 - Close vs. Far

4.8 - Summary

Two categories of methods, using accelerometer data and using vision, were compared to determine the best movement detector for use in the DigiHead system. This movement detection is critical in order to automatically annotate the video as it provides information that allows us to know how the driver is responding to his environment. How the human driver responds (based on the movement detected) can be used to predict which areas of the field of view are obstacles or safe when used in combination with the techniques found in the following chapter.

Chapter 5 : Video Region Annotator

The Video Region Annotator component of the DigiHead system contains several subcomponents that work together to identify the areas of each frame that are safe or unsafe to drive. In order to be able to accurately annotate automatically, some hand annotations are necessary to create the probabilistic predictors previously referenced, called Probability Grids (PGrids), which are used to annotate the remainder of the frames. To accomplish this, the application used for movement detection was extended to include Crayons-inspired hand annotation of the video frames for safe, unsafe and unsure areas. This functionality is shown in Figure 35. The set of Probability Grids created from the hand annotations can then be used in conjunction with a Movement Detector in order to annotate additional frames with a high degree of accuracy.

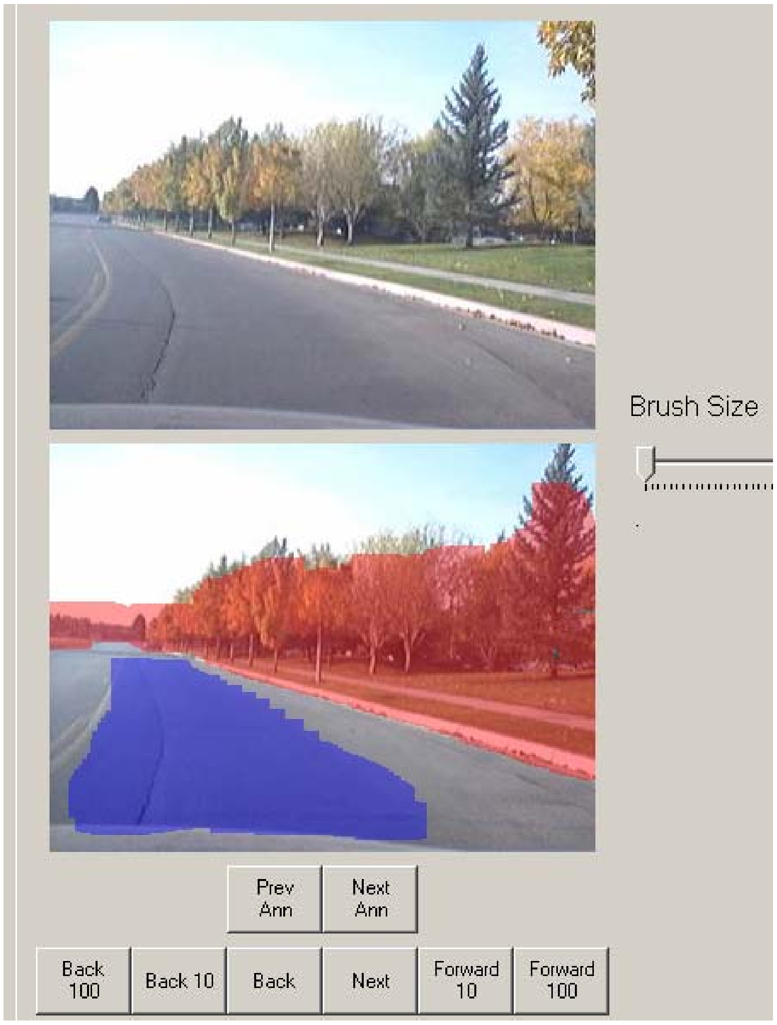


Figure 35 - Screenshot of the region annotation application
Blue area is annotated as safe; red area is annotated as unsafe.

5.1 - Probability Grids

Discrete Movement Classes, as explained earlier, quantize annotated or detected movement between frames. This movement indicates a turn and therefore navigation or obstacle avoidance. In either case the human driver is making a decision to continue driving on safe terrain. (This assumes that the vehicle does not collide with any obstacles). A PGrid indicates the probability that any given section of an image is safe to drive on. In the case of a right turn the area directly in front of the vehicle as well as that right of center are likely safe areas as sown in the right image of Figure 36. Left of center might be unsafe or unsure. By creating a PGrid for each DMC, we create the necessary link between movement and safe/unsafe annotation. Therefore when the movement detector infers a particular DMC, the PGrid associated with that DMC can be applied to the image.

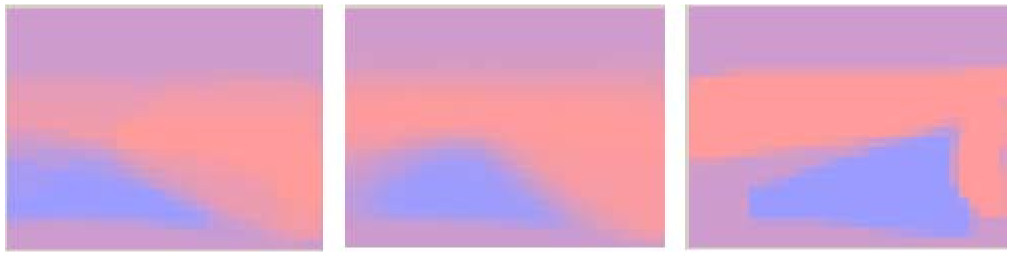


Figure 36 - Sample Probability Grids
Red represents obstacles, Blue represents safe areas, Blends (shades of purple) represent varying degrees of certainty;
Left is from Left Turns, Center is from Straight Movement, Right is from Right Turns

A PGrid represents the probability that any given region of an image should be annotated as either safe or unsafe. These probabilities are calculated as a weighted average of several hand annotations. After only a few annotated frames for a particular Discrete Movement Class are added, a clear pattern generally emerges such that additional hand annotations provide diminishing returns.

PGrids are a 2D array of PGrid cells. Each cell of a PGrid represents a region which can represent a single pixel or any rectangular group of pixels in the original data. Allowing for summarization above the pixel level can help to smooth out anomalies in the data including incorrect hand annotations. Each cell of a PGrid has a value between 0 and 1 representing a degree of "safety" with 0 being unsafe and 1 being safe. For testing and visual display purposes both here and in the application, safe (1) is represented as blue, unsafe (0) is represented as red and unannotated (.5) as purple. Any value between 0 and 1 is displayed visually as a blend between red (0) and blue (1), usually some shade of purple.

PGrids are designed to allow for arbitrary dimensions. The first annotation added to a PGrid determines its pixel dimensions. Figure 37 shows four 3 pixel by 3 pixel sample annotations. All other annotations added to it must have the same pixel dimensions. Attempting to add an annotation that is 4 x 4 to the same PGrid that the annotations in Figure 37 had been added to (which are 3 x 3) would not be successful.

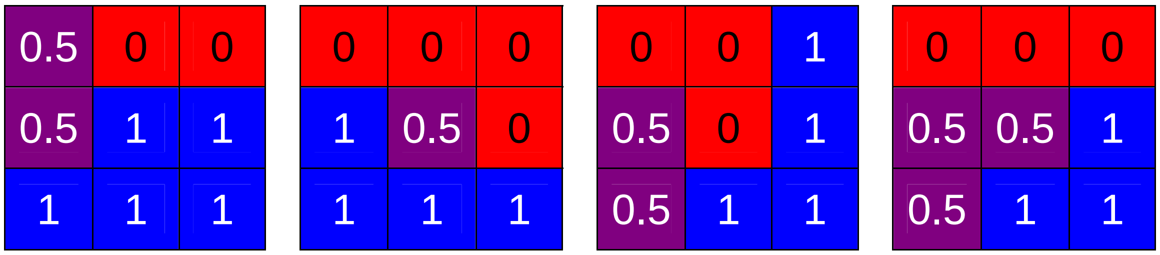


Figure 37 – Four Sample Annotations to be summarized with a PGrid

Three totals are stored at the pixel and cell level; the number of safe pixels, the number of unsafe pixels and the number of unannotated pixels added to the PGrid for that pixel location. These totals are shown in Figure 40 for the samples shown in Figure 37. A value between 0 and 1 can be calculated for each PGrid pixel by taking a weighted average of the three totals. Again using the data from Figure 37, weighted averages have been calculated and shown in Figure 40.

Unsafe : 3 Unsure : 1 Safe : 0	Unsafe : 4 Unsure : 0 Safe : 0	Unsafe : 3 Unsure : 0 Safe : 1
Unsafe : 0 Unsure : 3 Safe : 1	Unsafe : 1 Unsure : 2 Safe : 1	Unsafe : 1 Unsure : 0 Safe : 3
Unsafe : 0 Unsure : 2 Safe : 2	Unsafe : 0 Unsure : 0 Safe : 4	Unsafe : 0 Unsure : 0 Safe : 4

Figure 38 - PGrid Totals

Using this value, Probability Grids can infer if any given pixel of an image should be annotated as safe or unsafe. This is done through the use of a threshold value. If the resulting weighted average is less than the threshold value from 1 then the area is classified as safe. If the resulting weighted average is less than the threshold value from 0 then the area is classified as unsafe. Any values that fall in the middle are classified as unsure. This unsure classification indicates that the image data in that area would not generally be used as training data for a machine learning algorithm.

$$\frac{UnsafeCount \times UnsafeWeight + SafeCount \times SafeWeight + UannotatedCount \times UannotatedWeight}{UnsafeCount + SafeCount + UannotatedCount}$$

where

UnsafeWeight = 0

SafeWeight = 1

UnannotatedWeight = .5

Figure 39 – Weighted Average Calculation
This applies to single pixel averaging and region summarization

🟥 0.1	🟥 0	🟪 0.3
🟪 0.6	🟪 0.5	🟪 0.8
🟪 0.8	🟦 1	🟦 1

Figure 40 – PGrid Cell Values as each annotation from Figure 37 is added to the PGrid
For this example a threshold value of .25 was used to decide if the annotation would be Unsafe (<=.25), Safe (>=.75) and Unsure (>.25 and <.75)

The significance of using the unannotated areas in the calculations can be seen by observing the left center cell. Even though one of the annotations marks it as safe, the remaining annotations have it unannotated. This causes the PGrid cell value to be .6 which, with a threshold value of .25, is categorized as unsure instead of safe.

5.1.1 - Region Summarization

As was mentioned earlier, PGrids can be used with different degrees of granularity by summarizing the data at a region level. PGrid cells can represent a single pixel, the entire image or anywhere in between. Figure 41 shows the difference between three probability grids that represent straight driving created with the same hand annotations, but with different levels of granularity.

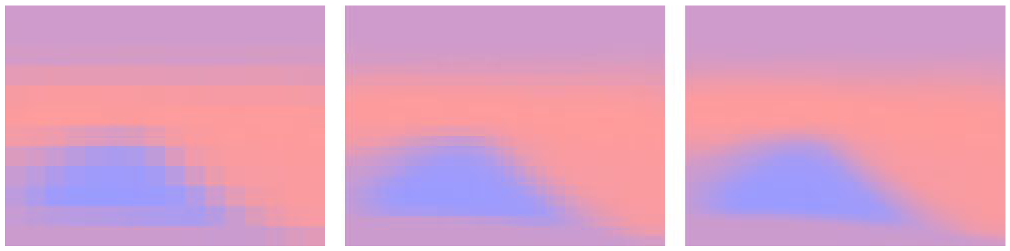


Figure 41 - Probability Grid Granularity
Red – Probable Obstacle; Blue – Probable Safe; Purple – Unsure
Left: 20x20 pixel cells; Center: 5x5 pixel cells; 1x1 pixel cells

In order to summarize the data at the region level, the same method that is used to combine several annotations together is used to combine the pixels for a given region. Counts of safe, unsafe annotated pixels as well as unannotated pixels are used together. These counts are then weighted based on safe (1), unsafe (0) or unannotated (.5) pixels. Calculating the weighted average of the pixel value results in a value between 0 and 1 (inclusive). Figure 43 shows how the 3 x 3 pixel data in Figure 42 would be combined to represent a single PGrid cell.

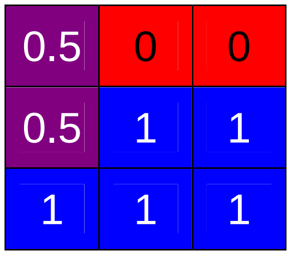


Figure 42 - Example data from a single frame to be summarized in a Probability Grid cell

$$\frac{2\times 0+5\times 1+2\times .5}{2+5+2}=\frac{6}{9}=.\overline{6}$$

Figure 43 – Calculation of weighted average for data in Figure 42

5.1.2 - Annotation Using Lack of Annotation

By using lack of annotation as a type of annotation and including it in the calculations of PGrid cell values, the cell becomes a better representation of the safety of the area. A dramatic example is where one pixel in a cell might be annotated as safe while the remainder is unannotated (or if one frame has part of the cell area annotated as safe and all others have it left unannotated). If lack of annotation was not used in the counts, a single annotated cell can overwhelm the entire cell. Assuming that the cell represents a large number of pixels (20 x 20 area), this would result in the PGrid cell value being changed from 1 to .501. If additional annotations were added that contained more safe annotations for the same region, those could overwhelm the unannotated frame and cause it to cross the threshold. Another example of this is that the weighted average for the data in Figure 42 would be approximately 0.714 instead of .667 if unannotated pixels were not used in the average.

Including lack of annotation in our calculations results in a more "conservative" PGrid. The resulting PGrid is not as aggressive in annotating the data and therefore neglects to annotate some of the data that it may have annotated. The primary benefit to this more conservative approach is that it avoids incorrectly annotating areas and therefore gives greater confidence in the correctness of the resulting annotation.

5.2 - Automatic Annotation using Probability Grids and Movement Detection

Once the set of Probability Grids has been created from the hand annotations, they can be integrated with the Movement Detector in order to automatically annotate new video data. First the Movement Detector decides what action the driver has taken by inferring a Discrete Movement Class based on the movement detected. This Discrete Movement Class corresponds to one of the Probability Grids. This PGrid is then associated with the corresponding video frame image.

The PGrid will indicate probabilities for each area of the frame image as to how safe that area is to drive towards. Using the threshold, certain areas can be annotated as safe or unsafe. This method produces a large amount of annotated data that can then be used to train a machine learning algorithm. If the target machine learning algorithm (which is not defined by this work) can accept confidence levels, the PGrid probability values could be used without a threshold value.

Using these two techniques together produces a reliable and accurate approach to creating large amounts of annotated data with minimal user input. By using the movements of the vehicle to help annotate the data, we can minimize the amount of additional user input to annotate the video.

5.3 - Video Region Annotator Evaluation

The performance of the Video Region Annotator was compared against a gold standard of hand annotated data. The best Movement Detector was used for the Video Region Annotator testing. It was proposed that DigiHead would achieve an average of 90% correct annotation. Together with the accuracy of the Movement Detector, this goal was achieved.

In evaluating the effectiveness of the Video Region Annotator, two metrics were measured: accuracy and completeness. Accuracy measures the correctness of the system-generated annotations. Therefore if an area is guessed to be safe but the hand annotation indicates that it is unsafe, that would negatively impact accuracy. Completeness measures what percentage of the area hand annotated as safe is inferred as safe or unsafe by the Annotator, indicating how much additional area could have been annotated. Therefore, area that the hand annotation indicates as safe but the inference leaves unannoated negatively impacts completeness. Accuracy and completeness together give a more complete view of how good an annotation is then either one alone.

The entire upper half of any given video frame is always unsafe (unless extremely hilly or mountainous terrain is encountered). Therefore the upper half has no bearing on which direction a car should move based solely on safe or unsafe paths. It is useful for visual point tracking in movement detection since landmarks in the upper region can be used to demonstrate movement visually especially since it is generally more descriptive of the environment than the lower half. To take this into account, only the annotations in the lower half of the video image were evaluated. When evaluating the automatic annotation this actually causes the measured annotation accuracy to decrease but makes the measurement more relevant.

5.3.1 - Region Annotation Gold Standard

In order to create a gold standard and provide training data for the Probability Grids, video frames were annotated by hand for obstacles and safety. Every hundredth frame was annotated for safety by marking areas as either safe or unsafe (in addition to the movement annotation that existed previously). Additionally the more "interesting" parts of the video (turning) were more heavily annotated (every 10 frames) in order to ensure that all Discrete Movement Classes were represented both for training and testing purposes.

5.3.2 - Evaluation Results

A sub-sample of each turn type in a given data set is used to create the probability grids. Grids are then compared to an entire set which may or may not be the same data set.

Two different parameters were varied in the results shown: threshold value and sub-sample rate. The threshold value indicates what probability to use in determining safe and unsafe from the PGrids. For example a threshold of .5 would mean that the entire PGrid would be classified as either safe or unsafe as anything above .5 would be safe and anything below .5 would be unsafe. Using .98 indicates that anything above .98 would be safe and anything below .02 would be unsafe. Any values in between would be considered unannotated. It is expected that the higher the threshold, the higher the accuracy as we are more sure of our annotation.

Sub-sample rate indicates what percentage of the total number of annotated images for a given Discrete Movement Class (DMC) to use in creating the PGrid. For example if the sub-sample rate is .2 then 20% of each DMC would be used for each of the PGrids. This ensures that each of the DMCs are represented while only using a subset of each them.

5.3.2.1 - Canyon Tech Park to BYU

Using data set "Canyon Tech Park to BYU" as the input and the test set, the accuracy results are shown in Figure 44 and Figure 45 and the completeness results are shown in Figure 46 and Figure 47.

		Threshold					
		0.5	**0.6**	**0.7**	**0.8**	**0.9**	**0.98**
Sub-sample Rate	**0.05**	69.24%	84.54%	87.70%	91.26%	93.41%	94.60%
	0.1	69.28%	84.59%	88.47%	91.79%	94.08%	95.24%
	0.2	69.62%	83.43%	87.77%	91.63%	94.17%	96.01%
	0.3	69.72%	83.74%	88.69%	92.41%	94.76%	96.60%
	0.4	69.92%	83.53%	88.87%	93.33%	95.82%	97.61%
	0.5	69.93%	83.75%	89.05%	93.36%	96.00%	97.68%
	0.6	70.10%	83.79%	88.99%	93.96%	96.63%	98.71%
	0.7	70.13%	83.74%	89.80%	94.04%	96.76%	98.74%

Figure 44 – Accuracy of PGrids
Used Canyon Tech Park to BYU data set to train and evaluate
Thresholds from .5 to .98 and Sub-sample rates from .05 to .7

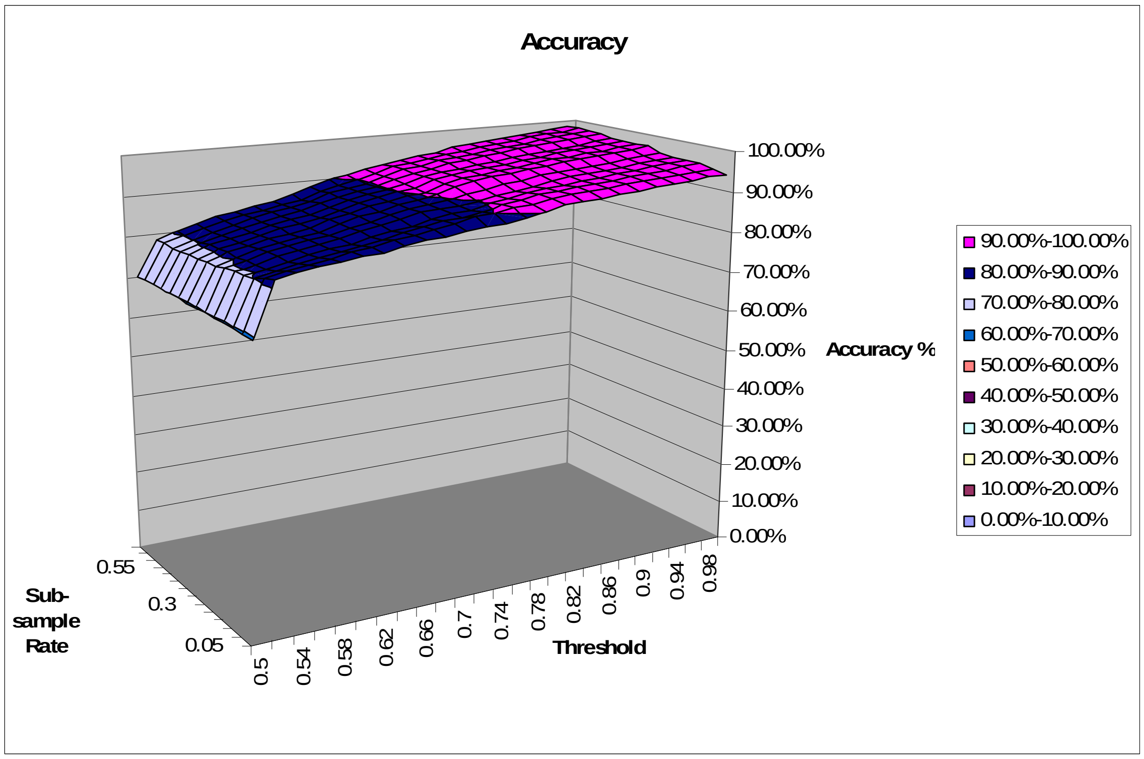


Figure 45 - Accuracy of PGrids Graph
Used Canyon Tech Park to BYU data set to train and evaluate
Thresholds from .5 to .98 and Sub sample rates from .05 to .7
Shows more increments than the table

Note that the sub-sample rate does not significantly impact the accuracy of the PGrids, varying anywhere from less than 1% to a little over 4%. In some cases (for instance threshold of .6 in Figure 44) the higher sample rate actually decreased the accuracy. On the other hand, the threshold does impact the accuracy rate as expected. The more selective we are about what is considered safe or unsafe (higher threshold) means that there is less annotation to be incorrect and most areas where there is any uncertainty are not marked as safe or unsafe. It is interesting to note that the biggest increase is from .5 to .52 which seems to indicate that allowing for that first bit of unknown area gives significant benefits.

		Threshold					
		0.5	**0.6**	**0.7**	**0.8**	**0.9**	**0.98**
Sub-sample Rate	**0.05**	100.00%	89.09%	82.82%	73.86%	66.39%	58.89%
	0.1	100.00%	90.31%	82.79%	73.41%	63.85%	51.02%
	0.2	100.00%	92.02%	84.34%	74.47%	63.58%	42.54%
	0.3	100.00%	92.25%	83.04%	72.98%	63.13%	41.40%
	0.4	100.00%	93.45%	83.69%	72.03%	60.37%	40.52%
	0.5	100.00%	93.09%	83.55%	72.13%	60.06%	40.27%
	0.6	100.00%	93.67%	83.98%	71.39%	58.78%	34.56%
	0.7	100.00%	93.98%	82.65%	71.70%	58.98%	36.29%

Figure 46 - Completeness of PGrid
Used Canyon Tech Park to BYU data set to train and evaluate
Thresholds from .5 to .98 and Sub-sample rates from .05 to .7

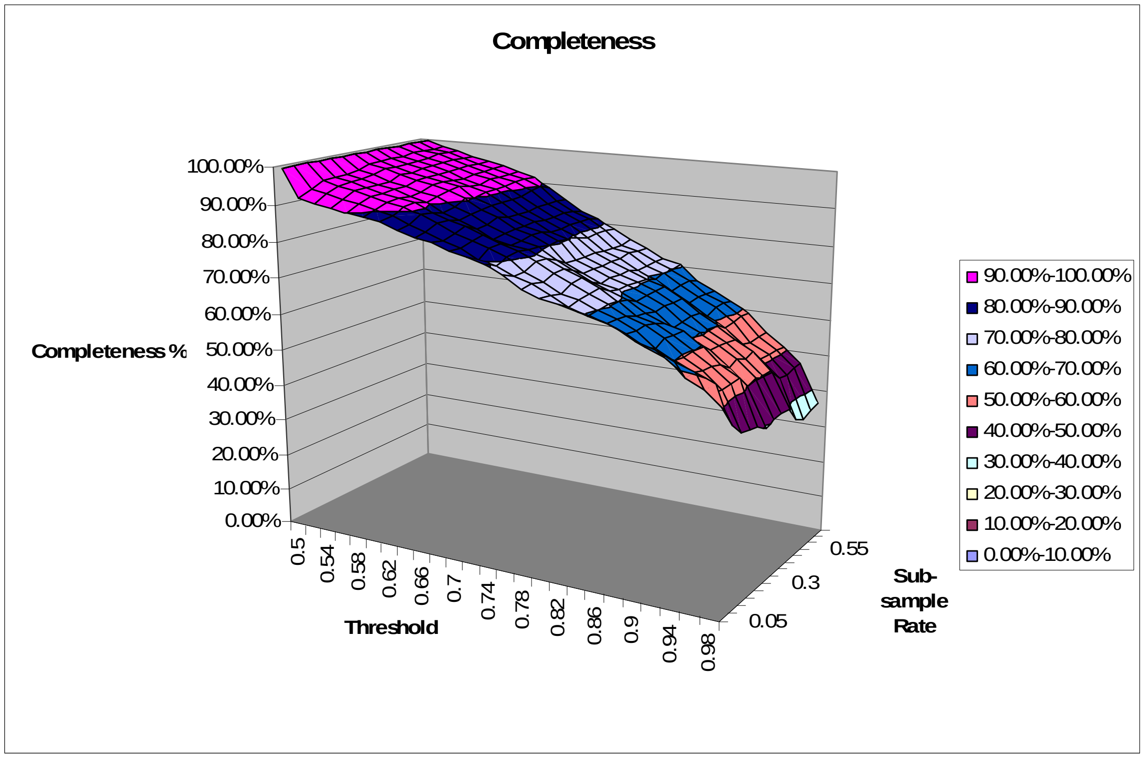


Figure 47 – Completeness of PGrid Graph
Used Canyon Tech Park to BYU data set to train and evaluate
Thresholds from .5 to .98 and Sub sample rates from .05 to .7
Shows more increments than the table

The completeness data in Figure 46 and Figure 47 indicate the somewhat obvious fact that as the threshold rate increases the less complete the annotation will be. It is interesting to note that sub-sample rate has little to no influence on completeness. Looking specifically at the 0.7 threshold value in Figure 46, the completeness does not correlate with the sub-sample rate.

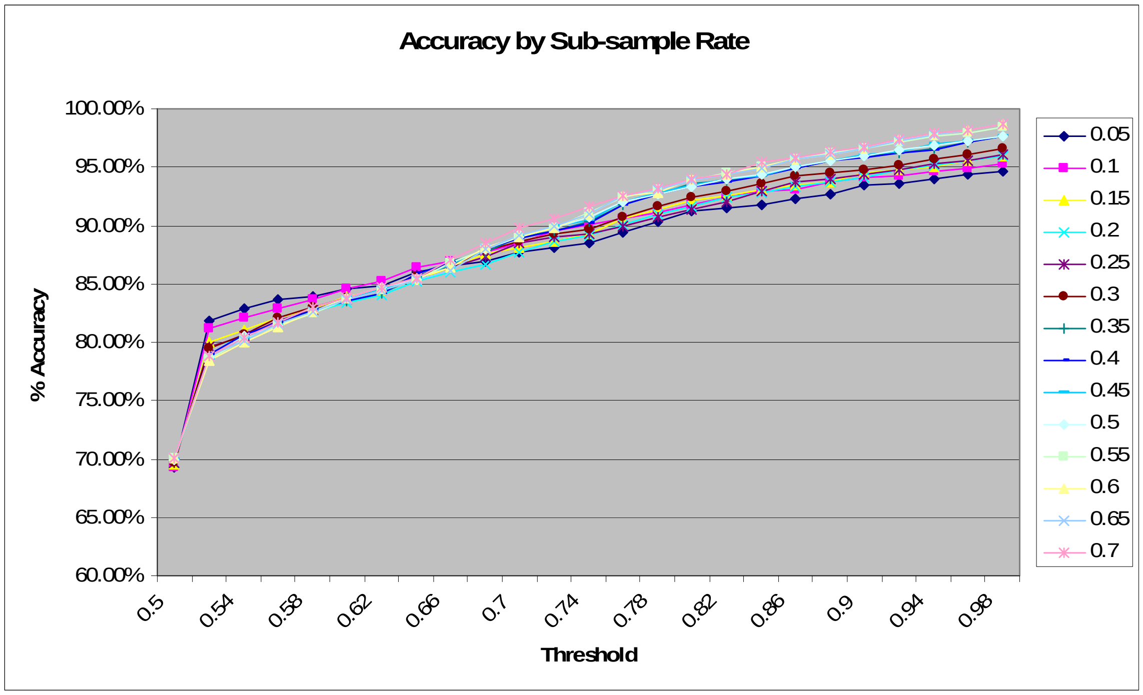


Figure 48 - Accuracy by Sub-sample rate Chart (data from Figure 45)
As the threshold value increases, the accuracy increases, independent of the sub-sample rate

Sub-sample rate and accuracy appear to be independent of each other as is demonstrated in Figure 48 and Figure 49. Figure 48 shows that as the accuracy is about the same for each threshold, regardless of what the sub-sample rate is. Figure 49 shows that changes in the sub-sample rate do not significantly impact the accuracy, independent of the threshold value.

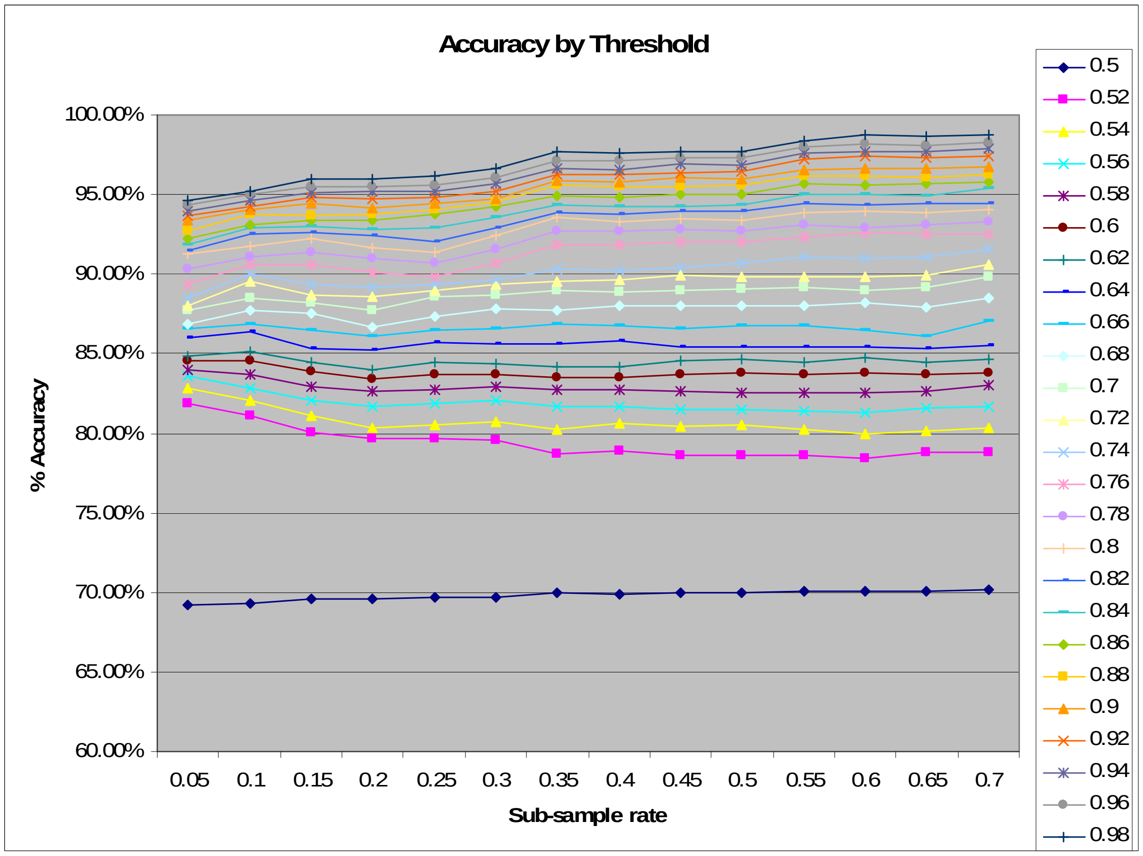


Figure 49 - Accuracy by Threshold Chart (data from Figure 45)
As the sub-sample rate varies the accuracy is not significantly impacted with any threshold value

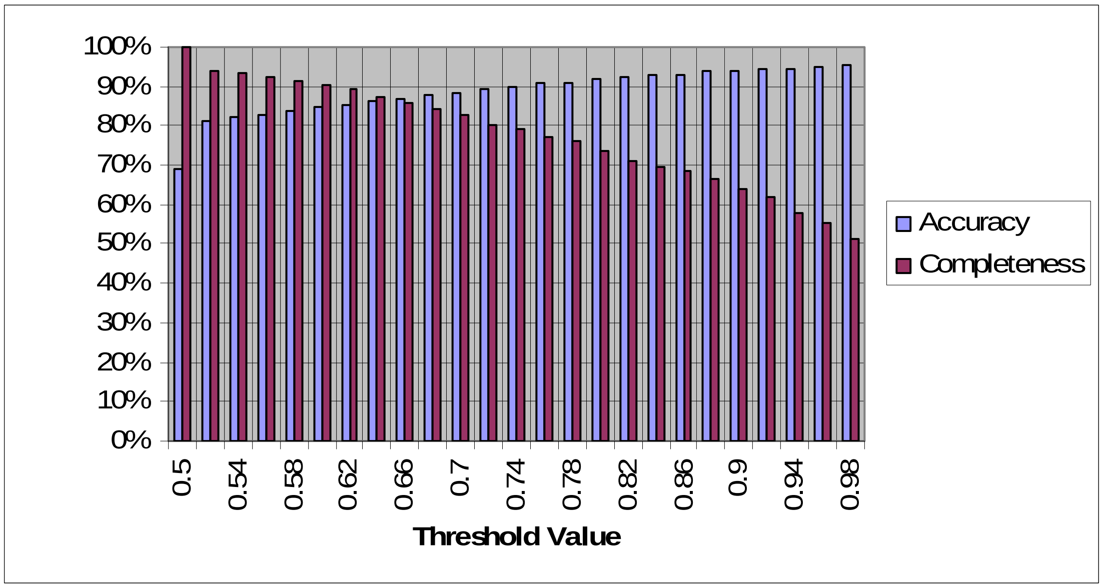


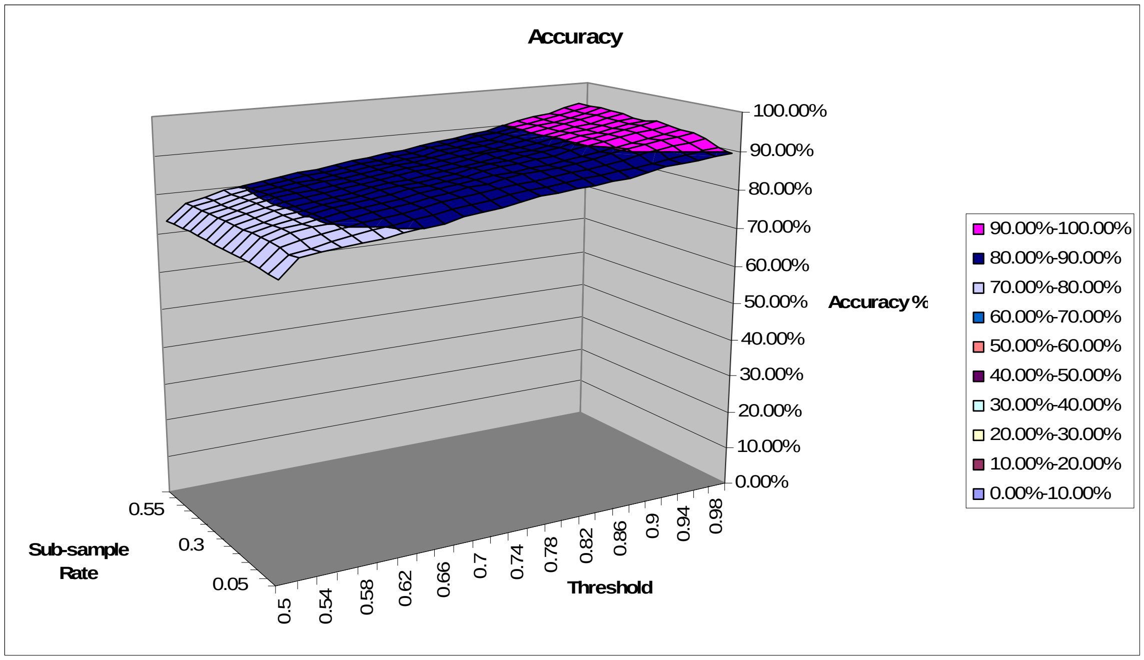
Figure 50 - Accuracy vs Completeness with Various Thresholds
Demonstrates that the higher the accuracy, the lower the completeness
Used Canyon Tech Park to BYU data set to train and evaluate
Sub-sample rate used was .1

Figure 50 shows accuracy and completeness for sub-sample rate .1 with varying thresholds in the same chart. While the natural tendency is to assume that the intersection of these (approximately .66) would be the “ideal”, it really depends upon the needs of the user. In many circumstances a higher accuracy with a lower completeness is necessary as the volume of annotation data generated (completeness) is not as important as the accuracy.

5.3.2.2 - Alpine Loop Up and Down

The primary goal of DigiHead is to allow for annotation of large amounts of data with little user interaction. This originally was meant to allow a video to be sparsely hand annotated and automatically annotated the rest of the way by DigiHead. The next step is to be able to apply PGrids from one video to another which allows for greater application of the small number of hand annotations.

Figure 51 shows accuracy and completeness with the Alpine Loop Up data set training and the Alpine Loop Down data set testing. In this case both of the videos were taken in the same trip, use similar driving styles (mountain road), with the camera mounted the same way in both videos. Accuracy and completeness are both lower but 90% accuracy was still attained with high enough threshold values and sub-sample rates.



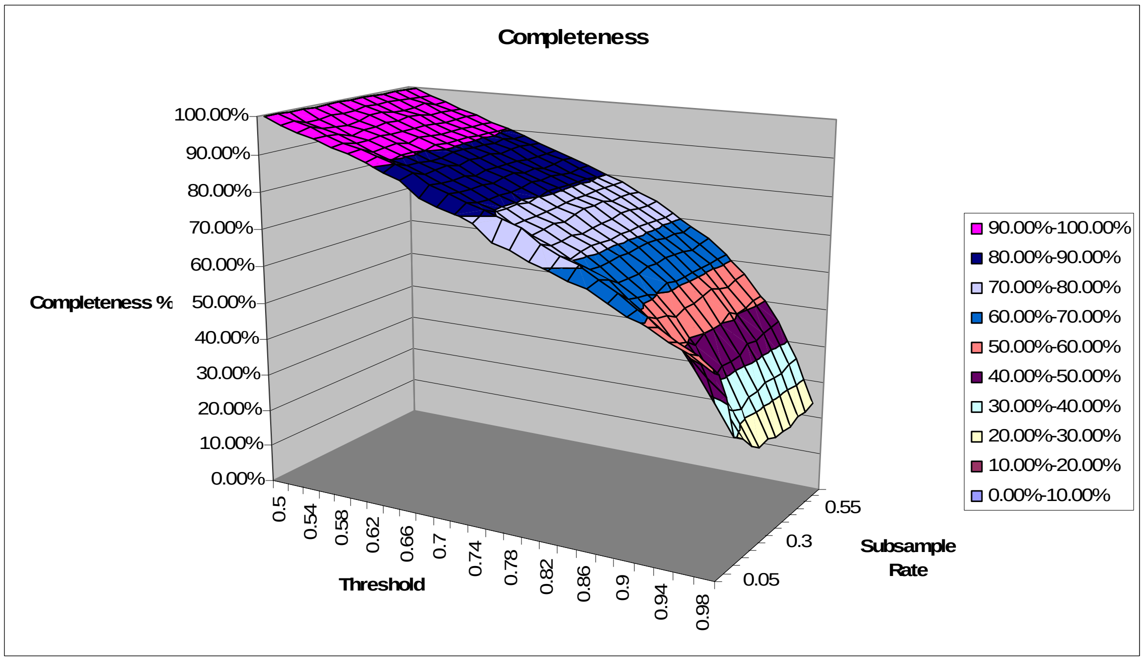
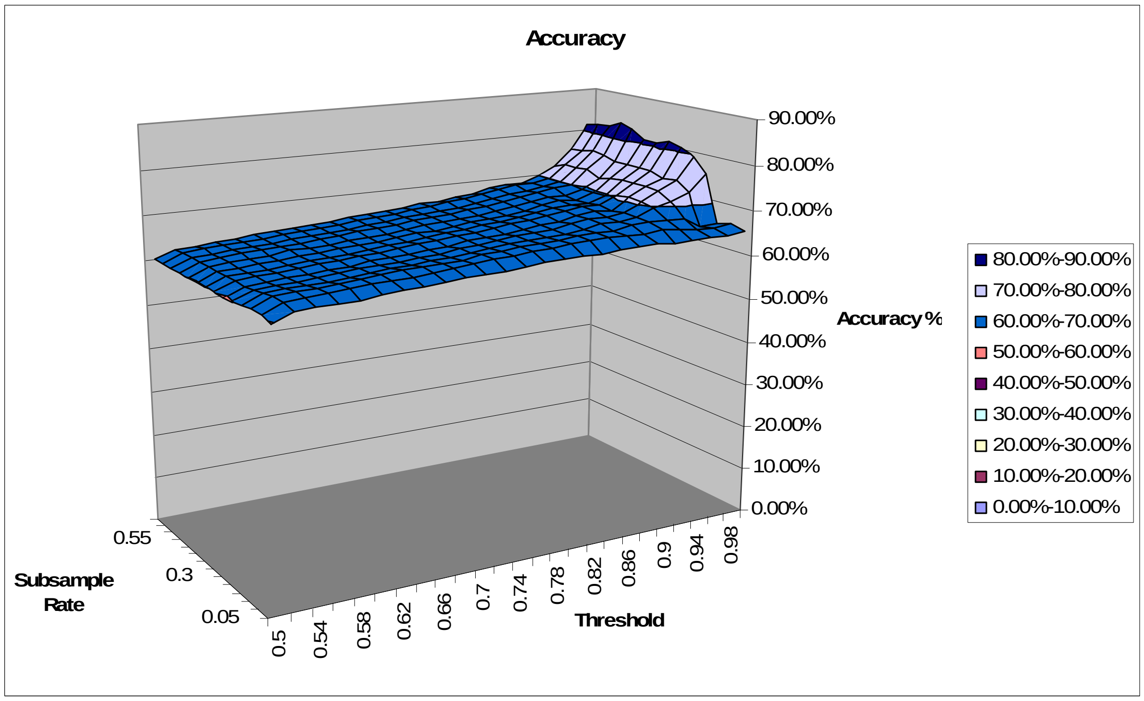


Figure 51 - Accuracy and Completeness
Input Alpine Loop Up; Test Alpine Loop Down

5.3.2.3 - Alpine Loop Up and Canyon Tech Park to BYU

The two data sets used in this test represent two different styles of driving (mountain road vs. city road) and therefore are somewhat incompatible. Alpine Loop Up is used as training data and Canyon Tech Park to BYU is used to test. Figure 52 shows accuracy and completeness data for this test. Completeness looks very similar to the other tests. Accuracy is much lower in this case and is even more flat than the other tests for most threshold values.



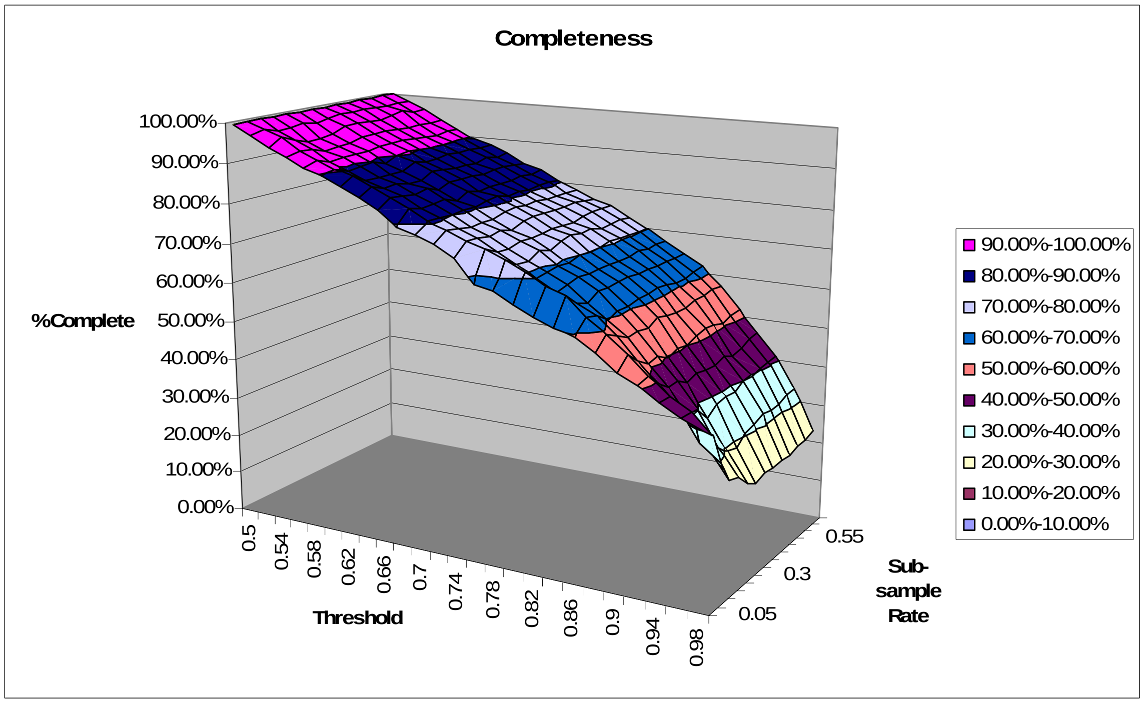


Figure 52 - Accuracy and Completeness
Input Alpine Loop Up; Test Canyon Tech Park to BYU

Since the camera was mounted slightly differently, the PGrid profile is also slightly different. The spike with higher thresholds supports this as this indicates that once the less sure area is no longer annotated because of the threshold, the accuracy increases sharply. Figure 53 shows the area that would be incorrectly annotated using Alpine Loop one as input and Canyon Tech Park to BYU as test data. As the area indicated is often less sure than the rest of the annotation, as the threshold increases, this area will no longer be annotated. This will obviously reduce the completeness but will result in the spike in the accuracy results around the threshold of .94. This is also obvious in other video pairs as shown in Figure 54. The figure shows video frames and where "forward" is for four different videos and how they don't match up due to variations in camera position and lane width.

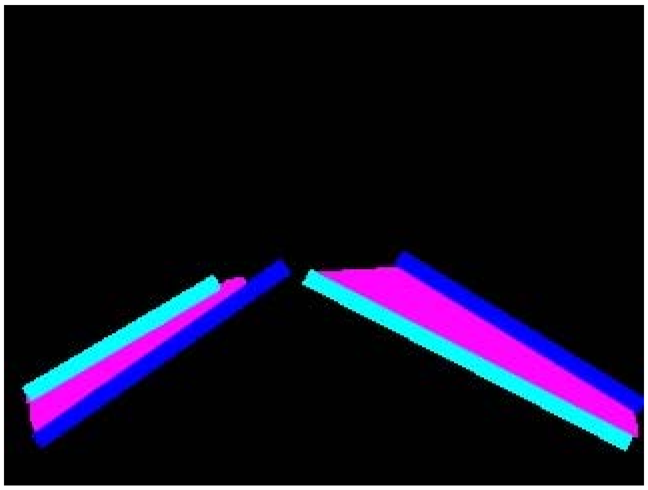
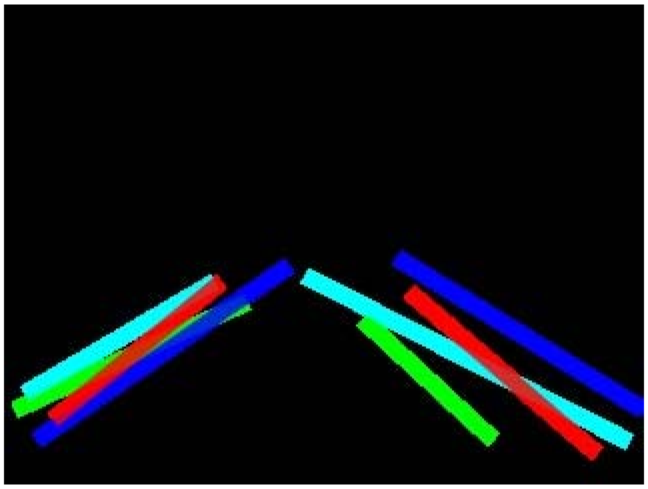


Figure 53 - Lane Profile for Canyon Tech Park to BYU (Blue) and Alpine Up (Aqua)
Purple indicates area that would be incorrectly annotated because of a mismatch in lane profile.



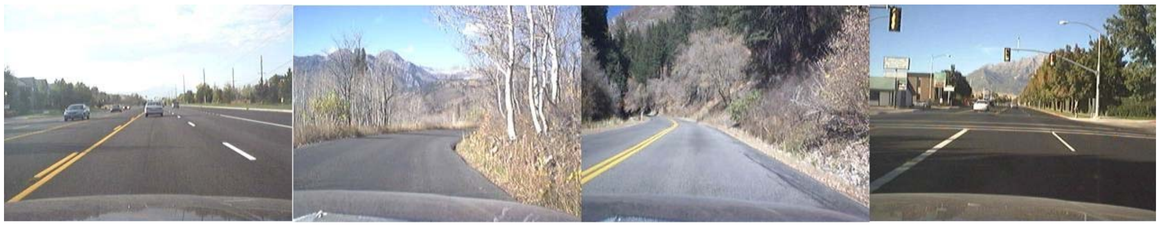


Figure 54 - Forward Lane Profiles
Shows slight variations in camera position and lane size
From left to right: Blue; Green; Aqua; Red
Blue is Canyon Tech Park to BYU and Aqua is Alpine Loop Up (Figure 52)

5.3.3 - Results Summary

The video region annotator performed quite well overall. Figure 55 shows a comparison of a few different videos and the results. Except for the test using the Alpine Loop Up data as input and Canyon Tech Park as test data, all accuracy readings are 90%+ which was the proposed goal.

	Sub-Sample Rate/Threshold Value		
	.7 / .98	**.7 / .9**	**.4 / .98**
Canyon Tech Park	98.74%	96.76%	97.61%
Alpine Loop Up	97.58%	94.98%	97.11%
Alpine Loop Down	98.04%	95.96%	96.49%
Alpine Loop Up - Alpine Loop Down	94.26%	90.63%	93.45%
Alpine Loop Up - Canyon Tech Park	81.38%	68.45%	80.81%

Figure 55 – Comparison of Accuracy of Various Tests

5.4 - Summary

Using hand annotations, Probability Grids and a Movement Detector, the Video Region Annotator can quickly annotate a large number of video frames. This is accomplished by taking the hand annotated frames and using the Discrete Movement Class assigned to each to create a PGrid representing typical annotation for each Movement Class. These PGrids can then be applied to additional video data by using the Movement Detector to determine the Discrete Movement Class and applying the appropriate PGrid. In this way large amount of video data can be accurately annotated automatically.

Chapter 6 : Overview and Conclusions

Visual obstacle detection by robots is a difficult problem. Humans have a visual obstacle detection system that works very well even in unfamiliar situations because of the years of experience that we have. Machine learning techniques can be used to approximate this learning process but they require training data, preferably large amounts of training data, to work well in a wide range of environments and situations. Hand annotating this data can require hundreds of man hours. It is impractical to annotate every frame (30 frames/sec x 60 sec/min x 60 min/hour x 5 seconds/annotation = 150 hours of annotation/hour of video) or even every few frames.

The DigiHead system eliminates the need for the majority of these hand annotations by extrapolating annotations based on a few hand annotations and the movement data. It classifies regions of the video frames as safe, unsafe or possibly safe. The system is able to, on average, detect movement with a 5.55% error rate. With this accurate movement detection, it is able to consistently produce annotations that are at least 90% accurate in the majority of situations. Accuracy and completeness can be tuned to specific needs by configuring the threshold value and sub-sample rate. Using DigiHead gives the robot hundreds, if not thousands of hours of prior knowledge by simply driving it in various environments, giving the robot the information that it needs to generalize its obstacle detection ability to a new environment.

The significant contribution of this work is demonstrating that human actions can be used to demonstrate to a computer/software system how to annotate video with a high degree of accuracy. DigiHead also demonstrates that visual movement detection is possible under many conditions. Additionally it can be performed with inexpensive off the shelf hardware.

6.1 - Possible Future Work

While the DigiHead system performs well, there is room for improvement, both in the area of movement detection and automatic annotation.

It has been shown that under some circumstances the movement detection technique using interesting points does not perform well. One example is that when the objects are too close to the camera, the movement of these objects does not represent the movement of the vehicle. A potential approach to overcoming some this is to use a laser range finder to indicate which areas of the field of view are further away. As distance objects give a better visual movement reading, these areas could be given more weight or a cutoff could be used (only areas a certain distance are used). Another example is that when the field of view is too uniform, the movement detector is often unable to detect sufficient interesting points to accurately measure movement. Additionally it sometimes is unable to match these up as the interesting points that are detected are all very similar.

DigiHead could be tested with better or more cameras. Higher resolution data could give the Interesting Point algorithms the ability to overcome some of the limitations highlighted in “4.7.3 -Visual Point Tracking”. The “uniform” grass and trees might, at higher resolution, yield more distinguishable interesting points. Additionally the human annotator could give more accurate sample annotations to construct the PGrids.

The anticipated use of this system is to use the annotation data and create a classifier. The annotations could be used by the classifier as discrete input using a threshold value or as continuous data using the probability or confidence level for each PGrid region. The resulting classifier could then be used to classify video frames and then feed that information into an autonomous or semi-autonomous robot or vehicle driving system. This would free the human driver to focus on high level tasks and goals which is the overall motivation for this project.

Bibliography

[ARKI91] R. C. Arkin, *Reactive Control as a Substrate for Telerobotic Systems*, IEEE AES Systems Magazine, vol. 6, no. 6, pg. 24-31, June 1991

[ARTH04] R. Arthur, *Vision-based Human Directed Robot Guidance*, Master's Thesis, BYU Computer Science Department

[BORE89] J. Borenstein and Y. Koren, *Real-time Obstacle Avoidance for Fast Mobile Robots*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 5, pg. 1179-1187, September/October 1989

[BORE91] J. Borenstein and Y. Koren, *The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots*, IEEE Journal of Robotics and Automation, vol. 7, no. 3, pgs. 278-288, June 1991

[CHO00] J. T. Cho and B. H. Nam, *A Study on the Fuzzy Control Navigation and the Obstacle Avoidance of Mobile Robot Using Camera*, IEEE International Conference on Systems, Man, and Cybernetics 2000, vol. 4, pgs. 2993-2997

[DAVI95] A. J. Davison, I. D. Reid, D. W. Murray, *The active camera as a projective pointing device*, Proc. 6th British Machine Vision Conference, 1995

[DAVI98] A. J. Davison, *Mobile Robot Navigation Using Active Vision*, PhD Thesis, Robotics Research Group, University of Oxford, 1998. Available at http://www.robots.ox.ac.uk/~ajd

[FAIL03] J. A. Fails and D. R. Olsen, *A Design Tool for Camera-based Interaction*, CHI '03, 2003

[FONG01] T. Fong, S. Grange, C. Thorpe and C. Baur, *Multi-robot remote driving with collaborative control*, IEEE International Workshop on Robot-Human Interactive Communication, September 2001

[FOX97] D. Fox, W. Burgard and S. Thrun, *The Dynamic Window Approach to Collision Avoidance*, IEEE Robotics and Automation Magazine, vol. 4, no. 1, March, 1997

[JENS99] P. Jensfelt and H. I. Christensen, *Laser Based Pose Tracking*, IEEE Intl. Conference on Robotics and Automation, May, 1999

[KATO02] T. Kato, Y. Ninomiya and I. Masaki, *An Obstacle Detection Method by Fusion of Radar and Motion Stereo*, IEEE Transactions on Intelligent Transportation Systems, vol. 3, no. 3, pg. 182-188, September 2002

[KO03] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, *A practical, decision-theoretic approach to multi-robot mapping and exploration*, 2003 IEEE/RSJ International Conference on

Intelligent Robots and Systems Proceedings, IROS 2003, vol. 4, pg. 3232-3238, 27-31 Oct. 2003

[KONO03] K. Konolige, D. Fox, B. Limketkai, J. Ko, B. Stewart, *Map merging for distributed robot navigation*, 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems Proceedings, IROS 2003, vol. 1, pg. 212-217, Oct. 27-31, 2003

[KWON95] Y. D. Kwon and J. S. Lee, *An Obstacle Avoidance Algorithm For Mobile Robot: The Improved Weighted Safety Vector Field Method*, 1995 International Symposium on Intelligent Control, pg. 441-446, 1995

[LANG99] S. Lang, F. Yili and S. K. Tso, *Visual Correction of Orientation Error for a Mobile Robot*, IEEE International Conference on Intelligent Robots and Systems, 1999

[LEVI99] S. P. Levine, D. A. Bell, L. A. Jaros, R. C. Simpson, Y. Koren and J. Borenstein, *The NavChair Assistive Wheelchair Navigation System*, IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, pg. 443-451, December 1999

[LI03] H. Li and S. X. Yang, *A Behavior-Based Mobile Robot With a Visual Landmark-Recognition System*, IEEE/ASME Transactions on Mechtronics, vol. 8, no. 3, pg. 309-400, September 2003

[MING00] J. Minguez and L. Montano, *Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach*, Internal Report RR-00-14 University of Zaragoza, pg. 60, February, 2000

[MORR03] A. Morris, R. Donamukkala, A. Kapuria, A. Steinfeld, J. T. Matthews, J. Dunbar-Jacob, and S. Thrun, *A robotic walker that provides guidance,* 2003 IEEE International Conference on Robotics and Automation Proceedings. ICRA '03, vol. 1, pg. 25-30 14-19 Sept. 2003

[MURR96] D. W. Murray, I. D. Reid and A. J. Davison, *Steering and Navigation Behaviours using Fixation*, Proceedings of the 7th British Machine Vision Conference, pg. 634-644, 1996

[NUCH04] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun, *6D SLAM with an application in autonomous mine mapping*, 2004 IEEE International Conference on Robotics and Automation Proceedings. ICRA '04, vol. 2, pg. 1998-2003, 26 April – 1 May 2004

[OHYA98] A. Ohya, A. Kosaka and A. Kak, *Vision-Based Navigation by a Mobile Robot with Obstacle Avoidance Using Single-Camera Vision and Ultrasonic Sensing*, IEEE Transactions on Robotics and Automation, vol. 15, no. 6, pg. 969-978, December, 1998

[OLSE04] D. R. Olsen, S. B. Wood and J. Turner, *Metrics for Human Driving of Multiple Robots*, International Conference on Robotics and Automation, April, 2004

[ROFE99] T. Rofer and A. Lankenau, *Ensuring Safe Obstacle Avoidance in a Shared-Control System*, in J. M. Fuertes (Ed.), Proc. Of the 7th Int. Conf. On Emergent Technologies and Factory Automation, pg. 1405-1414, 1999

[ROY99] N. Roy, W. Burgard, D. Fox, and S. Thrun, *Coastal navigation-mobile robot navigation with uncertainty in dynamic environments*, 1999 IEEE International Conference on Robotics and Automation Proceedings, vol. 1, pg. 35-40, 10-15 May 1999

[SCHU00] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A. B. Cremers, *Web Interfaces for Mobile Robots in Public Places*, IEEE Robotics & Automation Magazine, vol. 7, no. 1, pg. 48-56, March 2000

[THRU00] S. Thrun, W. Burgard, and D. Fox, *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping*, IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00, vol. 1, pg. 321-328, 24-28 April 2000

[THRU04] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, *A Real-Time Expectation-Maximization Algorithm for Acquiring Multiplanar Maps of Indoor Environments With Mobile Robots*, IEEE Transactions on Robotics and Automation, vol. 20, no. 3, pg. 433-442, June 2004

[TURN04] J. Turner, *Obstacle Avoidance and Path Traversal Using Interactive Machine Learning*, Master's Thesis Proposal, BYU Computer Science Department

[VIOL01] P. Viola, M. Jones., *Rapid Object Detection Using a Boosted Cascade of Simple Features*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), ISSN: 1063-6919, Vol. 1, pp. 511-518, December 2001

[WU02] T. Wu and H. He, *Learning Based Obstacle Detection with Uncalibrated Cameras*, IEEE Proceedings of the First International Conference on Machine Learning and Cybernetics, 4-5 November 2002

[ZHAN94] Z. Zhang, *Iterative Point Matching for Registration of Free-Form Curves*, International Journal of Computer Vision, vol. 13, no. 2, pg. 119-152, October, 1994