Color based terrain cover classification for off-road autonomous navigation

An environmental auto-calibrated approach

Paul Jansen
jansenp@science.uva.nl
Artificial Intelligence — Autonomous Systems
October 2004

Supervisor: Drs. Wannes van der Mark

A thesis submitted to the faculty of science of the University of Amsterdam in partial fulfillment of the requirements for the master of science degree.


Intelligent Autonomous Systems Group
Informatics Institute
Faculty of Science
University of Amsterdam


TNO
Physics and Electronics Laboratory
Electro-Optical Systems

Intelligent Autonomous Systems Group
Informatics Institute
Faculty of Science
University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam, The Netherlands
Tel: +31 20 525 74 61 Fax: +31 20 525 74 90
http://www.science.uva.nl/research/ias/

TNO Physics and Electronics Laboratory
Electro-Optical Systems
Oude Waalsdorperweg 63, P.O. Box 96864
2509 JG The Hague, The Netherlands
Tel: +31 70 374 00 00 Fax: +31 70 374 06 54
http://www.tno.nl/instit/fel/

Signature for approval of submission by thesis supervisor:

Drs. Wannes van der Mark,
October 2004,
Amsterdam.

Abstract

The University of Amsterdam (UvA) and the Netherlands Organisation of Applied Scientific Research (TNO), developed the Robojeep: a testbed for autonomous navigation in unstructured terrain. The Robojeep research is focused on enabling future robotic vehicles to perform useful and dangerous tasks autonomously. An appealing example is humanitary mine clearing: financial costs as well as casualty rates can be reduced significantly by automating this dangerous task.

Presently, the Robojeep depends primarily on its range sensors for observing the terrain conditions. The drawback of these sensors is that they do not provide information about the material types, such as grass or sand, that are present in the terrain. Terrain classification is needed for various applications, for example: identification of false obstacles such as tall grass or dirt road following.

We have developed a color based terrain cover classification algorithm. The apparent color of terrain types is heavily influenced by lightning, weather type and other environment conditions. Therefore, for color based classification to be successful in an outdoor environment, the environment state has to be estimated. Previous research on terrain cover classification still considered auto-calibration as an open problem. We were able to solve this problem by assuming that low variance image regions are heavily influenced by the environment state.

Our approach is able to train and classify without a-priori knowledge about the environmental state. For each environment state found by our unsupervised environment cluster algorithm, an environment hypothesis is learned as well as a terrain cover hypothesis. Both hypotheses learn mixtures-of-Gaussians in color-space by using a greedy variant of the Expectation Maximization (EM) learning algorithm. The environment estimation and the terrain cover classification are both based on Maximum Likelihood principles: classification can be based on the environment state with the maximum likelihood, and on an interpolation of environment states.

Our environment auto-calibration classifier performs as well as a ground-thruth based classifier. The classifier shows very reliable performance when used for discriminating between drivable and non-drivable areas. Discriminating between more classed, which are required for improving vehicle navigation or control, also showed satisfactory results.

The results indicate that a good first step towards a very robust terrain cover classification algorithm for autonomous navigation has been achieved.

Keywords: color, terrain classification, auto-calibration, environment clustering, off-road navigation, mixture models, greedy EM.

A paper describing the research in this thesis has been submitted to the *IEEE International Conference on Robotics and Automation (ICRA)* of 2005 in Barcelona, Spain: Paul Jansen, Wannes van der Mark, Johan C. van den Heuvel, Frans C.A. Groen, "Colour based Off-Road Environment and Terrain Type Classification".

Acknowledgment

In the time I worked on this thesis I was constantly supported by a number of people. Foremost, I would like to express my gratitude towards my supervisor Wannes van der Mark. His patient guidance, encouragement and excellent advice have been constant factors throughout this work. I also learned a great deal from Prof. Frans Groen. Especially his remarks on how a thesis should be written were very helpful.

My work experience at TNO-FEL presented me with insights about working in such a large research company. Moreover, I express my gratitude that they gave me their confidence to work on their Robojeep project. I would like to thank my former colleagues for making me feel welcome. Two people stand out: Johan van den Heuvel for his dedicated guidance on my research project, and Han van Bezooijen for his encouragement and social diversion.

Nikos Vlassis, Jakob Verbeek and Jan Nunnink provided me with the greedy EM learning algorithm. I would like to thank them for supplying this algorithm that was so important for this project.

Finally, I would like to thank the people I love. My mom and dad's support went the extra mile. They picked me up when I was down and they kept believing in me. My girlfriend, ♡nita, gave me the push that accelerated the process of finishing this thesis. I especially value her support in the final two months were all I could talk about was finishing this project in time.

Paul Jansen;
October 2004;
Amsterdam

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This report describes a classification algorithm for natural terrain cover. It has been developed for the Robojeep: a research platform for autonomous navigation in unstructured terrain. The algorithm should distinguish classes such as sand, rock, bark, grass and foliage, giving the Robojeep a better sense of its environment. This chapter introduces the Robojeep vehicle and its possible applications. Arguments are adduced to show the importance of terrain cover characterization for the Robojeep's navigation capabilities, which results in the definition of the problem statement. Finally, a chapter overview of the report is given.

This research project is a part of collaboration between the TNO Physics and Electronics Laboratory (TNO-FEL) and the University of Amsterdam (UvA). The principal aim of the Robojeep is the development of methods enabling the vehicle to conduct missions autonomously and reliably in an unstructured environment, making use of only onboard sensors, intelligence and computing. Different sensors such as: stereo-vision cameras, ultra-sonic sensors, odometers and a laser range scanner are applied for this goal. Techniques are developed by TNO and the UvA for acquiring a reliable representation of the environment from these sensors. A path can be planned on basis of this environment representation. This path is converted to control commands for the Robojeep's actuators: the gearbox, the steering wheel, the accelerator and the brake pedals. A picture of the Robojeep is shown in figure 1.1.

The technology developed for the Robojeep could enable future vehicles to perform dangerous tasks, such as landmine detection and rescue missions without fear of human casualties. Other applications could be to automate activities, such as digging on construction sites or harvesting in agriculture.

1.1 Need for natural terrain classification

From a pair of stereo images a height map can be calculated. Such a geometrical description of the terrain allows path planning avoiding positive and negative

Figure 1.1: The Robojeep

obstacles. This description is suitable for structured environments. An urban environment has clear geometrical features like roads, houses and walls. A powerful assumption can be made in this case, namely: everything in the world is rigid. This means that nothing in the world is penetrable or flexible. Roads can be robustly followed because they are easily identified, making it a closed environment. This closed environment enables the knowledge that no harm is done if one does not leave the road and avoids steep objects; then one does not drive in to a river and no people or cars are hit. Only much less occurring scenarios will not hold. For example a cardboard box on a road that is seen as an object, but in fact is penetrable.

Driving off-road does not restrict the environment very much, making it an open and not entirely rigid environment. The Robojeep, with its stereo-vision based navigation method [21], can wander off into all sorts of terrain. Driving behavior is related to the physical properties of the surface. The geometrical description gives no information about these properties and so the Robojeep can not alter its driving behavior to a surface. For example, it will not slow down when terrain changes from stone to sand. The rigid assumption has another major disadvantage: the geometrical description does not always give the load-bearing surface (where the wheels touch the ground). Patches of tall grass are easily traversable without risk of damaging the vehicle. They are however

detected by the stereo-vision algorithm as obstacles thus decreasing the number of places the jeep can reach. Moreover, lakes and other bodies of water present a danger because they are difficult to detect with only range information. These arguments show that the Robojeep needs a terrain cover classification algorithm next to its stereo-vision based navigation to interact well with its natural environment.

There are a number of examples that illustrate extra benefits of a terrain cover classification algorithm. Firstly, detecting soil from bushes is important for autonomous navigation and can not be achieved by range analysis alone. Secondly, detecting tree lines has a tactical value. Stereo-vision and other range sensors have a relative small range. An obstacle detection algorithm can provide the path-planning component with information about obstacles at a larger range, enabling the path-planning component to look ahead. Thirdly, distinguishing between paved and unpaved surfaces is helpful for road following. Finally, distinguishing between natural and man-made obstacles is important for interaction with human beings.

1.2 Problem statement

The Robojeep's terrain cover classification algorithm should satisfy some constraints:

1. the algorithm should be able to robustly classify new images as drivable or non-drivable areas under varying weather conditions, seasons and environments given a rich enough learning set and information from the stereo cameras;
2. the algorithm should have a measures that reflect the confidence that it has in its classification;
3. the algorithm should be fast enough for real-time applications, and
4. the algorithm should be able to be extended with other classification algorithms.

Constraint 1 reflects the primary purpose of the algorithm, but a broader terrain class taxonomy is preferred. Constraint 2 reflects the fact that outliers, like houses and people, should be detected. An autonomous system should be able to react fast enough to its environment, which is captured by constraint 3. Because the first three constraints are very hard to satisfy with just one classification method, the classification algorithm should be able to be extended with other classification methods. This notion is captured by constraint 4.

1.3 Overview report

This section provides a general overview of the report. The next chapter describes a literature study to provide a general, but well founded method for

tackling the problem statement. It focuses on feature selectors and classification methods. This research guided us towards developing a color classification algorithm that is auto-calibrated on the environment state.

Chapter 3 explains the method we used to solve the terrain cover classification problem. The three main components of our method are described here, namely: the environment cluster component, the learning component and the classification component. Moreover, the assumption we made to tackle the auto-calibration problem is explained.

Chapter 4 describes the experiments we performed to test our method. Our auto-calibration algorithm matched the results of a ground-thruth based method. Moreover, the terrain cover classification results indicate we have achieved a very good first step towards even better performing terrain classification algorithms.

Finally, chapters 5 and 6 respectively describe the conclusion and future research possibilities. In the conclusion, we summarize our method and discuss its advantages and disadvantages. The last chapter describes future research projects that could improve our method.

Chapter 2

Related research

In this chapter we will review the state-of-art research for outdoor classification of natural objects. It spans an area from research on statistics of natural images to a practical implementation of a vehicle somewhat capable of navigating in an outdoor environment. For robots such as the Robojeep to work well, real world image processing and classification has to be done in real-time.

The first sections discuss the features. Respectively color, texture, motion, 2D-geometry and 3D-geometry are dealt with respect to how well they represent the world. This is done in light of their physical properties and the properties of the natural environment. Next some classification methods are discussed. The information gathered by this literature research, guides the choices made in developing our terrain cover classification algorithm, which is described in the next chapter.

2.1 Geometry features

As discussed in the introduction, the Robojeep is equipped with a stereo camera. Stereo-vision enables autonomous vehicles to infer information on the 3D structure and distance of a scene using images from different viewpoints [21]. Retrieving a height map from the scene takes two steps:

1. solving the correspondence problem;
2. solving the reconstruction problem.

The correspondence problem is solved by matching pixels in the left image with the corresponding pixel in the right image. The reconstruction problem is solved by using basic geometry, and finds 3D locations and structures of objects.

Solving the correspondence problem is computational attractive because of geometry formed by the stereo cameras. A so called epipolar plane is defined by three 3D points: the optical centers of the two cameras and a 3D point in the world. By intersecting the epipolar plane with an image plane, the epipolar line is found. This epipolar line reduces the search for correspondences to a 1D

problem: a pixel in the left image defines a line in the right image on which the corresponding point must lie.

Figure 2.1 shows a solution of the correspondence problem. The distance between the most outer point and the middle point on the image planes ($a1$), is equal to the middle point and the most inner point ($a2$). The error in distance estimations ($d1$ and $d2$), however, raises to the square with distance. This illustrates a major disadvantage of a stereo-vision system: only the distances of points close to the camera can be accurately estimated. Distance estimation can be made more accurate by enlarging the distance between the cameras, the so called baseline, or by using telelenses.

Talukder developed an obstacle detection algorithm that is able to segment obstacle points into clusters [31]. Because background information is discarded in the object segmentation, shape-based reasoning is made possible. This facilitates the use of 3D shape and geometrical measures to classify obstacles. Talukder uses a rule-based system to reject obstacles with small bounding volume, maximum slopes or maximum height. Such systems can also aid the classification algorithm. Classifications that are incompatible with the shape can be revised. Common objects, such as stones, can vary significantly in size. This makes post-processing by means of geometrical data a very object restrictive operation. Suppose a classification algorithm is unsure about whether an area is grass or foliage. A rule-based system can label the object as penetrable if its volume is much smaller than the volume of foliage.

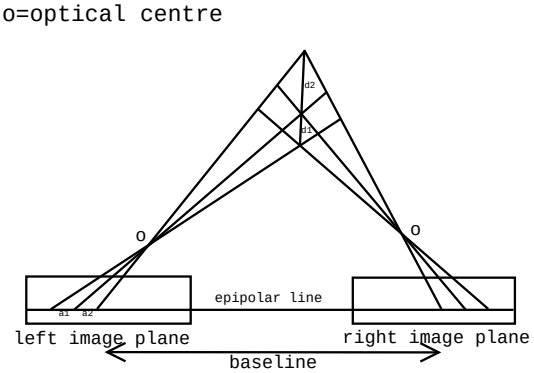


Figure 2.1: Accuracy of distance estimation by stereo-vision decreases at larger distances.

2.2 Color features

Research in natural image statistics shows that not many colors are necessary to provide a sparse representation of color images depicting natural scenes. However, these colors are very important for such a representation, because

their frequently used [18, 16]. This is a strong argument to investigate the use of color for a terrain cover classification algorithm.

2.2.1 Color physics

A camera comparable to the human eye senses the environment. In the eye reflection of the world is continuously projected on the retina covered by a grid of light-sensitive cells. We can compare this grid very well to video camera CCD chip rectangle composed from millions of pixels. The principle is simple — each cell measures the intensity of a range of light frequencies. For a CCD camera the color and brightness of a point is represented by intensity measurements of red, green and blue: the RGB values.

The experience of color is a result of how we interpret the wavelengths of radiation found across a range of frequencies in the visible spectrum. The CIE (the commission international d'éclaire) mapped the wavelengths (400nm-700nm), to triple number coordinate systems that mathematically define how humans perceive color. The RGB coordinates can be transformed to CIE XYZ coordinates by multiplication

$$P_{XYZ} = M \cdot P_{RGB}. \tag{2.1}$$

The matrix M has size 3×3 and is determined by a reference white.

Color based object recognition can be very complex in outdoor circumstances. The apparent color of an object depends on color of the light source, the reflectance of the object, illumination geometry, viewing geometry and sensor parameters [25]. Illumination and viewing geometry vary with changes in object and camera position as well as orientation. A pictorial description of the various processes involved in the formation of outdoor images is shown in figure 2.2.

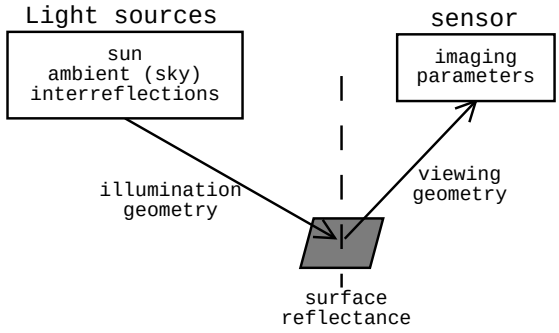


Figure 2.2: Image color formation in outdoor scenes.

In outdoor images, the color of the illuminant varies with the time of day, cloud cover and atmospheric conditions. The color of daylight varies along a characteristic curve. This so called CIE daylight curve is defined by

$$y = 2.87x - 3.0x^2 - 0.275$$

in xy-chromaticity space [17]. The xy-space is chosen to represent the chromaticity space. It intersects the XYZ-space with the plane $X + Y + Z = 1$. The apparent color of an object is not only affected by the color of the illumination, it is also affected by shadows and inter-reflections. [11].

There are two types of reflection: specular and diffuse reflection. The outdoor environment contains only a little amount of shiny surfaces. This allows omitting of the specular reflectance components in the reflection model. Only the Lambertian reflectance, having uniform reflectance in all directions, has to be modeled. For a Lambertian surface, the measured intensity ρ_c of channel c, $c = 1, 2, \ldots, n_c$, is

$$\rho_c = g \int_\lambda f_c(\lambda) s(\lambda) l(\lambda) d\lambda. \tag{2.2}$$

Here g is an effective light intensity determined by the scene geometry and the absolute light intensity; $f_c(\lambda)$ is the sensitivity function of channel c; $s(\lambda)$ denotes the reflectance; $l(\lambda)$ represents the normalized light spectrum; and all of the above variables are functions of the wavelength λ. Since we use a RGB color camera, we can assume $n_c = 3$ hereafter, without loss of generality.

2.2.2 Color interpretation

The physics of color shows that the apparent color can change at different times of day, under different weather conditions and at various positions and orientations. Extracting the object color out of the sensor responses without knowing the spectral composition of either the incident light or the surface reflectance, is known as color constancy. This is a challenge because numerous combinations of factors can result in the same set of sensor values. The combinations of factors can cause large variations in perceived color of an outdoor object. The variation in apparent color of a single surface can even be greater than the difference between two distinct colors [5]. The variation of the apparent color of more realistic objects can even be greater.

In the literature there are many traditional color constancy algorithms mentioned. These algorithms assume highly restricted images under constrained lighting. In outdoor circumstances these assumptions and techniques will not work, and consequently there are no color constancy algorithms that work well on real images [4, 10].

Colorspaces

The colorspace in which a pixel or region is classified, is of great importance. Colorspaces can inhibit several invariant properties like: viewing direction, surface orientation, highlights, illumination intensity, illumination direction, color of the light source and interreflections. Surfaces in natural environments are mainly matt and consequently will be well modeled by the Lambertian model. This means there is no essential need for highlight invariance.

Colorspaces are not only important for their invariance properties. They can also make the modeling of the objects simpler. There are three important object modeling properties:

1. the object model fit;
2. the object model separation, and;
3. the object comparability.

The first property describes how well the data can be fit to a arbitrary model. This can be a mathematical model such as a normal distribution, and can also be a decision boundary described by for example a decision tree or a neural network. The second property describes the separation of different objects. Two objects that have a large distance from each other are not that easily confused. The last property describes if the distance in the colorspace is a fair guide to the significance of the difference between two colors. In a nearest neighbor comparison this can be of great value. Property two is more related to the environment, while property three is more related to the physics and perception of color.

Objects can cluster when larger chromatic variations occur, which can be caused by different time of day or season, or by changing weather. The Jet Propulsion Laboratory (JPL) shows that small chromatic variations can be modeled reasonably well by a mixture-of-Gaussians class likelihood [1]. They increase performance by calibration on an outside reference when larger chromatic variations occur. Tsin et al. make their classification algorithm more robust to large chromatic variations by locally estimating the reflectance, geometry and light spectrum [33]. This estimation is done by locally solving a bilinear model for color image formation. A linear iterative procedure reduces processing time, nevertheless not enough for real-time purposes.

Figure 2.3 shows an image in which some areas are manually labeled: sky, sand, foliage and grass. Figure 2.4 shows the manually labeled areas in various colorspaces to illustrate the advantages and disadvantages. Here, sand and grass areas are sublabeled by if they are under influence of direct sunlight or are subject to shadows. Foliage is not shown for illustration reasons: it is large geometrical variance causes a distribution that clutters the other objects.

The RGB-space does not contain any invariance properties. In RGB-space, a object color distribution is not well described by a single Gaussian. A single object color distribution can produce two very distinct clusters, see figure 2.4. Both clusters seem to be normally distributed and can be explained by the incident light. One cluster corresponds to direct sunlight, the other corresponds to light that is reflected from other surfaces. The figure gives an indication that an object color distribution can be well fitted by a mixture-of-Gaussians. This indication is confirmed by research on terrain cover classification with DEMO III: a research platform of the JPL similar to the Robojeep [1]. Unfortunately objects are very close to each other, especially in shadow regions, making it hard to discriminate two classes. Neither is the Euclidean distance in RGB-space a

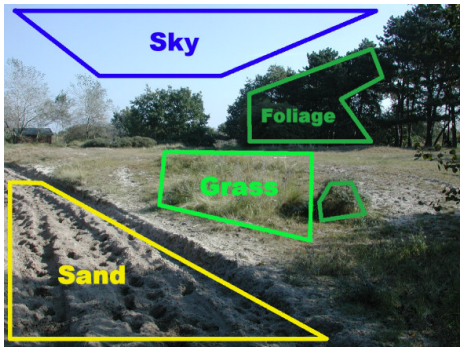


Figure 2.3: An image showing four manually labeled classes: sky, foliage, grass and sand. Grass and sand have labelled areas in direct and indirect sunlight.

fair indication of color difference. The difference is based on the difference between power levels of radiation, and not on the perception of color.

An intensity normalized space of the RGB-space, the rgb-space, is defined by:

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}.$$

This space is invariant to viewing direction, surface orientation, illumination intensity and illumination direction [12]. It however lacks important information about intensities and a supplementary classification is necessary for pixels with intensities close to zero. Since the r, g, b chromaticity values sum up to one, the rgb-space can thus be defined by only two variables. Figure 2.4 shows that modeling the clusters is more difficult. The clusters can not be fitted with a mixture-of-Gaussians very well and the clusters are not that separated.

The opponent colorspace [34], given by

$$wb = R+G+B, \quad rg = R-G, \quad yb = 2B-R-G,$$

is a linear transformation of RGB that matches the physiology of the human visual system. Moreover, it is shown that color opponent coordinate systems can represent natural images very efficient [18]. The three axes represent the intensity values and two chromatic values. The chromatic values give the red-green and yellow-blue differences. Opponent colorspaces tend to decorrelate the RGB components, which is a desirable characteristic for pattern recognition [32]. As figure 2.4 shows, the opponent space separates the classes well, how ever they are harder to model.

Using a statistical study of uncorrelated color components on a large population of typical images, yields an opponent chromatic separation [34]. This so called Otha space is given by

$$I_1 = \frac{R+G+B}{3}, \quad I_2 = \frac{R-B}{2}, \quad I_3 = \frac{2G-R-B}{4}.$$

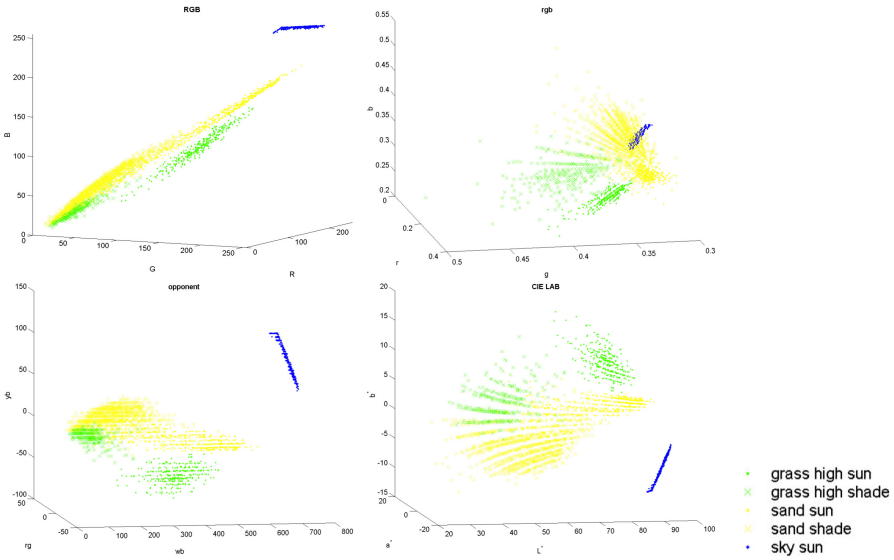


Figure 2.4: Manually labeled areas of figure 2.3 plotted in the RGB-space, rgb-space, opponent-space and CIE LAB-space (from left to right, top to bottom).

Its results are very similar to the opponent colorspace, thus no further discussed. One could also decorrelate color components based on the specific environment. Using a set of natural terrain images, principal-component analysis (PCA) can align the data with the axes. PCA uses an orthogonal transformation for this purpose. Non-orthogonal spaces can be even more efficient in describing the redundancies due to overlap of sensor excitations and across waveband correlations present in natural spectra. Independent component analysis (ICA) finds a linear non-orthogonal coordinate system, allowing higher than second order statistics. This can make the data simpler to analyze. Research on outdoor image statistics shows that both the PCA and the ICA find a colorspace somewhat similar to the opponent colorspace [30, 18]: they find one axis corresponds to radiance, another describes an opponent blue-yellow channel and the last describes an opponent red-green channel. These axes are mentioned in order of decreasing eigenvalue.

As discussed, time of day and shadows respectively influence illumination color and cause inter-reflections. The m1m2m3-space is also invariant to these properties; however, it can only describe the difference between surface albedos and thus contains no information about the actual color of the surface [12]. An important benefit is that it is not assumed that the neighboring points have the same surface orientation. The m1m2m3 color model is given by

$$m_1 = \frac{R^{\vec{x}_1} G^{\vec{x}_2}}{R^{\vec{x}_2} G^{\vec{x}_1}}, \quad m_2 = \frac{R^{\vec{x}_1} B^{\vec{x}_2}}{R^{\vec{x}_2} B^{\vec{x}_1}}, \quad m_3 = \frac{G^{\vec{x}_1} B^{\vec{x}_2}}{G^{\vec{x}_2} B^{\vec{x}_1}},$$

where $\vec{x}_1$ and $\vec{x}_2$ are image locations of neighboring pixels. There is an indication

that mammals use the same method: using two surfaces, the spatial ratio of cone-photoreceptors is almost invariant under illumination changes and under shadow existence [24]. One can only model object differences, which has the disadvantage that areas with high variance clutter areas of lower variance. This disadvantage can be overcome by computing the m1m2m3 values at different scales, or comparing areas with each other.

One common disadvantage of all above mentioned colorspaces is that colors can not be compared to each other; that is, all above mentioned spaces are not perceptually uniform. In 1976 the CIE defined the LAB-space to overcome this disadvantage. The L^*, a^* and b^* values are calculated in two stages. The first stage calculates the XYZ values according to function 2.1. The second stage transforms the XYZ values non-linear to LAB values using:

$$\begin{aligned} L^* &= 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16; \\ a^* &= 500\left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right]; \\ b^* &= 200\left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}\right], \end{aligned}$$

where X_n, Y_n and Z_n are the X, Y and Z coordinates of a reference white patch. Differences in LAB coordinates give a good guide to how different two colors will look to a human observer. The difference between colors $L_1a_1b_1$ and $L_2a_2b_2$ can be expressed as:

$$\Delta E = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2}.$$

The L^* represents luminance and ranges from 0 for black to 100 for white in uniform steps. The a^* and b^* values are represented as $+a/-a$ for red-green differences and $+b/-b$ for yellow-blue differences. Figure 2.4 shows the LAB space. The objects are still reasonably well separated, but they are more difficult to model.

2.3 Texture features

Color can provide valuable information about the environment, nevertheless not everything is distinguishable by its color. Color has some shortcomings. Firstly, it can not be used at night. Secondly, it will not allow robust separation of dry vegetation or tree bark from certain kinds of soil [29]. Finally, as the Lambertian reflectance model illustrates, the perceived color is a function of the reflectivity properties of the surface and of the spectrum of the illuminant; this makes color classification rather hard in circumstances like shading and changing weather conditions. Other visual features can complement color analysis. For example texture: the extra information needed apart from color to adequately describe an image region with local spatial correlations.

Features are often specified by a few properties called feature descriptors, namely:

- type;
- orientation;
- scale, and
- color.

Feature types include bars, spots and edges. Feature types can be of different orientations and different scales. A bar feature can be small and vertical; however, it could also be large and horizontal. Color transitions can also be of use, for example in searching for a red bar in a green plane. It is shown that color opponency is an efficient representation of spectral properties, although uniform chromatic filters and luminance edge filters are more important in an efficient coding of natural images [18].

Natural images can be efficiently coded by localized and oriented structures described by the feature descriptors [26]. The advantage of finding such feature descriptors in an image is that the local structure of the image becomes clear. There is a strong response when the image pattern in a neighborhood looks similar to the searched feature and a weak response when it does not. A benefit over color classification is that only one channel is required: texture may be computed on images from single-channel infrared cameras, allowing terrain cover classification at night. Notice that this is only possible if the resolution and quality of the infrared camera is sufficiently high.

2.3.1 Feature extraction

There are two principal methods two extract features from an image:

1. convolving an image with a mask, and
2. finding frequencies in local image windows.

A convolution with a mask replaces the pixel value $I(i,j)$ of an image I with a weighted sum of the values of I in a neighborhood of (i,j). The weights are the entries of the mask M. Finding the frequencies can be done by a Fourier transform. The two methods are related by the convolution theorem: the Fourier transform of the convolution of image I and mask M is the product of their Fourier transforms $\mathcal{F}(I)$ and $\mathcal{F}(M)$.

convolving an image with a mask

Convolution masks can be created in several ways. One way to obtain these filters is to form a weighted difference of Gaussian filters at different scales. A spot filter for example, can be made by $G = G_{0.62} - 2G_1 + G_{1.6}$, where

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right),$$

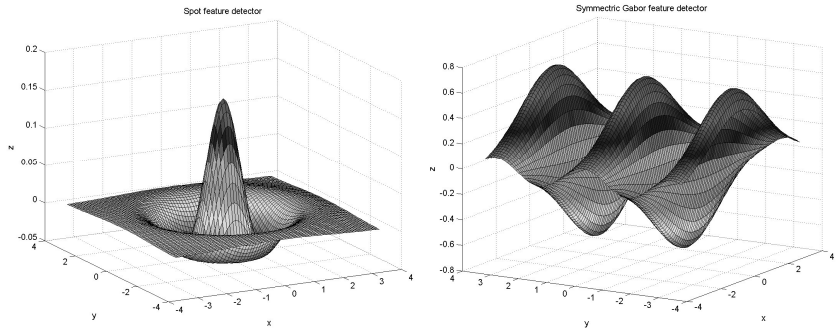


Figure 2.5: A spot filter made of the weighted sum of three concentric, symmetric Gaussians at different scales (left), and a symmetric Gabor function (right).

is the symmetric 2D Gaussian kernel, with the filter scale given by σ. Figure 2.5 illustrates the spot filter. Bar filters can also be obtained by differentiating oriented Gaussians.

Another way to build spot and bar filters is to use Gabor filters. Gabor filters strongly respond at points in an image where there are components that locally have a particular spatial frequency and orientation. Gabor filters come in pairs: a symmetric component and an anti-symmetric component, respectively given by

$$\begin{aligned} G_{symmetric}(x,y) &= \cos(k_x x + k_y y) \exp - \left(\frac{x^2+y^2}{2\sigma^2} \right), \\ G_{anti-symmetric}(x,y) &= \sin(k_x x + k_y y) \exp - \left(\frac{x^2+y^2}{2\sigma^2} \right). \end{aligned}$$

The filter scale is given by σ. The spatial frequency to which the Gabor filter responds most strongly is given by (k_x, k_y). Figure 2.5 shows the symmetric kernel with $\sigma = 1$, $k_x = 4$ and $k_y = 2$. Gabor filters can be thought of as assemblies of bars: as the spatial frequency increases compared to the scale, the filter looks for patches of parallel stripes rather than individual bars. This could be convenient for the classification of grass.

Opponent chromatic edges can be found by applying an edge detector. This edge detector is not applied on the intensities, but on the wavelengths. An adjustment of this method can even classify color transitions into shadow-geometry, illumination and material edges [13]. Incorporating the physics of light makes the color-texture filter more robust to shadows [15].

Because local features can occur at different orientation, scales and colors, a collection of filters is required. This collection usually contains spots and bars filters. It has been shown that implementing rotation invariance degrades the classifiers performance in the autonomous navigation task [6]. This can be explained by the fact that certain terrain types have a principal orientation: grass mostly has vertical edges. Therefore rotation invariance is not discussed

any further. Moreover, this research reports probability values of 0.70 and 0.68 of detecting correct terrain classes, where the first value is the normal probability and the second value is the probability given that all terrain classes occur equiprobable. For the obstacle detection task, where the is little or no margin for error, higher probability values seem to be a necessity.

finding frequencies in local image windows

Gabor functions filter out spatial frequencies, but these frequencies can also be found by means of analyzing an area. A Fourier analysis decomposes an image area into an integral over sine and cosine waves. This way, one obtains a representation of the original image area that allows one to identify which frequencies are contained in the image area. Wavelet analysis is an improvement over Fourier analysis. In order to isolate signal discontinuities, very short basis functions are practical. In contrast, in order to obtain detailed frequency analysis, very long basis functions are convenient. A way to achieve this is to have short high-frequency basis functions and long low-frequency ones. Wavelets to exactly this.

Using spectral energy distribution within the Fourier spectrum for color texture recognition is limited by the impossibility of handling vectors with more than two components. By reducing the colorspace to two dimensions with equal significance, accurate spectral techniques can be used on color images [34]. Wavelets can be used to produce wavelet correlation signatures which define the energy in each color plane and the cross-correlation between different planes [36].

2.3.2 Functional issues

Unfortunately finding texture features is computationally quite expensive, whether this is done by a convolution of a filter with an image or by frequency analysis on an image area. There are two reasons why it is hard to reduce the number of features that have to be found. Firstly, these transforms do not actually reduce the number of parameters needed to code the population of natural scenes. They only reduce the number of parameters needed to code a particular instance of a natural scene [9]. A general terrain classification method needs to work on the whole population, which keeps the number of filters high. Secondly, to make the texture analysis scale invariant, the image has to be filtered at different scales. Research in natural image statistics informs about what frequency steps to use. In natural images, edges are rarely very straight. Any given edge will typically shift location and orientation across scale. The degree of predictability is around the 1-2 octave range. This makes localized and oriented structures sparse when the image is analyzed at frequency steps that are a magnitude of 2 to 4 apart [7, 8].

Expensive computational approaches, such as extracting texture features, are unsuited for application on a robot vehicle. If the searched feature size in the real world is known, it is possible to extract features at the real world scale

in the image. This can improve real-time computation. For such an adaptive scale filtering method, range data acquired from a stereo camera pair is sufficient [27]. There are two disadvantages of this method. Firstly, the searched feature size is not always known. For example: stones come in different sizes. Secondly, the uncertainty in range estimation by stereo cameras increases with distance (see section 2.1). This means this method can only provide enough information to be used as an obstacle detection algorithm.

A general texture classification algorithm may constrain real-time computation too much. A more specific classification algorithm may be fast enough for real-time applications, because less filters are needed. Such an algorithm can aid the general classification algorithm with valuable information. Suppose the general classification algorithm has difficulty to robustly separate sand from grass. A fast grass classification algorithm based on texture features can separate both classes. De Bonet has developed an algorithm that can characterize non-homogeneous textures [2]. This includes textures with long-scale structure that are neither local in space nor local in frequency. Grass is such a texture. Section 2.6.5 discusses some more general methods to reduce computational costs.

2.4 Motion features

Motion features can be extracted from spatial and temporal changes occurring in an image sequence. These changes are caused by a relative motion between camera and scene. Apparent motion of objects onto the image plane is a strong visual cue for understanding structure and 3D motion. There will be a brief discussion of its possibilities, followed by arguments why not to implement it as a general real-time classification method.

The first essential step is determining the correspondence: which elements correspond to which elements of the next image. One option is the tracking texture features, such as corners. Estimating the apparent motion of the image brightness pattern, the optical flow, is another possibility. Both possibilities are computational trackable because the images are closely sampled: there are only small spatial and temporal differences. In the optical flow case, the apparent brightness of relatively moving objects does not change. In the tracking case, predictions of feature locations can be estimated out of previous feature locations.

By using motion features it is very difficult to recover a structure. This is because the baseline between two consecutive frames, regarded as a stereo pair, is very small (see section 2.1). However, finding the structure of the terrain may not be necessary. It could be that terrain types leave a specific signature in motion feature space.

A common concept of motion features is a point called the focus of expansion. This point occurs if the camera gets closer to the scene, which is mostly the case in the terrain navigation task. In this case the motion field is radial, with all vectors pointing away from the focus of expansion. There are two other motion

field properties in this case. The length of the motion field vectors is inversely proportional to the depth and is also inversely proportional to the distance from a point to the focus of expansion.

There are a number of reasons why not to develop a motion based classification method:

1. in the known literature to us, there is no well founded indication of usefulness of motion features for natural terrain classification;
2. the camera has to move for the algorithm to work;
3. the algorithm has to be invariant to speed, orientation and distance, which is difficult because of the earlier mentioned motion field properties;
4. the image resolution could be a problem because the images are closely sampled;
5. a segmentation step has to be performed to find what regions of the image plane correspond to different objects.

After a preliminary classification, motion features can provide extra information. However, they are not an attractive first step for terrain classification.

2.5 Other sensor features

Other sensors can also aid in the obstacle detection task. This section briefly discusses the laser rangefinder, the infrared camera and the near infrared camera.

Laser rangefinder features

The LADAR, an acronym of laser detection and ranging, uses laser light for detection of speed, altitude, direction and range. It scans the environment in horizontal and vertical directions, yielding a 3D representation of the terrain. DEMO III uses it not only for a geometrical representation, it is also used for terrain classification [14]. They make the observation that natural objects follow characteristic distributions in 3D space. Intuitively, points that are on a surface will yield "flat" distributions. Points on a elongated structure, like tree trunks, will yield mostly one dimensional structures. Points in the middle of widely scattered point clouds, like bushes, will yield distributions that are roughly equal elongated in all dimensions. Subsequently, neighboring points with similar shape distributions are grouped into image regions.

This method has a major benefit: tree branches and trunks are recovered even though they are partially obscured by vegetation. Another benefit is that the ground level can be robustly detected even though it is obscured by vegetation. Unfortunately there are some disadvantages too. Firstly, confidence in a classification stays very low until there is a sufficient density of points in a neighborhood. This means that an autonomous vehicle is not always able to

react to its environment. A second disadvantage is that it is hard to select the size of the neighborhood over what the features are calculated. This is a very important property for the robustness of the classification. Finally, the classes 2D surface, linear structures and 3D scatter do not represent the environment very well. Color and texture features are complemented very well by LADAR features. However, more research on LADAR's for terrain classification has to be done.

multispectral features

At wavelengths above 9 micron, the radiant component of the sun is small, therefore thermal measurements are rather independent of the sun light condition. This is an advantage with respect to color analysis. Another advantage is that thermal measurements can be used at night. The emissivity signature of dry grass and soil are rather distinct in the thermal measurement case, as opposed to the reflectance in the visible and in the-near infrared spectrum. Living vegetation is very reflective in the near infrared spectrum though. These findings were researched in the DEMO II project [22].

There are some disadvantages to thermal and near-infrared features. Firstly, more research has to be done to find what bands are needed for for discriminating among the classes of interest for autonomous navigation. Secondly, the temperature of the radiating surface determines together with the emissivity the measured emittance spectrum. Some way has to be found to make thermal measurements invariant to the temperature of the radiating surface. Robustly classifying live vegetation, dry vegetation and soil can be of great importance. Unfortunately it is not enough for a general classifier.

2.6 Classification

The features mentioned in the sections above are input parameters for a classification algorithm. This classification algorithm must able to map a feature set to a class, preferably together with a confidence measure. The terrain cover algorithm should be able to work in all sorts off terrain. It is probably too cumbersome for a human to deduce classification rules for each terrain type. Therefore only algorithms that learn a classification hypothesis are discussed.

Learning can be supervised or unsupervised. Supervised learning builds an hypothesis using classified instances. Unsupervised clustering receives no such feedback on an outcome. Supervised classification has some advantages over unsupervised segmentation. Firstly, the final result should be a classification. Secondly, classification allows more objective performance assessment, since the results of the classifier can be compared directly the manual labeling performed by a human operator. Finally, supervised classification generally does not use time consuming iterative processes. Unsupervised classification can counterbalance instances not present in the training examples. For terrain cover classification a hybrid classifier can be used [19]. This classifier uses a supervised color

classification algorithm and a texture cluster algorithm.

In this section decision trees, neural networks, nearest neighbor methods and Bayesian methods are discussed [23]. This is done in relation to how fast they classify a new instance, how well they describe the classes in feature space, how much training data is necessary and whether they allow a confidence measure. The previous feature sections have shown that one feature method probably is not enough for a robust classification. This makes the confidence measure of some importance for fusing the results of various classifications. Of course it is also important for the vehicle to have a degree of trust in its classifications. Finally, some remarks are made that relate to all discussed learning algorithms.

2.6.1 Decision tree learning

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Decision tree methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks. One of the reasons why decision trees are so popular is that the hypothesis is readable by humans and thus verifiable. Another reason is that the learning algorithm can handle errors in the training data.

However, the before mentioned features are all of continuous nature. A way to incorporate the continuous values can be by dynamically defining new discrete valued attributes that partition the continuous valued attributes into a discrete set of intervals. Multivariate decision trees define classes by thresholding linear combinations of several continuous valued attributes. They create piecewise-linear approximations to surfaces in feature space by recursively dividing the feature space with hyperplanes. These hyperplanes are not necessarily perpendicular to the coordinate axes.

Buluswar uses multivariate decision trees to learn regions in RGB colorspace [5]. He chose this method because it has been shown to produce good classification results from relatively few training samples. Moreover, they perform well on arbitrarily shaped clusters in feature space.

2.6.2 Neural network learning

Neural network learning methods provide a robust approach to approximating real valued and vector valued target functions. For certain types of problems, such as learning to interpret complex real world sensor data, neural networks are among the most effective learning methods currently known. The study of artificial neural networks are inspired in part by the observation that biological learning systems are build of complex webs of interconnected neurons. In analogy, artificial neural networks are build out of a densely interconnected set of simple units where each unit takes a number of real valued inputs and produces a single real valued output. A popular implementation of a neural network is the backpropagation algorithm. This implementation uses gradient descent to tune network parameters to best fit a training set of input-output pairs.

The input attributes in neural networks may be highly correlated or independent of one another. In the terrain characterization task this is a benefit, because it is not exactly known what features are important and how they interact. For instance, input to the neural network can be some color and texture features not knowing which features are important for classification. The network learns an hypothesis that inhibits a fusion of the input parameters giving more weight to important features. An extreme example is the ALVINN system that uses a neural network to steer a vehicle on a road [28]. The 960 camera outputs are directly fed into the network and the 30 output units encode the steering direction.

Neural networks can represent the classes very well, but its hypothesis is harder to interpret by humans. Moreover, neural networks require many training samples to estimate an accurate hypothesis. This is due to the fact that their is little inductive bias present in neural networks: the hypothesis space is only little constrained by the learning algorithm.

2.6.3 Nearest neighbor method

Nearest neighbor learning is the most basic method of the class of instant based learning methods. These instance based methods generalize only beyond training samples if a new instance is encountered. Then the relationship to the previously stored examples is examined in order to classify the new instance. The k-nearest neighbor algorithm selects the k nearest training samples according to the Euclidean distance measure. The new instance is classified as the most common value of the k neighbors. This method can be refined because the distances between the to be classified instance and the training samples are known: more weight can be given to training samples with a smaller distance.

It is robust to noisy training data and quite effective when it is provided with a sufficiently large set of training data. An advantage of nearest neighbor methods is that the sample that has to be classified can be considered to generalize beyond the training data locally. Other learning algorithms generalize the training data globally because they maximize the performance on the whole training set. An important disadvantage of nearest neighbor methods is that the computational costs of classifying new instances can be high. This is due to the fact that nearly all computation takes place at classification time rather than when the training samples are first encountered. As mentioned before, this is a major disadvantage for algorithms that are used in autonomous vehicle applications. Another disadvantage is that nearest neighbor methods typically consider all attributes of the instances when attempting to retrieve the similar training samples. If only a few attributes are important for classification, then instances that are truly most similar way well be a large distance apart.

2.6.4 Bayesian learning

Bayesian reasoning provides a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability

distributions. When this assumption is valid, optimal decisions can be made by reasoning about the probabilities together with the observed data.

Bayesian learning has some features that are not present in the above mentioned learning algorithms

- Each observed training example can decrease or increase the estimated probability that a hypothesis is correct.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions and thus provide a confidence measure.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

There is a difficulty though: they require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge or on previously available data, such as a learning set. Also assumptions about the form of the underlying distributions are often made. Terrain cover classes are reasonably well fitted by multiple normal distributions in some colorspaces. This is shown in section 2.2.2. Therefore, this difficulty can be overcome in some important cases if care is taken.

Bayes theorem is the cornerstone of Bayesian learning methods. In Bayesian learning methods the most probable hypothesis from some space H is to be determined given the training data D. In other words, the maximum posterior probability $P(h|D)$ of hypothesis h is to be found after seeing the training data D. The posterior probabilities can be calculated by Bayes theorem

$$P(h|D) = \frac{P(D|h)p(h)}{P(D)},$$

which is used to find the maximally probable hypothesis called a maximum a posteriori (MAP) hypothesis

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h).$$

The $P(D)$ term is dropped because it is a constant independent of h.

The probability that a new instance x belongs to hypothesis h is described by $P(h|x)$. By comparing the probabilities of the available hypotheses given an instance x, a confidence measure can be defined. For example, when $P(\text{grass}|x) \ll P(\text{foliage}|x)$, then the classification of the new instance as foliage has a higher confidence as classifying the new instance as grass.

EM learning algorithm

The EM algorithm is a widely used approach to learning in the presence of hidden variables. This is for example also the case in learning multiple normal distributions in a colorspace: there are a number of different normal distributions, and it is not observable which instances where generated by which distribution. The distributions are described by a model that consists of a weighted sum of basic model components. Such finite mixture densities are given by

$$f_k(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \phi(\mathbf{x}; \theta_j), \tag{2.3}$$

with $\mathbf{x}$ a random vector, k the number of components and $\phi(\mathbf{x}; \theta_j)$ the jth component model parameterized on θ_j. The mixing weights π_j are subject to the constraints $\pi_1 + \ldots + \pi_k = 1$ and $\pi_j \geq 0$.

The multivariate Gaussian density function is defined by

$$\phi(\mathbf{x}; \theta_j) = (2\pi)^{-d/2} |\mathbf{S}_j|^{-1/2} \exp\left[-0.5(\mathbf{x} - \mathbf{m}_j)^T \mathbf{S}_j^{-1} (\mathbf{x} - \mathbf{m}_j)\right].$$

The mean $\mathbf{m}_j$ and the covariance $\mathbf{S}_j$ are collectively denoted by the parameter vector θ_j. A multivariate Gaussian mixture is given by equation 2.3, where the jth component $\phi(\mathbf{x}; \theta_j)$ is the d-dimensional Gaussian density. When such Gaussian distributions are assumed, the hypothesis space only contains the mean and covariance matrices of the distributions. This is a major constraint of the hypothesis space that allows a smaller amount of training data to learn a hypothesis.

The EM learning algorithm iteratively re-estimates the hypothesis by repeating two steps. The first step uses the current hypothesis to estimate the hidden variables, and is given by

$$P(j|\mathbf{x}_i) = \frac{\pi_j \phi(\mathbf{x}_i; \theta_j)}{f_k(\mathbf{x}_i)}. \tag{2.4}$$

This gives an indication of what distribution was used to generate a particular value. The second step uses the estimates of the hidden variables to calculate the maximum likelihood hypothesis. The maximum likelihood hypothesis is the most probable hypothesis given the observed data. It assumes that every hypothesis is equally probable. The log-likelihood is given by

$$\mathcal{L}_k = \sum_{i=1}^{n} \log f_k(\mathbf{x}_i).$$

The task is to estimate the parameters $\{\pi_j, \mathbf{m}_j, \mathbf{S}_j\}$ that maximize the log-likelihood using the training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The estimation of the parameters can be carried out by the EM algorithm using the following iterative update functions for each component $j = 1, \ldots, k$,

$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} P(j|\mathbf{x}_i), \tag{2.5}$$

$$\mathbf{m}_j = \frac{\sum_{i=1}^{n} P(j|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} P(j|\mathbf{x}_i)}, \tag{2.6}$$

$$\mathbf{S}_j = \frac{\sum_{i=1}^{n} P(j|\mathbf{x}_i)(\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^T}{\sum_{i=1}^{n} P(j|\mathbf{x}_i)}. \tag{2.7}$$

The EM algorithm is very popular due to its simple implementation and the guaranteed increase of the likelihood during optimizations of the training set. There are some limitations though. Firstly, it assumes that the number k of mixing components is known. Secondly there is no widely accepted method for initializing the parameters. There are elaborations of the EM algorithm that try to overcome these problems, for example the greedy EM algorithm, developed by Vlassis and Likas [35].

The algorithm starts with one component. Regular EM steps are carried out until convergence. Then a new component is added to the mixture. The optimal position of the new component is found by first performing a global search among all input points. Looking at the current mixture f_k only m number of candidates per component are generated. This search step is followed by a local search based on partial EM steps for fine-tuning the parameters of the new component. This action is analogous with equations 2.4 to 2.7. An example is shown in figure 2.6. Input to the greedy EM algorithm are the RGB-points that lie in image areas with low variance. The greedy EM algorithm starts the learning process by fitting a mixture-of-Gaussians that only contains one component. On each iteration a component is added until all clusters are represented in the mixture.

EM color constancy extension

The standard EM learning algorithm is used by different researchers to learn color and texture features for terrain cover classification [1, 6, 19]. Values outside a support region are classified as outliers. In the case of color constancy there is a elaboration that is worth mentioning. Using the insights gained from Lambertian reflectance (see function 2.2), a separate estimate is determined of surface reflectance and illuminant. This is a benefit, because once the illuminant is determined this acts as a constraint when determining the surface type [3]. An iterative inference procedure was developed to speed up the classification process [33]. A table of typical colors of different material types under different lightning conditions, the so called mean color chart, is used for initialization. The initial surface reflectance and illumination is retrieved by selecting the vector in the mean color chart that has the smallest angle with the observed color vector. This initial step gives a fast classification that performs reasonably well. The iterative part of this procedure detects the outliers reasonably more robustly. This iterative inference procedure does speedup the classification, but not enough for real-time applications.

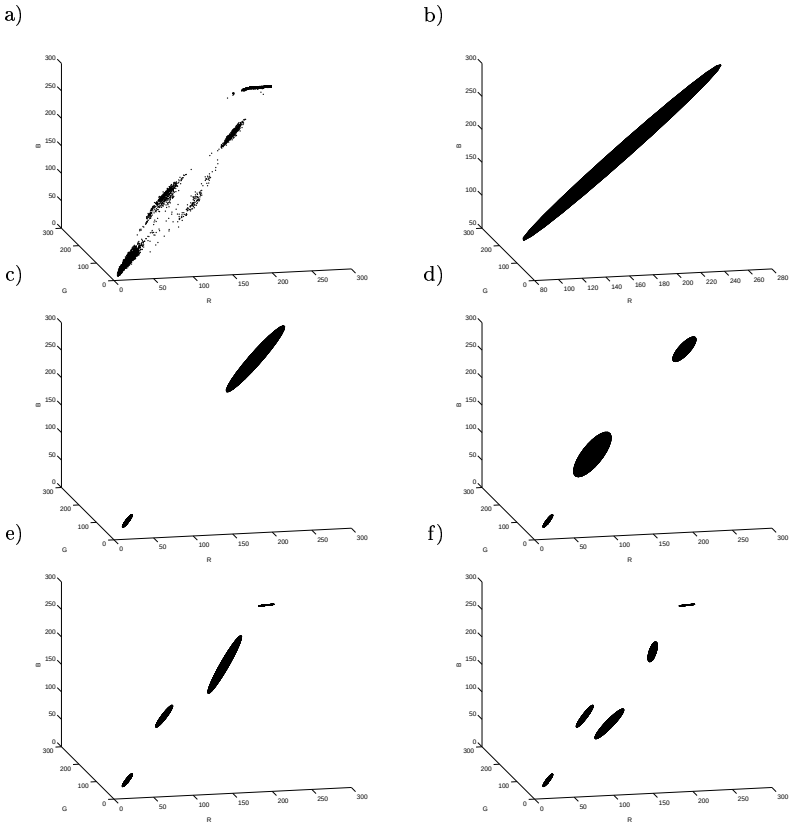


Figure 2.6: Greedy EM example. The fitted data points (a), and the other images represent iterations of the learning process (b to f).

2.6.5 General learning and classification remarks

When learning a hypothesis some care has to be taken. Firstly, learning algorithms can suffer from local maxima. The algorithm can get stuck in suboptimal solutions, and not find the best solution.

Secondly, learning techniques are susceptible to overfitting. A hypothesis that performs much better on the training data than on the unseen test data usually indicates overfitting of the data. Overfitting can occur when there is noise in the data set or the number of training samples is to small to generate a representative hypothesis. Cross-validation is a successful method to overcome overfitting. Both a training set and a validation set are used for learning. The hypothesis is learned with the training set, while its performance is evaluated with the validation data.

The training set learns the hypothesis. Hoewever, evaluation of the hypothesis is done on the validation data. This is done because the validation set gives a better indication of performance on the unseen data.

There are three ways to increase computational efficiency by reducing the image areas that have to be classified. Firstly, a region of interest can be established once the vehicles path is known. This region is to be classified robustly with high priority, while other regions can be searched if time allows it. A second approach to is to only apply more time demanding classification algorithms to areas with low confidence measures. For example, a texture classification can be performed in the color classified areas that have a low confidence measures. The last method is to only classify areas that 3D geometry based obstacle algorithm classifies as obstacles.

2.7 Conclusion

This exhaustive literature study guided the research toward implementing a color feature based approach. Color can tackle two frequently occurring problems. First, the classification by the stereo-vision component of patches of grass as non-drivable. It is important to minimize the number these false-positive objects. There are few impenetrable objects that are green and have a low height. Therefore, color and height information are powerful tools to minimize the number these false-positive objects. Secondly, bushes can now be separated from soil. This distinction between piles of soil and bushes is important for the obstacle detection task and is not achievable by range analysis alone.

Another benefit over other features exist. The benefit of color classification is that it can look further ahead. On basis of this auxiliary data more efficient paths could be planned. Possible objects, such as tree lines, can be detected before they are in the range of the stereo-vision system. The Robojeep can also gain a better notion of where roads or paths are heading. This could be used for new applications such as dirt road following.

Other features seem unable to match the color features classification performance. Color is able to combine various aspects that are important in clas-

sifying terrain for autonomous navigation:

- clear distinction between terrain types;
- performs in a large distance range, and
- no time consuming feature extraction operations have to be performed.

It is very difficult to classify terrain types on only geometry or motion features. This is also valid for thermal images of infrared cameras. Texture features are more promising, however, their performance is too low. Also, if stereo-vision provides the texture algorithm with scale information, still several time consuming feature extractions have to be performed.

Mixtures-of-Gaussians can model color distributions in certain color spaces very well and they can provide a confidence measure. The EM learning algorithm is sufficient to learn the mixtures and classify new images. DEMO III successfully implemented this and also satisfying the real-time constraint [1].

The following chapter presents our method and adds a valuable feature: environment auto-calibration.

Chapter 3

Method

In this chapter a method is proposed that could satisfy the problem statements constraints, as discussed in section 1.2. It is an elaboration of JPL's work: colors of the environment are learned in RGB-space using the EM algorithm [1]. Unfortunately, their method relies on advice from an external reference to deal with larger chromatic variations. In an overview of their recent research [20], they still consider the color shift caused by atmospheric conditions as an open problem. These chromatic variations can occur very fast. On windy days the environment conditions can change from sunny to clouded to rainy within the hour. If the system must be able to act autonomously, the environment advice must come from an internal reference. The method described here performs an environment auto-calibration step before classifying the terrain types.

First an overview of the method will be sketched, followed by a terminology introduction. Before describing each component of the method, the environment state assumption is described.

3.1 Method overview

We chose to develop an environment auto-calibrated color recognition algorithm. From the Lambertian reflectance (equation 2.2) it is know that the variance of the apparent color of an object can be very large. Therefore, before classifying a new instance it is important to know the state of the environment. By calibrating the obstacle classification on the environment state, more robust results can be obtained.

The environment calibration algorithm can be split into three components, which are depicted in figure 3.1.:

1. the environmental cluster component;
2. the learning component, and
    - environment learning algorithm;

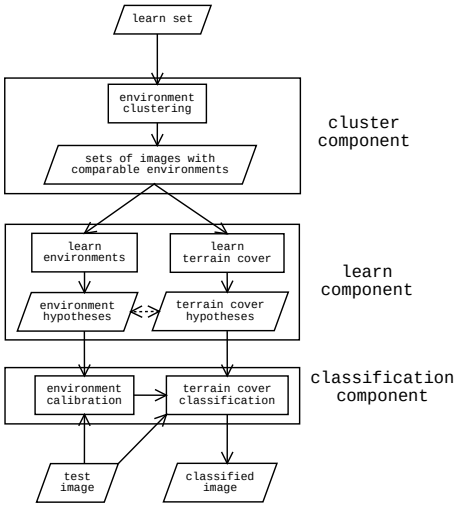


Figure 3.1: Component overview.

- terrain cover learning algorithm.

3. the classification component.
   - environment calibration;
   - terrain cover classification.

The first component groups images based on their environmental states. For example, sunny images in a particular scenery group together. These sets of images can be input to the learning component.

The second component learns environment hypotheses and terrain cover hypotheses. Input to both learning algorithms are the sets of images found by the cluster component or by an outside reference. Each image set should represent a specific environment state. Because both learning algorithms have the same input sets, and thus the same environment state λ, environment hypothesis $h_e(\lambda)$ is linked with cover hypothesis $h_c(\lambda)$. The sets H_c and H_e contain all cover and environment hypotheses.

The classification component classifies the terrain using H_e and H_c. The calibration part sends information about the environment to the terrain cover classification part. With this information appropriate subset of terrain cover hypotheses H_c can be selected based on environmental states. This subset can be used to classify new images. Output of this component is a terrain cover image in which the pixel value defines a label.

3.2 Terminology

This section introduces various data representations and terms used in the following sections. Figure 3.2 shows three data representations that are frequently used, namely:

Images and masks An image is the output of a camera. It is a two dimensional projection of 3D objects in the world. Each pixel in the image is encoded by its color using three color channels. The camera outputs images in RGB-format, but after transformation the color channels can be in other formats as well. Images are denoted by capitol letters, for example I.

The terrain cover masks are manually labeled images that define to which terrain cover each image area belongs. In such an image a one represents a pixel that is of the specified class, and a zero represents all other pixels. The locations of the ones in the classification masks correspond with the locations of a particular class in the original image.

Masks can also be generated, for example variance masks. In this case the local variance of an image is calculated. For all intensity pixels in a small local window A the variance σ^2 is calculated according to

$$\sigma^2 = \frac{1}{n-1} \sum_{i \in A} (i - \bar{i})^2, \tag{3.1}$$

with pixel intensities i. Image edges that are outside the filter range are padded with zeros. The application of a variance filter on a intensity image I_I resulting in a variance image I_v is notated as $I_v = F_v(I_I)$.

To create a mask out of an image I_I, a threshold operation is performed, denoted by $M = T_h(I_I, \min, \max)$. This operation sets all pixels in mask M to 1 if they are in the range [min, max]. All other pixels are set to zero.

Instances When an image I is transformed to a colorspace, it is called an instance and is denoted by $\hat{i}$. A mask M can select what points are transformed to an instance, denoted by $\hat{i}^M$. An instance has three dimensions, one for each color channel. A point in this space thus represents a particular color. In figure 3.2 the LAB-space is used. Note that there are 3 clusters: one for the sky class and two for grass class. Such clusters are called elementary clusters. A class cluster $\hat{i}^c$ contains all points that are labeled class c. In case c = grass, the class cluster contains two elementary clusters. One elementary cluster corresponds to grass that is illuminated by direct sunlight, and the other corresponds to grass illuminated by indirect sunlight. A union of multiple instances $\hat{i}_1 \cup \ldots \cup \hat{i}_n$ is called a generalized instance. With the $\cup$-operator performed on instances a concatenation of points is performed, not a union.

Modeled instances An instance $\hat{i}$ can be transformed to a modeled instance i by fitting the class clusters by mixtures-of-Gaussians. Each mixture

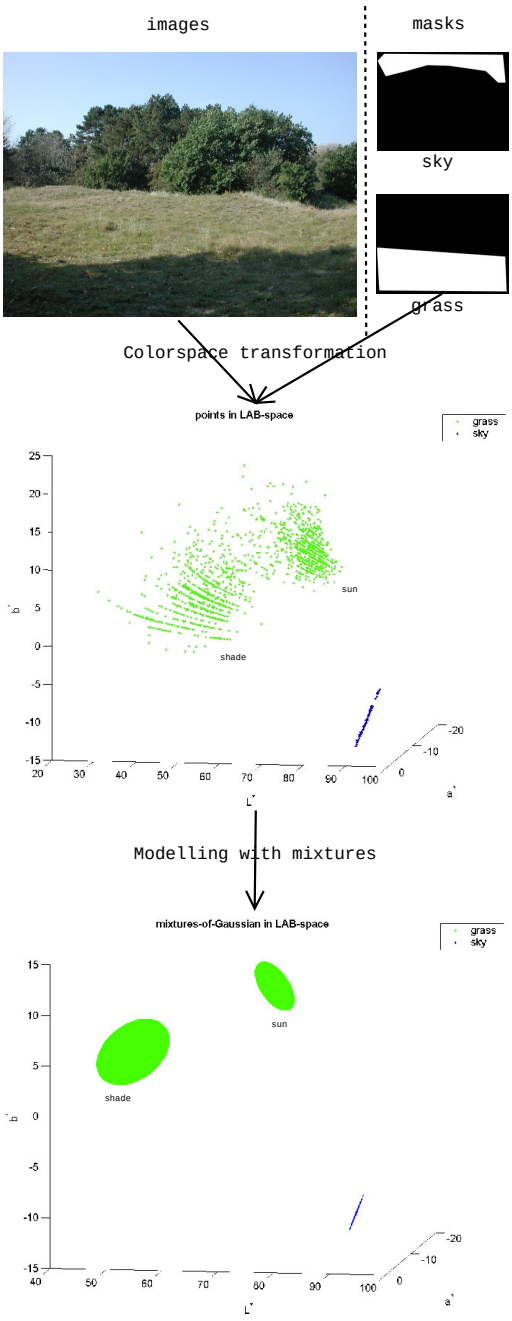


Figure 3.2: Data representation.

i^c thus represents a class. A hypothesis is a special case of a modeled generalized instance. A hypothesis h with environment state λ contains all modeled class instances, so $h(\lambda) = \{i^1, \ldots, i^n\}$, with n the number of classes. Figure 3.2 shows the three elementary clusters fitted by two mixtures. The first mixture models the sky class with one Gaussian. The second mixture models the grass class with two Gaussians: one for the sunny component and one for the shaded component.

3.3 Environment states assumption

For identifying the environment states, stable color features are needed. When the terrain type or illumination changes, the variance increases. Another source of variance can be the 3D geometry. In areas with complex geometry the variance is high. On flat surfaces it can be low. These reasons explain why the areas with low variance yield small and well separated clusters in colorspace, illustrated by figure 3.3. We assume that the colors of the low variance image regions are heavily influenced by the environment state, and thus are important features for estimating the environment state.

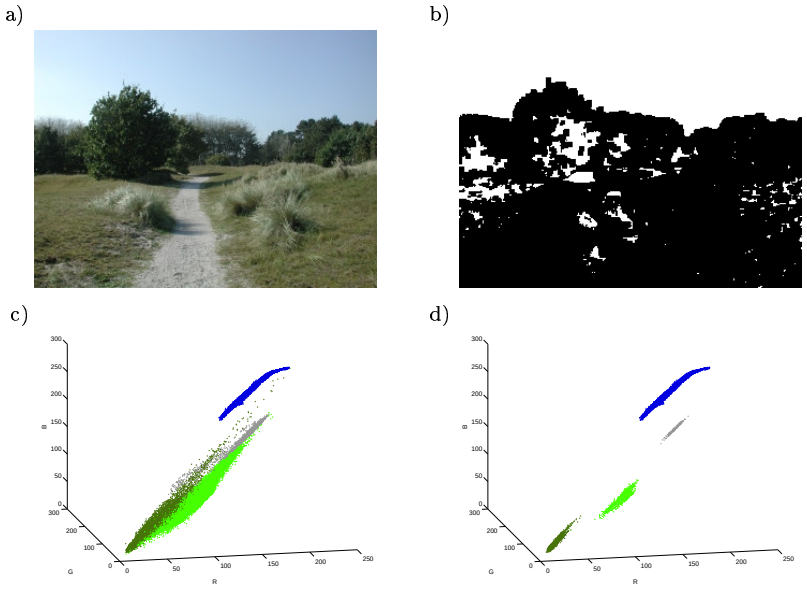


Figure 3.3: An image (a) with its variance mask (b) as well as the points in RGB-space (c) and the variance filtered points in RGB-space (d). The shown classes are grass (light green), foliage (dark green), gravel (gray) and sky (blue).

The position of a terrain class with low variance in colorspace is thus mainly due to the class color and the illuminator. The similarity of low variance clusters of a new image to clusters in a database, informs about the environment state of the new image. Thus, the color calibration "rig" is the scenery itself. In all

components of our method these areas of low variance are essential.

3.4 The cluster component

Input to this component is a set of labeled images I_{set}: the original images with their classification masks. These images can be shot in different environment states, and this component will group the images with similar environment states. With respect to the environment state, this clustering is unsupervised. The clustering component is subdivided in three parts: the preprocessing steps, the distance calculation and iterative clustering steps. A flowchart is shown in figure 3.4.

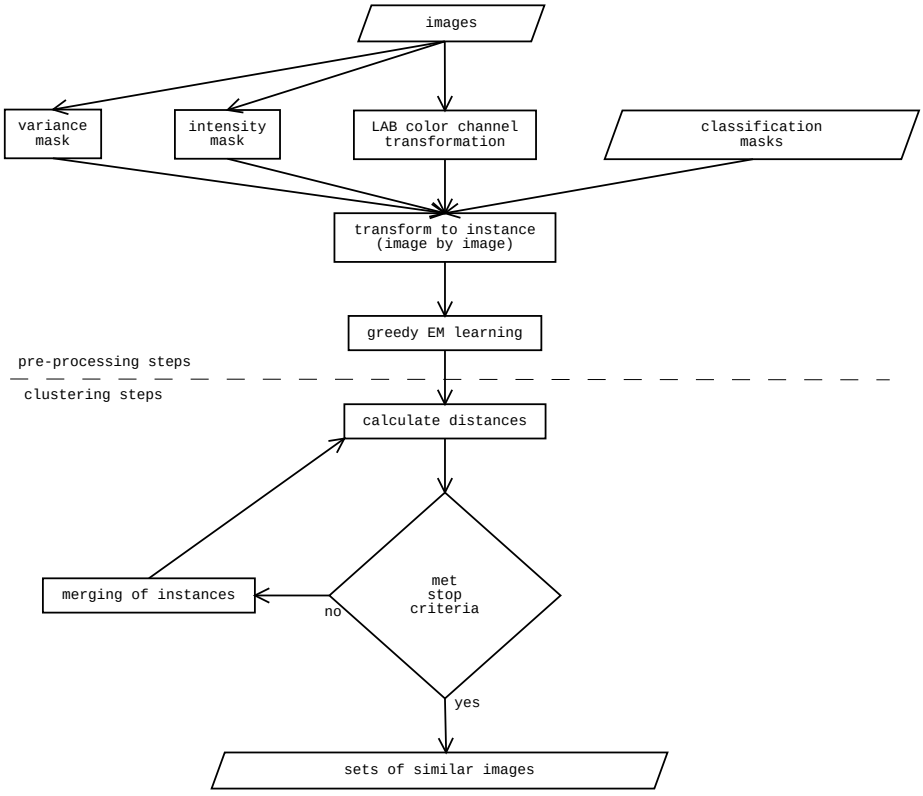


Figure 3.4: The cluster procedure.

3.4.1 Preprocessing steps

The images are subjected to three processing steps: variance thresholding, intensity thresholding and a color channel transformation. In the first two steps the RGB images are transformed to intensity images I_I. The operation

$M_v = T_h(F_v(I_I), 0, v_t)$ calculates variance masks M_v for an intensity image I_I using a constant variance threshold v_t. This mask is selects those pixels which are able to estimate the environment state.

The second step performs the operation $M_i = T_h(I_I, 0, i_t)$, creating a intensity mask using the constant i_t. Pixels with high intensity values are discarded because the sky tends to be saturated. In figure 3.3 the sky clusters are bend because of the saturation. These clusters with maximum intensity value causes distinct sky clusters to come to close to each other and obstruct the cluster process.

The final step performs a color channel transformation. Because the pixel data is still in RGB format the colors are not comparable: the distance in the colorspace is not a fair guide to the difference in color perception. In order to be able to apply a distance measure successfully, it is essential to transform to color channels in which objects are comparable. The LAB-space, discussed in section 2.2.2, is such a space. This transformation is applied on all the images in the learning set.

The next step transforms the images to instances represented in LAB-space. For each class in an image a selection mask is generated by intersecting between the variance mask M_v, the intensity mask M_i and the requested label mask M_c. This selection mask picks only those pixels of the requested class c, have variance lower than v_t and intensity values lower than i_t. For every class, the pixels in the images are transformed to points in 3D LAB-colorspace. So for every image I_{LAB} in I_{set} the following operation is performed:

$$\forall c \in C : \hat{i}^M, \text{ with } M = M_c \cap M_v \cap M_i.$$

Two steps are performed before fitting class clusters in the instances with mixtures-of-Gaussians and thus transforming them to modeled instances. First, all clusters with less than a certain number of points are discarded. This action makes sure that no Gaussians are fitted on too little data points. The second action is the cross-validation step. The points in the clusters are randomly divided between a learning set and a test set. An upper bound on the number of points can be given for both sets. After these steps are performed, the greedy EM learning algorithm is invoked. This algorithm is described in section 2.6.4. Input to this algorithm is the learning set, the test set, the maximum number of components and the number of candidates per component. To estimate environment states we need to assure that one cluster specific for an environment state is modeled with one component of a mixture-of-Gaussians. Therefore no elementary clusters of one class are modeled if the distance between there means is smaller than a specified value. Notice that the by using the greedy EM algorithm it is not necessary to specify the number of components per mixture, which is essential for the clustering algorithm. Output of the learning algorithm are sets of mixture-of-Gaussians. Each mixture represents a class cluster, and each set represents an instance.

3.4.2 Distance calculation

The distance calculation between two modeled instances returns two measures. This is because the distance between two Gaussians can only be calculated if both Gaussians are of the same class. The first measure is the distance measure, and the second measure is the unmatched measure. The distance measure returns the mean distance between corresponding elementary clusters in two instances. The unmatched measure counts how many Gaussians of a particular class are present in one instance, that are not present in the other instance. For example: if the first instance does not contain a grass cluster and the second does contain one consisting out of two elementary clusters, then its unmatched value is two. Otherwise, if the first instance would only contain one elementary grass cluster, its unmatched value would also be one.

In the following sections we describe the distance calculations. The distance between two Gaussians is described first. This distance is used to calculate the distance between two mixture of Gaussians. Finally, the distance between two instances is calculated.

Distance between Gaussians

The distance between two Gaussians is calculated by the Bhattacharyya distance

$$D_b = \frac{1}{8}(\mu_1 - \mu_2)^T \left[\frac{1}{2}(\Sigma_1 + \Sigma_2)\right]^{-1} (\mu_1 - \mu_2) + \frac{1}{2}\log \frac{\left|\frac{1}{2}(\Sigma_1 + \Sigma_2)\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}, \qquad (3.2)$$

in which μ_1 and μ_2 are the means of the normal distributions, and Σ_1 and Σ_2 are the corresponding covariance matrices.

Distance between mixtures-of-Gaussians

The distance between two mixtures-of-Gaussians f^c and g^c with the number of components respectively equal to n_f and n_g is calculated by first calculating all possibilities. These possibilities are stored in the matrix D^c,

$$D^c(f^c, g^c) = \begin{pmatrix} D_b(f^c(1), g^c(1)) & \dots & D_b(f^c(1), g^c(n_g)) \\ \vdots & \ddots & \vdots \\ D_b(f^c(n_f), g^c(1)) & \dots & D_b(f^c(n_f), g^c(n_g)) \end{pmatrix}. \qquad (3.3)$$

The smallest n distances in D^c are returned in a vector v^c with size n, with $n = min(n_f, n_g)$. Intuitively, n is the number of matches between components when a component can only be matched one time. For example, if $n_f = 3$ and $n_g = 2$ then g has only got two components to match two of the three from f. The distances between the matched components is returned, as well as the number of unmatched components for each mixture. These are found by $u_f^c = n_f - n$ and $u_g^c = n_g - n$. In case the u^c values are negative, then they are set to zero.

Distance between instances

Now we have the tools to calculate the distances and the number of unmatched Gaussians between two modeled instances: i and j. Both instances contain a set of mixture-of-Gaussians, where each mixture corresponds to a class cluster. According to the method described above, all distances and unmatched clusters are calculated for mixtures of the same class. The distances v^c are concatenated to a vector V. The number of unmatched u_i^c and u_j^c are respectively summed in u_i and u_j. For classes that are present in i but not in j the number of mixture components is added to u_i, which is also done vice versa. This procedure returns two variables: the mean distance $d(i,j) = \overline{V}$ and the minimum number of unmatched mixture components $u(i,j) = \min(u_i, u_j)$.

3.4.3 Iterative clustering steps

The merging procedure can be subdivided in two parts. The first part selects what instances to merge. The second part does the actual merging.

Select instances

The first step in selecting what instances to merge is the distance calculation over all instances. These distances are stored in a set of sparse matrices δ,

$$\delta_k = \begin{pmatrix} 0 & d(i_1,i_2) & \dots & d(i_1,i_n) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & d(i_{n-1},i_n) \end{pmatrix}, \tag{3.4}$$

where k the minimum number of unmatched mixture components and n is the number of instances. Thus all distances in δ_k are subject to the constraint $u(i,j) = k$ for all i and j in δ_k.

Three user defined thresholds guide the merging procedure. The first is the merging threshold T_m. This threshold specifies the maximum distance two instances may have. To merge two instances i and j, the constraint $d(i,j) < T_m$ should be satisfied.

The second threshold is the maximum unmatched threshold T_u. This threshold specifies the maximum number of unmatched elementary clusters for the merging of two instances. To merge two instances i and j, the constraint $u(i,j) < T_u$ should be satisfied.

Finally, the third threshold is the leave-one-out threshold T_l. This parameter deals with high distance values between two instances that are due to a wrong matching of Gaussians. For example, this is the case when a distance is calculated between two elementary grass clusters: one in the sun and the other in the shade. If the number of matched elementary clusters of an element in δ_k is higher than the leave-one-out threshold and the distance is higher than the merge threshold, the following operation is performed: the mean distance is calculated without using the highest distance between two components and this distance is inserted in δ_{k+1}.

The merging procedure starts with $k = 0$. The two instances are selected that have the smallest distance to each other. If this distance is smaller than the merge threshold, then both instances are merged and k is set back to zero. If this is not the case, k is raised by one, and the instances with the smallest distance in δ_k are selected. The merging procedure stops when k is equal to the maximum unmatched threshold T_u.

Merge instances

Suppose the two modeled instances i and j are selected to be merged. Both modeled instances are removed from the set of modeled instances. A new instance $\hat{h}$ is created by $\hat{i} \cup \hat{j}$. After modeling $\hat{h}$ with mixtures-of-Gaussians, h is added to the set of modeled instances.

Output of the cluster component are $|\Lambda|$ sets of images names, where Λ is the set of environment states.

3.5 The learning component

The learning component learns coupled environment and terrain cover hypotheses. Both hypotheses have $|\Lambda|$ sets of modeled generalized instances, where Λ is the set of environment states. The set of environment hypotheses is denoted H_e and the terrain cover hypotheses set is denoted H_c. The hypothesis $h_e(\lambda)$ is linked with hypothesis $h_c(\lambda)$. If $h_e(\lambda)$ represents dune landscape on sunny days, then $h_c(\lambda)$ also represents dune landscape on sunny days. The learning component is depicted in a flowchart in figure 3.5. The steps of this learning component are performed for all sets of images.

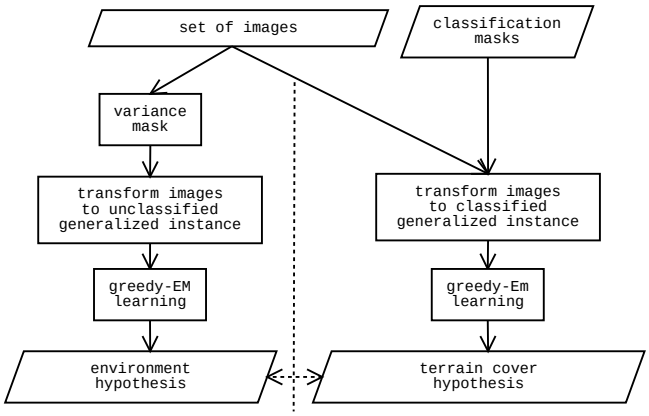


Figure 3.5: The learning procedure

Input to the learning component is a subset I_s of images from I_{set} where every image in I_s has the same environment state λ. This set can be output of

the cluster component, but it can also be defined by an outside reference.

Environment hypothesis

Before the calibration hypothesis can be learned, masks M_v have to be generated that can select areas of low variance. This operation is the same as described in the cluster component section.

The next step transforms the images to one generalized instance. In this step only the variance masks are used, so the pixels have not got a label assigned to them anymore.

The final step learns the environment hypothesis $h_e(\lambda)$. First, the points in the generalized instance are randomly divided between a learning set and a test set. An upper bound on the number of points can be given. Next the greedy EM learning algorithm is invoked. This algorithm is described in section 2.6.4. Input to this algorithm is the learning set, the test set, the maximum number of components and the number of candidates per component. Output is one mixture-of-Gaussians that represents the environment hypothesis $h_e(\lambda)$: a modeled unclassified generalized instance.

Terrain cover hypothesis

The learning of the terrain cover hypothesis is essentially the same as described by Bellutta [1]. It is also very similar to the learning of the environment hypothesis. In case of the terrain cover hypothesis also one generalized instance $\hat{h}_c(\lambda)$ is created.

Here, sets of pixels for each existing label are created using only the label masks c'. These masks select only pixels that are of the requested class. For every image and for all their labeled masks a selection is added to $\hat{h}_c(\lambda)$. The resulting generalized instance $\hat{h}_c$ is still classified: $\hat{h}_c^{c'}$ contains all points of class c'.

Finally the terrain cover hypothesis $h_c(\lambda)$ is learned. The points of each class are randomly divided in a learning set and a test set. An upper bound on the number of pixels can be given for each set of points. The greedy EM algorithm is invoked using the same input parameters as in the environment hypothesis case. Output is the modeled generalized instance $h_c(\lambda)$. This is a set of several Gaussian mixtures: one mixture $h_c^{c'}(\lambda)$ for each class c'.

3.6 The classification component

The last part of the method is the classification component. Its function is to classify an unlabeled sequence of images $U_1, \ldots, U_n$. For each input image, two images are produced: a classification image C and a certainty image P. In the classification image the pixel values are integers that correspond to the derived classes: it defines what terrain classes the areas in the input image are. In the certainty image the pixels values are real values that correspond to a certainty measure of the derived classes. The classification component is subdivided in

two parts: the terrain cover classification part and the calibration part. The classification component is depicted in figure 3.6. The left hand part shows the terrain cover classification part from top to bottom. In the right hand side the calibration part is depicted from bottom to top.

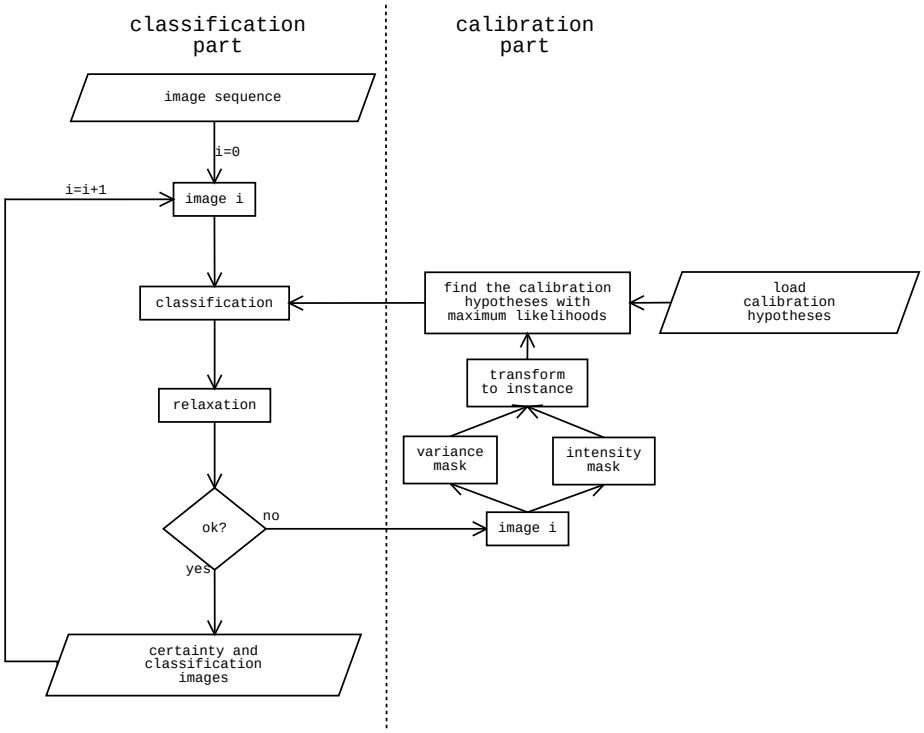


Figure 3.6: The classification procedure.

3.6.1 Calibration part

Although the classification component does not start with the calibration part, it is still explained first because of clarity reasons. To determine which terrain cover hypothesis should be used for image U, the image is first transformed to an instance $\hat{u}$. The instance is transformed to the same format as the instances used in learning the environment hypotheses, thus masks that define low variance areas are generate as well as intensity masks. The same variables are used for these operations as have been used in the learning of the environment hypothesis.

The final process calculates the likelihood of the points set $\hat{u}$ with every hypothesis in h_e. The likelihood of environments is returned in a vector where $E_\lambda = P(\hat{u}|h_e(\lambda))$, where λ is refers to the environment state.

3.6.2 Classification part

The image sequence $U_1, \ldots, U_n$ is read one image at a time. On the initial run of the algorithm the classification hypothesis is not yet specified. This hypothesis is called the 0-hypothesis and its special case is explained in the discussion of the decision node.

Calculating class likelihoods

Generally, the calibration part has determined what environment hypotheses performs best. Output of the calibration part was a vector E, that describes the likelihood of each environment state. A simple and computationally fast option is to select one terrain cover hypothesis. This cover hypothesis is linked with the environment state λ that has maximum likelihood $\max(E)$. The new image U is transformed to points in RGB-space $\hat{u}$. For each point $\hat{u}'$ in $\hat{u}$ and for each class c', all likelihoods in the terrain cover hypothesis are calculated: $l^{c'}(\hat{u}') = P(\hat{u}'|h_c^{c'}(\lambda))$. Each pixel now has $l^1, \ldots, l^n$ likelihoods assigned to it, with n the number of labels.

This method of finding the label likelihoods belonging to a pixel has some disadvantages. Because the classification solemnly relies on one environment state, catastrophic classification results can occur if a wrong environment state is selected. Also, because the environment states of the to be classified images are rarely the same as the learned states, there will be suboptimal classification results.

Our second method of calculating class likelihoods can select more than one environment state. An environment state λ is selected in Λ' if it satisfies $E(\lambda) \leq \max(E) \cdot (e_t/100 + 1)$. This operation selects all states that lie within $e_t\%$ of $\max(E)$. Every state λ' in the set Λ' is used to classify the new image. The likelihood that a pixel in U belongs to class c is calculated by

$$l^c(\Lambda') = \sum_{\lambda' \in \Lambda'} l^c(\lambda') \frac{E(\lambda')}{\Theta},$$

with $\Theta = \sum_{\lambda'' \in \Lambda'} E(\lambda'')$. In other words, the likelihood is weighted by the averaged likelihoods of the environments. If this step is performed for every class and pixel, the same number of likelihoods is returned as in the maximum likelihood method.

In the next chapter we will present results showing that selecting more environments states for classification has both benefits and a disadvantage.

Relaxation step

The classes with the highest likelihoods maximum likelihoods could directly be used for classifying a new image U. However, classification based on one pixel is noise sensitive. To attenuate this noise, a median filter can be performed for each class likelihood. The class likelihood value l^c of pixel (i, j) is subjected to

a median operation with rectangular box of size m of the same class likelihood values l_c around (i, j).

Calibration decision

After the relaxation step, it is determined if the classification is valid. This takes two subsequent steps. First, for each pixel the maximum likelihood l_m out of the set $l^1, \ldots, l^n$ is selected. These values are summed to yield a likelihood l_M over the entire image. The second step actually determines if the classification is valid. If the constraint $l_M < T_{ll}$ is not satisfied, the calibration part is invoked. In this constraint T_{ll} is a manually set threshold. The 0-hypothesis always false, so that in the initial case the calibration part is always called.

Output

The final operation outputs the classification image C and the certainty image P. The classification image C contains labels that correspond to the derived classes. For each pixel in U the class is selected that has the highest likelihood. For each pixel in the certainty image P the values are calculated by $\max(l^1, \ldots, l^n) / \sum(l^1, \ldots, l^n)$, thus providing a confidence measure.

Chapter 4

Experiments

This chapter is divided in two parts. The first part discusses the performance evaluation. Here the used datasets, labeling, taxonomy and the performance metrics are presented. The second part describes three experiments concerning our auto-calibration method and discusses their results. These experiments show how well the environment clustering performs and how well the environment estimation performs. Finally an experiment is performed that compares our environment auto-calibration method with a method that is given the environment state. Moreover, this test evaluates how well our method classifies the terrain. In conclusion, a post-processing step is reported that could boost the algorithms performance.

4.1 Performance evaluation

4.1.1 Datasets

Several datasets where recorded to evaluate our auto-calibration algorithm. This section describes the datasets in terms of: content, weather conditions, specifications, classes, training- and testsets.

All images are collected with a camera in the visible spectrum. The used camera is the Nikon Coolpix 990 set at a resolution of 2048×1536. These images are subsequently scaled to images of size 320×240 using a linear interpolation method.

The images are shot at the Waalsdorpervlakte, a dune scenery nearby the city of The Hague in The Netherlands. The environment contains several natural materials and obstacles. Most frequently occurring are sand, grass, bushes, gravel and trees. Three datasets where recorded in different weather circumstances: one with clear sky, one with an overcast, and the last one when it was drizzling. The number of images in the sets are respectively: 33, 28 and 28. The images in the datasets contain a number of different situations typically encountered while driving in vegetated terrain.

Each dataset is split into a training- and testset to obtain image sets that are as diverse as possible. These sets are not only used to find the best fitting mixture-of-Gaussian model using the expectation maximization algorithm, they are as well used for a quantitative performance measure.

4.1.2 Class selection and labeling

Taxonomy decisions should be driven by the application, selecting as many classes as deemed appropriate for the task to be carried out. However, a large taxonomy may simply not make sense, if the classifier is unable to discriminate among such a variety of classes. Such a situation occurs when a human would label two regions that have the same color but have different classes.

Six classes of interest have been identified, namely: grass, foliage, sand, gravel/stone, sky and uncertain. The following arguments, state why this set of classes is of interest:

- by finding sand or gravel, dirt roads can be followed;
- it is important to be able to distinct between grass and soil types, such as gravel, sand and stone, because grass can be misclassified by the stereo-vision system as an obstacle.
- finding tree lines has a navigational benefit in the sense that the vehicle will not explore paths that come to a dead end anyway;
- the ability to distinct between sand and gravel/stone classes is important for controlling the velocity to ensure the safety of the vehicle.
- the sky class is a valuable feature for estimating the environment states, because
  1. the sky is robustly and accurately detectable by the classifier;
  2. the sky is almost always present in every image, and
  3. the sky has distinct positions in colorspace depending on the weather circumstances;
- the uncertain class contains all pixels that can not be safely classified into any of the other classes, and thus have a certainty measure lower than a specified certainty threshold t_c.

This taxonomy does not only distinct between drivable and non-drivable areas, it also considers navigational and vehicle control applications. A smaller taxonomy set in combination with a stereo-vision system can be sufficient to distinct between the drivable and non-drivable areas. This smaller set contains three classes: grass/foliage, sand/stone/gravel, sky and uncertain. Green colored objects are penetrable if they are below a certain height threshold. In this case, the green objects can be low grass, patches of high grass or small bushes. If the objects exceed the height threshold, the green objects could be

larger bushes or trees. Note that the threshold must be set conservatively. If the area is classified as sand/stone/gravel, the area is more rigid and thus drivable if the obstacle is not to steep.

The labeled masks, mentioned in the method chapter, assign a class to an image area. These are created by labeling an image area by hand. Not all parts of an image have been hand-labeled because there are situations where any of the chosen classes does not represent the area well. This is especially the case near region boundaries, where an area contains colors from more than one class, and therefore should not be used for learning.

4.1.3 Performance metrics

The classification results on the labeled regions are collected in suitable matrices: so called confusion matrices introduced by Castano et al. [6]. The entry, $CM_{j,k}$ of such a matrix represents the number of times a pixel labeled as j has been classified as k. The confusion matrices provide three quantitative performance measures. The first quantitative assessment measure is the number of correctly classified points over the total number of points in the complete training set:

$$P_C = \frac{\sum_j CM_{j,j}}{\sum_{j,k} CM_{j,k}}.$$

An estimate of the probability of correct classification for each class j is given by:

$$P(C|j) = \frac{CM_{j,j}}{\sum_k CM_{j,k}}.$$

A problem with the global measure P_C is that it is biased toward the the class that occurs most in the training set. Therefore another global measure is introduced that gives the probability of correct classification assuming that the classes are equiprobable. This measure $\overline{P}_C$ corresponds to the average of $P(C|j)$.

4.2 Experiments

This sections discusses the experiments and results of our auto-calibrated terrain cover classifier. The first test discusses how the environment cluster component performs. The second experiment discusses the results of the environment assessment. The final test compares the our calibrated environment method with a method that is given the environment state. This experiment evaluates how well our method classifies the terrain types as well. After these experiments, a post-processing method is discussed to boost the performance.

Before these experiments were conducted, a small test was performed comparing the RGB, rgb, opponent, Otha and LAB spaces. The result of this experiment was that no colorspace increased the performance over the RGB-space. JPL concluded the same in their research on terrain cover classification [1]. In

the following experiments the cover hypotheses are represented in RGB-space. The cluster component still uses LAB-space.

4.2.1 Environment cluster experiment

In this experiment the cluster component is tested with regard to its performance and robustness. The task of this component is to group images with similar environment states. The results of the final terrain cover classification depend a great deal on the environment clustering.

If only images are merged that are very similar, there will be a lot of clusters containing few images. A disadvantage of a small similarity threshold is that some terrain types will not be represented well in the cover hypotheses. However, only grouping very similar images has the benefit that sub-environment types can be found. This is an advantage because the environment state chances significantly if the vehicle drives into a forest. Also, environment states that should not be part of any environment set can be filtered out. If images that are less similar are also grouped, there will be less clusters containing more images. In this case, the cover hypotheses are more likely to have a good representation of each terrain type.

Note that these arguments place the environment "ground-thruth" as a subject under discussion. The extra degree of freedom provided by the environment cluster component could make the terrain cover classification perform better than a classification based on the ground-thruth environment states. This is the reason we prefer the term given environment state over the term ground-thruth.

The merging of similar images is controlled by the merge threshold T_m, described in section 3.4.3. A small merge threshold will only merge images that have a small distance to each other. Other parameters that guide the cluster process, described in the same section, are the maximum unmatched threshold T_u and the leave-one-out threshold T_l. The maximum unmatched threshold describes how many unmatched mixture components are allowed for clustering. The leave-one-out threshold controls how many matched components are needed to discard a large distance between components.

Only the merge threshold T_m is varied in this experiment. The other thresholds T_u and T_l are set to the constant value 2. The terrain classification component interpolates between environment states using $e_t = 10$. So all environment states within 10% of the maximum likelihood are considered for classification. The Waalsdorpervlakte dataset is split into a training and testing set. Environment clustering and terrain cover learning are performed on the training set, while terrain cover classification is performed on the test set.

Table 4.1 presents the clusters found for different T_m values. As expected the number of clusters gets lower when the merge threshold increases. At most three images are part of an environment set that was not present in the original sets.

Figure 4.1 shows P_c and $\overline{P}_c$ measures for an increasing T_m. When there are a lot of small clusters, the P_c value is high and $\overline{P}_c$ is low. This can be explained as

T_m	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
sunny (16)	14×1, 1×2	5×1, 1×3, 1×8	3×1, 1×2, 1×3, 1×8	1×1, 1×2, 1×12	1×2, 1×15/1	1×2, 1×16/2	1×2, 1×15/1	1×1, 1×16/1	1×43
cloudy (14)	10×1, 1×4	2×1, 1×2, 1×4, 1×6	1×1, 1×2, 1×11	1×1, 1×2, 1×13/2	1×1, 1×13/1	1×13/1	1×13/1	1×13/1	1×1
rainy (14)	4×1, 1×2, 2×4	1×1, 1×4, 1×9	1×1, 1×13	1×13	1×13	1×13	1×13	1×13	0

Table 4.1: The resulted environment clusters for varying merge threshold, where $n \times m/o$ mean n clusters of m images of which o do not belong to the original state.

follows. The classes that occur frequently are well represented in the hypotheses and are classified correctly. The classes that occur less frequently are not that well represented. The conditional probabilities, on the right hand side of the same figure, show that $P(C|\text{gravel})$ and $P(C|\text{sand})$ have much lower values at $T_m = 0.5$ than at $T_m = 3$. It is impressive though that the interpolation of environment states has such a high P_c value as a result.

At $T_m = 0.5$ the difference of the maximum likelihood to the other environment likelihoods is small: most hypotheses do not represent the data well. When the clusters become larger, more classes are represented better. The gravel and sand class probabilities rise, and with them rises $\overline{P}_c$. In contrast, P_c descends due to a drop of the conditional probability of grass. Because grass areas have much low variance values than foliage areas, the auto-calibration algorithm selects mostly hypotheses in which grass is well represented. For higher T_m values more foliage is present in the cover hypotheses. Section 4.2.3 shows that a considerable amount of grass pixels is misclassified as foliage, thus for higher T_m values, grass has more competition from foliage.

Between T_m values 2 and 4, the main environment states are all fully grown: each class is represented well and the maximum P_c and $\overline{P}_c$ values are reached. For comparison, if the three original clusters are used for learning the probabil-

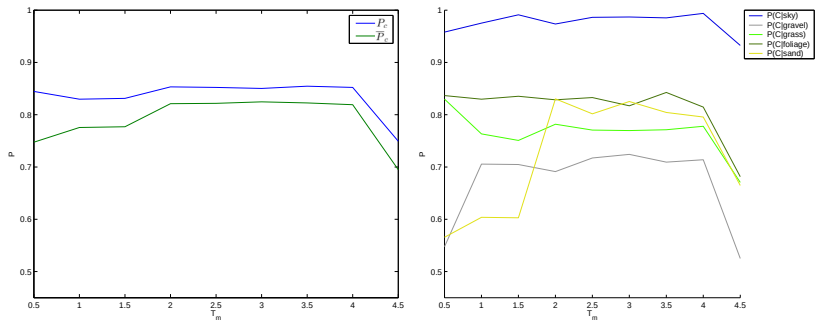


Figure 4.1: The left figure shows the probabilities P_c and $\overline{P}_c$ with varying merge threshold T_m. The right figure shows the corresponding terrain class conditional probabilities.

ities are: $P_c = 0.84$ and $\overline{P}_c = 0.83$. Figure 4.1 shows that these are comparable to the generated environment sets.

At $T_m = 4.5$, the original environment sets are merged to one set in which all environments are represented. Wrong environment states are used for classifying new images, which explains the drop in all probabilities.

The cluster component performs very well for values of T_m between 2 and 4. In this wide range the terrain cover probabilities P_c and $\overline{P}_c$ are higher than 0.8, indicating robust environment clustering results. Unfortunately no sub-terrain set where found that improved the classification results. Some images where filtered out, but this had no significant effect on the final results.

4.2.2 Environment estimation experiment

This experiment investigates the auto-calibration algorithm. Incorrect environment estimations have a large negative impact on the results of the terrain cover classification. This experiments tests the performance and robustness of the auto-calibration algorithm.

The auto-calibration algorithm was run with different variance threshold values. The variance threshold v_t selects the image areas that have low enough values to be suitable for estimating the environment state. If only a few areas are selected, some classes will not be used for environment estimation. If too many areas are selected, a class will spread out more in colorspace and will not represent the typical class color in an environment state. Both cases will reduce the number of correctly estimated environment states.

The environment and terrain cover hypotheses are learned for every v_t value. Classification is performed with corresponding values. The original dataset of the Waalsdorpervlakte is used. Again, learning and testing are performed on different sub-datasets. Only the environment state with the maximum environment likelihood is used for classification. Figure 4.2 shows the percentage of correct environment classifications. Note that for the same reasons as described in the previous experiment, the notion of "correct" environment state is somewhat a subject under discussion. In this experiment the original clustering can serve as a ground-thruth without out any problems though.

As stated before, if v_t is too small then not enough calibration areas are used for successful environment assessments.

The constant correctness percentage of about 85% at $v_t < 5$, can be explained as follows. With this low variance threshold, only sky areas are selected for the environment assessment. The sky clusters in different environments are well separated from each other, leading to stable environment estimations.

More classes are selected for environment assessment when $v_t > 5$, improving the percentage of correctly estimated environments. The auto-calibration algorithm misclassified at most 2 out of 44 images for v_t values higher than 15. Because most classes are less well separated than the sky class, some noise is introduced due to random processes in the learning and estimating of the environment states.

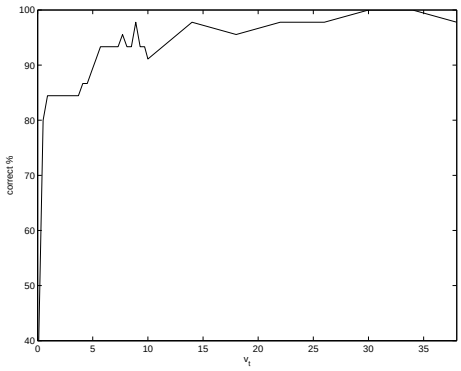


Figure 4.2: The probability for an altering variance threshold.

The percentage of correct environment assessments remains high for very large v_t values. The performance reduces only little if no variance threshold is used for assessment of the environment state. Probably the areas of low variance coincide in colorspace with high density areas. The greedy EM algorithm learns these densities, taking over the the role of the v_t parameter.

The auto-calibration algorithm has high performance values in a wide range of variance thresholds, making it a robust algorithm. The percentage of correct environment estimations does not drop under 95% for variance thresholds higher than 15.

4.2.3 Terrain classification experiment

This experiment investigates the performance of our terrain classification component. Its performance in distinguishing between drivable and non-drivable areas is considered, as well as its performance in distinguishing between terrain classes suitable for navigational tasks, such as dirt road following. Our auto-calibration method is compared to a method that is given the environment state.

The method that is given the environment state is our implementation of the color classification algorithm of JPL's autonomous vehicle DEMO III [1, 20]. This method uses no environment calibration, so it is given the environment state belonging to the original set of the new image. In other words, the environment "ground-thruth" is given. For learning JPL uses the standard EM algorithm with 3 to 5 components per class. In our implementation of their algorithm 4 components per mixture are used.

The results of our implementation of the DEMO III approach are compared with two variants of our environment auto-calibration method. The first method selects the terrain cover hypothesis belonging to the environment with maximum likelihood. The second variant interpolates between the cover hypotheses using $e_t = 40$.

The hypotheses all learn and test on the same Waalsdorpervlakte datasets. The DEMO III method uses the original clustering, while our methods use the clustering of merge threshold value 2.5 which is generated in the environment cluster experiment.

Quantitative results

The probabilities of the methods are shown in table 4.2. In this test, the standard EM algorithm used in our implementation of the DEMO III system is run only once and could have learned a sub-optimal solution. However, the test given environment state was also performed using the greedy EM learning algorithm, which led to no significant performance boost.

	P_c	$\overline{P}_c$	$P(C\|\text{sky})$	$P(C\|\text{gravel})$	$P(C\|\text{grass})$	$P(C\|\text{foliage})$	$P(C\|\text{sand})$
given env.	0.83	0.80	0.94	0.73	0.79	0.77	0.79
maximum env.	0.85	0.82	0.99	0.72	0.77	0.83	0.80
interpolating env.	0.84	0.82	0.96	0.74	0.77	0.80	0.80

Table 4.2: Probabilities for all terrain classes.

These quantitative results show a maximum difference of only 6% between the methods. This small maximum difference illustrates that the environment auto-calibration is a powerful technique for outdoor terrain classification and in no means is inferior to an external reference method. Auto-calibration even outperforms the method where the environment state is given on P_c, $\overline{P}_c$ and most of the conditional probabilities. Only the maximum conditional probability value for the grass is higher. This could be an indication that the environmental cluster algorithm boosts performance even further when more elaborate datasets are used.

Figure 4.3 shows normalized confusion matrices of the three tested methods. These matrices are normalized so that every class occurs equiprobable. The numerical results are shown in appendix 6. The first thing that attracts attention is the fact that there is no significant difference between the matrices. The largest percentages are all on the diagonal, illustrating that the labels are mostly classified correctly.

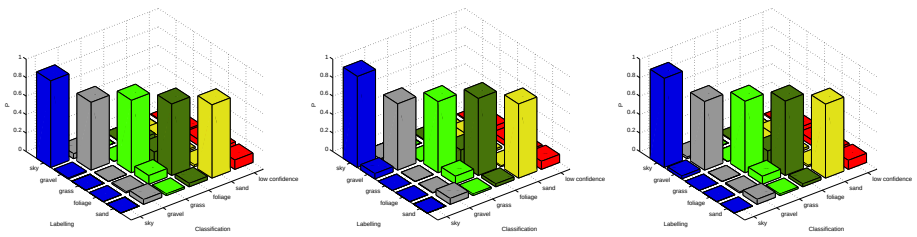


Figure 4.3: Confusion matrices simulating that every class occurs equiprobable. From left to right: given environment, maximum environment auto-calibration, interpolating environment auto-calibration.

In general, the sky label misclassifies a small portion to gravel, mostly due to the color resemblance between the sky in a rainy environment and gravel. Gravel also loses portions of correct classifications to the foliage and sand labels. For the obstacle detection task, misclassifying gravel as foliage is a problem because stones can be classified as green penetrable objects. Grass and foliage are mostly misclassified as each other, which is not a problem for the obstacle detection task: green objects with low height can be safely classified as penetrable. Most of the misclassifications of sand are classified as gravel which is also no problem for the obstacle detection task. However, some portions are also classified as grass and foliage.

The required taxonomy for distinguishing between drivable and non-drivable areas allows that some of the classes are merged. In this obstacle detection case, the gravel and sand classes are merged as well as the foliage and grass classes. Table 4.3 shows that the obstacle detection task performs significantly better by using a smaller taxonomy. These results show that color features perform

	P_c	$\overline{P}_c$	$P(C\|\text{sky})$	$P(C\|\text{gravel} \cup \text{sand})$	$P(C\|\text{grass} \cup \text{foliage})$
given env.	0.90	0.89	0.94	0.84	0.90
maximum env.	0.93	0.92	0.99	0.86	0.92
interpolating env.	0.92	0.91	0.96	0.85	0.91

Table 4.3: Probabilities for obstacle detection task terrain classes.

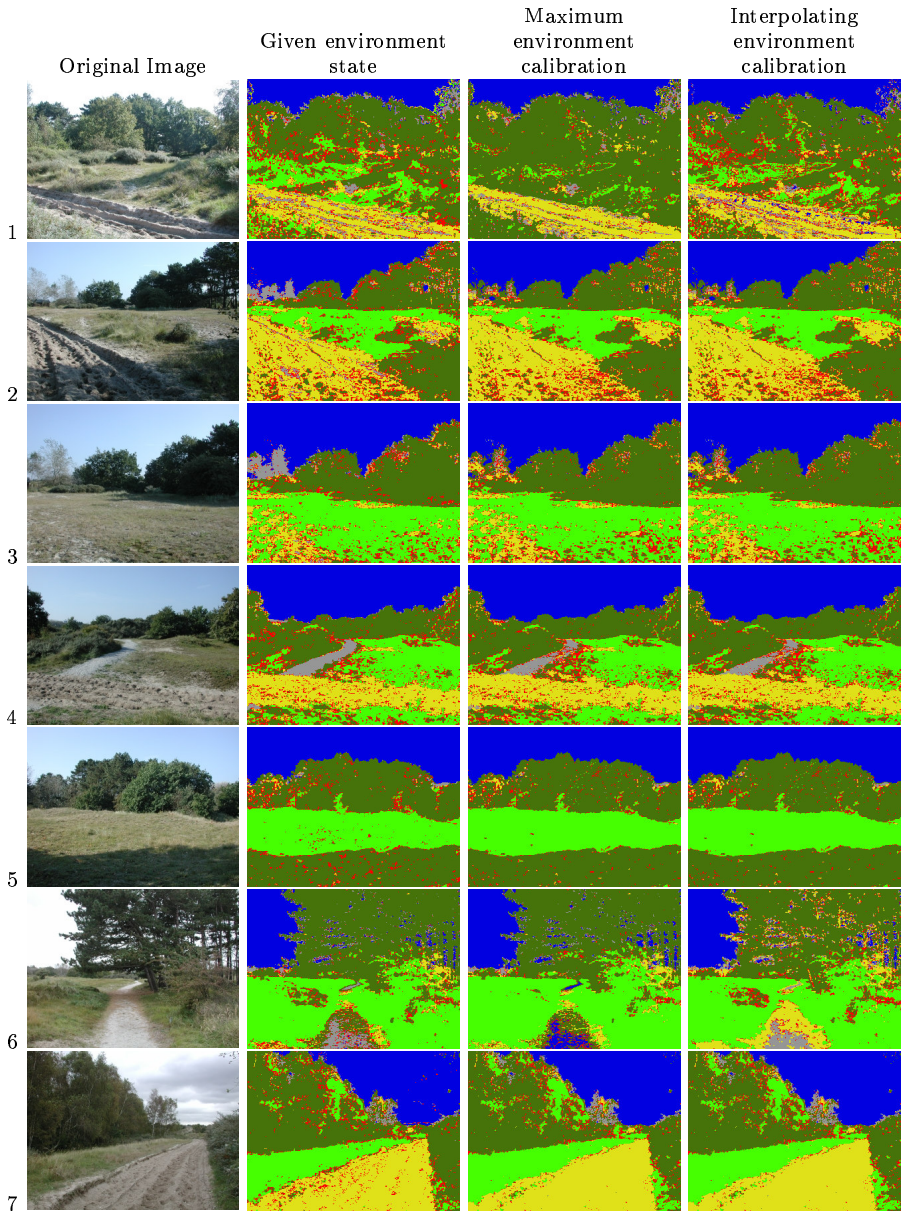
very well in discriminating obstacles in natural terrain. Especially if one takes into account that not all misclassified terrain types are potential obstacles.

Qualitative results

Figure 4.4 shows some classification results. The first column depicts the original RGB images. The rainy images were shot in very dark circumstances, making it hardly possible for a human to discriminate the classes. For illustration purposes these images are shown brightened.

The second column shows the results achieved by our implementation of the JPL method, which requires the environment state as an external reference. The third column shows the results of our automatic environment calibrated system using the terrain cover hypothesis based on the maximum environment likelihood. The fourth column shows the results for the same system using terrain cover interpolation during classification. In this case, the hypotheses are selected that are within 40% percent of the environment hypothesis with the maximum likelihood. These classification results expose some benefits and shortcomings of the used methods that will be discussed in the following paragraphs.

Firstly, interpolating between different terrain cover hypotheses can boost performance. Images 1 and 6 show cases where the interpolation method results are better than the method that selects the hypothesis corresponding with the maximum environment hypothesis. In case 1 more grass is recognized, whereas in case 6 the gravel is recognized. In these cases, the performance improvement is mostly due to supplementing the terrain cover hypothesis with class hypotheses that are not well represented in the terrain cover hypothesis corres-


Original Image
Given environment state
Maximum environment calibration
Interpolating environment calibration
1
2
3
4
5
6
7

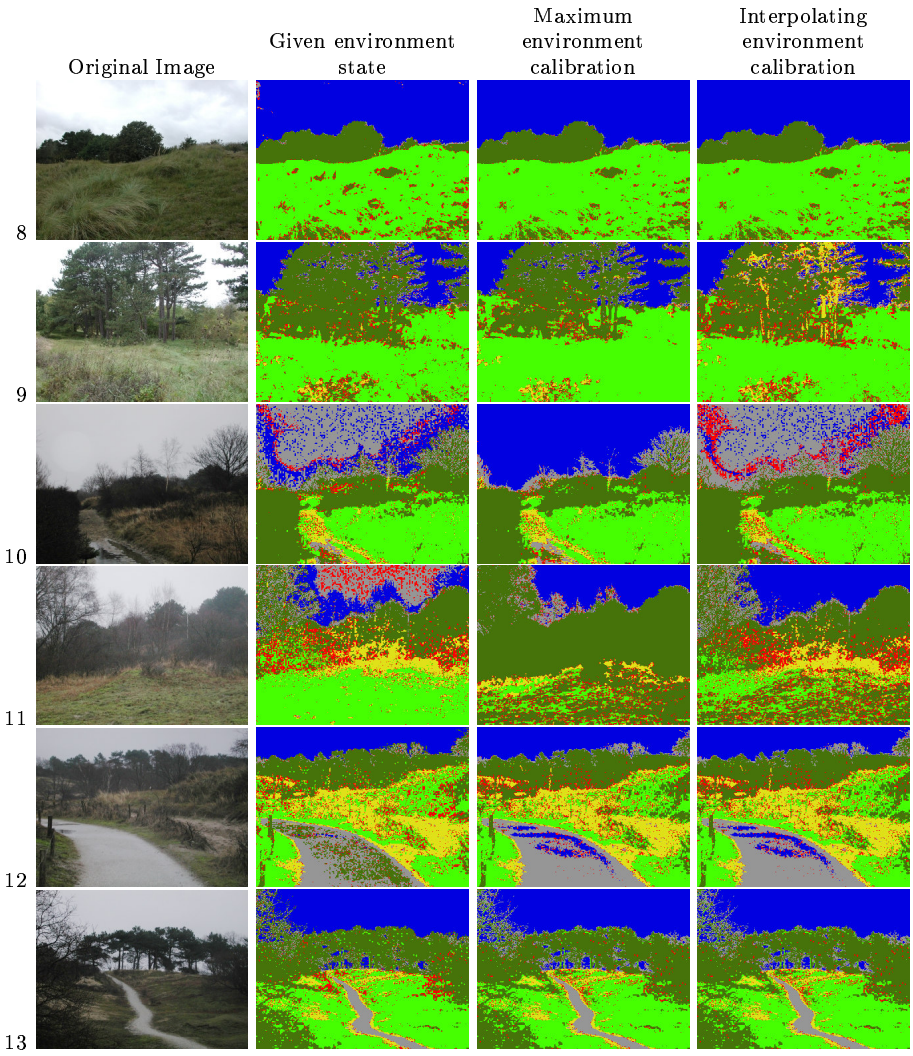


Figure 4.4: The terrain cover classification results (blue = sky, light green = grass, dark green = foliage, gray = gravel, yellow = sand and red = uncertain). Images 1 to 5 are captured in a sunny environment, 6 to 9 in a cloudy environment and 10 to 13 are captured in a rainy environment. The dark rainy images are shown brightened for illustration purposes.

ponding to the maximum environment likelihood. For example, in case 6 the gravel class is ill defined by the maximum environment terrain cover hypothesis. By interpolation with other hypotheses, the gravel class is defined better. Case 10 shows that interpolation between terrain cover hypotheses does not always lead to better results. Unfortunately, there are no images available that are shot in intermediate weather circumstances. However, in these cases it is expected the interpolation method outperforms the other methods.

Secondly, cases 2 and 5 show that heavy shadows on grass are misclassified as foliage. This explains the high number of misclassifications of grass as foliage in the quantitative performance measures. Case 2 also shows that other classes have less trouble with shadows: sand is correctly classified when it is directly and indirectly illuminated.

Finally, patches of high grass can easily be misclassified as foliage because in both cases there is a lot of self shadow casting. Cases 2, 8 and 9 show that the methods mostly correctly classify these patches of high grass. The darkest parts are unfortunately be misclassified as foliage.

To ensure the safety of the vehicle, their is only a small margin for error. For the obstacle detection task, which discriminates between drivable and non-drivable areas, the P_c and $\overline{P}_c$ probabilities are higher than 0.9. If navigational and vehicle control tasks are also required, for example to be able to follow dirt roads, these probabilities do not drop under 0.82.

4.2.4 Post-processing

The results are very encouraging, but a minor adaption may boost performance even more. The classification images contain a lot of pixel noise. By applying a filter on all terrain cover likelihoods, and then selecting only the pixels with the maximum likelihoods, can remove this pixel noise.

	P_c	$\overline{P}_c$	$P(C\|\text{sky})$	$P(C\|\text{gravel})$	$P(C\|\text{grass})$	$P(C\|\text{foliage})$	$P(C\|\text{sand})$
raw	0.84	0.82	0.96	0.74	0.77	0.80	0.80
filtered	0.86	0.84	0.96	0.77	0.79	0.83	0.85

Table 4.4: Raw and filtered terrain class probabilities.

In this test, the results of the classification based on interpolation of the previous section is filtered with a median filter of box size 5. Table 4.4 shows the raw results as well as the filtered results. The P_c and $\overline{P}_c$ probabilities are both boosted 2%. For better results one should vary the filter box size and find the box size with the best results. Pre-processing the original RGB data is likely to produce the same results.

Chapter 5

Conclusion

The University of Amsterdam (UvA) and TNO Physics and Electronics Laboratory (TNO-FEL), part of the Netherlands Organisation of Applied Scientific Research (TNO), developed a testbed for autonomous navigation in unstructured terrain, called the Robojeep. The Robojeep research is focused on developing technology that will enable future robotic vehicles to perform useful and dangerous tasks. An example is humanitary mine clearing: financial costs as well as casualty rates can be reduced significantly by automating this dangerous task.

Presently, the Robojeep depends primarily on its range sensors for observing the terrain conditions. Based on the geometrical model returned by the stereo-vision component, steep object can be evaded. The drawback of these types of sensors is that they do not provide information about the material types, such as grass or sand, that are present in the terrain. Terrain classification is needed for various applications, for example: identification of false obstacles such as tall grass or dirt road following. Adding terrain cover classification to a robot vehicle will enable it to perform its tasks in unstructured terrain more safely and efficiently.

We have developed a color based terrain cover classification algorithm. The apparent color of terrain types is heavily influenced by lightning, weather type and other environment conditions. Therefore, for color based classification to be successful in an unstructured outdoor environment, the environment state has to be estimated. Previous research on terrain cover classification, still considered auto-calibration as an open problem [20]. We were able to solve this problem by assuming that low variance image regions are heavily influenced by the environment state. We learned that color image pixels in low variance regions often correspond to points in high density clusters in the colorspace, which are more suitable for probabilistic modeling.

Our unsupervised environment cluster algorithm groups images with comparable environment states, making learning and classifying possible without a-priori knowledge of the environmental properties. For each environment state

found by the cluster algorithm, an environment hypothesis is learned as well as a terrain cover hypothesis. Both hypotheses learn mixtures-of-Gaussians by using a greedy variant of the EM learning algorithm [35]. The environment estimation and the terrain cover classification are both based on Maximum Likelihood principles.

Three experiments were performed to test our method. The first test evaluated the environment cluster component. The second experiment assesses the performance of the environment estimation. The final test investigated the performance in classifying the terrain types compared to a method that is given the environment state. For the latter approach, the environment state is provided by some external reference source such as a human.

The cluster component of our method performs very well. Besides providing reliable terrain classification results, it proved to be very robust in a wide range of parameter settings. The performance of the classifier, which uses automatic environment clustering, is competitive to that of the other method which is given the environment state. For more elaborate datasets, we expect that environment states based on our clustering method will perform even better. The cluster components provides an extra degree of freedom that able is to group sub-environment states or filter out images that are not part of any frequently occurring environment state.

The auto-calibration algorithm also performs very well. The percentage of correct environment estimations did not drop under 95%, when it was compared to the method that was given the environment state.

The final experiment, terrain type classification, also produced promising results. In distinguishing drivable from non-drivable areas, the reliability probabilities did not drop under 0.9. If navigational and vehicle control tasks are included, to follow dirt roads for example, the reliability probabilities did not drop under 0.82. For both tasks, our auto-calibration algorithm produced classification results that are competitive with the results of the method that is given the environment state.

In the introduction of this report, the problem statement was presented. For clarification reasons the problem statements constraints are repeated here:

1. the algorithm should be able to robustly classify new images as drivable or non-drivable areas under varying weather conditions, seasons and environments given a rich enough learning set and information from the stereo cameras;

2. the algorithm should have confidence measures that reflect the confidence that it has in a classification;

3. the algorithm should be fast enough for real-time applications, and

4. the algorithm should be able to be extended with other classification algorithms.

Concerning the first constraint: since the safety of the vehicle is critical there is only a small margin of error. However, the terrain cover experiments show that a very promising first step towards a more advanced obstacle detection algorithm has been made. Especially if one considers that not all classified terrain classes are obstacles.

Constraints 2 and 4 are solved simultaneously by providing a confidence measure. This measure enables more sophisticated fusion of classification results, and also provides a measure of terrain type certainty to the vehicle control components.

The final constraint is shown satisfied by the Jet Propulsion Laboratory's (JPL) research on autonomous navigation. They demonstrated that ML classification with mixtures-of-Gaussians can be used for real-time autonomous navigation tasks [1]. Besides the ML classification, we provide a classification method that interpolates between different environment states. This method requires additional classification passes and so it is still to be proved that this method can satisfy the real-time constraint.

The color feature was not only chosen for its potential to satisfy the problem statement. A major benefit of color classification is that it can look beyond the range of other sensors such as the stereo-vision cameras. On basis of this auxiliary data more efficient paths could be planned. Possible objects, such as tree lines, can be detected before they are in the range of the stereo-vision system. Autonomous vehicles can also obtain a better notion of where roads or paths are heading. This could be used for new applications such as dirt road following.

Some open problems still remain. Firstly, The misclassification of gravel or sand classes as grass or foliage classes is a problem. Secondly, the system is not able to detect of bodies of water. Another disadvantage of this method is that a lot of labeled images have to be input to the learning algorithm. Creating these labellings can be a tedious job.

However, both the results of the experiments and the mainly solved problem statements indicate that a good first step towards a very robust terrain cover classification algorithm for autonomous navigation has been achieved. False positive obstacles such as patches of tall grass, can now be safely identified. Also, more advanced navigational and vehicle control abilities are now well within the grasp of autonomous vehicles operating in rough terrain.

Chapter 6

Future research

This chapter proposes some research ideas to improve terrain cover classification. The first method to improve terrain cover performance, is to complement the color features with texture features. Methods that can distinct gravel from grass will improve the performance of obstacle detection. For navigation, the distinction between grass and foliage is interesting. JPL experimented with fusing color and texture features [19]. The classification performance did increase, however, they were not able to satisfy the real-time constraints. If two notions are taken into account, this real-time constraint can probably be satisfied. The first notion is that the amount of possible obstacles in the robot's vehicle path is small. The second remark is that the stereo-vision system can provide the texture filters with a distance estimation, and thus filter scale information [27]. By applying texture extraction given a specific scale on small regions of interest, a significant speedup can be gained.

Another idea worth investigating is trying to reduce the number of hand-labeled training examples. Time-lapse cameras could take pictures on different times of the day and under varying weather conditions. If the camera always focuses on the same scenery, one hand-labeling will label all the pictures taken. Research must show how well the cluster component can group the resulting environment states. It also must be tested how well the hypothesis based on the time-lapse training data classifies the variety of images typically seen by an autonomous vehicle navigating through unstructured terrain.

Experiments may show that classification based on interpolation of environment states is too slow. To gain a speedup, research can investigate the possibility that terrain cover mixtures belonging to the different environment states are merged. By interpolating the environment states at the hypothesis level, only a single pass for the calculation of pixel likelihoods has to be performed.

Finally, our method incorporates no feature to label obstacles that are unknown to the terrain hypotheses. Terrain types where the maximum likelihoods is below a certain threshold can be labeled unknown. The choice of this threshold is problematic because likelihood is not a intuitive criterion. The likelihood threshold can be linked to a probability value, making a more intuitive thres-

hold parameter [20]. We propose to incorporate this approach to robustly detect unknown terrain types.

Appendix A

Numerical confidence matrices

Classifications	Labels: sky	gravel	grass	foliage	sand
sky	**93.74**%	0.81%	0.00%	0.05%	0.00%
gravel	5.44%	**73.35**%	0.07%	1.39%	5.17%
grass	0.00%	0.09%	**78.53**%	9.27%	1.74%
foliage	0.00%	13.37%	13.04%	**77.27**%	3.72%
sand	0.00%	7.92%	1.78%	2.01%	**79.29**%
low confidence	0.81%	4.46%	6.57%	10.02%	10.08%

Classifications	Labels: sky	gravel	grass	foliage	sand
sky	**98.62**%	5.63%	0.01%	0.04%	0.00%
gravel	0.00%	**71.73**%	0.13%	0.32%	6.32%
grass	0.00%	0.27%	**77.06**%	8.56%	1.65%
foliage	0.55%	4.16%	15.19%	**83.28**%	3.17%
sand	0.00%	10.75%	1.92%	1.07%	**80.18**%
low confidence	0.82%	7.45%	5.70%	6.72%	8.68%

Classifications	Labels: sky	gravel	grass	foliage	sand
sky	**96.47**%	3.07%	0.01%	0.04%	0.11%
gravel	1.94%	**74.23**%	0.13%	1.15%	5.42%
grass	0.00%	0.00%	**77.47**%	8.51%	1.63%
foliage	0.55%	5.04%	14.20%	**80.39**%	3.08%
sand	0.00%	10.89%	2.04%	1.60%	**80.01**%
low confidence	1.04%	6.76%	6.16%	8.31%	9.75%

Numerically shown confusion matrices simulating every class occurs equiprobable (see section 4.2.3). From top to bottom: given environment, maximum environment auto-calibration and interpolating environment auto-calibration.

Bibliography

[1] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, A. Rankin. Terrain perception for DEMO III. In *Proceedings of the Intelligent Vehicles Symposium*, Dearborn, Michigan, October 2000.

[2] J. S. De Bonet, P. Viola. Texture recognition using a non-parametric multi-scale statistical model. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1998.

[3] David H. Brainard, William T. Freeman. Bayesian color constancy. In *Journal of the Optical Society of America A*, vol. 14, no. 7, July 1997.

[4] S.D. Buluswar, B.A. Draper. Color recognition in outdoor images. In *IEEE International Conference on Computer Vision*, January 1998.

[5] S.D. Buluswar, B.A. Draper. Color machine vision for autonomous vehicles. In *International Journal for Engineering Applications of Artificial Intelligence*, 11(2), pp. 245-256, 1998.

[6] R. Castaño, R. Manduchi, J. Fox. Classification experiments on real-world textures. In *Empirical Evaluation Methods in Computer Vision*, Kuaii, HI, Dec. 2001.

[7] D.J. Field. What the statistics of natural images tell us about visual coding. In *Proc. SPIE 1077*, pp. 269-276, 1989.

[8] D.J. Field. Scale-invariance and self-similar 'wavelet' transforms: an analysis of natural scenes and mammalian visual systems. "Wavelets, fractals and fourier transforms" (ed. M. Farge, J. Hunt, J.C. Vassilicos). Oxford University Press, 1993.

[9] D.J. Field. Wavelets, vision and the statistics of natural scenes. In *Phil Tran R. Soc. Lond. A 357*, pp. 2527-2542, 1999.

[10] D. Forsyth. A novel approach for color constancy. In *International Journal of Computer Vision*, 5:5-36, 1990.

[11] R. Gershon, A. Jepson, J. Tsotsos. "The effects of ambient illumination on the structure of shadows in chromatic images", RBCV-TR-86-9, Dept. of Computer Science, University of Toronto, 1986.

[12] Theo Gevers, Arnold Smeulders. Color based object recognition. In *Pattern Recognition 32*, pp. 453-464, March, 1999.

[13] Th. Gevers and H. M. G. Stokman. Classifying color transitions into shadow-geometry, illumination highlight or material edges. in *IEEE ICIP*, pp. 521-525, part I, 2000.

[14] M. Hebert and N. Vandapel. Terrain classification techniques from ladar data for autonomous navigation. Presented at *Collaborative Technology Alliances conference*, May, 2003.

[15] M. A. Hoang, J. M. Geusebroek. Measurement of color texture. In *Proceedings of the 2nd International Workshop on Texture Analysis and Synthesis (Texture 2002)*, pages 73-76. Heriot-Watt University, Copenhagen, Denmark, 2002.

[16] Patrik O. Hoyer, Aapo Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. In *Network; Computation in Neural Systems 11* pp. 191-210, 2000.

[17] D. Judd, D. MacAdam, G. Wyszecki. Spectral distribution of typical daylight as a function of correlated color temperature. In *Journal of the Optical Society of America*, 54(8):1031-1040, 1964.

[18] Te-Won Lee, Thomas Wachtler, Terrence J. Sejnowski. Color opponency is an efficient respresentation of spectral properties in natural scenes. In *Vision research 42*, pp. 2095-2103, 2002.

[19] Roberto Manduchi. Bayesian fusion of color and texture segmentations. In *International Conference on Computer Vision*, 1999.

[20] R.Manduchi, A. Castano, A.Talukder, L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. Submitted to *Autonomous Robots*, September 2003 — revised April 2004.

[21] W. van der Mark, F.C.A. Groen, J.C. van den Heuvel. Stereo based navigation in unstructured environments. In *Proc. IEEE Instrumentation and Measurement Conference*, pages 2038-2042, Budapest, Hungary, May 2001.

[22] L. Matthies, A. Kelly, T. Litwin, and G. Tharp. Obstacle Detection for Unmanned Ground Vehicles: A Progress Report. Presenten on *7th International Symposium on Robotics Research*, October, 1995.

[23] Tom Mitchell. Machine learning. McGraw Hill, 1997.

[24] Sérgio M. C. Nascimento, Flávio P. Ferreira. Statistics of spatial cone-excitation ratios in natural scenes. In *Journal of the Optical Society of America*, vol. 19, no. 8, august 2002.

[25] C. Novak, S. Shafer, R. Wilson. Obtaining Accurate color images for machine vision research. In *Proceedings of the SPIE*, v 1250, 1990.

[26] B.A. Olhausen, D.J. Field. Natural image statistics and efficient coding. In *Network: Computation in Neural Systems 7*, pp. 333-339, 1996.

[27] Clark F. Olson. Adaptive-scale filtering and feature detection using range data. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 983-991, September 2000.

[28] D.A Pomerleau. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell & S. Mahadevan (Eds.), *Robot learning*, 1993, pp. 19-43. Boston: Kluwer Academic Publishers.

[29] D.A. Roberts, J.B. Adams, M.O. Smith. Discriminating green vegetation, non-photosynthetic vegetation and soils in AVIRIS data. In *Rem. Sens. Environ.*, 44: 2/3 255-270, 1993.

[30] Daniel L. Ruderman, Thomas W. Cronin, Chuan-Chin Chiao. Statistics of cone responses to natural images: implications for visual coding. In *Journal of the Optical Society of America A*, 15:2036-2045, 1998.

[31] A. Talukder, R. Manduchi, A. Rankin, L. Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. Presented at *IEEE Intelligent Vehicles Symposium 2002*, Versailles, France, 2002.

[32] E. Todt and C. Torras. Color constancy for landmark detection in outdoor environments. In *Proc. 4th European Workshop on Advanced Mobile Robots (Eurobot'01)*, Lund, Sweden, Sept. 2001.

[33] Yanghai Tsin, Robert T. Collins, Visvanathan Ramesh, Takeo Kanade. "Bayesian color constancy for outdoor object recognition". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, December, 2001.

[34] C. Vertan, N. Boujemaa. Color texture classification by normalized color space representation. In *Proceedings of the International Conference on Pattern Recognition*, Barcelona, 2000.

[35] Nikos Vlassis, Aristidis Likas. A greedy EM algorithm for Gaussian mixture learning. In *Neural Processing Letters*, 15(1): 77-87, February 2002.

[36] G. van de Wouwer, S. Livens, P. Scheunders, D. van Dyck. Color texture classification by wavelet energy correlation signatures. In *Proceedings of the International Conference on Computer Analysis and Image Processing*, pp. 327-334, 1997.