UNSTRUCTURED ROAD DETECTION IN COLOR IMAGERY FOR THE PURPOSE OF THE AUTOMATIC DETECTION OF EXPLOSIVE DEVICES

A THESIS

PRESENTED TO

THE FACULTY OF THE GRADUATE SCHOOL

AT THE UNIVERSITY OF MISSOURI

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

BY

DAVID BARCLAY LEWIS

DR. JAMES KELLER, THESIS SUPERVISOR

DECEMBER 2012

THE UNDERSIGNED, APPOINTED BY THE DEAN OF THE GRADUATE SCHOOL,

HAVE EXAMINED THE THESIS ENTITLED

UNSTRUCTURED ROAD DETECTION IN COLOR IMAGERY FOR THE

PURPOSE OF THE AUTOMATIC DETECTION OF EXPLOSIVE DEVICES

PRESENTED BY DAVID BARCLAY LEWIS

A CANDIDATE FOR THE DEGREE OF

MASTER OF SCIENCE

AND HEREBY CERTIFY THAT, IN THEIR OPINION, IT IS WORTHY OF ACCEPTANCE.

_____

DR. JAMES KELLER

_____

DR. MIHAIL POPESCU

_____

DR. TONY HAN

To Mr. Kenneth Kolwyck, Ms. Tammy Popp, and Ms. Ellen Downey for helping me to realize I have what it takes, and to my parents for helping me every step of the way.

ACKNOWLEDGEMENTS

There are a number of people without whom this thesis may not have been written, and to whom I am greatly indebted. First I would like to thank Dr. James Keller for being my advisor, mentor, and teacher, and for granting me my position as a research assistant. I would like to thank Dr. Tony Han for being on my committee and for what I have learned from his classes. I would also like to thank Dr. Mihail Popescu for being on my committee and for his help and guidance over the last two years. Dr. Derek Anderson has also helped me in my transition from my undergraduate to graduate career, and guided me in my first year as a researcher.

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

ABSTRACT

This thesis proposes a method for segmenting an unstructured dirt road in color space images using color and texture analysis. A support vector machine (SVM), or a Random Forest classifier is trained on samples of on and off road patches from a similar road. Image patches are classified at an interval of 10 pixels at a fixed horizontal distance from the vehicle. Each patch is described by the Histogram of Oriented Gradients (HOG), the Local Binary Patterns (LBP), a histogram of the three color channels, and a set of statistics are calculated on the color histograms. The classified patches are transformed to the next frame of the sequence using the scale invariant feature transform (SIFT) to reduce classification of image patches that have been classified in previous frames. Morphological opening and closing are used to transform the points into a mask, and reduce errors. Further error reduction and smoothing of the boundary is made possible by following the corners of the positive road classification with a Kalman filter. Experimental results measured against a hand segmented ground truth indicated that the algorithm can accurately segment road images given a set of training data from similar road utilizing only color imagery.

1. INTRODUCTION

1.1 OVERVIEW OF PROBLEM

Image based road segmentation has been an active research area in computational intelligence, and computer vision for many years. Traditionally, autonomous vehicles and robots are required to follow roads and detect obstacles. More extensive work has been done on well-maintained roads, but unpaved, dirt roads have not been as thoroughly explored. These roads' appearances vary vastly from one area to the next, thus posing more challenges for a solution. Off road navigation has many applications in military and civilian areas [1] [2] [3].

This segmentation problem is different than traditional road detection problems. In this case, the vehicle has a driver. The visual cues are more ambiguous because there are no lane markings, and the road is poorly defined at times. Road color and texture features provide the information in this detector, but their appearance can vary greatly.

Segmenting roads is normally done by three methods: threshold, region growing, and feature-based segmentation [4] [5]. This paper will focus on the feature-based segmentation.

Explosive ordinance detection is a computationally expensive task. With the implementation of a simple road segmentation algorithm the complexity of the detection suite can be reduced. This simultaneously reduces the false alarm rate of individual detection methods by directing their attention to their intended search space. Different algorithms are used for above and below ground targets. With a road detector above

ground detectors can be directed to look at the sides of the road, and in ground detectors can be directed to look on the road.

In the past, the forward-looking explosive hazard research team at University of Missouri-Columbia used a fixed mask as a bound to the road area. This is effective only in the case that the road does not turn and maintains roughly the same width. This may be acceptable for the case of training algorithms, and learning from the data, but in theater the roads will not be straight lanes. Figure 1.1 displays two typical roads in our collection. Because of this, a dynamic road map is of paramount importance to the problem at hand [6] [7] [8] [9].

Since the frames of the video of the road are ordered, it can be assumed that the information in the images only slightly changes from frame to frame. This allows a spatial transform such as SIFT [10] to be utilized to track points through time and frames rather than treating each frame as a completely new and independent data set [11].

A hand segmented ground truth is utilized to evaluate the effectiveness of the algorithm.

Figure 1.1: A well formed dirt road, and a poorly formed dirt road from an arid test lane.

2. BACKGROUND AND RELATED WORK

2.1 TEXTURE FEATURE ANALYSIS

Various texture and color features will now be examined. These features will be utilized to describe road patches in chapter 3 where the system design is documented in detail.

2.1.1 LOCAL BINARY PATTERNS (LBP)

Local binary patterns are a computationally inexpensive method of describing the local variation in contrast and gray-level in a neighborhood around a pixel. LBPs are illumination invariant since the actual gray levels involved are only relative. The same texture under different illumination in a different image should exhibit the same LBP description [12].

The LBP texture descriptor is defined on P pixels at a radius R from the center of the image patch, C. If the center pixel C has a gray level g_c then the $LBP_{P,R}$ value is given by equation (2.1) and (2.2). The values of g_i are the gray levels of the P neighbors around center pixel C.

$$LBP(C) = \sum_{i=0}^{P-1} s(g_i - g_c)2^i \qquad (2.1)$$

$$s(x > 0) = 1, s(x < 0) = 0 \qquad (2.2)$$

Each of the P neighbors, 8 in this case, are assigned a value of 0 or 1. A histogram containing 10 bins is then created over the image patch.

Another feature of LBPs is that they can be rotation invariant. The LBP operator produces 2^P binary patterns that can be formed by the set of neighbors. To remove the effect of rotation from each pattern equation (2.3) is defined:

$$LBP_{P,R} = \min\{ROR(LBP_{P,R}, i) \mid i = 0,1,..., P-1\} \qquad (2.3)$$

Where $ROR(x,i)$ is the bit-wise shift on the P-bit number x, i times. Two identical textures can have the same bit pattern, but it could be rotated by some number of bits. The modification in equation (2.3) ensures that the bits are in the least significant position.

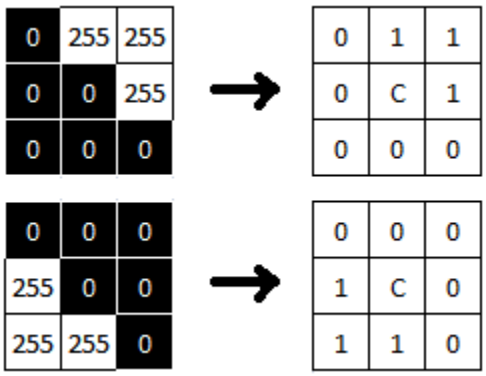

Figure 2.1: An example of two local binary patterns with the same rotation invariant value. The top image has an LBP of 01110000 and the bottom has 00000111 when started from the top left corner. The rotation invariant value comes to 00000111 for both of the patterns.

The rotation invariant approach introduces problems, and discards some data. To fix this problem the Uniform LBP was created. This adds a feature that counts the number of 0 to 1 transitions. If the number is less than or equal to 2 the integer label is set to the number

of 1s in the sequence. In Figure 2.1 it would be 3 in both cases. This is useful for patterns centered on an edge, or on a corner.

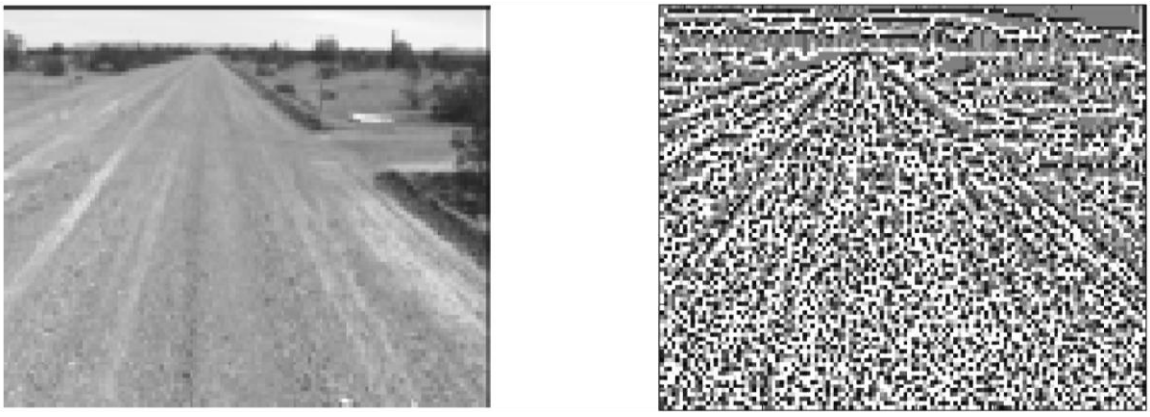

Figure 2.2: LBP applied over a whole image where the pixel values in the original have been replaced by the binary pattern value to for an LBP image.

To generate the LBP feature for an image patch to give to a classifier, the histogram of values from the LBP image is computed. The histogram has P+1 bins where P is the number of patterns possible with the given neighborhood size, and the P+1 bin represents the number of occurrences where there are more than 2 transitions.

LBP can be taken over an entire image. In Figure 2.2 an example of an LBP of an entire image can be seen. Normally, it is just computed on an image patch returning an image where the center pixels are set to the integer value of the binary number, or a histogram of all the values is generated as a feature [13]. Figure 2.3, and Figure 2.4 show an example of a histogram for an on, and off road sample from the test data.

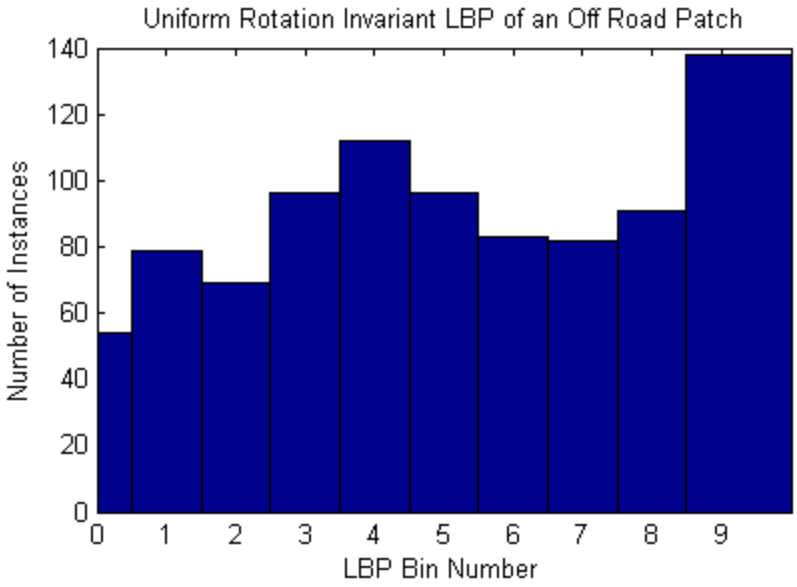

Figure 2.3: A histogram output of the LBP features for an off road patch in Figure 2.2.

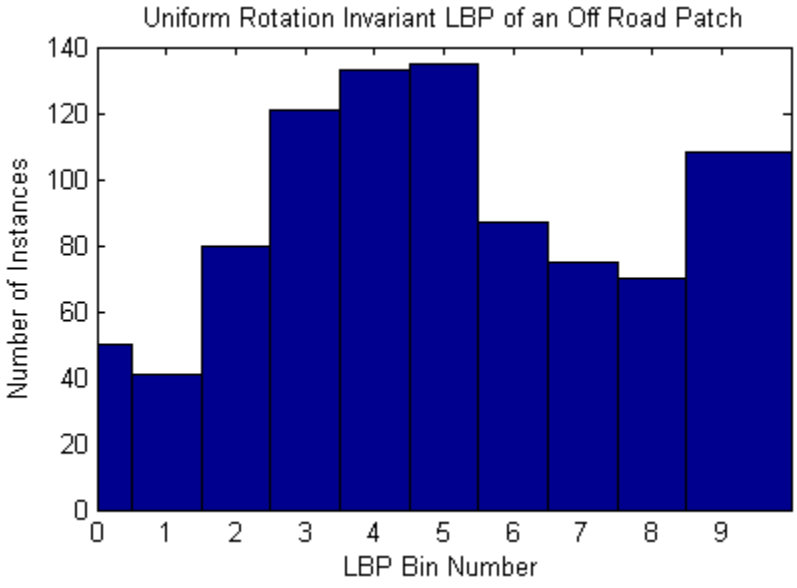

Figure 2.4: histogram output of the LBP features for an on road patch in Figure 2.2.

2.1.2 HISTOGRAM OF ORIENTED GRADIENTS (HOG)

The Histogram of Oriented Gradients (HOG) feature descriptor is used in computer vision for the purpose of object recognition and detection. One downfall to the descriptor is that it is dependent on the object orientation. The reasoning behind the HOG is that a human can still differentiate images by the edges and their orientations alone. In Figure 2.5 anyone who has seen a guitar before can easily recognize it.

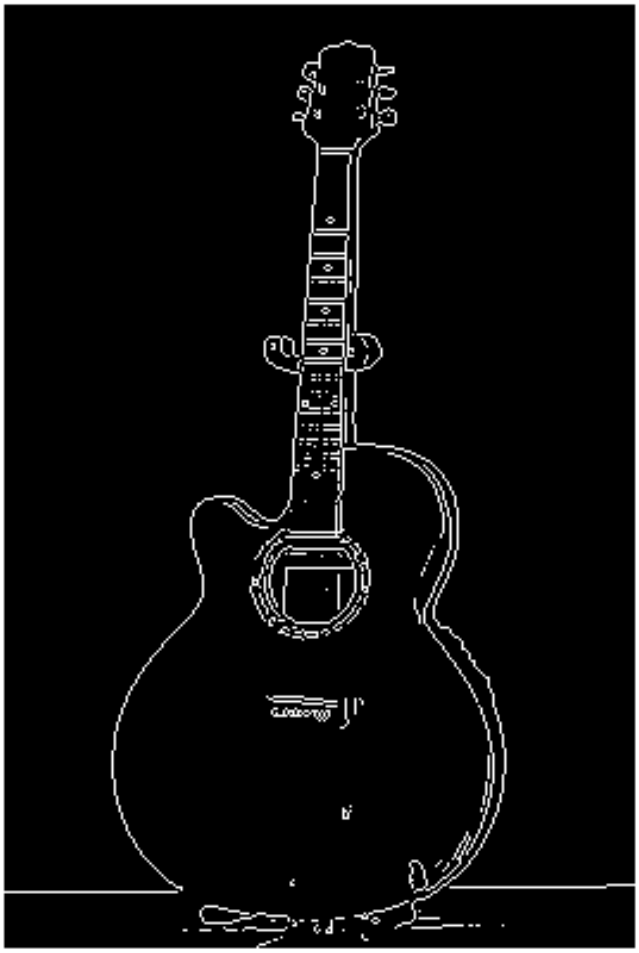

Figure 2.5: An edge image of a guitar.

One method of edge detection is to find the gradient vector by approximating the partial derivatives at each pixel location.

$$\nabla I(x,y) = \left(\frac{dI}{dx}, \frac{dI}{dy} \right) \tag{2.4}$$

$$\frac{dI}{dx} = I(x-1) - I(x+1) \tag{2.5}$$

$$\frac{dI}{dy} = I(y-1) - I(y+1) \tag{2.6}$$

The gradient is computed by filtering the patch with $[-1,0,1]$ and $[-1,0,1]^T$ kernels to get the horizontal and vertical gradients, respectively. This produces two images of the x and y gradients. The two images are combined using (2.7), and the gradient direction is calculated with (2.8).

$$|\nabla I| = \sqrt{\left(\frac{dI}{dx} \right)^2 + \left(\frac{dI}{dy} \right)^2} \tag{2.7}$$

$$\theta = \tan^{-1} \left(\frac{\left(\frac{dI}{dy} \right)}{\left(\frac{dI}{dx} \right)} \right) \tag{2.8}$$

After calculating the gradient image, statistics are generated based on the magnitude and orientation of the gradients at each point in the image. These statistics will hopefully be unique to particular textures. The HOG descriptor divides each image, or patch into cells,

and builds a histogram of the gradient directions in each cell. The magnitude of the gradient is used to weight the cell. The final value for each cell histogram bin b is the sum of the gradient magnitudes w for the directions d that map to that bin. If n pixels have gradient orientations that fall into the range for bin b.

$$b = \sum_{i=1}^{n} w_i \tag{2.9}$$

After calculating cell histograms they are combined into blocks. The individual blocks are normalized by taking the L1-norm as in (2.10) with ε set to a small value as suggested in [14] by Dalal.

$$v_N = \frac{v}{\|v\|_1 + \varepsilon} \tag{2.10}$$

The feature vector is the concatenated histograms of all of the cells. A HOG histogram with 9 bins, and 9 cells will have 81 features [13].

2.1.3 COLOR HISTOGRAMS

Color channel histograms are the simplest feature used to describe image patches in this method for road detection. In each color channel of the image patches, the values are collected into 10 discrete bins. Each bin of the resulting histogram is then treated as a feature.

As an extension to the histograms a measure for the mean, standard deviation, skewness, energy, and entropy of the histogram is also calculated [15].

The skew of the histogram is a measure of the asymmetry of the histogram about the mean. If the histogram spreads more to the right it will be positive, and to the left it will be negative, and is described in (2.11).

$$SKEW = \frac{1}{\sigma_g^3} \sum_{g=0}^{L-1} \left(g - \bar{g}\right)^3 P(g) \tag{2.11}$$

Another method for measuring the skew exists that uses the mean, mode, and standard deviation, and is described in equation (2.12).

$$SKEW = \frac{\bar{g} - MODE}{\sigma_g} \tag{2.12}$$

The energy measure is a value between 0 and 1 where 1 is a histogram with all bin values equal. The energy measure is calculated with equation (2.13). The entropy is a measure of how many bits are needed to code the histogram. The more values the histogram contains, the higher the entropy will be. This measure varies inversely with energy, and is calculated with equation (2.14)

$$ENERGY = \sum_{g=0}^{L-1} [P(g)]^2 \tag{2.13}$$

$$ENTROPY = -\sum_{g=0}^{L-1} P(g) \log_2 [P(g)] \tag{2.14}$$

2.2 IMAGE TRANSFORMS

To convert data from one image to the next, image transforms were used. The two methods described are Scale Invariant Feature Transform, and Speed Up Robust Features.

2.2.1 SCALE INVARIANT FEATURE TRANSFORM (SIFT)

The Scale Invariant Feature Transform (SIFT) is the most widely utilized method for extracting features from images that can be matched between scenes taken from different viewpoints [10]. In the case of road detection the vehicle is changing its viewpoint as it moves forward down the lane.

The algorithm has several stages. The first stage is extracting key points of interest, and creating feature vectors from them to describe the image. The features that are extracted are invariant to scale, rotation, and viewpoint. Key points are found by running the image through several difference-of-Gaussian functions to find interest points that are invariant to scale. Using (2.17) where k is a multiplicative factor, scale-space extrema are found.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.15}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp(-(x^2 + y^2)/2\sigma^2) \tag{2.16}$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.17}$$

Extrema are located by comparing a pixel with its 8 neighbors, and the 9 neighbors in the octave above, k=2, and below k=1/2.

Key points found by this method are then checked for stability. For example, a key point that has low contrast is susceptible to noise, and is thrown out. The function $D(\hat{x})$ is *the function value at the extremum*, and is given by (2.18). This value is used to reject extrema with low contrast. Lowe suggests throwing out values less than 0.03 [10].

$$D(\hat{x}) = D + \frac{1}{2}\frac{\delta D^T}{\delta x}\hat{x} \tag{2.18}$$

Key points that are from an edge response are also thrown out. To deal with this, a 2x2 Hessian matrix is calculated at the location and scale of the key point using (2.19).

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.19}$$

The ratios of the eigenvalues are used to find the ratio of principal curvatures that are below some threshold, r [16], and is computed by (2.22) using (2.20) and (2.21). A value of 10 is suggested by Lowe in [10].

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{2.20}$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{2.21}$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{2.22}$$

The orientation and magnitude values for the Gaussian smoothed image is calculated in exactly the same way as the HOG in (2.23) and (2.24).

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (2.23)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \qquad (2.24)$$

The orientation histogram is formed from this step, and the orientations are binned into 36 bins. That is 10 degrees per bin over 360 degrees.

The gradient magnitudes and orientations are sampled around the key point to form a histogram. Histograms are compared to neighboring histograms. The histograms are normalized to reduce the effects of contrast. Bin entries that are greater than 0.2 are replaced with 0.2 and the histogram is renormalized [10].

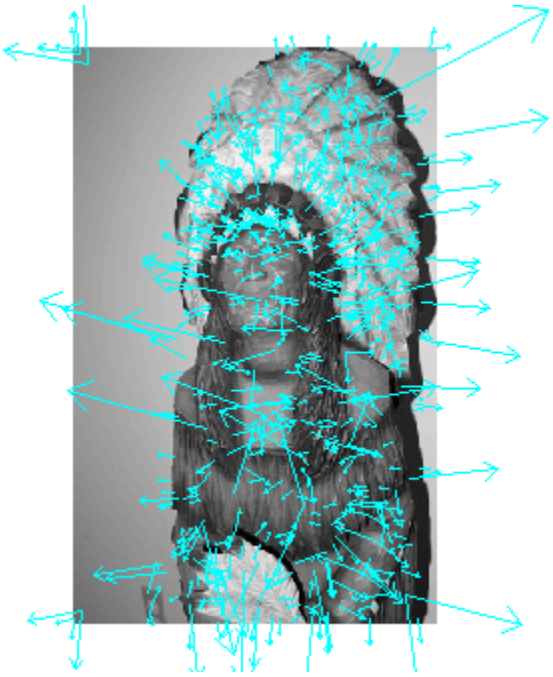

Figure 2.6: Here the key points orientations and their magnitudes can be seen on a wooden Indian by Lowe's SIFT demo [17].

The second stage of the algorithm matches the features found in the first image with features found in a second image of a different perspective. Key point histograms are compared directly, and if a histogram is close enough, it is then compared to the next closest key point. If a key point in question is close enough to two key points it is assumed that it is not clutter, and is an important feature in the images. This gives rise to a high complexity problem because every key point must be compared to every other key point.

The third stage is where the algorithm solves for the affine parameters from the matches. The affine solution requires at least 3 point matches, and can be calculated from (2.26). Each additional point adds two rows to the A, and b matrixes in (2.32)(2.25).

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \tag{2.25}$$

$$Ax = b \tag{2.26}$$

By representing (2.25) in the form of (2.26) the least-squares solution for x can be solved by the normal equation in (2.27).

$$x = [A^T A]^{-1} A^T b \tag{2.27}$$

After the matching step, outliers are removed by checking the distance from the projected location to the corresponding image location [10]. At least 3 points must remain after the check for the transformation to be valid. In Figure 2.7 an example of matched key points can be seen on two images of a wooden Indian with different perspectives.

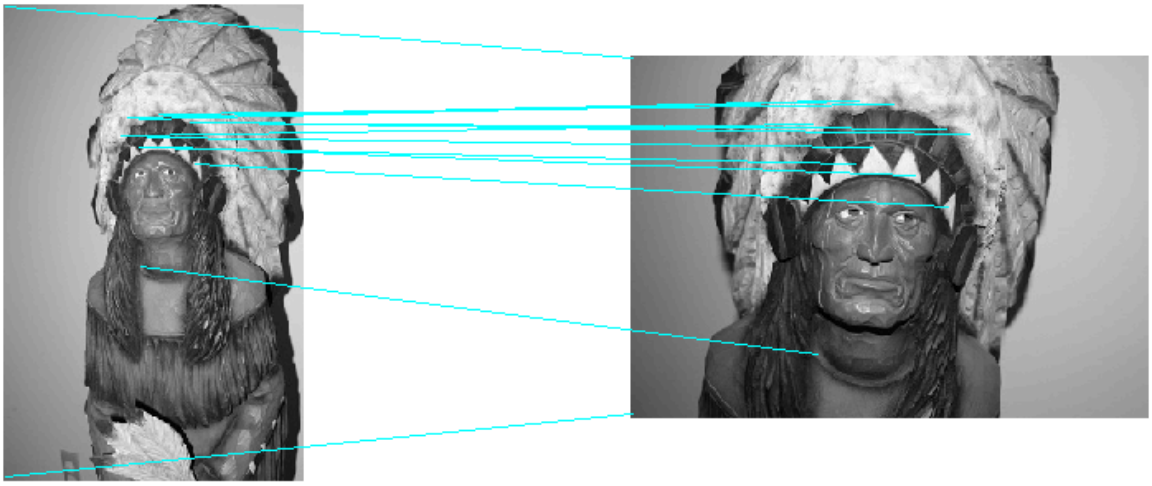

Figure 2.7: The key points matched on a wooden Indian between two viewpoints by Lowe's SIFT demo [17]. It can be seen that the majority of the matches are correct, but the left corners each have an incorrect match.

2.2.2 SPEEDED UP ROBUST FEATURES (SURF)

SURF is an alternative method to SIFT and is known for its speed increases over SIFT [18]. The way SURF achieves its speed increase over SIFT is through reduced descriptor dimensionality. There is also a performance increase that is situational. If it is known that rotation invariance is unnecessary the complexity of the descriptor is further reduced. The process for SURF is basically the same as SIFT. The three steps are interest point selection, interest point representation, and interest point matching between different images.

The detector is based on the Hessian matrix (2.29) where $L_{xx}(x,\sigma)$ is the convolution of a 2nd order Gaussian with the image I given in (2.28).

$$L_{xx}(x,\sigma) = \frac{d^2}{dx^2} g(\sigma) \tag{2.28}$$

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2.29}$$

$$\det(H) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.30}$$

The filters are suggested to be used in octaves where the filter size doubles from 6 to 12 and to 24 [18] with an initial scale of σ=1.2.

The orientation of interest points is calculated by first taking the Haar-wavelet response in the x and y directions in a radius of 6σ. After calculating the wavelet response a Gaussian of 2.5σ is centered on the interest point. The orientation is found by measuring the responses with a sliding window at angles of $\pi/3$, and summing the responses into a new vector. The orientation is the longest vector.

The descriptor is formed by breaking the region around the interest point into 4x4 bins. The wavelet response is calculated over each region in 4 orientations. The final feature vector is 4x4x4, or 64 features. This is half of the length of a SIFT feature vector. Wavelet responses are inherently immune to illumination bias. The contrast invariance is created by normalizing the feature vector from 0 to 1.

The distance between points in imagery is calculated the same way as SIFT. The Euclidean distance is taken from each key point descriptor to that of each interest point in the second image. The transform parameters are also calculated in the same way as the matched interest points described at the end of section 2.2.1.

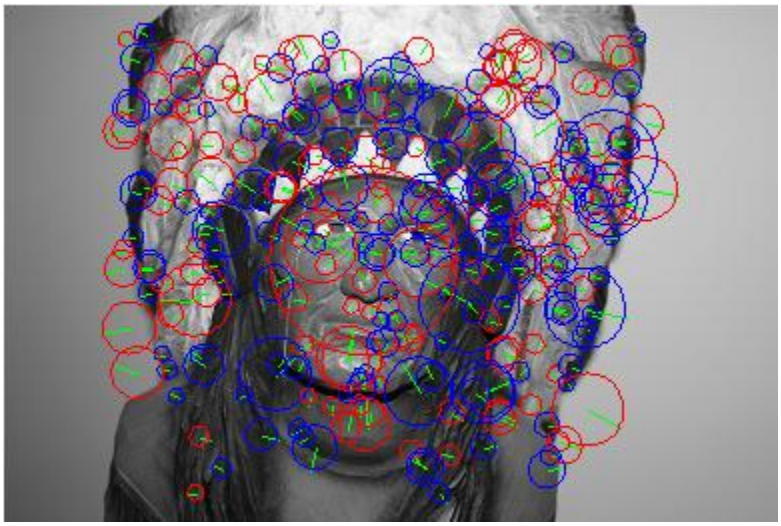

Figure 2.8: Interest points on a wooden Indian from the OpenSURF implementation of SURF. The orientation and magnitude can be seen by the direction of the green line, and the size of the circle.

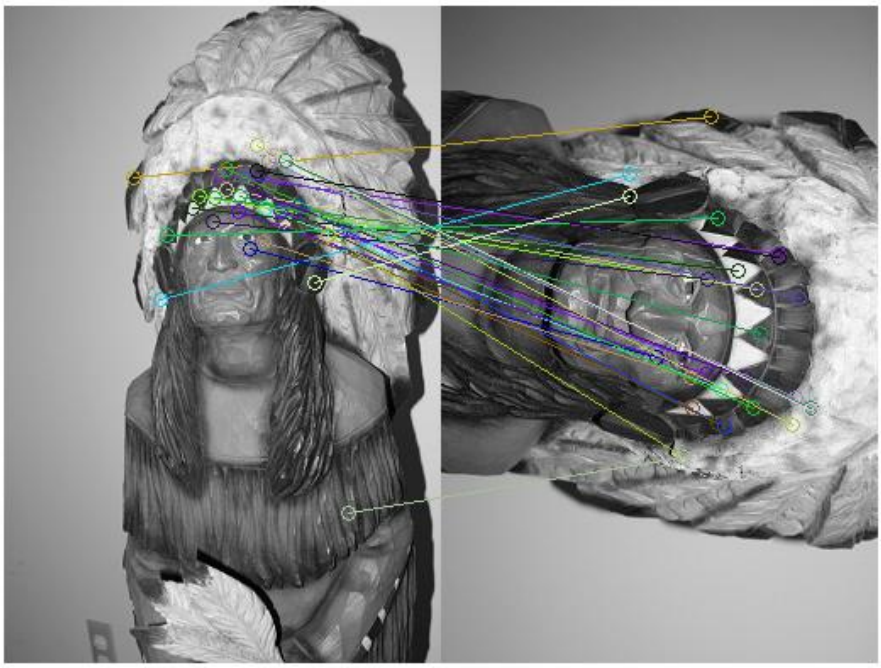

Figure 2.96: Interest points on a wooden indian matched by OpenSURF implementation of SURF. Here one bad match can be seen on the Indian's left arm.

In Figure 2.8 the location, orientation, and magnitude of key points can be seen on an image of a wooden Indian. Figure 2.96 shows matched key points on two different perspectives of the same wooden Indian.

2.3 CLASSIFIERS

After the texture, and color features are investigated and implemented, it is necessary to choose a classifier to determine if the patch is on, or off-road. The methods used in this thesis are Support Vector Machines, and Random Forests. Decision trees will also be covered because of their use in the Random Forest algorithm.

2.3.1 SUPPORT VECTOR MACHINE (SVM)

The Support Vector Machine is a supervised pattern recognition technique to classify patterns. The SVM constructs a high-dimension hyper plane boundary that maximizes the margin between classes. The traditional two class SVM labels its output classes as 1 for a positive case, and -1 for a negative case [19].

The hyper plane boundary can be defined in terms of:

$$w^T x + b = 0 \qquad (2.31)$$

w is the weight vector

x is a feature vector

b is the bias term that adjusts the intercept

The first goal of implementing an SVM is the placement of the dividing hyper plane. In the construction of the hyper plane during training the SVM places the boundary equidistant to the nearest points of each class. These points are known as the support vectors. In Figure 2.7 the support vectors can be seen circled in the 2D SVM diagram.

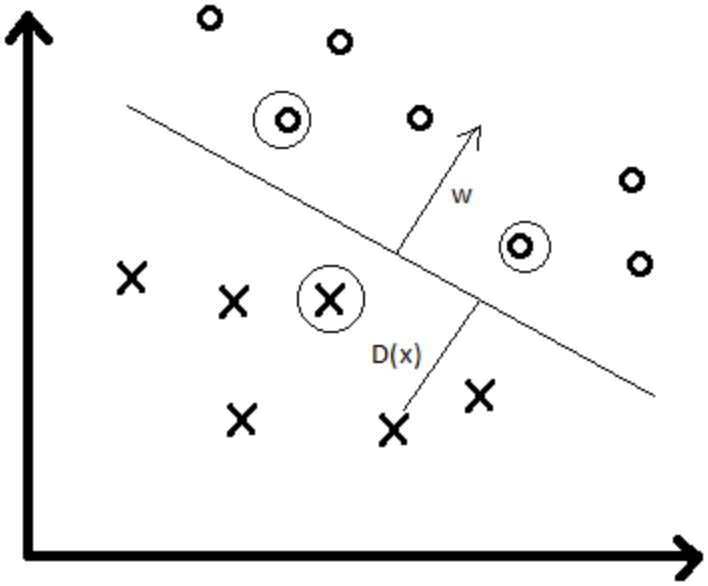

Figure 2.70: Diagram of 2D SVM. The support vectors are circled. Note that a data point entered behind a support vector would not change the margin.

Adding more data points that are not support vectors, the points which are not circled in Figure 2.70, do not change the position of the margin. This gives rise to a feature of the SVM that gives it an advantage over a classifier such as k-nearest neighbor which depends on all of the data points. The unneeded data-points can be discarded creating a sparse representation of the data. In Figure 2.7 it can be seen that w is the vector normal to the boundary, and $D(x)$ is the distance of a point to the boundary in the direction of w, and can be used as the confidence of the classifier. The output of the classifier is a function that takes the position of a data point in terms of w and b, as seen in equation (2.32) and assigns the data point a class depending on which side of the margin it is positioned.

$$D(x^{(i)}) = y^{(i)}\left(\left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}\right) \qquad (2.32)$$

$D(x^{(i)})$ is the *geometric margin* [19] and is the distance of each data point $x^{(i)}$ from the boundary. The objective of training is to maximize the margin. $y^{(i)}$ is the class label which is either -1 or 1. Therefore, when the class is negative the term $\left(\left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}\right)$ must also be negative.

Training the classifier requires the term w, and b be found to maximize the distance D(x). Solving this function, (2.35), can be done with quadratic programming to find the undetermined Lagrangian multipliers, α_i, subject to the constraints (2.33), and (2.34).

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \qquad (2.33)$$

$$a_k \geq 0 \qquad (2.34)$$

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \qquad (2.35)$$

Only the training points that qualify as support vectors will need to be summed over. This result is sparse in that many of the data points will not meet this qualification. The learned weights will be used to make predictions using (2.36).

$$w^T x + b \tag{2.36}$$

Substituting (2.36) into the SVM function results in (2.37) where $K(x^{(i)}, x)$ is called the kernel function. Any kernel can be used as long as it satisfies Mercer's theorem. The kernel must be symmetrical, and positive definite.

$$w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) \tag{2.37}$$

In a linear SVM it is simply the inner product of $x^{(i)}$ and x, but can be replaced with many functions. In this thesis the radial basis function (2.38), inhomogeneous polynomial (2.39), and sigmoid (2.40) functions will be used as kernels as well as the inner product. Each of the kernels have a scaling factor, γ, and (2.39), and (2.40) have a shifting factor, c.

$$K(x^{(i)}, x) = \exp\left(\gamma * \left\| x^{(i)} - x \right\|^2\right) \tag{2.38}$$

$$K(x^{(i)}, x) = (\gamma * x^{(i)} \cdot x + c)^d \tag{2.39}$$

$$K(x^{(i)}, x) = \tanh(\gamma * x^{(i)} \cdot x + c) \tag{2.40}$$

The introduction of the kernel takes the data into a higher dimensional space where it is hopefully linearly separable. When the dimensionality is brought back down to the lower dimensionality the mapping is non-linear. The ability for the SVM to train depends

heavily on the kernel function. The choice in which kernel is used will influence how well the SVM learns and generalizes [20].

Figure 2.8, and Figure 2.9 show how a linearly inseparable set of points from two classes can be mapped into a higher dimension by squaring their values, and using those values as a second set of features. This allows them to be separated linearly.

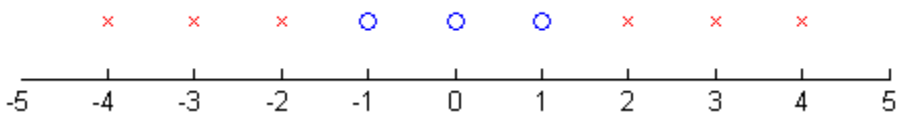

Figure 2.81: A linearly inseparable classification problem in one dimension.

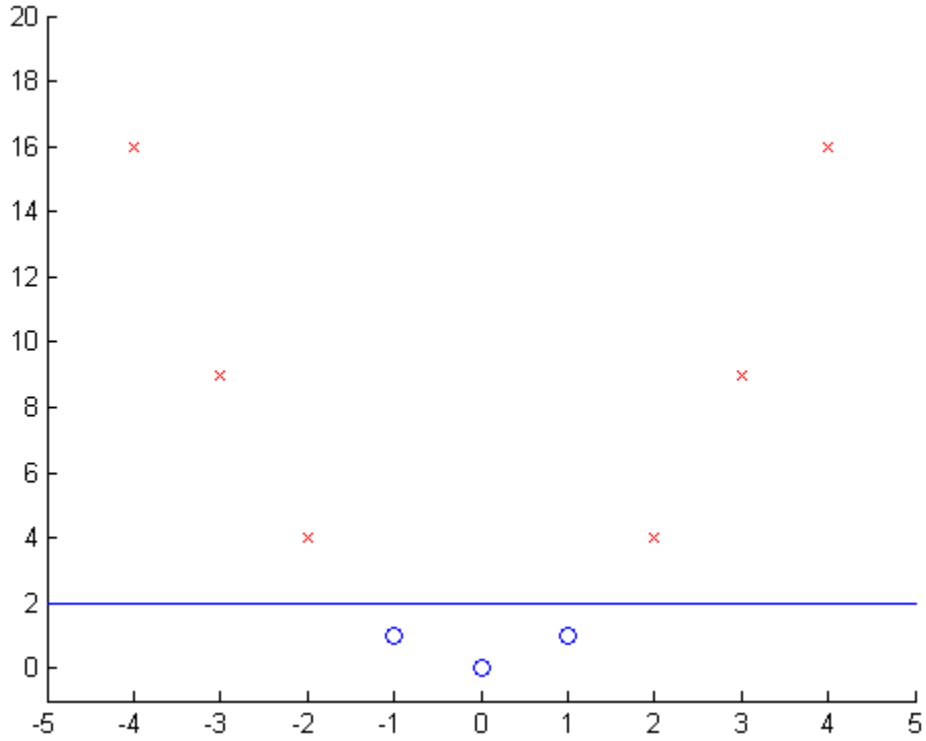

Figure 2.92: The non linearly separable classification in Figure 2.11 with a new dimension of the value squared added making the problem linearly separable in two dimensions.

Data may still be inseparable after being mapped into a higher dimension with a kernel. However, there is a way to deal with this possibility. This method is called the soft margin classifier. It deals with how the SVM handles misclassified points [21]. If the distance of a training point to the boundary is D(x) then the distance for the misclassified point to the boundary is ξ and is considered to be a negative margin. The value of the error is called the penalty, E, and takes the degree of misclassification into account. The error is calculated by summing the negative margins and multiplying them by a value of C which is the weight of error values.

$$E = C\sum_{i=1}^{N}\xi_i \tag{2.41}$$

Despite the SVM's sensitivity to the kernel type, and kernel parameters, it is an extremely effective classifier. It deals well with noisy, and high dimensional data with ease [13].

2.3.2 DECISION TREES (CART)

Decision trees are simple classifiers that are based around if-then conditions. Accurate classifications can be learned by decision trees. Decision trees are a desirable classifier because of their simplicity. Unlike an artificial neural network, or an SVM, the inner workings of CART are very easily visualized. CART can also classify problems with more than two classes, or categorical data. One of the problems with this classifier is that decision trees often over segment, and do not generalize data well [22]. This problem will be addressed by tree pruning, and also in section 2.3.3 Random Forest.

In CART there are several details that must be considered. First, the accuracy desired must be decided upon; second, how the splits are determined must be defined; and third, when to stop splitting must be defined. In a final step the tree is pruned back. Subsets created by a split are known as nodes, and each node contains a feature and a threshold value. Subsets which will not be split are known as terminal nodes, and will be where the class is assigned. Figure 2.103 shows a simple tree classifier. The circles are nodes which split depending on a threshold value. The squares are terminal nodes which contain the class label.

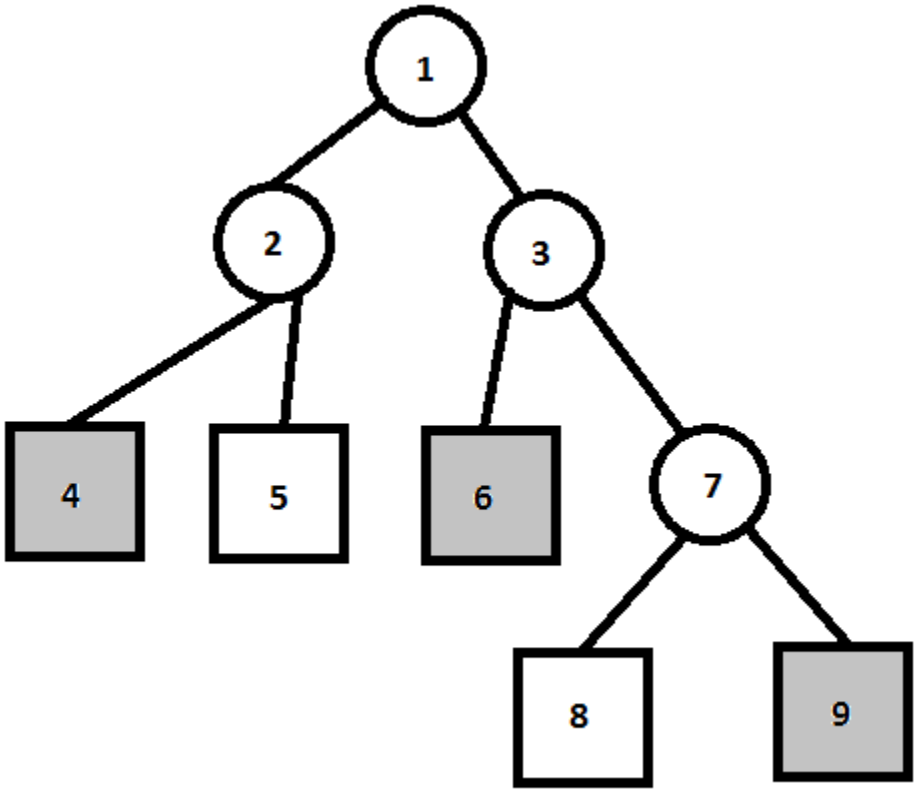

Figure 2.103: A tree structure. The circles are nodes, and the squares are terminal nodes. The gray boxes would be one class A while the white boxes are class B.

The first task is to decide where to split the data. CART utilizes the Gini impurity in (2.42). CART splits the data at the node based on how much it will lower this impurity value. The Gini impurity measures how often data points would be misclassified if they were randomly labeled by the distribution of labels in its subset.

$$i(p) = 1 - \sum_{i=1}^{n} p_i^2 \qquad (2.42)$$

The value p_i is the probability that the data point is labeled i in the subset with the classes (1,...n). This value is used to find the feature, and value that produces the greatest separation in the classification.

The tree can be grown until it reaches depth D, or the test data classification stops increasing. CART does not stop at a particular depth, but rather it stops when the terminal node is pure.

A problem with growing a tree to the maximum depth is that it over fits the data. While the training error continues to decrease, at a point the test data error begins to increase. This problem is illustrated in Figure 2.11. An example of a good fit and an over fit data set can be seen in Figure 2.6.

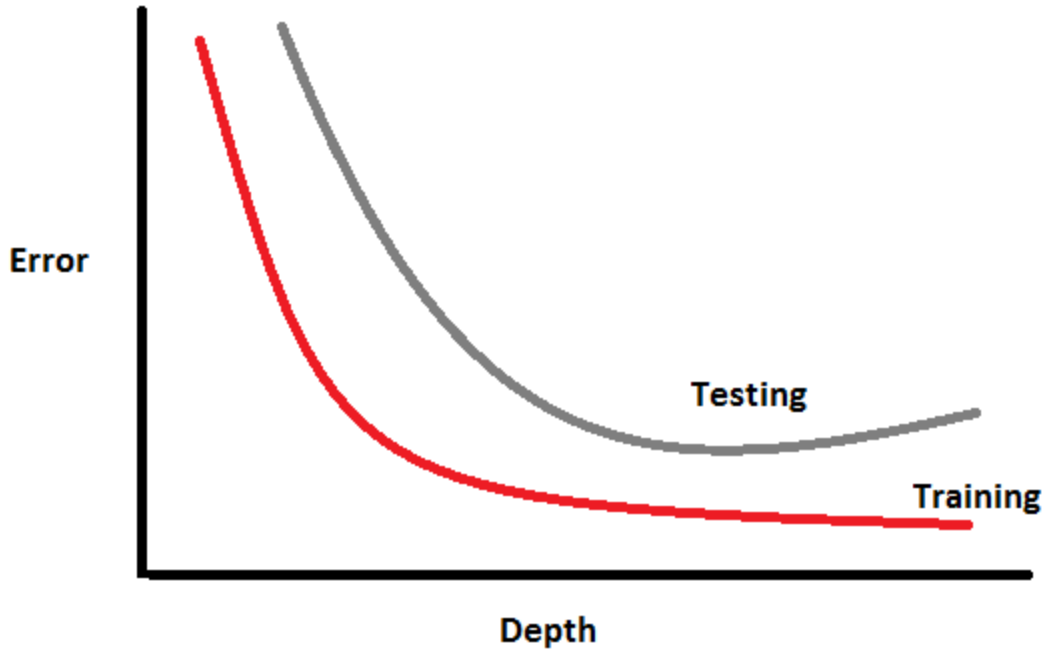

Figure 2.114: The training data error rate vs. the depth of the tree is in red. In gray it can be seen that even though the training error is decreasing the test data error rate is increasing because of over segmentation. A good time to stop increasing the depth of the tree would be when the test error rate stops decreasing.

After the tree is grown, pruning occurs. The misclassification rate is used to decide where the tree is pruned. The CART algorithm removes splits that give the lowest improvement to the impurity measure. The exact method is called *Cost-Complexity Pruning* [23]. In (2.43) the value MC is the number of misclassifications, and is a penalty scale for the number of leaves L. The objective is to find the sub tree with a minimal value of R_a.

$$R_a = MC + aL \tag{2.43}$$

The CART algorithm calculates the value of MC by using 10 fold cross validation. These values are then used to rank the sub trees using (2.43). The tree is pruned by removing

the weakest nodes until an acceptable error rate is reached. In the case of CART a happy medium is desired. This reduces over fitting, and maximizes performance.

Cart may not be used in practice frequently, but it does help to give a feel for data sets because of its ease of interpretation. It is easy to see which features contain the most information gain at node splits. It is also useful as a weak classifier in a bagging case like Random Forest, which will be discussed in the next section.

2.3.3 RANDOM FOREST

Random forest is a method of bagging decision trees that are each trained on a random subset of features selected with the same random distribution. It takes a large number of weak classifiers and creates a stronger classifier from them. In random forest each tree votes on the data presented to it, and the class with the most votes is assigned to the data. Because of the independence between trees of the classifier, it is easily parallelizable [24] [25] [26].

Some of the features of random forest are that it runs efficiently, gives feel for feature importance, the training data itself does not need to be saved, and cross-validation is unnecessary. Brieman conjectures that the error will converge without overfitting when adding trees to the classifier [26].

The random forest classifier consists of K tree structured classifiers $h_1(x,\Theta_1), h_2(x,\Theta_2)$,..., $h_K(x,\Theta_K)$. Each tree is grown in the same method of CART on a bootstrap set of the features $\Theta_k \in \Theta$. The pruning step, however is skipped. The bootstrap is a subset of the original training data. Each bootstrap of the features is independently selected. If, for

example, $\Theta = \{1,2,3,4,5\}$, then Θ_k could be {2,5,1}. A simple representation of a random forest is shown in Figure 2..

Figure 2.12: K tree classifiers with depth, D=3

As the trees are constructed the correlation between trees needs to be minimized. It has been shown that the lower the correlation is between the trees, the better the classifier will generalize and perform. Figure 2. shows an example of how over segmentation can cause poor classification. The point being classified in the middle should be classified as a square, but in the over segmentation example it is classified as a circle.

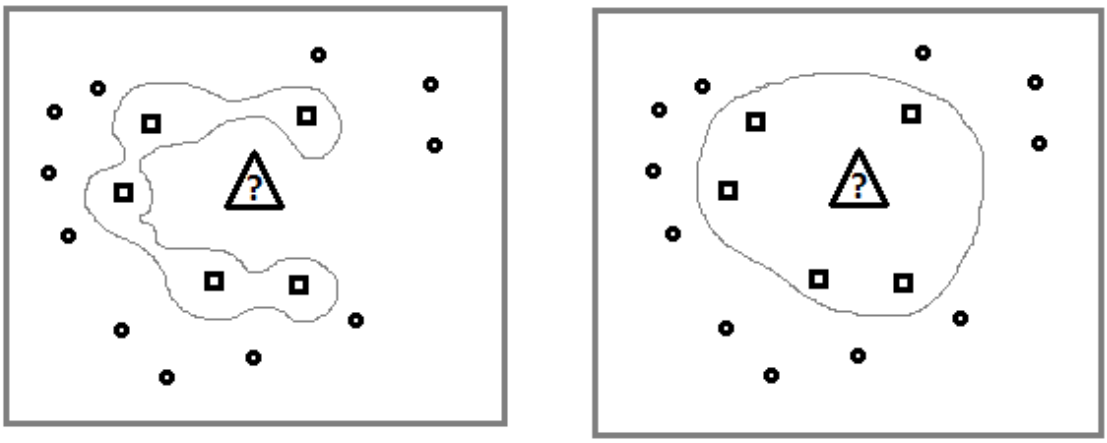

Figure 2.13: The left shows poor generalization, or over segmentation where the gray line is tight against the square points. The right shows good generalization where the grey line encircles all of the square points.

The margin is defined in (2.44). $h_k(X)$ is the classification of tree k on vector X. I is the indicator function, av is the average, and j is the class.

$$mg(X,Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j) \qquad (2.44)$$

A large margin translates directly to a higher confidence in classification.

The generalization error is defined in (2.45). Breiman proved in his paper 'Random Forests' that the generalization error converges with the addition of trees, and the classifier cannot over fit [26].

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \qquad (2.45)$$

Overall the random forest is a robust classifier that is not sensitive to tuning parameters. It is not picky about the features that it is given, and can easily ignore features that are unimportant, or contain no information.

2.4 BINARY IMAGE METHODS

In this section the methods used on the binary image created from positive classifications will be detailed. The methods used are morphological dilation, and the convex hull operation.

2.4.1 MORPHOLOGY

There are several mathematic morphological operations that have many uses for filtering binary images. The two basic operations are dilation, and erosion. From there an erosion followed by a dilation can be used to reduce noise in a binary image. In this thesis the only morphological operation used will be the morphological dilation.

2.4.1.1 DILATION

The dilation of binary image A by structuring element B is defined in the set operation (2.46) [27].

$$A \oplus B = \{z \mid (B)_z \cap A \neq \varnothing\} \tag{2.46}$$

Figure 2.147 shows a matrix representation of a binary image dilated by a structuring element, and the resulting matrix. Figure 2. shows the morphological dilation of an binary edge image dilated by a circle structuring element with radius 10.

0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0		0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0	0 1 0	0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 0 0 0 0	1 1 1	0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0 0	0 1 0	0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 1 0 0 0 0 0

Figure 2.147: The matrix of ones and zeros is dilated by the structure in the middle to result in the matrix on the right.

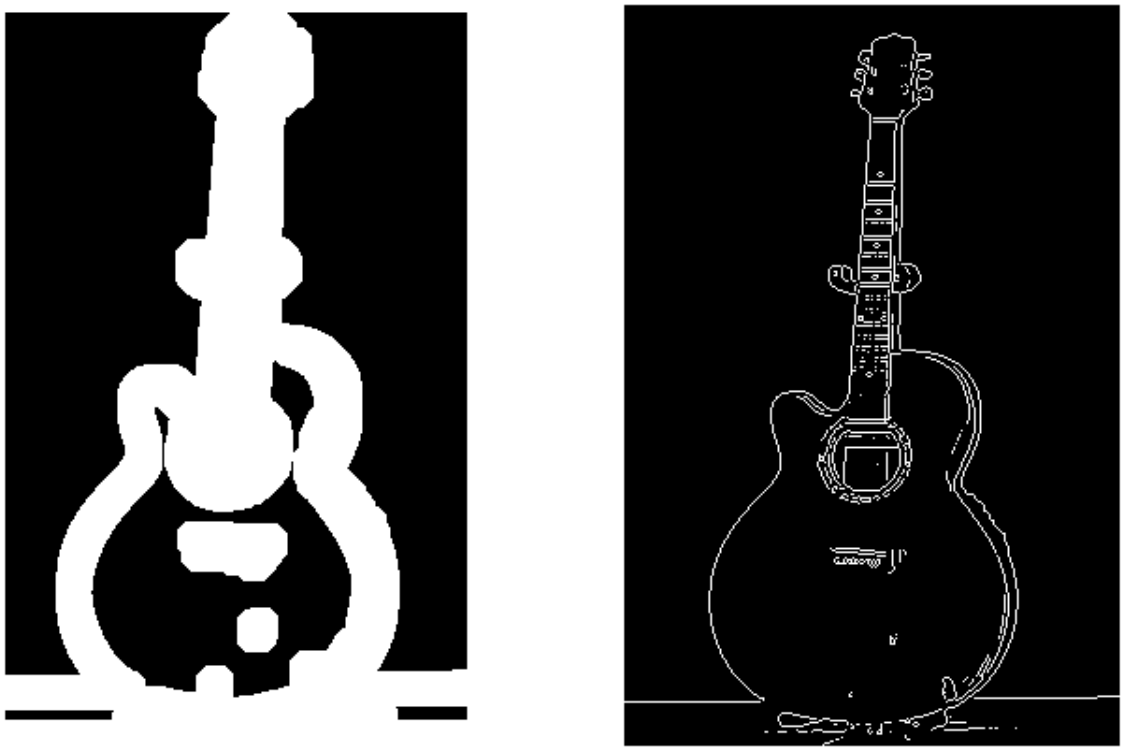

Figure 2.15: On the left the guitar outline image dilated by a circle of radius 10. On the right the original image.

2.4.2 CONVEX HULL

The convex hull of a set of points X is the smallest polygon Y that contains all points in X. An easy way to think about the convex hull is if the points were nails stuck in a board, and a string was drawn around them the polygon would be the shape of the string when it is pulled tight.

Figure 2.19: The left image shows 5 points in 2D image space. The right image shows the convex hull of the points.

The Quickhull convex hull is calculated by finding the left and right extrema of the set. A line is formed by these points, and then the farthest points are found from this line. The set is again partitioned, and the extrema points are again found based on the x component, and then y component each time recursively. When no more points are found the convex points in the hull have been located [28]. This process can be parallelized, and made to run in a fraction of a second on tens of thousands of points [29].

2.5 SMOOTHING AND FILTERING

The images analyzed in the road detector have causal data. This means that the next position of the road depends strongly on the last position found. If there is an error that causes the detector to jump an unreasonable distance a Kalman filter will help deal with that jump. In the next section the Kalman filter will be described.

2.5.1 KALMAN FILTER

The Kalman filter is a linear quadratic estimation algorithm, or a linear estimate of a quadratic system, that uses a series of noisy measurements and inaccuracies, to estimate unknown values. This method tends to be more accurate than a single measurement, and can run in real time using only the input measurements, and the previous output state. Traditionally the Kalman filter is used to estimate the position of aircraft, spacecraft, and vehicles, but it can be applied to any dynamic system with noise in the measurements.

There are two steps involved in a Kalman filter estimation: prediction, and updating. In these calculations only the previous estimation, and the current measurement matter [30].

The first step is the measurement update, or correction step. The Kalman gain, K_k, is the factor that minimizes the a posteriori covariance, P_k, and is calculated in (2.47) from the predicted posteriori covariance, P_k^-,the noise covariance matrix, R, and the observation model, H_k. The estimate of the position is updated in (2.48) with $\hat{x}_k^-$`, the a posteriori estimate, and the actual measurement, z_k. The error covariance matrix is updated in (2.49) with the a posteriori covariance, the Kalman gain, the observation model, and the identity matrix.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.47}$$

$$x_k = \hat{x}_k^- + K_k (z_k - Hx_k^-) \tag{2.48}$$

$$P_k = (I - K_k H) P_k^- \tag{2.49}$$

The second step is the time update, or prediction step. The next state, and the error covariance is predicted using A, the transition matrix.

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{2.50}$$

$$P_k^- = AP_{k-1}A^T \tag{2.51}$$

The filter is smoothed by running two filters. One predicts forward, and the other one predicts backwards in time. The estimates are averaged to gain a smoothed output [31] [32].

Figure 2.160, and Figure 2.171 show the Kalman filtered result of a noisy sine wave, and the smoothed Kalman result of the same noisy sine wave. The blue dots are the noisy measurements, the red line is the estimate by the Kalman filter, and the green line is the real signal. It can be seen in these figures that with a noisy sampling of data an accurate signal reconstruction can be obtained.

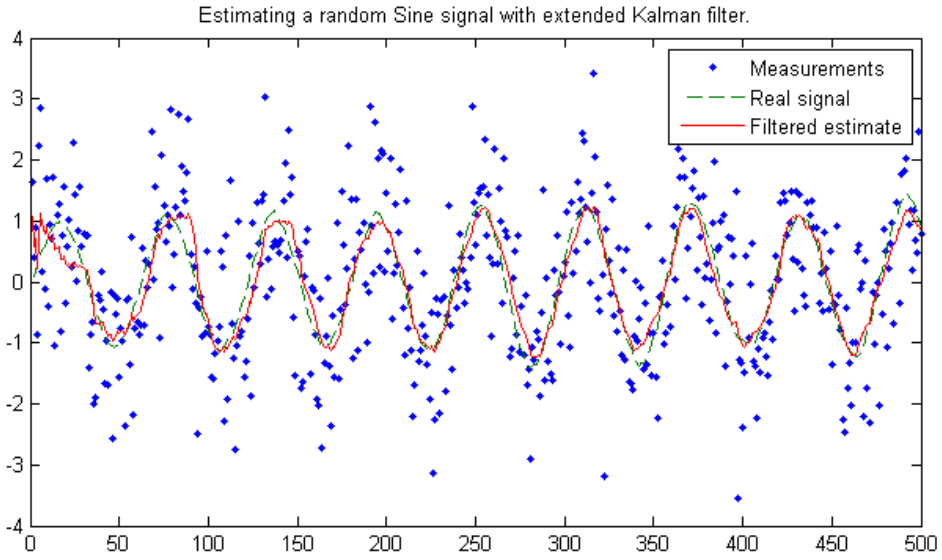

Figure 2.160: Kalman filtered noisy sine function [33].

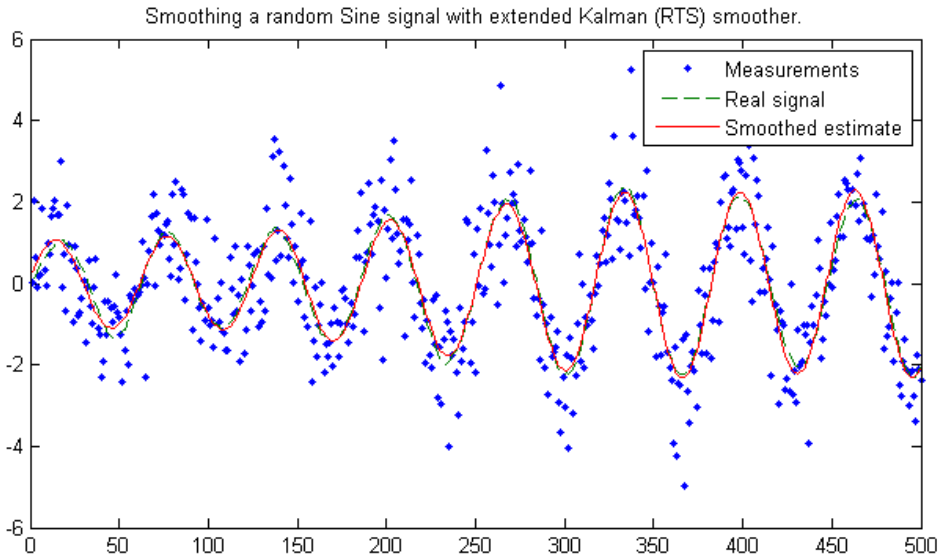

Figure 2.171: Kalman filtered output of a noisy sine function smoothed by forward and backward prediction [33].

2.6 OPTIMIZATION METHOD

The only method utilized for optimization of variables was the genetic algorithm. The genetic algorithm will be discussed in this section. It is the method of choice for this thesis because it can minimize a black box type function.

2.6.1 GENETIC ALGORITHMS

A genetic algorithm is a method of searching a space based upon the genetic process of evolution. It utilizes a fitness function, and chromosomes to minimize the function. The chromosome is a genetic representation of the solution. Each of the variables in the problem is a gene in the chromosome. There are four basic steps in a genetic algorithm: initialization, selection, reproduction, and termination [34] [35].

A set of initial chromosomes are populated with candidate solutions randomly. This can be represented by an N by M matrix where N is the number of chromosomes, and M is the number of variables in the solution. The population can also be partially initialized with answers that are known to be good. In this thesis preliminary tests were performed to find parameters to initialize some of the chromosomes.

Figure 2.182: A representation of a chromosome with five genes.

The second step in a genetic algorithm is selection. A fitness value is assigned to each chromosome in the population. In this implementation elitism was utilized. This means the most fit individual was carried over to the next generation. The next chromosomes

were selected by stochastic universal sampling. This is a method of fitness proportionate selection. The chromosomes are ranked by (2.52), and then selected at a fixed interval. [36]. This is achieved by mapping the chromosomes to a line of length 1, where each chromosomes segment length is equal to RP(c). Equally spaced selections are made by putting a pointer at a random number between 0, and 1/N where N is the number of selections to be made. The next pointers are put at intervals of 1/N. If a pointer falls within the range of a chromosome that chromosome is selected.

$$RP(c) = \frac{Rank(c)}{\sum_{j-1}^{M} Rank(c)} \tag{2.52}$$

The next step in the genetic algorithm is reproduction. In this step genes are crossed over, and the children are mutated. The crossover function utilized in this thesis is the position independent crossover. This is a uniform crossover, or each gene has the same probability of being selected by the child [35]. The children then have a low probability of being mutated. If a mutation occurs it is by adding a Gaussian random number to the gene.

The final step is the termination. When the termination criterion is met the algorithm will discontinue its search, and return the best chromosome, or chromosomes. The termination criteria can be a minimum fitness found, a maximum number of generations, or a low improvement over several generations. In this thesis the termination criteria is when the population ceases to improve by an amount of 0.0001 errors per frame.

3. SYSTEM DESIGN

3.1 SYSTEM DESIGN

This system is composed of three major components, the first is the feature extractor and classifier. The second is the positive sample tracking through SIFT or SURF. The third component is the positive sample manipulation that forms the road mask, smoothes the boundary using morphology, and filters the tracking of points.

The algorithm's training implementation can be written as follows:

1. Let S be a sequence of training images from lane A

2. Let step be the size of the patch

3. Let on and off be the collected features

4. **For** each frame, f in S **Do**

    4.1 **For** each sample point, p in f, **Do**

        4.1.1 LBP = Calculate LBP features (Section 2.1.1)

        4.1.2 HOG = Calculate HOG features (Section 2.1.2)

        4.1.3 HIST = Calculate patch histogram

        4.1.4 HFEAT = Calculate patch histogram properties (Section 2.l.3)

        4.1.5 FEATURES = [LBP HOG HIST HFEAT]

4.1.6 **IF** patch is on-road

a. append FEATURES to on

Else (Step 4.1.6)

b. append FEATURES to off

End If (Step 4.1.6)

End For (Step 4.1)

End For (Step 4)

5. Return on, off

6. Normalize Data (Equation (3.1))

7. Train Classifier C (Section 2.3.1, 2.3.2)

The algorithm for the testing of the detection of the road can be written as follows:

1. Let S be a sequence of test images from lane B

2. Let P be an empty vector of positive classifications

3. Let xi and yi be vectors of the positions of the patch centers to be classified

3. **For** each frame, f in S **Do**

3.1 Calculate affine transformation matrix, tr (Section 2.2.1, 2.2.2)

3.2 Transform points, P, from f-1 to f with P*tr

3.3 **For** each patch p(xi, yi)

3.3.1 Calculate the patch features (4.1.1-4.1.5 in training algorithm)

3.3.2 Normalize feature using training mean and standard deviation (3.1)

3.3.2 Classify patch p

3.3.3 **If** patch is classified road

a. Append xi,yi to positive classifications P

End If (Step 3.1.3)

3.4 Let BW, BW2, and BW3 be matricies of zeros the size of frame f

3.5 Plot each center point in image BW

3.6 Dilate each center point using structuring element se

3.7 Calculate size of blobs created by dilation

3.8 Calculate ratio of largest blob, b, over second largest blob, b2

3.9 **If** b/b2 < ratio r

a. Add blob b1 and b2 to image BW2

Else If b/b2>r **(Step 3.9)**

b. add blob b1 to image BW2

End If (Step 3.9)

3.10 Calculate convex hull of BW2, and add to BW3

3.11 Meaure top left, tl, and top right, tr, points

3.12 Calculate prediction step of Kalman Filter (Section 2.5.1)

3.13 Calculate update step of Kalman Filter on tl and tr

End For (Step 3)

3.2 IMPLEMENTATION IN DETAIL

In this section the implementation and training of the road detector will be described in detail. There were two lanes utilized in this process. They will be called lane A for the training lane, and lane B for the test lane. In Figure 3.1 and Figure 3.2 sample frames from lanes A and B can be seen respectively.

Figure 3.1: Sample frame from the training lane A

Figure 3.2: Sample frame from the test lane B.

3.2.1 TRAINING DATA

To obtain training data for the classifiers 31 by 31 pixel image patches are extracted from lane A. Training patches were also extracted from the first 200 frames of lane B. Each patch is extracted at a fixed distance from the vehicle, or a fixed height in the image of 250 pixels. The value of 250 pixels loosely corresponds to 30 meters from the vehicle.

This method was chosen because there is no camera orientation information available. Each image patch contains 3 color channels on which the HOG, LBP, and color feature histogram are calculated. The LBP contains 10 bins, the HOG contains 9 bins, and the color histograms contained 10 feature bins with 6 histogram properties in each color channel. The features from each of the three methods were performed on each color channel, and concatenated into one feature vector. This results in 135 features for each image patch.

The amount of training data can easily be adjusted by either skipping frames, or manipulating the interval at which the data points are selected. In this implementation the sample data is taken at an interval of 30 pixels on every fourth frame.

The features for sample i, x_{fi} are normalized to the Z-Score by subtracting the mean of the feature in the training data, μ_f, and dividing by the standard deviation of each set of features, σ_f, in the training set. This helps prevent one feature from dominating the training of the classifiers. The SVM is more susceptible to the domination of one feature than the Random Forest, but for the sake of continuity they will both have their inputs normalized [37].

$$r_{fi} = \frac{x_{fi} - \mu_f}{\sigma_f} \tag{3.1}$$

In Figure 3.1 a sample frame training lane can be seen. In Figure 3.3 the positions of the training data samples can be seen in a sample of one frame. This sampling density can be

increased or decreased. For this implementation the interval was picked to be 31 pixles so they do not overlap.

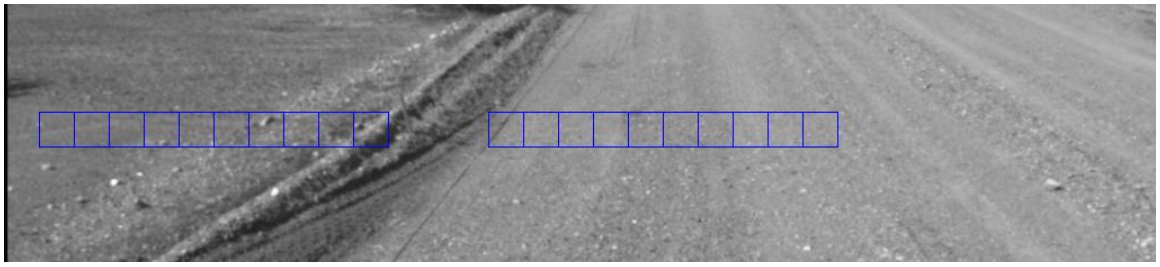

Figure 3.3: Training data sample patches. The samples are only taken from the left side of the road because the area to the right of lane A is also considered a positive road sample.

3.2.2 ROAD PATCH CLASSIFICATION

A random forest, or an SVM classifier is trained on the training data which consists of 40,000 road patch feature vectors. Half of the training samples are positive road, and half are negative. After the classifier is trained the image patches at the same distance of 250 pixels from the top of the image, or around 30 meters from the vehicle are converted into feature vectors, normalized by subtracting the mean, and dividing by the standard deviation of the training data, and classified at a fixed interval of 10 pixels per classification point center, and can be seen in Figure 3.4. These patches do overlap. Positive road patches have their center pixel location saved. Negative classifications are ignored. This process is repeated for each frame. The performance results of each classifier will be discussed in chapter 4.

Figure 3.4: Patch centers at an interval of 10 pixels at the fixed distance of 30 meters from the vehicle.

3.2.3 PATCH CENTER TRANSFORM

When the next frame is classified, the previous frame's positive classifications are transformed into the image coordinates of the new image using the affine transformation obtained from SIFT, or SURF. In equation (3.2) it can be seen that the points (x,y) are transformed into the next frame using a 2D transformation with m values obtained from SIFT which is described in section 2.2.1.

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix} \qquad (3.2)$$

If the points are transformed outside of the range of the image, or the lower bound of the classification range they are discarded. This is done because the individual land mine detection algorithms work in a fixed range from the vehicle. When the targets are too far from the vehicle they can not be seen, and when they are too close it is out of the detectors optimal range. This value depends on the land mine detection algorithms.

3.2.4 BINARY IMAGE MANIPULATION

The road mask is formed by using the morphological dilation over the center points of the patches. This aggregates the points into groups. Since the points are classified at intervals of 10 pixels horizontally the dilation uses a disk structuring element with radius 6 pixels. This makes the windows overlap to an extent. When they move farther down the image a road blob begins to form. Making the detection regions larger prevents them from being removed in the later steps.

Next, the connected groups of pixels are labeled, and the number of pixels in each group is recorded. The largest blobs are saved if the ratio of the size of the largest to second largest is lower than a ratio of 5. This prevents vertically divided blobs from being removed. This happens frequently in classification because the center of the road is the most poorly graded section. In Figure 3.5 one such case can be seen. Without the blobs with a small enough ratio being retained the right blob would have been removed. This would have resulted in a large misclassification of the road.

A convex hull of the remaining points is computed as a preliminary road mask. This allows for easier tracking in the Kalman filter. An alternative to the convex hull is to fill the holes in the image. This is not as well suited a solution as the convex hull because it does not join the blobs if they are divided completely such as in Figure 3.6.

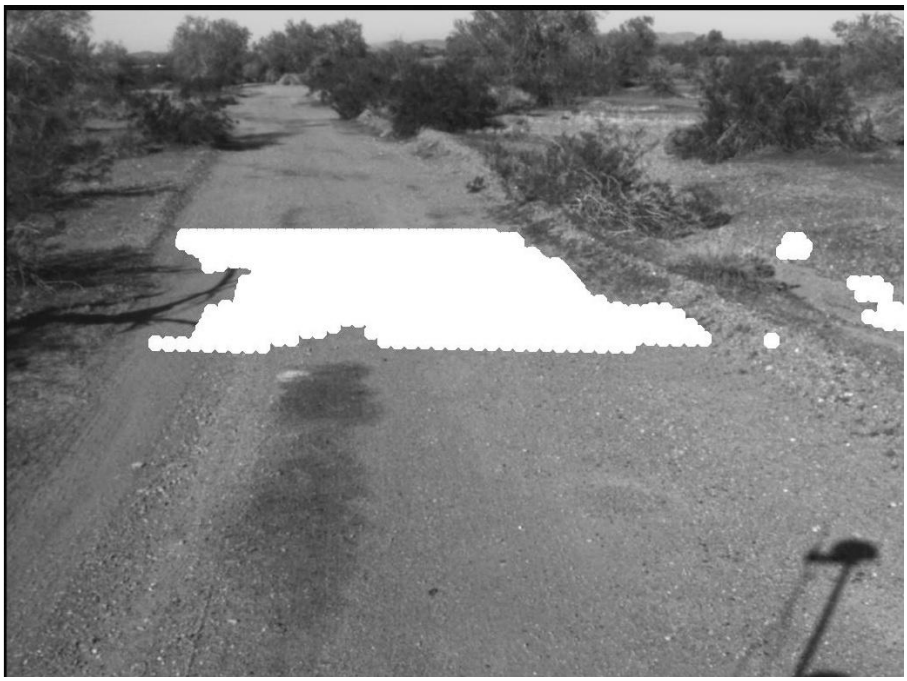

Figure 3.5: The dilated points can be seen forming a mask. There are false positives to the right of the blob that will be removed, and a false negative to the left that will be filled in when the convex hull is calculated.

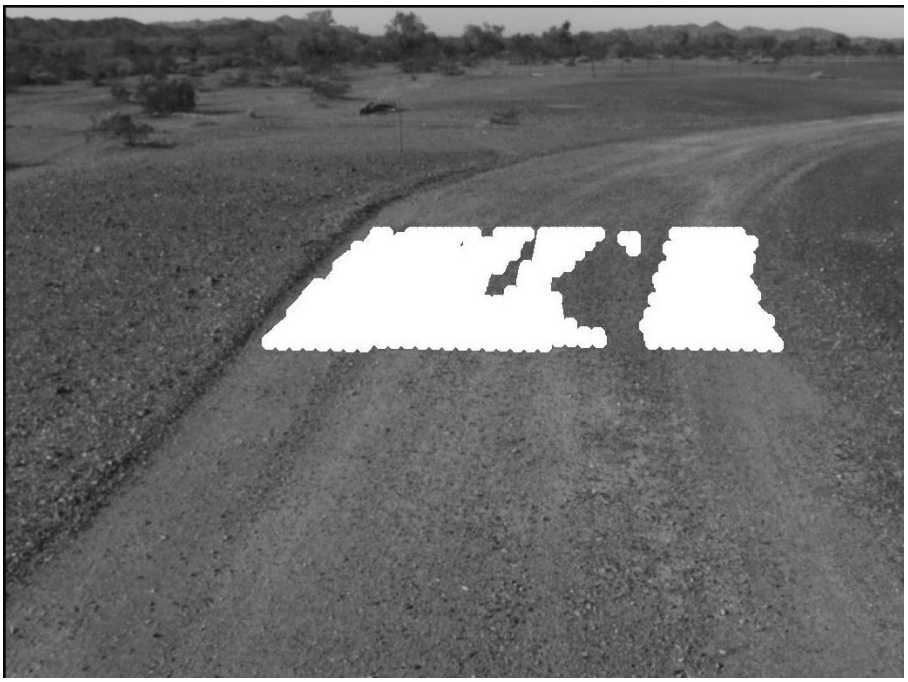

Figure 3.6: The dilated patch centers with a vertical division.

Figure 3.7: The result of the convex hull operation calculated on the blobs in Figure 3.6

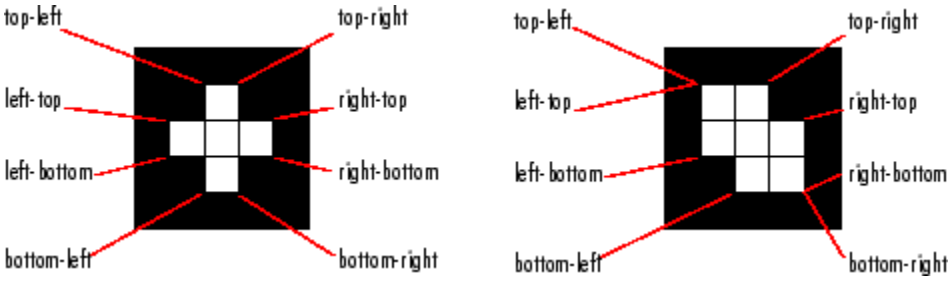

Figure 3.8: The 8 extrema points in a binary image. Image from MATLAB documentation [27].

The top left, and top right corners of the convex hull of the blob are measured for the purpose of generating a ground truth. These locations can be seen in Figure 3.8 on two different 5x5 binary images.

3.2.5 CORNER MEASURING

The top left and top right corners of the convex hull are measured for use in the Kalman filter. The assumption that is made for the use of the Kalman filter is that the corners of the blob will not move in a jerky fashion. For example the corner should only move at around 10 pixels per frame at most. If the corner moves more than that there is assumed to be an error. Implementation of a Kalman filter in the post processing helps to reduce the effect of misclassifications that would drag the corners an unreasonable distance in one frame.

3.3 EXPERIMENTS AND TESTS

This system requires the tuning of many parameters. This includes the kernel parameters in the SVM or the number of trees in the Random Forest. The size of the dilation and the ratio of the road blobs in the first post processing step need to be determined. Finally, the parameters of the Kalman filter need to be chosen. To decide on appropriate values for these parameters a number of tests were performed.

All of the quantitative data is obtained by measuring the accuracy of each step to a hand segmented ground truth. The hand segmented ground truth is not a truth in that the boundary of the road is crisp, but that it is a fair representation of what a human would deem "road" and "not road".

Two hundred frames through a turn were hand segmented to form a 200 by 100 matrix of ground truth values. The top left of the matrix is the left most classification point of the

road in the first test frame, and the bottom left the 200th frame. shows the ground truth matrix in an image format.

Figure 3.9: The hand-segmented ground truth. The top left corner of the frame represents the left classification point in the first frame.

To obtain the quantitative accuracy at each step of the segmentation process a ground classification map is generated. After the ground segmentation has been made by the algorithm it can easily be differenced by subtracting the ground truth, and taking the absolute value. This gives a map of the misclassifications, and false positives. Each is an error value of 1. The errors are summed over the mask and divided by 200 to give an average number of misclassifications per frame..

3.3.1 CLASSIFIER COMPARISON

The decision upon which classifier to use is the first objective in reducing the errors in the road detector. Finding the kernel with the least number of classifications is of paramount importance because one small error can mean the difference in a large false positive blob being included in the main road blob or not.

The SVM has 4 kernels that can be used. The linear kernel is the simplest, and is used for a base line accuracy. The accuracy can further be improved by using a non linear kernel. The sigmoid, the polynomial, and the Gaussian radial basis function each have a gamma value that can be chosen. The gamma value is a scaling coefficient that is performed within the kernel. The recommended value of gamma is $1/\|features\|$. In this thesis the size of features is 135. The coefficient is set to zero, and the cost is set to 1. The tested values of gamma will be from 0.002 to 0.017 which will go from much below the recommended gamma to more than double the recommended value of 0.0074. The purpose of this set of experiments is to find a baseline accuracy for the classifiers.

After the values of gamma are varied on each kernel independently a genetic algorithm described in section 2.6.1 is performed on each kernel. The chromosome consists of gamma, the coefficient, and the cost. The polynomial kernel contains an extra parameter d, the degree. The genetic algorithm could be run on all of the kernels with them as a gene in the chromosome, but with the search space narrowed the results were found in a more timely manner.

Lowering the value of gamma will make a smoother decision surface. Choosing the right value will also help reduce over fitting, and increase classification accuracy.

The random forest classifier will be tested by starting with one tree, and growing trees up to 250.

3.3.2 POSTPROCESSING ACCURACY IMPROVEMENTS

In the binary image manipulation stage of the post processing there are several variables that need to be determined. The first is the size and shape of the structuring element, and the second is the blob retention ratio.

3.3.2.1 DILATION STRUCTURING ELEMENT SIZE AND SHAPE, AND BLOB RETENTION RATIO

The blob retention ratio will be adjusted along with the structuring element because they directly influence each other. If the structuring element is too small the dilated points will not join to form a blob. Individual blobs that are thrown out are orders of magnitude smaller than the largest blob. That means that the primary guess will be between 1 and 5 for the ratio. The ratio of 1 means that the two largest blobs are the same size, and 5 means that the largest blob is 5 times bigger than the second largest blob.

In mathematical morphology there are a seemingly infinite number of structuring elements that can be used. There will be three investigated in this thesis. They will include a square, a diamond, and a circle element. Because the points are sampled at an interval of 10 pixels, the structuring element's radiuses will be initialized from 5 to 8 pixels. The genetic algorithm described in section 2.6.1 will be used to simultaneously select the retention ratio, and the structuring element size. The algorithm will be run on each of the 3 element shapes.

Figure 3.10: From left to right, the circle, square, and diamond structuring elements.

The road will be mapped again and compared to the ground truth by tracking the top left and top right corners of the blob. The pixels in between the top left and top right point will be labeled positive road.

3.3.3 KALMAN FILTER PARAMETERS

The corners of the blob will be filtered by the Kalman filter. This will give new edge points that move in a smoother way. These points will be used to generate another road map that can be compared to the hand segmented ground truth.

The process of Kalman filtering gives rise to another way of measuring error. The mean squared error (MSE) of the Kalman filtered output to the ground truth road edge can be minimized. In (3.3) MSE is the error, t_k is the true position, and o_k is the observed position at frame t over N frames.

$$MSE = \frac{\sum_{k=1}^{N} (t_k - o_k)^2}{N} \tag{3.3}$$

4. RESULTS

4.1 CLASSIFIER ACCURACY

Decreasing the number of false alarms and misclassifications improves the system all together. This means that picking the classifier that best performs will help the blob retaining method as well as the Kalman filter at the final step. In the next two sections, the classifiers that were tested will be analyzed and compared for accuracy. The speed of each classifier is also noted. The speed will not be taken into account in the selection of the winning classifier because the objective of this thesis was not to find a fast road detector. Speed improvements can also be made at later point in time.

4.1.1 SVM

The first set of experiments is to determine the performance of individual kernels in the SVM classifier. The kernel of choice is the one with the lowest errors per frame when compared to the hand segmented ground truth.

Table 1: List of kernels optimized by varying gamma, their best gamma, the resulting average error per frame, the training time, and the test time per frame.

Kernel	Error	Gamma	Train Time	Test Time	Support Vectors
Linear	8.01	N/A	11.19	0.004	199
Radial Basis	7.18	0.0014	12.29	0.0191	792
Sigmoid	7.04	0.0023	5.95	0.0127	624
Polynomial d=2	7.54	0.0023	70.31	0.166	4532
Polynomial d=3	6.96	0.0025	45.26	0.113	6441
Polynomial d=4	**6.85**	**0.0045**	**67.80**	**0.179**	**7308**
Polynomial d=5	7.08	0.0068	33.57	0.095	4852

Table 2: Genetic algorithm minimum values.

Kernel	Error	Gamma	Coefficient	Cost	Support Vectors
Linear	7.99	N/A	N/A	0.018	414
Radial Basis	7.04	0.0014	N/A	0.4275	1089
Sigmoid	6.58	0.0048	-0.8897	1.903	590
Polynomial 4	6.845	0.0045	0	1.045	8181

Table 1 shows the performance of the individual kernels with their best gamma value with the coefficient set to 0, and the cost set to 1. Table 2 shows the result of the genetic algorithm optimization. The genetic optimization allows more values to be changed easily, and a lower error to be achieved. The error is not decreased by a significant amount, and Table 2 shows that changing the cost function and coefficient do not offer a way to make a large improvement.

4.1.1.1 LINEAR KERNEL

The first kernel tested is the simplest kernel: The linear kernel by the dot product of $x^{(i)}$ and x. The linear kernel is found to have the lowest accuracy of the 4 kernels tested. The error rate was 8.01 errors per frame, but it is the fastest to train and test.

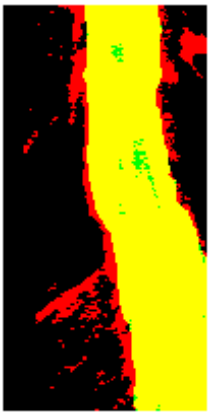

Figure 4.1: The linear kernel classification. Red shows false positives, green shows false negatives, and yellow shows correct classifications.

The higher error rate resulted in false alarms being included in the post processing section of the algorithm. The SVM with the dot product kernel was more optimistic than the other kernels tested, and a low number of misclassifications resulted. At the left side of the classification in Figure 4.1 the large blob of false alarms that will be included in the road blob can be seen.

4.1.1.2 RADIAL BASIS KERNEL

The equation for the radial basis kernel can be seen in (2.38). In Figure 4.3 demonstrates that the minimum error was not found for the originally specified values for gamma. After lowering the range of the gamma to .001 the minimum was located at 0.0014 with an error rate of 7.18 errors per frame. Figure 4.2 shows the classification for the radial basis kernel with the best gamma.

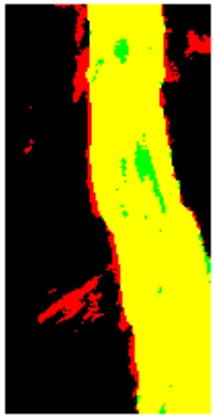

Figure 4.2: For the radial basis kernel in red the false alarms can be seen. Yellow is correct classifications. Green is the false negatives.

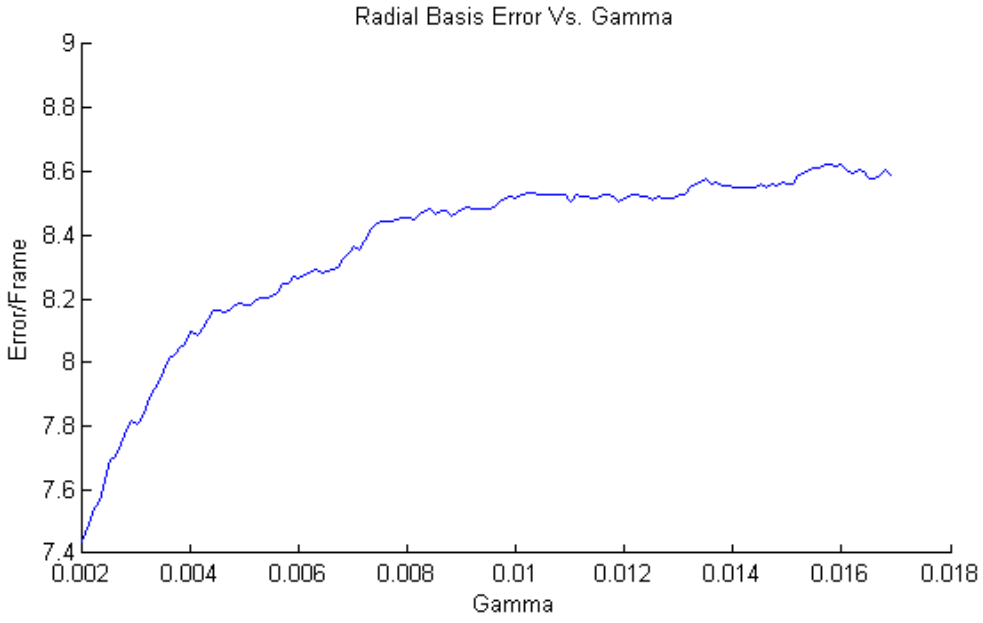

Figure 4.3: The error vs. gamma on the radial basis kernel over the initial gamma range of 0.002 to 0.017.

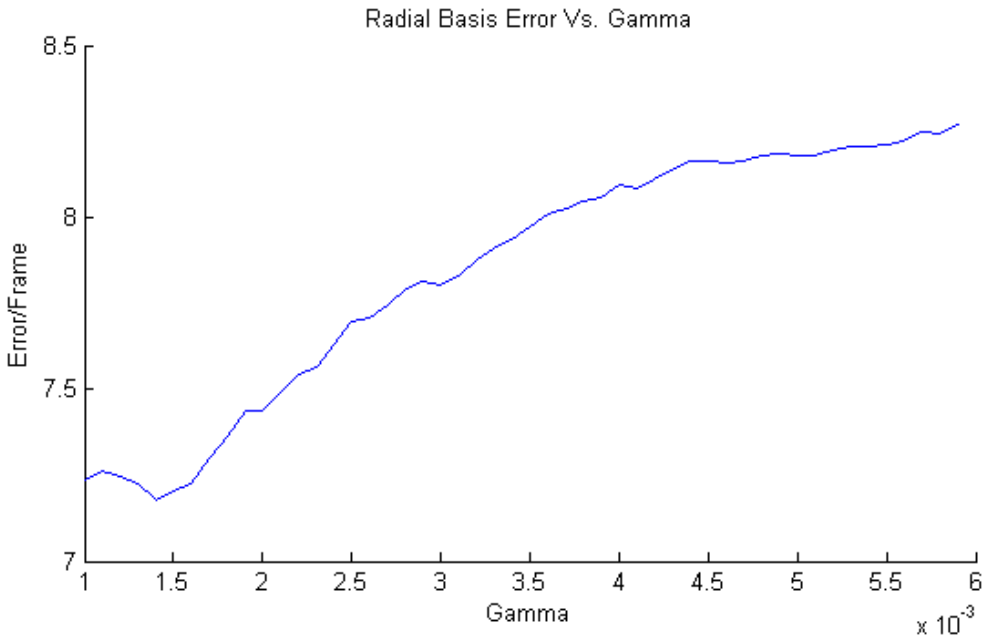

Figure 4.4: The error vs. gamma on the radial basis kernel over the lower gamma range of 0.0001 to 0.0006.

It can be seen in Figure 4.3 that the error does not reach a local minimum, and is still decreasing with gamma. Figure 4.4 shows the value of gamma tested from 0.001 to 0.0006, and locating a local minimum at 0.00014.

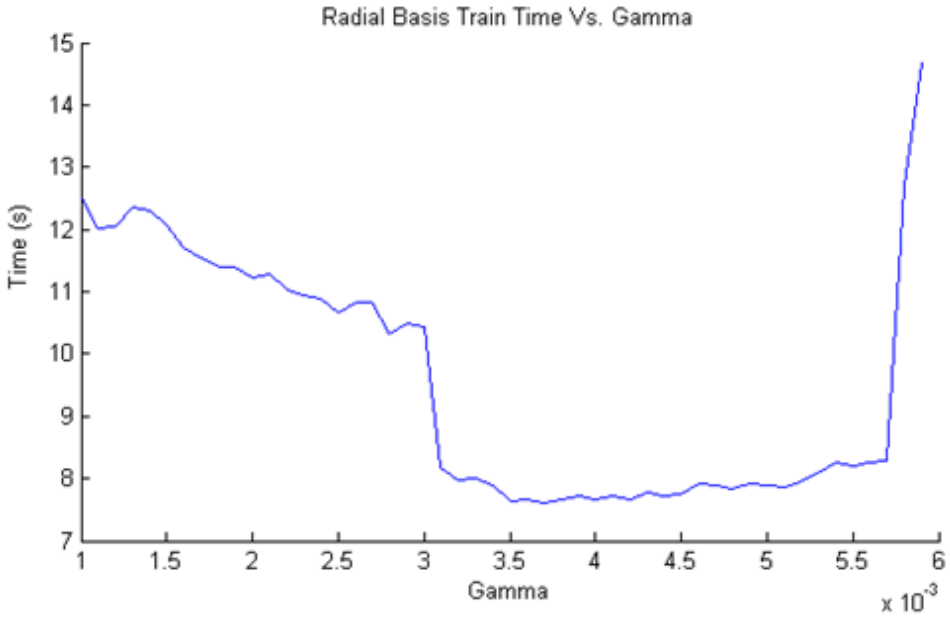

Figure 4.5: The graph shows the training time of the SVM vs. the gamma value. At the lowest error rate the training time was found to be 12.29 seconds.

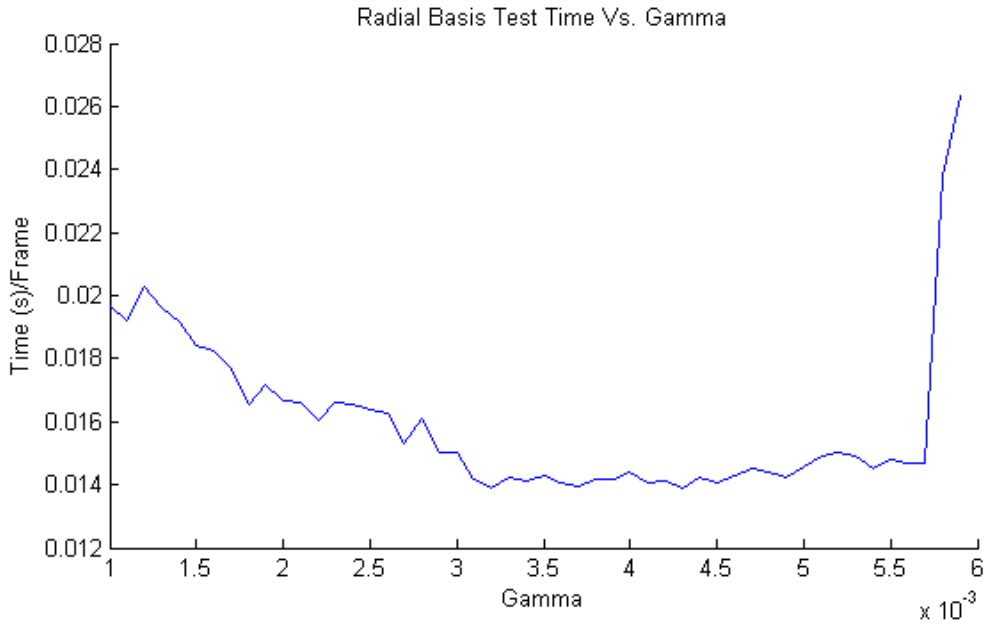

Figure 4.6: The graph shows the test time per frame of the SVM vs. the gamma value. At the lowest error rate the classification time for each frame was found to be 0.0191 seconds.

In Figure 4.2: For the radial basis kernel in red the false alarms can be seen. Yellow is correct classifications. Green is the false negatives. The left side false alarm area is now not connected to the main road blob. This small improvement over the dot product kernel will prevent the post processing from lumping that blob into the positive road classification area.

4.1.1.3 POLYNOMIAL KERNEL

The polynomial kernel has two parameters that will be varied and can be seen in (2.39). The first is the degree, and the second is the gamma parameter. The degree was tested over a range of 2 to 5 with a gamma range of 0.001 to 0.018. Table 3 shows the performance of each individual degree, and Figure 4.7 shows the best kernel classification against the ground truth.

Table 3: List of polynomial kernel degrees, and their respective lowest error achieved by varying gamma.

Degree	**Best Gamma**	**Lowest error**
2	0.0023	7.54
3	0.0025	6.96
4	0.0045	6.85
5	0.0068	7.08

From Table 3 it can be seen that raising the degree does not necessarily improve the performance of the classifier. The error improves from d=2 to d=4, but increases again with d=5. This is because the decision boundary becomes more flexible as the degree is

increased. At a point the flexibility leads to over fitting, and reduces the classification of the test set.

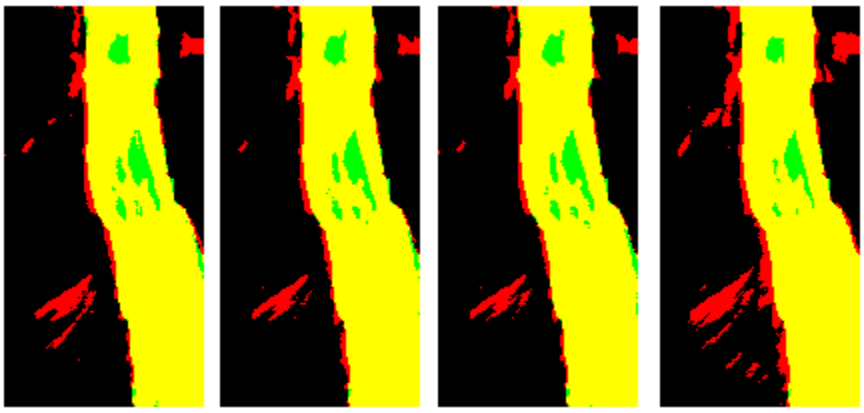

Figure 4.7: From left to right the best gamma for each dimension, d=2, d=3, d=4, d=5. The red shows false positives, the green shows misclassifications, and the yellow shows correct classifications.

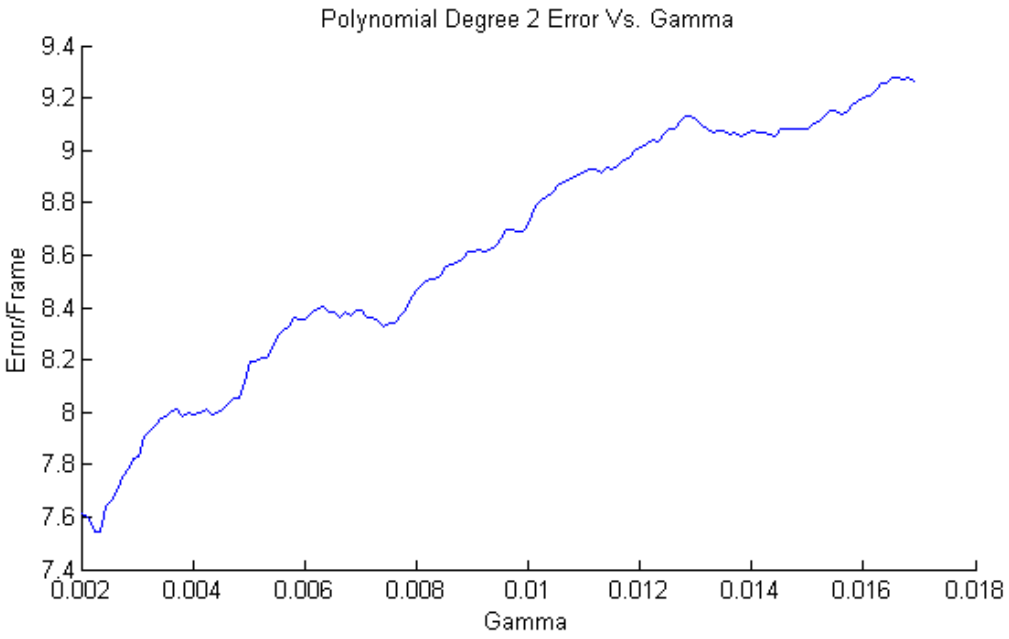

Figure 4.8: The graph shows the error vs. gamma on the polynomial kernel with d=2. The lowest error was found to be 7.54 at a gamma of 0.0023.

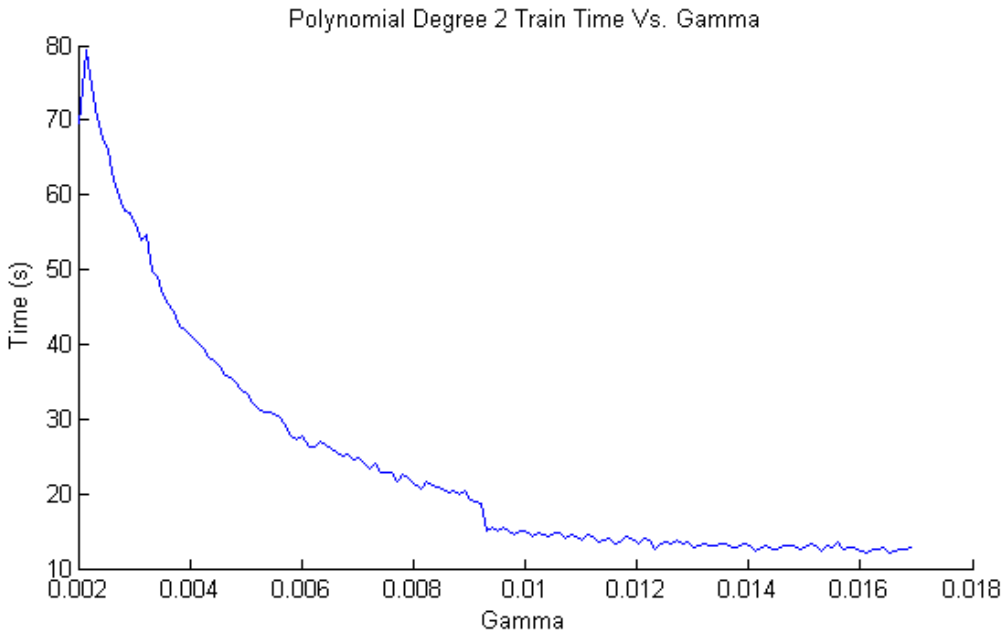

Figure 4.9: The graph shows the training time vs. the gamma value. At the lowest error rate the training time was found to be 70.31 seconds.

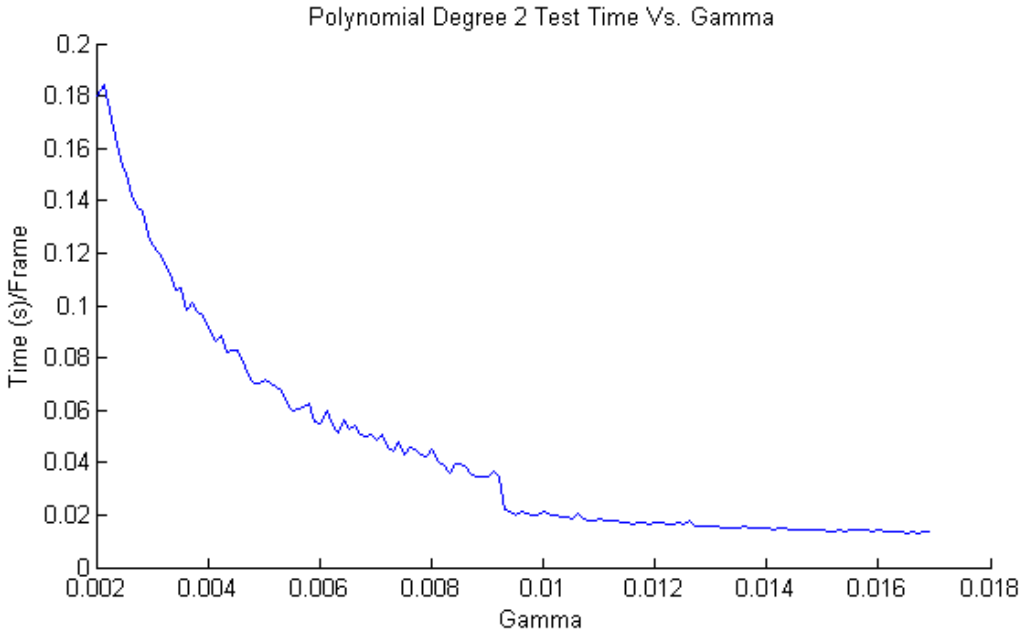

Figure 4.10: The graph shows the test time per frame vs. gamma. The test time was found to be 0.164 seconds/frame at the lowest error rate.

With the degree set to 2 the error rate increases in Figure 4.8 with gamma while the classification time, and training time decrease as seen in Figure 4.12, and Figure 4.13.

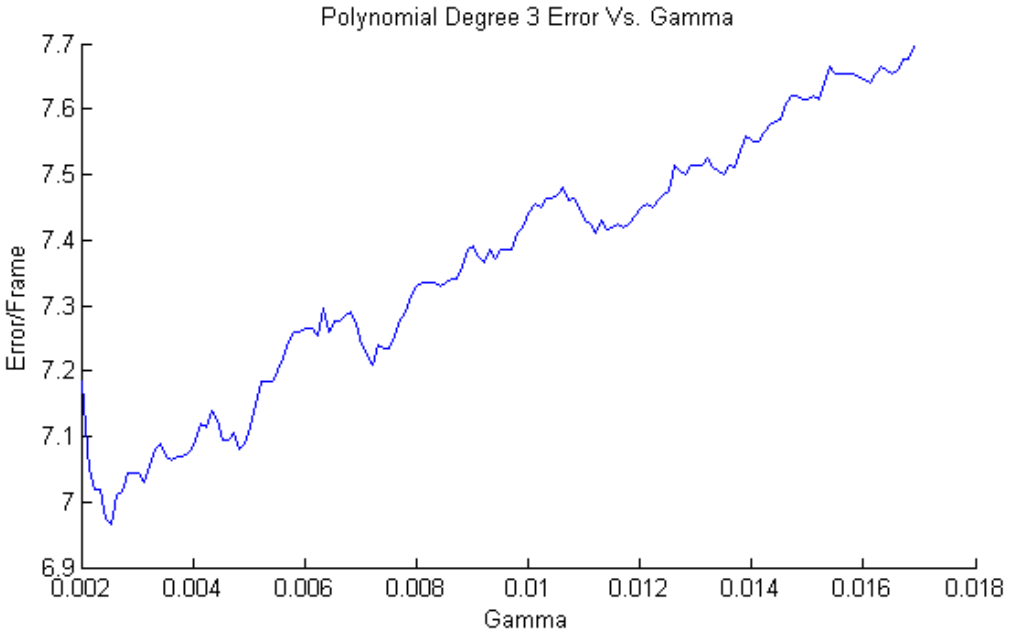

Figure 4.11: The graph shows the error vs. gamma on the polynomial kernel with d=3. The lowest error was found to be 6.96 at a gamma of 0.0025.

With the degree set to 3 it can be seen that the classification error in Figure 4.11 increases with gamma in a similar way to the polynomial kernel of degree 2. This behavior changes with the degree set to 4 and 5.

The training time, and classification time also exhibit similar behavior to the polynomial kernel of degree 2 as seen in Figure 4.12, and Figure 4.13.

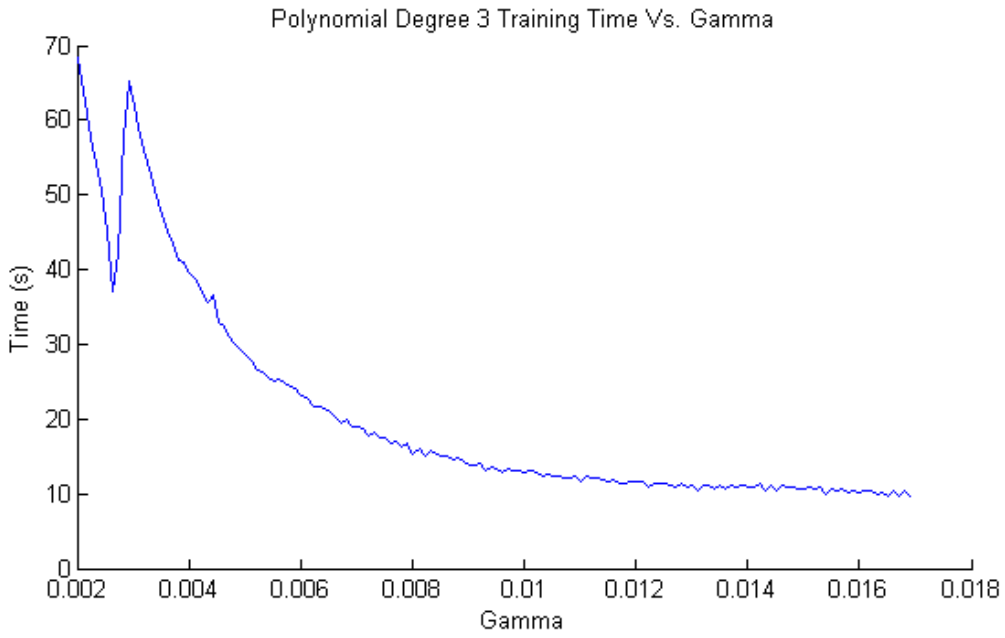

Figure 4.12: The graph shows the training time vs. the gamma value. At the lowest error rate the training time was found to be 45.26 seconds.

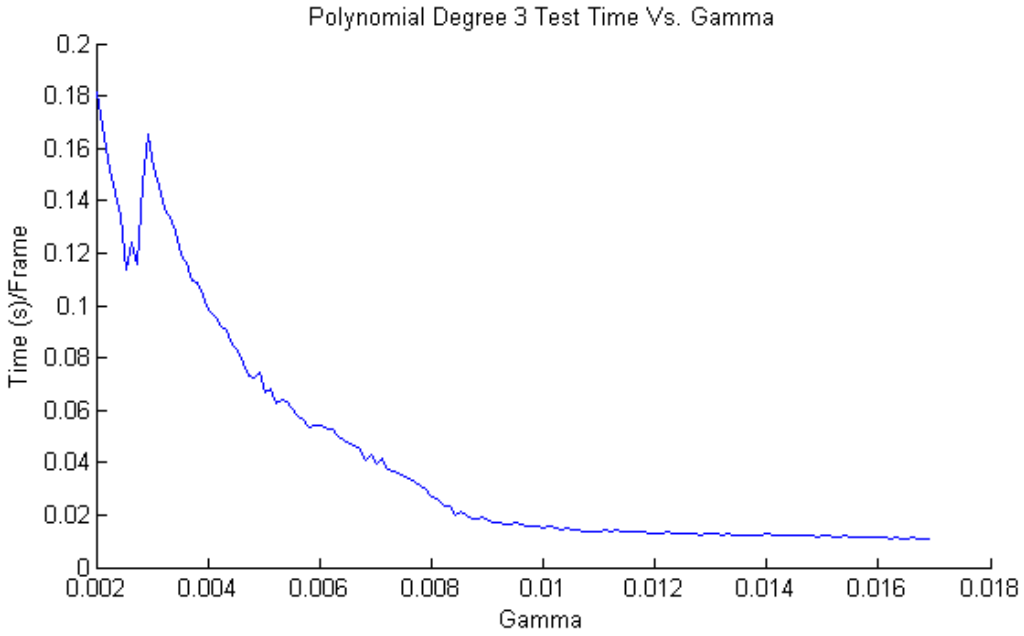

Figure 4.13: The graph shows the test time per frame vs. gamma. The test time was found to be 0.113 seconds/frame at the lowest error rate.

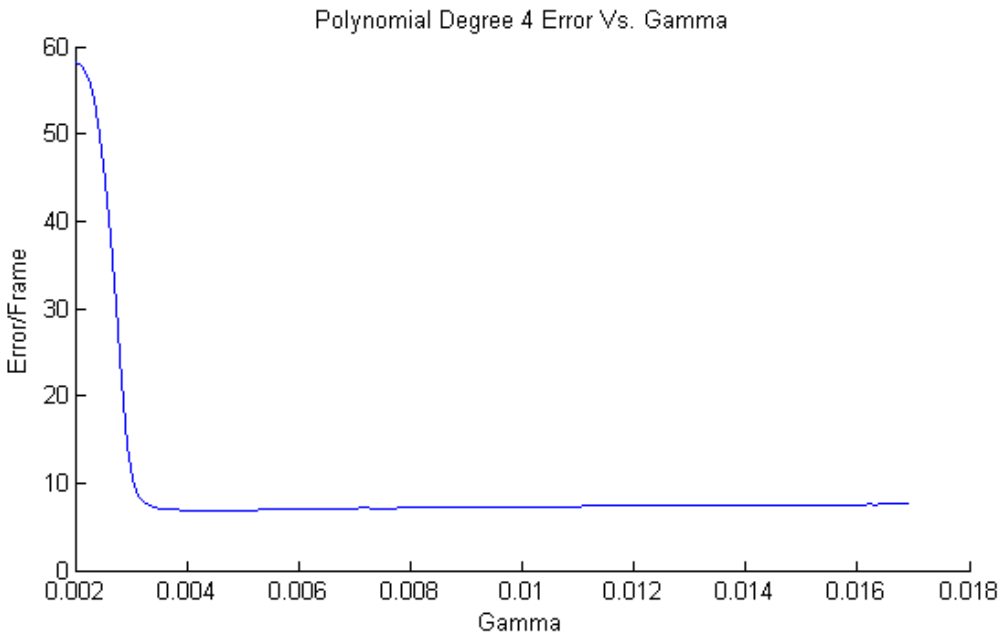

Figure 4.14: The graph shows the error vs. gamma on the polynomial kernel with d=4. The lowest error was found to be 6.85 at a gamma of 0.0045.

In the case with d=4 shown in Figure 4.14 the error stabilizes, and does not increase quickly when gamma is increased. In implementation this would allow for an easier tradeoff between accuracy and speed.

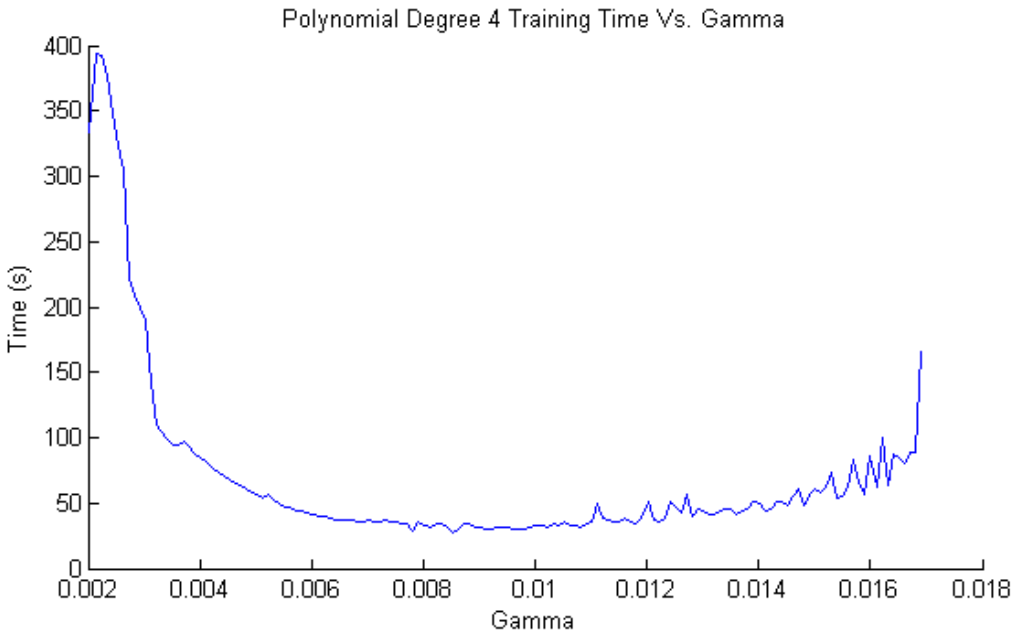

Figure 4.15: The graph shows the training time vs. the gamma value. At the lowest error rate the training time was found to be 67.80 seconds.

In Figure 4.15 it can be seen that the degree 4 polynomial kernel has a much higher training time for low values of gamma. The training time reaches 400 seconds for gamma equal to 0.0022, where the other kernels have training times of 50 to 70 seconds at their slowest. Figure 4.16 shows the test time having a similar response to the variation on gamma as the training time in Figure 4.15, however the classification time does not increase like the training time.

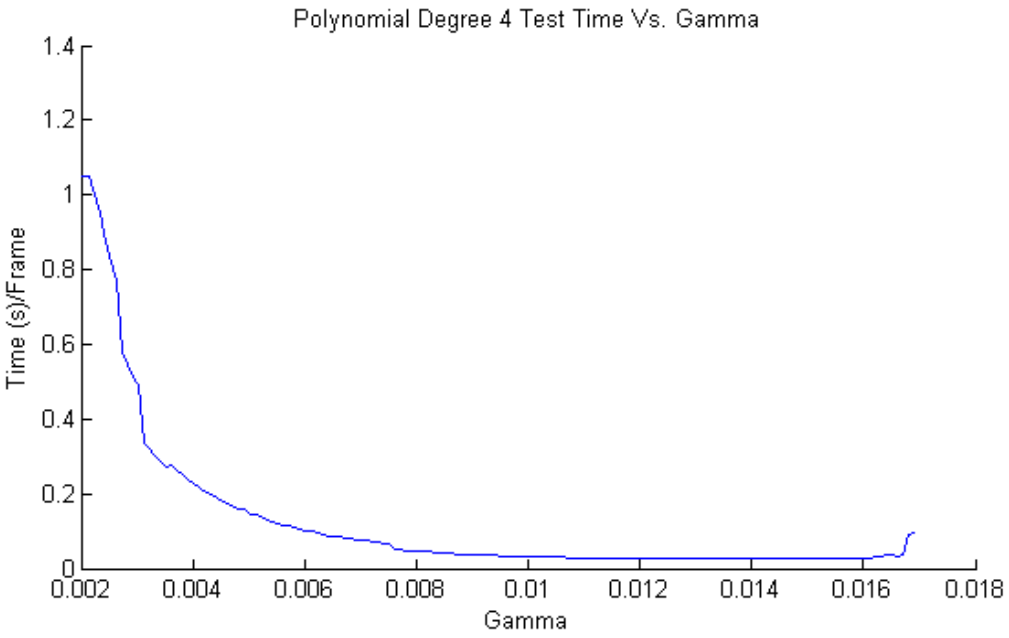

Figure 4.16: The graph shows the test time per frame vs. gamma. The test time was found to be 0.179 seconds/frame at the lowest error rate.

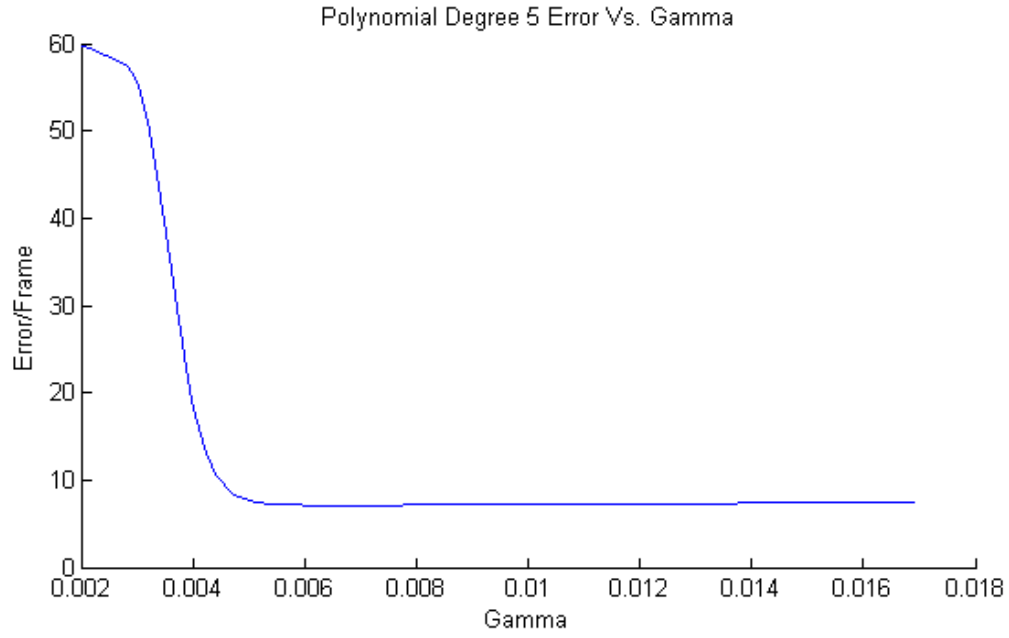

Figure 4.17: The graph shows the error vs. gamma on the polynomial kernel with d=5. The lowest error was found to be 7.08 at a gamma of 0.0068.

In Figure 4.17 it can be seen that the error rate of the degree equal to 5 changes similarly to the degree equal to 4 in Figure 4.15. The best error rate is not as good as with d=4. This is due in part to the increased flexibility of the boundary. If the boundary is too flexible, the classifier will begin to over fit the data.

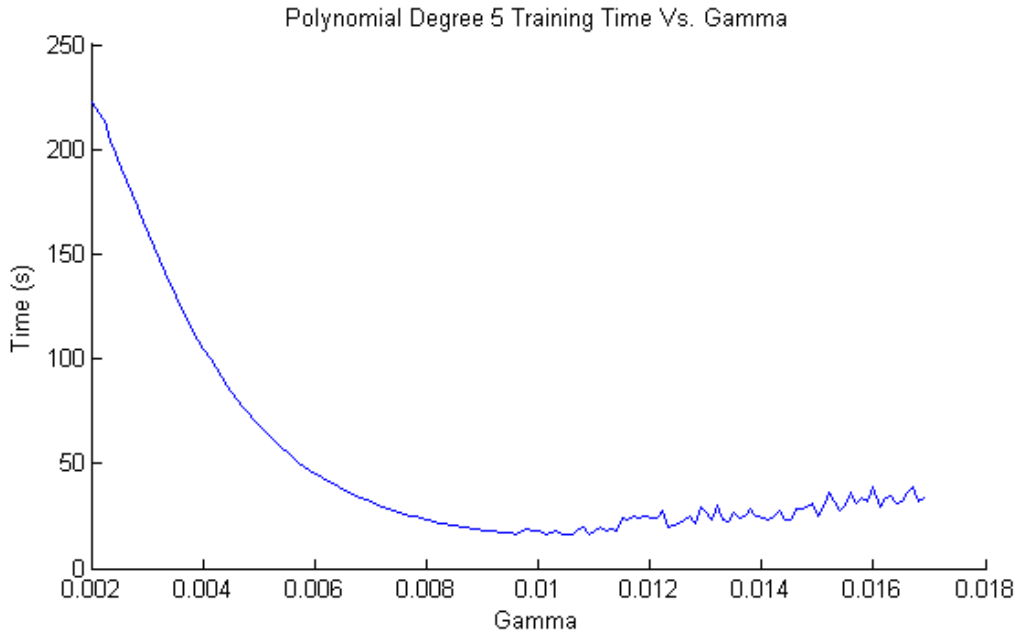

Figure 4.18: The graph shows the training time vs. the gamma value. At the lowest error rate the training time was found to be 33.57 seconds.

In Figure 4.18 the training time can be seen decreasing smoothly until gamma reaches about 0.01. After gamma = 0.01 the time increases, but not as smoothly as it decreased. In figure 4.19 it can be seen that the training time falls off with increased gamma values.

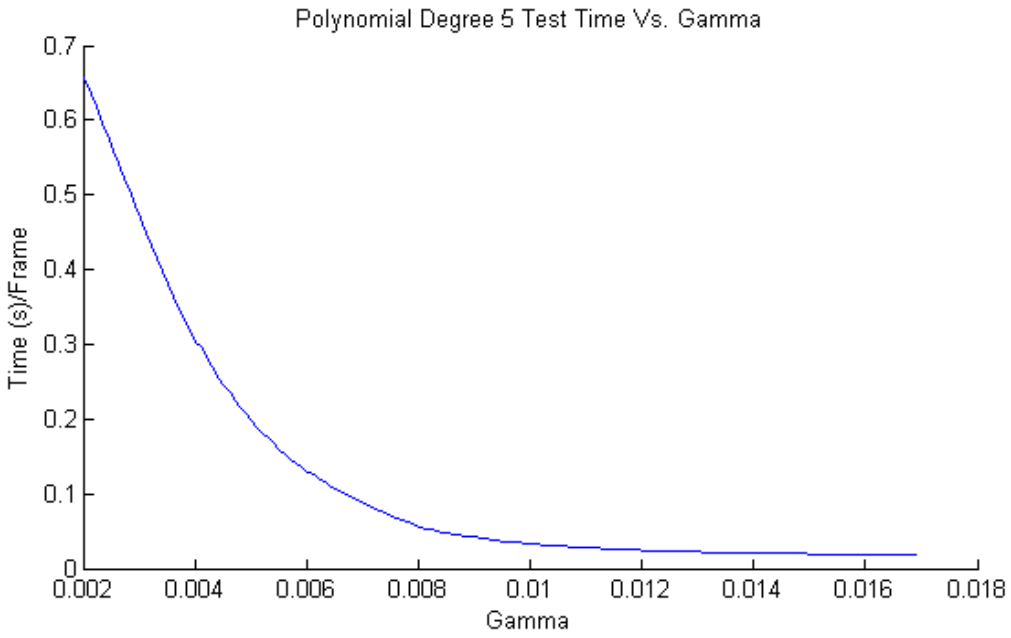

Figure 4.19: The graph shows the test time per frame vs. gamma. The test time was found to be 0.095 seconds/frame at the lowest error rate.

In this section it is clear that the polynomial kernel offers many options. Each one has its advantages and disadvantages that can be taken into consideration. If there is a tradeoff needed between training time and accuracy, or test time and accuracy, then there are options available.

4.1.1.4 SIGMOID KERNEL

The final kernel investigated was the sigmoid kernel. The kernel is described in (2.40).

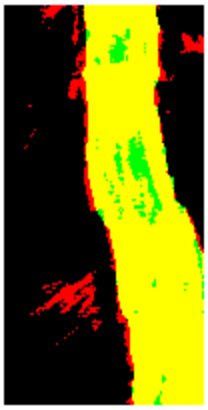

Figure 4.20: For the sigmoid kernel in red the false alarms can be seen. Yellow is correct classifications. Green is the false negatives.

By changing only the gamma parameter, the sigmoid kernel is not the highest performing, but the genetic algorithm was able to tune the parameters in it to the lowest error rate. The sigmoid kernel also has the fastest training and second fastest test time. The next fastest kernel is the radial basis function that takes 12.29 seconds, or 7.34 seconds longer.

Next to the dot product kernel, the sigmoid kernel requires 0.0127 seconds per frame, or could classify around 78 frames per second. The best performing polynomial kernel of degree 4 requires 0.179 seconds per frame, or 5.6 frames per second. 5.6 frames per second is not acceptable for a real time application.

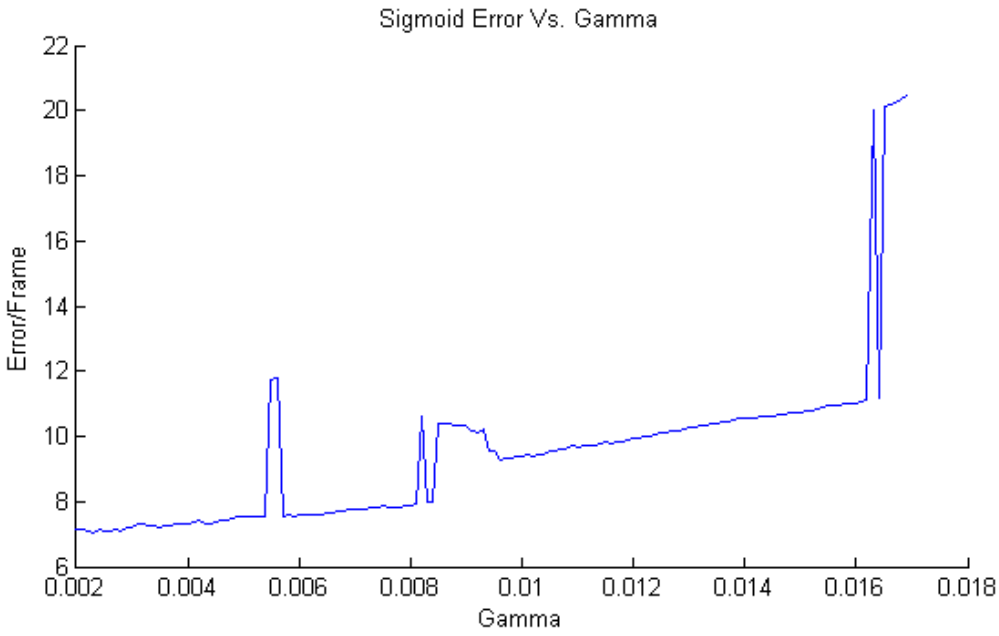

Figure 4.21: The minimum error by varying gamma was found to be 7.04 at gamma=0.0023.

Figure 4.21 shows that the testing error of the sigmoid kernel SVM increases with an increase in the scaling factor, gamma. This behavior is similar to the polynomial kernels of degree 2 and 3. Unlike those kernels, the training, and test times increase with gamma as seen in Figure 4.22, and Figure 4.23.

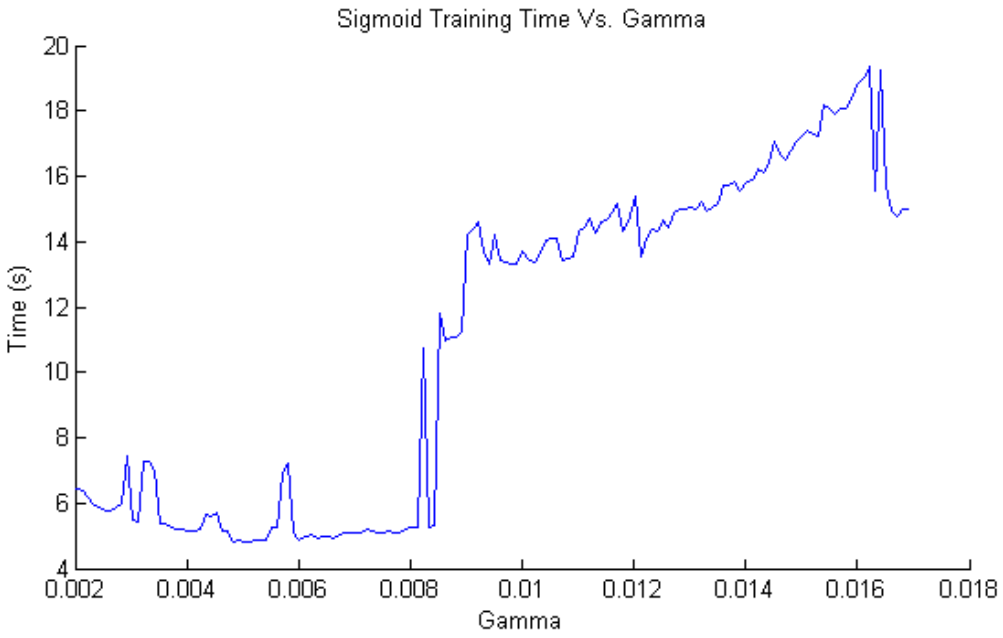

Figure 4.22: The training time was found to be 5.95 seconds at the best gamma.

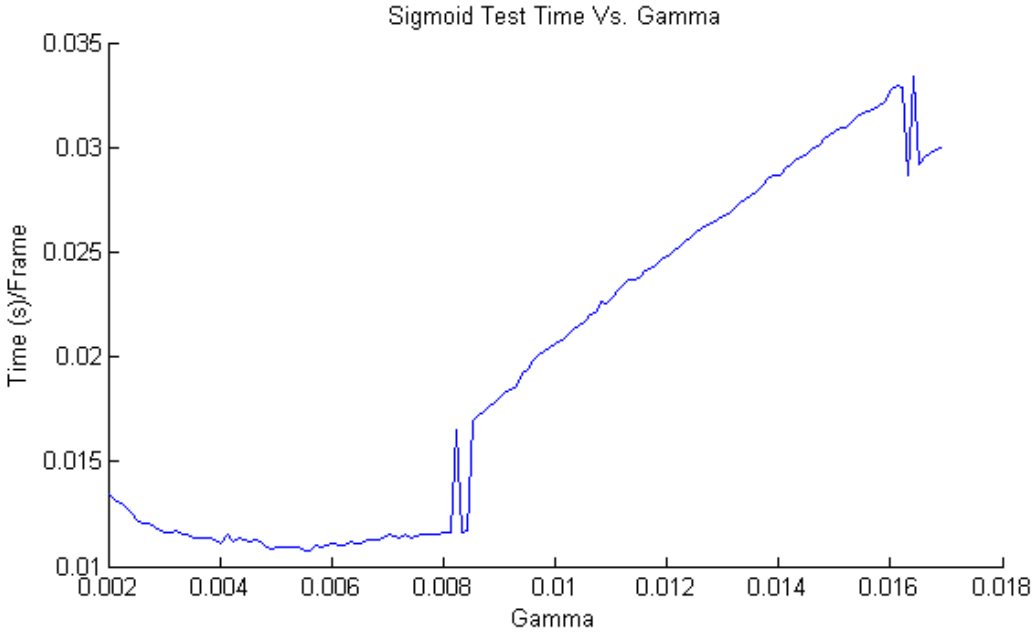

Figure 4.23: The test time was found to be 0.0127 seconds per frame at the best gamma.

4.1.2 RANDOM FOREST

The random forest overall did not perform as well as the SVM, but there is one big advantage which will be discussed at the end of this section. The error rate was fairly consistent across all of the training tests seen in Figure 4.24. The expected decreasing trend of the addition of trees to the random forest is not seen in Figure 4.24.

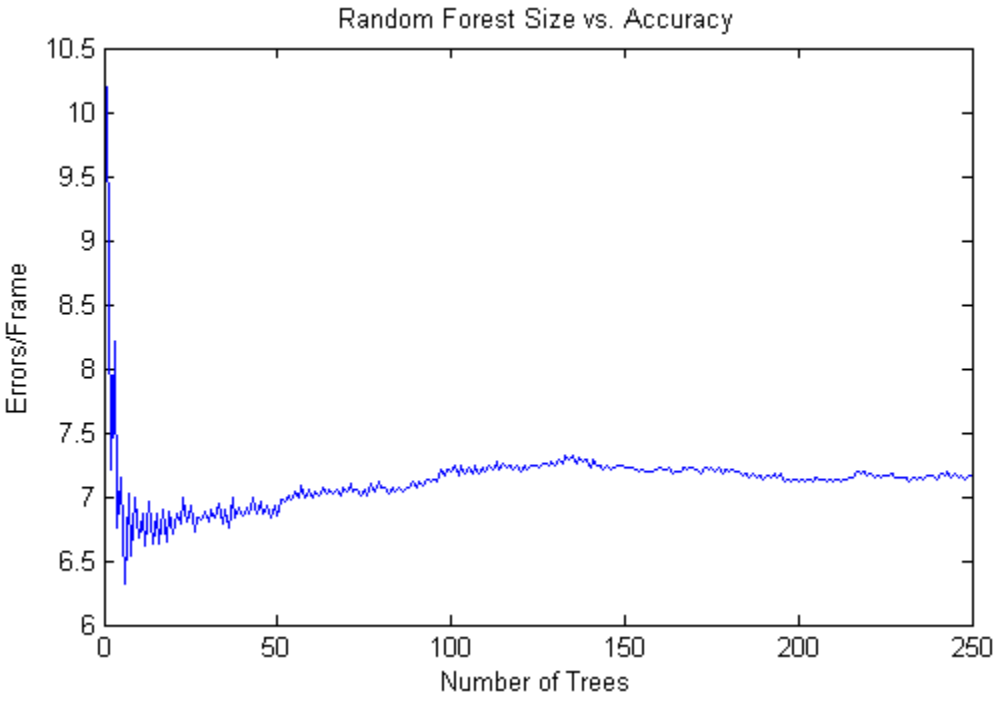

Figure 4.24: The error decreases quickly, and then evens out as trees are added.

The average depth of the trees for T = 250 was 14.4. Each tree

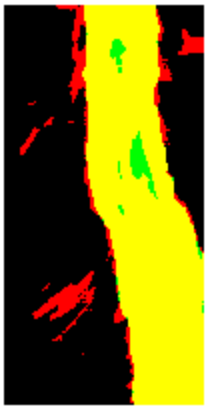

Figure 4.25: The classification for a random forest with T = 250. In yellow, correct classifications can be seen. Red shows false positives, and green shows false negatives.

Figure 4.25 shows that even though the random forest does not achieve as low an error rate as the SVM, it still has good separation between false positive blobs, and the true positive road. This is essential in the post processing step. The random forest generalizes well, and quickly. The main advantage in this application is that the random forest does not require much tuning if any, at all. Simply choosing a sufficiently large number of trees will perform well out of the box. In Figure 4.26 it can be seen that the classification is very similar to the classification with T=250.

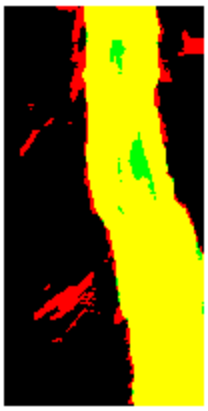

Figure 4.26: The classification for a random forest with T = 50. In yellow, correct classifications can be seen. Red shows false positives, and green shows false negatives.

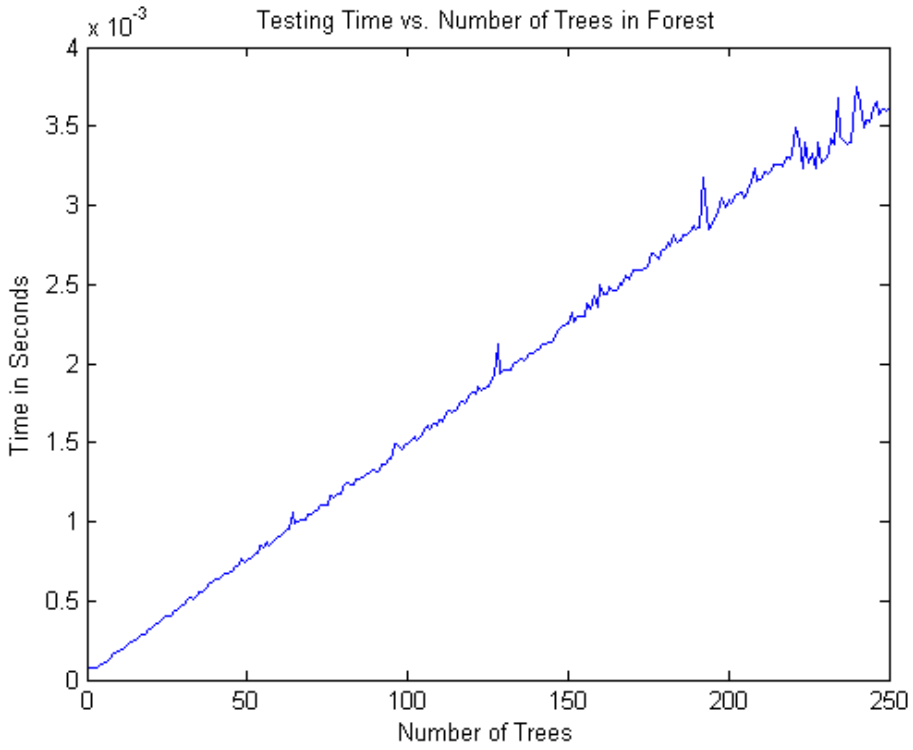

Figure 4.27: The classification time per frame can be seen to increase almost linearly with the addition of trees.

While the classification time increases linearly from 0.0005 seconds to 0.0035 seconds per frame with the addition of trees, the accuracy does not. This allows for a tradeoff to be made between accuracy, and classifier complexity. This test time value could easily be reduced with more threading capability.

On average each tree took 0.3 seconds to train. The time it would take to train T trees would be extremely close to T*0.3 seconds. When tested this value came to 61.1 seconds which results in 0.24 seconds per tree.

In a final implementation of this algorithm the random forest would be the best candidate for a patch classifier. The random forest easily generalizes the decision boundary on the training data. Classification of road and non road patches with the given features is achieved with almost no tuning required. The only requirement is that the number of trees in the forest be set sufficiently high. This is a value of around 50 trees. The random forest is also easily parallelizable because the individual forests do not depend on each other. With 4 processors available the test time came to 0.001 seconds per frame with 50 trees.

4.2 POST PROCESSING ACCURACY IMPROVEMENT

By selecting the proper structuring element, and blob retention ratio the error can be greatly reduced. In each of the next sections, the values were trained by the use of a genetic algorithm.

4.2.1 DILATION STRUCTURING ELEMENT SIZE AND SHAPE, AND BLOB RETENTION RATIO

Table 4 shows that the method of dilating the center points with a structuring element, removing smaller blobs, and retaining large ones is an effective method of reducing false alarms. The use of the diamond structuring element reduced the errors by almost 70%.

Table 4: The result of the genetic algorithm on three structuring elements, and their size along with the blob retention ratio.

Structure	Errors	Size	Ratio
Square	3.37	11	5.4
Circle	3.35	10	4.1
Diamond	**3.33**	**5**	**5.8**

In Figure 4.28, Figure 4.29, and Figure 4.30 the steps taken in post processing with the diamond structuring element described in Table 4 can be seen in detail.

Figure 4.28 shows all of the positive classifications up to the time of this frame. Figure 4.29 shows the largest two blobs retained. Finally, Figure 4.30 shows the convex hull of the two largest blobs.

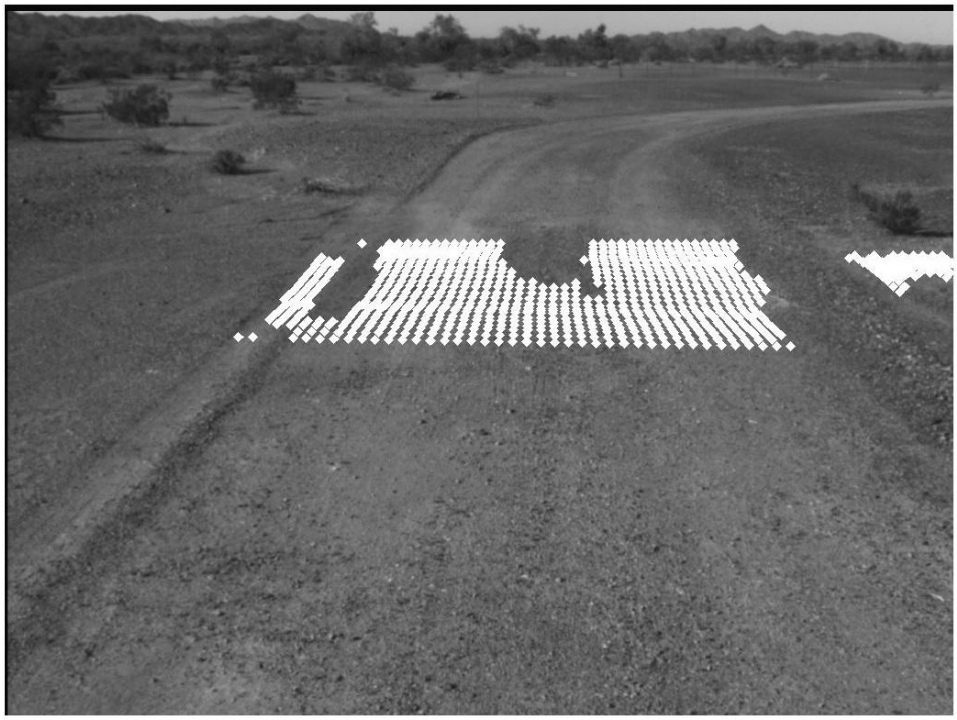

Figure 4.28: All of the positive classification centers dilated.

Figure 4.29: The two largest blobs retained.

Figure 4.30: The convex hull taken over the dilated image.

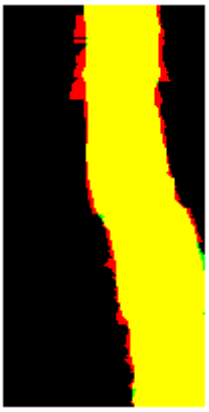

Figure 4.31: The road map of the edges after the morphological dilation, and convex hull. Red shows false positives. Green shows false negative and yellow shows correct classifications.

Figure 4.31 shows that measuring the top left and right corners, and comparing them to the ground truth reduces misclassifications significantly. This is due to holes in the road

being filled, and false positive blobs being removed. It can be noted that the edge of the road is jagged. This will be addressed in the next section.

4.2.2 KALMAN FILTER

The Kalman filter did not improve the accuracy of the classifier. The error rate went from 3.33 errors per frame to 4.14 when the Kalman filter was applied. The Kalman filter will still be included in the final implementation because of the need to remove jerky movements in the road boundary.

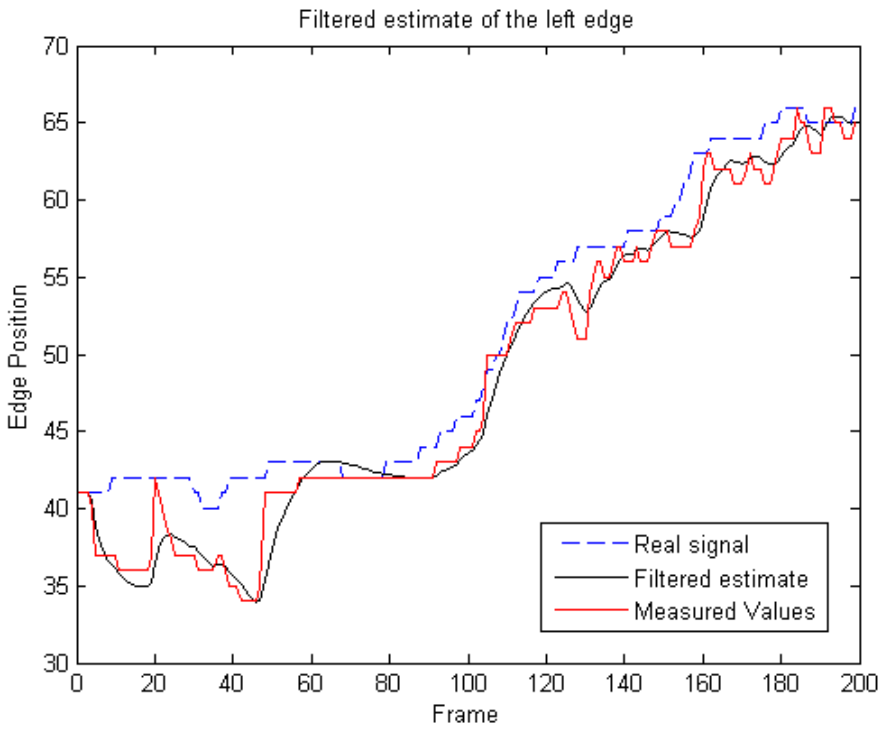

Figure 4.32: The measured and ground truth values of the left road edge can be seen.

The MSE error for the Kalman filtered result in Figure 4.32 came to 3.196, where the error of the measured edge to the actual edge came to 3.076.

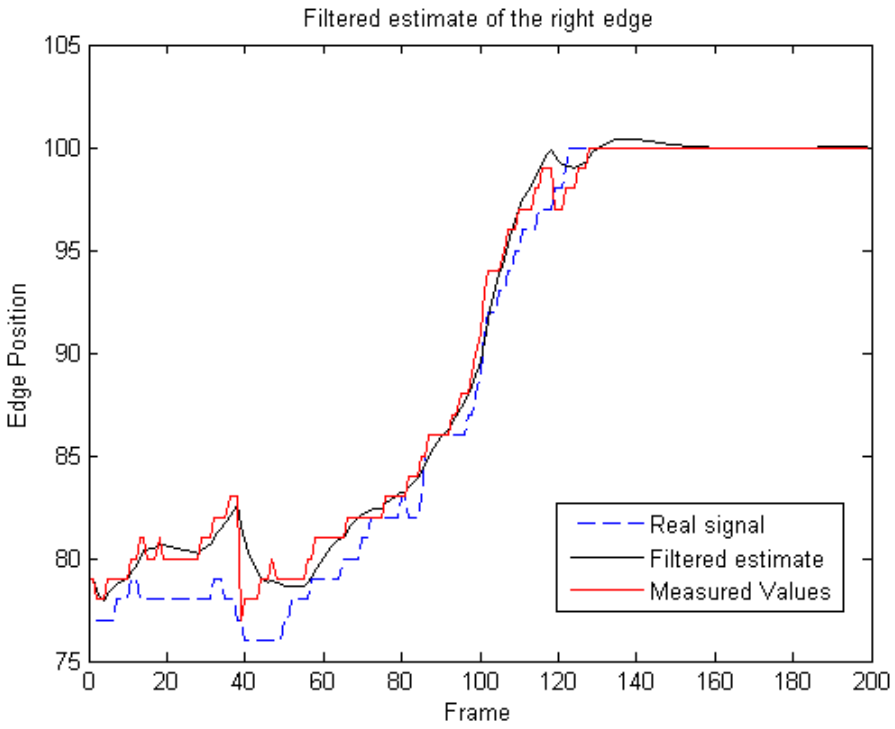

Figure 4.33: The measured and ground truth values of the right road edge can be seen.

The MSE did decrease with the right edge of the road. The Kalman filtered MSE was measured to be 1.58 where the error of the measured edge to the actual edge came to 1.602.

After frame 120 it can be seen that the road edge was out of the frame, and the detector used the edge of the road image as the road edge.

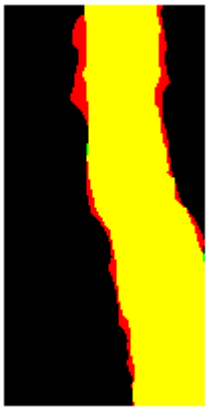

Figure 4.34: The Kalman filtered result of the road map. The red shows false positives, and the yellow shows correct classifications.

Figure 4.34 shows that the Kalman filter removes the jagged edges that can be seen in Figure 4.31 after the dilation, and blob retention. The Kalman filtered map is also seen to be optimistic in its boundary. This is not a problem in this application because the cost of the road segmentation being too large is a little extra time in mine detection, where a pessimistic boundary may miss a land mine all together.

5. IDEAS FOR THE FUTURE

This thesis has focused on color imagery detection of low contrast dirt or gravel roads. It could easily be expanded to include features from other image sources. The most obvious new source would be the IR camera. The IR imagery could be registered to the color camera, and formed into new features. Addition of more cameras also gives rise to depth features. One of the best ways to get depth features is through laser ranging or LADAR [4]. The purpose of this thesis, however was to deal with the absence of these additional sources of information.

The implementation of a method for classifying at a fixed distance from the vehicle, and translating that into a pixel value would improve the implementation of the road detector. In lane A, the ground is almost flat. The fixed pixel value works well enough for this case. In lane B, the vehicle's field of view changes vertically by an amount large enough to cause errors such as the classification line being pointed at the sky, and off the road all together. Implementing a distance measurement and transform would move the classification boundary up and down inversely to the field of view's motion.

One of the most obvious issues with this method is that the MATLAB implementation does not run in real time. The problem, however, is extremely parallelizable. The first improvement to this system would be to shift from the SIFT transformation method to the SURF method. SURF is easily implemented on a GPU which drastically reduces the required time for calculation by orders of magnitude. For one example: the MATLAB

implementation of SIFT used in this thesis requires 30 seconds per frame. This is completely unacceptable in theater. A SURF implementation that could take its place claims to only need 30 milliseconds [38]. To further reduce classification time, the features also could be trimmed down. This thesis did not focus on feature optimization, and included a large selection of features.

With increased power in hardware, the density of the road patch classification could also be increased. The horizontal classification distance could be reduced from 10 to 5 pixels with sufficient hardware power. This would give a finer mask of positive road detections, and could allow for more options in post processing. As the patches move in relation to the vehicle they get farther apart. More patches could be classified vertically to further increase the density of the road mask points.

6. CONCLUSION

Color imagery appears to contain sufficient information to find the boundary of a poorly maintained dirt road. The algorithm itself is, however, reliant on proper parameters being selected. There are an innumerable amount of tests that could be performed on this algorithm to increase its performance.

The results indicate that even in areas of poor classification appropriate post processing can greatly reduce errors. The convex hull, after suitable morphological filtering, can create a good representation of the road boundary. The Kalman filter helps to smooth the edge boundary as the vehicle commences down the test lane.

Overall the results are promising. The detector, in most cases, only has issues with the road detection when a human would also have difficulties.

The random forest is the classifier of choice for this application. Although it did not reach an error rate as low as the SVM was capable of, it did not require the tuning that the SVM needed. In theater there will be no time for a set of tests to be run to optimize the classifier. The random forest is the least picky of the two classifiers.

BIBLIOGRAPHY

[1] D. Pommerleau, "RALPH: Rapidly adapting lateral position handler," Proc. IEEE Intelligent Vehicles Symp. pp. 506-511, 1995.

[2] C. Taylor, J. Malik, J. Weber, "A Real-Time Approach to Stereopsis and Lane-Finding," Proc. IEEE Intelligent Vehicles Symp. 1996.

[3] J. Zhang, H. Nagel, "Texture-Based Segmentation of Road Images," Proc. IEEE Intelligent Vehicle Symp, 1994.

[4] C. Rasmussen, "Combining Laser Range, Color, and Texture Cues for Autonomous Road Following," Washington D.C. Proc. IEEE. Int. Conf. Robot Automation, pp. 4320-4325, 2002.

[5] G. Shi-yi Xie, G. Yi-xin Yin, "An Optimalizing Threshold Segmentation Algorithm for Road Images Based on Mathematical Morphology," IITA, pp. 518-521, 2009

[6] K. Stone, J. Keller, C. Spain, "Buried Explosive Hazard Detection Using FLIR Imagery," Proc. SPIE Defense, Orlando, 2011.

[7] C. Spain, M. Popescu, J. Keller, K. Stone, "Automatic Detection of Targets in Medium-Wave Infrared Imagery Using Adaptive Background Mixture Models," Proc. SPIE Defense, Orlando, 2011.

[8] M. Popescu, K. Stone, J. Keller, "Detection of Targets in Forward-Looking Infrared Imaging Using a Multiple Instance Learning Framework," Proc. SPIE Defense, Orlando, 2011.

[9] M. Popescu, A. Paino, J. Keller, K. Stone, "Detection of Buried Objects in FLIR Imaging using mathematical morphology and SVM," CISDA, Ottawa, 2012

[10] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," IJCV, pp. 91-110, 2004.

[11] D. Lewis, J. Keller, K. Stone, M. Popescu, "Dirt Road Segmentation Using Color and Texture Features in Color Imagery,"CISDA, Ottawa, 2012

[12] T. Ojala, M. Pietikainen, T. Maenpaa."Multiresolution Gray-Scale and Rotation Invarian Texture Classification with Local Binary Patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, pp. 971-987, 7, July 2002.

[13] C. Spain, "Automatic Detection of Explosive Devices in Infrared Imagery Using Texture with Adaptive Background Mixture Models," M.S. Thesis, University of Missouri: Columbia, 2011.

[14] N. Dalal, B. Triggs, "Histograms of Oriented Gradients for Human Detection," Computer Vision and Pattern Recognition. Vol. 1, pp. 886-893, 2005

[15] S. Sergyan, "Color Histogram Features Based Image Classification in Content-Based Image Retrieval Systems," Applied Machine Intelligence and Informatics, pp. 221-224, January 2008.

[16] C. Harris, M. Stephens, "A Combined Corner and Edge Detector," Fourth Alvey Vision Conference. pp. 147-151, Manchester, 1988.

[17] D. Lowe, SIFT demo program. [Software]. http://www.cs.ubc.ca/~lowe/keypoints/, 2005.

[18] H. Bay, T. Tuytelaars, L. Van Gool, "SURF: Speeded Up Robust Features" CVIU, Vol. 110, pp. 346-359, 2008.

[19] B. Boser, I. Guyon, V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," Proceeding of the Fifth Annual Workshop on Computational Learning Theory, pp. 144-152, Pittsburgh, 1992,.

[20] R. Sood, S. Kumar. "The Effect of Kernel Function on Classification," XXXII National Systems Conference, 2008.

[21] C. Burgess, "A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, pp. 121-167, 1998.

[22] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen. "Classification and regression trees (CART)," 1984.

[23] J. Guszcza, "CART from A to B. CAS Predictive Modeling Seminar," Chicago, 2005.

[24] A. Criminisi, J. Shotton, E. Konukoglu, "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning, Dimensionality Reduction, and Semi-Supervised Learning," [Online] http://research.microsoft.com/groups/vision, 2012

[25] A. Cutler, L. Breiman, "Random Forests," [Online] http://www.math.usu.edu/~adele/forests/cc_home.htm, 2012. Utah State University.

[26] Brieman, L. "Random Forests," 2001.

[27] The Math Works, Inc. Image Processing Toolbox Documentation. 2012.

[28] C. Barber, D. Dobkin, H. Huhdanpaa, "The Quickhull Algorithm for Convexhull," SIGGRAPH, p. 6, 2007.

[29] S. Srungarapu, D. Prasad, K. Kothapalli, P. Narayanan, "Fast Two Dimensional Convex Hull on the GPU," WAINA. pp. 7-12, Hyderabad, India 2011.

[30] S. Postalcioglu, K. Erkan, E. Bolat, "Comparison of Kalman Filter and Wavelet Filter for Denoising," ICNN&B. pp. 951-954, 2005.

[31] Han, T. Lecture Notes on Kalman Filters. [Online] http://web.missouri.edu/~hantx/ECE8001/notes/CCV_Lecture_Notes/CCV_Lect11_Kalman.pdf, 2012.

[32] G. Welch, G. Bishop. "The Kalman Filter," [Online] http://cs.unc.edu/~welch/kalman/, 2012.

[33] S. Sarkka, J. Hartikainen, A. Solin, EKF/UKF Toolbox for Matlab V1.3. [Software] 2011.

[34] A. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer, 2008. 3540401849.

[35] The Math Works, Inc. Global Optimization Toolbox Documentation. 2012.

[36] W. Abdulal, "Reliability-Aware Genetic Scheduling Algorithm in Grid Environment," CSNT, Hyderabad, India 2011.

[37] R. Duda, P. Hart. D. Stork. Pattern Classification. 2nd Edition. New York : Wiley, 2001.

[38] N. Comelis, L. Van Gool, "Fast Scale Invariant Feature Detection and Matching on Programmable Graphics Hardware," IEEE CVPR. pp. 1-8, 2008.