Visual Terrain Classification for Outdoor Mobile Robots

Visual Terrain Classification for Outdoor Mobile Robots

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

M.S. Yasir Niaz Khan
aus Faisalabad, Pakistan

Tübingen
2013

Tag der mündlichen Qualifikation:	24.01.2013
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. rer. nat. Andreas Zell
2. Berichterstatter:	Prof. Dr. rer. nat. Andreas Schilling

For my family

Abstract

In this thesis we present a comparison of multiple approaches to visual terrain classification for outdoor mobile robots based on local features. For this purpose, we put a camera on a mobile robot and use it to capture images which are then analyzed to recognize the terrains present in these images. There are two sets of approaches that we use to classify terrains. The first is based on greyscale images and the second one is based on color images.

For greyscale images, we use two different robot platforms for two different scenarios. The first robot platform is a wheeled outdoor robot. The second platform is a flying robot. For terrain classification, we modify and test three approaches called SURF, Daisy and Contrast Context Histogram, which are traditionally not used for texture classification. We compare these with more traditional texture classification approaches, such as Local Binary Patterns (LBP), Local Ternary Patterns (LTP) and a newer extension Local Adaptive Ternary Patterns (LATP). The image is divided into a grid and local features are calculated on the cells of this grid. These features are then used to train a classifier that can differentiate between different terrain classes.

Images of different terrain types are captured using a single camera mounted on a mobile outdoor robot. We drove our robot under different weather and ground conditions and captured data of different terrain types for our experiments. We did not filter out blurred images which occur due to robot motion and other artifacts caused by rain, etc. We used a Random Forest classifier for classification and cross-validation for the verification of our results. It is shown that SURF features perform better than other descriptors for smaller cell sizes and LTP works best for larger grid cell sizes. The results show that these approaches work well for terrain classification in a fast moving mobile robot, despite image blur and other artifacts induced due to variant weather conditions.

Furthermore we investigate the effectiveness of local image features for visual terrain classification for outdoor flying robots. A quadrocopter fitted with a single camera is flown over different terrains to take images of the ground below. Six different terrain types are considered in this approach. The images captured have artifacts like blur and scale variations. It is shown that SURF features also perform better here than other descriptors for smaller grid cell sizes and LTP performs better for larger cell sizes.

We also test color image based terrain classification. For this purpose, we use two different types of camera mounted on our wheeled outdoor robot and capture five different terrain types traversed by the robot. We use two different image descriptors that can work on color images. The first descriptor is the co-occurrence matrix and the second descriptor is the SURF descriptor. Each of these descriptors is applied to the color channels of the image to extract a feature vector. These features are then used to train and test classifiers like Random Decision Trees and Support Vector Machines. We test these classification techniques on different color spaces for the images containing terrains. We also apply Principle Component Analysis (PCA) to reduce the dimensionality of the feature vectors. The co-occurrence matrix produced the best result in this case.

Kurzfassung

In dieser Arbeit werden mehrere Ansätze zur visuellen Terrainklassifizierung für mobile Roboter vorgestellt und verglichen. Dafür wurden monochrome und farbige Bilder, die von mobilen Outdoor-Robotern aufgenommen wurden, mit Hilfe von lokalen Merkmalen analysiert, um verschiedene Terraintypen zu erkennen.

Für die Analyse der monochromen Bilder wurden zwei verschiedene Roboter-Plattformen mit zwei verschiedenen Szenarien verwendet. Bei der ersten Plattform handelte es sich um einen fahrenden Outdoor-Roboter und bei der zweiten Plattform um einen fliegenden Roboter. Bei der Terrainklassifizierung wurden zum einen die klassischen Ansätze Local-Binary-Patterns, Local-Ternary-Patterns und die neue Erweiterung Local-Adaptive-Ternary-Patterns verwendet. Zum anderen wurden neuere Verfahren wie SURF, Daisy und Kontrast-Kontext-Histogramme untersucht und miteinander verglichen. Das Bild wurde dafür in ein Gitter aufgeteilt, an dessen Schnittpunkten die Deskriptoren berechnet wurden. Anhand dieser Merkmale wurden anschließend Klassifikatoren trainiert, um die verschiedenen Terraintypen zu unterscheiden.

Die Bilder für die Klassifizierung wurden für verschiedene Bodentypen bei unterschiedlichen Witterungen aufgenommen. Dabei wurden Bilder mit Bewegungsunschärfe oder anderen Artefakten, wie Regen, nicht herausgefiltert. Als Klassifikator kamen Random-Forests zum Einsatz, deren Ergebnisse über eine Kreuzvalidierung verifiziert wurden. Dabei zeigten SURF-Features bei kleinen Gittergrößen die besten Ergebnisse, wogegen bei großen Gittern die Local-Ternary-Patterns am besten abschnitten. Bei diesen Versuchen hat sich gezeigt, dass die Terrainklassifizierung durch einen sich schnell bewegenden mobilen Roboter, auch bei Bewegungsunschärfe und bei Wettereinflüssen, sehr gute Ergebnisse erreicht.

Außerdem wird in dieser Arbeit die visuelle Terrainklassifizierung auf Basis von lokalen Bildmerkmalen bei fliegenden Robotern untersucht. Ein Quadrocopter mit einer einzelnen Kamera wurde dafür über verschiedene Terraintypen geflogen und nahm dabei Bilder des Bodens auf. Die so entstandenen Bilder haben Artefakte wie Bewegungsunschärfe und durch die Flughöhe eine unterschiedliche Skalierung. Auch bei diesen Versuchen schnitten die SURF-Deskriptoren bei kleinen Gittern und die Local-Ternary-Patterns bei größeren Gittern am besten ab.

Zuletzt wurde die Terrain-Klassifizierung mit Hilfe von Farbbildern untersucht. Dafür wurden eine Farb- und eine Monochromkamera auf einen fahrenden Outdoor-Roboter montiert und fünf verschiedene Terraintypen aufgenommen. Es kamen zwei verschiedene Deskriptoren für Farbbilder zum Einsatz. Der erste Deskriptor ist die Co-Occurrence-Matrix und der zweite der SURF-Deskriptor. Jeder der Deskriptoren wurde auf die Farbkanäle der Bilder angewendet, um einen Merkmalsvektor zu generieren. Diese Merkmale wurden dann mit Klassifikatoren wie Random-Decission-Trees und Support-Vector-Machines für unterschiedliche Farbräume trainiert und getestet. Mit Hilfe der Principle-Component-Analysis wurde außerdem die Dimension der Merkmalsvektoren verkleinert. Bei diesen Versuchen zeigte die Co-Occurrence-Matrix die besten Ergebnisse.

Acknowledgments

I would like to thank Prof. Dr. Andreas Zell for providing a great environment and constant guidance to do good quality of research. I would thank Prof. Dr. Andreas Schilling for being the second supervisor of my thesis. Appreciation for Klaus Beyreuther, who keeps the robotic equipment, our network and cluster up and running with constant effort; and for Vita Serbakova, who takes care of all of our official processes and documentation.

I would also like to thank my colleagues who make the group environment pleasant and for giving tips frequently. Especially Philippe Komma, Stefan Laible, Georg Hinselmann, Philipp Vorst and Sebastian Scherer. Thanks to Karsten Bohlmann and Henrik Marks for keeping our outdoor robots in top shape. Thanks to Karl E. Wenzel and Andreas Masselli for flying the robots for my experiments. The work of a diploma thesis also contributed to this work. Thanks to Julian Jordan for his excellent work in this diploma thesis. Also thanks to Stefan Laible, Jacobo Jimenez and Artur Koch for reviewing parts of my thesis. Thanks to Blaine Nelson, Florian Mittag, Dr. Andreas Dräger, Nedim Srndic and Duc My Vo for an excellent fun environment at work.

Special mention of my wife is needed here, who supported me and gave me peace of mind to work. This work would not have been possible without my parents who provided me support during my studies. And most of all thanks to the one God: Allah, who created me and enabled me to serve my fellow human beings. He also sent Prophet Muhammad (pbuh) who showed me the best example of a life, which taught very high values of humanity and morality, and how to treat my colleagues and friends.

Contents

Chapter 1

Introduction

Estimation of the ground surface is essential for a safe traversal of terrain for an outdoor autonomous robot. It is employed for a variety of outdoor assignments, such as rescue missions or surveillance operations (Nüchter *et al.*, 2005), taking care of the elderly, etc (Mehdi *et al.*, 2011, 2009).

In this thesis we present a comparison of multiple approaches to visual terrain classification for outdoor mobile robots based on local features. We drove our wheeled outdoor robot under different weather and ground conditions and captured images of different terrain types for our experiments. These images have multiple artifacts, like blur, which is due to robot motion, and other artifacts caused by rain, shadows, etc. We used Random Forests for classification, and cross-validation for the verification of our results. The results show that most of the approaches work well for terrain classification in a fast moving mobile robot, despite image blur and other artifacts induced due to extremely variant weather conditions.

1.1 Motivation

When operating ourdoors, a robot must be aware of ground surface hazards induced by the presence of slippery and bumpy surfaces (Iagnemma *et al.*, 2004; Iagnemma and Dubowsky, 2004). These hazards are known as non-geometric hazards (Wilcox, 1994) and pose difficulties to robot movement on this rough terrain (Lamon and Siegwart, 2005; Lamon *et al.*, 2006). There are also many objects present on the ground which act as obstacles and hinder the movement of the robot (Matthies *et al.*, 1995; Milella *et al.*, 2006) including negative obstacles (Matthies and Rankin, 2003). These environments are also called unstructured environments, as the components of this environment are not placed in an order and change unpredictably (Otte *et al.*, 2007; Sun *et al.*, 2006).

Terrain identification techniques can be classified into at least two different groups: retrospective and prospective terrain identification. Whereas retrospective techniques predict the current ground surface from data recorded during robot traversal (Iagnemma and Dubowsky, 2002; Komma *et al.*, 2009a), prospective techniques classify terrain sections that are located on the current path, i.e. in front of the robot. The latter approaches can rely on the environment's geometry at short and long range acquired using either

LADAR sensors (Vandapel *et al.*, 2004) or stereo cameras (Bajracharya *et al.*, 2008; Schauwecker *et al.*, 2012; Chang and Lee, 2005). Stereo cameras are good in that they are not very expensive (Iocchi *et al.*, 2000). However, they suffer from the disadvantages of having a short range and a very complex computation model. Vibration based terrain classification is a method thoroughly examined (Komma *et al.*, 2009b; Komma and Zell, 2009, 2010a,c; Brooks *et al.*, 2005) in literature using techniques such as Markov Random Fields (Chatzis and Tsechpenakis, 2009) and weighted gradients, etc. (Weiss *et al.*, 2007b,a,c; Weiss and Zell, 2008; Weiss *et al.*, 2008). Another related research has been done for classifying plants based on 3D laser data (Weiss *et al.*, 2010). Laser sensors are also used for terrain classification in the past (Andersen *et al.*, 2006; Hebert and Vandapel, 2003; Macedo *et al.*, 2000). Yet, classifying terrain based on geometrical reasoning alone gives rise to ambiguities which cannot be resolved in some situations. For example, tall grass and a short wall provide similar geometrical features. Additionally, laser sensors are generally expensive devices. Furthermore, stereo cameras only yield little information at long range. This information, however, is important for generating pathways which safely guide the robot toward distant targets (Brooks *et al.*, 2006; Brooks and Iagnemma, 2007) or are useful in mapping and localization (Wolf *et al.*, 2005b, 2002a).

Hence, in this thesis, we consider another class of prospective terrain classification techniques which relies on texture features acquired from monocular cameras. In comparison with geometrical features, these texture features provide meaningful information about the ground surface even at long-range distances. Some work has been done in this regard (Angelova *et al.*, 2007a). Using the extracted visual cues we then apply a Random Forests based approach to the problem of terrain classification. That is, after training a model which establishes the assignment between a visual input and its corresponding terrain class, this model is then employed to predict the ground surface of a respective visual clue. As in (Dima *et al.*, 2004; Kim *et al.*, 2006), texture features are extracted from image patches which are regularly sampled from an image grid. We perform terrain classification on a patch-wise basis rather than on a pixel-wise basis because the latter tends to produce noisy estimations which complicates the detection of homogeneous ground surface regions (Davis *et al.*, 1995). For outdoor robots, it is very useful to know about ground surfaces for many purposes (Birk *et al.*, 2007; Booij *et al.*, 2007; Bradley *et al.*, 2005).

Terrain classification can be performed by a wheeled robot (Komma and Zell, 2010b; Bradley *et al.*, 2007), but it may not easily reach all areas and some areas may be hazardous for a wheeled robot. Hence, we also use a flying robot (Wenzel *et al.*, 2010a, 2009) to test our terrain classification approaches, since it can fly over almost any area and is not harmed or blocked by ground hazards. Employed for a variety of outdoor assignments, such as rescue missions or surveillance operations, a flying robot should also be able to recognize ground surfaces for successful completion of several outdoor tasks (Spero, 2004). The classification data gives important information about possible places to land, it can be used to guide a wheeled robot along safe paths, or it can be stored in a map for later use of robots or humans (Bailey, 2002).

A flying robot can also be used in combination with a wheeled robot. Most of the flying robots are small vehicles which cannot carry a big payload. They can only carry a limited number of sensors. They can also not carry big batteries, hence their flying time is limited with one full charge of the batteries. So, flying robots are good for flying over an area and making a map. Then a wheeled robot can go in and carry out some tasks in that area. This can become an effective system for complex tasks, such as mapping (Wolf *et al.*, 2005a, 2002b) or localization by a swarm of robots (Kronfeld *et al.*, 2010).

1.2 Outline

The remainder of this thesis is organized as follows: Chapter 2 presents the various machine learning techniques used to learn and classify the different image descriptors. Chapter 3 briefly summarizes the adopted techniques for representing acquired terrain patches in terms of meaningful image descriptors. These image descriptors constitute the basis on which the terrain classification relies. Chapter 4 introduces the grid-based approach that we have used in our work. All of the image descriptors are computed based on this grid. In chapter 5 we provide details of the equipment and environment used in our classification experiments. This includes both of our wheeled and flying robots, as well as the operating environment. Our experiments using grey-scale images are described and discussed in chapter 6. Techniques used for terrain classification in colored images are presented in chapter 7. Finally, chapter 8 gives summary of the thesis and conclusions.

Chapter 2

Machine Learning Schemes

We performed the classification task using several classifiers (Mitchell, 1997). Therefore, we used the machine learning software Weka (Hall *et al.*, 2009) to train and test these classifiers. The adopted classifiers were Random Forests, Support Vector Machine (SVM) using the Sequential Minimal Optimization (SMO) training algorithm (Platt, 1999), the Multilayer Perceptron (MLP), Linear SVM, J48 Decision Tree, Naive Bayes and k-Nearest Neighbor. All of them are effective tools of machine learning and many are good for novelty detection (Markou and Singh, 2003a).

2.1 Naive Bayes

The Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (Zhang and Su, 2004; Zhang, 2004). This theorem is applied here with the assumption of a strong (naive) statistical independence. It assumes that any particular feature of a class is unrelated to any other feature.

Considering a dependent class variable C with a small number of outcomes which depends on a number of features $F_1, F_2, ..., F_n$. Using Bayes' Theorem, we can state:

$$p(C|F_1, F_2, ..., F_n) = \frac{p(C)\ p(F_1, F_2, ..., F_n|C)}{p(F_1, F_2, ..., F_n)} \tag{2.1}$$

Pratically, only the numerator of equation ((2.1)) is of interest, since the denominator does not depend on

$$C$$

and the values of features

$$F_i$$

are given. The numerator is the same as the joint probability model:

$$p(C, F_1, F_2, ... \tag{2.2}$$

When we consider the naive conditional independence, we assume that each feature F_i is independent of every other feature F_j for $j \neq i$. So we get

$$p(F_i|C,F_j) = p(F_i|C) \tag{2.3}$$

for $j \neq i$. Solving this further results into the following conditional distribution over the class variable C:

$$p(C|F_1,F_2,...,F_n) = \frac{1}{Z}\, p(C) \prod_{i=1}^{n} p(F_i|C) \tag{2.4}$$

where Z is a scaling factor called the evidence dependent on all of the features.

Naive Bayes is an efficient classifier with good speed and results (Rish, 2001). Bayesian techniques are also used in image processing (Geman and Geman, 1987) and other areas of robotics (Ollis *et al.*, 2007).

2.2 K-Nearest Neighbor

K-nearest neighbor (k-NN) algorithm is a classification technique which classifies objects based on the closest training pattern in the feature space efficiently (Cover and Hart, 1967). It is one of the most simple classification algorithms, as there is no training involved. The training samples are just stored with their class labels. Each test sample is matched with k nearest training samples to determine the class. k is an arbitrary number specifying the number of neighbors to be considered. Large values of k reduce the influence of noise in the classification process but enforce smoother boundaries. A good choice of k depends on the application. For $k = 1$ the algorithm gets reduced to a simple nearest neighbor algorithm, that associates the sample to the class of the closest matching instance in the training data. For two class problems, k is chosen to be an odd number to avoid tied classification.

Although prone to noisy data, the nearest neighbor algorithm is a very stable algorithm. It is guaranteed to give an error rate not worse than twice the Bayes error rate, which is the minimum error rate depending on the data distribution. k-NN can also be used for continuous variables. In this case, the inverse distance weighted average is used to estimate the distance of the neighbors. There are many modifications of the k-NN algorithm in use. (Zhang *et al.*, 2006)

2.3 K* Nearest Neighbor

K* (Cleary and Trigg, 1995) (also called K*-NN) is an instance-based classifier. A test instance is classified based upon the class of those training instances similar to it as determined by some similarity function. Most of the similarity based classifiers have problems with missing values in the data. K* differs from other instance-based learners in that it uses an entropy based distance function. In this approach, the distance between two instances is defined as the complexity of transforming one instance into the other. This calculation involves two steps. The first is to define a finite set of transformations that map instances to other instances. Usually this is done by taking a single shortest transformation out of many possible transformations, but this is too sensitive to small changes in the instance space. So in K*, a summation is done over all transformations by associating a probability with each of them. Complexity is then calculated by taking the logarithm of this sum.

Let I be a set of instances and T a set of transformations on I. Each transformation $t \in T$ maps an instance to another $t : I \rightarrow I$. A special element σ in T maps instances to themselves for completeness $\sigma(i) = i$. The set of all prefix codes from $T*$ terminated by σ is denoted as P. Members of $T*$ define a transformation on I:

$$\bar{t}(A) = t_n(t_{n-1}(...t_1(a)...)) \text{ where } \bar{t} = t_1, t_2, ..., t_n$$

The probability function P* defines the probability of all paths from instance a to instance b:

$$P^*(b|a) = \sum_{\bar{t} \in P : \bar{t}(a) = b} p(\bar{t})$$

The K* function can then be defined as:

$$K^*(b|a) = -log_2 P^*(b|a)$$

The K* function holds the following properties:

$$K^*(b|a) \geq 0$$

$$K^*(c|b) + K^*(b|a) \geq K^*(c|a)$$

2.4 C4.5/J-48 Decision Tree

This classifier is based on the famous C4.5 algorithm developed by J. Ross Quinlan (Quinlan, 1993), which is an extension of Quinlan's earlier ID3 algorithm. This algorithm generates a decision tree which is used for classification. Decision trees are a good way to represent information from a machine learning algorithm, which is a fast and powerful method to denote structures in data. C4.5 builds decision trees from training data using the concept of information entropy (Quinlan, 1996). J48 is an implementation of the C4.5 algorithm.

Considering a training set $T = t_1, t_2, t_3, ...$ of samples with their class labels. Each training sample consists of feature values such that $t_i = x_i, x_2, x_3, ...$ The class label of each training sample is also provided in a vector $C = c_1, c_2, c_3, ...$ When the decision tree is constructed, each node is generated out of a feature that most effectively splits the data samples in the subsets of the classes. The feature chosen is the one which provides the most information gain for splitting the data. The algorithm then recurses on smaller sublists.

Pruning is performed on the decision trees to reduce complexity and increase speed efficiency. Pruning also reduces the potential of over-fitting on the training data. It does, however, slightly reduces the accuracy of the decision trees. Therefore, the decision tree is gradually generalized until it reaches a balance of flexibility and accuracy. J48 implementation uses two methods for tree pruning. The first method applies subtree replacement, where a node is replaced by a leaf. This reduces the number of levels of tests to be performed on that path. The second method is called subtree raising, where a node is moved upwards in the tree replacing other nodes. This is a complex and time consuming method but has little effect on the accuracy.

2.5 Support Vector Machines

Support vector machines (SVMs) (Cortes and Vapnik, 1995; Burges, 1998) are supervised learning models with associated learning algorithms that analyze data and recognize patterns. They take an input vector and determines its class out of the two classes in question. Thus it is a linear binary classifier. The SVM makes a model based on the training data and then classifies further input data into the learned classes. The underlying model consists of points in space representing training examples. These examples are mapped into separate categories with wide boundaries between them. New input data are also mapped and then predicted to belong to a category based on their closeness. SVM actually creates a set of hyperplanes in a high dimensional space. It can also perform non-linear classification using a technique called kernel trick, i.e. mapping their input into high dimensional feature spaces (Chang and Lin, 2005; Erästö, 2001; Hsu *et al.*, 2003). SVMs are extensively used for machine learning in different fields of science (Mittag *et al.*, 2012).

The hyperplane algorithm by the SVM developer (Vapnik and Lerner, 1963) was a linear classifier. Later Vapnik and others (Boser and Vapnik, 1992) proposed to use the kernel trick to create a non-linear classifier. The main algorithm remains the same except that every dot product is replaced by a non-linear kernel function. This allows for a fitting of the maximum margin hyperplane in a transformed feature space. Using a Gaussian radial basis function as a kernel results in the feature space becoming a Hilbert space of infinite dimensions. Some of the commonly used kernels are as follows:

- Polynomial (homogeneous): $k(x_i,x_j) = (x_i \cdot x_j)^d$
- Polynomial (inhomogeneous): $k(x_i,x_j) = (x_i \cdot x_j + 1)^d$
- Gaussian radial basis function: $k(x_i,x_j) = exp(-\gamma||x_i - x_j||^2)$ for $\gamma > 0$
- Hyperbolic tangent: $k(x_i,x_j) = tanh(\kappa x_i \cdot x_j + c)$ for some $\kappa > 0$ and $c < 0$

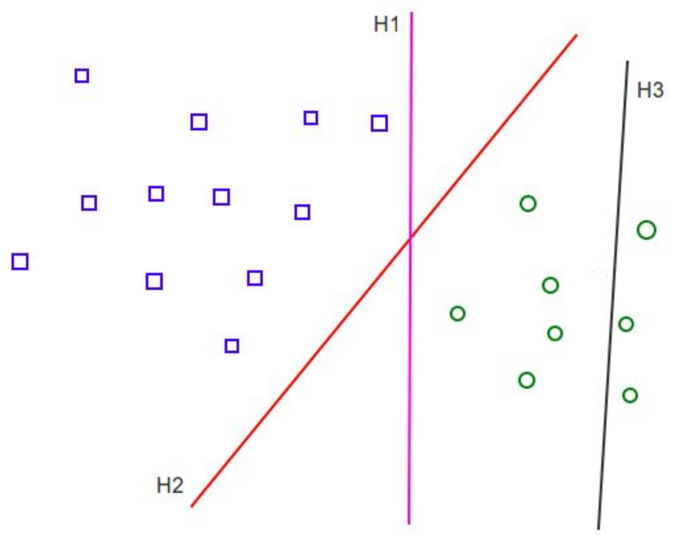


Figure 2.1: Classification of a dataset consisting of circles and squares with hyperplanes. Hyperplane H1 and H2 classifies them correctly, whereas, H3 does a wrong classification. H2 offers a better linear classification since it has maximum distance to the sample points.

2.6 Linear SVM

A linear classifier is a classifier which performs the classification by a linear combination of the features of the dataset. It is a very fast method to solve classification problems of extremely large datasets. It is supposed to be linearly scalable, which means that it uses an SVM model with a running time that scales linearly with increasing size of a typical dataset. Thus it can solve multi-class classification problems in linear time. Different kind of SVMs have been used often in pattern classification (Osuna *et al.*, 1997; Joachims, 1998; Weiss *et al.*, 2007d).

Considering $\overrightarrow{x}$ as the input feature vector, the output of a typical linear classifier can be specified as:

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right)$$

where $\vec{w}$ is the weight vector and f is the function that converts the dot product of the two vectors into the desired output. A lot of people are contributing improvements to SVMs (Schölkopf *et al.*, 2000; Schölkopf *et al.*, 2000; Platt, 2000) in different fields such as biology, chemistry, etc (Hinselmann *et al.*, 2011b).

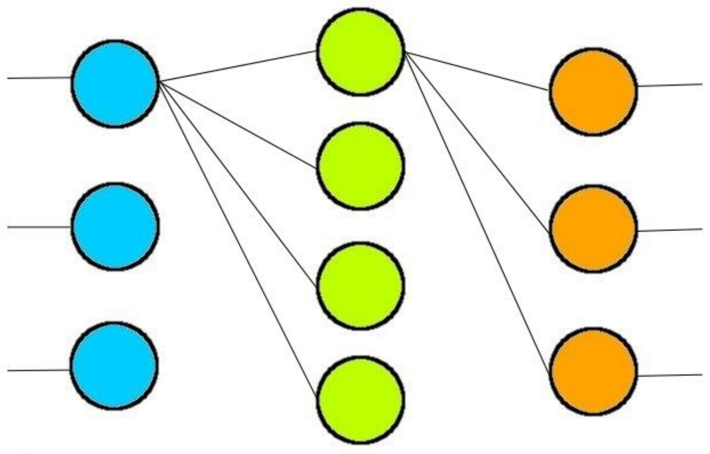


Figure 2.2: A typical 3-layer Neural Network. For clarity, connections of only the first neuron of every layer are shown.

2.7 Neural Networks

Neural Networks are computational models based on the biological neural networks in the brain. They consist of layers of neurons connected to each other which process and pass data among themselves to produce an output. This is a connectionist approach of learning. Neural Networks are used to model complex non-linear pattern classification (Bishop, 1995, 1994) and are used for multiple applications (DuPont *et al.*, 2005a, 2006).

The artificial neurons in one layer take some input, process it and pass it on to the next layer. Such neural networks are also a good tool for novelty detection (Markou and Singh, 2003b; Specht, 1990). Weights are assigned to the connections between different neurons. These weights alter the input when passed on between neurons. Based on the feedback of the output, the weights are adjusted to get a better output. When the inputs finally reach the output layer, an activation function is applied on them to determine the final output. A typical neural network consists of three layers: input layer, hidden layer and output layer as shown in Fig. 2.2. The input layer receives the input and passes it on to the hidden layer after applying the associated weights. The hidden layer does the same procedure to pass the input to the output layer. The output layer finally applies the activation function to generate the output. More complex neural networks have more layers and more complex connections. An activation function is applied on every neuron to determine the output of the neuron. Many different activation functions have been used in literature. One common activation function is the sigmoidal function, which can described in the following form:

$$f(y_i) = (1 + exp(-v_i))^{-1}$$

which is a logistic function.

Once the output is calculated against an input, it is compared to a desired output in the case of a supervised network, and the difference of the two outputs is used to improve the network. There are many learning algorithms which are used in neural networks, mostly gradient descent class of algorithms. Derivative of the cost function is calculated in such cases according to the parameters and the network parameters are then changed in a gradient descending direction. Some of the commonly used algorithms are simulated annealing, evolutionary algorithms, expectation maximization, etc. Hence, neural networks are an effective algorithm for pattern classification (Khan and Mehdi, 2002), including different approaches to terrain classification (DuPont *et al.*, 2005b, 2008).

2.8 Random Forest

Decision Trees (Quinlan, 1986, 1993) have shown their applicability in various classification tasks (Birk *et al.*, 2008; Wilking and Röfer, 2005). Random Forests (Breiman, 2001) are an ensemble classifier consisting of a collection of individual decision tree predictors. These binary trees are grown randomly with controlled variation (Breiman, 1996). That is, for each tree an individual bootstrap sample is drawn. Further, each node of a tree uses a varying feature subset of the complete feature set on which a binary decision is based on. Given an input vector, this test decides whether to traverse the left or right child of the tree. Leaf nodes are assigned the actual class labels. If during tree traversal a leaf node is reached, the tree casts a unit vote for the class represented by the leaf label. The class with the largest number of votes is then defined as the predicted class.

Concerning prediction accuracy, using a larger number of trees reduces the generalization error for forests. However, this also increases the run-time complexity of the classification process. Hence, a compromise has to be found between accuracy and speed by varying the number of trees. We found that in our case 100 trees gave good accuracy without a significant decrease in speed.

Random forests (Lepetit and Fua, 2006; Ozuysal *et al.*, 2007) try to reduce their problem of over-fitting by injecting randomness into the tree generation procedure and combining the output of multiple randomized trees into a single classifier. The trees are established by recursively bisecting the data set into smaller subsets at each inner node R_i. As splitting criterion the Gini-index (Gini, 1913) is employed, which is defined by:

$$I_G(i) = \sum_{j=1}^{k} \hat{p}_{ij}(1 - \hat{p}_{ij}),$$

where k is the number of classes to discriminate and $\hat{p}_{ij}$ denotes the probability of observing a measurement of class j with respect to all instances provided for node R_i. At each splitting step, the remaining data is separated into two distinct subspaces or subnodes, R_{c_1} and R_{c_2}, using a random feature subset.The splitting procedure is recursively adopted until a maximum tree depth is reached. Random forests classifiers grow trees of maximum depth without performing subsequent pruning steps. After tree generation, each leaf node stores several instances along with their respective class membership. The latter can be adopted to assess the posterior distribution $p(c = k^*|x_i)$:

$$\begin{aligned} p(c = k^*|x_i) &= F(\{t_1,\ldots,t_N\},x_i,k^*) \\ &= \frac{1}{N}\cdot\sum_{j=1}^{N}\frac{N_r + f(t_j,x_i,k^*)}{K\cdot N_r + \sum_{l=1}^{K} f(t_j,x_i,k_l)}, \end{aligned}$$

where $f(t_j,x_i,k_l)$ denotes the number of estimation examples which belong to class k_l and which are assigned to the same leaf as instance x_i in t_j. In this work the approaches of Breiman (1996) and Breiman (2001) have been followed which suggest to choose the estimation examples to be identical to the original set of training examples. If an instance assigned to a specific leaf node is not encountered during training, the inclusion of the additional terms assigns a non-zero value to the corresponding probability.

During the recall phase, the test pattern traverses each random tree until a leaf node is reached. The posterior distributions assigned to the respective nodes are then averaged over all members of the ensemble. Finally, the class k^* which maximizes $p(c = k^*|x_i)$ is chosen to be the classification result of the test pattern. Free parameters such as the patch size and the hyper-parameters for all the applied classification approaches were found using a grid-search. Using a larger number of trees reduces the generalization error for random forests. However, this also increases the run-time complexity of the classification process. Hence, a compromise has to be found between accuracy and speed by varying the number of trees. We found that in our case 50 trees gave adequate accuracy without a significant loss in speed. We adopted a 10-fold cross-validation scheme to verify the accuracy of the results.

2.9 Weka Tool

Weka is a Java based tool incorporating a large number of machine learning algorithms developed at the University of Waikato, New Zealand (Hall *et al.*, 2009; Witten *et al.*, 2011). It has algorithms in many categories of machine learning techniques, like function based, rule based, trees, Bayes, etc. It has a mechanism to train and test the algorithm on a given dataset. This can be done by splitting the dataset into training and test subsets, or by using cross-validation. Data can be visualized using the built-in visualizer. It has a graphical interface completely built in Java so it can be run on any system with a Java VM. The data to be processed is prepared in a text file with a specific format (Attribute Relationship File Format, ARFF). This data file is then loaded into a database in the tool and various machine learning operations are performed. There is also a command line based interface for batch processing or automatic processing. Through this interface, we were able to run the software on our cluster in parallel using batch processing. This was needed to find the best parameters of descriptors and optimum grid size of images.

Chapter 3

Image Descriptors

Following are the image descriptors that were used in this work. Some of them are texture based and others are interest point based. There are many approaches for texture or pattern recognition in literature, some of which consist of visual features (Bishop, 2006; Duda *et al.*, 2001; Deselaers, 2003) and others of lidar features (Castano *et al.*, 2001; Castano and Matthies, 2003). Some other techniques try to fuse multiple inputs to get a better performance (Früh and Zakhor, 2003, 2002) and some use various clustering techniques (Giguere and Dudek, 2008). Pattern recognition is a very important field in computer science and finds its use in many different areas (Nabney, 2002).

3.1 SURF

Speeded Up Robust Features (SURF) (Bay *et al.*, 2006) are an extension of the famous SIFT features (Lowe, 2004). SURF is used to detect interest points in a greyscale image and to represent them using a 64- or 128-dimensional feature vector. These features can then be used to track the interest points across images and thus prove suitable for localization tasks. In this paper, we considered SURF features for a new application: texture classification. In SURF interest points are detected across the image using the determinant of the Hessian matrix. Box filters of varying sizes are applied to the image to extract the scale space. Then the local maxima are searched over space and scale to determine the interest points at the best scales. The key-point extraction capabilities of SURF, however, have been omitted. This is because the interest points detected by SURF are usually concentrated around sharp gradients, which are likely not present within homogeneous terrain patches. Instead we manually choose the interest point location and scale from which the SURF descriptor is determined. This renders our approach much faster.

In our approach we divide the image in a grid and use the generated patches or subwindows to calculate the descriptors. Each image patch is then classified individually. We use 64-dimensional Upright-SURF (U-SURF) descriptors, in which the rotation invariance factor is removed. Still they are rotation invariant up to +/-15 degrees. Furthermore, we only consider a single scale for descriptor extraction which was determined experimentally using a grid-search approach. We call this modified approach TSURF or

Terrain-SURF. It can also be called as GSURF or Grid-SURF denoting the calculation of SURF features across a grid. The SURF descriptor describes how the pixel intensities are distributed within a scale dependent neighborhood of each interest point. Haar wavelets are used to increase robustness and speed over SIFT features. First, a square window of size 20σ is constructed around the interest point, where σ is the scale of the descriptor. The descriptor window is then divided into 4×4 regular subregions. Within each subregion, Haar wavelets of size 2σ are calculated for 25 regularly distributed sample points. If x and y wavelet responses are referred by dx and dy respectively, then for the 25 sample points,

$$v_{subregion} = [\sum dx, \sum dy, \sum |dx|, \sum |dy|]$$

are collected. Hence, each subregion contributes four values to the descriptor vector resulting in a final vector of length 64 ($4\times4\times4$).

SURF features are used for many applications just like the SIFT features (Barfoot, 2005).

3.2 Local Binary Patterns

Local Binary Pattern (LBP) (Ojala *et al.*, 1996) is a very simple, yet powerful texture descriptors. It is a local descriptor which is computed by thresholding the neighborhood of a pixel and then using the bit pattern produced as a descriptor. A 3×3 window is placed over each pixel of a grayscale image and the neighbors are thresholded based on the center pixel. A bigger neighborhood window can also be used depending on the need (Ojala *et al.*, 2002). Neighboring pixels having a value greater than the center pixel are assigned a value of 1, otherwise 0. Then the thresholded neighbors are concatenated to create a binary code which defines the texture at the considered pixel. These patterns are calculated for all or a specific number of pixels in the image.

For our application, we divide the image into a grid and calculate a histogram of binary patterns of each pixel within a patch. Thus each grid cell yields a histogram which is then used to assign a terrain class to the respective cell. Since the 8-bit binary pattern can have 256 values, we have a histogram containing 256 dimensions for classification.

Below is an example of a 3×3 neighborhood of a pixel in an image. Thresholding is performed with respect to the center pixel to obtain a binary pattern. The resulting bits are combined starting from the top left pixel and combining in a clockwise direction. Any pixel can be taken as the starting pattern and any direction can be used, as long as it is constant for all pixels.

94	38	54
23	**50**	78
47	66	12

1	0	1
0		1
0	1	0

Binary Pattern = 10110100

Many extensions of LBP have been presented in literature including usage in supervised and unsupervised segmentation (Ojala and Pietikaeinen, 1997) and using multiple blocks of neighborhood (Liao *et al.*, 2007). LBP have been tested on a big robot for off-road navigation in (Zolynski *et al.*, 2008).

3.3 Local Ternary Patterns

Local Ternary Patterns (LTP) (Tan and Triggs, 2007) are a generalization of Local Binary Patterns. Here, a ternary pattern is calculated by using a threshold k around the value c of the center pixel instead of generating a binary pattern based on the center pixel. Neighboring pixels greater than $c+k$ are assigned a value of 1, smaller than $c-k$ are assigned -1, and values between $c+k$ and $c-k$ are mapped to 0.

$$T = \begin{cases} 1 & T \geq (c+k) \\ 0 & T < (c+k) \; and \; T > (c-k) \\ -1 & T \leq (c-k) \end{cases} \tag{3.1}$$

where c is the intensity of the center pixel.

Instead of using a ternary code to represent the 3×3 matrix, the pattern is divided into two separate matrices. The first one contains the positive values from the ternary pattern, and the second contains the negative values. From both matrices a LBP is determined resulting in two individual matrices of LBP codes. Using these codes two separate histograms are calculated. In this approach, we also divide the image into a grid and calculate histograms for each cell. The two histogram parts are concatenated to form a histogram of 512 dimensions. (Tan and Triggs, 2010).

Below is an example of a 3×3 pixel pattern of an image. A threshold parameter ($k=5$) is used to obtain a ternary pattern, which is then divided into two binary patterns:

94	38	54
23	**50**	78
47	66	12

1	-1	0
-1		1
0	1	-1

Ternary Pattern ($k=5$): $1(-1)01(-1)10(-1)$

Part-1=10010100, Part-2=01001001

3.4 Local Adaptive Patterns

Local Adaptive Ternary Patterns (LATP) (Akhloufi and Bendada, 2010) are based on the Local Ternary Patterns. Unlike LTP, they use simple local statistics to compute the local pattern threshold. This makes them less sensitive to noise and illumination changes. LATP have been successfully applied to face recognition in (Akhloufi and Bendada, 2010). We test this operator in the domain of texture classification. The basic procedure is the same as LTP. Instead of a constant threshold, the threshold (T) is calculated for each local window using local statistics as given in the equation:

$$T = \begin{cases} 1 & T \geq (\mu + k\sigma) \\ 0 & T < (\mu + k\sigma) \; and \; T > (\mu - k\sigma) \\ -1 & T \leq (\mu - k\sigma) \end{cases} \tag{3.2}$$

where μ and σ are mean and standard deviation of the local region, respectively, and k is a constant.

The resulting ternary pattern is divided into two binary patterns like LTP and separate histograms are calculated and concatenated for classification forming a 512 dimensional vector. Below is an example of such pattern calculation:

94	38	54
23	**50**	78
47	66	12

1	0	0
-1		1
0	0	-1

Ternary Pattern ($k = 1$): $1001(-1)00(-1)$
Part-1=10010000, Part-2=00001001
In this case: $\mu = 51.33$, $\sigma = 25.74$
So $\mu + k\sigma = 77.07$, and $\mu - k\sigma = 25.59$

3.5 Daisy Dense Descriptor

The Daisy descriptor (Tola *et al.*, 2010) is inspired from earlier ones such as the Scale Invariant Feature Transformation (SIFT) and the Gradient Location Orientation Histogram (GLOH) descriptor (Mikolajczyk and Schmid, 2005) but can be computed much faster for this purpose. Unlike SURF, which can also be computed efficiently at each pixel, it does not introduce artifacts that degrade the matching performance when used densely. For each image, first H orientation maps, G_i, $1 \leq i \leq H$, are computed, one for each quantized direction, where $G_o(x,y)$ equals the image gradient norm at location (x,y) for direction o if it is bigger than zero, else it is equal to zero. Each orientation map is then convolved several times with Gaussian kernels of different $\sum$ values to obtain convolved orientation maps for different sized regions. Daisy uses a Gaussian kernel, whereas SIFT and GLOH use a triangular shaped kernel. (Winder *et al.*, 2009) improve the parameters for choosing the best Daisy descriptor application.

Originally, Daisy features are calculated as dense features on the entire image. We instead divide the image into a grid of a specific size and calculate daisy features on this grid, like our TSURF approach. We call this approach as TDaisy or Terrain-Daisy. It can also be called as GDaisy or Grid-Daisy denoting the calculation of daisy features across a grid. We then perform classification on these local features. Each local feature is a

200-dimensional vector. Daisy has performed well in object recognition scenarios (Zhu *et al.*, 2011). Here we test whether it performs equally well for terrain classification.

3.6 Contrast Context Histograms

The Contrast Context Histogram (CCH) (Huang *et al.*, 2006) and (Huang *et al.*, 2008) is a new invariant local descriptor for image matching and object recognition. The motivation was to develop a computationally fast descriptor, which uses fewer histogram bins and has a good matching performance. This approach considers a histogram-based representation of the contrast values in the local region around the salient corners. First, corners are extracted from a multi-scale Laplacian pyramid by detecting the Harris corners at each level of the pyramid. For each salient corner p_c, in the center of a $n \times n$ local region R, the center-based contrast $C(p)$ of a point p in R is calculated by the formula:

$$C(p) = I(p) - I(p_c), \tag{3.3}$$

where $I(p)$ and $I(p_c)$ are the intensity values of p and p_c, respectively. A log-polar coordinate system (r, θ) is used to divide the local region R into several non-overlapping regions $R_1, R_2, ..., R_t$. This makes it more sensitive to the points closer to the center. The direction of $\theta = 0$ in the log-polar system is set to coincide with the edge orientation of p_c, to ensure rotation invariance. The sub-regions are then represented by histograms. Since summation of positive and negative contrast values can damage the discriminative ability of the bin, separate histograms are calculated for positive and negative values. Hence, for each point p in region R_i, the positive histogram bin with respect to p_c is given as:

$$H_{R_i}+(p_c) = \frac{\sum\{C(p)|p \in R_i, C(p) \geq 0\}}{\#_{R_i+}}, \tag{3.4}$$

where $\#_{R_i+}$ is the number of positive contrast values in the i-th region R_i. The negative histogram bin is calculate as:

$$H_{R_i}-(p_c) = \frac{\sum\{C(p)|p \in R_i, C(p) < 0\}}{\#_{R_i-}}, \tag{3.5}$$

where $\#_{R_i-}$ is the number of negative contrast values in the i-th region R_i.

Finally, histograms of all subregions are combined to form the CCH descriptor of p_c for the local region R as:

$$CCH(p_c) = (H_{R_1+}, H_{R_1-}, H_{R_2+}, H_{R_2-}, \ldots, H_{R_t+}, H_{R_t-}) \tag{3.6}$$

The resulting local descriptor is a 64-dimensional vector.

Chapter 4

Grid Based Approach

The first question that arises when classifying terrains is that of segmentation. Since we rarely see only one terrain in an image, it does not make sense to classify whole images. Most of the images from the robot's camera when it is driving over terrains, contain multiple terrain types.

One approach which comes to mind for segmentation is based on pixel based classification and then segmentation of the resulting map. There are multiple problems with this approach. The first problem is that pixel based processing will take a long time. One pixel does not have information to be able to be classified. Neighborhood information is required for this purpose, which in turn slows the processing even more. Secondly, if we do pixel based classification, the resulting map is largely perforated and then segmentation becomes a tedious task.

The approach we have used for our solution is to divide each image into cells and perform classification on those cells. For this purpose, we draw a grid on each image which needs to be classified. Then the cells created by this grid are used for classification. The size of the grid can be adjusted freely to increase or decrease the cell size. Bigger cells will have more information and smaller cells will have less information. Cell size can be changed according to the intended application. Applications that require just an overview of the terrain can use larger cells, since this is faster and is easier to do segmentation. On the other hand, applications that require detailed terrain view can use smaller cells, which gives much more terrain cells and thus leads to more accurate output.

Figure 4.1 shows a sample image from the robot camera with grids of different sizes overlaid on it. The first image has a grid with cell sizes of 100×100, the second image has 50×50 and the third image has 10×10. Other resolutions are also possible depending upon the application intended. A larger cell size will yield a small number of cells and correspondingly fewer descriptors. A smaller cell size, on the other hand, yields more cells and more descriptors, although each cell has less information about the pattern.

4.1 Grid Based Texture Descriptors

We test the three texture based descriptors LBP, LTP and LATP described in section 3. All of these descriptors are histogram based, i.e. they calculate a histogram of all the

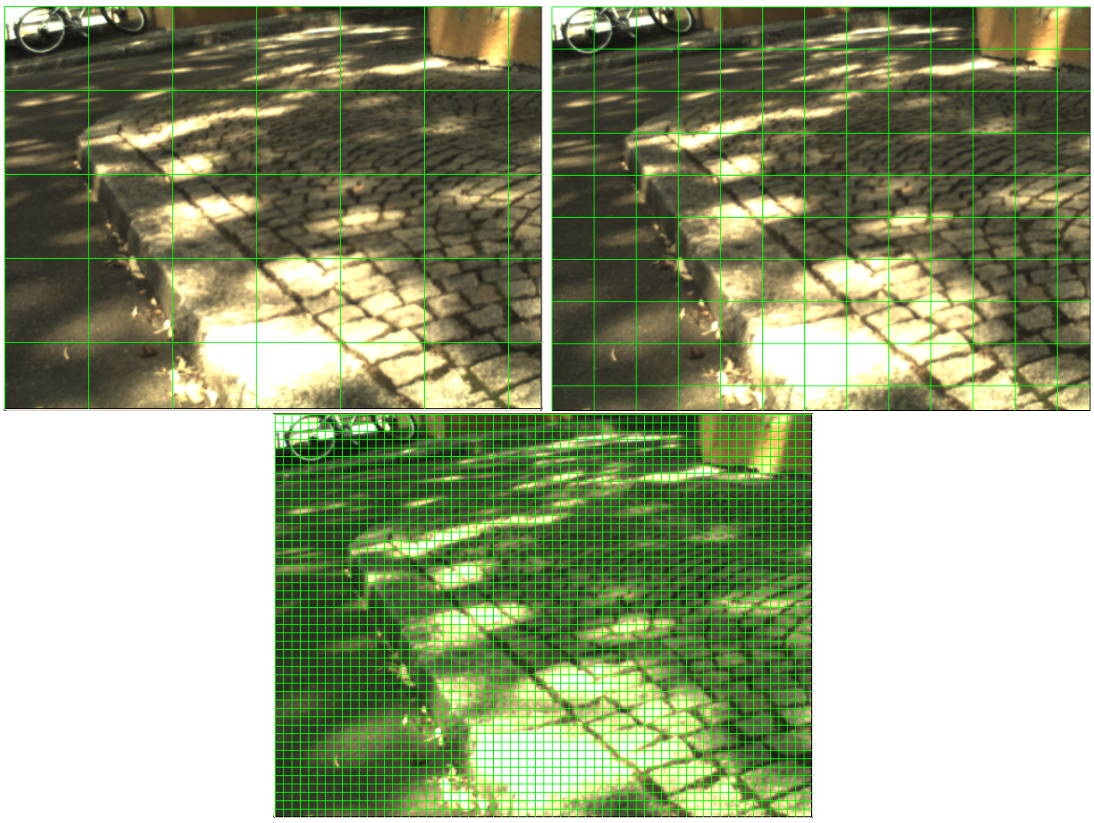


Figure 4.1: An image with a grid of size 100×100, 50×50 and 10×10 for texture based descriptors

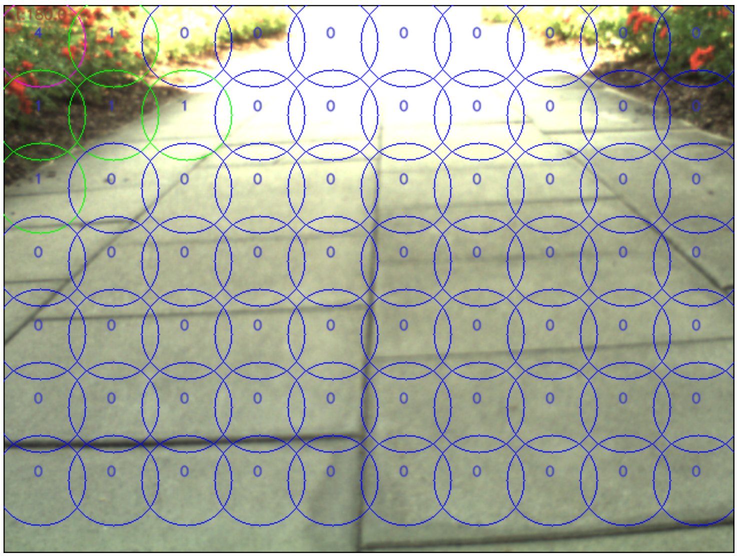


Figure 4.2: An image divided into a grid for key-point based descriptors

pixels in the region after applying some threshold. This fits well with our grid based approach. Each of these descriptors is calculated on a grid cell from the grid drawn on the whole image. The pixels in the cell are used to calculate a corresponding histogram after thresholding.

Since each pixel can have a maximum of 256 values in a greyscale image, the histogram has 256 dimensions in LBP. LTP and LATP use two different thresholded patterns to calculate the histograms, which are then concatenated. Thus their descriptors have 512 dimensions. The number of dimensions stays the same no matter what grid size is chosen. A grid with small cell size yields more descriptors than a grid with large cell size. But the descriptor size is fixed in every case. For example, if an image of size 640×480 is divided into a grid of cell sizes 10×10, then there are a total of $64\times48=3072$ cells which give 3072 descriptor vectors. Similarly an image divided into cells of size 80×80 gives $8\times6=48$ descriptors.

It is to be noted that a bigger cell size will give fewer cells but each cell will have more information. On the other hand, a smaller cell size will give more cells but each cell will have less information about the texture therein. For example, a grid of cell sizes 80×80 will yield cells each of which has 6400 pixels, whereas a grid of cell sizes 40×40 will give cells each of which has only 1600 pixels.

4.2 Grid Based Keypoint Descriptors

Key-point based descriptors operate differently than the texture based descriptors. Key-point based descriptors focus on a single pixel and observe the pixels around this pixel to calculate the descriptor. SURF and Daisy fall in this category. When used with a grid, this kind of descriptors can be calculated on grid intersections or on the center of a cell. When calculated on grid intersections, the number of cells formed is less than those in the case of texture descriptors. This is because there is one row and one column less of intersections than cells. Hence, an image of size 640×480 divided into a grid of cell sizes 10×10 gives $63\times47=2961$ cells and the corresponding descriptor vectors. Whereas in the case of texture descriptors, the same scenario produced 3072 descriptor vectors. This is not a major difference and a large number of images in the database can make this difference negligible.

On the other hand, if the descriptors are calculated on the center pixel of each cell, we get the same number of descriptor vectors from the image as in the case of texture descriptors. For an image of 640×480, we shall get the same 3072 descriptor vectors. Figure 4.2 shows a 640×480 image divided in a grid and then used for key-point based descriptor calculation. A grid with bigger cell size gives fewer cells or keypoints. These keypoints are farther away from each other. Correspondingly, a grid with smaller cell size yields more keypoints and these keypoints are located close to each other. When the keypoints are located near each other, then larger scales can have more overlaps between the regions considered around the keypoints, whereas, when the keypoints are far away from each other, then fewer overlaps occur. Since different scales are tested on different grid sizes, the overlapping does not matter a lot. If it deteriorates the results, it will be filtered out and better scale-cell size combinations will be chosen as the best ones.

Chapter 5

Experimental Setup

Here we describe the setup in which we conducted our experiments. The experiments were performed on two different types of robots, however, there were multiple robots of each type. Different robots were used for collecting different datasets, depending on which robot was available at that time and the sensors present on that specific robot. The environment used for our experiments was an outdoor environment in our university campus. There are many different types of terrain at our campus over which both types of robots were tested.

5.1 Experimental Platforms

One of the first things to consider was which platform to use for our experiments. There are many robotic platforms available at our research group, each of which has its own advantages and disadvantages. Most of them are fitted with laser range finders, which are effective in distance estimation and related applications (Roth and Wibowo, 1996; Sadhukhan and Moore, 2003). But we use robotic vision for our experiments and hence the following robots.

5.1.1 Wheeled Robot

Two different robots were used for our experiments. One is a wheeled robot and the other one is a quadrotor flying robot. This allows us to tackle the problem of terrain classification from multiple perspectives, i.e. close to ground and high above the ground. Both of these robots have been developed at our research group.

Our outdoor wheeled robot (Fig 5.1) (Bohlmann *et al.*, 2012a,b) is a modified RC-model truck whose body was removed and replaced with a dual-core PC, a 32-bit micro-controller and different sensors attached to the vehicle, including a Point-Grey Firefly color camera with a 6 mm lens to capture images at a resolution of 640×480 pixels. It is one of eight such robots developed and built at our department (Bohlmann *et al.*, 2009). These robots are used for experiments requiring multi-robot systems. They are also used for practical robotics courses of students at the university. They have evolved into a mature system which has won a European competition (the SICK robot day held

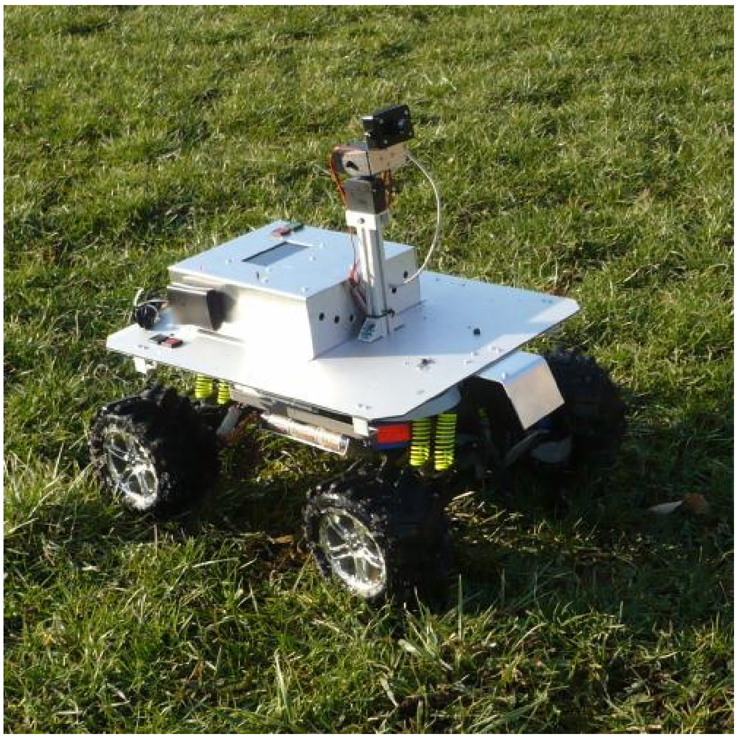


Figure 5.1: Wheeled outdoor robot used for experiments

in 2010) (Scherer *et al.*, 2011). Each robot can be fitted with two different sets of tires. One set consists of more smooth road-type tires. These tires are used for driving indoors or on smooth roads. They don't create a lot of slippage and thus can drive at higher speed because of more accurate navigation. The other set consists of more rugged tractor tires. These tires have protruding patterns on them which help them to grip the ground surface while traversing non-smooth ground surfaces. We used tractor tires for our experimentation, since we needed to traverse harsh surfaces like gravel and grass.

For our experiments, we ran the robot at about 1 m/s speed while capturing images from the camera, hence not all of the acquired images are sharp due to motion blur artifacts. The height of the mounted camera is approximately 50 cm from the ground. The robot is equipped with tractor tires to be able to run on very rough terrain. However, these tires produce an increased amount of vibration while traversing even a smooth surface. The camera is tilted down so as to capture the terrain directly in front of the robot. Hence, the camera captures images starting from a distance of 30 cm with respect to the robot's front.

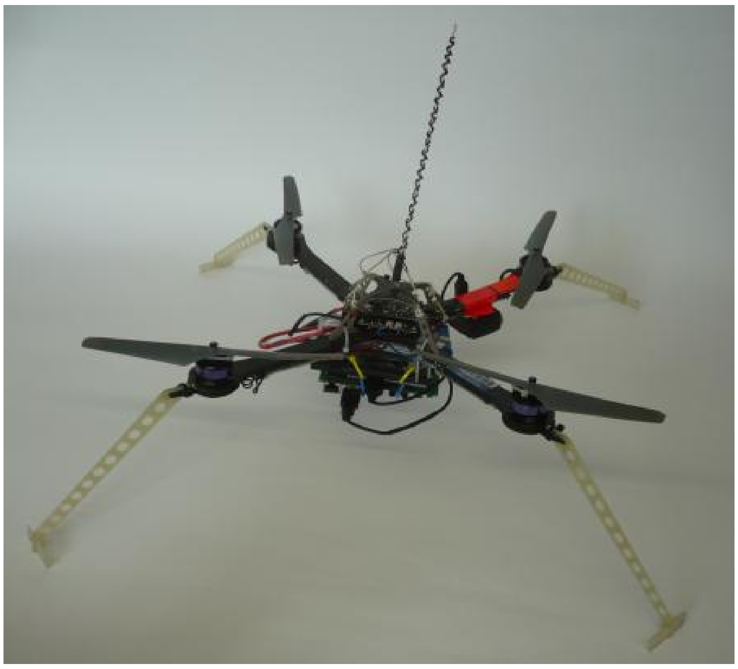


Figure 5.2: Flying robot used for experiments

5.1.2 Flying Robot

We also used another robot, which is a flying robot flown across the university campus with a camera. We used an *AscTec X3D-BL Hummingbird* quadrocopter (Fig. 5.2) which has a diameter of 53 cm and weighs 0.5 kg (Wenzel *et al.*, 2010b). It has been equipped with a *PointGrey FireFly USB* color camera with VGA resolution and a *Gumstix Overo Fire* single-board computer with a 600 MHz *ARM* processor. Pictures were taken during remotely controlled flight with a frame rate of 1 Hz and stored on a MicroSD card for later processing. (Masselli and Zell, 2012)

By varying the speed of the four motors the aircraft can tilt, roll, yaw and change its altitude. The quadrotor and its peripheral devices are powered by a 2000 mAh lithium-polymer battery and allow a flight time of up to 15 minutes, depending on the flight maneuvers. The *X3D-BL Hummingbird* platform comes with a circuit board including two 60 Mhz 32 bit ARM micro controllers, a three-axis gyroscope, an accelerometer, a compass module, a GPS sensor and pressure sensor (Wenzel *et al.*, 2010c). This and other types of quadrotors are being used in our lab (Yang *et al.*, 2012).

Earlier we tried different hardware configurations for image capture aboard a flying robot. The first attempt was to use the AR.Drone made by the French company Parrot. It is a radio controlled flying quadrocopter made out of plastic and foam. It can be controlled by an Android or an iOS device using Wi-Fi. It is an easy to control hobby electronics equipment. It has an ARM9 468 MHz microcontroller with 128 MB RAM

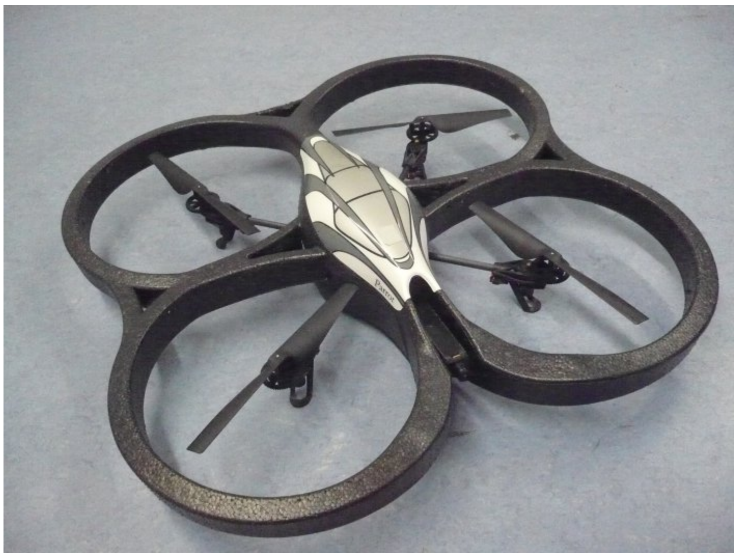


Figure 5.3: AR.Drone robot used for initial experiments

and Linux as the operating system. Sensors include a 3-axis accelerometer, a 2-axis gyro and a 1-axis precision gyrometer. Also included is an ultrasonic altitude meter of 6 m range, which keeps it vertically stable. The Parrot AR.Drone is finding increasing use in research. Fig 5.3 shows such a quadrocopter. At our research group, the AR.Drone is also used as a research platform (Wenzel *et al.*, 2012).

It has two cameras mounted on it: one is a front facing and the second is downward facing camera. The front camera captures images at VGA resolution (640×480) with a wide-angle lens (93°), whereas the downward camera captures only at QVGA resolution (320×240) with 64° lens. Since the quadrocopter flies at some height from the ground, only the down-facing camera can be used to take images of the terrain below. This down-facing camera has a low resolution and so it cannot capture a lot of information of the ground below. This problem gets worse with increasing height. So we decided not to use this robot for our experiments.

The second attempt was to use the AscTec Hummingbird, but with a cell phone to capture the images. The cell phone was carried in a custom built gondola mounted on the Hummingbird. It was mounted such that the cell phone was facing downward with its camera, to be able to capture images of the ground below. The cell phone used was Nokia's N95 phone. It has the Symbian operating system version 9.2 with a S60 3rd Edition user interface. It has an ARM11 based 332 MHz processor with 160 MB RAM. There is a 5 megapixel digital camera with Carl Zeiss optics. It can be used to capture high resolution images and videos. It has GPS to determine its coordinates and WiFi and bluetooth to transmit data. A SIM card from a mobile carrier with data option can be used to connect to the Internet and transmit data over the Internet. The cell phone weighs about 120 g and is within the payload capacity of the quadrocopter. This combination

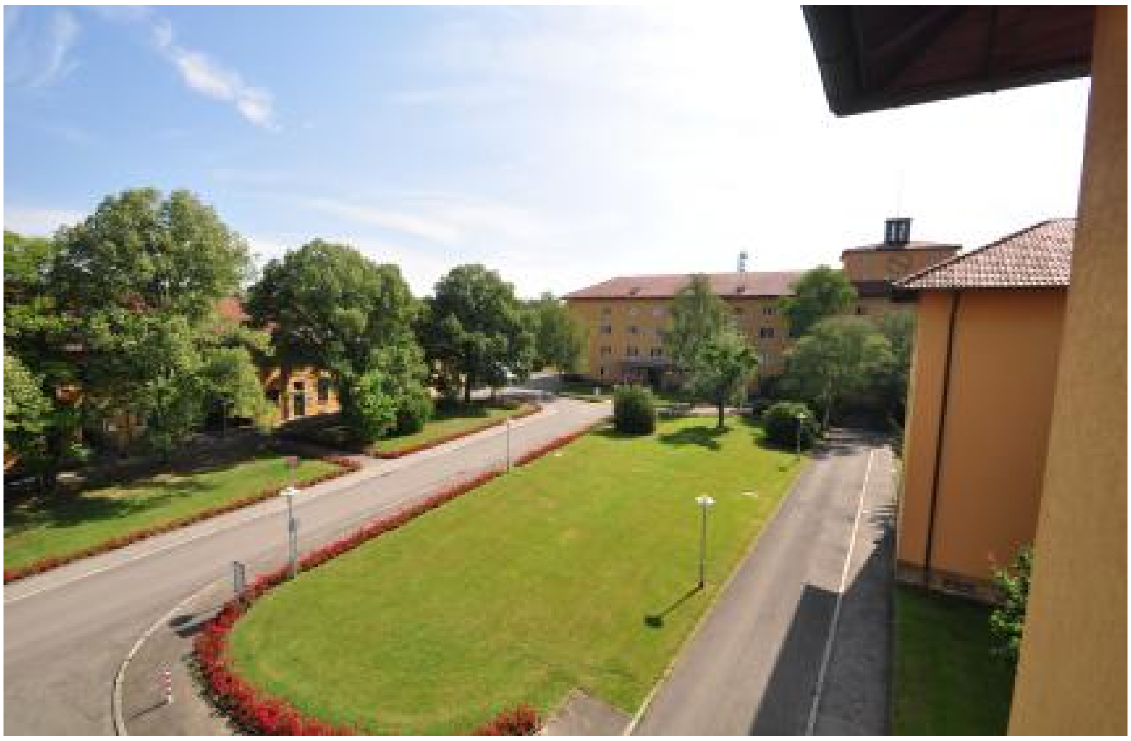


Figure 5.4: Campus area for experiments

of the cell phone with the quadrocopter was used in some experiments at our research group as in (Erhard *et al.*, 2009a).

A python script was used to capture images. Many experiments were done with the cell phone flying with the quadrocopter. But the images captured by the cell phone were almost always very blurry. Image capturing also consumed a lot of time. Another problem was that the cell phone was not connected to the microcontroller of the quadrocopter and it was not possible to control the cell phone through commands given to the quadrocopter. The software on the cell phone was not completely controllable, and required deep knowledge of the Symbian system, which was too time consuming. In this time, we were able to connect a Gumstix to the quadrocopter which is lighter and offers much more control. So the quadrocopter fitted with a Gumstix was used for further experimentation.

5.2 Experimental Environment

For experiments with the wheeled robot, we drove the wheeled robot outdoors in our campus (Fig 5.4) and observed the terrain types visible to the robot through the camera. The outdoor area of the “Sand 1-14“ campus consists of roads, meadows and some parking areas covered with gravel or tiles and is a fairly large area, which is needed to test long range sensing (Erkan *et al.*, 2007). We were able to identify five different classes: asphalt, gravel, grass, big-tiles and small-tiles. We navigated the robot multiple times over different routes at varying times of the day. One of these experiments was carried out when the sun was about to set which resulted in a direct irradiation of the camera. In this case, the image colors were extremely distorted.

The second experimental setup was a heavily clouded sky after rainfall. Some of the terrain types contained wet and dry patches, e.g. asphalt, gravel, etc. In this case, a single terrain type contained different colors. The third scenario was at noon on a sunny day. Note that not all terrain types were captured in each scenario.

While driving on the campus we found that all terrain types contained many different features depending on the location and time at which the pictures were acquired. Fig. 5.5 shows different terrain types indicating the artifacts introduced under different scenarios. For example, Fig. 5.5(a) shows a blurred image of the grass terrain type along with small plants and their flowers. Fig. 5.5(b) shows the asphalt terrain type with a wet patch after rain. In Fig. 5.5(c) the gravel terrain type is depicted after rainfall. Here, water was gathered in a bigger amount. Fig. 5.5(d) shows a sample image from the big-tiles terrain type. Since parts of the terrain are shadowed, its intensity changes a lot and the boundary also becomes difficult to classify. Similarly, Fig. 5.5(e) shows an image from the small-tiles terrain type. It is also noticeable that the shadow of a tree induces texture artifacts of its own.

Fig. 5.6 shows the grass terrain type under two different weather conditions. The image on the left is taken in winter (middle of March) one hour before sunset. The sun was looking into the camera at that time. The image on the right is taken in spring (middle of June) on a cloudy afternoon. Moreover, the image on the left also has a patch of snow among the grass. Both of these scenarios were included in the dataset. This clearly shows that color based descriptors will not work in all cases. Also note that under similar conditions, color based descriptors can misjudge the wet and dry or shaded and open parts of the same terrain type. Other than that, color will only accurately distinguish grass from other terrain types as is obvious from the sample terrain images. Finally, Fig 5.7(a) shows the small-tiles terrain type with blur induced due to robot motion and Fig. 5.7(b) shows the same terrain type with over-exposure due to sun.

All images are characterized by the presence of not only one but multiple terrain types. These images were labeled manually to generate training images for each class. Almost all of the images contained diagonal or irregular boundaries between two terrain types. Hence, even after clipping, most of the images contained other terrain types at the borders. Note, that this interferes with the terrain descriptors which are based on a rectangular grid and hence results in a decrease in classification accuracy. Images containing blur were not filtered out, except in extreme cases where the blur artifacts were too dominant.

For our experiments with the flying robot, we flew the robot outdoors at our university campus in the “Sand“ area at Tübingen and observed the terrain types below the robot through the camera. The outdoor area of the campus consists of roads, meadows, bushes

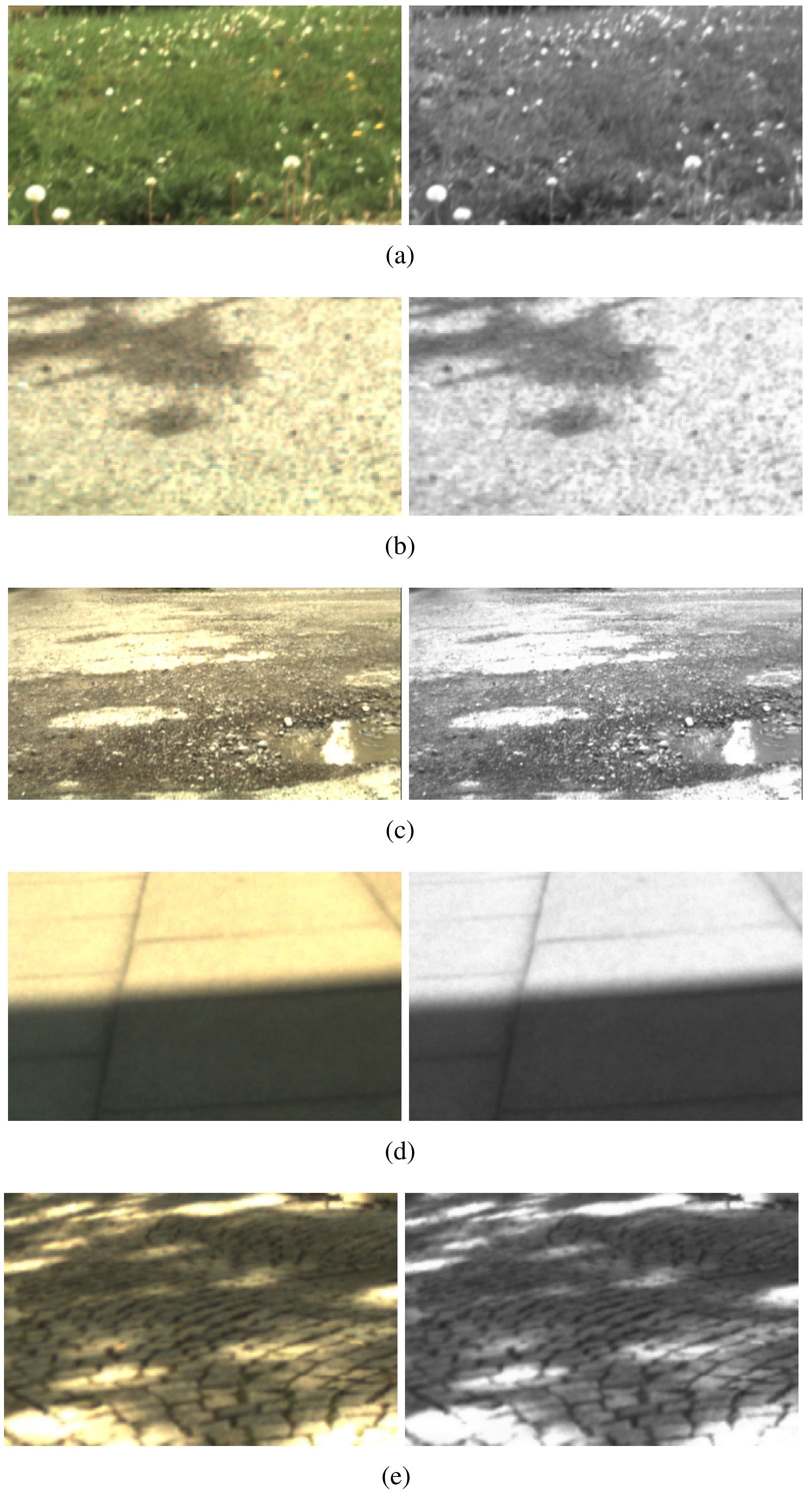


Figure 5.5: Sample images of different terrain types: (a) grass, (b) asphalt, (c) gravel, (d) big-tiles and (e) small-tiles, both in color and in greyscale

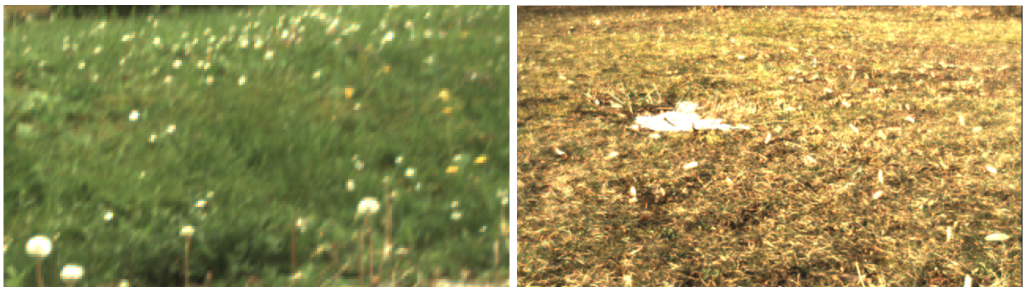


Figure 5.6: Difference of grass color under different sun angle and weather conditions

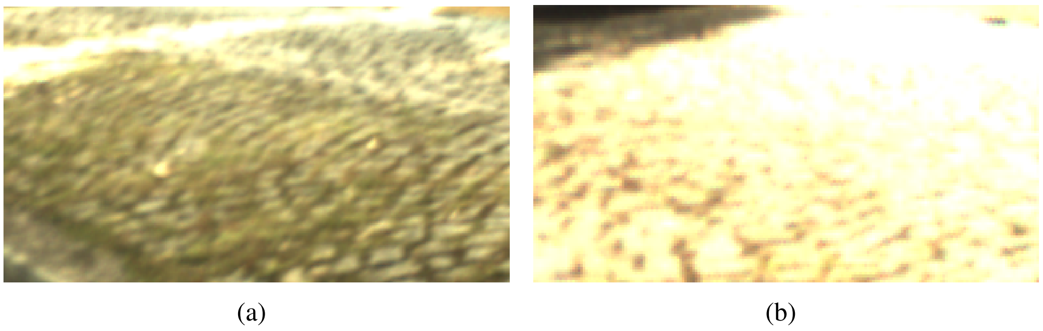


Figure 5.7: Samples from small-tiles terrain type under: (a) blur, (b) over-exposure

and some parking areas covered with gravel or tiles as shown in Fig 5.4. We were able to identify six different classes: asphalt, gravel, grass, bushes, big-tiles and small-tiles.

While flying in the campus we found that all terrain types contained many different features depending on the location and time at which the pictures were acquired. Fig. 5.8 shows different terrain types to indicate the artifacts introduced under different scenarios. For example, Fig. 5.8(a) shows an image of the grass terrain type along with small plants and their flowers. Fig. 5.8(b) shows the asphalt terrain type with a line formed from a recent construction. In Fig. 5.8(c) the gravel terrain type is depicted which contains a lot of soil as well. Fig. 5.8(d) shows a sample image from the big-tiles terrain type on the left side along with some other terrains on the right. Similarly, Fig. 5.8(e) shows an image from the small-tiles terrain type. Finally Fig. 5.8(f) shows one type of bush that we observed.

Fig. 5.9 shows the grass terrain type at two different heights of the quadrocopter. The image on the left is taken at a height of about 1 m, whereas the image on the right is taken at several meters height. Both of these scenarios were included in the dataset for all terrain types. Also note that under similar conditions, color based descriptors can misjudge the shaded and open parts of the same terrain type. Color will only accurately distinguish grass and bushes from other terrain types as is obvious from the sample terrain images, but not within similarly colored terrain types.

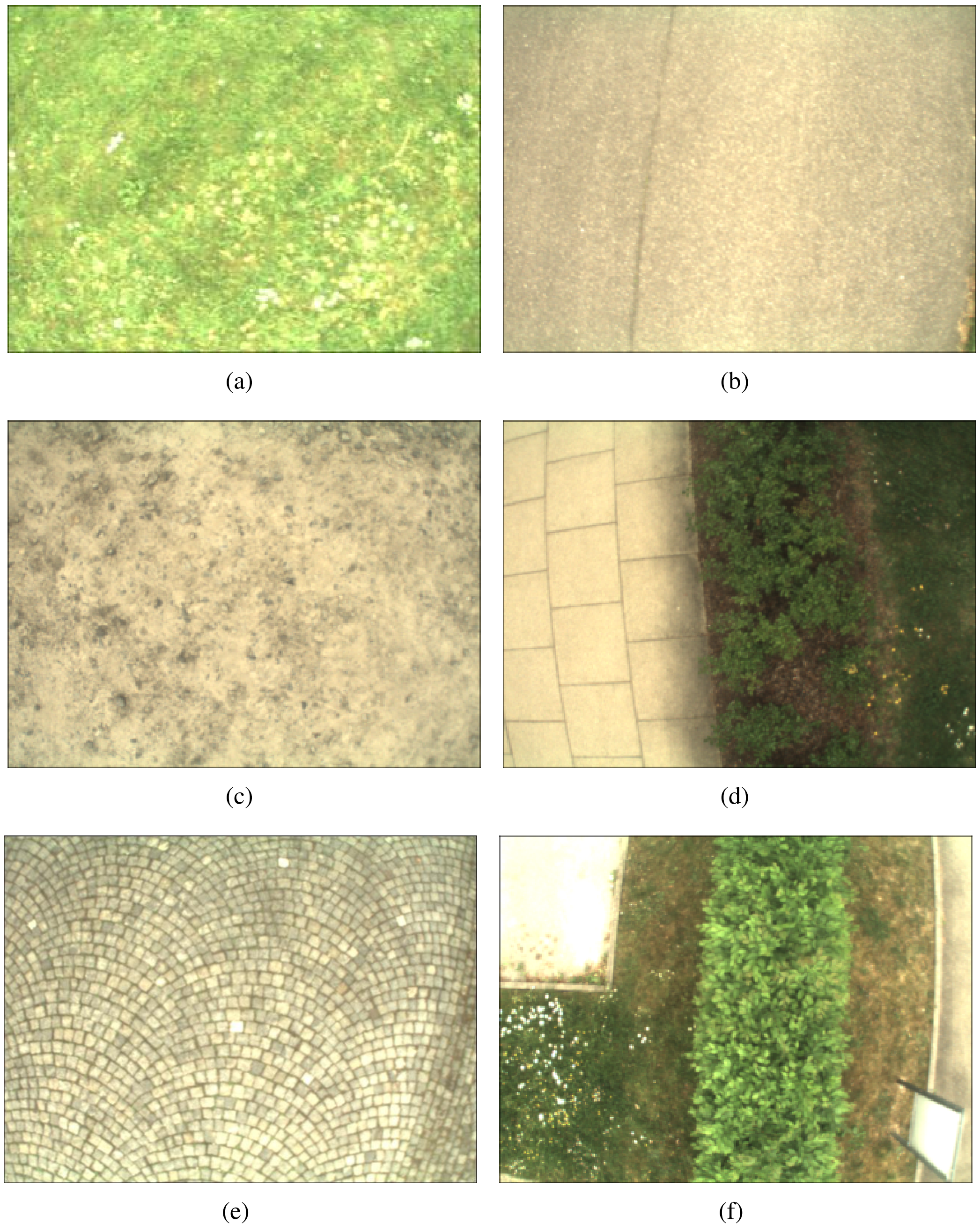


Figure 5.8: Images of various terrain types: (a) grass, (b) asphalt, (c) gravel, (d) big-tiles (left), (e) small-tiles and (f) bushes

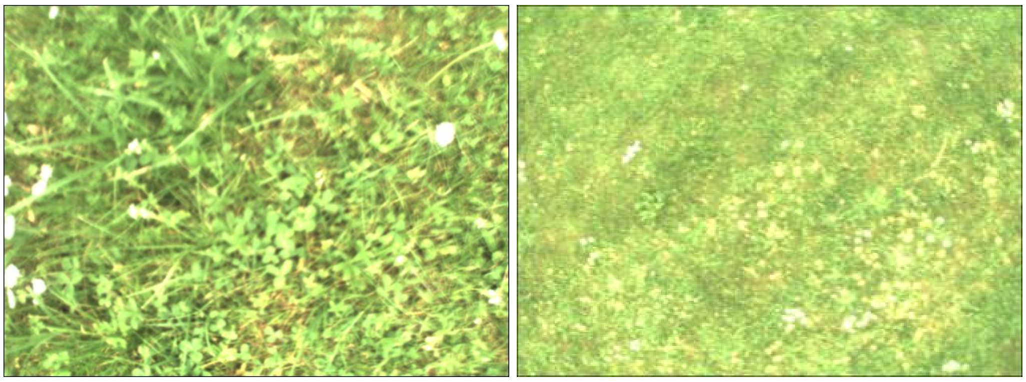


Figure 5.9: Difference of scale for grass

Chapter 6

Terrain Classification based on Greyscale Images

In this chapter we present our first set of techniques. These techniques are applied to terrain classification on greyscale images. Greyscale images have the advantage over colored images that they do not have color artifacts which can really distort the images. Secondly, they have less information than colored images which makes it faster to process them.

6.1 Related Work

Several authors have addressed the problem of representing texture information in terms of co-occurrence matrices (Haralick *et al.*, 1973), Markov modeling (Manjunath and Chellappa, 1991; Vernaza *et al.*, 2008), Local Binary Patterns (LBP) (Ojala *et al.*, 2002), and texton-based approaches (Varma and Zisserman, 2005; Angelova *et al.*, 2007b) to name a few. Some more work has been done using global or local features (Artač and Leonardis, 2004; Artač *et al.*, 2005). Yet, it remains unclear which approach is suited best for an online application on a real outdoor robot both related to prediction accuracy and training time performance. Hence, the main motivation of our experiments is a thorough comparison of different texture descriptors for representing different terrain types. Here, the data originate from a real robot traversal whose camera images contain artifacts such as noise and motion blur. These data differ from the ones included in the Brodatz data set (Brodatz, 1966) or Calibrated Color Image Database (Olmos and Kingdom, 2004). There, the images have been acquired under controlled conditions lacking dark shadows and overexposure. Note that latter sources of noise are often present in images taken outdoors. Furthermore, terrain class prediction should be performed on-board the mobile robot. These kinds of robots are expected to navigate freely in off-road driving (Stavens *et al.*, 2007) including driving through uneven (Braun *et al.*, 2008) or forest terrain (Föhst *et al.*, 2010).

Our work focuses on using SURF descriptors for terrain classification. This descriptor is not used for texture based classification as it is an interest point based descriptor and is better suited for visual tracking like some other similar descriptors. We modify the

SURF descriptor to be suitable for texture classification. As a second contribution we introduce five further texture descriptors, the Local Ternary Patterns descriptor (LTP) (Tan and Triggs, 2007), the Local Adaptive Ternary Patterns descriptor (LATP) (Akhloufi and Bendada, 2010), the SURF descriptor (Bay *et al.*, 2006), the Daisy descriptor (Tola *et al.*, 2010) and the Contrast Context Histograms (CCH) descriptor (Huang *et al.*, 2008), which, to our knowledge, have not been applied to the domain of terrain identification before. We use a histogram based approach for many of these local image descriptors (Halawani and Burkhardt, 2005; Siggelkow and Burkhardt, 2002).

We also test our techniques for visual terrain classification from a flying robot. Aerial images can also be used for ground sensing (Früh and Zakhor, 2001). (Hudjakov and Tamre, 2011) used an artificial neural network to classify terrain from static aerial images. Patterns of 29×29 pixel size were taken from these images and fed into a big neural network containing three hidden layers, to be classified either as houses, roads, grass or debris. This approach does not extract any features, rather processes raw patches of the image. Sofman *et al.* (2006) used camera and laser range data from a UAV to classify terrain into road, grass, tree and building areas. Laser scanners are generally much more expensive than cameras and have heavy resource consumption especially when they return reflectance data, thus requiring heavy duty UAVs. A mobile phone mounted on a quadrocopter is used in (Erhard *et al.*, 2009b) to perform visual localization using GPS data as ground truth.

(Laible *et al.*, 2012) and (Lalonde *et al.*, 2006) perform terrain classification using a 3-D lidar sensor. (Laible *et al.*, 2012) compares it with visual terrain classification techniques and others use it to generate 3D models (Hähnel *et al.*, 2003). Some others have also tried to classify ground surfaces for navigation (Castelnovi *et al.*, 2005; Dahlkamp *et al.*, 2006; Stavens and Thrun, 2006). One such application was on an outdoor robot Stanley that won the DARPA grand challenge (Thrun *et al.*, 2006b,a).

6.2 Texture and Key-point Descriptors

The texture based descriptors that we use for classification are Local Binary Patterns, Local Ternary Patterns and Local Adaptive Ternary Patterns. These descriptors are calculated across a grid drawn across the image. Each pixel of the grid cell is thresholded according to some criteria to generate a pattern. A histogram of these thresholded pixels of a grid cell is computed to give the final feature vector. The feature vector size in this category varies from 256 to 512 values.

The key-point descriptors used in this work focus on the neighborhood around a specific pixel to compute the descriptor. Instead of key-point detection, chosen pixels across a grid on the image are given as key-points. The number of such pixels can vary according to the grid size chosen. The algorithms then compute the respective descriptors on these pixels and store them together. There is no histogram needed in this case and the feature vectors computed in this category have a length between 64 and 200 values.

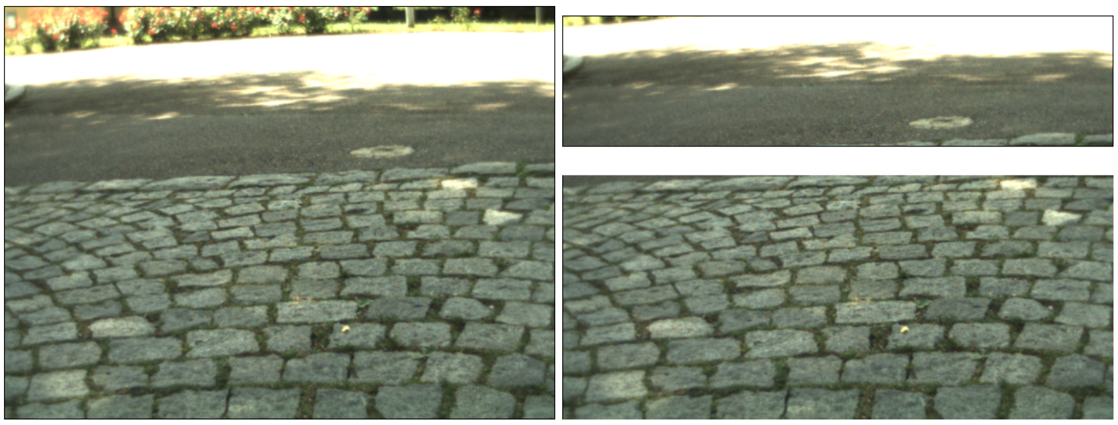


Figure 6.1: An image containing multiple terrains, which is divided into two patches: asphalt and small-tiles

6.3 Ground-truth Method 1

The first method adopted for ground-truth generation is a simple process in which the sample images are divided into smaller sections according to terrain types. This amounts to fixed segmentation of the images. Sections of an image are cut according to their terrain type and placed in a folder. A folder is created for each terrain type under each dataset. So the asphalt folder will include all image sections containing asphalt terrain and so on. This way all datasets have folders containing samples from all of the terrain types. The program goes through these datasets and calculates features on terrain samples of each dataset. This is a relatively simple and fast method of ground-truth generation.

Fig. 6.1 shows an image frame from the camera containing multiple terrain types. This image is divided into sections containing individual terrains. Each divided section has a different size and hence can contain less or more information about the terrain.

It is important to note that most of the times the boundaries between different terrain types are not very crisp. So when a terrain region is divided by straight lines, it still contains minute portions of other terrains. Secondly, terrain boundaries are mostly not aligned along horizontal or vertical axes, whereas, the image is divided into rectangular sections. So each terrain section has portions from other terrains along image edges.

6.4 Ground-truth Method 2

The second method for ground-truth generation involves labeling the whole images with colors. A color coding is chosen such that each terrain type is assigned an RGB color which represents this terrain type. Then all pixels containing that terrain type in each image are painted with that color. All of the images are thus color coded and placed in

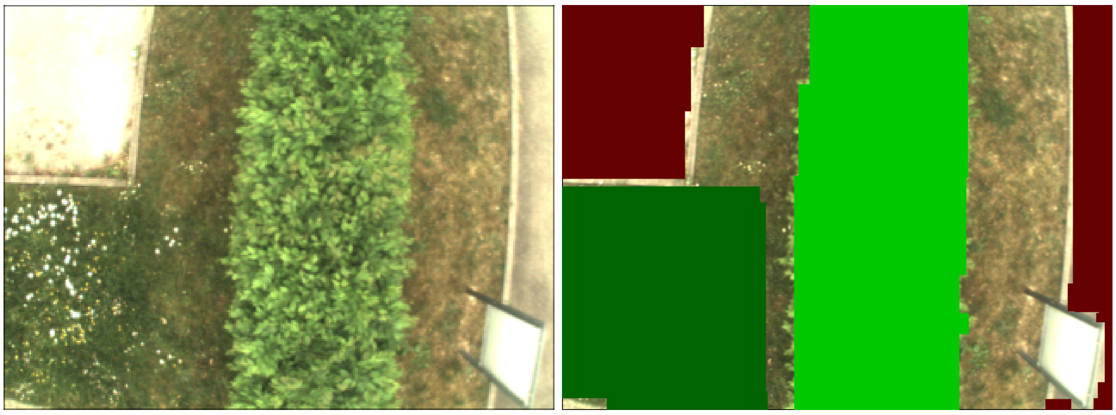


Figure 6.2: An image containing multiple terrain types and the corresponding ground-truth image

a separate folder. All of the areas which do not represent any terrain are neither painted nor removed. So all of the labeled images retain their size of 640 × 480. Fig 6.2 shows an image with multiple terrain types and the corresponding ground-truth image.

The algorithm for ground-truth generation goes through all of the camera images and the corresponding ground-truth images. It calculates descriptors on a grid drawn on the image. For each cell of the grid, it calculates an image descriptor and looks at the corresponding cell in the ground-truth image. This grid cell in the ground-truth image is checked for the most frequent color code representing any terrain type. This most frequent color is checked to see if it covers at least 40 % of the cell pixels. If this holds true then the image descriptor is assigned to the terrain type representing that color code.

For example, Fig 6.3 shows an image with a grid of cell size 80 × 80 laid on it. Cell (6,4) has about 80 % grass so it is marked as a grass cell. Whereas, cell (6,3) has about 60 % asphalt and about 30 % grass. So the most frequent terrain in this cell is asphalt, which occurs more than 40 % and hence this cell is marked as an asphalt patch. On the other hand, cell (7,4) has no dominant terrain type, since all of the terrains occur less than 40 % of the image, so this cell is ignored and its descriptor is not stored. Fig 6.4 displays the ground-truth generated for this image.

This method guarantees that every part of a terrain will be used in the experiment and no part is left due to clipping of the image. Moreover, this method insures that samples of each terrain type have only that terrain type and do not contain portions of other terrain types. Although this method is also tedious to perform by hand, it uses less effort than method 1, since we do not create multiple sub-images from an image and store them in different folders.

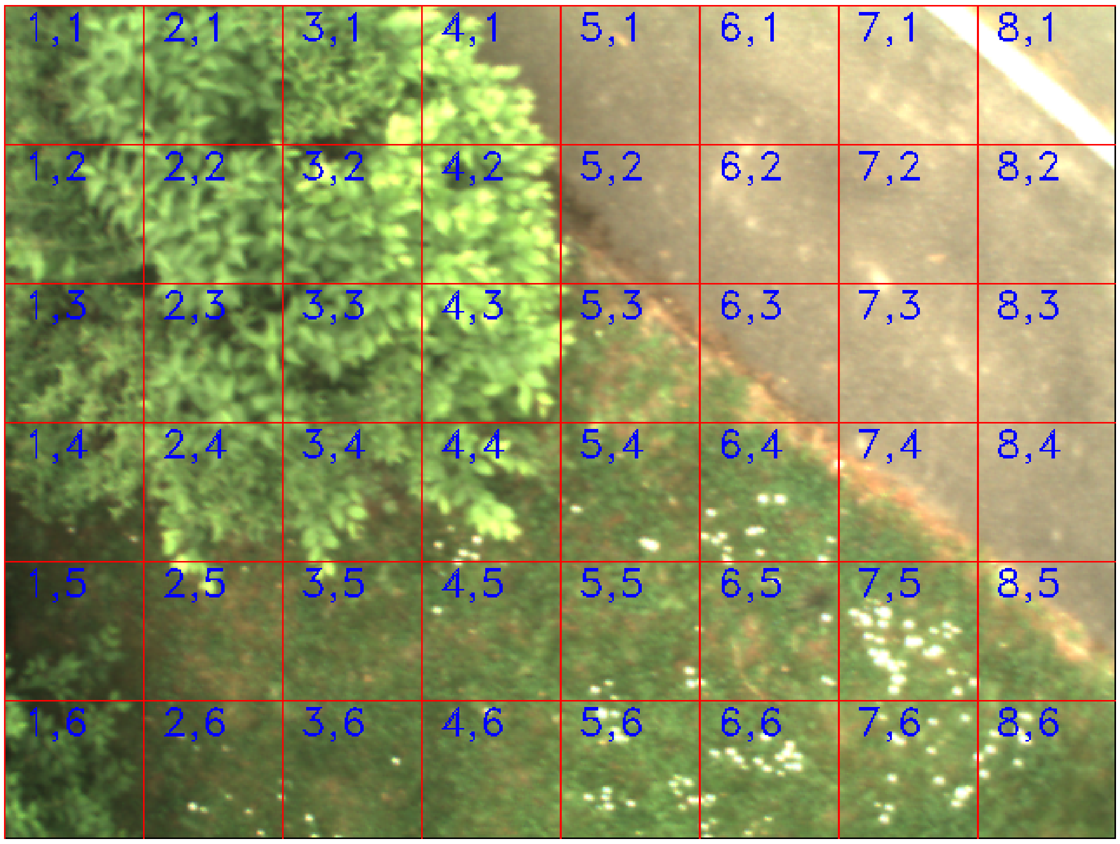


Figure 6.3: An image containing multiple terrains with a grid cell size 80×80 laid on it

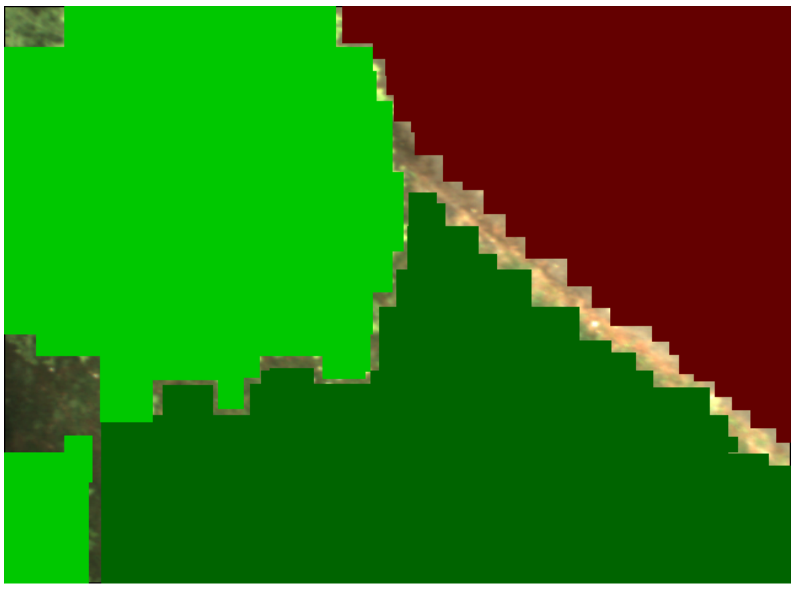


Figure 6.4: Ground-truth of the image in Fig. 6.3

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	55.0%	67.2%	54.2%	**99.2%**	79.2%	40.4%
20	75.7%	83.0%	73.7%	**97.8%**	74.9%	39.5%
30	85.2%	90.0%	83.3%	**96.6%**	72.4%	41.4%
40	90.2%	93.3%	88.4%	**95.7%**	70.8%	45.0%
50	92.6%	94.7%	91.4%	**94.9%**	70.0%	44.0%
60	94.4%	**95.7%**	93.4%	95.2%	69.0%	48.0%
70	95.5%	**96.8%**	95.0%	94.5%	69.0%	44.9%
80	95.8%	**96.8%**	95.8%	93.1%	67.8%	47.6%
90	96.9%	**97.4%**	96.5%	93.6%	68.5%	47.4%
100	96.9%	**98.1%**	97.2%	94.0%	69.1%	49.1%

Table 6.1: Classification results of the five descriptors

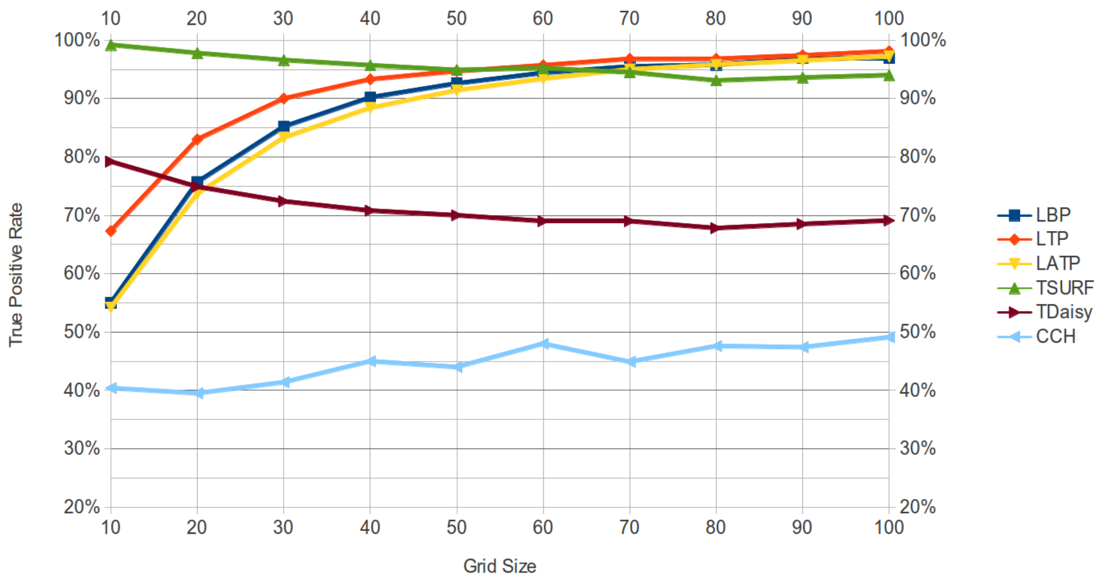


Figure 6.5: Graph of descriptor accuracies at different grid-sizes with Random Forest classifier

6.5 Experiments with a Wheeled Robot

Classifiers were applied on each descriptor and the true positive rate (TPR) of the entire dataset was obtained. The TPR is the ratio of the correctly classified instances to the number of all test patterns contained in the data set. It is also called the Recall or Sensitivity value:

$$True\ Positive\ Rate\ (Recall) = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{6.1}$$

Sensitivity (TPR) measures the proportion of actual positives which are correctly identified as such. Sensitivity relates to the test's ability to identify positive results. If a test has high sensitivity then a negative result would suggest the absence of desired sample. Since Random Forests produced the best overall result, we describe those results in detail here. Table 6.1 presents a summary of results of the five approaches on the five terrain types.

Note that high resolution means that the image is divided into more parts, meaning that each grid cell is very small and so we get a lot of grid cells. For example, a 640×480 image divided into 10×10 patches gives $64 \times 48 = 3072$ grid cells. This is the highest resolution we tested, i.e. with a grid size of 10×10. On the other hand, low resolution means that the image is divided into fewer cells and that the size of each grid cell is large. So in this case, a 640×480 image divided into 80×80 patches gives just $8 \times 6 = 48$ grid cells.

The same data is plotted in Fig. 6.5 for visualization. Here it is clear that, although at lower resolutions the texture descriptors such as Local Binary Patterns, Local Ternary Patterns and Local Adaptive Ternary Patterns perform best, at higher resolutions, TSURF and TDaisy features produce much better results. At a grid-size of 50×50, TSURF matches the performance of the best texture descriptor, as both TSURF and LTP have an accuracy of about 95%. For higher resolutions, TSURF performs best. At a grid-size of 10×10, TSURF has a performance of 99%, whereas LTP only gives a performance of 67%.

It is to be noted that for grid-sizes lower than 30×30, the performance of the three texture descriptors decreases sharply. This is due to the fact that such a small cell does not include enough neighboring information for adequate feature description. Even below a grid-size of 40×40, the performance of the three texture descriptors falls below 90% and hence they may not be usable.

The performance of the TDaisy descriptor improves with increasing resolution. Although worse than TSURF, it performs better than the three texture descriptors only at a grid-size of 10×10. A sample confusion matrix for TDaisy is depicted in table 6.6. The most confusing was the distinction between grass and gravel in both directions. The second most confusion occurred between gravel and small-tiles. These are large values of confusion and so the results are not that good for this descriptor.

The TSURF descriptor is the smallest descriptor consisting of only 64 dimensions.

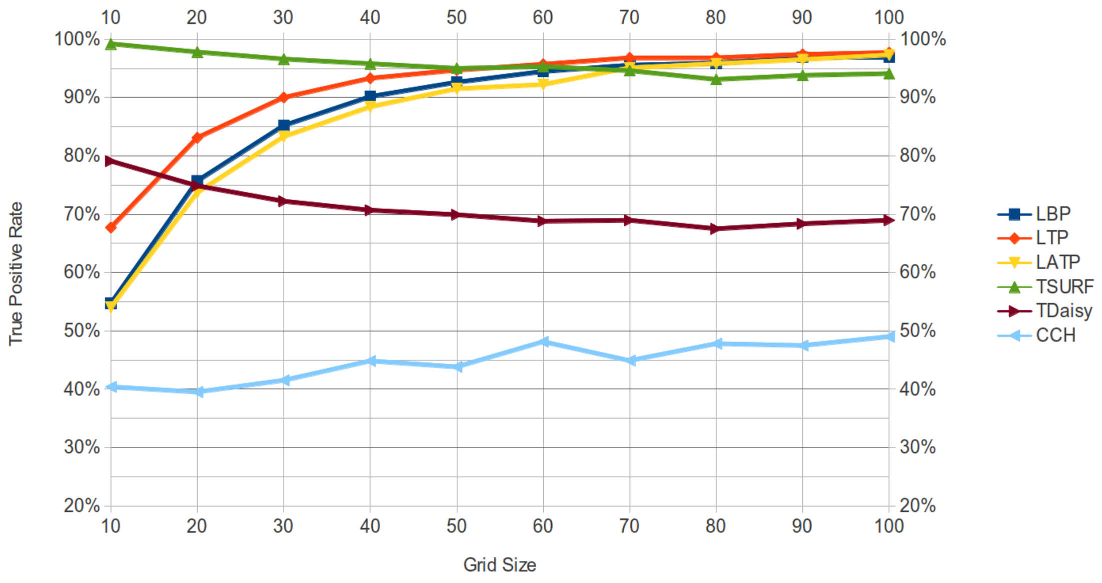


Figure 6.6: Graph of precision values at different grid-sizes with Random Forest classifier

The LTP and LATP descriptors are the longest descriptors consisting of a 512 dimensional vector each. LBP and TDaisy are intermediate length descriptors. LBP produces a 256-dimensional descriptor and the TDaisy descriptor consists of 200 dimensions. For TSURF based classification, different scale levels (σ) described in section 3.1, ranging from 2 to 20, were tried. Higher values of this scale parameter for descriptor calculation give the best result in all of the cases.

Another performance metric is the precision value of the classifier. Precision is the fraction of retrieved instances that are relevant. High precision means that an algorithm returned more relevant results than irrelevant. It is calculated as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{6.2}$$

Table 6.6 shows these values plotted on a graph for all the descriptors on various grid sizes. It can be seen that the precision values of this dataset shows the same behavior as the true positive rate values. Hence we present only true positive rates (recall) for further experiments to reduce the amount of redundant graphs in this work.

Table 6.2 shows a sample confusion matrix for TSURF. This matrix shows that the largest confusion occurs between small-tiles and gravel, although not a very large value. The terrain small-tiles further has confusion with asphalt and grass as well. There is also some confusion between asphalt and big-tiles.

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	309	2	5	8	3
asphalt	3	302	5	15	2
grass	11	0	308	0	8
big-tiles	2	5	0	321	0
small-tiles	26	15	14	5	268

Table 6.2: Confusion Matrix for TSURF

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	610	6	29	0	0
asphalt	2	637	6	0	0
grass	29	3	614	0	0
big-tiles	0	1	0	645	0
small-tiles	1	2	0	12	631

Table 6.3: Confusion Matrix for LBP

A sample confusion matrix for LBP is shown in table 6.3. The maximum confusion is between grass and gravel. Other than that there is not much confusion, apart from a little between small-tiles and big-tiles.

For LTP-based classification, we also tried different values for the threshold value k described in section 3.3 having values between 2 and 20. It was observed that small values of the threshold give better results. The best results were mostly found with the threshold value of 2, 3 or 4. Table 6.4 lays out a sample confusion matrix for LTP at a resolution of 30×30. The table shows that most of the confusion was created between grass and gravel, which although is not very significant. The second highest confusion is between small-tiles and big-tiles. It is interesting to note that most of the confusion values are zero meaning that there is no confusion in this case.

Similarly for LATP-based classification, each image was again divided into 100×100 patches. We tried different values of the threshold value k described in section 3.4 ranging from 0.1 to 1.9. The optimum threshold was found to be 0.4. In this case big-tiles

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	624	10	11	0	0
asphalt	3	634	3	4	1
grass	26	9	611	0	0
big-tiles	0	0	0	644	2
small-tiles	0	1	0	15	630

Table 6.4: Confusion Matrix for LTP

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	600	6	39	0	0
asphalt	2	635	6	0	2
grass	24	2	619	0	1
big-tiles	0	0	0	642	4
small-tiles	0	1	0	9	636

Table 6.5: Confusion Matrix for LATP

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	359	29	115	53	89
asphalt	21	535	36	42	11
grass	119	58	386	52	31
big-tiles	17	47	58	495	28
small-tiles	69	5	23	56	493

Table 6.6: Confusion Matrix for TDaisy

was the best identified and gravel the worst identified terrain class. Table 6.5 presents the confusion matrix of the LATP approach. Some confusion occurs between gravel and grass. The LATP descriptor has the same size as the LTP descriptor and hence also consists of a 512 dimensional vector.

For TDaisy based classification the descriptors calculated on a grid of 100×100 pixels gave the best results. In this case asphalt was the best identified and gravel the worst identified terrain class. Table 6.6 presents the confusion matrix of the TDaisy approach. There is a large number of gravel patterns classified as grass and vice versa. A grid size of 30×30 produced slightly better results: Total 72.0%, gravel 58.1%, asphalt 86.7%, grass 66.9%, big-tiles 77.5%, small-tiles 70.9%.

Finally, for the Contrast Context Histogram (CCH) based classification approach, the descriptors were calculated on different sizes of the grid. The confusion matrix of the CCH approach is presented in table 6.7. There is a huge amount of confusion between gravel and grass, which even exceeds the correctly classified instances of gravel itself.

	gravel	asphalt	grass	big-tiles	small-tiles
gravel	186	83	199	91	86
asphalt	69	383	92	66	35
grass	137	56	338	36	79
big-tiles	63	48	31	467	37
small-tiles	110	63	169	74	230

Table 6.7: Confusion Matrix for CCH

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	64,963	88,356	93,139	**34,714**	96,359	46,099
20	13,417	16,279	16,654	**5,196**	17,669	12,815
30	3,479	5,448	6,054	**1,576**	6,946	3,324
40	1,599	2,784	2,451	**625**	2,921	1,460
50	808	1,420	1,285	**313**	1,614	768
60	532	776	863	**178**	833	503
70	388	573	499	**119**	546	362
80	190	349	256	**81**	304	228
90	169	282	213	**43**	239	169
100	121	232	148	**31**	169	119

Table 6.8: Time taken in seconds for cross-validation by different descriptors at different grid-sizes for the random forest classifier

Grid-size	TSURF-σ	LTP-k	LATP-k
10	20	4	1.0
20	20	4	1.1
30	20	3	1.1
40	20	3	1.1
50	19	4	1.0
60	20	3	1.0
70	20	3	1.0
80	18	2	1.1
90	20	2	1.0
100	15	5	1.0

Table 6.9: Best parameter values for the three descriptors at different resolutions

Similarly there is a very large confusion between small-tiles and grass and also between grass and gravel. The CCH descriptor only consists of 64 dimensions and so is among the smallest of the descriptors.

The time taken for 5-fold cross-validation for all of the descriptors is described in Table 6.8. These times are in seconds and are for the random forests classifier used for validation. Note that the values are plotted on a logarithmic scale. Here we can observe that for all grid-sizes, TSURF takes the least amount of time. The most amount of time is mostly taken by LTP or similar descriptors. This is natural, since TSURF has the smallest descriptor vector.

Only three descriptors, TSURF, LTP and LATP, have an additional parameter each, which also needs to be optimized. Table 6.9 shows the respective parameters of each of these descriptors, which performed the best at each resolution. These results have been

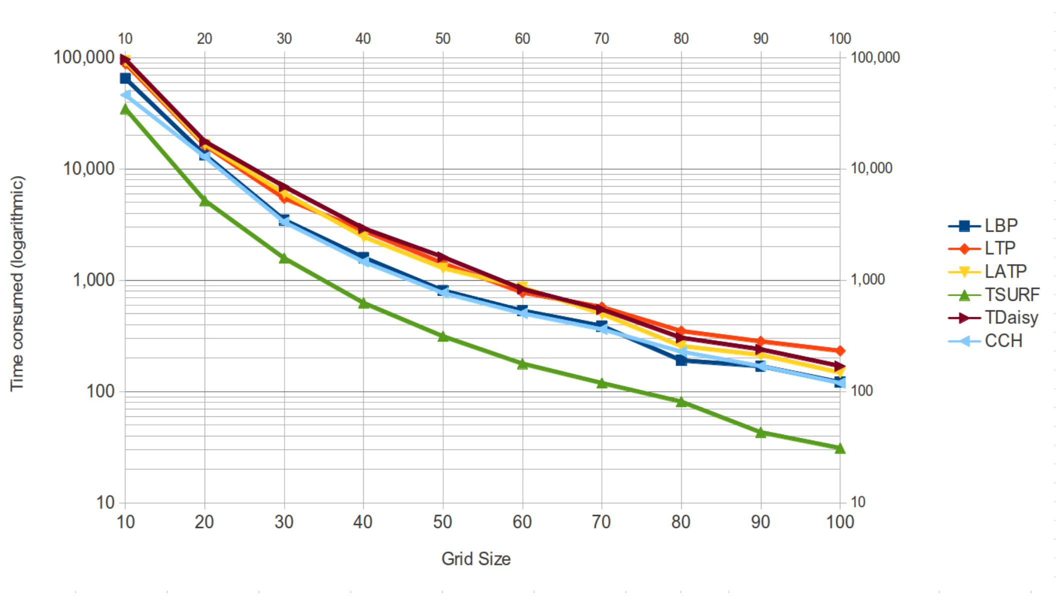


Figure 6.7: Graph of time taken for cross-validation depicted on logarithmic scale for Random Forest classifier

presented in (Khan *et al.*, 2011a) and (Khan *et al.*, 2011b).

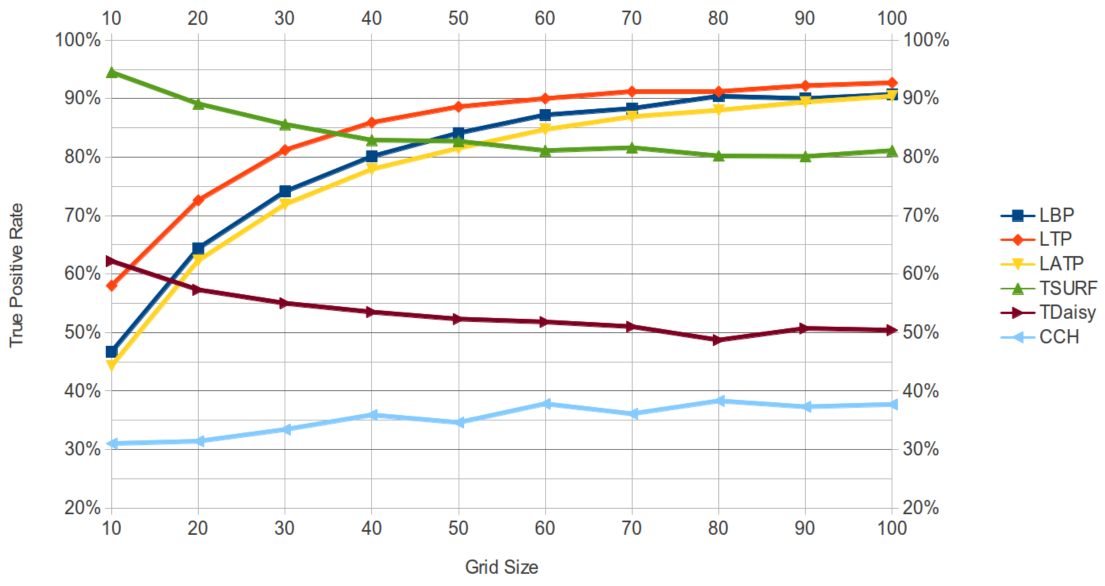


Figure 6.8: Graph of descriptor accuracies at different grid-sizes with J48 Decision Tree

6.5.1 Other Classifiers

Fig 6.8 shows a graph of classification results obtained at different grid sizes by training on the C4.5/J48 Decision Tree described in section 2.4. These results are consistent with the results obtained through Random Forest classifier (Fig.6.5). Here also LTP and TSURF are the best performing descriptors at different resolutions, although absolute accuracies are less than that of the random forest.

Results of running the classification experiments using a Support Vector Machine (SVM) (section 2.5) are depicted in the graph in Fig 6.9. This classifier shows a very different behavior than the random forest classifier. Here the descriptor TDaisy performs better than many other descriptors. Whereas, with the random forest the performance of TDaisy is very low. LBP, LTP and LATP perform very badly most of the time and only perform well at very low cell sizes.

Accuracies of terrain classification through a Naive Bayes classifier (section 2.1) are shown as a graph in Fig. 6.10. These results are more consistent with the Random Forest classifier, except for TSURF. TSURF does not perform very well for even low cell sizes. LBP, LTP and LATP perform the best as in the Random Forest case. TDaisy performs as usual. However, it does not exceed the performance of LBP, LTP and LATP at the lowest cell sizes, as it does with other classifiers.

Fig. 6.11 displays a graph for the results of the classification obtained by using a k-NN classifier (section 2.2). This classifier performs very similarly to the random forest.

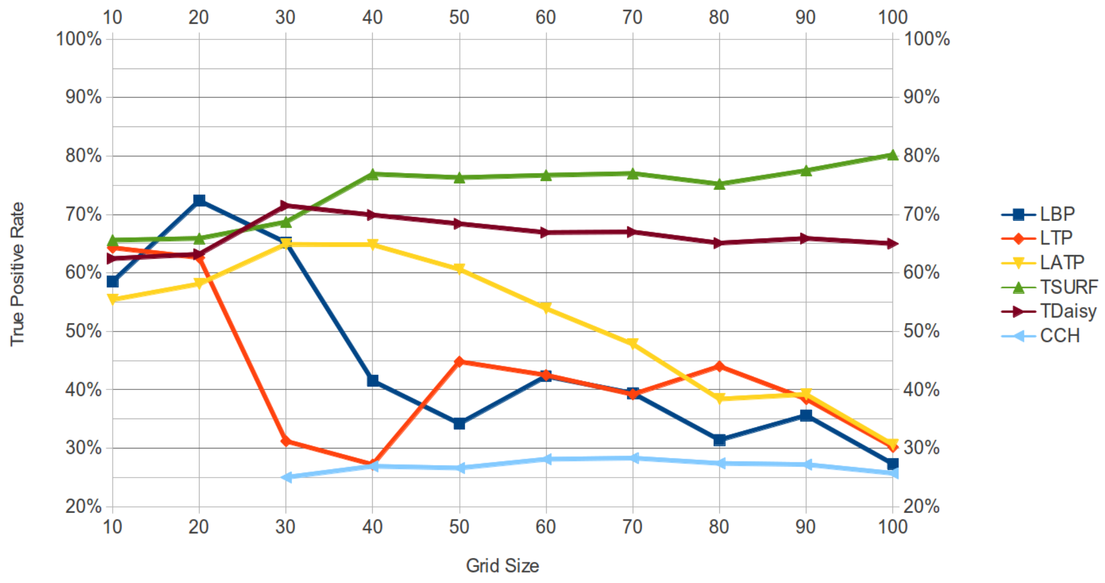


Figure 6.9: Graph of descriptor accuracies at different grid-sizes with SVM

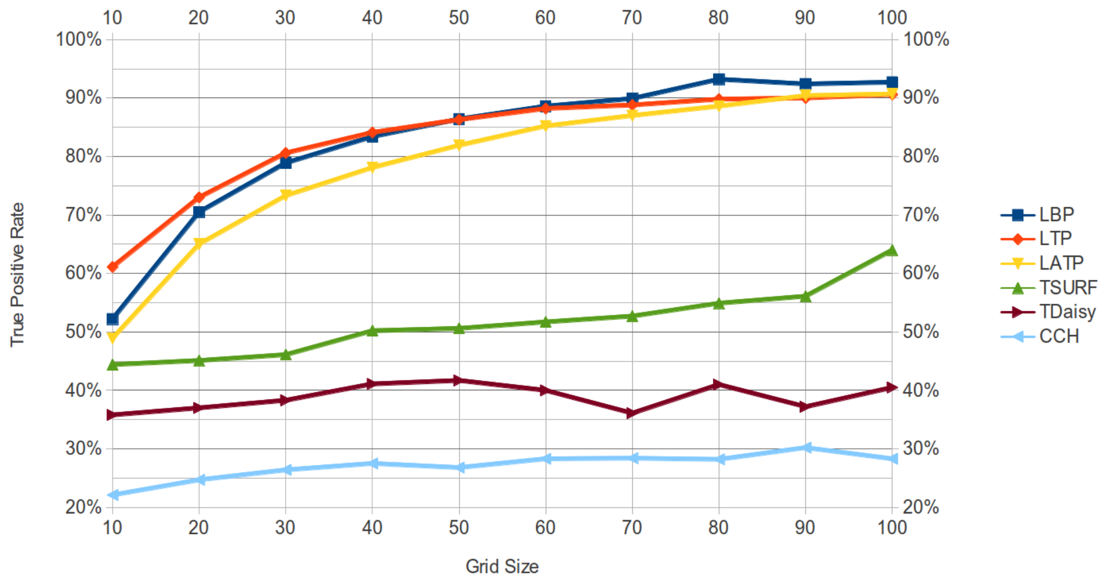


Figure 6.10: Graph of descriptor accuracies at different grid-sizes with Naive Bayes

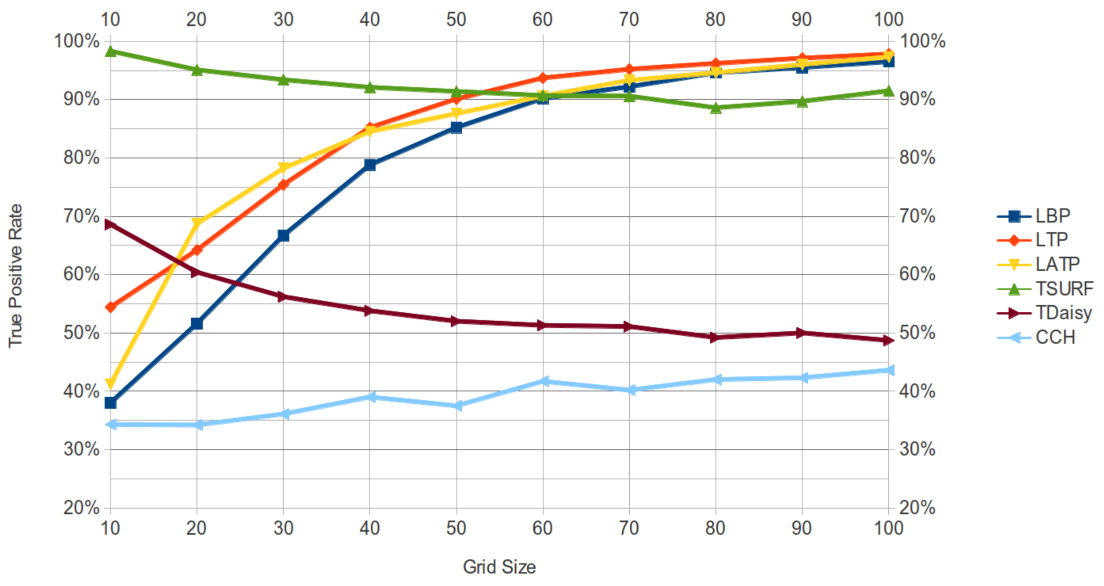


Figure 6.11: Graph of descriptor accuracies at different grid-sizes with the k-NN classifier

However there are some bumps of performance at some resolutions.

The classification results obtained through a K* classifier (section 2.3) are shown with a graph in Fig. 6.12. This classifier produced completely unpredictable and non uniform results. Only TSURF and LATP perform consistently well. LBP and LTP have very inconsistent performance. At the lowest cell size most of the descriptors fail to produce a result in a reasonable amount of time.

6.5.2 Time comparison

Here we present the time taken by different classifiers for cross-validation. This cross-validation was performed on our cluster nodes. The cluster was used for grid search of optimal parameters of some of the image descriptors and for testing different grid resolutions. In some cases, the time required to perform the full cross-validation was huge, so the tests were cancelled after a few days of running. Weka software has a feature that it can predict the expected time required for the cross-validation based on the amount of data and the type of the classifier chosen. So, in such cases, where the tests were cancelled, we show the expected time required by Weka for the cross-validation.

The time taken by each descriptor with the Naive Bayes algorithm is depicted in Fig. 6.13. This graph is drawn is logarithmic scale, since the time taken increases very steeply when we move from large cell sizes to smaller cell sizes. The most time consuming

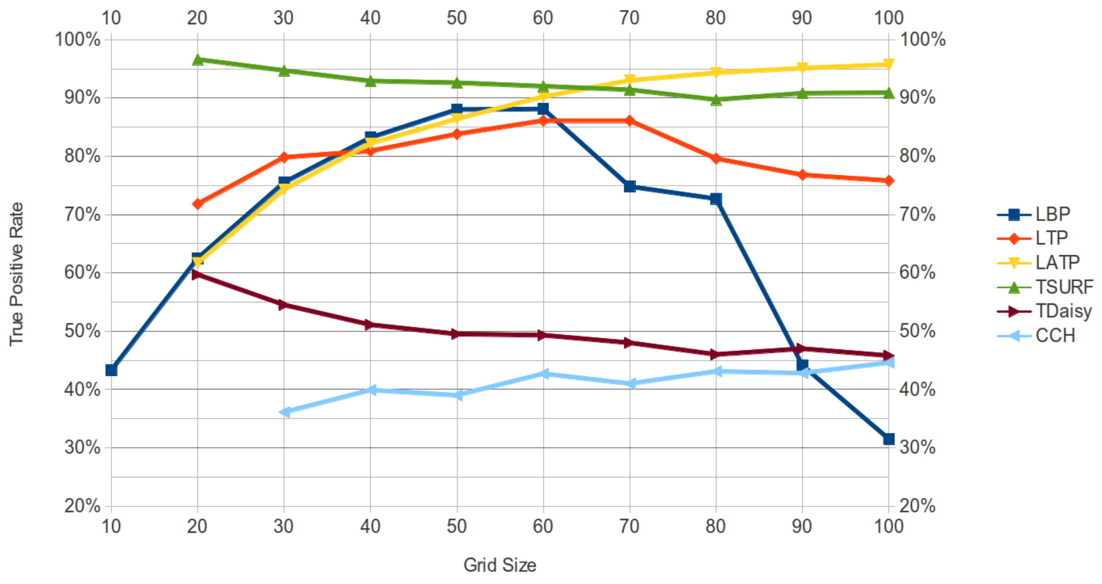


Figure 6.12: Graph of descriptor accuracies at different grid-sizes with K*

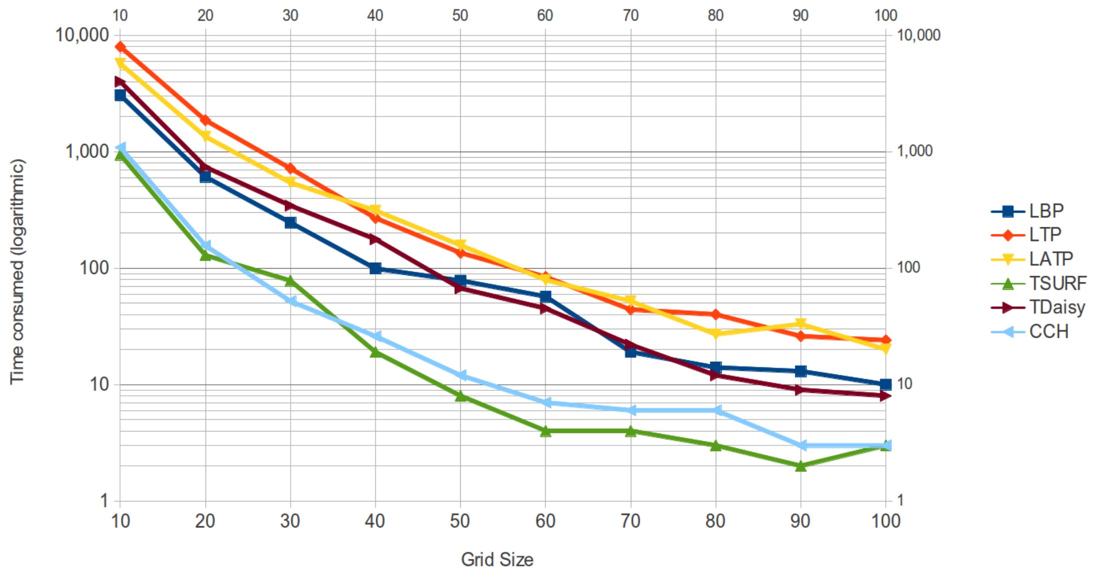


Figure 6.13: Time consumed for cross-validation by the descriptors at different grid-sizes with Naive Bayes algorithm

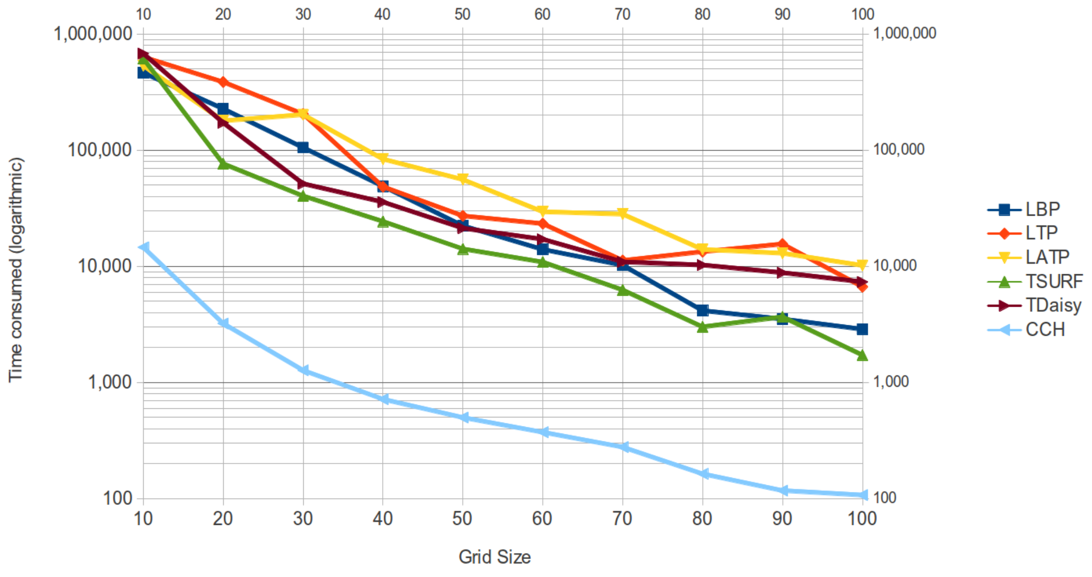


Figure 6.14: Time consumed for cross-validation at different grid-sizes with Linear SVM classifier

image descriptors were both LTP and LATP. This is understandable since both of the descriptors are the longest of the image descriptors in our set. The least time taken was mostly by TSURF and sometimes by CCH. This is also evident since these two descriptors are the smallest of the image descriptors that we used. Time consumed for training and testing on LBP and TDaisy descriptors is in the middle of the two extremes. LBP has a slightly bigger descriptor vector size than TDaisy.

The time taken by Linear SVM algorithm for training and testing is shown as a graph in Fig. 6.14. In this case, all of the descriptors take a similar amount of time, except the CCH descriptor, which takes significantly less time.

The graph in Fig. 6.15 shows the time consumed by the C4.5 algorithm. For this algorithm, the lowest time consuming image descriptor is again TSURF. All other descriptors take a similar amount of time, which is about 10 times the time taken by TSURF.

The graph in Fig. 6.16 shows the time taken by k-NN classifier for different image descriptors. Here TSURF is the fastest image descriptor, whereas LTP is the slowest one. CCH is also faster than others excluding TSURF.

Finally, Fig. 6.17 shows the graph detailing the time taken by K*-NN algorithm. It is different than others in that TDaisy proves to be the slowest of the image descriptors. TSURF is still the fastest image descriptor. LBP also gets fast at lower cell sizes.

Now we compare the overall time consumption by the different classifiers. Overall, Naive Bayes proves to be the fastest classifier, with its maximum time consumption in the order of 10,000 seconds. It is followed by Random Forest, which has a maximum

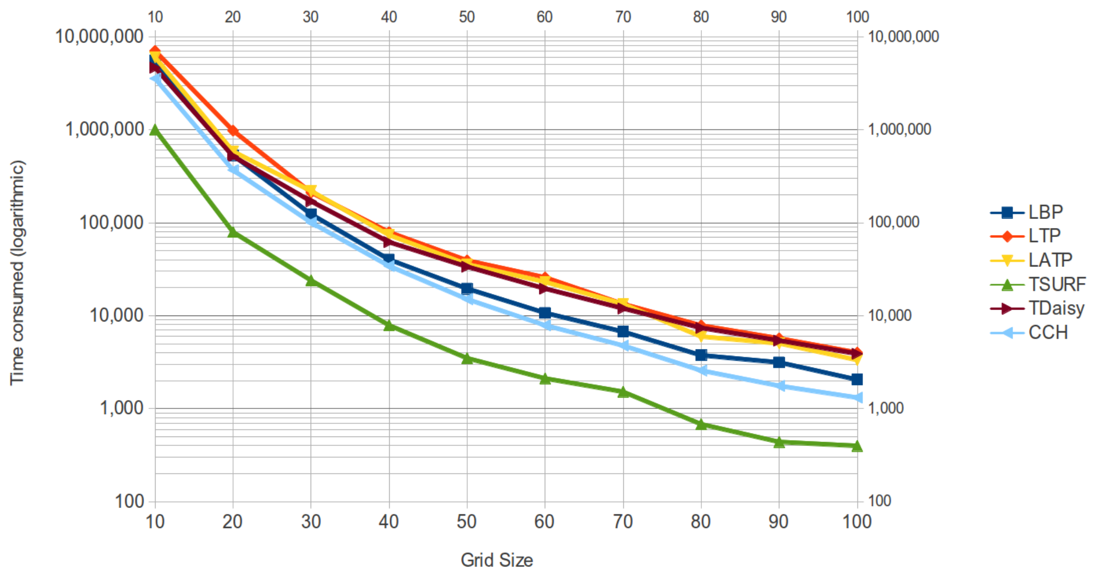


Figure 6.15: Time consumed for cross-validation at different grid-sizes with C4.5/J48 classifier

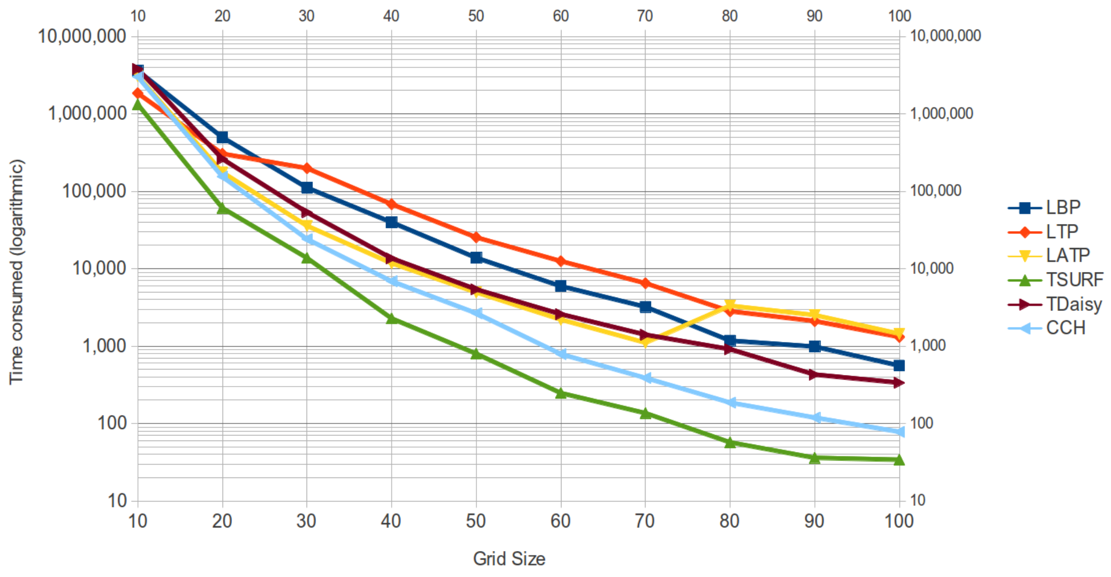


Figure 6.16: Time consumed for cross-validation at different grid-sizes with k-NN classifier

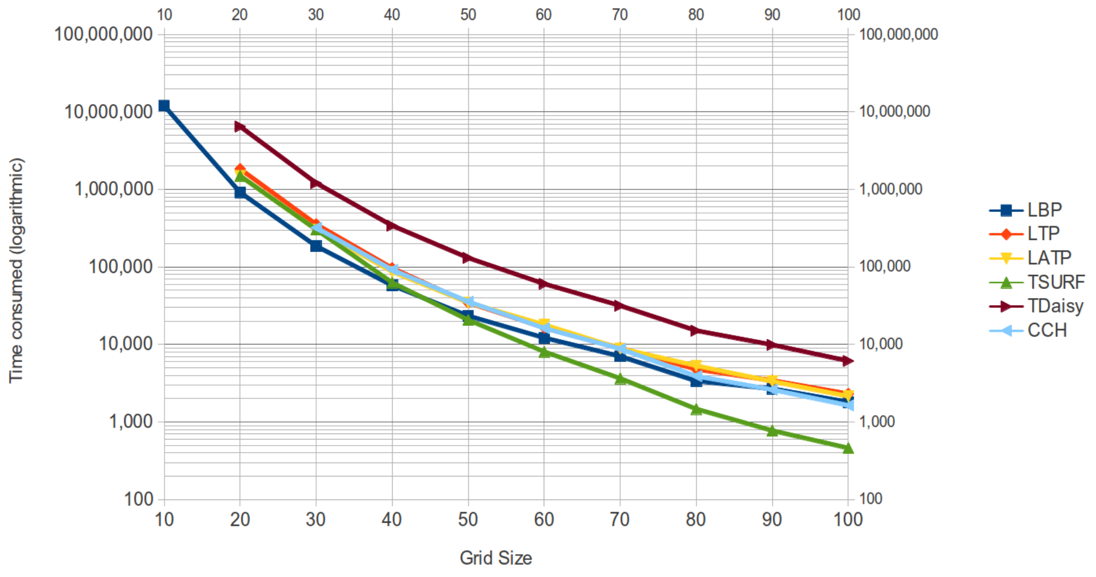


Figure 6.17: Time consumed for cross-validation at different grid-sizes with K*-NN classifier

time consumption of the order of 100,000 seconds. Then comes the Linear SVM having a maximum time consuption in the order of 1,000,000 seconds. After that come the remaining three classifiers C4.5, k-NN and K*-NN, which have their maximum time consumption in the order of 10,000,000 seconds. Experiments with these classifiers were not run to the end, since the time consumption predicted for them was too large. The Weka toolbox has the ability to observe the data and the classifier trend to predict the amount of time to be consumed by a particular classifier.

Although Naive Bayes proves to be the fastest classifier, its results are not the best ones. The second fastest classifier Random Forest has better results so we used it for most of our tests.

Grid-size	LBP	LTP	TSURF
10	33.6%	38.9%	**99.6%**
20	48.4%	55.7%	**96.8%**
30	59.4%	67.3%	**88.7%**
40	66.7%	73.6%	**81.5%**
50	71.0%	**78.1%**	76.6%
60	74.8%	**80.3%**	71.9%
70	77.3%	**81.7%**	71.2%
80	78.2%	**83.3%**	66.0%
90	79.6%	**84.7%**	67.7%
100	79.5%	**84.4%**	63.3%

Table 6.10: Classification accuracy of the three descriptors at different grid sizes for the flying robot dataset with the random forest classifier

6.6 Experiments with a Flying Robot

In this section, we present the results of our experiments performed on the flying robot. As mentioned in section 5.1.2, we used an AscTec Hummingbird flying robot fitted with a downward facing VGA camera for our experiments. The robot was flown around at the campus and six terrain types were captured to be classified.

Since only LBP, LTP and TSURF performed well in the wheeled robot case, we only present results of these descriptors in this section. We tested different classifiers on each descriptor and obtained the true positive rate (TPR) of the entire dataset. The TPR is the ratio of the correctly classified instances to the total number of test patterns. Since random forests performed best, only those results are described here. For flying robot experiments, we included an additional terrain class ”bush”, since the robot could fly over bushes and they often appeared in the robot’s field of view. Table 6.10 presents accuracy results of the three approaches on the six terrain types. Here we used 10-fold cross-validation to verify our results.

Fig. 6.18 shows a plot of accuracies for visualization. Here it is clear that, although at lower resolutions (less patches) the texture classifiers such as Local Binary Patterns and Local Ternary Patterns perform the best, at higher resolutions, TSURF features produce much better results. At a grid-size of 50×50, TSURF lags the performance of the best texture descriptor LTP by only 1.5 %. For higher resolutions, TSURF performs better than LTP. At a grid-size of 10×10, TSURF has a performance of 99.6 %, whereas LTP only gives a performance of 38.9 %.

It is to be noted that for grid-sizes lower than 40×40, the performance of the LBP and LTP decreases sharply. This is due to the fact that such a small patch does not include enough neighborhood information for adequate feature description.

The graph in Fig 6.19 shows the area under the curve for ROC curve (Receiver oper-

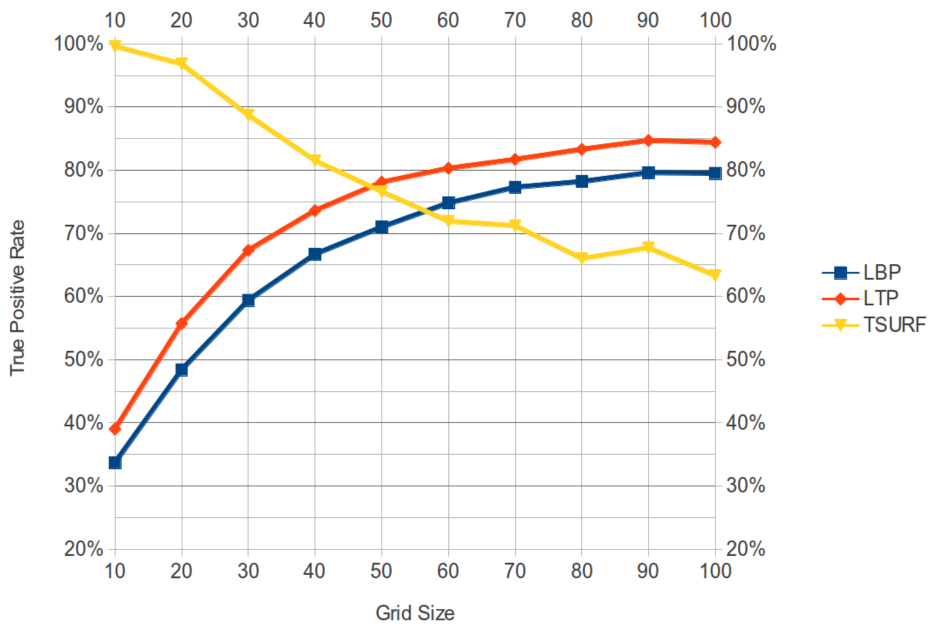


Figure 6.18: Graph of descriptor accuracies at different grid sizes with Random Forest classifier

ating characteristic) (Fawcett, 2006; Flach, 2003) plotted for all grid sizes for the three descriptors with random forest classifier. We show this graph only for random forest classifier, since it was the most successful classifier and would be used in further applications.

In terms of descriptor size, the TSURF descriptor is the smallest descriptor consisting of only 64 dimensions. The LTP descriptor has the longest descriptor consisting of a 512 dimensional vector. LBP is an intermediate length descriptor with 256 dimensions. This has a big impact on training times for the classifier.

For TSURF based classification, different scale levels (σ) described in section 3.1 ranging from 2 to 20 were tried. Higher values of this scale parameter for descriptor calculation close to 20 gave the best result in all of the cases. For LTP-based classification, we also tried different values for the threshold value k described in section 3.3 having values between 2 and 20. It was observed that small values of the threshold close to 5 gave better results. These parameters were optimized by grid search.

Table 6.11 shows an example confusion matrix. This matrix has resulted from the validation of TSURF descriptors on a grid cell size of 20×20 and a scale of 20. Here it is evident that there was some confusion between grass and big-tiles, and between grass and asphalt.

Table 6.12 shows an example confusion matrix resulting from the validation of LBP descriptor on a grid of cells of size 80×80. Here, we choose a different resolution than

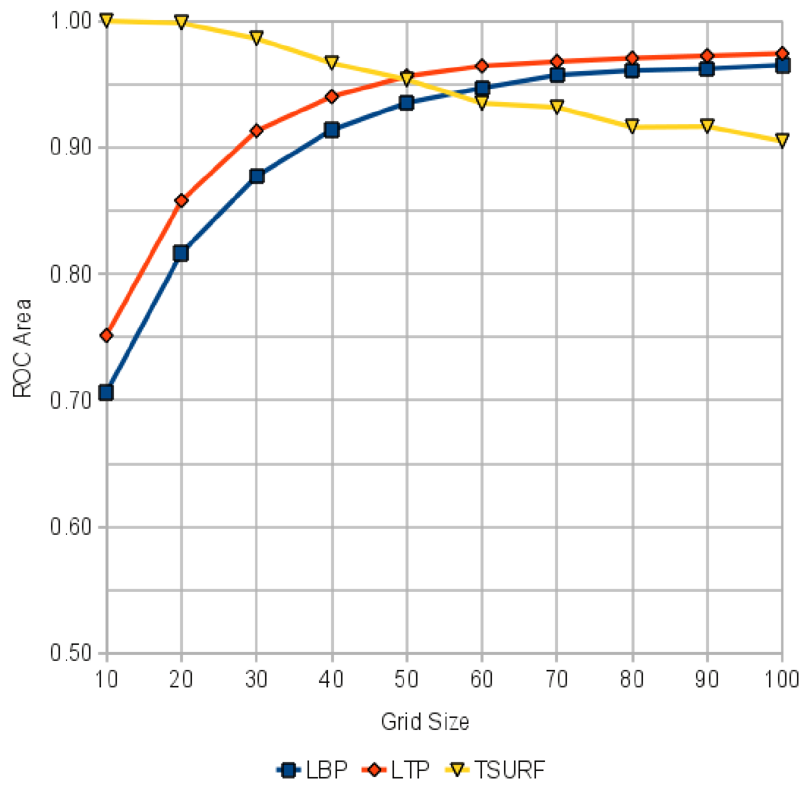


Figure 6.19: Graph of ROC area under the curve at different grid-sizes with Random Forest classifier

	gravel	grass	big-tiles	small-tiles	asphalt	bush
gravel	6,497	62	29	87	44	103
grass	29	6,255	223	6	206	103
big-tiles	5	125	6,606	0	72	14
small-tiles	20	7	1	6,781	2	11
asphalt	2	70	38	0	6,698	13
bush	16	13	7	35	1	6,750

Table 6.11: Confusion matrix for TSURF at grid cell size 20×20

	gravel	grass	big-tiles	bush	small-tiles	asphalt
gravel	426	32	5	86	21	4
grass	25	335	36	20	82	76
big-tiles	1	25	485	0	38	24
small-tiles	18	4	1	529	20	2
asphalt	17	34	22	21	450	29
bush	10	18	41	8	33	464

Table 6.12: Confusion matrix for LBP at grid cell size 80×80

	gravel	grass	big-tiles	bush	small-tiles	asphalt
gravel	466	28	4	57	7	12
grass	36	400	30	6	38	64
big-tiles	2	27	492	0	19	33
small-tiles	28	6	1	522	14	3
asphalt	14	62	14	8	457	18
bush	13	20	17	1	2	521

Table 6.13: Confusion matrix for LTP at grid cell size 80×80

TSURF, since LBP performs better with larger cell sizes, whereas, TSURF performs better with smaller cell sizes. So, it makes sense to compare the better performing resolutions of these descriptors. We can observe here that there was some confusion between gravel and small-tiles, between grass and asphalt, and between grass and bush. Other confusion values are not as high.

Table 6.13 shows an example confusion matrix from the validation of LTP descriptors on a grid cell size of 80×80 and a scale of 5. The biggest confusion occurs between grass and bush, between grass and asphalt, and between gravel and small-tiles. So we see that all of the TSURF, LBP and LTP have always more confusion between grass and asphalt. Other high confusion values vary between different terrain types.

The time taken for 10-fold cross-validation for all of the image patches is described in Table 6.14. These times are in seconds and are for training and classification through the random forests classifier used for validation. Here we can observe that for all grid sizes, TSURF takes the least amount of time. The most amount of time is taken by LTP. This is natural, since TSURF has the smallest descriptor vector as described before and LTP has the largest. These values can be visualized in the graph given in Fig. 6.20. These results have been presented in (Khan *et al.*, 2012).

Grid cell size	LBP	LTP	TSURF
10	79,156	109,123	**37,981**
20	12,830	19,364	**6,025**
30	4,544	7,038	**2,090**
40	2,328	3,794	**990**
50	1,217	1,763	**464**
60	891	1,219	**336**
70	504	713	**148**
80	451	643	**154**
90	291	427	**92**
100	185	260	**46**

Table 6.14: Time taken in seconds for cross-validation with the random forest classifier

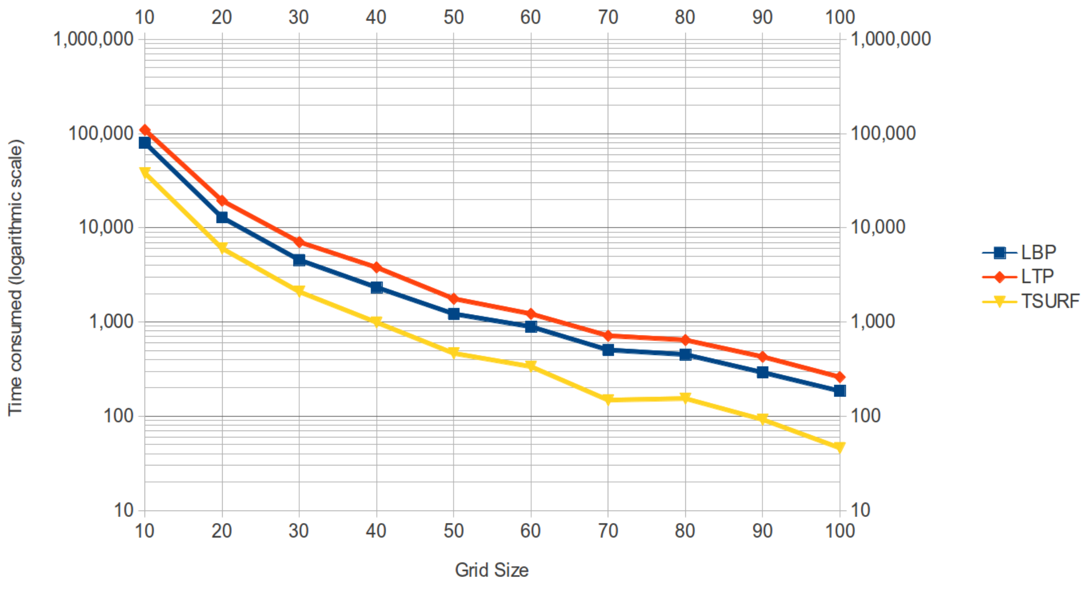


Figure 6.20: Graph of time taken in seconds for cross-validation with Random Forest classifier for flying robot

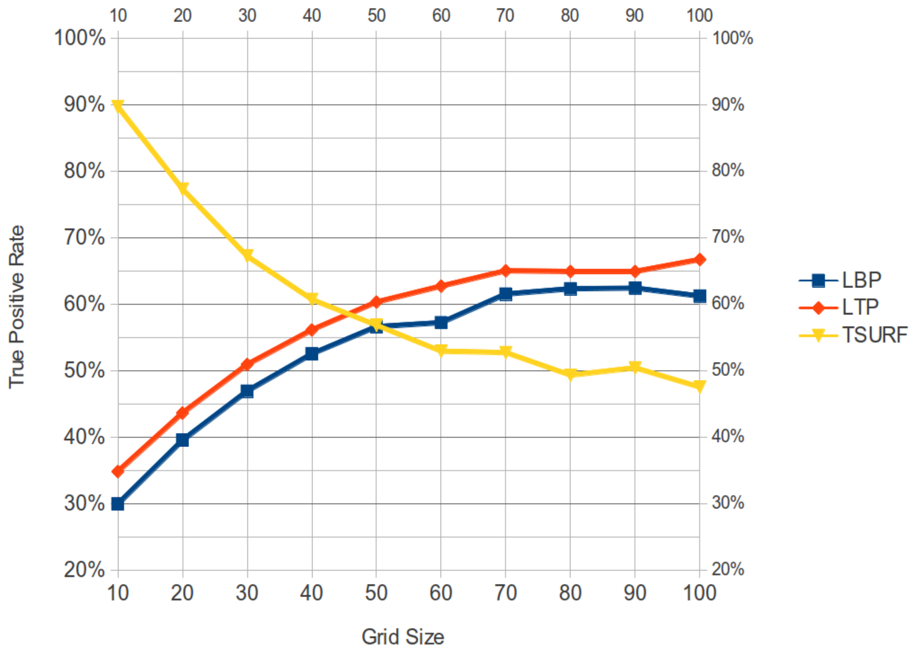


Figure 6.21: Graph of descriptor accuracies at different grid sizes with C4.5/j48

6.6.1 Other descriptors

Here we describe results obtained from classifiers other than random forest. Since it is already established that neither TDaisy nor CCH perform well, they are omitted for clarity in the results graphs. Initial tests with these two descriptors on the flying robot data also didn't provide good results.

Results of running the classification experiments using a C4.5 algorithm with J48 implementation (section 2.4) are depicted in the graph in fig 6.21. These results are consistent with the results obtained through Random Forest classifier (Fig.6.5). Here also, LTP and TSURF are the best performing descriptors at different resolutions, although absolute accuracies are less than that of Random Forest. LTP starts off with good accuracy at cell size 100 $\times$100 and is the best one till 50$\times$50. After that, TSURF becomes the best one and goes up to 90% accuracy at the grid size 10$\times$10. LBP gives similar result to LTP, but has slightly lower accuracy than LTP at all grid resolutions. The best accuracy of LTP is around 75% at the resolution of 100$\times$100, which is not a very good accuracy rating.

Accuracies of terrain classification through a Linear SVM classifier (section 2.6) are shown as a graph in fig 6.22. This classifier shows very different behavior from Random Forest classifier. Here the descriptor LTP has the best overall performance. However, it fails to produce an output at the lowest cell size of 10$\times$10 because of probably too many samples. TSURF performs low at all resolutions. Only at 10$\times$10 does it perform better

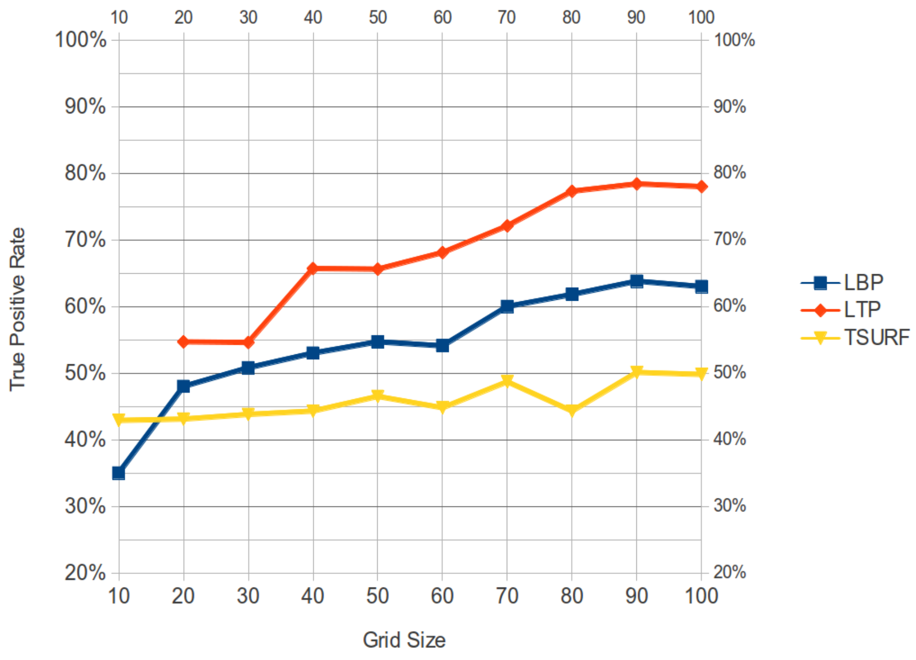


Figure 6.22: Graph of descriptor accuracies at different grid sizes with linear SVM

than LBP, but still not good enough.

The classification results obtained through a Naive Bayes classifier (section 2.1) are shown with a graph in Fig. 6.23. This classifier produced completely unpredictable and non uniform results. Only LBP and LTP perform consistently well. TSURF has very inconsistent performance. Contrary to usual trend, TSURF does not improve with decreasing grid cell size. It starts off bad at the resolution of 100×100, but gets worse as it moves towards smaller cell sizes. LBP and LTP on the other hand have consistent performance as in the case of the other classifiers. Their best performance is at the biggest grid cell size of 100×100 and lower at smaller cell sizes, dropping below 40% at the smallest cell size of 10×10. Only at this resolution do they get worse than TSURF. For all other resolutions, their result is better than TSURF. LTP has a slightly better performance than LBP at all resolutions, just like with other classifiers.

Fig. 6.24 shows a graph of classification accuracy obtained at different grid sizes by training on the k-NN described in section 2.2. These results are more consistent with the Random Forest classifier, although much more steep. TSURF starts at very low performance of just below 60% for the resolution of 100×100 but improves sharply to above 95% at the resolution of 10×10. LBP and LTP start with a high performance with the larger cell sizes and drop sharply in performance towards smaller cell sizes reaching around 30% towards the end. There is also a much wider gap between the performances of LBP and LTP through all resolutions. TSURF has some performance bumps at at least

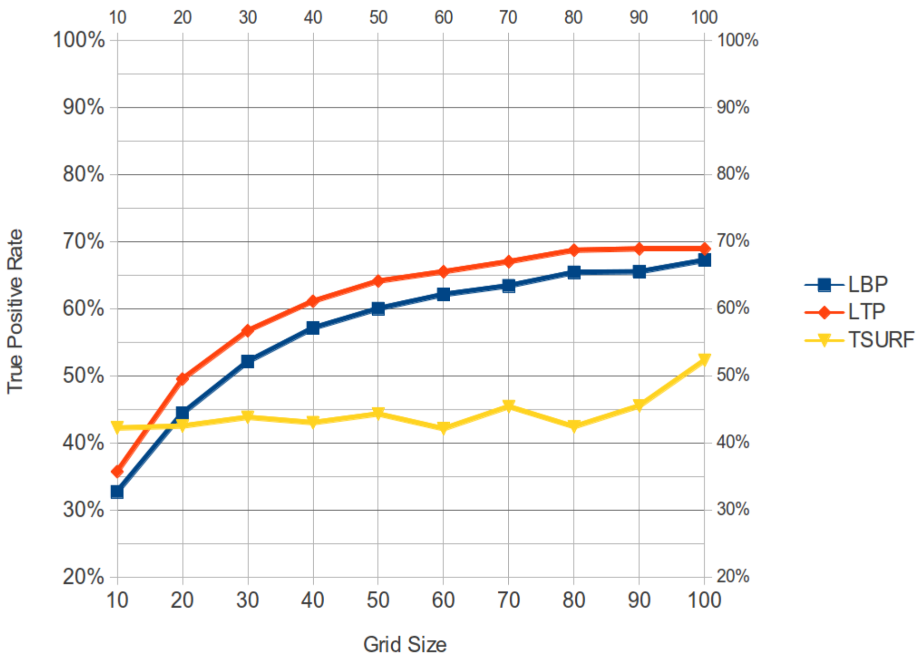


Figure 6.23: Graph of descriptor accuracies at different grid sizes with Naive Bayes

two places.

Fig. 6.25 displays a graph for the results of classification obtained by using a K*-NN classifier (section 2.3). This classifier performs similar to Random Forest for LTP and TSURF. However LBP has an abnormal performance. Only LBP produces an output at the smallest cell size of 10×10. TSURF starts with a very low performance of about 50% for a resolution of 100×100 but increases its performance sharply to above 90% for smaller cell sizes. LTP starts with low performance at the largest cell size and deteriorates further for smaller cell sizes. LBP has a bell shaped curve for the spectrum of resolutions.

All of the other classifiers described in this section either have an inferior performance than Random Forest or a much slower run time. So it is evident that Random Forest is the best choice for experiments with the flying robots, as was the case with the wheeled robot. The time required by different classifiers for training and testing follows a similar pattern as in the case of the wheeled robot experiments, so we don't repeat them here.

6.7 Summary

In this chapter we provided details and results of our terrain classification experiments based on greyscale images. The first robot used for these experiments was our wheeled outdoor robot. This robot was driven around at the campus and images were captured

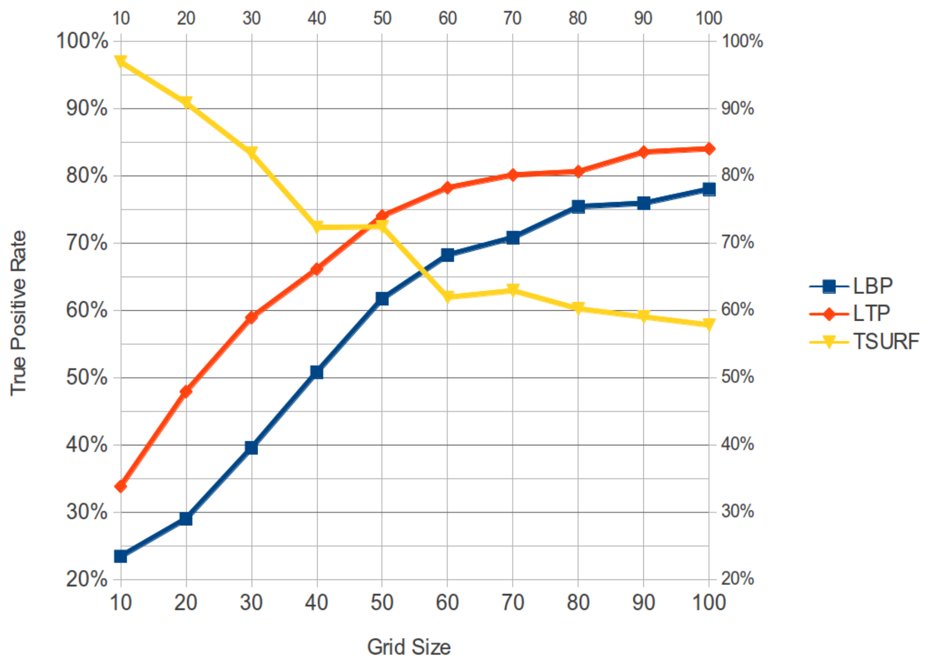


Figure 6.24: Graph of descriptor accuracies at different grid sizes with k-NN

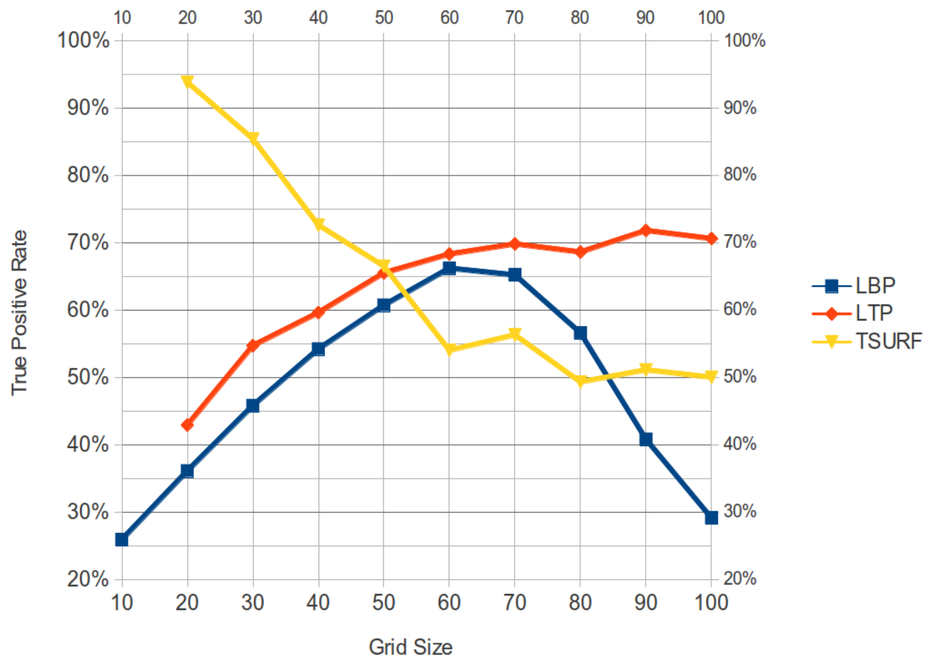


Figure 6.25: Graph of descriptor accuracies at different grid sizes with K*-NN

using a VGA camera. Five terrain types were classified in this experiment. Terrains encountered during these runs were of five different types: asphalt, grass, gravel, small-tiles and big-tiles. Many different runs were performed at different times of the day and different months of the year. This included summer and winter, sunny and shadowed, dry and wet surfaces. Different artifacts were observed in the images including blur due to rough surfaces and fast motion, under and over exposure due to angle of the sun and shadows, difference of texture due to rain, etc. Different machine learning algorithms were used to learn and classify the datasets. Random forests proved to be the best classifier. Other classifiers either had worse results or were too slow. The results for the features were different with different grid size. For grid sizes smaller than 50$\times$50, TSURF performed better; whereas, for larger grid sizes, LTP performed better. The classification accuracy was above 90% for all grid resolutions. This proves that terrain classification can be performed at different resolutions. The image descriptor to be used depends on the choice of grid resolution.

Furthermore, a similar set of experiments was performed using a flying robot. The robot was flown around at the campus and a VGA camera was used to capture the images. The images had a different viewpoint than the images from the wheeled robot. The number of terrains captured increased to six namely asphalt, grass, gravel, small-tiles, big-tiles and bushes. This time, another technique was used to generate groundtruth which is more realistic. These images also had some artifacts. An additional problem with this scenario is the scale of the terrain, which varies a lot with the varying height of the quadrocopter. Again, multiple machine learning techniques were used to train and test the data. The results produced a similar pattern as in the case of the wheeled robot. For cell sizes of the grid larger than or equal to 50$\times$50, LTP performed better, whereas for smaller cell sizes TSURF performed better than other descriptors. Here again Random Forests emerged as the best classification algorithm.

These techniques for terrain classification form the basis on which many different types of applications can be built for outdoor robots (Eck *et al.*, 2007).

Chapter 7

Terrain Classification Based on Color Images

So far we have looked at the problem of terrain classification using greyscale images only. When we look at color images, they show a large variation of color in different scenarios. The same object or surface can show a different color under different lighting conditions. Orientation and strength of the light source plays an important role in defining the color of a scene. Shades also introduce variations in color. The motivation of using greyscale images was to avoid the variation and artifacts that occur in color images. Color images have more information than greyscale images and this could slow down the process. However, this increased information of color can also bring some benefits. Especially in fields like object recognition, color information can play an important role. Color can also be used as one of the texture units in scenarios of texture recognition. (Manduchi *et al.*, 2005)

So we decided to also test the color images for the problem of terrain classification and investigate their usefulness. We implemented and tested different color spaces for this purpose (Lindbloom, 2001). Terrain classification is an important step for outdoor robot navigation (Karlsen and Witus, 2007). This chapter includes work done during a bachelor thesis by Julian Jordan supervised by Yasir Khan.

7.1 Related Work

Some work has been done in the area of Visual Terrain Classification. (Khan *et al.*, 2011a) tests different image descriptors for this purpose, but for greyscale images. In (Brooks and Iagnemma, 2009) a method is described to divide the terrain into known and unknown types. The descriptor used for this purpose combines visual and geometrical information. Wavelets are calculated on a modified HSV color space. (Blas *et al.*, 2008) describes a method to segment the terrains in an image. The descriptor used in this case is the color value of each pixel and the difference of intensities to its neighbors. These descriptors are then trained for specific texture classes, called Textons. Different image regions are then used to calculate histograms. Terrain classification is an important task for autonomous outdoor robots (Kramer and Scheutz, 2007), e.g navigation (Kelly *et al.*, 2007; Procipio *et al.*, 2007; Rao *et al.*, 1993), mapping (Kleiner and Dornhege, 2007), localization (Kosecka and Li, 2004), etc.

The co-occurrence matrix is introduced by (Haralick *et al.*, 1973). 14 different features can be extracted from the co-occurrence matrix, which can be used for texture classification. Intensity values are used in connection with color values in the co-occurrence matrix to improve the accuracy by (Vadivel *et al.*, 2007). A similar approach is used by (Rajadell Rojas, 2008) to improve the accuracy through different color spaces.

The well-known SIFT descriptor (Lowe, 2004) is rotation and scale invariant and also invariant against some other affine distortions. (van de Sande *et al.*, 2008) tests the SIFT descriptor for different color spaces. An invariant color space is combined with the SIFT descriptor in (Abdel-Hakim and Farag, 2006). Both approaches improve the accuracy using color information. In (Bay *et al.*, 2008) the SURF descriptor is introduced as an improvement over the SIFT descriptor in terms of speed.

Some others have fused laser data with color data to generate more complex models for recognition (Andreasson *et al.*, 2005; Konolige, 2000; Vo-Duc *et al.*, 2012) or used them side-by-side for different tasks (Biber *et al.*, 2005; Weiss and Zell, 2005). Along with that (Davidson and Hutchinson, 2003), (Manduchi, 2006) and (Ulrich and Nourbakhsh, 2000) test color images for different purposes in image processing, sometimes in outdoor environments. (Poppinga *et al.*, 2008) combines time of flight camera output with stereo cameras to detect drivable ground in 3D, whereas (Stamos and Allen, 2000) used range and image sensors to generate photo-realistic 3D models. (Schäfer *et al.*, 2005) also use stereo vision for obstacle avoidance in off-road navigation of their large outdoor robot RAVON (Schäfer *et al.*, 2006).

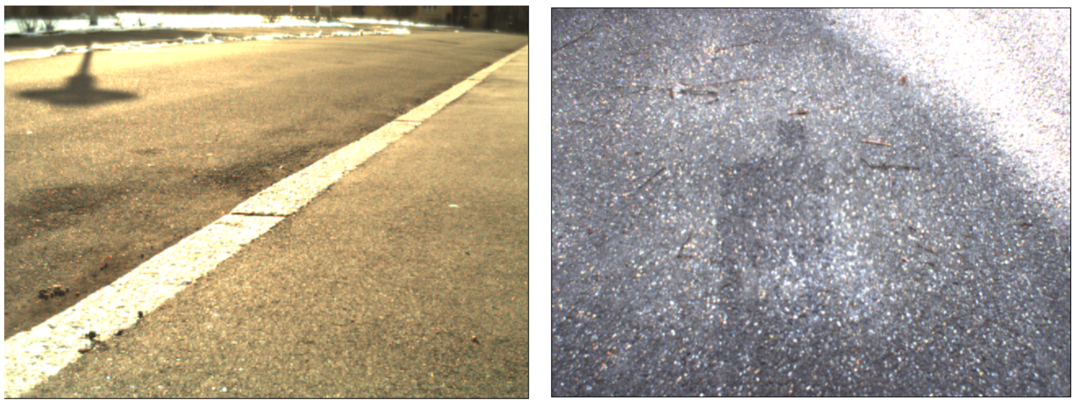
7.2 Terrain classes

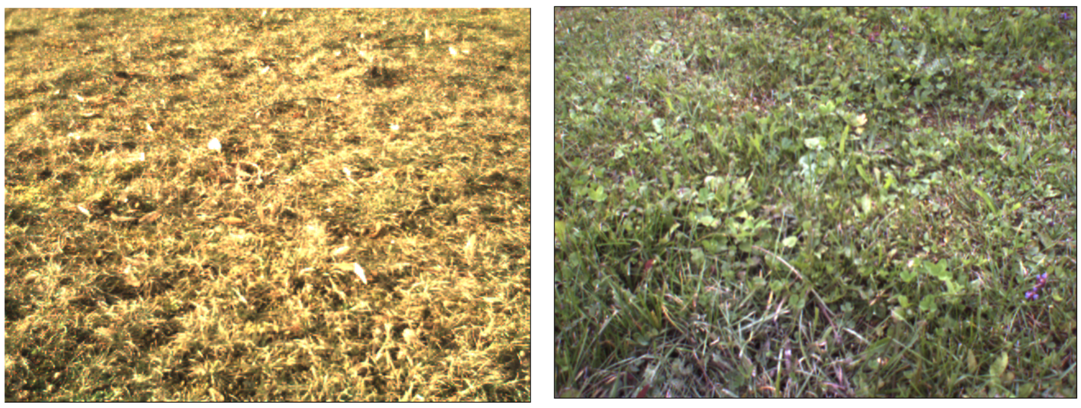
Five terrain types were chosen for carrying out these experiments. These terrains are present on the university campus and are captured through two different cameras mounted on the wheeled robot. All of these terrain types are drivable by the robot.

The robot used for these experiments is our wheeled outdoor robot described in section 5.1.1. The robot was driven around the university campus on each of the terrains and images were captured using the on-board camera. We used two types of cameras to capture two different datasets.

The first camera used is a Point-Grey Firefly color camera (Point-Grey Research, 2012) with a 6 mm lens and VGA (640×480) resolution at 30 to 60 Hz. This is a very light weight camera and is used for most of the experiments in this thesis, since it can be easily fitted both on a wheeled as well as a flying robot. It is availabe either with USB or FireWire (IEEE 1394) interface.

The second camera used is a more powerful Marlin F-04C color camera from Allied Vision Technologies GmbH (Allied Vision Technologies, 2012). It can capture images of a maximum resolution of 780×582 pixels at a frame rate of up to 53 Hz. It has enhanced


(a)


(b)

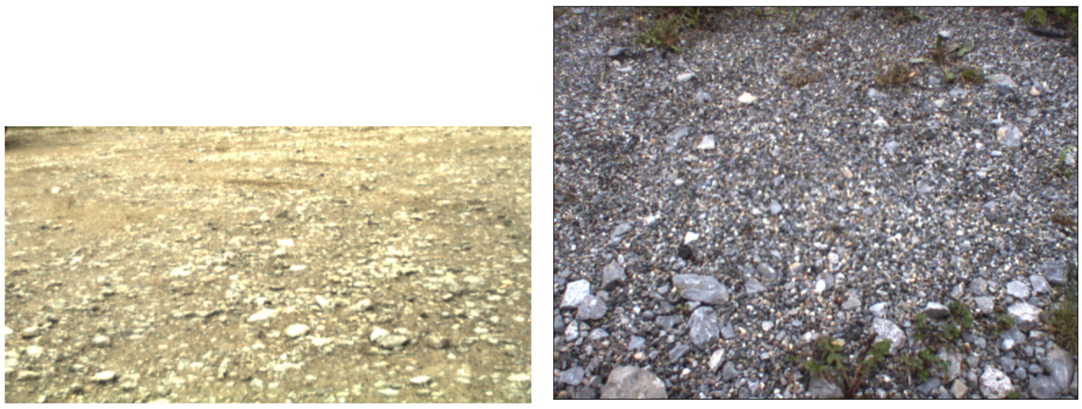

(c)

Figure 7.1: Samples of different terrain types in set-1 (left) and set-2 (right): (a) asphalt, (b) grass, (c) gravel

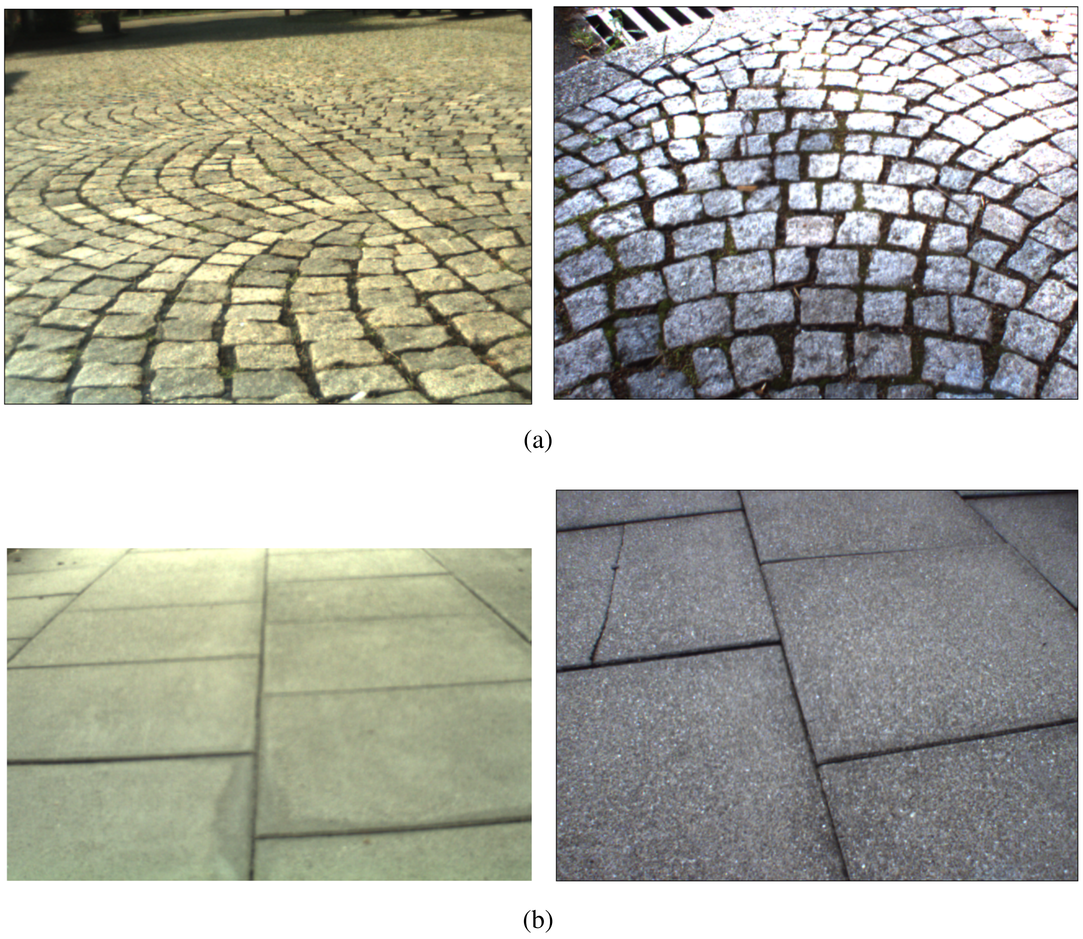


Figure 7.2: Samples of different terrain types in set-1 (left) and set-2 (right): (a) small-tiles, (b) big-tiles

features such as automatic gain, shutter and white balance, which stabilize the image colors under varying light conditions. However, some times these automatic features can change the image too much such that they introduce their own artifacts.

7.3 Color and Intensities

In the hope of increasing the classification accuracy, the descriptors could also be calculated on 14 color spaces, along with the normally used grey-scales. By using color images, we get three color channels in comparison to the grey-scale images, which could improve the terrain classification. The effectiveness of color images has been investigated by van de Sande *et al.* (2008), Rajadell Rojas (2008) and Vadivel *et al.* (2007). In van de Sande *et al.* (2008), different color spaces are investigated for the problem of object recognition. For terrain classification in an outdoor environment, the system should work under greatly varying light conditions. So van de Sande *et al.* (2008) describe five types of light variations:

- Light intensity change:

$$\begin{pmatrix} R_L \\ G_L \\ B_L \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Light intensity shift:

$$\begin{pmatrix} R_L \\ G_L \\ B_L \end{pmatrix} = \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} o_1 \\ o_1 \\ o_1 \end{pmatrix}$$

- Combining both, we get light intensity change and shift:

$$\begin{pmatrix} R_L \\ G_L \\ B_L \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} o_1 \\ o_1 \\ o_1 \end{pmatrix}$$

- Light color change:

$$\begin{pmatrix} R_L \\ G_L \\ B_L \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Light color change and shift:

$$\begin{pmatrix} R_L \\ G_L \\ B_L \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} o_1 \\ o_2 \\ o_3 \end{pmatrix}$$

7.4 Color Spaces

We implemented all together 14 color spaces (Süsstrunk *et al.*, 1999). Some of them have been taken from (van de Sande *et al.*, 2008) whereas others have been taken from the OpenCV image processing library (Bradski, 2000). Most of them have been used in literature and some have compared them for different applications (Schwarz *et al.*, 1987). We briefly describe these colorspaces below:

- RGB

  The normal RGB colorspace.

- RG
  The RG colorspace introduced in (van de Sande *et al.*, 2008) is the normalized version of the RGB colorspace:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix}$$

- Opponent
  Also the opponent colorspace is described in (van de Sande *et al.*, 2008):

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

- Transformed
  In the colorspace introduced in (van de Sande *et al.*, 2008) every channel is normalized individually:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu R}{\sigma R} \\ \frac{G-\mu G}{\sigma G} \\ \frac{B-\mu B}{\sigma B} \end{pmatrix}$$

- HSI
  The HSI color space as used in (Röfer *et al.*, 2003).

$$R' = R/255 \quad G' = G/255 \quad B' = B/255$$

$$H' = \frac{atan2(2R' - G' - B', \sqrt{3}(G' - B')}{2\pi}$$

$$S' = 1 - \frac{min(R', G', B')}{I}$$

$$I' = 0.3 * R' + 0.59 * G' + 0.11 * B'$$

$$H = H' * 255 \quad S = S' * 255 \quad I = I' * 255$$

- HSV
  The HSV colorspace (Halawani and Burkhardt, 2004) as used in OpenCV.

$$V = max(R, G, B)$$

$$S = \begin{cases} \frac{V - min(R,G,B)}{V} & if\, V \neq 0 \\ 0 & otherwise \end{cases}$$

$$H = \begin{cases} 60\frac{G-B}{S} & if\, V = R \\ 120 + 60\frac{B-R}{S} & if\, V = G \\ 240 + 60\frac{R-G}{S} & if\, V = B \end{cases}$$

$$ifH < 0 \quad then\, H = H + 360$$

$$H = \frac{H}{2} \quad with\, 0 \leq H \leq 255$$

- HLS
  The HLS colorspace as used in OpenCV.

$$V_{min} = min(R, G, B)$$

$$V_{max} = max(R, G, B)$$

$$L = \frac{V_{max} + V_{min}}{2} S = \begin{cases} \frac{Vmax - Vmin}{Vmax + Vmin} & if\, L < 0.5 \\ \frac{Vmax - Vmin}{2 - (Vmax + Vmin)} & if\, L \geq 0.5 \end{cases}$$

$$H = \begin{cases} 60\frac{G-B}{S} & if\, V_{max} = R \\ 120 + 60\frac{B-R}{S} & if\, V_{max} = G \\ 240 + 60\frac{R-G}{S} & if\, V_{max} = B \end{cases}$$

$$if\, H < 0 \quad then\, H = H + 360$$

$$H = \frac{H}{2} \quad \textit{damit } 0 \leq H \leq 255$$

- Yiq
  The Yiq color space as described in (Schwarz *et al.*, 1987; Wilhelm Burger, 2005):

$$\begin{pmatrix} Y \\ i \\ q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.321 \\ 0.211 & -0.523 & 0.311 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Yuv
  The Yuv color space as described in (Vadakkepat *et al.*, 2008; Wilhelm Burger, 2005):

$$\begin{pmatrix} Y \\ u \\ v \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Ycrcb
  The Ycrcb colorspace as used in OpenCV and in (Hao and Shi, 2000).

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$C_r = (R - Y) * 0.713 + 128$$
$$C_b = (B - Y) * 0.564 + 128$$

- XYZ
  The XYZ colorspace as used in OpenCV and in (Martinez-Alajarin *et al.*, 2005).

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Lab
  The Lab colorspace as used in OpenCV and in (Gao *et al.*, 2001; Tomasi and Manduchi, 1998). Here the 8-bit images must first be converted into floating-point images with range [0..1].

$$R' = R/255 \quad G' = G/255 \quad B' = B/255$$

  then convert to XYZ-colorspace:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} * \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}$$

XYZ after CIE L*a*b*

$$X' = \frac{X}{0.950456}$$

$$Z' = \frac{Z}{1.088754}$$

$$L' = \begin{cases} 116 + Y^{1/3} & if\, Y > 0.008856 \\ 903.3 * Y & if\, Y \leq 0.008856 \end{cases}$$

$$a' = 500 * (f(X') - f(Y)) + 128$$

$$b' = 200 * (f(Y) - f(Z')) + 128$$

with

$$f(t) = \begin{cases} t^{1/3} & if\, t > 0.008856 \\ 7.787 * t + 16/116 & if\, t \leq 0.008856 \end{cases}$$

Since the result has the ranges:

$$0 \leq L' \leq 100$$

$$-127 \leq a' \leq 127$$

$$-127 \leq b' \leq 127$$

we must adjust the ranges accordingly:

$$L = L' * 255/100$$

$$a = a' + 128$$

$$b = b' + 128$$

- Luv
  The Luv colorspace as used in OpenCV and in (Yang *et al.*, 2005). Here the 8-bit images must first be converted into floating-point images with range [0..1].

$$R' = R/255 \quad G' = G/255 \quad B' = B/255$$

then convert to XYZ-colorspace:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} * \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}$$

XYZ after CIE L*u*v*

$$L' = \begin{cases} 116 * Y^{1/3} & if\, Y > 0.008856 \\ 903.3 * Y & if\, Y \leq 0.008856 \end{cases}$$

$$u' = 4 * X/(X + 15 * Y + 3 * Z)$$

$$v' = 9 * X/(X + 15 * Y + 3 * Z)$$

$$u'' = 13 * L * (u' - 0.19793943)$$

$$v'' = 13 * L * (v' - 0.46831096)$$

Since the result has the ranges:

$$0 \leq L' \leq 100$$

$$-134 < u'' < 220$$

$$-140 < v'' < 122$$

We adjust the ranges accordingly:

$$L = L' * 255/100$$

$$u = (255/354) * (u'' + 134)$$

$$v = (255/256) * (v'' + 140)$$

- Gaussian
  This colorspace is described in van de Sande *et al.* (2008).

$$\begin{pmatrix} \hat{E} \\ \hat{E}_\lambda \\ \hat{E}_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.3 & 0.04 & -0.35 \\ 0.34 & -0.6 & 0.17 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

7.5 Color Descriptors

We used two descriptors for our tests. One is the SURF descriptor described in (Bay *et al.*, 2008) and the other is the Co-occurrence Matrix described in (Haralick *et al.*, 1973). Both of these descriptors are originally used for greyscale images, but we use them on color images. For this purpose, we calculate each descriptor on each color channel of the image. Each color channel can be regarded as a sort of a greyscale image on its own. The 3 descriptors of the 3 color channels are then concatenated to form the final descriptor as depicted in Fig. 7.3.

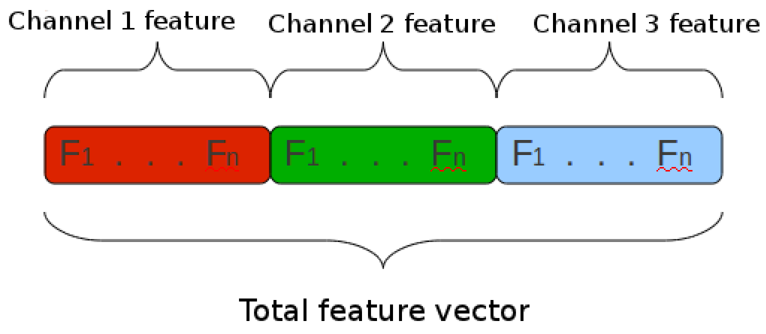


Figure 7.3: Channels and Features

7.5.1 Co-occurrence Matrix

The co-occurrence matrix is introduced by Haralick in (Haralick *et al.*, 1973) and used in many different fields (Pesaresi *et al.*, 2008; Soh and Tsatsoulis, 1999). This matrix describes the probability that a greyscale value i has a certain distance $d = \{d_x, d_y\}$ to another greyscale value j. This information is used to build up a matrix. Features are then extracted from this matrix. In our tests, we extracted 14 features out of the matrix. Since there are 3 color channels, we create a matrix for each channel. Each color channel consists of 8 bits, and each greyscale value can have every other greyscale value in its neighborhood, hence we get a resultant matrix of 256×256. To introduce rotation invariance in the Co-occurrence matrix, 4 matrices rather than one matrix are calculated at rotation angles of 0°, 45°, 90 °, 135°. Let $d_{0^\circ} = \{x, 0\}$, $d_{45^\circ} = \{x, y\}$, $d_{90^\circ} = \{0, y\}$, $d_{135^\circ} = \{-x, y\}$. Then the feature vectors F_{0°, F_{45°, F_{90°, F_{135° are calculated from each of the matrices. The resultant feature vector F is the average and range of the individual vectors and thus has 28 elements.

$$F_{0..13} = \frac{F_{0^\circ} + F_{45^\circ} + F_{90^\circ} + F_{135^\circ}}{4} \tag{7.1}$$

$$F_{14..27} = max(F_{0^\circ}, F_{0^\circ}, F_{0^\circ}, F_{0^\circ}) - min(F_{0^\circ}, F_{0^\circ}, F_{0^\circ}, F_{0^\circ}) \tag{7.2}$$

The Co-Occurrence-Matrix Co has the form:

$$Co = \begin{pmatrix} P_d(0,0), & ..., & P_d(N_g,0) \\ ... & ... & ... \\ P_d(0,N_g), & ..., & P_d(N_g,N_g) \end{pmatrix} \tag{7.3}$$

where $P_d(i, j)$ is the probability that the greyscale value i has a distance of $d = \{d_x, d_y\}$ from greyscale value j. To reach this probability, the number $G_d(i, j)$ of neighborhood of the greyscale value i and j is distributed through the number of all neighborhoods. If $I(x,y)$ is the value of a color channel at position (x,y), then:

$$G_d(i,j) = \sum_{x=0}^{S_x-1} \sum_{y=0}^{S_y-1} \begin{cases} 1 & if\ I(x,y) = i\ and\ I(x+d_x, y+d_y) = j \\ 0 & otherwise \end{cases} \tag{7.4}$$

$$P_d(i,j) = G_d(i,j)/((S_x - |d_x|) * (S_y - |d_y|)) \tag{7.5}$$

We divide an image into grid cells to determine the terrains in those cells. Then a Co-occurrence matrix is determined for each of these cells and a feature vector is determined. The dimension of the feature vector is $Channels * 28$ when all directions are observed and $Channels * 14$ when only one direction is considered.

There are four possible settings to calculate Haralick features:

- 12 features, one observed direction
  The features *contrast* and *maximal correlation coefficient* are not calculated. Only a matrix for the direction $d = \{x,y\}$ is calculated. The descriptor vector then has 12 values.

- 14 features, one observed direction
  All 14 features are calculated with only one matrix for the direction $d = \{x,y\}$. The descriptor is 14 dimensional.

- 12 features, four observed directions
  Here also the features *contrast* and *maximal correlation coefficient* are not calculated. However, four matrices are calculated: $d_{0^\circ} = \{x,0\}$, $d_{45^\circ} = \{x,y\}$, $d_{90^\circ} = \{0,y\}$, $d_{135^\circ} = \{-x,y\}$. Average and range is calculated for all 12 features which gives 24 values.

- 14 features, four observed directions
  All 14 features are calculated along with four matrices $d_{0^\circ} = \{x,0\}$, $d_{45^\circ} = \{x,y\}$, $d_{90^\circ} = \{0,y\}$, $d_{135^\circ} = \{-x,y\}$. Average and range is calculated for all 14 features which gives 28 values.
  Where x and y are the prescribed distance parameters.

When all of the Haralick features are not used, the entries for the unused features in the descriptor vector are set to zero, so the feature vector always has 14 or 28 dimensions.

7.5.2 Color SURF

SURF is described in (Bay *et al.*, 2008) and is an optimized version of the SIFT descriptor (Lowe, 2004). It is already explained in section 3.1.

Since the original SURF descriptor works on greyscale images, to apply it on color images we calculate the SURF descriptor on every color channel, similar to co-occurrence matrix. Thus we get a 64-dimensional vector for each channel. These are then concatenated to get a 192-dimensional vector.

When we look at the variation of R, G and B channels in section 7.3, we can deduce that SURF is color-invariant in color spaces RGB and Transformed. In these color spaces, the variation of light intensity is equalized to variation in light color, since the different values for scaling and shifting effect only one color channel. So, the color invariance of SURF is independent of the color space employed.

7.5.3 Principal Component Analysis

Principal Component Analysis (Smith, 2002; Jolliffe, 1986) can be used to reduce the size of descriptors. With this technique the correlation of the features is minimized (Hinselmann *et al.*, 2011a). For this purpose the basis is changed with the help of a matrix composed of the eigenvalues of the covariance matrix of the training data. The steps for calculation of this basis changing matrix are:

Calculate the mean of the individual features:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{7.6}$$

Subtract the mean:

$$X_i' = X_i - \bar{X} \tag{7.7}$$

Calculate the covariance matrix:

$$M_{cov} = \begin{pmatrix} cov(X_1',X_1'), & ..., & cov(X_n',X_1') \\ ... & ... & ... \\ cov(X_1',X_n'), & ..., & cov(X_n',X_n') \end{pmatrix} \tag{7.8}$$

where

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X-\bar{X})(Y-\bar{Y})}{n-1} \tag{7.9}$$

Eigenvector analysis:
For this we use the OpenCV function

$$cvEigenVV(covariancematrix, Eigenvectors, Eigenvalue)$$

This function delivers the eigenvectors sorted by decreasing eigenvalue.
The change of basis matrix is:

$$M_{pca} = \begin{pmatrix} Eigenvector_1 \\ ... \\ Eigenvector_T \end{pmatrix} \tag{7.10}$$

Where T is the dimension of the descriptor vector after reduction.
The M_{pca} matrix and the mean are saved for later use. To conduct a dimension reduc-

tion of a descriptor vector F, first the mean value of the features is subtracted, $F' = F - \bar{X}$. Then the vector is multiplied with the M_{pca} matrix: $F'' = M_{pca}F'$. The vector F'' has length T.

7.6 Classifier

Here we also used the Random Forest classifier described in section 2.8, since it has already proved to be the classifier in the two previous experiments. It gives the best classification results and the time consumed is also less. The random forest classifier was used here with similar settings as the previous experiments.

Two types of classifiers were used in this case: a multi-class classifier and a one-vs-all classifier (Weston and Watkins, 1999; Wu *et al.*, 2004). A multi-class classifier holds multiple models. The properties of a descriptor are compared to the trained model to assign a label to the descriptor whose model it matches most closely. Whereas, with a one-vs-all classifier, a separate instance of the classifier is generated for each class, each of which holds two models: one for the specified class and the other for all other classes. Such type of classifiers have the advantage that if there is a descriptor which is not close to any class, no label is assigned to this descriptor.

7.7 Experiments with the Robot

In this section, we present the results of our experiments on our wheeled robot. The testing was mostly done on an Athlon-64 3000+ with 1.8 GHz processor and 2 GB RAM under Ubuntu. Tests for both co-occurrence matrix and SURF are conducted on both of the datasets set-1 and set-2. The results given in the following sections are based on Random Forest as a classifier because of its better performance.

7.7.1 Co-occurrence matrix

The graph in Fig. 7.4 shows the results of tests performed using the co-occurrence matrix with different matrix sizes and observation areas. Here different sizes are tried on greyscale images. It is clear from the graph that the accuracy decreases with decreasing size of the observation area. Increasing the matrix size increases the accuracy to a certain limit, after which it starts decreasing again.

Since the number of neighbors is $R = (S_x - |d_x|) * (S_y - |d_y|)$, we had the observation area of 8×8 pixels, matrix size of 32×32 and direction $d = (1,0)$ 54 neighbors, which are distributed on 1024 matrix entries. Hence, a maximum of 5.3% matrix entries are non-zero. We can observe that a cell size bigger than 64×64 does not give good results for an image of 640×480. Similarly, a cell size smaller than 32×32 also decreases in performance. So the optimal cell size in this case is between 32×32 and 64×64. Some

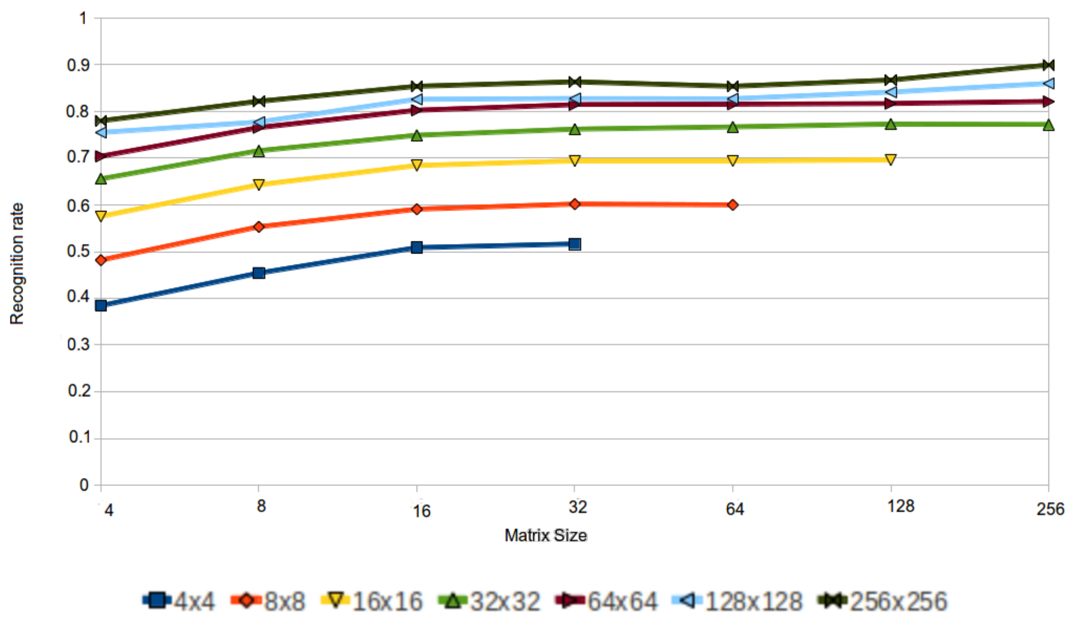


Figure 7.4: Recognition results of the co-occurrence matrix in greyscale. It compares different matrix sizes and observation areas.

results are missing in the graph, since a very low cell size produced a lot of descriptors and the training and testing run took multiple days and thus was canceled.

In Fig 7.5 we compare the accuracy of different color spaces when used with the co-occurrence matrix. These tests are performed on set-1. The matrix size used for these tests is 32×32, whereas the observation area is 64×64 pixels. The best performing color spaces are HSV, HSI and Gaussian. It is interesting to note that except for Transformed and Yiq, the setup with 14 features and 4 directions could not perform better than with 12 features and 4 directions. Fig. 7.6 shows the results obtained from set-2. The best performing colorspace here is the Lab. Luv also has a closely good performance. The worst performance is shown by Transformed and Grey colorspaces.

The time taken to compute the descriptors on set-1 is shown in Fig. 7.7. Here we show the time for HSV and Greyscale colorspaces for comparison. It can be observed that HSV always takes more time than Greyscale. This is natural, since HSV has more data than Greyscale. Also the time taken by different parameters follows a linear trend. It takes the least time for 12 feature with 1 direction. Then comes the 14 features 1 direction configuration which takes lesser time than 12 features 4 directions configuration. Time taken on set-2 follows the same trend.

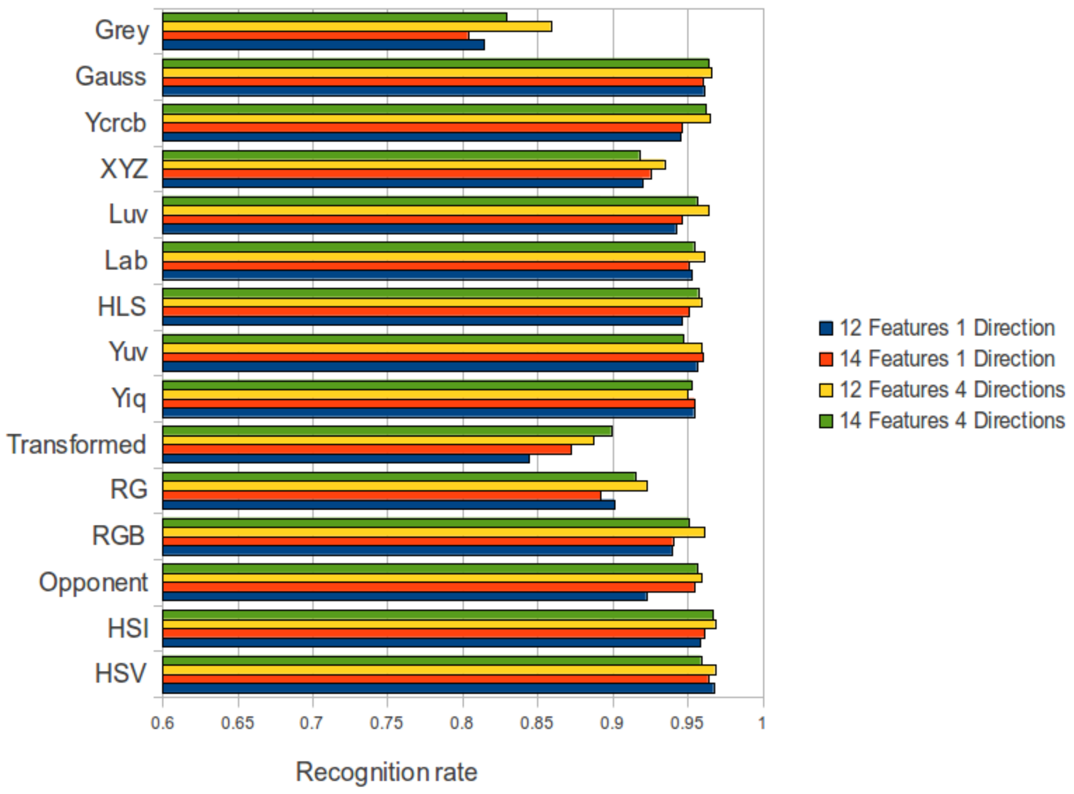


Figure 7.5: Recognition results of the co-occurrence matrix on set-1 with different color spaces. The horizontal scale starts from 0.6 to show the difference clearly.

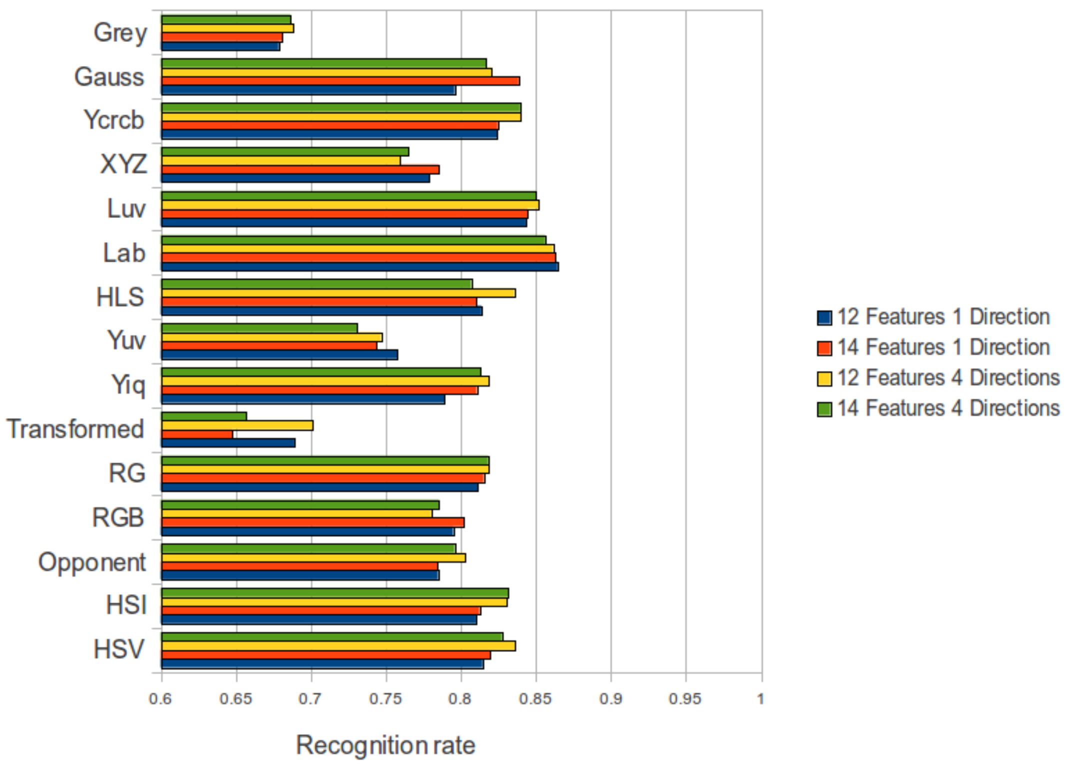


Figure 7.6: Recognition results of the co-occurrence matrix on set-2 with different color spaces. The horizontal scale starts from 0.6 to show the difference clearly.

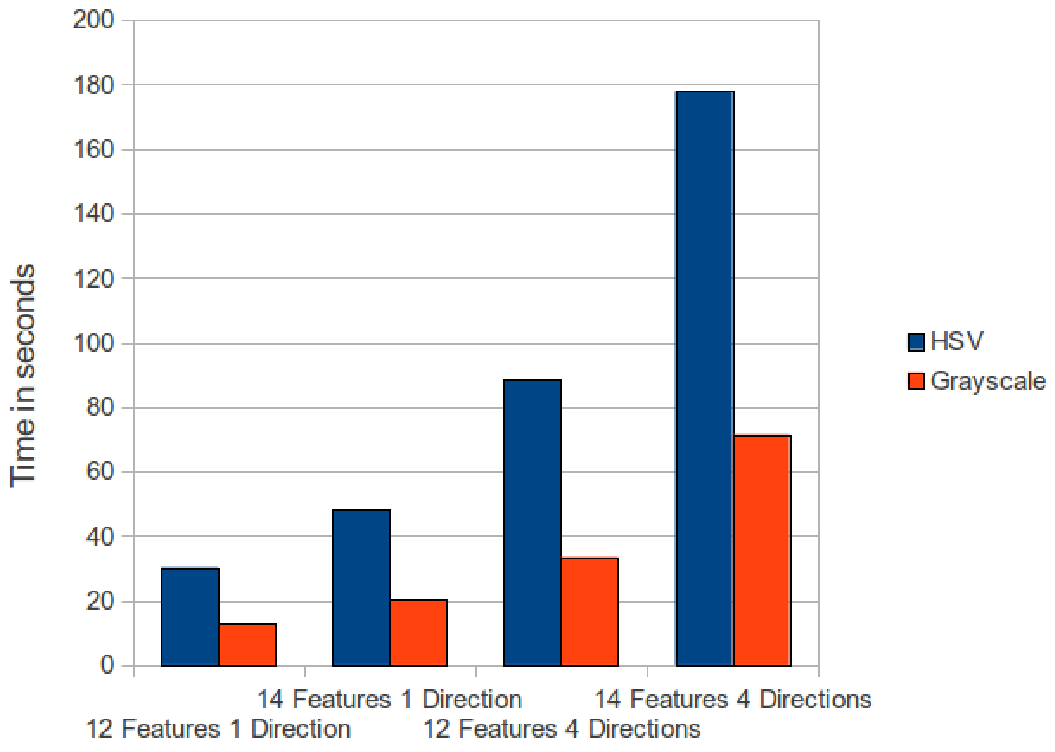


Figure 7.7: Time taken to compute the descriptors on set-1.

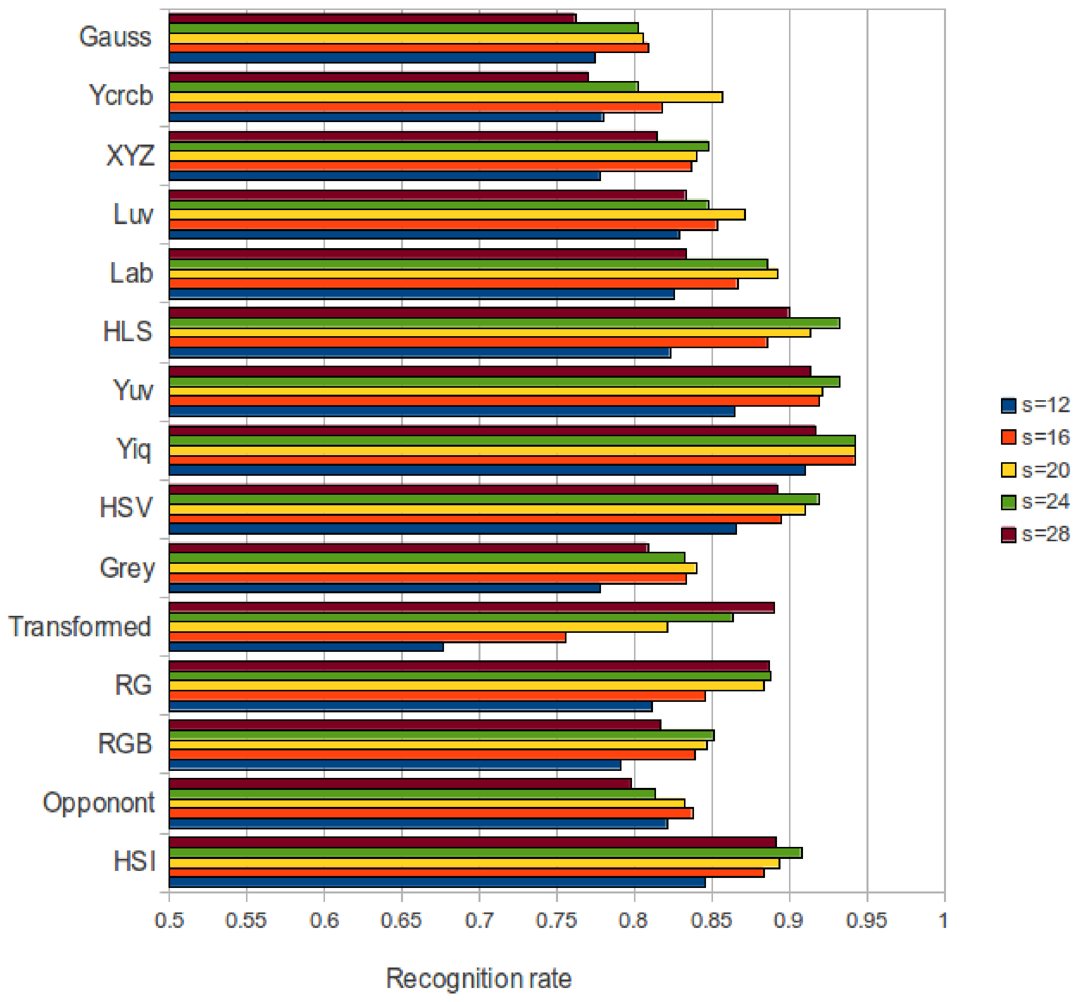


Figure 7.8: Recognition results of the SURF descriptor on set-1 with different color spaces. The horizontal scale starts from 0.5 to show the difference clearly.

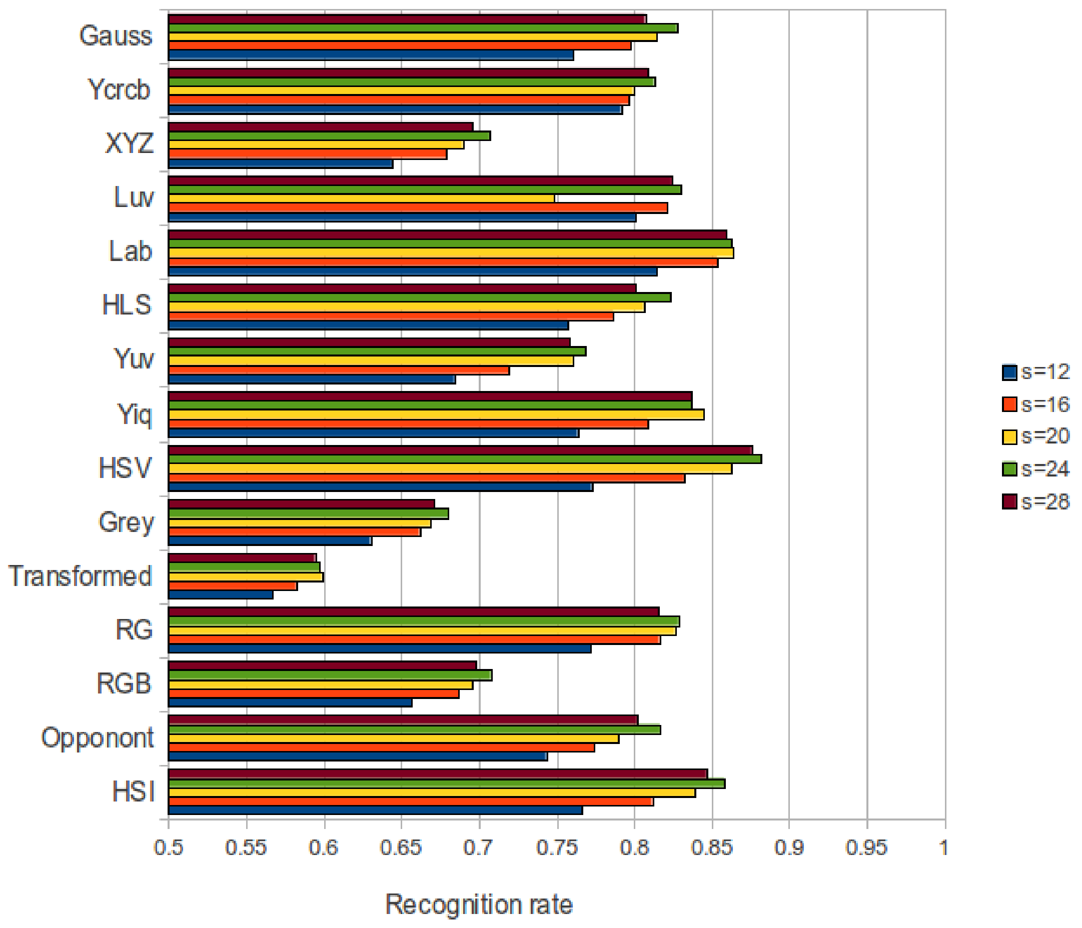


Figure 7.9: Recognition results of the SURF descriptor on set-2 with different color spaces. The horizontal scale starts from 0.5 to show the difference clearly.

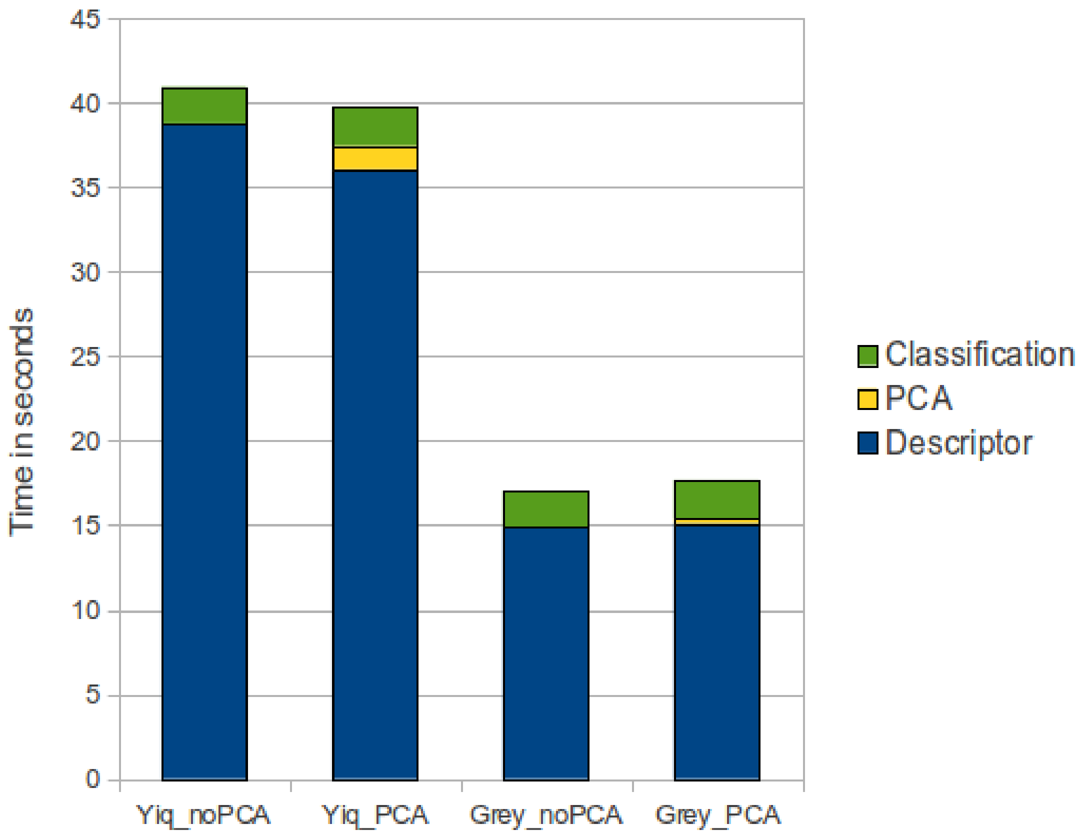


Figure 7.10: Time taken to compute and classify the SURF descriptors along with time for PCA calculation.

7.7.2 SURF

Fig. 7.8 shows accuracy results of the SURF descriptor on set-1. The only adjustable parameter with SURF was the scale factor s. Since this parameter does not affect the classification time, we just need to find the best value of this parameter for every color space. Mostly, we tested the following scale values for all the of the color spaces: 12, 16, 20, 24, 28; as they performed better than other values. The best performing color space is the Yiq color space with 94.3% accuracy. The variation of accuracy in this case is greater than in co-occurrence matrix.

Results of running SURF on set-2 are depicted in Fig. 7.9. With SURF, the results of set-2 are worse than set-1, just like in the co-occurrence matrix. The best performance was achieved by HSV at 88.2%. After that come Lab and HSI color spaces. The worst performance was given by Transformed color space.

Type	Co-oc Grey	Co-oc HSV	SURF Grey	SURF YIQ
SVM Accuracy	0.491	0.643	0.364	0.890
RDT Accuracy	0.815	0.966	0.831	0.945
SVM time in sec.	0.57	1.92	1.26	1.8
RDT time in sec.	0.54	0.47	0.58	0.53

Table 7.1: Comparison of Random Decision Trees and Support Vector Machine with the co-occurrence matrix and the SURF features

Type	PCA Time (sec)	Classification Time (sec)	Accuracy
RDT, PCA-32	0.19	0.57	91.7%
RDT, No PCA	0.0	0.53	94.5%
SVM, PCA-32	0.19	0.36	84.7%
SVM, No PCA	0.0	1.80	89.0%

Table 7.2: Comparison of Random Decision Trees and Support Vector Machine with and without using PCA

Fig. 7.10 shows a graph of times taken for computation and classification of SURF descriptor with and without using PCA. Here we compare the Yiq color space with the Greyscale color space. As expected the computation time increases when using PCA with the Yiq color space. However, the classification time is reduced because of the reduced dimensions. The computation time of PCA is much less than the computation time of the SURF descriptor. The time taken to compute PCA in Yiq is greater than in greyscale. In case of greyscale, there is not much difference in classification with or without PCA, and PCA turns out to be just an overhead.

7.7.3 Classification

Table 7.1 shows a comparison of Random Decision Trees and Support Vector Machine for some classification tests on two color spaces. The SVM required more time for the tests but still gave worse results than RDT.

Table 7.2 shows the results with and without using PCA. Through the PCA we tried to reduce the descriptor dimensions from 192 to 32. The reduction of accuracy by using PCA is only between 2.8% to 4.3%. There is no increase of speed when using PCA with Random Decision Trees. Rather, considering the time to calculate the PCA, the total time actually increases. For SVM, the execution time is decreased by a factor of 5 from 1.8 s to 0.36 s. The PCA requires 0.19 s for calculation, so the total time taken is still reduced from 1.8 s to 0.55 s.

7.8 Summary

In this chapter, we have described terrain classification using image descriptors computed on color images. 14 features were calculated from the co-occurrence matrix, but it is evident that not all 14 features are needed. 12 features give better results with 4 directions. The most accurate results are obtained with the co-occurrence matrix with 12 features and 4 directions and HSV color space, which gives an accuracy of 96.8%. Also, the setup with 12 features and 1 direction gives a usable result with the HSV color space, up to 96.7% on set-1. However, on set-2, it gives only up to 86.4% with LAB color space.

With the SURF descriptor, the best result is with the YIQ color space on set-1, which is 94.3%. On set-2, the best result was 88.2% with the HSV color space. We also tried to use PCA to reduce the dimensions of the feature vector in the hope of speeding up the classification process. There was a little loss of accuracy, whereas not much gain in speed in case of the random forest. With the SVM, the classification time was reduced significantly without reducing the accuracy too much.

We employed two datasets for our experiments. There are major differences between both of the datasets, which influence the performance of different color spaces. Set-1 performed better than set-2 in almost all cases, although the camera used for set-2 had an automatic white color correction. Different color spaces gave different performance. Interesting to note is although the Transformed color space is supposed to be the most color invariant color space, its performance was not so good.

Chapter 8

Conclusion

In this thesis, we thoroughly investigated the applicability of various local descriptors for visual terrain classification on outdoor mobile robots. Many of the current texture classification approaches use sharp images containing mostly a single texture captured from a fixed camera angle under controlled conditions. We used images from real runs of the robot containing blurred images and consisting of multiple terrains. We used two different robots for this purpose. The first was a wheeled robot, which can move over rough terrain at fast speeds. So the images captured by this robot contain a lot of blur and varying light conditions. The second robot we used was a flying robot, which flew over different terrains at variable heights. Images from this robot also contain blur and large changes in scale of terrains.

For greyscale images, We modified three image descriptors: SURF, Daisy and CCH descriptors. We compare their results to three texture-based descriptors, LBP, LTP and LATP. SURF and Daisy are modified to be calculated at keypoints on a grid drawn across the image. This is against the traditional way of using them, where they are mostly required to find their own keypoints. The modified versions are called TSURF and TDaisy respectively. Other descriptors are also calculated on this grid. At smaller grid cell sizes TSURF performs much better than the other descriptors. In addition TSURF has one of the smallest feature vectors and is fast to train. LTP gave the best performance with larger grid cell sizes. Random Forest proved to be the best classifier in these experiments.

We also tested terrain classification based on color images. We computed two different types of image descriptors. One of them is the co-occurrence matrix and the other one is the SURF descriptor. We tested these descriptors on multiple colorspaces. Random Forest and Support Vector Machines were used for training and testing of the feature vectors. From the co-occurrence matrix, multiple features were extracted. The best performance was achieved with 12 features and 4 directions on the HSV colorspace on one dataset. With SURF, we also tested applying PCA to reduce the dimensionality and increase the learning speed. The best performance in case of SURF was with the YIQ colorspace on a dataset-1.

Hence, we have demonstrated that visual terrain classification can be performed at different resolutions using TSURF and LTP image descriptors on greyscale images. Furthermore, it is demonstrated that visual terrain classification can be successfully performed on an outdoor driving robot even in non optimal conditions, such as motion blur induced by a fast moving robot and its vibrating camera, different weather conditions, both wet and dry ground surfaces and a low camera viewpoint. On a flying robot, motion blur and scale changes also do not pose a big problem. We used images from real runs of the robot containing blurred images with non-sharp terrain boundaries.

Also on color images, terrain classification can be effectively applied on-board a wheeled robot. This also works when the images have artifacts due to motion and weather conditions.

Future work can focus on inclusion of additional terrain types and more statistical methods (Thrun, 2000; Thrun *et al.*, 2005). Outdoor mapping and localization based on terrain classification is also an interesting research direction along with using teams of robots to solve this task (Thrun, 2001). A mix of flying and driving robots can be used to first survey some area and then execute different tasks.

Appendix A

Further results of Greyscale Terrain Classification

A.1 Results of Other Classifiers on Wheeled Robot

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	46.7%	58.0%	44.3%	94.5%	62.2%	31.0%
20	64.4%	72.6%	62.3%	89.1%	57.3%	31.4%
30	74.1%	81.2%	71.9%	85.6%	55.0%	33.4%
40	80.1%	85.9%	77.9%	82.9%	53.5%	35.9%
50	84.1%	88.6%	81.5%	82.7%	52.3%	34.6%
60	87.2%	90.0%	84.7%	81.1%	51.8%	37.8%
70	88.3%	91.2%	86.9%	81.6%	51.0%	36.1%
80	90.4%	91.2%	88.0%	80.2%	48.7%	38.3%
90	90.0%	92.2%	89.4%	80.1%	50.7%	37.3%
100	90.7%	92.7%	90.4%	81.1%	50.4%	37.7%

Table A.1: Classification results of the six descriptors with C4.5/J48 classifier

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	58.5%	64.3%	55.4%	65.6%	62.4%	N/A
20	72.3%	62.6%	58.1%	65.9%	63.2%	N/A
30	65.2%	31.2%	64.9%	68.7%	71.5%	25.0%
40	41.5%	27.2%	64.8%	76.9%	69.9%	26.9%
50	34.2%	44.8%	60.6%	76.3%	68.4%	26.6%
60	42.3%	42.5%	53.9%	76.7%	66.9%	28.1%
70	39.4%	39.2%	47.8%	77.0%	67.0%	28.3%
80	31.4%	44.0%	38.4%	75.2%	65.1%	27.4%
90	35.6%	38.4%	39.2%	77.5%	65.9%	27.2%
100	27.3%	30.2%	30.6%	80.2%	65.0%	25.7%

Table A.2: Classification results of the six descriptors with SVM classifier

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	57.2%	65.8%	54.1%	59.1%	47.0%	25.2%
20	76.9%	82.7%	71.9%	60.3%	47.8%	25.7%
30	84.0%	86.9%	78.8%	62.4%	50.5%	25.9%
40	87.5%	90.1%	83.4%	63.2%	49.3%	27.9%
50	86.8%	93.3%	90.6%	64.9%	51.2%	26.7%
60	92.7%	95.8%	92.6%	67.6%	48.8%	29.5%
70	93.9%	96.7%	94.7%	69.3%	49.8%	29.1%
80	96.5%	97.9%	96.2%	70.8%	48.7%	30.7%
90	96.9%	98.3%	97.4%	73.8%	52.4%	28.2%
100	96.5%	98.3%	97.9%	73.1%	50.7%	30.0%

Table A.3: Classification results of the six descriptors with Linear SVM classifier

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	52.2%	61.1%	48.9%	44.4%	35.8%	22.1%
20	70.5%	73.0%	65.0%	45.1%	37.0%	24.7%
30	78.9%	80.6%	73.3%	46.1%	38.3%	26.4%
40	83.4%	84.1%	78.1%	50.2%	41.1%	27.5%
50	86.4%	86.3%	81.9%	50.6%	41.7%	26.8%
60	88.6%	88.2%	85.2%	51.7%	40.0%	28.3%
70	89.9%	88.8%	87.0%	52.7%	36.1%	28.4%
80	93.2%	89.8%	88.6%	54.9%	41.0%	28.2%
90	92.4%	90.0%	90.4%	56.1%	37.2%	30.2%
100	92.7%	90.6%	90.7%	64.0%	40.5%	28.3%

Table A.4: Classification results of the six descriptors with Naive Bayes classifier

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	43.3%	N/A	N/A	N/A	N/A	N/A
20	62.5%	71.8%	61.7%	96.6%	59.7%	N/A
30	75.5%	79.8%	74.3%	94.7%	54.5%	36.1%
40	83.2%	80.9%	82.2%	92.9%	51.1%	39.9%
50	88.0%	83.8%	86.4%	92.6%	49.5%	39.0%
60	88.1%	86.1%	90.2%	92.0%	49.3%	42.7%
70	74.8%	86.1%	93.0%	91.4%	48.0%	41.0%
80	72.7%	79.6%	94.3%	89.7%	46.0%	43.1%
90	44.1%	76.8%	95.1%	90.8%	47.0%	42.8%
100	31.5%	75.8%	95.7%	90.9%	45.8%	44.6%

Table A.5: Classification results of the six descriptors with K* classifier

Grid cell size	LBP	LTP	LATP	TSURF	TDaisy	CCH
10	38.0%	54.4%	41.2%	98.3%	68.6%	34.3%
20	51.6%	64.2%	68.7%	95.1%	60.4%	34.2%
30	66.7%	75.4%	78.2%	93.4%	56.2%	36.1%
40	78.8%	85.2%	84.5%	92.1%	53.8%	39.0%
50	85.2%	90.1%	87.6%	91.4%	52.0%	37.5%
60	90.2%	93.7%	90.6%	90.7%	51.3%	41.7%
70	92.2%	95.2%	93.3%	90.6%	51.1%	40.2%
80	94.6%	96.2%	94.6%	88.6%	49.2%	42.0%
90	95.5%	97.1%	96.0%	89.7%	50.0%	42.3%
100	96.5%	97.8%	97.3%	91.5%	48.7%	43.6%

Table A.6: Classification results of the six descriptors with k-NN classifier

A.2 Results of Other Classifiers on Flying Robot

Grid cell size	LBP	LTP	TSURF
10	29.9%	34.8%	89.7%
20	39.5%	43.6%	77.3%
30	46.9%	50.9%	67.2%
40	52.5%	56.1%	60.7%
50	56.6%	60.3%	56.8%
60	57.2%	62.7%	52.9%
70	61.5%	65.0%	52.7%
80	62.3%	64.9%	49.3%
90	62.4%	64.9%	50.4%
100	61.2%	66.7%	47.5%

Table A.7: Classification results of the descriptors with C4.5/J48 classifier

Grid cell size	LBP	LTP	TSURF
10	35.0%	N/A	42.9%
20	48.0%	54.7%	43.1%
30	50.8%	54.6%	43.8%
40	53.0%	65.7%	44.3%
50	54.7%	65.6%	46.5%
60	54.1%	68.1%	44.8%
70	60.0%	72.1%	48.7%
80	61.8%	77.3%	44.3%
90	63.8%	78.4%	50.1%
100	63.0%	78.0%	49.8%

Table A.8: Classification results of the descriptors with Linear SVM classifier

Grid cell size	LBP	LTP	TSURF
10	32.7%	35.7%	42.2%
20	44.4%	49.5%	42.5%
30	52.1%	56.7%	43.8%
40	57.1%	61.1%	43.0%
50	60.0%	64.1%	44.3%
60	62.1%	65.5%	42.1%
70	63.4%	67.0%	45.4%
80	65.4%	68.7%	42.4%
90	65.5%	68.9%	45.5%
100	67.2%	68.9%	52.3%

Table A.9: Classification results of the descriptors with Naive Bayes classifier

Grid cell size	LBP	LTP	TSURF
10	25.9%	N/A	N/A
20	36.1%	42.9%	93.8%
30	45.8%	54.7%	85.4%
40	54.2%	59.6%	72.6%
50	60.7%	65.5%	66.5%
60	66.2%	68.3%	54.0%
70	65.2%	69.8%	56.3%
80	56.6%	68.6%	49.3%
90	40.8%	71.8%	51.1%
100	29.1%	70.6%	50.0%

Table A.10: Classification results of the descriptors with K*-NN classifier

Grid cell size	LBP	LTP	TSURF
10	23.4%	33.8%	96.9%
20	29.0%	47.9%	90.8%
30	39.5%	58.9%	83.3%
40	50.8%	66.1%	72.3%
50	61.7%	74.0%	72.4%
60	68.2%	78.2%	61.9%
70	70.8%	80.1%	62.9%
80	75.4%	80.6%	60.2%
90	75.9%	83.5%	59.0%
100	78.0%	84.0%	57.8%

Table A.11: Classification results of the descriptors with k-NN classifier

Abbreviations

API	Application programming interface
CCH	Contrast Context Histogram
FNR	False Negative Rate/Ratio
k-NN	K Nearest Neighbor
K*-NN	K* Nearest Neighbor
MLP	Multi-layer Perceptron (Artificial Neural Network)
LATP	Local Adaptive Ternary Pattern
LBP	Local Binary Pattern
LTP	Local Ternary Pattern
NN	Neural Network
PCA	Principle Component Analysis
RDT	Random Decision Trees
RF	Random Forest
RGB	Red-Green-Blue values (channels) in a color image
SIFT	Scale Invariant Features
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
TDaisy	Terrain Daisy algorithm
TPR	True Positive Rate/Ratio
TSURF	Terrain SURF algorithm

List of Tables

List of Figures

Bibliography

Abdel-Hakim, A. and Farag, A. (2006). Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978 – 1983.

Akhloufi, M. and Bendada, A. H. (2010). Locally adaptive texture features for multi-spectral face recognition. In *IEEE International Conference on Systems, Man, and Cybernetics*, Istanbul, Turkey. IEEE.

Allied Vision Technologies, G. (2012). Marlin F-046C color camera. Websie. `http://www.alliedvisiontec.com/us/products/cameras/firewire/marlin/f-046bc.html`.

Andersen, J. C., Blas, M. R., Andersen, N., Ravn, O., and Blanke, M. (2006). Traversable terrain classification for outdoor autonomous robots using single 2D laser scans. *Integrated Computer-Aided Engineering*, **13**(3), 223–232.

Andreasson, H., Triebel, R., and Burgard, W. (2005). Improving plane extraction from 3D data by fusing laser data and vision. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, Alberta, Canada. IEEE.

Angelova, A., Matthies, L., Helmick, D. M., and Perona, P. (2007a). Fast terrain classification using variable-length representation for autonomous navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA.

Angelova, A., Matthies, L., Helmick, D., and Perona, P. (2007b). Learning and prediction of slip from visual information: Research articles. *Journal of Field Robotics*, **24**(3), 205–231.

Artač, M. and Leonardis, A. (2004). Outdoor mobile robot localisation using global and local features. In D. Skočaj, editor, *Proc. of the 9th Computer Vision Winter Workshop (CVWW)*, pages 175–184, Piran. Slovenian Pattern Recognition Society.

Artač, M., Jogan, M., Bakstein, H., and Leonardis, A. (2005). Panoramic volumes for robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3776–3782, Edmonton, Alberta, Canada.

Bailey, T. (2002). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Ph.D. thesis, University of Sydney.

Bajracharya, M., Benyang, T., Howard, A., Turmon, M., and Matthies, L. (2008). Learning long-range terrain classification for autonomous navigation. In *IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008)*, pages 4018–4024, Pasadena, CA.

Barfoot, T. D. (2005). Online visual motion estimation using fastslam with sift features. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3076 – 3082, Edmonton, Canada.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV 2006)*, pages 404–417, Graz, Austria.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, **110**(3), 346 – 359. Similarity Matching in Computer Vision and Multimedia.

Biber, P., Fleck, S., Wand, M., Staneker, D., and Straßer, W. (2005). First experiences with a mobile platform for flexible 3D model acquisition in indoor and outdoor environments – the wägele. In *3D-ARCH'2005: 3D Virtual Reconstruction and Visualization of Complex Architectures*, Mestre-Venice, Italy.

Birk, A., Pathak, K., Poppinga, J., Schwertfeger, S., and Chonnaparamutt, W. (2007). Intelligent behaviors in outdoor environments. In *Proceedings of the ISASTED International Conference on Robotics and Applications (RA 2007)*, Würzburg, Germany.

Birk, A., Stoyanov, T., Nevatia, Y., Ambrus, R., Poppinga, J., and Pathak, K. (2008). Terrain classification for autonomous robot mobility: from safety, security rescue robotics to planetary exploration. In *IEEE International Conference on Robotics and Automation (ICRA), Planetary Rovers Workshop*, pages 1–5.

Bishop, C. M. (1994). Novelty detection and neural network validation. *IEE Proceedings: Vision, Image and Signal Processing*, **141**(4), 217–222.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blas, M., Agrawal, M., Sundaresan, A., and Konolige, K. (2008). Fast color/texture segmentation for outdoor robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 4078 –4085.

Bohlmann, K., Biber, P., and Zell, A. (2009). Using geographical data and sonar to improve GPS localization for mobile robots. In *Proceedings of the 4th European Conference on Mobile Robots (ECMR 2009)*, pages 55–60, Mlini/Dubrovnik, Croatia.

Bohlmann, K., Marks, H., and Zell, A. (2012a). Automated odometry self-calibration for car-like robots with four-wheel-steering. In *ICINCO International Conference on Informatics in Control, Automation and Robotics*, Rome, Italy. Accepted for publication.

Bohlmann, K., Beck-Greinwald, A., Buck, S., Marks, H., and Zell, A. (2012b). Autonomous person following with 3d lidar in outdoor environments. In *1st International Workshop on Perception for Mobile Robots Autonomy (PEMRA 2012)*, Poznan, Poland.

Booij, O., Terwijn, B., Zivkovic, Z., and B., K. (2007). Navigation using an appearance based topological map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 3927–3932, Rome, Italy.

Boser, Bernhard E., I. M. G. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT 92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA. ACM Press.

Bradley, D. M., Patel, R., Vandapel, N., and Thayer, S. M. (2005). Real-time image-based topological localization in large outdoor environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3062 – 3069, Edmonton, Canada.

Bradley, D. M., Unnikrishnan, R., and Bagnell, J. (2007). Vegetation detection for driving in complex environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 503 – 508, Rome, Italy.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Braun, T., Bitsch, H., and Berns, K. (2008). Visual terrain traversability estimation using a combined slope/elevation model. In A. R. Dengel, K. Berns, T. Breuel, F. Bomarius, and T. R. Roth-Berghofer, editors, *KI 2008: Advances in Artificial Intelligence*, volume 5243, pages 177–184, Kaiserslautern, Germany. Springer Berlin / Heidelberg.

Breiman, L. (1996). Bagging predictors. In *Machine Learning*, volume 24, pages 123–140, Hingham, MA, USA. Kluwer Academic Publishers.

Breiman, L. (2001). Random forests. In *Machine Learning*, volume 45, pages 5–32, Hingham, MA, USA. Kluwer Academic Publishers.

Brodatz, P. (1966). Textures: A photographic album for artists & designers. New York: Dover, New York, NY.

Brooks, C. A. and Iagnemma, K. (2007). Self-supervised classification for planetary rover terrain sensing. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, USA.

Brooks, C. A. and Iagnemma, K. (2009). Visual detection of novel terrain via two-class classification. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 1145–1150, New York, NY, USA. ACM.

Brooks, C. A., Iagnemma, K., and Dubowsky, S. (2005). Vibration-based terrain analysis for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3426 – 3431, Barcelona, Spain.

Brooks, C. A., Iagnemma, K., and Dubowsky, S. (2006). Visual wheel sinkage measurement for planetary rover mobility characterization. *Autonomous Robots*, **21**(1), 55 – 64.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**, 121–167.

Castano, A. and Matthies, L. (2003). Foliage discrimination using a rotating ladar. In *Proccedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 1–6, Taipei, Taiwan.

Castano, R., Manduchi, R., and Fox, J. (2001). Classification experiments on real-world textures. In *Proceedings of the Workshop on Empirical Evaluation in Computer Vision*, Kauai, HI.

Castelnovi, M., Arkin, R. C., and Collins, T. R. (2005). Reactive speed control system based on terrain roughness detection. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 891–896, Barcelona, Spain.

Chang, C.-C. and Lin, C.-J. (2005). *LIBSVM: a Library for Support Vector Machines*. Department of Computer Science and Information Engineering, National Taiwan University, Taipeh 106, Taiwan.

Chang, W.-C. and Lee, S.-A. (2005). Feature-based stereo visual guidance and control of mobile robots for autonomous hallway following tasks. In *36th International Symposium on Robotics (ISR 2005)*, Tokyo, Japan.

Chatzis, S. and Tsechpenakis, G. (2009). The infinite hidden Markov random field model. In *IEEE International Conference on Computer Vision (ICCV 2009)*, pages 654–661, Kyoto, Japan.

Cleary, J. G. and Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. *MACHINE LEARNINGINTERNATIONAL WORKSHOP THEN CONFERENCE*, **5**, 1–14.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**, 273–297. 10.1007/BF00994018.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, **13**(1), 21 –27.

Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., and Bradski, G. (2006). Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems (RSS 2006)*, Philadelphia, PA, USA.

Davidson, J. C. and Hutchinson, S. A. (2003). Recognition of traversable areas for mobile robotic navigation in outdoor environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 297 – 304, Las Vegas, USA.

Davis, I., Kelly, A., Stentz, A., and Matthies, L. (1995). Terrain typing for real robots. In *Proceedings of the Intelligent Vehicles '95 Symposium*, pages 400–405, Detroit, MI.

Deselaers, T. (2003). *Features for Image Retrieval*. Master's thesis, Lehrstuhl für Informatik VI, Rheinisch-Westfälische Technische Hochschule Aachen.

Dima, C., Vandapel, N., and Hebert, M. (2004). Classifier fusion for outdoor obstacle detection. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 665 – 671, New Orleans, LA, USA.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley.

DuPont, E. M., Moore, C. A., Selekwa, M., Collins Jr., E. G., and Roberts, R. G. (2005a). Online terrain classification for mobile robots. In *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, Orlando, Florida, USA.

DuPont, E. M., Moore, C., Roberts, R. G., and Collins Jr., E. G. (2005b). Terrain classification using probabilistic neural networks. In *Proceedings of the Florida Conference on Recent Advances in Robotics*, Gainesville, Florida, USA.

DuPont, E. M., Roberts, R. G., and Moore, C. A. (2006). The identification of terrains for mobile robots using eigenspace and neural network methods. In *Proceedings of the Florida Conference on Recent Advances in Robotics (FCRAR 2006)*, Miami, FL, USA.

DuPont, E. M., Moore, C. A., Collins Jr., E. G., and Coyle, E. (2008). Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, **24**(4), 337–347.

Eck, D., Stahl, M., and Schilling, K. (2007). The small outdoor rover MERLIN and its assistance system for tele-operations. In *Proceedings of the International Conference on Field and Service Robotics (FSR 2007)*, Chamonix, France.

Erästö, P. (2001). *Support Vector Machines - Backgrounds and Practice*. Master's thesis, Rolf Nevanlinna Institute, Helsinki, Finland.

Erhard, S., Wenzel, K. E., and Zell, A. (2009a). Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter. In *Proceedings of UAV'09 2nd International Symposium on Unmanned Aerial Vehicles*, pages 451–465, Reno, Nevada, USA.

Erhard, S., Wenzel, K. E., and Zell, A. (2009b). Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter. *Journal of Intelligent & Robotic Systems*, **57**, 451–465.

Erkan, A., Hadsell, R., Sermanet, P., Ben, J., Muller, U., and LeCun, Y. (2007). Adaptive long range vision in unstructured terrain. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 2421 – 2426, San Diego, CA, USA.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, **27**, 861–874.

Flach, P. (2003). The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In *Proc. of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC.

Föhst, T., Gava, C., Arndt, M., Berns, K., and Vassallo, R. (2010). Off-road place recognition using fused image features. In *Proceedings for the joint conference of ISR 2010 and ROBOTIK 2010*, pages 151–156, Munich, Germany. VDI Düsseldorf.

Früh, C. and Zakhor, A. (2001). 3D model generation for cities using aerial photographs and ground level laser scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2.2, pages II–31–38, Kauai, USA.

Früh, C. and Zakhor, A. (2002). Data processing algorithms for generating textured 3D building faade meshes from laser scans and camera images. In *Proc. 3D Data Processing, Visualization and Transmission*, pages 834–847, Padua, Italy.

Früh, C. and Zakhor, A. (2003). Constructing 3D city models by merging ground-based and airborne views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II–5 62 – 69, Madison, USA.

Gao, H., Siu, W.-C., and Hou, C.-H. (2001). Improved techniques for automatic image segmentation. *Circuits and Systems for Video Technology, IEEE Transactions on*, **11**(12), 1273 –1280.

Geman, S. and Geman, D. (1987). *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Giguere, P. and Dudek, G. (2008). Clustering sensor data for terrain identification using a windowless algorithm. In *Proceedings of Robotics Science and System (RSS)*, pages 25–32, Zürich, Switzerland.

Gini, C. (1913). Variabilita e mutabilita. *Journal of the Royal Statistical Society*, **76**(3), 326–327.

Hähnel, D., Burgard, W., and Thrun, S. (2003). Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, **44**(1), 15–27.

Halawani, A. and Burkhardt, H. (2004). Image retrieval by local evaluation of nonlinear kernel functions around salient point. In *Proc. of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 955–960, Cambridge, UK.

Halawani, A. and Burkhardt, H. (2005). On using histograms of local invariant features for image retrieval. In *Proceedings of the IAPR Workshop on Machine Vision Applications (MVA 2005)*, pages 538–541, Tsukuba Science City, Japan.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, **Volume 11, Issue 1**.

Hao, P. and Shi, Q. (2000). Comparative study of color transforms for image coding and derivation of integer reversible color transform. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 224 –227 vol.3.

Haralick, R., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, **3**(6), 610–621.

Hebert, M. and Vandapel, N. (2003). Terrain classification techniques from ladar data for autonomous navigation. In *Collaborative Technology Alliances conference*.

Hinselmann, G., Jahn, A., Fechner, N., Rosenbaum, L., and Zell, A. (2011a). Approximation of graph kernel similarities for chemical graphs by kernel principal component analysis. In *LNCS 6623 (EvoBio 2011), Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics: 9th European Conference*, pages 123–134. Springer.

Hinselmann, G., Rosenbaum, L., Jahn, A., Fechner, N., Ostermann, C., and Zell, A. (2011b). Large-scale learning of structure-activity relationships using a linear support vector machine and problem-specific metrics. *Journal of Chemical Information and Modeling*, **52**, 203–213.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan.

Huang, C.-R., Chen, C.-S., and Chung, P.-C. (2006). Contrast context histogram - a discriminating local descriptor for image matching. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 53–56, Washington, DC, USA. IEEE Computer Society.

Huang, C.-R., Chen, C.-S., and Chung, P.-C. (2008). Contrast context histogram-an efficient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, **41**(10), 3071–3077.

Hudjakov, R. and Tamre, M. (2011). Ortophoto analysis for ugv long-range autonomous navigation. *Estonian Journal of Engineering*, **17**(1), 17–27.

Iagnemma, K. and Dubowsky, S. (2002). Terrain estimation for high-speed rough-terrain autonomous vehicle navigation. In *Proceedings of the SPIE Conference on Unmanned Ground Vehicle Technology IV*, Orlando, FL, USA.

Iagnemma, K. and Dubowsky, S. (2004). *Mobile Robots in Rough Terrain - Estimation, Motion Planning, and Control with Application to Planetary Rovers*. Springer, Berlin.

Iagnemma, K., Kang, S., Shibly, H., and Dubowsky, S. (2004). Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *IEEE Transactions on Robotics*, **20**(5), 921 – 927.

Iocchi, L., Konolige, K., and Bajracharya, M. (2000). Visually realistic mapping of a planar environment with stereo. In *International Symposium on Experimental Robotics (ISER)*.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML 1998)*, pages 137–142, Chemnitz, Germany.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer, New York.

Karlsen, R. E. and Witus, G. (2007). Terrain understanding for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 895 – 900, San Diego, CA, USA.

Kelly, A., Nagy, B., Stager, D., and Unnikrishnan, R. (2007). An infrastrucure-free automated guided vehicle based on computer vision. *IEEE Robotics and Automation Magazine*, **14**(3), 24 – 34.

Khan, Y. N. and Mehdi, S. A. (2002). Sign language recognition using sensor gloves. In *9th International Conference on Neural Information Processing*, volume 5, pages 2204 – 2206.

Khan, Y. N., Komma, P., Bohlmann, K., and Zell, A. (2011a). Grid-based visual terrain classification for outdoor robots using local features. In *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS 2011)*, pages 16 – 22, Paris, France.

Khan, Y. N., Komma, P., and Zell, A. (2011b). High resolution visual terrain classification for outdoor robots. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1014 –1021, Barcelona, Spain.

Khan, Y. N., Masselli, A., and Zell, A. (2012). Visual terrain classification by flying robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 498 –503, St. Paul, Minnesota, USA.

Kim, D., Sun, S., Oh, S., Rehg, J., and Bobick, A. (2006). Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006)*, pages 518–525, Orlando, FL, USA.

Kleiner, A. and Dornhege, C. (2007). Real-time localization and elevation mapping within urban search and rescue scenarios. *Journal of Field Robotics*, **24**(8-9), 723–745.

Komma, P. and Zell, A. (2009). Posterior probability estimation techniques embedded in a Bayes filter for vibration-based terrain classification. In *7th International Conference on Field and Service Robots (FSR 2009)*, pages 1–10, MIT, Cambridge, Massachusetts, USA.

Komma, P. and Zell, A. (2010a). Clustering vibration data using a temporally coherent Expectation Maximization approach. In *7th Symposium on Intelligent Autonomous Vehicles (IAV 2010)*, pages 1–6, Lecce, Italy.

Komma, P. and Zell, A. (2010b). Markov random field-based clustering of vibration data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 1902–1908, Taipei, Taiwan. Best Paper Award Nominee.

Komma, P. and Zell, A. (2010c). Posterior probability estimation techniques embedded in a bayes filter for vibration-based terrain classification. *Springer Tracts in Advanced Robotics*, **62**, 79–89.

Komma, P., Weiss, C., and Zell, A. (2009a). Adaptive Bayesian filtering for vibration-based terrain classification. In *IEEE International Conference on Robotics and Automation (ICRA 2009), Kobe, Japan*, pages 3307–3313.

Komma, P., Weiss, C., and Zell, A. (2009b). Improved vibration based terrain classification using temporal coherence. In *40th International Symposium on Robotics (ISR)*, pages 359–364, Barcelona, Spain.

Konolige, K. (2000). A gradient method for realtime robot control. In *International Conference on Intelligent Robots and Systems (IROS)*.

Kosecka, J. and Li, F. (2004). Vision based topological markov localization. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1481–1486, New Orleans, LA, USA.

Kramer, J. and Scheutz, M. (2007). Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, **22**(2), 101 – 132.

Kronfeld, M., Weiss, C., and Zell, A. (2010). Swarm-supported outdoor localization with sparse visual data. *Robotics and Autonomous Systems. Selected papers from the 2007 European Conference on Mobile Robots (ECMR 2007)*, **58**(2), 166–173.

Laible, S., Khan, Y. N., Bohlmann, K., and Zell, A. (2012). 3d lidar- and camera-based terrain classification under different lighting conditions. In K. H. Paul Levi, Oliver Zweigle and B. Eckstein, editors, *Autonomous Mobile Systems 2012*, Informatik aktuell, pages 21–29. Springer Berlin Heidelberg.

Lalonde, J.-F., Vandapel, N., Huber, D. F., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, **23**(10), 839–861.

Lamon, P. and Siegwart, R. (2005). Wheel torque control in rough terrain - modeling and simulation. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA 2005)*, pages 867 – 872, Barcelona, Spain.

Lamon, P., Kolski, S., and Siegwart, R. (2006). The SmartTer - a vehicle for fully autonomous navigation and mapping in outdoor environments. In *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR 2006)*, Brussels, Belgium.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 1465–1479.

Liao, S., Zhu, X., Lei, Z., Zhang, L., and Li, S. (2007). Learning multi-scale block local binary patterns for face recognition. In S.-W. Lee and S. Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, pages 828–837. Springer Berlin / Heidelberg.

Lindbloom, B. J. (2001). Rgb working space information. http://www.brucelindbloom.com/index.html?WorkingSpaceInfo.html.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**(2), 91–110.

Macedo, J., Matthies, L., and Manduchi, R. (2000). Ladar-based discrimination of grass from obstacles for autonomous navigation. In *Proceedings of the International Symposium on Experimental Robotics (ISER 2000)*, pages 111 – 120, Hawaii, USA.

Manduchi, R. (2006). Learning outdoor color classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(11), 1713 – 1723.

Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2005). Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, **18**, 81–102.

Manjunath, B. and Chellappa, R. (1991). Unsupervised texture segmentation using markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(5), 478–482.

Markou, M. and Singh, S. (2003a). Novelty detection: a reviewpart 1: statistical approaches. *Signal Processing*, **83**(12), 2481 – 2497.

Markou, M. and Singh, S. (2003b). Novelty detection: a reviewpart 2: neural network based approaches. *Signal Processing*, **83**(12), 2499 – 2521.

Martinez-Alajarin, J., Luis-Delgado, J., and Tomas-Balibrea, L. (2005). Automatic system for quality-based classification of marble textures. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **35**(4), 488 – 497.

Masselli, A. and Zell, A. (2012). A Novel Marker Based Tracking Method for Position and Attitude Control of MAVs. In *Proceedings of International Micro Air Vehicle Conference and Flight Competition*, pages 1–6, Braunschweig, Germany. DGON.

Matthies, L. and Rankin, A. (2003). Negative obstacle detection by thermal signature. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 906– 913, Las Vegas, Nevada, USA.

Matthies, L., Kelly, A., Litwin, T., and Tharp, G. (1995). Obstacle detection for unmanned vehicles: A progress report. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 1995)*, pages 66–71, Detroit, MI, USA.

Mehdi, S. A., Armbrust, C., Koch, J., and Berns, K. (2009). Methodology for robot mapping and navigation in assisted living environments. In *PETRA '09: Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, number ISBN: 978-1-60558-409-6, Corfu, Greece. ACM, New York, NY, USA.

Mehdi, S. A., Wettach, J., and Berns, K. (2011). A simulated environment for elderly care robot. In *Pervasive and Embedded Computing and Communication Systems (PECCS)*, pages 562–567.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **27**(10), 1615 – 1630.

Milella, A., Reina, G., and Siegwart, R. (2006). Computer vision methods for improved mobile robot state estimation in challenging terrains. *Journal of Multimedia*, **1**(7), 49–61.

Mitchell, T. T. (1997). *Machine Learning*. McGraw-Hill.

Mittag, F., Büchel, F., Saad, M., Jahn, A., Schulte, C., Bochdanovits, Z., Simón-Sánchez, J., Nalls, M. A., Keller, M., Hernandez, D., Gibbs, R., Lesage, S., Brice, A., Heutink, P., Martinez, M., Wood, N. W., Hardy, J., Singleton, A. B., Zell, A., Gasser, T., and Sharma, M. (2012). Use of support vector machines for disease risk prediction in genome-wide association studies: Concerns and opportunities. *Human Mutation*. Accepted for publication.

Nabney, I. T. (2002). *NETLAB: Algorithms for Pattern Recognition*. Springer.

Nüchter, A., Lingemann, K., Hertzberg, J., Surmann, H., Pervlz, K., Hennig, M., Tiruchinapalli, K. R., Worst, R., and Christaller, T. (2005). Mapping of rescue environments with Kurt3D. In *Proceedings of the International Workshop on Safety, Security and Rescue Robotics (SSRR 2005)*, pages 158 – 163, Kobe, Japan.

Ojala, T. and Pietikaeinen, M. (1997). Unsupervised texture segmentation using feature distributions. *Lecture Notes in Computer Science*, **1310**, 311–??

Ojala, T., Pietikainen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, **29**(1), 51–59.

Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis Machine Intelligence*, **24**(7), 971–987.

Ollis, M., Huang, W. H., and Happold, M. (2007). A bayesian approach to imitation learning for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 709 – 714, San Diego, CA, USA.

Olmos, A. and Kingdom, F. (2004). Mcgill calibrated colour image database.

Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: an application to face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, pages 130 – 136, San Juan, Puerto Rico.

Otte, M. W., Richardson, S. G., Mulligan, J., and Grudic, G. (2007). Local path planning in image space for autonomous robot navigation in unstructured environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 2819 – 2826, San Diego, CA, USA.

Ozuysal, M., Fua, P., and Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Pesaresi, M., Gerhardinger, A., and Kayitakire, F. (2008). A robust built-up area presence index by anisotropic rotation-invariant textural measure. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, **1**(3), 180 –192.

Platt, J. (2000). *Advances in Large-Margin Classifiers*, chapter Probabilities for SV Machines, pages 61–74. MIT Press, Cambridge, MA, USA.

Platt, J. C. (1999). Using analytic qp and sparseness to speed training of support vector machines. In *Neural Information Processing Systems 11*, pages 557–563. MIT Press.

Point-Grey Research, I. (2012). Firefly color camera. website. `http://www.ptgrey.com/products/fireflymv/fireflymv_usb_firewire_cmos_camera.asp`.

Poppinga, J., Birk, A., and Pathak, K. (2008). Hough based terrain classification for realtime detection of drivable ground. *Journal of Field Robotics*, **25**(1-2), 67–88.

Procipio, M. J., Mulligan, J., and Grudic, G. (2007). Long-term learning using multiple models for outdoor autonomous robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 3158 – 3165, San Diego, CA, USA.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA.

Quinlan, J. R. (1996). Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, **4**, 77–90.

Rajadell Rojas, Olga; Garca Sevilla, P. (2008). Influence of color spaces over texture characterization. *Research in Computing Science*, **38**, 273–281.

Rao, N., Kareti, S., Shi, W., and Iyenagar, S. (1993). Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, Oak Ridge (Tennessee).

Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Röfer, T., Burkhard, H.-D., Düffert, U., Hoffmann, J., Göhring, D., Jüngel, M., Lötzsch, M., v. Stryk, O., Brunn, R., Kallnik, M., Kunz, M., Petters, S., Risler, M., Stelzer, M., Dahm, I., Wachter, M., and Engel, K. (2003). Germanteam robocup 2003. Technical report.

Roth, G. and Wibowo, E. (1996). A fast algorithm for making mesh models from multiple-view range data. In *DND/CSA Robotics and Knowledge Based Systems Workshop*, pages 120–128, St.Hubert, Qubec, Canada.

Sadhukhan, D. and Moore, C. (2003). Online terrain estimation using internal sensors. In *Proceedings of the Florida Conference on Recent Advances in Robotics (FCRAR 2003)*, Boca Raton, FL, USA.

Schäfer, H., Proetzsch, M., and Berns, K. (2005). Stereo-vision-based obstacle avoidance in rough outdoor terrain. In *Proceedings of the International Symposium on Motor Control and Robotics*.

Schäfer, H., Proetzsch, M., Braun, T., Koch, J., Schmitz, N., and Berns, K. (2006). RAVON - a robust autonomous vehicle for off-road navigation. In *European Land Robot Trial (ELROB 2006)*, Hammelburg, Germany.

Schauwecker, K., Ke, N. R., Scherer, S. A., and Zell, A. (2012). Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing. In *22nd Conference on Autonomous Mobile Systems (AMS)*. Springer. Accepted for publication.

Scherer, S. A., Dube, D., Komma, P., Masselli, A., and Zell, A. (2011). Robust real-time number sign detection on a mobile outdoor robot. In *Proceedings of the 6th European Conference on Mobile Robots (ECMR 2011)*, Örebro, Sweden.

Schölkopf, B., Smola, A., Williamson, R., and Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, **12**, 1207–1245.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (2000). Support vector method for novelty detection. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 582–588.

Schwarz, M. W., Cowan, W. B., and Beatty, J. C. (1987). An experimental comparison of rgb, yiq, lab, hsv, and opponent color models. *ACM Trans. Graph.*, **6**(2), 123–158.

Siggelkow, S. and Burkhardt, H. (2002). Improvement of histogram-based image retrieval and classification. In *Proceedings of the International Conference of Pattern Recognition (ICPR)*, volume 3, pages 367–370, Quebec, Canada.

Smith, L. I. (2002). *A Tutorial on Principal Component Analysis*.

Sofman, B., Bagnell, J. A. D., Stentz, A., and Vandapel, N. (2006). Terrain classification from aerial data to support ground vehicle navigation. Technical Report CMU-RI-TR-05-39, Robotics Institute, Pittsburgh, PA.

Soh, L.-K. and Tsatsoulis, C. (1999). Texture analysis of sar sea ice imagery using gray level co-occurrence matrices. *Geoscience and Remote Sensing, IEEE Transactions on*, **37**(2), 780 –795.

Specht, D. (1990). Probabilistic neural networks. *Neural Networks*, **3**(1), 109–118.

Spero, D. J. (2004). A review of outdoor robotics research. Technical Report MECSE-17-2004, Monash University, Department of Electrical and Computer Systems Engineering, Victoria, Australia.

Stamos, I. and Allen, P. K. (2000). Integration of range and image sensing for photorealistic 3D modeling. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1435–1440, San Francisco, USA.

Stavens, D. and Thrun, S. (2006). A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Boston, USA.

Stavens, D., Hoffmann, G., and Thrun, S. (2007). Online speed adaptation using supervised learning for high-speed, off-road autonomous driving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2218–2224, Hyderabad, India.

Sun, J., Metha, T., Wooden, D., Powers, M., Rehg, J., Balch, T., and Egerstedt, M. (2006). Learning from examples in unstructured outdoor environments. *Journal of Field Robotics*, **23**(11/12), 1019–1036.

Süsstrunk, S., Buckley, R., and Swen, S. (1999). Standard rgb color spaces. In *In The Seventh Color Imaging Conference: Color Science, Systems, and Applications*, pages 127–134.

Tan, X. and Triggs, B. (2007). Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *Proceedings of the 3rd international conference on Analysis and modeling of faces and gestures (AMFG 07)*, pages 168–182, Berlin, Heidelberg. Springer-Verlag.

Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, **19**(6), 1635 –1650.

Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine*, **21**(4), 93–109.

Thrun, S. (2001). A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, **20**(5), 335–363.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press.

Thrun, S., Montemerlo, M., and Aron, A. (2006a). Probabilistic terrain analysis for high-speed desert driving. In *Proceedings of Robotics: Science and Systems (RSS 2006)*, Philadelphia, PA, USA.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006b). Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, **23**(9), 661 – 692.

Tola, E., Lepetit, V., and Fua, P. (2010). Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 815–830.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839 –846.

Ulrich, I. and Nourbakhsh, I. R. (2000). Appearance-based obstacle detection with monocular color vision. In *Proceedings of the National Conference on Artificial Intelligence*, pages 866 – 871, Austin, TX, USA.

Vadakkepat, P., Lim, P., De Silva, L., Jing, L., and Ling, L. L. (2008). Multimodal approach to human-face detection and tracking. *Industrial Electronics, IEEE Transactions on*, **55**(3), 1385 –1393.

Vadivel, A., Sural, S., and Majumdar, A. K. (2007). An integrated color and intensity co-occurrence matrix. *Pattern Recognition Letters*, **28**(8), 974–983.

van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2008). Evaluation of color descriptors for object and scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, USA.

Vandapel, N., Huber, D., Kapuria, A., and Hebert, M. (2004). Natural terrain classification using 3-d ladar data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 5117–5122, New Orleans, LA.

Vapnik, V. and Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, **24**, 774780.

Varma, M. and Zisserman, A. (2005). A statistical approach to texture classification from single images. *International Journal of Computer Vision*, **62**(1-2), 61–81.

Vernaza, P., Taskar, B., and Lee, D. (2008). Online, self-supervised terrain classification via discriminatively trained submodular Markov random fields. In *IEEE International Conference on Robotics and Automation (ICRA 2008)*, pages 2750–2757.

Vo-Duc, M., Masselli, A., and Zell, A. (2012). Real time face detection using geometric constraints, navigation and depth-based skin segmentation on mobile robots. In *2012 IEEE International Symposium on Robotic and Sensors Environments, Accepted for publication.*, Magdeburg, Germany.

Weiss, C. and Zell, A. (2005). Automatic generation of indoor VR-models by a mobile robot with a laser range finder and a color camera. In *Autonome Mobile Systeme (AMS)*, pages 107–113, Stuttgart, Germany.

Weiss, C. and Zell, A. (2008). Novelty detection and online learning for vibration-based terrain classification. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS 2008)*, Baden-Baden, Germany. to appear.

Weiss, C., Fechner, N., Stark, M., and Zell, A. (2007a). Comparison of different approaches to vibration-based terrain classification. In *Proceedings of the 3rd European Conference on Mobile Robots (ECMR 2007)*, pages 7–12, Freiburg, Germany.

Weiss, C., Masselli, A., and Zell, A. (2007b). Fast vision-based localization for outdoor robots using a combination of global image features. In *Proceedings of the 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2007)*, Toulouse, France.

Weiss, C., Tamimi, H., Masselli, A., and Zell, A. (2007c). A hybrid approach for vision-based outdoor robot localization using global and local image features. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, San Diego, CA, USA.

Weiss, C., Stark, M., and Zell, A. (2007d). SVMs for vibration-based terrain classification. In *Proceedings of Autonome Mobile Systeme (AMS 2007)*, pages 1–7, Kaiserslautern, Germany.

Weiss, C., Tamimi, H., and Zell, A. (2008). A combination of vision and vibration-based terrain classification. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, Nice, France. to appear.

Weiss, U., Biber, P., Laible, S., Bohlmann, K., and Zell, A. (2010). Plant species classification using a 3d lidar sensor and machine learning. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 339–345, Washington, D.C., USA. IEEE Computer Society, IEEE.

Wenzel, K. E., Rosset, P., and Zell, A. (2009). Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller. In *Proceedings of UAV'09 2nd International Symposium on Unmanned Aerial Vehicles*, pages 1–15, Reno, USA. Kimon P. Valavanis.

Wenzel, K. E., Masselli, A., and Zell, A. (2010a). A Quadrocopter Hovering above a Person Wearing a Modified Cap. In *Proceedings of International Micro Air Vehicle Conference and Flight Competition*, pages 1–7, Braunschweig, Germany. DGON.

Wenzel, K. E., Masselli, A., and Zell, A. (2010b). Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. In *Proceedings of UAV'10 3rd International Symposium on Unmanned Aerial Vehicles*, pages 1–18, Dubai, UAE. Kimon P. Valavanis.

Wenzel, K. E., Masselli, A., and Zell, A. (2010c). Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle. *Journal of Intelligent & Robotic Systems*, **61**, 221–238.

Wenzel, K. E., Masselli, A., and Zell, A. (2012). Visual tracking and following of a quadrocopter by another quadrocopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Algarve, Portugal. IEEE.

Weston, J. and Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN 1999)*, pages 219–224, Bruges, Belgium.

Wilcox, B. H. (1994). Non-geometric hazard detection for a mars microrover. In *Proceedings of the NASA Conference on Intelligent Robots in Field, Factory, Service and Space*, pages 675–684, Houston, TX.

Wilhelm Burger, M. J. B. (2005). *Digitale Bildverarbeitung: Eine Einfuerung Mit Java und Image*. Springer DE.

Wilking, D. and Röfer, T. (2005). Realtime object recognition using decision tree learning. In *RoboCup 2004: RobotWorld Cup VII*, pages 556–563, Heidelberg. Springer.

Winder, S., Hua, G., and Brown, M. (2009). Picking the best daisy. In *Computer Vision and Pattern Recognition, CVPR 2009. IEEE Conference on*, pages 178–185. IEEE.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, 3 edition.

Wolf, D., Sukhatme, G. S., Fox, D., and Burgard, W. (2005a). Autonomous terrain mapping and classification using hidden markov models. In *IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 2038–2043, Barcelona, Spain.

Wolf, J., Burgard, W., and Burkhardt, H. (2002a). Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*.

Wolf, J., Burgard, W., and Burkhardt, H. (2002b). Using an image retrieval system for vision-based mobile robot localization. In *Proc. of the International Conference on Image and Video Retrieval (CIVR)*.

Wolf, J., Burgard, W., and Burkhardt, H. (2005b). Robust vision-based localization by combining an image retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, **21**(2), 208–216.

Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, **5**, 975–1005.

Yang, L., Meer, P., and Foran, D. (2005). Unsupervised segmentation based on robust estimation and color active contour models. *Information Technology in Biomedicine, IEEE Transactions on*, **9**(3), 475 –486.

Yang, S., Scherer, S. A., and Zell, A. (2012). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. In *2012 International Conference on Unmanned Aircraft Systems (ICUAS'12)*, Philadelphia, PA, USA.

Zhang, H. (2004). The optimality of naive bayes. In V. Barr and Z. Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS*, Miami Beach, Florida, USA. AAAI Press.

Zhang, H. and Su, J. (2004). Naive bayesian classifiers for ranking. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *15th European Conference on Machine Learning, Proceedings*, volume 3201 of *Lecture Notes in Computer Science*, pages 501–512, Pisa, Italy. Springer.

Zhang, H., Berg, A., Maire, M., and Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 2126–2136, Washington, DC, USA.

Zhu, C., Bichot, C., and Chen, L. (2011). Visual object recognition using daisy descriptor. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Zolynski, G., Braun, T., and Berns, K. (2008). Local binary pattern based texture analysis in real time using a graphics processing unit. In *Proceedings of Robotik 2008*, volume 2012 of *VDI-Berichte*, pages 321–325, Munich, Germany. VDI, VDI Wissensforum GmbH. ISBN 978-3-18-092012-2.