Faculty of Aerospace Engineering

The influence of a bottom camera in indoor ground-segmentation based obstacle avoiding performance for MAVs

M. W. M. Kuijpers

November 9, 2016

TU Delft Delft University of Technology

Challenge the future

The influence of a bottom camera in indoor ground-segmentation based obstacle avoiding performance for MAVs

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

M. W. M. Kuijpers

November 9, 2016

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"The influence of a bottom camera in indoor ground-segmentation based obstacle avoiding performance for MAVs"** by **M. W. M. Kuijpers** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: November 9, 2016

Readers:

Prof. Dr. Ir. M.Mulder

Ir. C. De Wagter

Dr. Ir. G.C.H.E. de Croon

Ir. P. Roling

Acronyms

ACAS	Active Contour Affine Scale
APF	Artificial Potential Field
AR	Augmented Reality
DTAM	Dense Tracking and Mapping
DWA	Dynamic Window Approach
FAST	Features From Accelerated Segment Test
FGM	Follow the Gap Method
FOV	Field Of View
FP	False Positive
GPS	Global Positioning System
GPU	Graphics Processing Unit
HSV	Hue Saturation Value
IBD	Image Brightness Derivatives
MAV	Micro Aerial Vehicle
MOPS	Multi-Scale Oriented Patches
ND	Nearness Diagram
PRM	Probabilistic Road Maps
PTAM	Parallel Tracking and Mapping
RGB	Red Green Blue
RGB-D	Red Green Blue - Depth
RRT	Rapidly Exploring Random Trees
SIFT	Scale Invariant Feature Transform
SIRS	Scale Invariant Ridge Segments
SLAM	Simultaneous Localization and Mapping
SURF	Speed Up Robust Features
SUSAN	Smallest Univalue Segment Assimilating Nucleus
SVO	Semi dense Visual Odometry

TP True Positive
TTC Time To Contact
VFH Vector Field Histogram
VO Visual Odometry

Preface

This work contains the thesis project I worked on, as a final step to acquire the degree of Master of Science in Aerospace Engineering. My interests in automation, led me to contact my supervisors Christophe De Wagter and Guido De Croon to ask them to be the supervisors of my thesis.

I would like to thank Christophe and Guido for their input and feedback. I want to thank my father and Stijn for their love, support, for the good advice and tolerance. Furthermore, I want to thank Stijn, Xiong Hao, Serhat, Yonas and Chabely for the good times after the working hours.

Contents

Chapter 1

Thesis article

The influence of a bottom camera in indoor ground-segmentation based obstacle avoiding performance for MAVs

M.W.M. Kuijpers C. De Wagter G.C.H.E. de Croon

November 9, 2016

Abstract

Navigation in unknown indoor environments is a challenging task, requiring the avoidance of unknown obstacles. The available moving space is limited and the variety of potential obstacles is large. Active sensors that detect obstacles are heavy and power consuming. Monocular obstacle avoidance methods often require texture or a priori knowledge. This paper introduces a novel method to use just a single forward looking camera for avoiding obstacles. In particular, an approach is extended in which a Micro Air Vehicle (MAV) detects obstacles by recognizing the floor. The extension is to use not only the MAVs frontal images but also its bottom camera images for learning floor appearance. Color and texture information is retrieved from the front and bottom camera images and stored as features. The color features used are the HSV and RGB color space and their respective histogram distances between the front and bottom camera images. For texture, the difference in edgeness and Law's energy measures histogram distances are used as features. These features are extracted from the front and bottom images and their distance is used to classify each pixel of the front image as belonging either to the ground or to an obstacle. This is first done with a priori information by training a Reduced Error Pruning (REP) tree for a single image, followed by training a REP tree for an entire dataset without bottom camera information and with bottom camera information. The best results without bottom camera information are obtained with the HSV features only and are 93.3% accurate. With bottom camera, the best results are also obtained for the HSV features and are 97.5% accurate. To eliminate the dependency on a priori information, two approaches are taken to segment the images. The first method is Ulrich's image segmentation, which uses a trapezoid directly in front of the robot as a reference for obstacle-free space. The second approach is the detection of outliers with respect to bottom camera information, classifying the outliers as obstacles. Classifications using Ulrich on the same dataset are 59.4% accurate and with outlier 82.7%, when using HSV features. Based on this classification, a navigation controller is designed. The proposed solution is tested while navigating in a new untrained environment and shown to select the ideal navigation solution in 64 % of the cases, with 27 % neutral decision and 9 % wrong decision.

I. Introduction

The use of autonomous MAVs opens new doors for exploration and surveillance in unknown, GPS denied environments. The lack of a priori knowledge of the environment and thus of potential obstacles, makes indoor navigation a challenging task. Active sensors such as Lidar and Kinect cameras can be used for obstacle avoidance, but are heavy [16] and power consuming [3].

Passive sensors such as cameras can be light weight and their power consumption depends on the computer vision algorithm only [4]. Stereo vision requires a minimum baseline [1], that puts a minimum on the size of the MAV. Furthermore, it puts a stiffness constraint on the cameras, that leads to an in-

crease in weight. It also requires that the cameras see the same object at the same time, with sufficient difference between the object position in the image [10]. Monocular vision omits these restrictions. In pursuance of increasing the potential mission time, the power consumption of the MAV is preferably minimized. In addition to that, minimizing the size of the MAV, offers the opportunity to navigate through smaller openings and to use a swarm of MAVs to simultaneously explore unknown indoor environments. The use of a monocular camera is therefore promising.

Previous research has been done on monocular obstacle avoidance. Optic flow, inspired by insects, uses motion to detect obstacles [9], but requires texture [11] and, similar to perspective cues, has difficulties with frontal obstacles [14], [2], [11], [13]. Navigation algorithms such as SLAM can be used for indoor obstacle avoidance [5], but are computationally expensive [7].

Image segmentation techniques divide images in obstacle and non obstacle regions and are promising for both obstacle avoidance and navigation. They are computationally inexpensive [6]. In addition to information on both sidal and frontal obstacles, the segmentation can provide information on the pitch and roll angle of MAVs, from the resulting horizon line in outdoor flight. This is done in [6], provided that the reference obstacle free region is visible and can be extracted. Using a training set has been proposed to achieve obstacle avoidance based on color segmentation in [17], [12]. [17] has shown that a ground based robot can be navigated without collisions using a training set with HSI histogram information. The consequence, however, is that the test environment must resemble the training set and that the ground and obstacle must be visually distinct.

The purpose of this research is to investigate whether the use of a bottom camera as a reference for obstacle free space in an image segmentation approach, can help reduce this resemblance requirement. It provides live evidence of obstacle free space for MAVs.

In section *II*, the method used for this research is presented. Section *III* discusses the results and the conclusions can be found in section *IV*.

II. Methodology

The method proposed in this research, is to segment images, made by the front camera of the MAV, by classifying them as either an obstacle free region or as an obstacle. The assumption used is that the floor is obstacle-free [17]. The driving idea is that in regions where ground-based obstacles are present, the floor is not visible. Thus, where the floor is visible, the region is obstacle free. Current image segmentation methods for obstacle avoidance are restricted by their dependency on texture or on a predefined color of the floor. Either when too little texture is present, the color of the floor changes or the scene is too different from the training set, these methods do not work. Figure 1, shows an example of the lack of texture in indoor environments and the variety of the appearance of the floor within the same building.

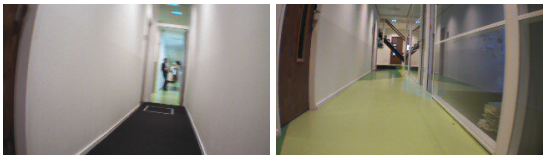

Figure 1: *Example images showing a typical indoor environment where moving space is limited, texture is not abundant and scene variations are obvious*

In an attempt to overcome these dependencies, this study investigates whether the properties of the floor as seen by the bottom camera can overcome these limitations.

First, the setup of the experiment is presented, followed by the approach used to compare the current bottom and front images and a method that uses a sequence of front and bottom images to perform the segmentation. Finally, a method is presented to segment the front image based on information on the bot-

tom image only, which is compared to the method used in [17].

i. Setup

To ensure visibility of the floor under all flying conditions, it is proposed to use an MAV with a bottom camera, where the images taken by the bottom camera are a reference for obstacle free space. The MAV used for this research is the $Parrot^{TM}$ Ardrone2, since it has both a front and bottom camera. The front and bottom camera have an effective FOV of 75 deg and 45 deg, respectively. As can be seen in figure 2, the minimum distance (d1) at which the floor can still be perceived in the front image, is dependent on the height (h1) of the MAV above the ground. Flying at a lower height, thus allows a closer approach to obstacles and overall more visibility of the floor in the front images.

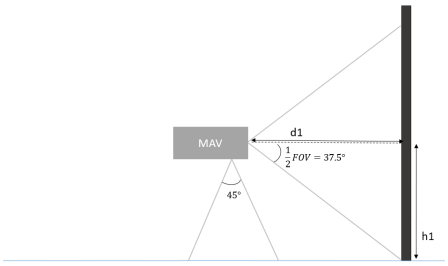

Figure 2: *Setup of MAV with 2 cameras. Front and bottom camera with respective FOV and minimum distance at which the floor can be perceived with the front camera*

To test whether the use of the bottom camera helps in obstacle avoidance by segmenting the image in floor and obstacle, a dataset with varying indoor and outdoor environments is created. When analysing the images, a number of challenges come to light. The main challenges are illustrated with an example in figure 3. As demonstrated in figure 2, the bottom camera has a more narrow field of view and a closer distance to the registered object, which results in a more detailed image. The front camera focuses on the distance ahead of the MAV. The resolution of the front camera is too low to distinguish the same level of detail at a larger distance. To make the bottom and front resolution match, low pass filtering or subsampling of the bottom images is required. Furthermore, a perspective transformation occurs from the bottom image to the front image, due to the difference in angle, which needs to be compensated for with non-uniform floors. Another difference between the front and bottom camera image that can be seen, is the difference in color and lighting, due to automatic lighting adjustments of the camera. It can be seen that the black floor appears to be gray in the bottom camera image. Finally, smooth floors, do not only reflect light, they also show shadows. These shadows might not be present in the front image, dependent on where the light comes from.

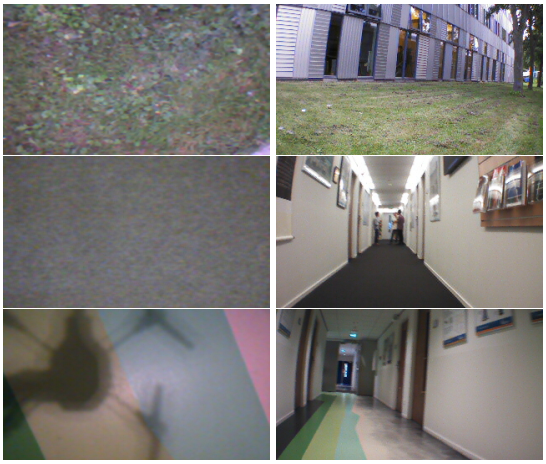

Figure 3: *Example differences bottom (left) and corresponding front images (right)*

ii. Feature extraction current image

The goal is to classify the pixels which represent the floor in the front camera, using the bottom camera as additional classification information. The floor as seen by the bottom camera needs to be retrieved from the front image in order to segment the front image. This is done extracting features from the current bottom and front image. To de-

termine which feature or combination of features results in the best classification performance, several combinations of color and texture based features are investigated. The first feature that is extracted from the bottom image is color. Both the RGB and HSV color space are extracted. The histogram of each color channel is determined for the entire bottom image. For each pixel in the front image, the distance from the pixel histogram to the matching bottom histogram is then determined as in equation 1. In equation 1, the distance is defined as d, x_i is the i^{th} bin value of the histogram of the front image pixel and y_i is the i^{th} bin value of the histogram of the bottom images pixels.

$$d = \sqrt{(x_i - y_i)^2} \tag{1}$$

The segmented image is presented as a binary image. Albeit not much texture is present in the floor in most images, it is expected that it is still a useful addition to the color based features, for instance because obstacles have more texture. For the texture features, information on the edges in the images are extracted. Many indoor environments are characterized by edges, such as for example walls, doors and windows. Information on the texture of a region can be expressed by the amount of edges, larger than a threshold value, in that region. Edges can be detected with a gradient based edge detector, such as the Canny edge detector. The amount of edges larger than a threshold value in a region, $F_{edgeness}$, can be calculated with equation 2, where G is the gradient, and N the number of pixels in the region.

$$F_{edgeness} = \frac{\left|\{p|G(p) \geqslant threshold\}\right|}{N} \tag{2}$$

This technique, however, does not see the difference between a smooth floor and a smooth wall. This approach is therefore seen as an additional feature to the color-based approach, to limit the matching areas to those that are relevant. Another way of obtaining information on the texture in images, is by using Laws' texture energy measures [8]. The energy measures are determined by applying 2D convolution kernels to the image. The 2D kernels are formed by the 1D kernels as presented in the matrix below (equation 3), to both the front and bottom images. When convolving the 1D kernels, for instance L5 and R5, the resulting 2D measure is named L5R5. The 16 resulting 2D kernels can be combined, to replace the pairs with their averages. The combination of E5L5 and L5E5, the horizontal and vertical edge content, respecively, produces the total edges content, for instance [8]. This results in 9 2D kernels. Figure 5, shows examples of the resulting 9 applied 2D kernels to a front camera image. The input image for this, is figure 4.

$$\begin{bmatrix} L5 \\ E5 \\ S5 \\ R5 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ -1 & 0 & 2 & 0 & -1 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} \tag{3}$$

Figure 4: *Front camera image at which Law's measures are applied*

For the 1D Laws kernels, L5 is the Level mask and represents the center-weighted local average, E5 is the Edge mask and detects edges, S5 is the Spot mask and detects spots and R5 is the Ripple mask that detects ripples [15].

One aspect that needs to be considered when using texture, is the fact that the bottom camera sees images in a different perspective than the front camera sees it. Not only are objects further away from the observer seen as smaller than those close by, a perspective

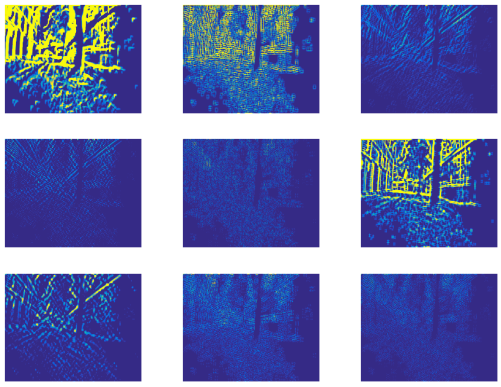

Figure 5: *The 9 resulting 2D laws measures: Top row left to right: L5E5 & E5L5, L5R5 & R5L5, E5S5 & S5E5, middle row left to right: S5S5, R5R5, L5S5 & S5L5, bottom row left to right: E5E5, E5R5 & R5E5, S5R5 & R5S5*

transformation occurs due to the difference in angle of observation. Therefore, to optimally use texture as a feature, the bottom images require a suitable perspective transformation. In this research texture features are applied to images without perspective transformation.

iii. Feature extraction sequence of images

A dataset was created with large variations in ground appearance and lighting. To asses if a single classification can reach good classification performance over a broad range of floors, a classifier is trained on an entire dataset. This is done by first identifying the actual class of each pixel by manually labelling the floor in a large dataset of front images. An example of labelled front images is illustrated in figure 6. The next step is to extract features for each pixel in each image and store these in one large matrix, including the class for each pixel, obtained from the floor labelling. For every new pixel, a new row is added to the matrix, where each column contains a different feature parameter. The features extracted for the analysis of the generalizability of the color features, are the HSV histogram values, their mean, standard deviation and entropy values,

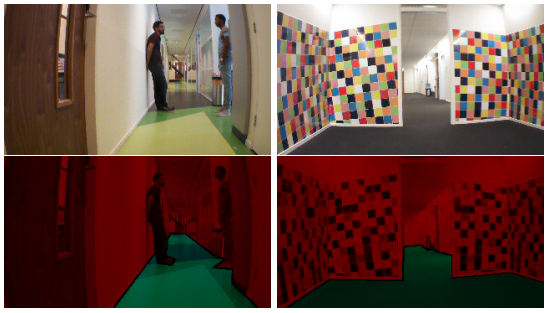

Figure 6: *Example floor labelling, all pixels labelled as obstacle are made red, non-obstacle pixels are green*

including the ground truth of the pixel. To see the influence of the bottom image, a second matrix is generated, that includes the difference of these values with the bottom image. Another aspect that is studied, is whether a combination of different types of features results in a better classifying performance. The combinations that are considered are: HSV with Edges and HSV with Laws. The resulting matrices are used to train a REP decision tree, of which the depth will be limited, to keep a clear overview of the decisions.

iv. Outlier detection bottom image

As a final image segmentation method, it is determined whether segmentation acceptable for navigation can be achieved by using the bottom features as a baseline for outlier detection. A pixel is considered an outlier when its properties differ from the properties of the last 3 bottom images with the amounts resulting from equation 4 or 5, where Q1 represents the first quartile, Q3 represents the third quartile and IQR is the inter quartile range. Outliers are detected by determination of the quartiles of each features of 3 subsequent bottom images.

$$Feature < Q1 - 1.5IQR \tag{4}$$

$$Feature > Q3 + 1.5IQR \tag{5}$$

The method as described by Ulrich in [17], is implemented and used as a benchmark for this research. The method uses a trapezoid

area in front of the robot as a reference for potential ground. The robot in [17], learns the appearance of the ground while driving over it. If the robot has driven over the trapezoid area, the area was obstacle free. It then classifies each pixel, based on its HSI color appearance, as either ground or an obstacle, by histogram comparison. This method is implemented as follows: The trapezoid area RGB histograms of 3 front camera images are first filtered and converted to HSI color space. The resulting H and I histograms are then compared with the H and I histograms of the pixels in the following front image. If the sum of the differences for H and I, each individually, is larger than 0.19, the pixel is considered as obstacle. In addition to studying the image segmentation performance of the outlier detection method, the navigation capabilities are investigated. This is done by scanning the segmented image for pixels classified as ground, starting at the image bottom row. From here, the largest concentration of neighbouring pixels is identified and used to indicate an obstacle free path. The resulting binary image is segmented in 3 columns, a left column, a centred column and a right column. The column with the largest number of neighbouring obstacle-free pixels, starting from the bottom row, indicates the navigation direction to be followed by the MAV. This is done for the estimated grounds resulting from the outlier detection methods, which is compared to the correct flying direction, resulting from applying the same method to the hand-labelled front images, which were also used for the classifier training.

III. Results

i. Histogram-difference-threshold-classification within current image

First, the difference between the current front and bottom images is used, to segment the front images. The histogram distance between the bottom and front image regions is subjected to a threshold value for the HSV, RGB, Edges and Laws features. Although 9 masks result from applying Laws energy measures, one is presented here, to improve readability. This can be done since a number of masks gave similar good results. To reduce computation time, the mask that overall gave the best combination of error, mean and variance is used. This is the mask that results from the combination of L5R5 and R5L5. This produces the total ripple content of the image. The combination of R5 with R5 and L5S5 with S5L5 produce similar results. Laws' method for segmenting a single image, is first applied on a dataset with mostly black carpet floors and white walls. The error between the estimated class of the pixel and the actual class, as taken from the labelled images, is presented for all methods in figure 7. The error is determined as the sum of the difference between the hand-labelled image pixels and the pixels classified by the histogram differences. The

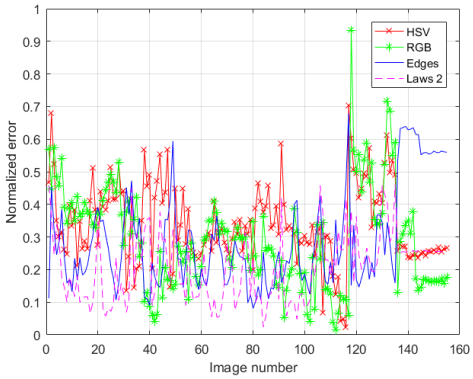

Figure 7: *Normalized error feature distance during classification of ground pixels in a single bottom-front image pair, thresholds selected using dataset 1, tested on dataset 1*

mean and variance of the method errors can be found in table 1. In figure 7, it can be seen that the largest errors occur at images 117 and 118. When looking at the front and bottom image 117 as presented in figure 8, one can see that the floor cannot be seen in the front image, while the white of the wall has a gray appearance, instead of perfect white. Furthermore, the bottom image contains white, which

is notable in the histogram of the bottom image. For this dataset, Laws' method works best, since it produces the smallest mean error.

Table 1: *Mean and variance of error dataset 1, thresholds selected using dataset 1*

Feature	Mean	Variance
HSV	0.3388	0.0149
RGB	0.2921	0.0241
Edges	0.2891	0.0227
Laws	0.2072	0.0106

Figure 8: *Image 117: Front image and bottom image*

Table 2: *Accuracy by class for dataset 1, thresholds selected based on dataset 1*

	TP Rate	FP Rate	Class
HSV	0.7733	0.2665	Ground
	0.7335	0.2267	Obstacle
RGB	0.7135	0.1994	Ground
	0.8006	0.2865	Obstacle
Edges	0.7589	0.2165	Ground
	0.7835	0.2411	Obstacle
Laws	0.9507	0.2332	Ground
	0.7668	0.0493	Obstacle

The results of the accuracy analysis for dataset 1, can be seen in table 2. When applying the same conditions to a different dataset, where more color variation is present in the dataset, the errors are as presented in figure 9. One can see that in general, using the RGB color space, gives the largest error between the labelled and estimated floor. The HSV color space overall gives the smallest error here. The overall error for each method is larger than for the simpler dataset, described above.

The results of the resulting segmentation for images from dataset 1 can be seen in figure

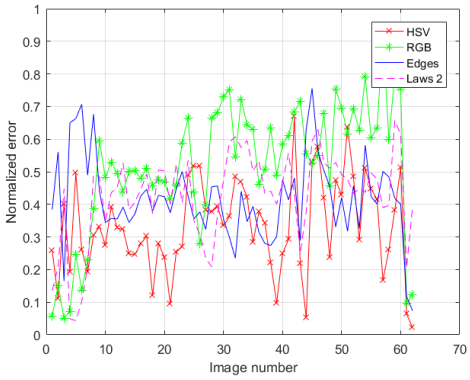

Figure 9: *Normalized error feature distance during classification of ground pixels in a single bottom-front image pair, thresholds selected using dataset 1, tested on dataset 2*

10. One can see that the HSV comparison

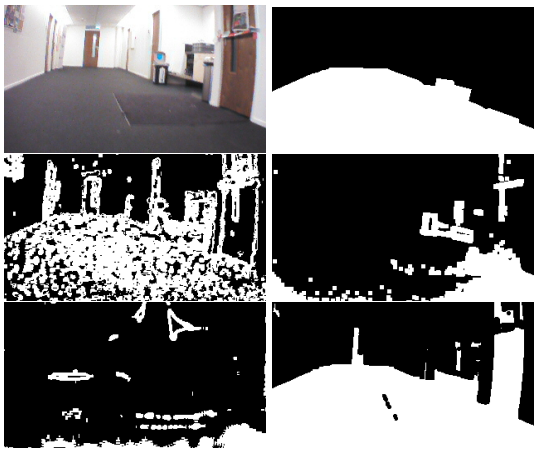

Figure 10: *Top row: input image and labelled ground, middle row: HSV and RGB segmentation, bottom row: Edges and Law's segmentation*

and Law's energy measure give the best results. From the plot in figure 7, it is harder to see which method gives the best performance. This is because the performance is greatly dependent on the properties visible in the front image and bottom image. These do not always match. When texture, such as edges are present in the images, Law's energy measures work better than when little texture is present. The same holds for the edgeness.

ii. Decision tree HSV features

It is examined whether using a set of images improves the performance of obstacle recognition, with respect to using information of the current images only. This is done by gathering the respective pixel feature information in a matrix, used to train a classifier. For each method, this is first done without the difference with the bottom image included, then with these features included.

In the first test, the HSV histograms and the mean, standard deviation and entropy of the front image pixel neighbourhood are used as information to train a REP decision tree. For each pixel, the correct class is added, taken from the pre-labelled binary images. The results for the sensitivity analysis of a tree of depth 10, is given in table 3. The depth of the trees used, is dependent on where the accuracy stagnates. The accuracy is determined by the percentage of pixels correctly classified over the entire dataset.

Table 3: *Accuracy by class HSV features, without bottom, by class for tree of depth 10, 93.3% accurate, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.831	0.040	Ground
0.960	0.169	Obstacle

The difference with the bottom image of the above mentioned features is then added to the feature matrix, to train a REP decision tree of depth 10. The results can be seen in table 4. It can be seen that, when comparing the true positive rates for the ground, the true positive rate has increased from 0.831 to 0.932.

Table 4: *Accuracy by class HSV, with bottom, by class for tree of depth 10, 97.5% accurate, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.932	0.014	Ground
0.986	0.068	Obstacle

An example of the resulting segmentation for the features without the ground information and with the ground information included can be seen in figure 11. It can be seen

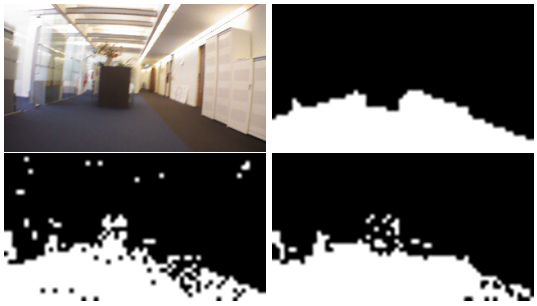

Figure 11: *Top row: input image and labelled ground, bottom row: segmentation by HSV without bottom and with bottom included, trained on dataset 1, tested on dataset 1*

that the addition of the bottom features, improves the segmentation performance.

iii. Decision tree HSV and edge features

To see the effect of the combination of HSV features and edge properties, two decision trees are trained. One that contains information on the front images only and one that includes the difference with the bottom features. The features used for this tree, are those as used for the HSV features only, complemented with the edgeness of the image. When the bottom is added, the difference in edgeness with the bottom is added too. The results of the REP tree, based on the front properties only, can be seen in table 5. The result of the front properties augmented with the bottom property difference, is presented in table 6. The resulting segmentation is illustrated in figure 12. It can be seen that the inclusion of the bottom

Table 5: *Accuracy by class HSV with Edges, without bottom, by class for tree of depth 10, 91.3% accurate, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.777	0.048	Ground
0.952	0.223	Obstacle

Table 6: *Accuracy by class HSV with Edges, with bottom, by class for tree of depth 10, 96.2% accurate, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.900	0.020	Ground
0.980	0.100	Obstacle

information increases the segmentation performance. The segmentation for the front image

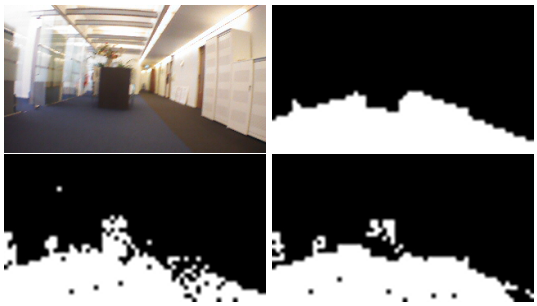

Figure 12: *Top row: input image and labelled ground, bottom row: segmentation by HSV and Edges without bottom and with bottom included, trained on dataset 1, tested on dataset 1*

features only gives similar good results as the HSV method. One can see that the result of including the bottom information, is better than the tree with the front features only. It is thus better to include both the front and bottom information.

iv. Decision tree HSV and Laws features

For the combination of HSV properties and Laws features, a tree is trained with the front images properties only, followed by a tree trained with both front and bottom properties. The front images properties are the HSV properties as describe above, and the histogram of Law's energy measures. The difference with the bottom includes the histogram differences for the HSV features and Law's energy measures. The results for the TP Rate and FP Rate can be found in table 7.

Table 7: *Accuracy by class HSV with Laws, without bottom, by class for tree of depth 10, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.814	0.053	Ground
0.947	0.186	Obstacle

For comparison, the differences with the bottom images properties of HSV and Laws are added. The results, as can be seen in table 8, indicate that the difference with the bottom improves the classification performance.

Table 8: *Accuracy by class HSV with Laws, with bottom, by class for tree of depth 15 , 97.1% accurate, trained on dataset 1, tested on dataset 1*

TP Rate	FP Rate	Class
0.927	0.017	Ground
0.983	0.073	Obstacle

The resulting image segmentation is illustrated seen in figure 13. Here, again, it is notable that the inclusion of the difference with the bottom improves the segmentation performance, which matches with the analysis. Again, it can be seen that the segmentation resulting from the front images produces similar results to the other methods and is comparable to the hand-labelled ground.

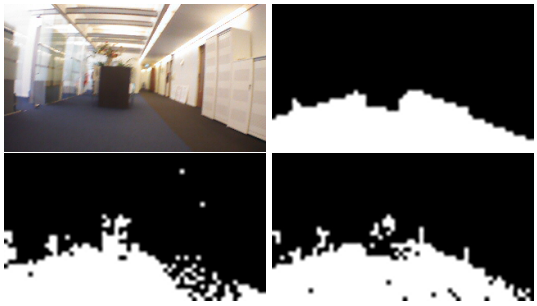

Figure 13: *Top row: input image and labelled ground, bottom row: segmentation by HSV and Laws without bottom and with bottom included, trained on dataset 1, tested on dataset 1*

Overall, that the combination of features does not necessarily improve the result, which means a single method can be used.

v. Outlier detection vs Ulrich's method

In figure 14, the resulting error between the estimated segmentation and the actual segmentation by detecting outliers of HSV, RGB, Edge and Laws features can be seen. For comparison, Ulrich's method is added. The first implementation is tested on dataset 1, which has black floors and white walls.

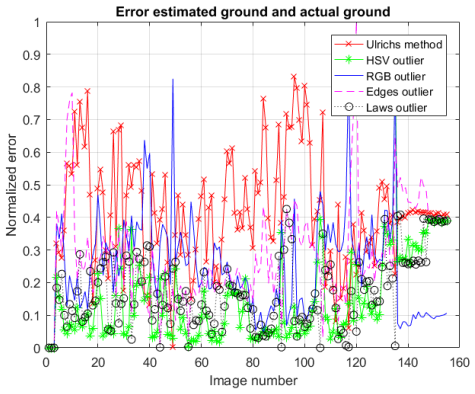

Figure 14: *Error between estimated pixel class and actual pixel class, thresholds selected using dataset 1, tested on dataset 1*

Figure 15: *Example segmentation, all pixels classified as obstacle are made red, non-obstacle are green. Left to right: input image, hand-labelled image, Ulrich*

The error for the edges features is very large for the first 15 images, as presented in figure 14. This is because in some of the bottom images, edges are present in the floor, that are not as explicit in the front images. For the entire dataset, 71.2 % of the pixels are classified correctly. What is also notable is that the

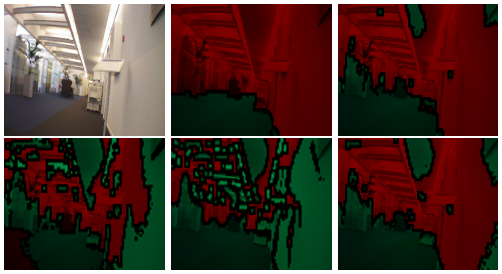

Figure 16: *Example segmentation, all pixels classified as obstacle are made red, non-obstacle made green. Left to right top row: input image, hand-labelled image, HSV. Left to right bottom row: RGB, Edges, Laws*

error resulting from Ulrich's method is larger than the other methods. It classifies 59.4 % of the pixels correctly. This can be explained in 2 ways: the thresholds used for the histogram differences need to be adjusted and due to the variation in the bottom area of the front image. In one image, the floor itself varies but in other images, pieces of the wall are present in the trapezoid, which results in large histogram differences. Furthermore, the current threshold setting is an 'or' statement. This classifies pieces of the ceiling as floor, due to the resemblance in intensity. An example segmentation is illustrated in figure 15. Figure 14 also illustrates, that the RGB outlier method gives a higher overall error than both the HSV and Laws method. The RGB classifies 67.7 % of the pixels over the entire dataset correctly, where HSV and Laws classify a respective 82.7 % and 68.8 % of the pixel over the entire dataset correctly. Segmentation examples are illustrated in figure 16. When applying the same method, with the same threshold values, to a different dataset, with more color variation and obstacles, it can be seen in figure 17 that the error is larger for each method, than when applied to the previous dataset. The edge difference deviates more from the rest, due to its large peaks. These are caused again by edges in the bottom images that are not as explicitly present in the front image. The HSV outliers show the best results. The largest errors are caused by a large difference in appearance of the floor in

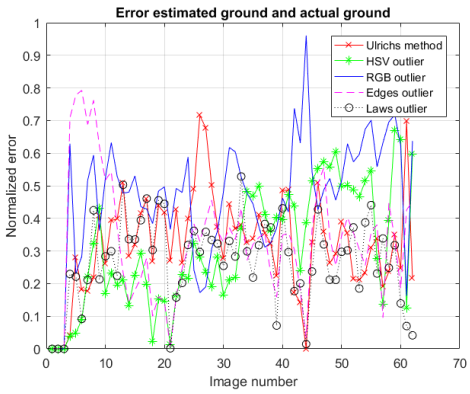

Figure 17: *Error between estimated pixel class and actual pixel class, thresholds selected using dataset 1, tested on dataset 2*

the bottom and front image. Ulrich's method shows a relative large error in our test set. It has difficulties with the large variation in the trapezoid area, that results from the large variation in floor and the frequent occurrence of obstacles in this area. Adapting the used threshold values can further improve the results, but requires finding the optimal threshold values.

vi. Navigation decisions based on ground segmentation

The navigation direction to be followed by the MAV, is determined for each outlier detection method, to see the difference in navigation performance and potential for navigation for each method. For comparison, the navigation direction resulting from Ulrich's method is added. In figure 18 an example of the resulting navigation path and included navigation direction can be seen for all outlier methods and Urlich's method, for dataset 1. 'Go' presents the navigation instruction determined by the outlier ground segmentation and 'Should go' represents the navigation instruction resulting from the hand-labelled ground segmentation. The results for dataset 2 can be seen in figure 19.

The percentage of ideal navigation instructions for both datasets can be found in table

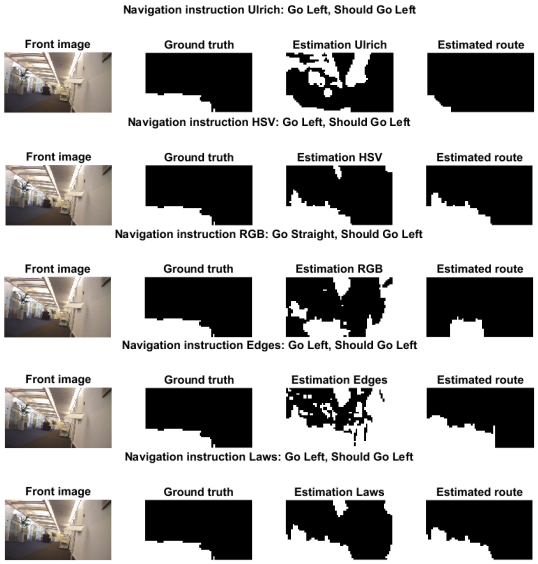

Figure 18: *Example navigation results dataset 1, thresholds determined with dataset 1*

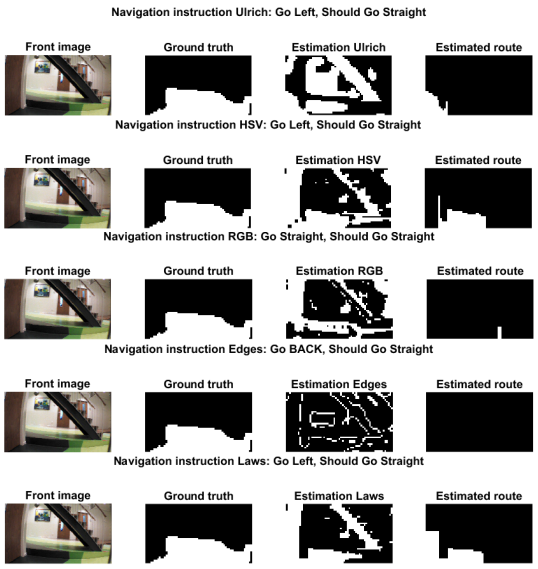

Figure 19: *Example navigation results dataset 2, thresholds determined with dataset 1*

9. A navigation instruction is considered ideal when exactly the same navigation instruction results from Ulrich or the outlier segmentation method as for the hand-labelled segmentation. A wrong navigation instruction is given when the MAV should to turn away from an obsta-

cle but instead turns toward the obstacle. One

Table 9: *Correctly estimated navigation directions for dataset 1 and dataset 2, thresholds determined with dataset 1*

	Ulrich	HSV	RGB	Edges	Laws
Dataset 1 (ideal)	0.41	**0.64**	0.45	0.11	0.36
Dataset 1 (neutral)	0.48	0.27	0.46	0.8	0.57
Dataset 1 (wrong)	0.11	0.09	0.09	0.09	0.07
Dataset 2 (ideal)	0.19	0.34	0.36	0.40	0.14

can see that for the HSV method, 64 % of the navigation instruction are given correctly, for any given information without a priori information. It can also be seen that the navigation instructions rarely resulted in collision, for all methods. The navigation instructions, however, result from 1 image only. It is expected that, when a sequence of images are used, the navigation instruction will be more accurate. Closed loop testing onboard of an MAV is needed to determine the changes of collisions based on bad segmentation. This can be investigated by implementation on the MAV.

IV. Conclusion

This research investigates the effect of including a bottom camera on image segmentation, to be used for obstacle avoidance. The difference between the properties of the bottom image and those of each pixel of the front image gives similar results for the HSV, RGB and Edges method for dataset 1, but depend greatly on the visible environment. Training a REP tree to learn the appearance of the ground, for HSV features and for the combinations of HSV and Edges, HSV and Laws is a good way to segment images for obstacle avoidance. The use of a trained classifier can segment images similar to the training data without information from the bottom camera, but the bottom information improves segmentation performance. Finally, a classification method is examined that does not depend on training data. This is done by the identification of outliers with respect to the bottom image, which are used to segment the front image. For both datasets, the HSV outliers give the best performance. The method performs better than Ulrich's method due to the large variation in the floors and due to the fact that objects are often visible in the trapezoid. The trapezoid used for this research, is small compared to the entire image. Although useful for ground-based vehicles, the trapezoid as seen by an MAV is not directly in front of the MAV, but somewhere in the distance, as illustrated in figure 2. When considering MAV navigation, potential can be seen for the HSV outlier method. Although the navigation instructions are not perfect, they do not necessarily result in collision. It is recommended that the navigation instructions are sophisticated and that this method is implemented on an MAV, to test the obstacle avoiding performance.

References

[1] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3056–3063, 2011.

[2] H. Alvarez, L.M. Paz, J. Sturm, and D. Cremers. Collision Avoidance for Quadrotors with a Monocular Camera. *Proc. of The 12th International Symposium on Experimental Robotics (ISER)*, pages 1–13, 2014.

[3] C. Bills, J. Chen, and A. Saxena. Autonomous MAV flight in indoor environments using single image perspective cues. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5776–5783, 2011.

[4] M. Bl, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision Based MAV Navigation in Unknown and Unstructured Environments. pages 21–28, 2010.

[5] G.C.H.E. de Croon, C. de Wagter, B.D.W. Remes, and H.M. Ruijsink. The Appear-

ance Variation Cue for Obstacle Avoidance. *IEEE Transactions on Robotics IF - 2.571*, 28(2):529–534, 2012.

[6] G.C.H.E. De Croon, C. De Wagter, B.D.W. Remes, and R. Ruijsink. Sky segmentation approach to obstacle avoidance. *IEEE Aerospace Conference Proceedings*, pages 1–31, 2011.

[7] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456, 2013.

[8] K.I. Laws. *Textured Image Segmentation*. PhD Thesis - Microfilm. University of Southern California, 1980.

[9] V. Lippiello, G. Loianno, and B. Siciliano. MAV indoor navigation based on a closed-form solution for absolute scale velocity estimation using Optical Flow and inertial data. *Proceedings of the IEEE Conference on Decision and Control*, pages 3566–3571, 2011.

[10] A. Mcfadyen and L. Mejias. A survey of autonomous vision-based See and Avoid for Unmanned Aircraft Systems. *Progress in Aerospace Sciences*, pages 1–17, 2015.

[11] T. Mori and S. Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1750–1757, 2013.

[12] H. Parag and S. Singh. Obstacle Detection Using Adaptive Color Segmentation and Color Stereo Homography. pages 705–710, 2001.

[13] J. Sagar and A. Visser. Obstacle avoidance by combining background subtraction , optical flow and proximity estimation. *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014*, pages 142–149, 2014.

[14] S. Saha, A. Natraj, and S. Waharte. A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment. *2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology*, pages 189–195, 2014.

[15] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[16] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation for autonomous rotorcraft MAVs in complex environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1758–1764, 2013.

[17] I. Ulrich and I. Nourbakhsh. Appearance-Based Obstacle Detection with Monocular Color Vision. *Artificial Intelligence*, (August):866–871, 2000.

Appendix A

Literature review

Nowadays, more and more exploration and surveillance tasks are performed by MAVs. Due to their small size and great flexibility, they are a useful resource for diverse types of missions. They can be used to fly in restricted and/or dangerous areas to search for people or to detect dangerous objects. This can not only be done by using a single Micro Aerial Vehicle (MAV), but also with swarms of MAVs. Furthermore, MAVs are not limited to outdoor environments, but are useful to explore indoor environments as well. However, in indoor environments, Global Positioning System (GPS) and a priori information on the environment are usually not available. The variety of obstacles can be large, while the manoeuvring space is small. Thus, there is a need for autonomous, robust MAVs, capable of avoiding obstacles in unknown indoor environments. Due to the limited indoor space and to allow for MAV swarming, the payload capacities of MAVs are limited. To extend the mission time of the MAV, the payload and thus on board computational efforts should be minimized or performed off board. Active sensors such as Lidar and Kinect cameras can be used for obstacle detection and avoidance but are heavy (Shen, Mulgaonkar, Michael, & Kumar, 2013) and power consuming (Bills, Chen, & Saxena, 2011). Stereo vision requires a minimum baseline (Achtelik, Achtelik, Weiss, & Siegwart, 2011), that limits the useful range (Celik, Chung, Clausman, & Somani, 2009) and requires that the cameras see the same object at the same time. Furthermore, it puts a minimum to the size of the MAV and the object position in the image must be sufficiently different to be able to triangulate the position and estimate the range (Mcfadyen & Mejias, 2015). In addition, stereo images require more intensive computation power than monocular images (Sa, He, Huynh, & Corke, 2013).

The solution proposed here, is to use a monocular camera as single sensor to avoid obstacles in unknown indoor environments. The method proposed for obstacle avoidance is to use floor recognition to detect obstacles by classifying pixels as belonging to the floor or to an obstacle.

A-1 Obstacle detection

The first part of this literature review is focused on obstacle detection. One way to avoid obstacles is by determining the distance to an object via depth perception. Depth perception

methods can rely on motion, but also on static cues such as perspective and texture. First, as background information, the feature detection methods that are used in the various obstacle detection methods, are discussed. Next, the methods that use motion are presented, followed by the static methods. Finally, the research on image segmentation, used for obstacle detection, is presented.

A-1-1 Feature detection

In this section, the most common feature detection methods are discussed. They are commonly used in many obstacle detection or image comparison techniques and serve as an elucidation to the sections on depth perception.

Canny Edge Detection

The Canny edge detector can be used to detect edges in images (Saxena, Chung, & Ng, 2005). It uses a gray scale image to produce an image showing the positions of tracked intensity variations. The image is first smoothed by Gaussian convolution, to which a first derivative operator is applied. Ridges are caused by edges, and non-maximal suppression is applied to the pixels that are not actually on the top of a ridge, to create a thin output line. A problem with the basic canny operator occurs when one of the edges is partially occluded by an object (Canny, 1986).

Harris Corner Detection

The Harris corner detector uses the auto-correlation function of a signal to measure the local changes in the signal. This is done by shifting patches a little, in various directions. It is invariant to rotation and illumination.(Derpanis, 2004). It is also immune to noise and affine transformations (Celik et al., 2009). Its features are high quality, but the approach is computationally expensive (Rosten & Drummond, 2006). The detector also has difficulties with tracking agile motion (Celik et al., 2009).

SUSAN Corner Detection

Smallest Univalue Segment Assimilating Nucleus (SUSAN) is based on the association of each image point with a local area of similar brightness. SUSAN generates high quality features, that can be used for corner detection and detects corners as the occasion where the centroid is not near the nucleus. It is not affected by noise but is computationally expensive. (Rosten & Drummond, 2006), (Smith & Brady, 1997)

FAST Corner Detection

Features from Accelerated Segment Test (FAST). uses a segment test criteria by considering a circle of pixels around a candidate corner. The candidate corner is classified as a corner is a set of pixels in that circle exist that are brighter than the intensity of the candidate pixel. It is a FAST high quality corner detection method. The method however, is dependent on a threshold and sensitive to noise. (Rosten & Drummond, 2006)

Shi and Tomasi Corner Detection

The Shi and Tomasi corner detection technique works by minimizing the differences between past images and the present image in a sequence of images. (Celik et al., 2009) The approach assumes that the deformation of features is affine. (Rosten & Drummond, 2006) The corner detection technique can be used for feature detection and tracking as well. It cannot make a distinction between a useless feature and a landmark. (Celik et al., 2009)

Multi-Scale Oriented Patches

Multi-Scale Oriented Patches (MOPS) is a quick and accurate corner point matching method (Sagar & Visser, 2014) that identifies the same points among multiple images. It can be used to determine the distances to obstacles. (J. Lee, Lee, Park, Im, & Park, 2011) The method, however, is computationally expensive. (Saha, Natraj, & Waharte, 2014)

Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is an image matching and recognition technique to detect and describe features. The image data is transformed into scale invariant coordinates that are relative to local features. (Lowe, 2004), (Saha et al., 2014). It can be used for obstacle detection (Saha et al., 2014) and scene recognition (Lowe, 2004). It is very useful since it is invariant to scaling, rotation, illumination and the camera viewpoint. (J. Lee et al., 2011),(Lowe, 2004) Its features are hihgly distinctive and it is also not very sensitive to noise (Lowe, 2004). However, the method is computationally expensive (Saha et al., 2014) and the specific features are not always useful in an unknown environment (Soundararaj, Sujeeth, & Saxena, 2009). Furthermore, the quantity of features is important (Lowe, 2004) and the method had difficulties with determining the size and shape of objects (J. Lee et al., 2011).

Speed Up Robust Features SURF

Speed Up Robust Features (SURF) is a local feature detector and descriptor, that is partly inspired by the SIFT descriptor since it is also invariant to scaling and rotation and computationally less expensive than SIFT(Saha et al., 2014). It can be used for obstacle detection (Saha et al., 2014), (Mori & Scherer, 2013) and can be useful to detect relative size changes. (Mori & Scherer, 2013). The approach is robust but requires texture to work (Mori & Scherer, 2013).

A-1-2 Depth perception from motion

Depth can be perceived by using motion to determine the optic flow of the scene or the optical expansion of the objects.

Optic flow

The concept of optic flow can be explained as the apparent motion due to the relative motion between the camera and the scene. (Lippiello, Loianno, & Siciliano, 2011) It is found that in nature, bees use optic flow to control their flight and humans to control their balance. The optic flow field contains spatio-temporal information about the environment. This can be described by a set of vectors that characterize the features in the image space. (Arokiasami

& Chen, 2015) Optic flow is related to the Time To Contact (TTC) (Mori & Scherer, 2013), since the flow of the image points away from the focus of expansion can be used to estimate the TTC. (de Croon et al., 2012) This can be done without knowing the velocity or distance to an object. (Arokiasami & Chen, 2015)
Optic flow is useful for navigation (Arokiasami & Chen, 2015) (Wang, Liu, & Meng, 2015) for wall following (Mori & Scherer, 2013), and obstacle avoidance (Arokiasami & Chen, 2015)(Wang et al., 2015) (de Croon, de Wagter, Remes, & Ruijsink, 2012). It can also be used for depth map reconstruction (Lippiello et al., 2011). However, computing the optic flow is computationally heavy (Alenya, Negre, & Crowley, 2009),(Saha et al., 2014) and not very accurate (Alenya et al., 2009). It it also is sensitive to noise (Ancona, n.d.), (Wang et al., 2015), (Dey et al., 2015). Furthermore, it has difficulties with frontal obstacles (Saha et al., 2014), (Alvarez, Paz, Sturm, & Cremers, 2014), (Mori & Scherer, 2013), (Sagar & Visser, 2014), does not work in textureless (man-made) environments (Mori & Scherer, 2013),(de Croon et al., 2012) and drifts over time (Achtelik et al., 2011). It also has difficulties with objects at distances over 5 meters (Arokiasami & Chen, 2015) and it requires stable tracking (Bipin, Duggal, & Madhava Krishna, 2015).

In literature, optic flow is used in many studies. The work of (Arokiasami & Chen, 2015) uses optic flow to estimate the TTC to avoid obstacles. The TTC estimate is accurate, as long as the distances to the obstacles are relatively small. The method, however, is only tested on simulations. Optic flow can also be used to compute depth maps, which can be used for obstacle avoidance, as done in (Lippiello et al., 2011). This approach uses an omnidirectional camera and an IMU. The testing of this approach is done on simulations of a very long and width-varying indoor environment. Another way to use optic flow, is to detect obstacles on a ground plane, by looking at the variation of the optic flow modulus at one point a the ground, as done in (Ancona, n.d.). The camera in this study, directed towards the ground, is mounted on a slowly moving robot, with only 3 degrees of freedom. Obstacles can only be detected when they are close enough to the camera. In (Wang et al., 2015), the TTC is determined by combining optic flow with machine learning. The method uses a combination of dense and sparse optic flow and the algorithm is tested both with a camera directed towards objects mounted on a table and with an MAV in an indoor garage environment. The method works 75 % of the tested cases.

Optical expansion

The rate of optical expansion can be used as a monocular depth cue. It relies on the fact that objects appear to increase in size when approaching them. The rate of optical expansion can be used for obstacle avoidance, since it can be used to determine the TTC (Mori & Scherer, 2013), (Sagar & Visser, 2014). It does however, require texture and it is dependent on the image resolution (Sagar & Visser, 2014).

The relative change in scale of SURF features alone can be used to determine the TTC, as done in (Mori & Scherer, 2013). Reactive control based on the TTC is used to avoid obstacles, with the height of the MAV known from the sensor. The algorithm is tested with an MAV in a forest. Problems occur due to the slow response time of the MAV. Also,

the algorithm requires texture to work. Others, such as (Chavez & Gustafson, 2009), use SIFT features for scale comparison in consecutive images. It requires straight ahead flight without turns at a constant velocity to work. The images must also be taken at constant time intervals. This method is tested on images of simulated near collisions and gives good performance only when objects are at a sufficient distant from the camera and the scale of the features does not increase with a scale higher than 2. Improvements can be made to minimize false collision detections and missed potential collisions. In (Byrne & Taylor, 2009), visual collision detection is combined with maximum likelihood estimation of the TTC, to localize significant collision danger regions in a forward looking aerial video. The method divides pixels in collision and non collision regions of joint probabilities and is tested on flight data from a small air vehicle. Another way to determine the TTC, is by using Scale Invariant Ridge Segments (SIRS), which uses the variation of the intrinsic scale. Features can be detected with a feature detector such as SIFT. (NEgre, Braillon, Crowley, & Laugier, 2008) uses the extremal points in the Laplacian space for image matching and comparing pixels to the neighbouring pixels. The algorithm is tested on a moving vehicle with a camera and laser range finder, that is used to find manually placed trackers. The TTC estimations are quite accurate, but are initially unstable. In (Alenya et al., 2009), a comparison of Active Contour Affine Scale (ACAS), SIRS and Image Brightness Derivatives (IBD) is made to estimate the TTC. The method proposed, uses ACAS, which uses a set of projected points between two different frames with an affine transformation to determine the TTC. They test the method on a simulation, where the ground truth is assumed to be exactly known and all contours are clearly visible. When the ground truth is not exactly known or the contours are not clearly visibly, the method does not work. This is also the case when the illumination changes, when the velocity of the vehicle is not constant or when the camera frame rates are perturbed. This method outperforms SIRS and IBD, but only in a perfect testing environment. IBD is used in (Horn, Fang, & Masaki, 2007), which gives reliable results as long the images are averaged and sub-sampled. When the sequence is close to its end, the results are not reliable because the temporal aliasing cannot handle too much change between two frames. The authors in (Saxena et al., 2005) propose a supervised learning approach to predict the depth map of monocular images, based on the change in scale and image intensities. The system is trained with a dataset of which the depths are known. Their algorithm works well for relative depths, since it has no problems with shadows, but gives faulty values for absolute depths. The majority of errors occur due to objects that differ from the training data.

Visual Odometry

Visual Odometry (VO) examines the changes in the images caused by motion to incrementally estimate the pose of the vehicle. To determine the changes in the images, features need to be detected. The approach aims at local consistency rather than global consistency of the path. VO is Simultaneous Localization and Mapping (SLAM) before the loop is closed and thus can be used as a building block for SLAM. (Scaramuzza & Fraundorfer, 2011)
VO is applicable for pose estimation in GPS denied environments. However, it does suffer from drift (Scaramuzza & Fraundorfer, 2011) and scale ambiguity. (Daftry et al., 2015) The method is also computationally expensive due to the feature matching.(Engel, Sturm, & Cremers, 2013), (Daftry et al., 2015), (Bills et al., 2011)

(Semi) Dense Visual Odometry

Semi dense Visual Odometry (SVO) is a direct approach for VO, that uses the image intensities to map the motion of an object in a scene. It uses probabilistic models to build semi-dense depth maps.(Shridhar, 2015)
Since it is a direct method, no feature matching is needed (Engel, Sch, & Cremers, 2014). This makes the approach computationally less expensive (Shridhar, 2015). It is also more accurate and robust than feature-based methods (Engel et al., 2014), (Engel et al., 2013), (Shridhar, 2015). Furthermore, it can be used in GPS denied, indoor environments (Schops, Engel, & Cremers, 2014). However, this approach requires a large dataset and the scale of the environment cannot be recovered (Daftry et al., 2015). The global map of the environment is not considered and the method is highly sensitive to different lighting situations (Shridhar, 2015).

In (Daftry et al., 2015), the camera pose of new frames is estimated based on a known semi-dense inverse depth map of an image in combination with direct image alignment. The depth measurements are obtained by a probabilistic approach for adaptive-baseline stereo. It is assumed that the camera rotations are small. The scale is obtained by an onboard sensor. The method is tested and verified on an MAV in an indoor environment. Finally, testing is done in an outdoor, cluttered environment, but the MAV is attached to a tether to restrain its motion. (Engel et al., 2013) also uses semi-dense visual odometry. Their approach continuously estimates a semi-dense inverse depth map for the current frame to estimate the pose of the next frame. A prior knowledge about the depth of pixels is used to limit the search range. It uses stereo depth measurements to improve accuracy. The approach is tested on a publically available dataset, with a hand hold camera. The calculations are done on a laptop. This approach fails in case of pure camera rotation because it cannot determine the depth of new regions. In (Forster, Pizzoli, & Scaramuzza, 2014), a method is proposed that uses the pixel intensities directly, which improves the accuracy and decreases the computation time. Testing of the algorithm is done on datasets recorded from a downward looking camera, mounted on an MAV and on image sequences from a handhold camera. Best results are obtained when the scene has little and repetitive texture. The system suffers from drift. Visual odometry is promising for Augmented Reality (AR), the approach in (Schops et al., 2014) uses semi-dense visual odometry to estimate the depth map of the currently available image by direct image alignment and by filtering over many small pixel-wise stereo comparisons. The algorithm can run in real time on a smartphone and can be used for AR. Problems occur with rotation and forward translation. To prevent overestimation, the camera must move at a certain minimum velocity. In (Alvarez et al., 2014), the proposed method tries to compute a dense depth map from a structure from motion approach with a small set of images to determine an obstacle free path. It combines Dense Tracking and Mapping (DTAM) with feature-based methods, but without loop closure. The set of images is gathered by hovering at known time intervals. The images are in gray scale. An inverse depth map and cost volume are then calculated based on a depth guess of the initial image. The depth map is very noisy. To test the approach, an MAV with monocular camera has to avoid obstacles and doors. The MAV hovers a couple of seconds in front of an obstacle, to calculate the depth map, which is done off board. Doors lead to failures, since they are narrow.

Simultaneous localization and mapping

SLAM is used to obtain a globally consistent map of the environment by loop closure, while simultaneously estimate the pose of the robot in GPS denied environments(Scaramuzza & Fraundorfer, 2011). It can also be used for obstacle avoidance (de Croon et al., 2012). Furthermore, it is accurate and robust against camera occlusion (Weiss, Achtelik, Kneip, Scaramuzza, & Siegwart, 2011). However, it is computationally heavy, since it relies on feature matching (Engel et al., 2013), (Daftry et al., 2015), (de Croon et al., 2012). Incremental SLAM is computationally less expensive than regular SLAM but has no loop closure(Shen, Michael, & Kumar, 2011).

In (Weiss et al., 2011) visual SLAM is used for pose estimation. To reduce computational costs, the feature points that are used for localization and control are also used for the 3D terrain reconstruction mesh. Approach has difficulties with concave terrain situations and scenes with little texture. Their final map is also never globally consistent. To build accurate 3D models, monocular visual SLAM is combined with dense visual reconstruction techniques in (Paalanen, Kyrki, & Kamarainen, 2008). The technique is tested with a hand hold camera. The approach in (Shen et al., 2011) uses a simplified occupancy grid-based incremental SLAM algorithm. The sensors used are an IMU and a laser range sensor. A vocabulary of SURF features from different environments is constructed offline. The goal locations must be set by a user on the map. Testing is done on an MAV in an indoor environment. Method suffers a little from drift. To create a map from one floor to another, the MAV is carried to the next floor. In (Heng, 2014), pose estimation is done with pose graph SLAM. An optic flow sensor and stereo camera are used. All computations are done off board and communicated via a Wifi connection. However, this connection is not reliable, which makes it not possible to use the global pose estimations of SLAM. The algorithm is tested in both an indoor and an outdoor environment. The method requires a flat ground, because of the optic flow sensor. For the indoor experiment, the MAV has to fly through a straight corridor with an open space in the middle. The visual odometry does not work properly due to poor lightning conditions or over lighted environments and the lack of texture. The MAV is dependent on the optical flow sensor. The pose estimation significantly drifts downwards. For the outdoor experiment, the MAV has less problems with drift. However, the MAV sees phantom obstacles. (Engel et al., 2014) uses an LSD-SLAM algorithm to construct large-scale consistent maps of the environment. Testing is done on a publically available Red Green Blue - Depth (RGB-D) dataset. A 500 m trajectory needs 6 minutes for loop closure.

Parallel Tracking and Mapping

Parallel Tracking and Mapping (PTAM) is a feature-based SLAM algorithm. It tracks and maps large amounts of features, which makes it more robust. Because of the parallel threading of motion estimation and mapping, the algorithm can run in real time. It often uses keyframe based Bundle Adjustment (Forster et al., 2014). The algorithm is relatively simple and computationally inexpensive (Shridhar, 2015). It does not work well in large scale environments (Forster et al., 2014) (Mercado, Castillo, & Lozano, 2015) and it does not provide the absolute scale (Mercado et al., 2015). Furthermore, no structural meaning can be extracted from the reconstructed environment and the user must be trained to work with this algorithm (Shridhar, 2015).

Dense Tracking and Mapping

DTAM computes a dense depth map for each keyframe. It is a direct method that minimizes a global spatially regularised energy function. It estimates the camera pose through direct whole image alignment with the depth map.(Forster et al., 2014). The tracking and reconstruction of the environment relies on dense every pixel methods.(Newcombe, Lovegrove, & Davison, 2011) More data is available from the environment (Alvarez et al., 2014), which means accurate depth maps can be created (Alvarez et al., 2014) (Newcombe et al., 2011). This approach is computationally expensive and requires Graphics Processing Unit (GPU) parallelization (Forster et al., 2014) (Newcombe et al., 2011). For good results, the camera must move slowly and make enough translational motions (Alvarez et al., 2014). The method is sensitive to illumination (Newcombe et al., 2011) and breakage (Bipin et al., 2015).

A-1-3 Depth perception from perspective

Perspective cues are monocular cues to perceive depth. It uses the converging of vanishing lines to determine waypoints and can be used for navigation in GPS denied environments, since it works well for wall following or corridor centering applications (Bills et al., 2011), (Saha et al., 2014), (Mori & Scherer, 2013). No 3D model is needed and the algorithms are computationally inexpensive. Furthermore, no high resolution camera is needed.(Bills et al., 2011). In unstructured environment, however, this approach does not work well. (Saha et al., 2014), (Mori & Scherer, 2013) It also has difficulties with frontal obstacles (Mori & Scherer, 2013).

In (Bills et al., 2011), a method is proposed that first classifies the type of indoor environment the MAV is in. It then uses edge detection to find vanishing points to estimate the preferred flight direction. An extra sensor is used to determine proximity to walls and it uses sonar sensors to detect openings in the walls. Testing of the method is done by flying an MAV in a variety of indoor environments, with corridors and staircases. Difficulties arise when flying in dark and low contrast environments. The MAV has problems due to its turbulence and control. (Garcia, Mattison, & Ghose, 2015) uses the visual contours of the environment to navigate an MAV through corridors and turns. It uses vanishing lines as markers to determine proximity to walls. Image processing, flight trajectory planning and control are done off board. Testing is done with a Parrot AR. The system is successful in corridors but encounters problems when an empty wall is seen for a long time where no lines can be detected due to too little contrast. Furthermore, narrow halls give problems for the large MAV and intersections are only detected when the distance to the intersection is sufficiently large. In (Kessler, Ascher, Frietsch, Weinmann, & Trommer, 2010) an attitude estimation algorithm is created for indoor navigation, based on vision and supported with a laser scanner. It uses the geometrical constraints of indoor environments, such as walls, doors and windows, because they have parallel or perpendicular edges. It compares different line extraction techniques. Testing is done on the Integrated Pedestrian Navigation System, a system to help the blind with indoor navigation. Another approach such as in (Celik et al., 2009), combines the use of straight lines, relative height, texture gradient and motion parallax with FAST SLAM. The straight lines of the ground and ceiling are used to determine the infinity point, which is used for state estimation and range measurements. The depth estimation is done with the altitude above the ground known from the altimeter and the assumption that the landmarks

are stationary. The landmarks are extracted with feature detection. For this, the Shi and Tomasi detector is used. Multiple sensors are used and all computations are done on board of the MAV. The system is limited by the availability of good corners and the capabilities of the camera. In (Okada, Taniguchi, Furukawa, & Onoguchi, 2003), vehicles on the road are detected with a monocular camera. The method uses a combination of the cross ratio and vanishing lines. An obstacle is defined if the motion of a group of three horizontal segments satisfies the motion constraint better than the ground plane motion constraint. The vanishing lines vary when pitching and rolling. The algorithm is tested on a simulation and has difficulties with detecting distant vehicles. It also has difficulties to detect obstacles in poor lightning conditions and noise data. The approach in (Kim & Kwon, 2015) uses vanishing points and the obstacle angle for obstacle avoidance and lane following. The obstacle detection area is dependent on the moving speed of the robot and the obstacle angle remains zero in the obstacle detection area as long as there is no object in that area. Testing is first done by simulating an environment with cylindrical obstacles with 1 m diameter on the path for different velocities of the robot (between 0.1 and 0.5 m/s). It is then tested on a real robot with a webcam, in an indoor environment of a 70 m long hallway of 2.5 width, which is proven to work. It would not work in an outdoor, unstructured environment.

A-1-4 Depth perception from texture

The variation in texture and/or color provides information about the depth in an image. This monocular cue is also called the appearance variation cue. It is based on the assumption that, when the camera is close to an obstacle, there is less variation in the image. It is a valuable addition to optic flow, that can also be used for frontal obstacle detection. The method does not require motion of the observer or accurate sub-pixel measurements. Furthermore, it is robust to noise.(de Croon et al., 2012) The approach is, however, heavily influenced by blob-based texture variations in the environment and its performance is limited in outdoor situations. (Sagar & Visser, 2014)

The authors in (de Croon et al., 2012) proposes a method that uses texture as monocular cue to detect obstacles. The algorithm is tested on the DelFly II in two different office environments, with the rudder controlled off board and the height controlled by a human pilot. The computations are done off board. The MAV can detect collisions when it is not in a turn. The appearance variation cue needs to be complemented with other cues to be useful. In the work of (de Croon et al., 2012), the authors use a combination of the appearance variation cue and time to impact to detect obstacles. The height of the MAV is controlled with a pressure sensor. The magnitude of the flow is used for obstacle avoidance. The testing is done in an indoor office environment. Collisions only occur due to faulty decisions made by the control system. In (Bipin et al., 2015), the authors propose a support vector machine approach to estimate the depth map from a sequence of images, generated by a monocular camera, mounted on an MAV. The perception framework generates a minimum time collision free flight path and control approach to fly through an unknown, cluttered environment. The system is trained with an existing dataset of images, which are divided into small patches that must contain texture energy, texture gradients and haze, with known distances to the nearest obstacle. The system can then accurately give estimations of the distances to the surrounding obstacles. The algorithm was tested on an MAV in indoor and outdoor environments similar

to the training environment. Failures mainly occur due to a too narrow field of view.

A-1-5 Image segmentation

Image segmentation is a summarizing term created to describe all techniques that segment images to classify features in the scene. In indoor scenes the floor or ceiling can be classified as non-obstacle and everything else as obstacle, for outdoor scenes this can also be done, using the sky as non-obstacle. Defining a piece of the image as floor or ceiling can be done by using the structure of the scene for indoor environments, or by assuming a piece of the image must be part of the floor or ceiling because of the location in the image. The properties of floor or ceiling can be compared to the properties of the rest of the images based on color, intensity, texture or a combination of these.

In (Ulrich & Nourbakhsh, 2000), image pixels are classified to belonging either to the ground or to an obstacle, based on its color appearance. The robot is trained to recognize the appearance of the ground over multiple observations. This ground reference area must not be free of obstacles to be used as reference. The ground however, must be flat to estimate the distance to obstacles and have a different appearance than the obstacles. Furthermore, no overhanging obstacles may be present. Problems occur when one or multiple of these assumptions is violated. Many improvements to this approach are still possible, such as using different color spaces or complement the approach with the texture variation cue. Similarly, in (Parag & Singh, 2001), the system is trained to recognise traversable areas with several images of grass taken in various lightning conditions. The system classifies pixels either as obstacles or as freespace, based on their color appearance. This works well for areas with a constant ground color. For testing, only a small outdoor area is used, with non varying ground. Objects are always recognised but false positives occur due to varying sunlight conditions. In (Ta, Ok, & Dellaert, 2014), landmarks far away from the MAV are used to determine an obstacle free trajectory. It is assumed that the size of these landmarks does not change significantly when approaching them. The relative size change is used to determine the distance. Furthermore, wall-floor features are used, that uses the perpendicular intersection of the walls with the horizontal floor, to recognize walls. The use of these features is combined with PTAM, to also be able to avoid lateral collisions and to generate a map of the environment which can be used for path planning. Problems occur due to failure to detect landmarks and the instability of the drone itself. Furthermore, the method has difficulties with large area corner turning and finding openings in lateral directions. It is also sensitive to thresholds, lightning conditions and the camera quality. The work in (De Croon, De Wagter, Remes, & Ruijsink, 2011) classifies images in sky and non-sky regions. The system is trained by learning decision trees, based on a large database of training images. The MAV avoids obstacles by ascending to a height higher than the highest obstacle in its trajectory. If the state estimates are available, this approach is successful in avoiding all obstacles. This is also the case when the states are not available, however, the MAV ascends too much in this case. The method only provides information on the relative yaw directions of possible obstacles. Large cities and mountains will lead to too high ascendance, which might lead to a too low ground resolutions. The authors in (Wedel, Schoenemann, Brox, & Cremers, 2007) propose a method for FAST and accurate obstacle segmentation. The method is to be used for ground traffic vehicles. They segment the image in three parts: the ground, the background and the obstacle, based on intensities. Based on the set up of the camera and the camera parameters, the horizon can

be determined. The horizon is then used as a separation between the street and background. The motion of the obstacles and the rest of the image are calculated separately. The method is sensitive to noise and has difficulties with occlusions of the ground plane by the obstacle. The method proposed in (B. Lee, Daniilidis, & Lee, 2015), the system is trained to recognize the appearance of the ground by analyzing the image and use the properties of the ground to recover the scale of the scene. The scale can be retrieved geometrically by using the height of the scene, which is known due to its fixed position. The appearance of the ground can be related to the geometric ground estimation by color space classification. Testing of the method was done on a publicly available dataset. (Verbickas & Whitehead, 2014) uses convolutional neural networks to classify the image into ground and sky segments. To achieve this, they attempt to recognize the horizon by classifying pixels as belonging to the ground or to the sky, dependent on whether a pixel is black or white. The method performs well when the image is well illuminated, but has difficulties with scenes where the color differences between sky and ground are small. The authors in (Gini & Marchi, 2002) classify pixels as belonging to the floor or to obstacles, by analyzing the intensity and the saturation of the pixels. They assume that the texture of the floor is uniform. The method cannot detect frontal obstacles without using sonars.

Various techniques exist to avoid obstacles. Which technique or combination of techniques is useful, is dependent on the application. Many work on optic flow exist, however, since optic flow is less suitable for textureless (indoor) environments and it has difficulties to detect frontal obstacles, the literature on optic flow discussed here, is limited. To determine the relative scale change of obstacles in images, the objects need to be detected first. This is generally done with SIFT or SURF features. SIFT can be used for obstacle detection (Saha et al., 2014) and scene recognition (Lowe, 2004). It is very useful since it is invariant to scaling, rotation, illumination and the camera viewpoint (J. Lee et al., 2011)(Lowe, 2004). Its features are hihgly distinctive and it is also not very sensitive to noise (Lowe, 2004). However, the method is computationally expensive (Saha et al., 2014) and the specific features are not always useful in an unknown environment (Soundararaj et al., 2009). Furthermore, the quantity of features is important (Lowe, 2004) and the method had difficulties with determining the size and shape of objects (J. Lee et al., 2011). SURF has similar properties to SIFT, but it is computationally less expensive (Saha et al., 2014). When the objects can be detected, the TTC is relatively easy to determine. Visual odometry and SLAM algorithms are generally more applicable to determine the pose of the MAV and to create a map of the environment, than for obstacle avoidance. Perspective cues are a computationally inexpensive solution for wall following approaches in indoor environments. Perspective cues can take advantage of the structure of indoor environments, but require a completion to deal with frontal obstacles. To determine the vanishing lines, an edge detector is needed. Another useful monocular cue, is texture. Its applications are often in indoor environments in combination with other monocular cues. The last method discussed here, are the image segmentation techniques. For obstacle avoidance, this has great potential. It is simple, since part of the image can be classified as obstacle free and the rest as obstacles. However, the reference part of the image may, and probably will change. Therefore, to keep this approach robust, the reference image must be updated. The distance to the obstacles can be determined by using geometry and assuming a fixed height and angle, as already done in literature.

A-2 Obstacle avoidance

This part of the literature review focuses specifically on obstacle avoidance techniques, where it is assumed that obstacles are being detected. Obstacle avoidance control techniques can be classified in two different classes. The techniques are either global or local. Global techniques generally are based on a priori information, which can also be referred to as motion planning. Local obstacle avoidance techniques are based on sensory information and are also referred to as reactive navigation.

A-2-1 Motion planning

Global obstacle avoidance approaches can be used in known or partially known environments. They are relatively slow, due to the complexity of the required motion planning. This makes them not suitable for FAST obstacle avoidance. These methods also experience major problems when the global world model is inaccurate or when the model is not available. (Fox, Burgard, & Thrun, 1997), (Heng, Meier, Tanskanen, Fraundorfer, & Pollefeys, 2011).

Probabilistic Road Maps

Probabilistic Road Maps (PRM) is an algorithm that preprocesses a 3D world model offline, to generate a roap map graph. To plan the path towards a goal location, the method samples the configuration space for collision-free locations. These collision-free locations are added as nodes to the road map graph. The planning quality increases when increasing the number of nodes on the road map graph, however, increasing the nodes will increase the time needed to generate a plan. Online, the initial location and the goal locations are connected to the roap map configurations. (Wzorek, Kvarnstrom, & Doherty, 2010) (Geraerts & Overmars, 2004)

Rapidly Exploring Random Trees

The road map in the Rapidly Exploring Random Trees (RRT) algorithm is created online. It uses a random search approach from the start and end configurations, to explore the configuration space. With this information, two trees are generated. The algorithm tries to connect these trees at specific time intervals. The advantage of RRT is that it is no preproccessing is needed and that RRT planning can be used in situations where PRM planners cannot be used. However, the path planning quality of RRT is less than that of PRM planners. (Wzorek et al., 2010)

Anytime Dynamic A

The Anytime Dynamic A* algorithm is planning and replanning algorithm, based on graphs that generates a bounded suboptimal, but valid, solution to a problem, even if it is interrupted before it finalizes. The algorithm improves its solution and corrects it when new information is available.(Likhachev, Ferguson, Gordon, Stentz, & Thrun, 2005)

A-2-2 Reactive navigation

Local obstacle avoidance techniques only require a fraction of the world model. This makes reactive navigation computationally inexpensive. However, because the techniques are local,

optimal solutions cannot be generated. Furthermore, the approach easily gets trapped in local minima, such as obstacle configurations that are U-shaped. (Fox et al., 1997), (Heng et al., 2011)

Artificial Potential Field

The Artificial Potential Field (APF) method presents the goal as an attractive potential and the obstacles encountered as repulsive potentials. This way, obstacles can be avoided without being computationally expensive. (Sezer & Gokasan, 2012) This technique is relatively easy but has as drawback that it gets stuck in local minima. This happens when the attractive potential and repulsive potentials cancel each other out (Sezer & Gokasan, 2012).

Bug algorithms

Bug algorithms move directly towards a goal destination, usually by wall following, until an obstacle is encountered. The obstacle is then circumnavigated until the goal location is reachable again. (Sezer & Gokasan, 2012) Bug algorithms are easy to tune but can be time consuming (Kim & Kwon, 2015). Furthermore, the robot often comes too close to obstacles. (Sezer & Gokasan, 2012)

Vector Field Histogram

The Vector Field Histogram (VFH) divides the environment in several segments. The obstacle density according to the polar histogram is then used to determine the obstacle free path. When considering the potential turning radii and robot dimensions, obstacles can be avoided more smoothly, which is called VFH+. (Kim & Kwon, 2015), (Sezer & Gokasan, 2012). The approach is goal oriented, but considers only holomic constraints. (Sezer & Gokasan, 2012)

Dynamic Window approach

The Dynamic Window Approach (DWA) approach, is a reactive approach that considers a short time interval to determine what the next steering direct needs to be. The trajectory can then be simplified to a 2D search space with translational and rotational velocities, that are small enough to stop the robot safely. (Fox et al., 1997) In this search space, the optimal obstacle avoidance velocities are determined within the dynamic window (Kim & Kwon, 2015). The kinematics, dynamics and geometric constraints are applied directly to the velocity space and can thus move the robot safely at high velocities (Li, Wu, & Wei, 2006). However it has problems with dead ends (Kim & Kwon, 2015),(Li et al., 2006).

Nearness Diagram

The Nearness Diagram (ND) is a reactive approach that analyzes the information available from the obstacles in the images to determine the obstacle free path to the desired goal location. The method creates actions suitable for a set of situations. (Li et al., 2006) The ND method is computationally inexpensive (Li et al., 2006) and beneficial in crowded areas (Kim & Kwon, 2015). However, the robot kinematics and dynamics are not taken into account when determining the motion commands (Li et al., 2006) and it assumes a holonomic and circular robot.

Follow the gap

The Follow the Gap Method (FGM) searches for the gap between obstacles and calculates the gap angle while taking a goal point into consideration. This way, the robot is always led towards the center of the maximum gap, while keeping the goal point into account. The advantages of the approach are that the trajectories are safer overall and that the approach considers nonholonomic robots and the field of view constraints. Furthermore it is easy to tune since it has only one tuning parameter.(Sezer & Gokasan, 2012) The method, however, is unable to avoid U-shaped obstacles. (Kim & Kwon, 2015)

In this literature study, the focus lies on unknown environments. Therefore, motion planning approaches are less suitable than reactive approaches. Hybrid approaches exist, combining motion planning and reactive methods and can be used when part of the scene is known. All reactive methods discussed here, have trouble getting stuck in local minima. Therefore, the an additional mode must be programmed to deal with local minima situations.

Appendix B

Preliminary Analysis

As it was introduced in Appendix A, the methods that allow low computational effort and can be applied to unknown, indoor environments, are perspective cues and the image segmentation (floor recognition) techniques. These allow for obstacle avoidance without a priori knowledge on the environment or obstacles present. To obtain a first indication of potential problems that may occur when attempting to analyze the image, a series of videos were made in the indoor office environment of the Faculty of Aerospace Engineering building of the TU Delft. Some images were taken from the video, as can be seen in figure $B-1$, to demonstrate potential difficulties that may occur, when processing the images.

A first visible difficulty, can be seen in figures $B-1$, is that the floor in all pictures looks slightly different. This is due to different lightning conditions that occur at different positions in the corridor. The floor in all the images of figure $B-1$, is exactly the same floor. The other part of the floor, that has a darker color than the floors in the first 6 figures is also visible. Here, the same problem occurs, since the color appearance of the floor differs too with varying lightning conditions. It can thus be concluded that the color of the floor is not constant. To show the effect of this, the images are processed in Matlab, by using a large part of the bottom row of the image that belongs to the floor and comparing the colors of the reference region to the colors of the rest of the image. The results can be seen in figure $B-2$.

Another difficulty that can be observed from the images in figure $B-1$, is that the lines visible in the image are not all straight, which might lead to problems when searching for vanishing points. Furthermore, lines are visible that are not actually of any use, but are visible due to floor transition. To show the influences of these, various images have been processed by Matlab, by detecting the lines in the images via a canny edge detector, followed by a Hough transform to identify the lines in the image. The results can be seen in figure $B-3$. Furthermore, not in all images sufficient perspective cues are visible to determine the vanity point. This occurs in general when approaching a wall, or when the camera angle is not sufficiently pointed downwards. Another problem that might occur is when the walls of the corridor are not parallel. This makes it also difficult to determine the correct vanishing point.

Figure B-1: Examples from the initial dataset created at the Faculty of Aerospace Engineering of the TU Delft

As can be seen from figure $B-3$, the canny edge detector detects many different lines in the image. It can be seen that the transition from one floor color to another, due to shadow and the one due to paint, are detected as edges/lines. For the Hough transform, this can be solved by adjusting the parameters of the function, however, this also results in less actual lines being identified by the Hough transform. The curved walls, are recognized as multiple smaller straight lines, most probably due to the large radius of curvature of the walls.

Finally, the texture in the images was also determined, as can be seen in figure $B-4$. It can be seen that the floor has less texture than the rest of the scene. However, this can not be generalized.

As an initial solution to the limited floor visibility and the changing appearance of the floor, it is proposed to use the bottom camera of the MAV, that is used for stabilization, to create the reference image of the floor.

Figure B-2: Matlab color processing of images

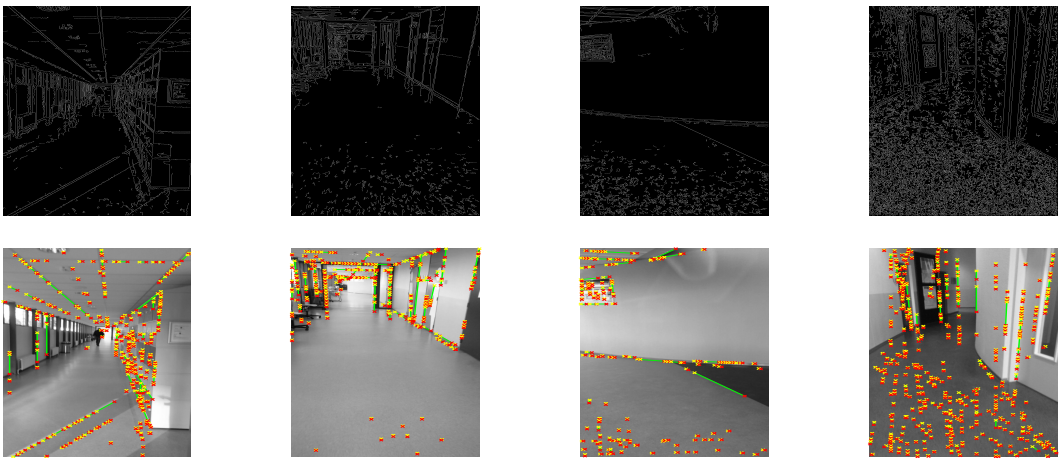

Figure B-3: Matlab canny edge detection and Hough transform processing of images

Figure B-4: Example image where Matlab's texture detection is applied

Appendix C

Dataset

To test the hypothesis, several datasets were made in various indoor environments. To create the datasets, the MAV was held by hand, while walking around slowly in the different environments, going from one room or corridor to another. A shot is taken every 2 seconds with both the front and bottom camera, while logging the height measured by the sonar.

C-1 Visibility

To ensure visibility of the floor under all flying conditions, it is proposed to use an MAV with a bottom camera, where the images taken by the bottom camera are a reference for obstacle free space. The MAV used for this research is an Ardrone2, since it has both a frontal and bottom camera. The front and bottom camera have an effective Field Of View (FOV) of 75 deg and 45 deg, respectively. The first collection of datasets is made with the drone held at a height of about 1.5 meters. As can be seen in figure $C-1$, the minimum distance (d1) at

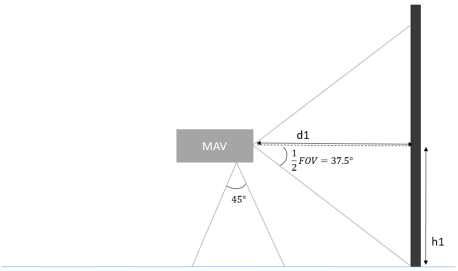

Figure C-1: Illustrtion of MAV with 2 cameras: front and botton. Minimum distance from a wall at which the floor can be perceived with the front camera

which the floor can still be perceived in the front image, is dependent on the height (h1) of the MAV above the ground. Flying at a lower height, thus allows a closer approach to obstacles and overall more visibility of the floor in the front images. With the MAV at a height of 1.5

meters, the minimum distance from the wall to be able to see the floor in the front image is 1.65 meters. In many situations in this first collection of datasets, this distance is sufficient for the drone to navigate around the area, requiring that the area is large enough for the drone to turn. The visibility of the floor in this dataset varies from 85 % to 70%. Although this seems quite high at first, the situations where the floor is not visible are in most case exactly the critical moments of the exploration. Situations where the ground is not visible occur when approaching a wall, or when the front camera of the drone is not near parallel to the corridor. This is as expected, and the basis of what will be used for navigation. However, when the corridors or rooms are narrow and a turn is approached, the drone might stop seeing the floor due to the approaching frontal wall, before it sees the turn. To increase the visibility of the floor in the datasets, a new collection of datasets was made at a lower height of about 0.5 meters. The floor is now visible up to a distance of about 0.65 meters. This increases the visibility to about 95 %. The 5 % where the floor is not visible, are situations where the MAV is closer to a wall than 0.65 meters. The drone is has dimensions of 517 mm x 517 mm, and a wingspan of 731 mm, thus would not be able to turn anymore. The final flying altitude should therefore be increased, to increase the distance from a wall. For the dataset, however, it is more useful to use a lower altitude, to test the proposed methods.

C-2 Difficulties

When analyzing the dataset, 5 challenges for feature detection can be identified, even for the dataset at lower altitude. The first example can be seen in figure $C-2$. It can be seen that the same floor, looks different in the bottom image than in the front image. In this example, the bottom image is lighter than the front image floor and it is less uniform than the floor in the front image.

Figure C-2: Example of bottom camera image with matching front camera image, different color appearance of the floor. Bottom image: floor appears gray, front image: floor appears black

In figure $C-3$, another difficulty can be seen. The floor in the front image changes abrubtly from black or gray to bright green, while the floor under the MAV is still gray. When using the current images and comparing the bottom image of this situation to the front image, the green area will be seen as an obstacle.

Figure C-3: Example of bottom camera image with matching front camera image. Bottom image: gray uniform floor, front image: black and green floor

Dependent on the type of floor and illumination, the shadow of the drone is visible in the bottom camera image, as can be seen in figure $C-4$.

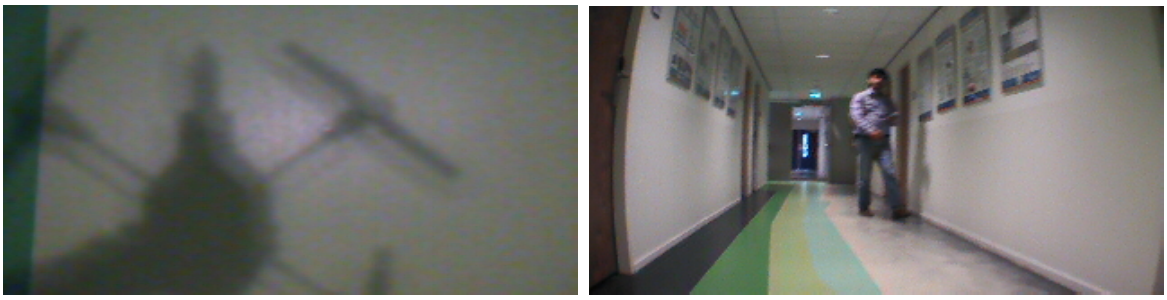

Figure C-4: Example of bottom camera image with matching front camera image. Bottom image: mostly uniform light green floor with shadow of MAV, front image: striped 5-colored floor, without shadows of the MAV

Furthermore, images such as those in figure $C-5$, show that in the bottom image, more texture can be seen than in the front image. This is because the bottom camera is relatively close to the ground. The flowers in the bottom image cannot be recognized by humans in the front image. The resolution of the camera is too low. However, one can reduce what can be distinguish in the bottom image to plain grass, either by filtering or by 'zooming out'.

Figure C-5: Example of bottom camera image with matching front camera image. Bottom camera: dark green grass with purple flowers, textured, front image: light green grass, with some brown spots from sand, no purple flowers

Appendix D

Feature detection

First, the current front and bottom images, as seen by the MAV are used. The features are extracted from the bottom image and compared to the pixels of the matching front images.

D-1 Color based features

In this section, both the Red Green Blue (RGB) and Hue Saturation Value (HSV) color space are used as features for comparison of the front and bottom image.

D-1-1 RGB color space

The RGB color space represents colors that are created by adding red, green and blue light. The first approach taken to compare the front and bottom image, was to compare the red, green and blue values of the bottom camera image, to those of the front camera image. This is done, by decomposing the front and bottom image into the R, G and B channels. The Euclidian distance between the mean of the bottom image channel values and the front image pixels of the matching channel is then determined by equation $D-1$. In equation $D-1$, d is the distance between the n^{th} mean of the entire bottom camera image p and the front camera image q pixel value, per color channel.

$$d(p,q) = \sqrt{(p_n - q_n)^2} \tag{D-1}$$

When the result of equation $D-1$, is smaller than a predefined threshold value, the pixel is classified as non-obstacle. Otherwise, it is classified as obstacle. The segmentation result for a scenario with black floor and white walls can be seen in figure $D-1$, for 2 different threshold values. It can be seen that the performance of this method depends greatly on the threshold values chosen.

Figure D-1: Comparison different thresholds RGB color matching. Left to right: front camera image,thresholds:R<56 & G<64 & B<66, thresholds: R<60 & G<78 & B<87

Figure D-2: Example application same thresholds RGB color matching to different image. Left to right: front camera image,thresholds:R<=60,G<=78,B<=87

The method has difficulties with the brown doors in both images. It also has difficulties with other darker objects in the room, such as the plant. This method is also tested on a different dataset, with a red floor, white walls and blue objects in it. An example of the dataset is shown in figure $D-3$.

Figure D-3: Left to right: input front and matching bottom image from MAV cameras. Obvious color difference in front image floor and bottom image floor

Figure D-4: Comparison different thresholds RGB color matching. Left to right: front image, threshold values: R<40 & G<65 & B<55, thresholdvalues R<32 & G<40 & B<60

Figure $D-4$ demonstrates that for different environments, different threshold values are needed. A comparison with different thresholds is made to indicate the influence of the

thresholds. It can be seen that a piece of white wall matches the red floor, while not all of the red floor is seen as a match. Varying the thresholds, results in either seeing less of the actual floor or in seeing more of the white walls. The main reason for this varying performance is the way RGB colors are composed. This is demonstrated the easiest with an example: what most humans consider as a red (there are many shades) can be represented with an RGB value of [255 0 0] but can also by setting RGB to a value of [215 59 62] or [179 27 27]. When looking at the differences between each channel, it can be seen that it can be [40 59 62] but also [76 27 27] for example. Then when looking at the color green, the same can be done: green can be created with RGB [124 252 0], [119 221 119] and [0 128 0] where the differences are [5 3 119] and [124 124 0], so setting a threshold does not guarantee that the various shades are being recognized without including other colors. Thus, good performance requires finding the right threshold values for each image. To do this, the entire dataset must be known a priori.

D-1-2 HSV color space

The HSV color space composes colors in a different way than the RGB color space does. Instead of summing the amount of red, green and blue light to compose a color, a color is composed by the Hue, Saturation and Value. For color, the most important component is Hue. The color is dependent on the Hue angle, as can be seen in figure $D-5$. This means that red has a hue range from 355° to 10° and green has a hue range from 81° to 140°. The hue ranges do not interfere and ranges can be made smaller to specify the color even more, or larger to allow for more color variation.

Figure D-5: Hue range visualization in degrees

Initially, the same approach is taken as for the RGB color space. Thus, the Euclidian distance between the mean values of the bottom image for each channel and the front image pixels is determined. The result is illustrated in figure $D-6$.

Figure D-6: Left to right: front image, HSV averages threshold segmentation with thresholds H<=0.5 S<=0.1 V<=0.1

The same is done for an environment with black floor and white walls, as was done in the previous section. The results can be seen in figure $D-7$.

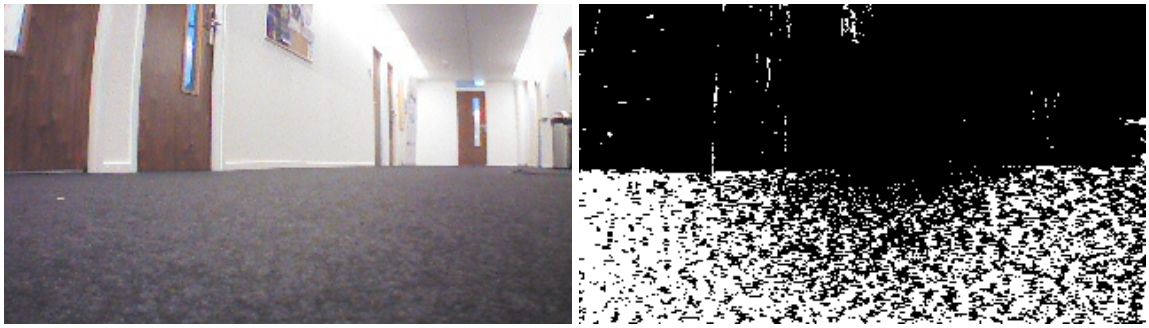

Figure D-7: Left to right: Front image, threshold based average HSV color matching with thesholds values H<=0.4 S<=0.27 V<=0.47

It can be seen in figure $D-6$ that the concentration of obstacle free pixels is at the right location, but that the segmentation can still be improved. When looking at figure $D-7$, the segmentation performance is better. However, the test environment is simpler, since the color difference in the images are large (black and white). It must also be noted that the threshold values are different for the segmentation in figure $D-6$ than in figure $D-7$.

D-1-3 HSV and RGB histogram comparison

Instead of looking at one pixel individually, it is more interesting to include the surrounding pixels as well. By including the surrounding pixels, it can be seen whether a pixel is part of an obstacle free region or that its match is just a coincidence, due to shadow, light or any other reason. To include more information in the comparison, one can compare histograms of images, to determine the similarity or difference between the bottom image and regions of the front image. For this approach, the histograms of the decomposed color channels of the bottom image are first determined. To find matching regions in the front image, histograms are made for each pixel, based on the surrounding pixels.

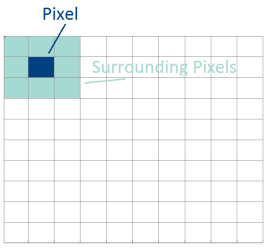

Figure D-8: Information used for pixel histogram clarification

In figure $D-8$, an example of how the histograms for each pixel are determined with window size 3 is illustrated. One can see that the pixel of which the histogram is determined, is the center pixel and it is colored dark blue. The surrounding pixels are light blue and are used to complete the histogram. One can then determine to move the window, thus centred pixel, with a stepsize of 1 or more pixels. The window can be moved until the edge of the window reaches the edge of the image. The reason to use a larger step size is to speed up the calculations. However, this results in a lower accuracy. The size of the window is also a factor in the resulting accuracy. Taking a window too large, the possibility exists that a pixel that should be classified as ground, is still classified as obstacle, because it is near an obstacle or vice versa. Furthermore, a too large window results in the window reaching the edge of the image faster, which reduces the available information.

This histogram approach is first applied to the RGB images, to illustrate the differences with the approach explained in the previous sections. The results for 2 different images and different threshold values are shown in figures $D-9$ and $D-10$.

Figure D-9: Comparison different thresholds RGB color matching. Left to right: front camera image,thresholds:R<=1.97,G<=1.97,B<=1.97, thresholds: R<=2,G<=2,B<=2

Figure D-10: Comparison different thresholds RGB color matching. Left to right: front camera image,thresholds:R<=2,G<=2,B<=2, thresholds: R<=2,G<=2,B<=1.999

The threshold values working for one image, do not give similar performance in other images. An example of using the using the same R, G and B threshold values of 2, as used in figure $D-1$, can be seen in figure $D-2$. The threshold values of 2 for this image results in bad

segmentation performance. Changing the threshold value for B from 2 to 1.999, results in a much better performance. The threshold difference is minor, which implies high sensitivity. However, when applying this difference to the image in figure $D-1$, the segmentation performance is a lot worse than when using the threshold value of 2 for each color channel. This is illustrated in figure $D-11$.

Figure D-11: Comparison different thresholds RGB color matching. Left to right: front camera image,thresholds:R<=2,G<=2,B<=2, thresholds: R<=2,G<=2,B<=1.999

Figure D-12: Comparison different thresholds RGB color matching. Left to right: front image, threshold values: R<= 1.97 G<= 1.97 B<= 1.97, thresholdvalues R<= 1.89 G<= 1.88 B<=1.70

When studying the thresholds used, the threshold value differences for dataset 1 are very small. Adjusting the threshold value with an increase of decrease of 0.001, results in large segmentation performance differences. For dataset 1, the threshold values must be close to 2 to work.

The histogram approach is also applied to the HSV color space images. For comparison, the same images are used as for the RGB color space. The segmentation results can be seen in figures $D-13$ and $D-14$.

Figure D-13: Left to right: front image, HSV histogram segmented image with threshold values H<=1.97 S<=1.90 V<=1.80

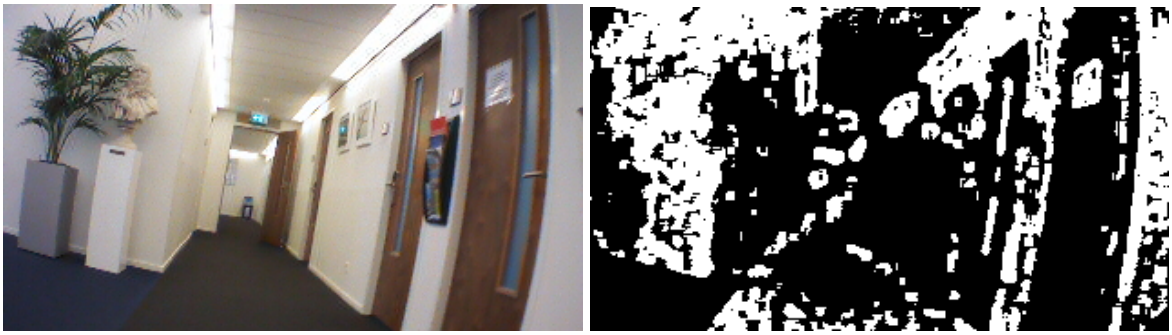

Figure D-14: Left to right: front image, HSV histogram segmented image with threshold values H<=1.3 S<= 1.3 V<=2

The resulting segmentation for the HSV color space can be seen in figure $D-15$, where pink indicates obstacle.

One can see that for a for a simple environment the method can produce quite good segmentation. The normalized error for each image in the dataset can be seen in figure $D-16$.

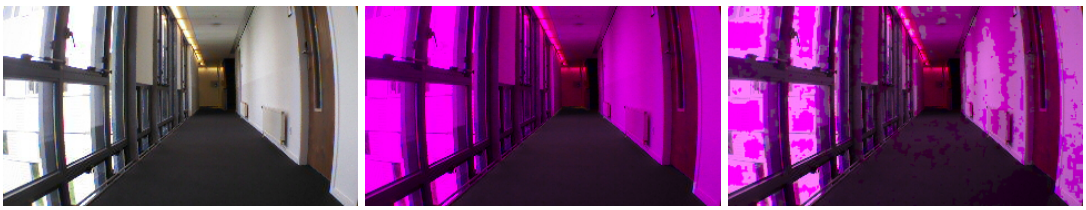

Figure D-15: Left to right: Front image, Validation segmentation, HSV segmentation. Pink indicates obstacle

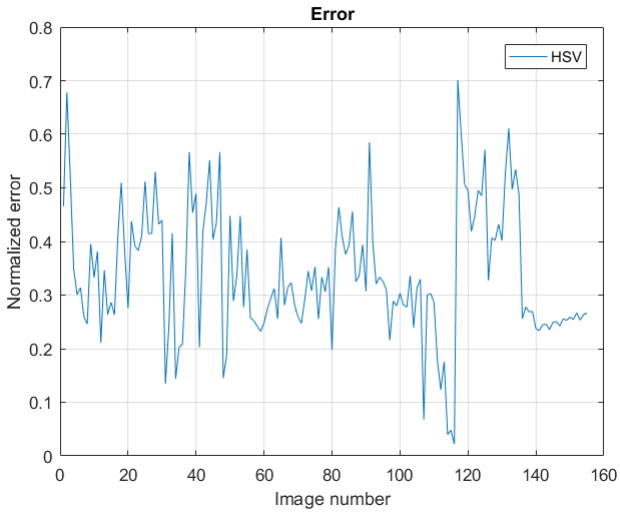

Figure D-16: Normalized error between HSV histogram segmentation and hand-labelled classification, for dataset 1

D-2 Texture based features

Another type of features that can be used are features based on the texture in an image. One aspect that needs to be considered, is the change due to the perspective transformation that occurs from the bottom to the front camera. This transforms the texture appearance: straight lines as seen by the bottom camera, appear to approach each other in the vanishing point in the front camera image. Albeit not much texture is present in most images, it is expected that it is still a useful addition to the color based features.

D-2-1 Edges

One way of quantifying texture in an image region, is by examining the edges in that region. By looking at the amount of edges in a certain region, information on the concentration of texture can be obtained and one can distinguish regions with a large concentration of edges from regions with few or no edges. To detect edges in an image, a gradient-based edge detector is used. The amount of edges is determined with equation $D-2$, where p is pixel, G the gradient magnitude and N the size of the image region used for analysis.

$$F_{edgeness} = \left| \{p|G(p) \geqslant threshold\} \right| /N \tag{D-2}$$

For this approach, a moving window, such as for the color histograms is used, to determine the amount of edges in the various regions of the front image. To compare to this bottom image, the Euclidean distance between the region edgeness and the bottom edgeness is determined. Distances below a threshold value, are considered as obstacle free regions. The threshold value used in equation $D-2$, is the mean of the gradient magnitudes of the bottom image. The resulting differences with the front image are then subjected to a threshold value of 0.3, meaning that if the error is smaller than or equal to 0.3, the pixel is considered as ground and thus obstacle free.

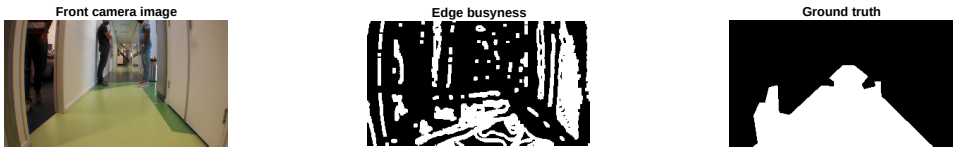

Figure D-17: Example of good segmentation result by using the edgeness feature. Left to right: input image, resulting segmentation, ground truth

In figure $D-17$, an example result of using the edgeness feature for matching regions in the front image with the bottom image can be seen, together with the actual ground truth. It can be seen that for this image, most of what is the actual ground, is also indicated as ground by the edgeness feature. However, it can be seen that the plain wall is also considered as floor, due to the equal lack of edges. This insinuates that adding the color features to texture would help distinguish the wall from the floor. Yet, when looking at the results for the front image in figure $D-18$, it can be seen that instead of the floor and the wall, a door is now selected as the best match with the bottom image edge properties.

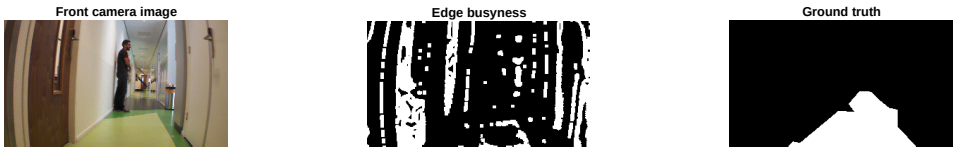

Figure D-18: Example of a bad segmentation when using the edgeness feature. Left to right: input image, resulting segmentation, ground truth

For the majority of the images, the results are such as in figure $D-17$. This however, requires sufficient edges to be present in the front image and that the bottom image ground matches the front image ground in texture appearance. Figure $D-19$ shows the error between the ground as indicated by the edgeness feature and the ground truth. The mean error is 0.3045. The largest error is found for image 117, which is presented in figure $D-20$. The error is determined by the total sum of the error resulting from applying equation $D-1$ to the difference of the front image pixel histogram and the bottom image overal histogram values. It can be seen that the front image shows a wall, because the MAV is too close to the wall. The floor cannot be seen in the front image, but similarities to the floor can be found, due to smooth surface of the floor and the walls. A certain error will always be present, in non perfect scenarios. The error will decrease when decreasing the threshold value, but this does not necessarily give better results. It is considered more desirable to have a greater error to find regions that are a potential match and then validate the match with an additional feature.

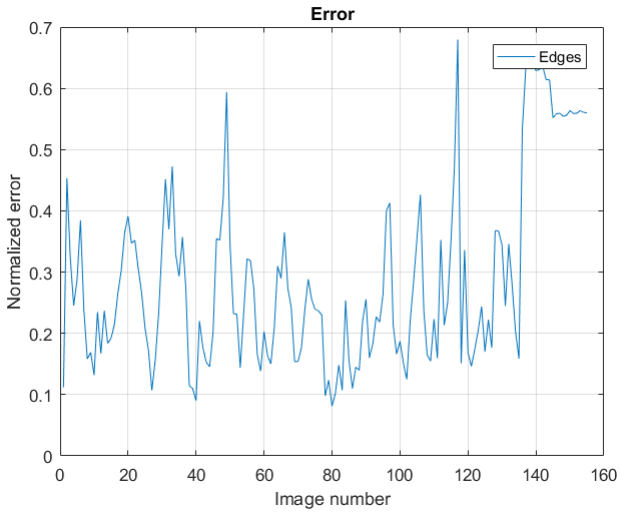

Figure D-19: Error edgeness with ground truth dataset black floor, white walls

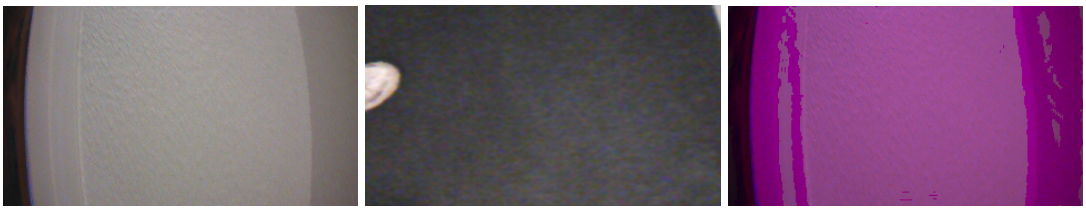

Figure D-20: Left to right: Front image of image 117, Bottom image of image 117, Resulting segmentation, pink represents obstacle

D-2-2 Law's masks

A different approach in identifying texture in images, is by using Laws' energy measures. These are a set of energy transforms, where texture energy is presented as the amount of variation in a fixed size window. The set of texture energy transforms describes the amount of texture in each pixel. The texture energy measures depend on the spatial filter, the size of the window used and on the technique used for measuring the average window variation. They can be made invariant to illumination, contrast and rotation, but are not invariant to change in scale. The images to be used for this technique are monochrome images. The texture variation in an image can be determined by the following procedure:

An RGB-input image needs to be converted to gray-scale and filtered with a limited number of convolution 1D masks, as presented in equation $D-2-2$. The filtered images resulting from this step are then filtered by determining the outer products of all combinations of equation $D-2-2$, which results in 16 2D masks. Combining these texture energy measures reduces the number of measures, while improving the usefulness of the measures. This can be done by normalization, which can be achieved by extracting the principle components to reduce the set of features in the classifier and results in illumination and contrast invariance. The measures can be made rotationally invariant by averaging matched pairs of texture energy

$$\begin{array}{rll} & L5: & [1 \quad 4 \quad 6 \quad 4 \quad 1] \\ & E5: & [-1 \quad -2 \quad 0 \quad 2 \quad 1] \\ \text{measures. (Laws)} & S5: & [-1 \quad 0 \quad 2 \quad 0 \quad -1] \\ & R5: & [1 \quad -4 \quad 6 \quad -4 \quad 1] \end{array}$$

In equation $D-2-2$, L5 is the Level mask and represents the center-weighted local average, E5 is the Edge mask and detects edges, S5 is the Spot mask and detects spots and R5 is the Ripple mask that detects ripples (Shapiro & Stockman, 2001). The texture energy map for a filtered image window can be calculated with equation $D-3$, where F_{vh} is the filtered row

$$E_{vh}(r,c) = \sum_{j=c-7}^{c+7} \sum_{i=r-7}^{r+7} |F_{vh}(i,j)| \tag{D-3}$$

The resulting Laws masks can be seen in figure $D-21$.

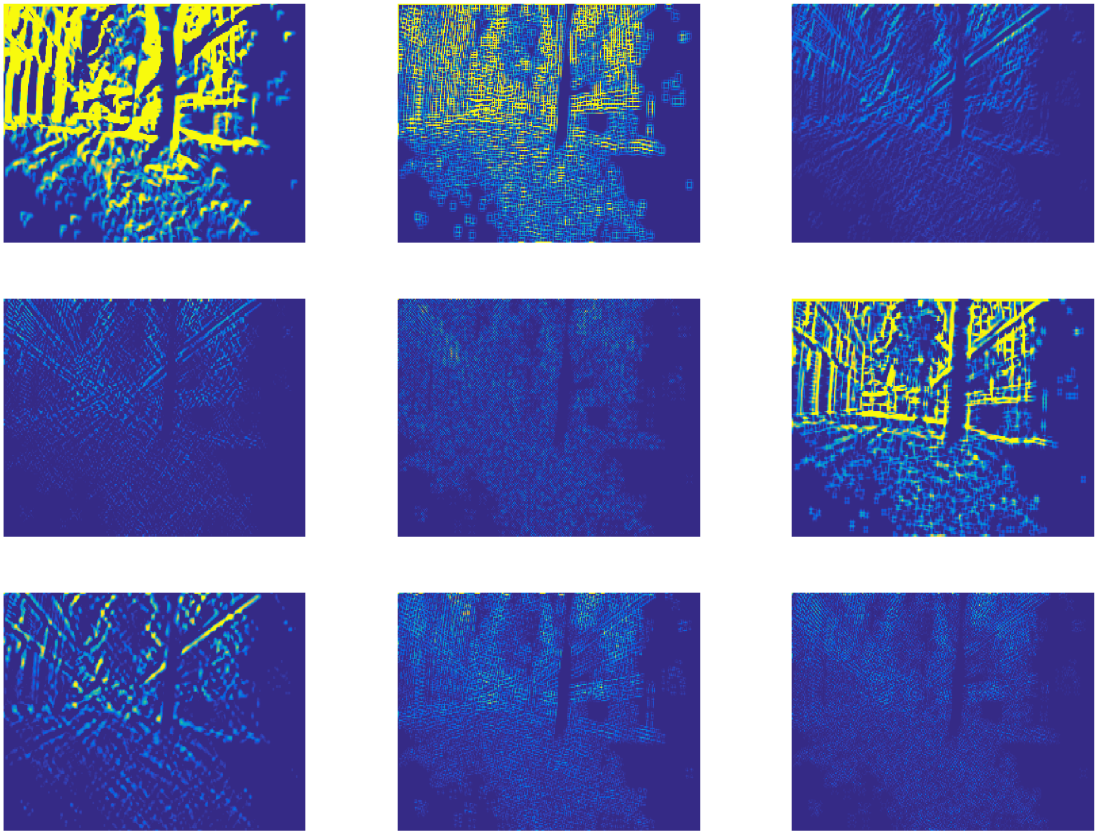

Figure D-21: The 9 resulting 2D laws measures

These masks are applied to both the front and bottom images. The pixel difference is then determined with equation $D-1$ for each pixel, with the difference of the front image pixel histogram and the bottom image histogram. The overall sum of this difference of the class of all front image pixels and the front image hand-labelled class of all pixels is then used to be subjected to a threshold value. For each mask, a different threshold value is used, determined on the error in dataset 1. The threshold values can be found in table $D-1$.

Mask 1	0.7748
Mask 2	0.4
Mask 3	0
Mask 4	0
Mask 5	0.2653
Mask 6	0.3
Mask 7	0.0606
Mask 8	0.5
Mask 9	0.5

Table D-1: Laws' masks and their respective treshold values

Initially, all 9 Laws' masks are applied to all images. The resulting error for dataset 1, for each Laws' mask is presented in figure $D-22$. The error is determined by the difference of the Law's mask segmentation and the ground truth segmentation.

When looking at figure $D-22$, it can be seen that the error for the different Laws masks is similar for all masks. Some peaks can be seen for the 3rd, 4th and 9th mask. Since generating all Laws masks for all pixels with their surrounding window is a time consuming process, it is decided to only use the 2nd Laws' mask as texture feature information and will be used to represent all Laws' masks.

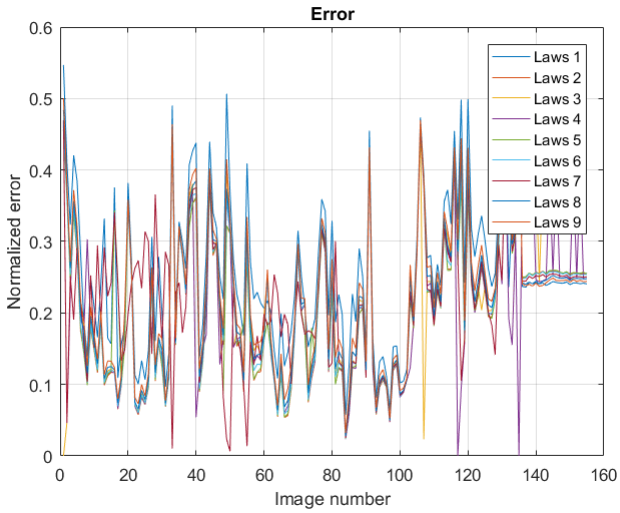

Figure D-22: The normalized error between the segmentation for each laws mask and the hand-labelled classification

An example of the resulting segmentation by applying Laws' mask to dataset 1 can be seen in figure $D-23$.

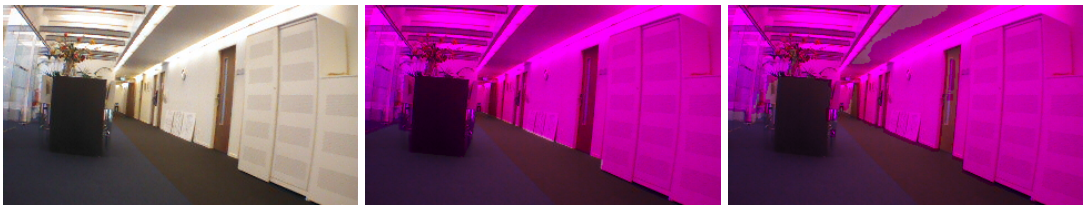

Figure D-23: Left to right: Front image, segmentation by hand-labelled image, segmentation by Laws' mask, pink represents obstacle

Figure D-24: Left to right: Front image, segmentation by hand-labelled image, segmentation by Laws' mask, pink represents obstacle

In figure $D-24$, the Laws' mask, with threshold determined on dataset 1, is tested on dataset 2. It can be seen that it has difficulties with the variations in color in the floor and the edges present in the floor and only classifies the dark green edges as floor, while classifying the lighter green parts of the floor as obstacle.

Appendix E

Feature extraction from a set of images

In this chapter, it is investigated whether the appearance of the floor can be learned from a large training set of about 100 images for each camera. This is done for features from both the front image only and the bottom image. The class of each pixel of the trainings data needs to be known. Therefore, the ground is hand-labelled for each front image, as can be seen in figure $E-1$.

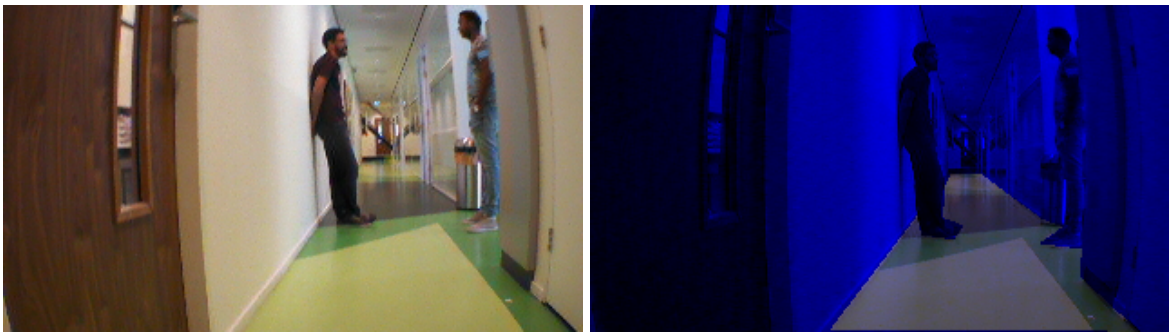

Figure E-1: Left to right: Front image, example labelled ground, blue indicates obstacle

E-1 Ground training with HSV features

As done for the current images, the first features extracted from the images are color features. To set an initial benchmark on how well the floor can be classified from the rest of the scene, classifiers are trained with features taken from the front image only. The first collection of features is created by collecting histogram information of the H, S and V channels of each pixel of the front image, within their surrounding window, including the mean, standard deviation and entropy of the H, S and V histograms for that pixel. For each pixel the class is indicated

with the ground truth value of the pixel, as taken from the hand-labelled images. A REP tree is trained with the maximum tree depth set to 6. The resulting tree classifies 91.1% of the pixels correctly out of a set of 174375 datapoints. To see whether an even greater decrease in the size of the tree still produces acceptable results, the maximum depth of the tree is first set to 4 and then to 2. As a result, the correctly classified percentage of the pixels are then 88.3 % and 84.4%.

Table E-1: Accuracy by class for tree depth 6

True Positive (TP) Rate	False Positive (FP) Rate	Class
0.789	0.056	Ground
0.944	0.211	Obstacle

Table E-2: Accuracy by class for tree depth 4

TP Rate	FP Rate	Class
0.553	0.029	Ground
0.971	0.447	Obstacle

Table E-3: Accuracy by class for tree depth 2

TP Rate	FP Rate	Class
0.386	0.034	Ground
0.966	0.614	Obstacle

Using this feature set, other classifiers were tested. The most promising were the Naive Bayes and the Lazy IBK. A Naive Bayes classifier uses the assumption that the value of each feature is independent of the value of each other feature (Rish, 2016). The Lazy IBK is a K-nearest neighbours classifier that uses cross validation to select the suitable K value (Weka information). The Naive Bayes classifies 79.4% of the pixels correctly and Lazy IBK classifies 93.2% of the pixels correctly. The TP Rate and FP Rate for ground and obstacle can be found in tables $E-4$ and $E-5$.

Table E-4: Accuracy by class for Naive Bayes

TP Rate	FP Rate	Class
0.945	0.247	Ground
0.753	0.055	Obstacle

Table E-5: Accuracy by class for Lazy IBK

TP Rate	FP Rate	Class
0.857	0.041	Ground
0.959	0.143	Obstacle

However, for convenience and ease of implementation, the REP tree is used. The accuracy of the REP tree increases with increasing tree depth. At a certain point, here at a depth of about 10 the accuracy stagnates. To see whether adding ground features to the front features

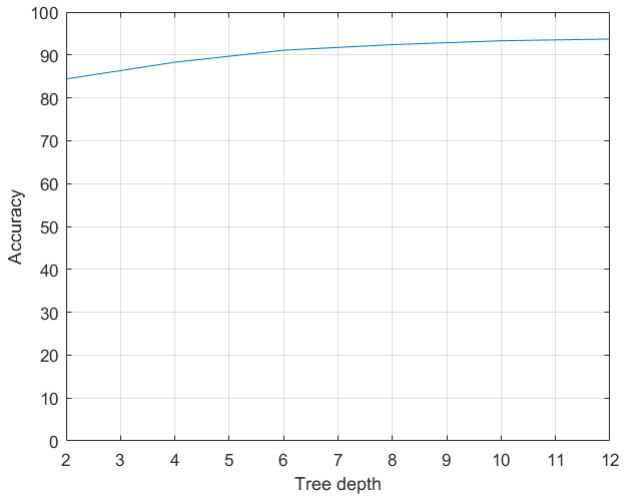

Figure E-2: Tree depth vs accuracy REP tree

helps in recognizing the ground, the difference of the front and bottom histograms for each color channel are added to the feature matrix. For the ease of implementation, a REP Tree with a maximum depth of 2, is trained to classify pixels as belonging to the ground or to an obstacle. In table $E-6$, the resulting properties of the tree can be found. The addition of the

Table E-6: Accuracy by class for tree of depth 2

TP Rate	FP Rate	Class
0.771	0.050	Ground
0.950	0.229	Obstacle

difference with the bottom image properties to the feature set, results in a tree that classifies 91.2 % of the pixels correctly, whereas when only the features of the front image were used, a tree with depth 2, can classify 84.4% of the pixels correctly.

The accuracy of the REP tree for using HSV features, with and without bottom information, stagnates at a tree depth of about 10. For both the stagnated accuracy of the HSV features without bottom information and the HSV features with bottom information, the REP tree is trained, the results can be found in table $E-7$ and in table $E-8$, respectively.

Table E-7: Accuracy by class HSV features, without bottom, by class for tree of depth 10, 93.3% accurate, trained on dataset 1, tested on dataset 1

TP Rate	FP Rate	Class
0.831	0.040	Ground
0.960	0.169	Obstacle

It can be seen that the TP Rate increases for both the ground and obstacle class, when bottom information is added. An example of the resulting segmentation for both the HSV features without bottom and the HSV features with bottom can be found in figure $E-3$.

Table E-8: Accuracy by class HSV, with bottom, by class for tree of depth 10, 97.5% accurate, trained on dataset 1, tested on dataset 1

TP Rate	FP Rate	Class
0.932	0.014	Ground
0.986	0.068	Obstacle

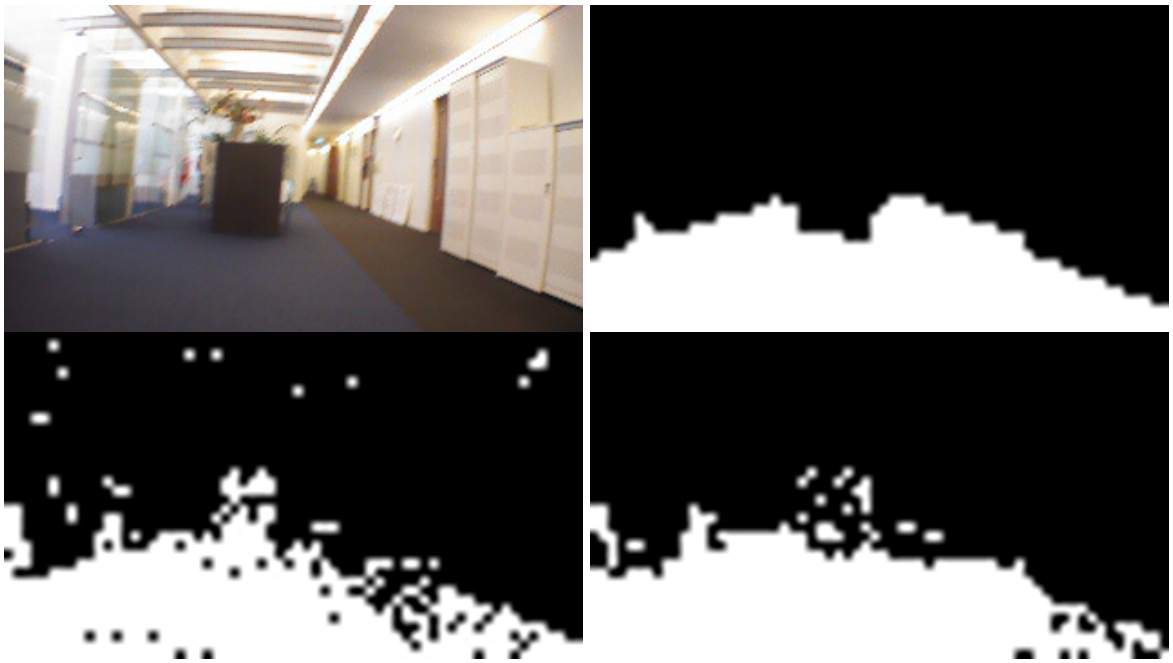

Figure E-3: Top row: input image and labelled ground, bottom row: segmentation by HSV without bottom and with bottom included, trained on dataset 1, tested on dataset 1

Due to the ease of implementation and good results, the REP tree is the only classification learner used in the following sections.

E-2 Ground training with HSV and edges features

As indicated before, edges are a promising feature when it is added to other features, such as color. Therefore, a feature matrix is created with the HSV properties as described in the previous section, complemented with the edgeness of the front image. To observe the effect of the combination of HSV features and edge properties, two decision trees are trained. One that contains information on the front images only and one that includes the difference with the bottom features. The features used for this tree, are those as used for the HSV features only, complemented with the edgeness of the front image. When the bottom is added, the difference with the bottom is added too. The results of tree based on the front properties can be seen in table $E-9$. The result of the front properties augmented with the bottom property difference, can be seen in table $E-10$. The resulting segmentation can be seen

Table E-9: Accuracy by class HSV with Edges by class for tree of depth 10, 91.3% accurate

TP Rate	FP Rate	Class
0.777	0.048	Ground
0.952	0.223	Obstacle

Table E-10: Accuracy by class HSV with Edges, with bottom, by class for tree of depth 10, 96.2% accurate

TP Rate	FP Rate	Class
0.900	0.020	Ground
0.980	0.100	Obstacle

in figure $E-4$. It can be seen that the inclusion of the bottom information increases the segmentation performance.

It can also be seen that the inclusion of edgeness in the set of features, does not significantly improve the segmentation performance. Which implies that the addition of edges, is not necessary to obtain proper segmentation performance.

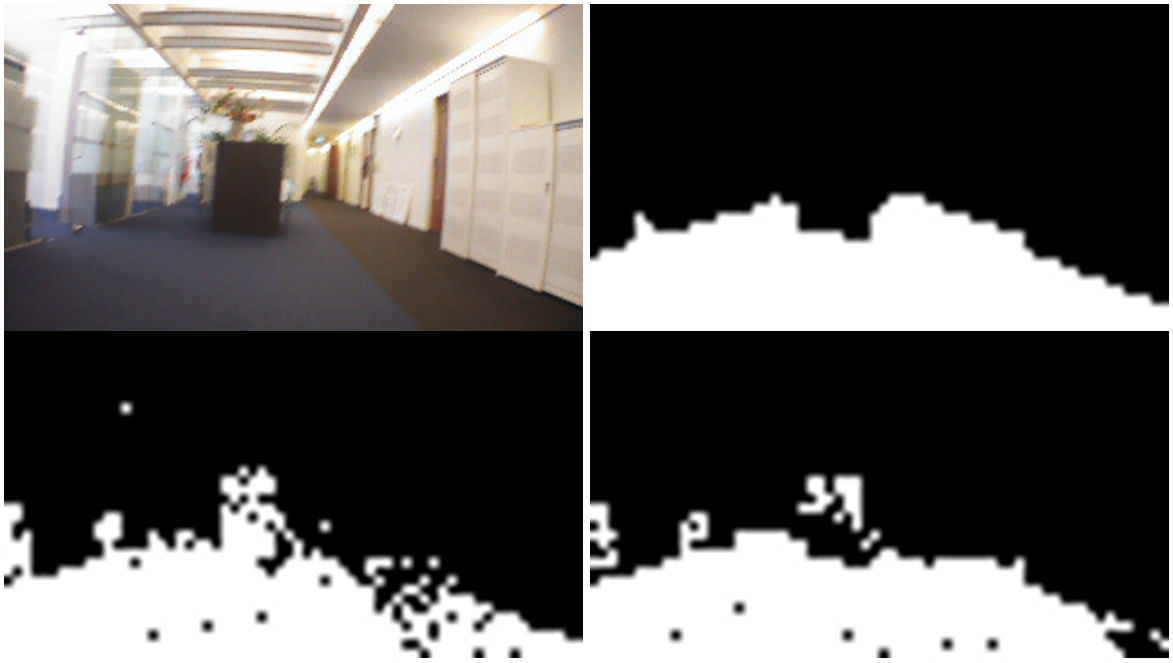

Figure E-4: Top row: input image and labelled ground, bottom row: segmentation by HSV and Edges without bottom and with bottom included, trained on dataset 1, tested on dataset 1

E-3 Ground training with HSV and Laws' features

For the combination of HSV properties and Laws' features, a tree is trained with the front images properties only, followed by a tree trained with both front and bottom properties. The front images properties are the HSV properties as described above and the histogram of Law's energy measures. The difference with the bottom includes the histogram differences for the HSV features and Law's energy measures. The results for the TP Rate and FP Rate can be found in table $E-11$.

Table E-11: Accuracy by class HSV with Laws, without bottom, by class for tree of depth 10, trained on dataset 1, tested on dataset 1

TP Rate	FP Rate	Class
0.814	0.053	Ground
0.947	0.186	Obstacle

For comparison, the differences with the bottom images properties of HSV and Laws are added. The results, as can be seen in table $E-12$, indicate that the difference with the bottom improves the classification performance.

The resulting image segmentation can be seen in figure $E-5$. Here, once again, it can be seen that the inclusion of the difference with the bottom reduces the segmentation performance, even though the sensitivity analysis promises better results. Recurrently, it can be seen that the segmentation resulting from the front images produces similar results to the other methods and is comparable to the hand-labelled ground. The inclusion of the bottom differences improves the segmentation performance.

Table E-12: Accuracy by class HSV with Laws, with bottom, by class for tree of depth 15 , 97.1% accurate, trained on dataset 1, tested on dataset 1

TP Rate	FP Rate	Class
0.927	0.017	Ground
0.983	0.073	Obstacle

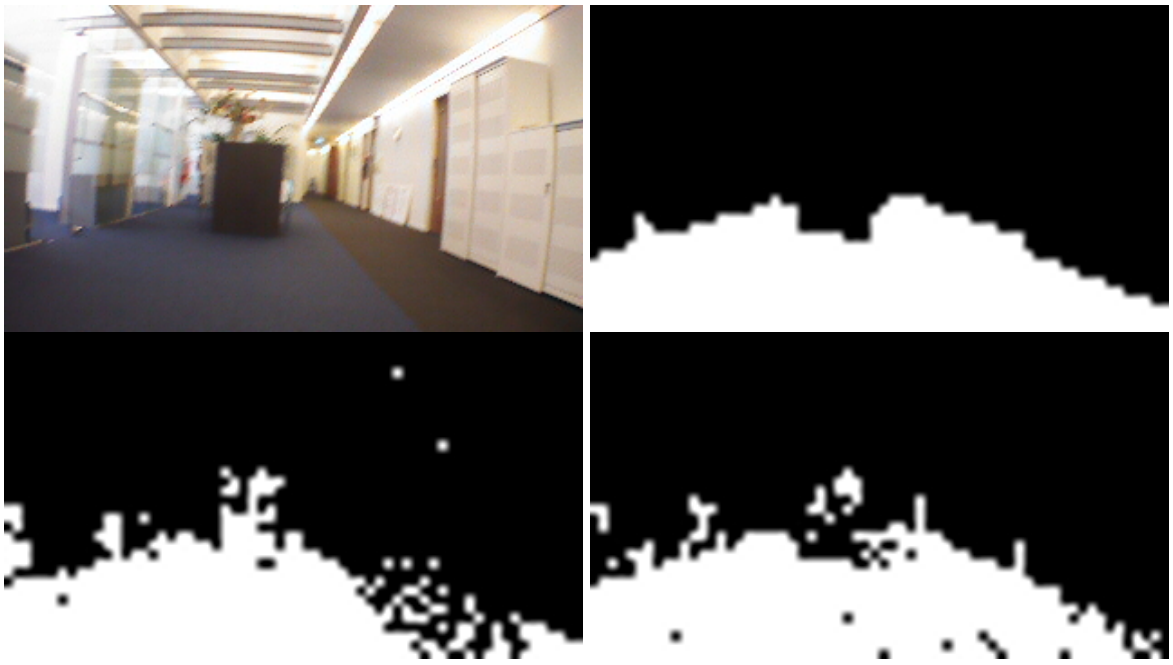

Figure E-5: Top row: input image and labelled ground, bottom row: segmentation without bottom and with bottom included

Overall, it can be seen that the combination of features does not necessarily improve the result, which means one method can be used.

E-4 Ground recognition with bottom image information based on outlier detection

To see whether the bottom camera image alone contains sufficient information to be used to classify the front image pixels correctly, bottom image features are used for outlier detection in the front camera image. The outliers are detected by determining the quartiles and the inter-quartile range of the 3 last seen bottom images. A value is considered an outlier if it complies with the statements in equation $E-1$, where Q1 and Q3 are the first and third quartile and IQR is the inter-quartile range.

$$Pixel<Q1-1.5*IQRPixel>Q3+1.5*IQR \tag{E-1}$$

Since the histogram values, mean, standard deviation and entropy are used as features, each of these values can be an outlier with respect to the values of the bottom image. Having

one outlier in a set of 39 features, does not necessarily mean that the pixel is an outlier with respect to the other pixels. The first method sums up the number of outliers found in this set of outliers. If the remaining sum is greater than a threshold value, the pixel is considered an outlier.

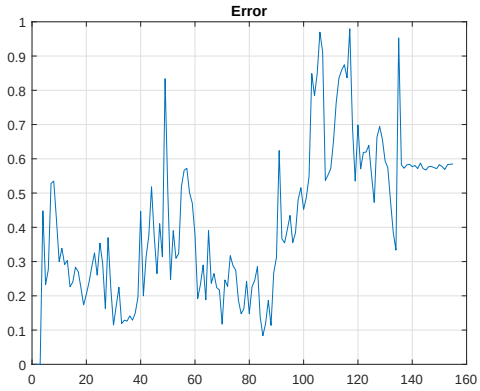

Figure E-6: Normalized error between estimated ground by HSV feature outliers and actual ground, thresholds based on dataset 1, tested on dataset 1

As can be seen in figure $E-6$, the errors between the estimated ground and actual ground are significantly large. Although for some images this method of determining whether a pixel is an outlier or is part of the ground seems to work well, for most images, the error is too large to be useful to determine an obstacle free route. Changing the threshold value does not improve the quality of this method. Therefore, a REP tree is trained to determine which outliers are more important and use these as thresholds.

As discussed in the literature review, Ulrich uses a similar approach as the approach proposed in this thesis. The method as described by Ulrich in (Ulrich & Nourbakhsh, 2000), is implemented and used as a benchmark for this research. The method uses a trapezoid area in front of the robot as a reference for potential ground. The robot in (Ulrich & Nourbakhsh, 2000), learns the appearance of the ground while driving over it. If the robot has driven over the trapezoid area, the area was obstacle free. It then classifies each pixel, based on its HSI color appearance, as either ground or an obstacle, by histogram comparison. This method is implemented as follows: The trapezoid area RGB histograms of 3 front camera images are first filtered and converted to HSI color space. The resulting H and I histograms are then compared with the H and I histograms of the pixels in the following front image. If the sum of the differences for H is larger than 0.19 and for I is larger than 0.19, the pixel is considered as obstacle.

In figure $E-7$, the resulting segmentation of Ulrich's method is illustrated. The segmentation result is due to the previous images as seen by the MAV. The previous 3 images had green color variation in the floor. This results in worse performance. When the floor has been constant for a number of shots, the segmentation performance increases. An example of the segmentation resulting from Ulrich's method on a different dataset, is illustrated in figure $E-8$. However, this part of the dataset is similar to dataset 1. When the appearance of the floor varies more, the segmentation performance decreases. The error between the estimated segmentation and actual segmentation for every image can be seen in figure $E-9$. It can be

Figure E-7: Example segmentation on dataset 1. Left to right: input image, Ulrich thresholds based on dataset 1, ground truth

seen that the segmentation is not perfect, but sufficient to have a robot move around without hitting an obstacle. The method has difficulties segmenting images where the floor consists of multiple colors.

Figure E-8: Example segmentation by Ulrich's method on dataset 2. Left to right: front image, Ulrich segmented image, ground truth

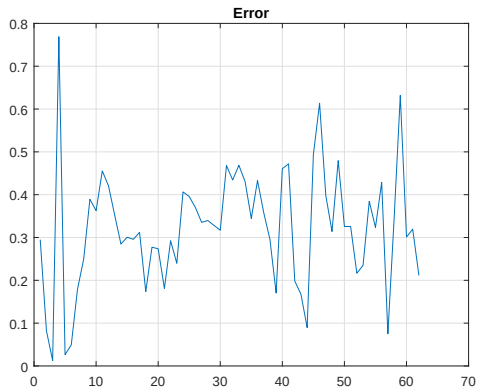

Figure E-9: Error between estimated ground and actual ground, segmentation method by Ulrich, tested on dataset 2

Another aspect that is important in this method is the size of the trapezoid used. Different sizes have been used and tested. A larger trapezoid decreases the segmentation performance due to too many variation in the pixels in the trapezoid. This is due to difference of using an MAV with respect to a ground based vehicle. When using an MAV, the MAV is always at a certain height above the ground. This means that what is seen as ground in front of the MAV, is further away from the MAV than from a ground based vehicle at that same location. This is demonstrated in figure $C-1$. Therefore, Ulrich's method is less suitable for MAVs in combination with appearance varying floors.

The outlier method as discussed above, is tested on the HSV, RGB, Edges and Laws' features.

The resulting normalized error is determined by the sum of the difference of the outlier detection classification of the front image, with the hand-labelled classification. It can be seen in figure $E-10$.

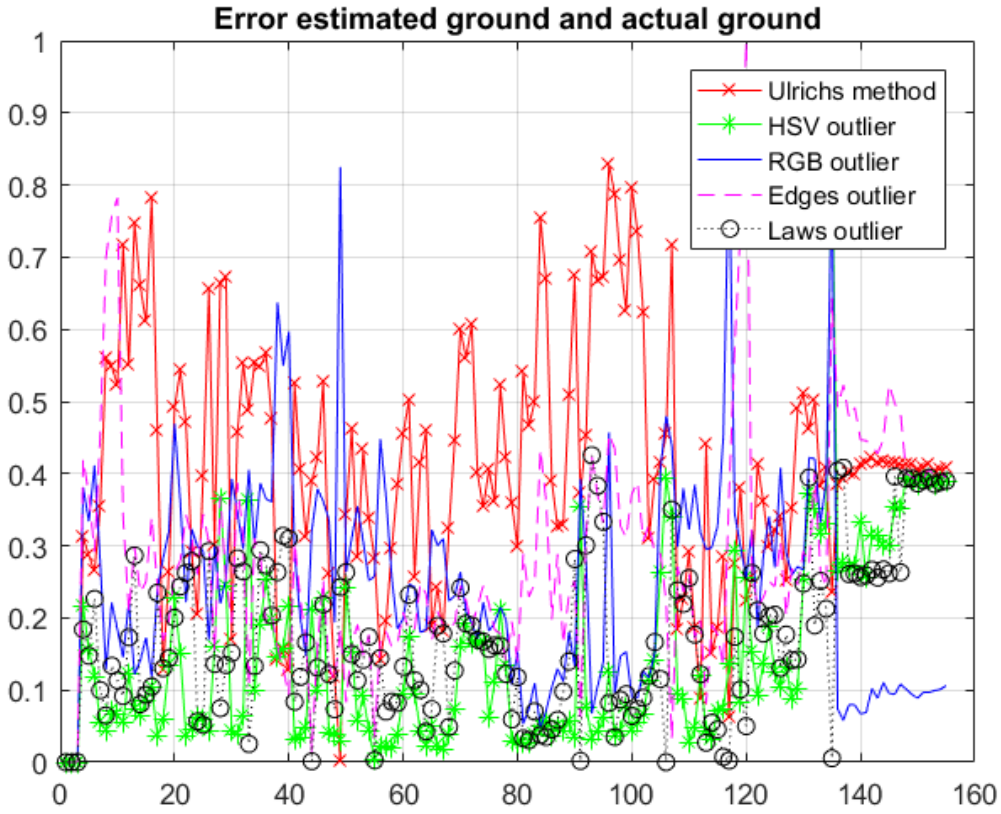

Figure E-10: Normalized error between estimated class and actual class for each image, for all features, tested on dataset 1, threshold determined with dataset 1

It can be seen that the error for the edges features is very large for the first couple images. This is because in some of the bottom images, edges are present in the floor that are not as explicit in the front images. What also can be seen is that the error resulting from Ulrich's method is larger than the other methods. This can be explained in two ways: First, the thresholds used for the histogram differences need to be adjusted and due to the variation in the bottom area of the front image. In one image, the floor it self varies but in other images, pieces of the wall are present in the trapezoid, which results in large histogram differences. The other explanation is that the current threshold setting is an 'or' statement. This classifies pieces of the ceiling as floor, due to the resemblance in intensity. It can also be seen from figure $E-10$, that the RGB outlier method gives a higher overall error than both the HSV and Laws method.

An example of the resulting segmentation, by using the thresholds determined based on dataset 1, tested on dataset 1, for all methods can be seen in figure $E-11$.

The same method with the same threshold values is applied to a different dataset with more

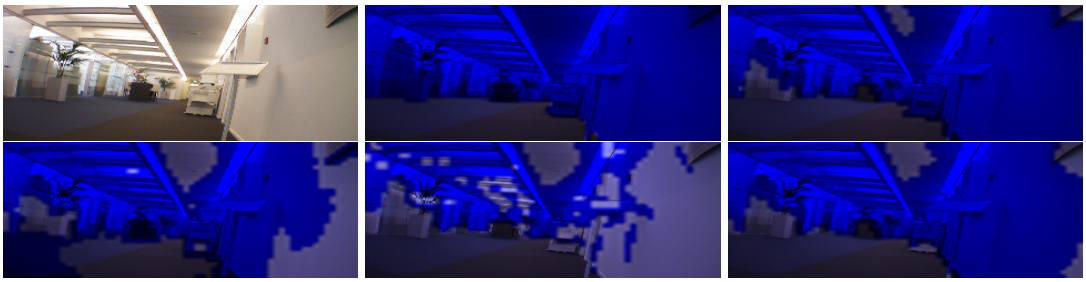

Figure E-11: Example segmentation, all pixels classified as obstacle are made blue. Left to right top row: input image, hand-labelled image, HSV. Left to right bottom row: RGB, Edges, Laws

color variation and obstacles. It is notable that the error is larger for each method, than when applied to the previous dataset. The edge difference deviates more from the rest due to its large peaks. These are caused again by edges in the bottom images that are not as explicitly present in the front image. The HSV outliers show the best results. The largest errors are caused by a large difference in appearance of the floor in the bottom and front image. Ulrich's method shows a relative large error. It has difficulties with the large variation in the trapezoid area, that results from the large variation in floor and the frequent occurrence of obstacles in this area. Adapting the used threshold values can further improve the results, but requires finding the optimal threshold values.

E-4-1 Navigation possibilities

The navigation direction to be followed by the MAV, is determined for each outlier detection method. For comparison, the navigation direction resulting from Ulrich's method is added. In figure $E-13$ an example of the resulting navigation path and included navigation direction can be seen for all outlier methods and Urlich's method, for dataset 1. The results for dataset 2 can be seen in figure $E-14$.

The percentage of correct navigation instructions for both datasets can be found in table $E-13$. One can see that for the HSV method, 64 % of the navigation instruction are given

Table E-13: Correctly estimated navigation directions for dataset 1 and dataset 2, thresholds determined with dataset 1

	Ulrich	HSV	RGB	Edges	Laws
Dataset 1	0.41	0.64	0.45	0.11	0.36
Dataset 2	0.19	0.34	0.36	0.40	0.14
No crash (dataset 1)	0.89	0.91	0.91	0.91	0.93

correctly. The navigation instructions, however, result from 1 image only. It is expected that, when a sequence of images are used, the navigation instruction will be more accurate. Furthermore, one false navigation instruction not necessarily leads to collision. This can be investigated by implementation on the MAV.

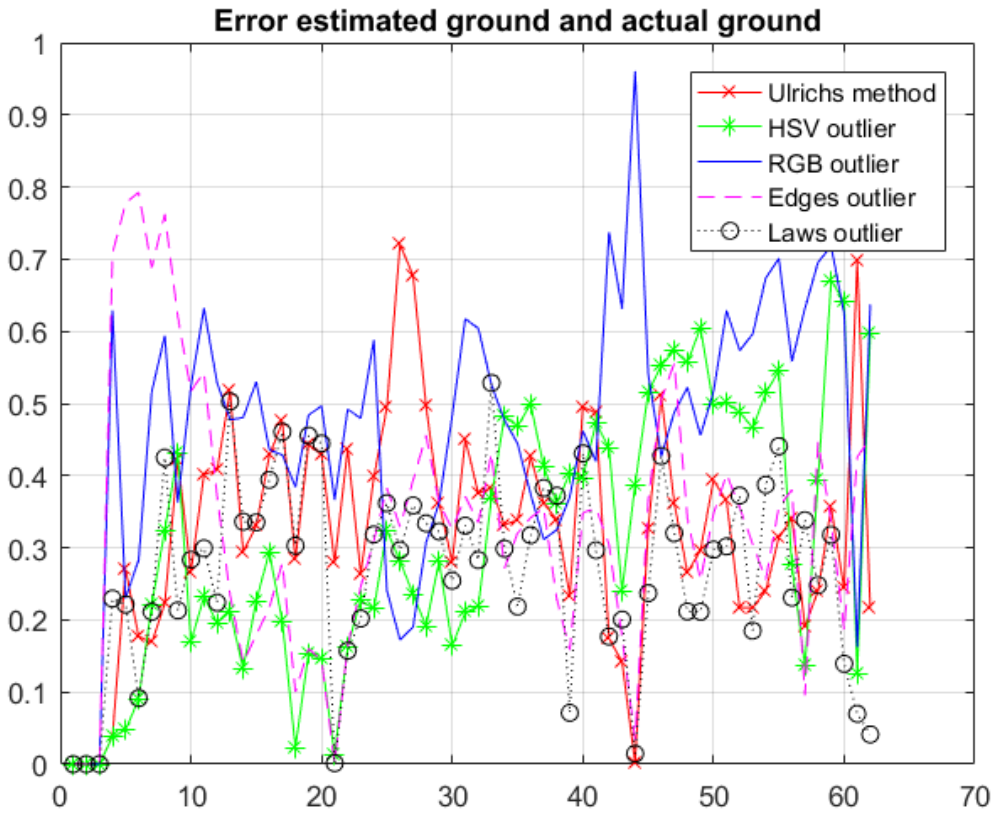

Figure E-12: Normalized error, for all methods, between estimated class and actual class, thresholds determined on dataset 1, tested on dataset 2

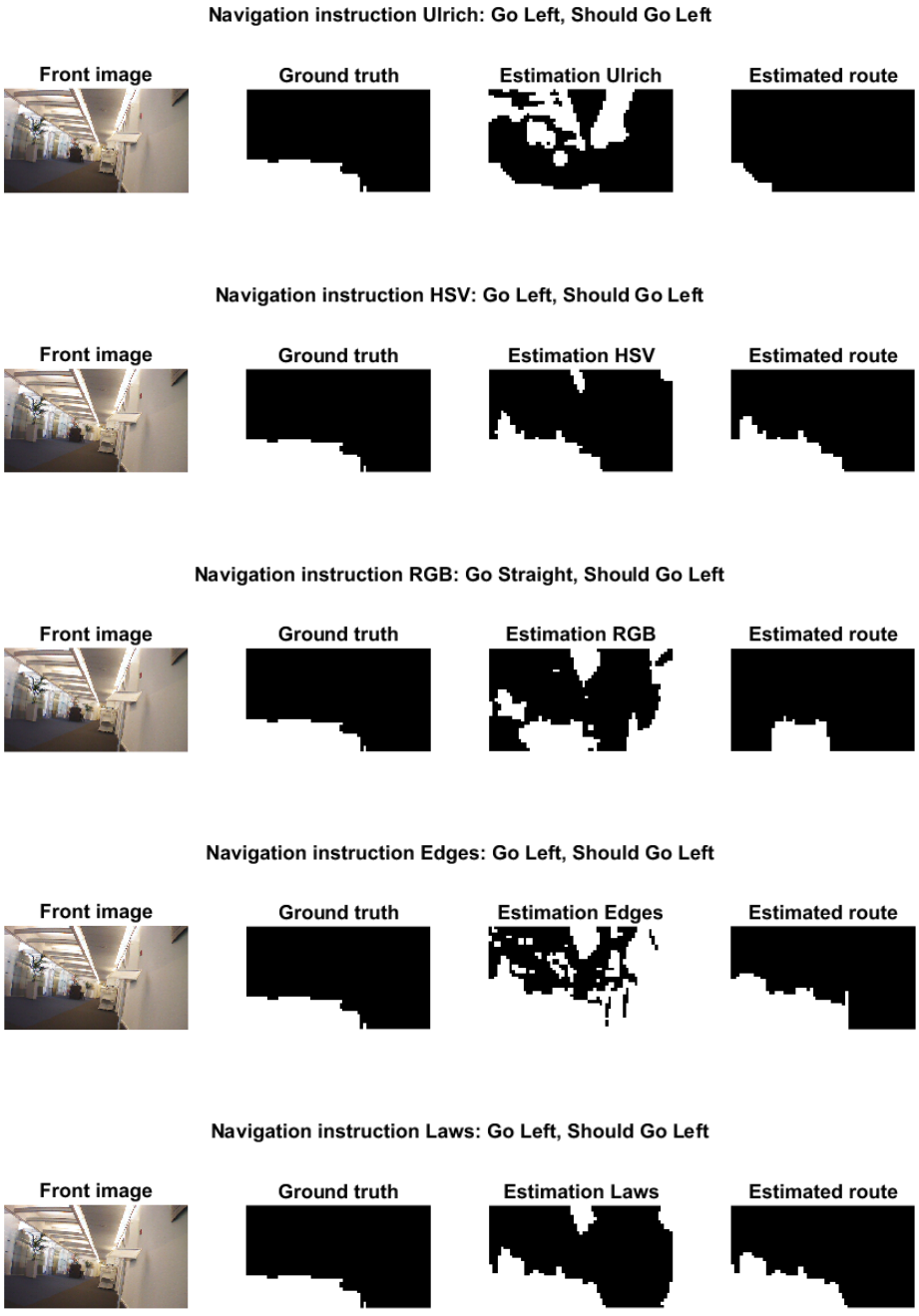

Figure E-13: Example navigation results dataset 1

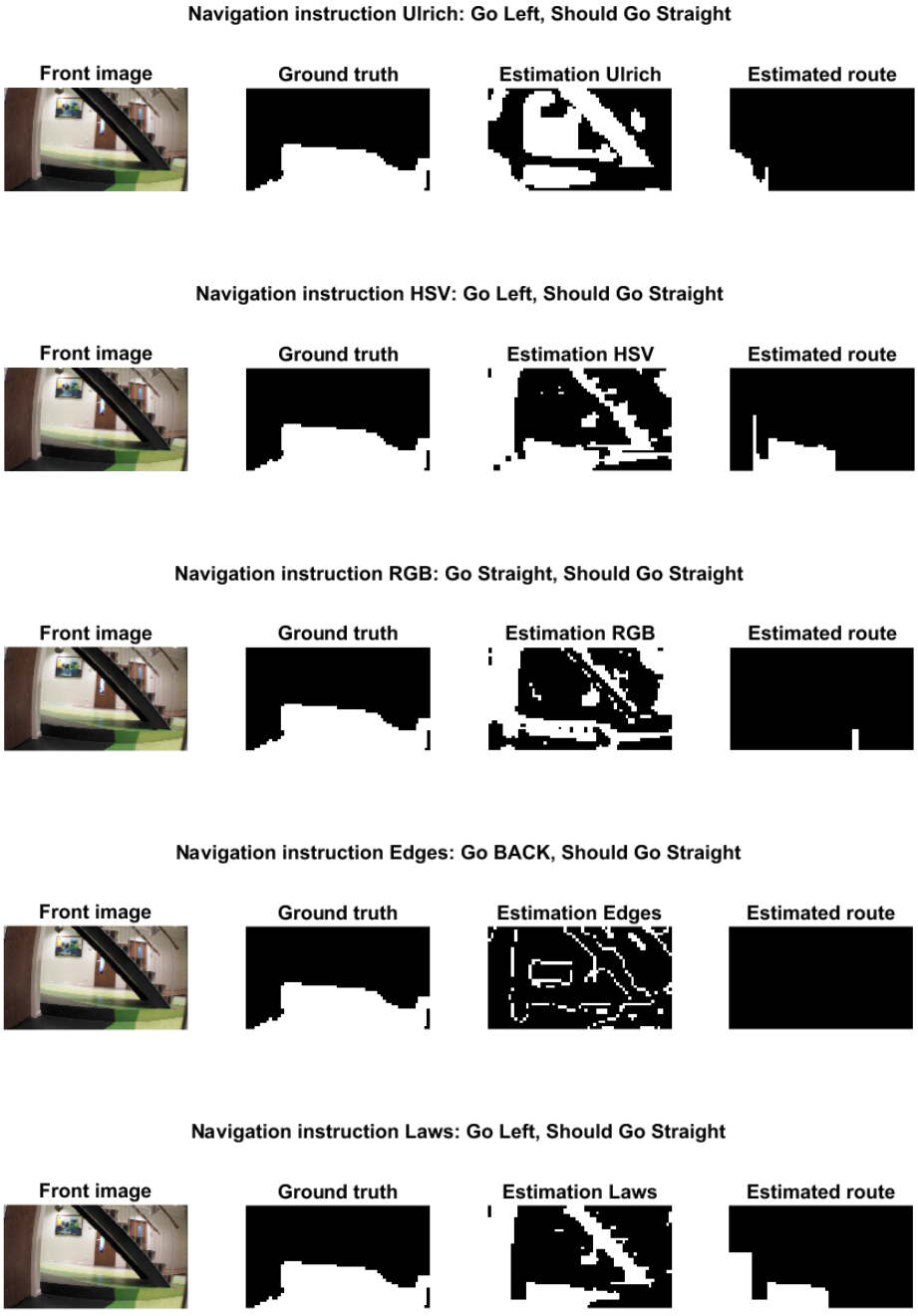

Figure E-14: Example navigation results dataset 2

Bibliography

Achtelik, M., Achtelik, M., Weiss, S., & Siegwart, R. (2011). Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. *Proceedings - IEEE International Conference on Robotics and Automation*, 3056–3063.

Alenya, G., Negre, A., & Crowley, J. (2009). A comparison of three methods for measure of time to contact. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 4565–4570.

Alvarez, H., Paz, L., Sturm, J., & Cremers, D. (2014). Collision Avoidance for Quadrotors with a Monocular Camera. *Proc. of The 12th International Symposium on Experimental Robotics (ISER)*, 1–13.

Ancona, N. (n.d.). A Fast Obstacle Detection Method based on Optical Flow.

Arokiasami, W., & Chen, T. (2015). Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2. , *2*, 201–213. doi: 10.1007/978-3-319-13356-0

Bills, C., Chen, J., & Saxena, A. (2011). Autonomous MAV flight in indoor environments using single image perspective cues. *Proceedings - IEEE International Conference on Robotics and Automation*, 5776–5783.

Bipin, K., Duggal, V., & Madhava Krishna, K. (2015). Autonomous navigation of generic monocular quadcopter in natural environment. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1063–1070.

Byrne, J., & Taylor, C. (2009). Expansion segmentation for visual collision detection and estimation. *2009 IEEE International Conference on Robotics and Automation*, 875–882.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698. doi: 10.1109/TPAMI.1986.4767851

Celik, K., Chung, S., Clausman, M., & Somani, A. (2009). Monocular vision SLAM for indoor aerial vehicles. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, *2013*, 1566–1573.

Chavez, A., & Gustafson, D. (2009). Vision-based obstacle avoidance using SIFT features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial In-*

telligence and Lecture Notes in Bioinformatics), *5876*(2), 550–557.

Daftry, S., Dey, D., Sandhawalia, H., Zeng, S., Bagnell, J., & Hebert, M. (2015). Semi-Dense Visual Odometry for Monocular Navigation in Cluttered Environment.

De Croon, G., De Wagter, C., Remes, B., & Ruijsink, R. (2011). Sky segmentation approach to obstacle avoidance. *IEEE Aerospace Conference Proceedings*, 1–31. doi: 10.1109/AERO.2011.5747529

de Croon, G., Groen, M., De Wagter, C., Remes, B., Ruijsink, R., & van Oudheusden, B. (2012). Design, aerodynamics and autonomy of the DelFly. *Bioinspiration & Biomimetics*, *7*(2), 025003. doi: 10.1088/1748-3182/7/2/025003

de Croon, G., de Wagter, C., Remes, B., & Ruijsink, H. (2012). The Appearance Variation Cue for Obstacle Avoidance. *IEEE Transactions on Robotics IF - 2.571*, *28*(2), 529–534.

Derpanis, K. (2004). The harris corner detector. *York University*, 2–3. Retrieved from `http://windage.googlecode.com/svn/trunk/Mindmap/Tracking/Papers/[2004] The Harris Corner Detector.pdf`

Dey, D., Shankar, K., Zeng, S., Mehta, R., Agcayazi, M., Eriksen, C., ... Bagnell, J. (2015). Vision and Learning for Deliberative Monocular Cluttered Flight. *Field and Service Robotics*.

Engel, J., Sch, T., & Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. *Eccv*, 834–849.

Engel, J., Sturm, J., & Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. *Proceedings of the IEEE International Conference on Computer Vision*, 1449–1456.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO : Fast Semi-Direct Monocular Visual Odometry. *IEEE International Conference on Robotics and Automation (ICRA)*, 15–22.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, *4*(1), 23–33. doi: 10.1109/100.580977

Garcia, A., Mattison, E., & Ghose, K. (2015). High-speed Vision-based Autonomous Indoor Navigation of a Quadcopter.

Geraerts, R., & Overmars, M. (2004). A comparative study of probabilistic roadmap planners. *Springer Tracts in Advanced Robotics*, *7 STAR*, 43–57. doi: 10.1007/978-3-540-45058-0_4

Gini, G., & Marchi, A. (2002). Indoor Robot Navigation With Single Camera Vision. *Pris*, 67–76. doi: 10.1.1.97.4390

Heng, L. (2014). Autonomous Visual Mapping and Exploration With a Micro Aerial Vehicle. *Journal of Field Robotics*, *24*(5), 421–434. doi: 10.1002/rob

Heng, L., Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing. *Proceedings - IEEE International Conference on Robotics and Automation*, 2472–2477. doi: 10.1109/ICRA.2011.5980095

Horn, B., Fang, Y., & Masaki, I. (2007). Time to Contact Relative to a Planar Surface. *2007 IEEE Intelligent Vehicles Symposium*, 68–74. doi: 10.1109/IVS.2007.4290093

Kessler, C., Ascher, C., Frietsch, N., Weinmann, M., & Trommer, G. (2010). Vision-based attitude estimation for indoor navigation using Vanishing Points and lines. *IEEE/ION Position, Location and Navigation Symposium*, 310–318. Retrieved from

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5507247 doi: 10.1109/PLANS.2010.5507247

Kim, Y., & Kwon, S. (2015). A heuristic obstacle avoidance algorithm using vanishing point and obstacle angle. *Intelligent Service Robotics*, *8*(3), 175–183. Retrieved from http://link.springer.com/10.1007/s11370-015-0171-4 doi: 10.1007/s11370-015-0171-4

Lee, B., Daniilidis, K., & Lee, D. (2015). Online Self-Supervised Monocular Visual Odometry for Ground Vehicles. , 5232–5238.

Lee, J., Lee, K., Park, S., Im, S., & Park, J. (2011). Obstacle avoidance for small UAVs using monocular vision. *Aircraft Engineering and Aerospace Technology*, *83*(6), 397–406. Retrieved from http://www.emeraldinsight.com/doi/abs/10.1108/00022661111173270 doi: 10.1108/00022661111173270

Li, G., Wu, G., & Wei, W. (2006). ND-DWA: A Reactive Method for Collision Avoidance in Troublesome Scenarios. *2006 6th World Congress on Intelligent Control and Automation*, 9307–9311. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1713802 doi: 10.1109/WCICA.2006.1713802

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., & Thrun, S. (2005). Anytime Dynamic A *: An Anytime , Replanning Algorithm. *Science*, 262–271. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Anytime+Dynamic+A+*:+An doi: 10.1.1.63.689

Lippiello, V., Loianno, G., & Siciliano, B. (2011). MAV indoor navigation based on a closed-form solution for absolute scale velocity estimation using Optical Flow and inertial data. *Proceedings of the IEEE Conference on Decision and Control*, 3566–3571. doi: 10.1109/CDC.2011.6160577

Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. , 1–28. doi: 10.1023/B:VISI.0000029664.99615.94

Mcfadyen, A., & Mejias, L. (2015). A survey of autonomous vision-based See and Avoid for Unmanned Aircraft Systems. *Progress in Aerospace Sciences*, 1–17. Retrieved from http://www.sciencedirect.com/science/article/pii/S0376042115300208 doi: 10.1016/j.paerosci.2015.10.002

Mercado, D., Castillo, P., & Lozano, R. (2015). Quadrotor s Trajectory Tracking Control using Monocular Vision Navigation *.

Mori, T., & Scherer, S. (2013). First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. *Proceedings - IEEE International Conference on Robotics and Automation*, 1750–1757. doi: 10.1109/ICRA.2013.6630807

NEgre, A., Braillon, C., Crowley, J., & Laugier, C. (2008). Real-time time-to-collision from variation of intrinsic scale. *Springer Tracts in Advanced Robotics*, *39*, 75–84. doi: 10.1007/978-3-540-77457-0_8

Newcombe, R., Lovegrove, S., & Davison, A. (2011). DTAM: Dense tracking and mapping in real-time. *Proceedings of the IEEE International Conference on Computer Vision*, 2320–2327. doi: 10.1109/ICCV.2011.6126513

Okada, R., Taniguchi, Y., Furukawa, K., & Onoguchi, K. (2003). Obstacle Detection Using Projective Invariant and Vanishing Lines.

Paalanen, P., Kyrki, V., & Kamarainen, J. (2008). Towards monocular on-line 3d reconstruction. *Workshop on Vision in Action: ...*. Retrieved from

http://hal.inria.fr/inria-00325795/

Parag, H., & Singh, S. (2001). Obstacle Detection Using Adaptive Color Segmentation and Color Stereo Homography. , 705–710.

Rish, I. (2016). An Empirical Study of the Naïve Bayes Classifier. (January 2001).

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. *Lecture Notes in Computer Science*, *3951 LNCS*, 430–443. doi: 10.1007/11744023_34

Sa, I., He, H., Huynh, V., & Corke, P. (2013). Monocular vision based autonomous navigation for a cost-effective MAV in GPS-denied environments. *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics: Mechatronics for Human Wellbeing, AIM 2013*, 1355–1360. doi: 10.1109/AIM.2013.6584283

Sagar, J., & Visser, A. (2014). Obstacle avoidance by combining background subtraction , optical flow and proximity estimation. *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014*, 142–149.

Saha, S., Natraj, A., & Waharte, S. (2014). A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment. *2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology*, 189–195. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7024382 doi: 10.1109/ICARES.2014.7024382

Saxena, A., Chung, S., & Ng, A. (2005). Learning depth from single monocular images. *Advances in Neural ...*. Retrieved from http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2005_684.pdf

Scaramuzza, D., & Fraundorfer, F. (2011). Tutorial: Visual odometry. *IEEE Robotics and Automation Magazine*, *18*(4), 80–92. doi: 10.1109/MRA.2011.943233

Schops, T., Engel, J., & Cremers, D. (2014). Semi-dense visual odometry for AR on a smartphone. *IEEE International Symposium on Mixed and Augmented Reality*, 1–6. doi: 10.1109/ISMAR.2014.6948420

Sezer, V., & Gokasan, M. (2012). A novel obstacle avoidance algorithm: "Follow the gap method". *Robotics and Autonomous Systems*, *60*(9), 1123–1134. Retrieved from http://dx.doi.org/10.1016/j.robot.2012.05.021 doi: 10.1016/j.robot.2012.05.021

Shapiro, L., & Stockman, G. (2001). *Computer vision*. Prentice Hall. Retrieved from https://books.google.nl/books?id=FftDAQAAIAAJ

Shen, S., Michael, N., & Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. *Proceedings - IEEE International Conference on Robotics and Automation*, 20–25. doi: 10.1109/ICRA.2011.5980357

Shen, S., Mulgaonkar, Y., Michael, N., & Kumar, V. (2013). Vision-based state estimation for autonomous rotorcraft MAVs in complex environments. *Proceedings - IEEE International Conference on Robotics and Automation*, 1758–1764. doi: 10.1109/ICRA.2013.6630808

Shridhar, M. (2015). Monocular SLAM for Real-Time Applications on Mobile Platforms. , 1–10.

Smith, S., & Brady, J. (1997). SUSANa new approach to low level image processing. *International journal of computer vision*, *23*(1), 45–78. Retrieved from http://link.springer.com/article/10.1023/A:1007963824710 doi: 10.1023/A:1007963824710

Soundararaj, S., Sujeeth, A., & Saxena, A. (2009). Autonomous indoor helicopter flight using a single onboard camera. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 5307–5314. doi: 10.1109/IROS.2009.5354617

Ta, D., Ok, K., & Dellaert, F. (2014). Vistas and parallel tracking and mapping with Wall-Floor Features: Enabling autonomous flight in man-made environments. *Robotics and Autonomous Systems*, *62*(11), 1657–1667. doi: 10.1016/j.robot.2014.03.010

Ulrich, I., & Nourbakhsh, I. (2000). Appearance-Based Obstacle Detection with Monocular Color Vision. *Artificial Intelligence*(August), 866–871. Retrieved from `http://www.aaai.org/Papers/AAAI/2000/AAAI00-133.pdf` doi: 10.1.1.43.2596

Verbickas, R., & Whitehead, A. (2014). Sky and Ground Detection Using Convolutional Neural Networks. (64), 1–10.

Wang, C., Liu, W., & Meng, M. (2015). Obstacle Avoidance for Quadrotor Using Improved Method based on optical flow. (August), 1674–1679.

Wedel, A., Schoenemann, T., Brox, T., & Cremers, D. (2007). WarpCut Fast Obstacle Segmentation in Monocular Video. *Pattern Recognition*, 264–273. Retrieved from `http://link.springer.com/10.1007/978-3-540-74936-3_27` doi: 10.1007/978-3-540-74936-3_27

Weiss, S., Achtelik, M., Kneip, L., Scaramuzza, D., & Siegwart, R. (2011). Intuitive 3D maps for MAV terrain exploration and obstacle avoidance. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *61*(1-4), 473–493. doi: 10.1007/s10846-010-9491-y

Wzorek, M., Kvarnstrom, J., & Doherty, P. (2010). Choosing Path Replanning Strategies for Unmanned Aircraft Systems. *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*(Icaps), 193–200.