Sky Segmentation Approach to Obstacle Avoidance

G.C.H.E. de Croon, B.D.W. Remes, C. De Wagter, and R. Ruijsink

Abstract

One capability that is essential for avoiding both other flying vehicles and static obstacles is to visually discern possible obstacles in a UAV's environment from the sky. The main contribution of this article is the presentation of a feasible approach to obstacle avoidance based on the segmentation of camera images in sky and non-sky regions. The approach is named the *Sky Segmentation Approach* to obstacle avoidance (SSA). The central concept is that potentially threatening static obstacles protrude from the horizon line. The main challenge for SSA is automatically interpreting the images robustly enough for use in various environments and fast enough for real-time performance. In order to achieve robust image segmentation, machine learning is applied to a large database of images with many different types of skies. From these images, different types of visual features are extracted, among which most of the features investigated in the literature. In the interest of execution speed and comprehensibility, decision trees are learned to map the feature values at an image location to a classification as sky or non-sky. The learned decision trees are fast enough to allow real-time execution on a Digital Signal Processor: it is run onboard a small UAV at ~ 30 Hz. Experiments in simulation and preliminary experiments on a small UAV show the potential of SSA for achieving robust obstacle avoidance in urban areas.

1 Introduction

Small Unmanned Air Vehicles (UAVs) hold a promise as sensors in the sky for many applications. Recent developments have led to the wide availability of autopilots that allow even the smallest of UAVs to fly autonomously in open outdoor areas [1, 2]. These systems are based on the use of GPS and IMU, which can ensure safe operation high in the sky. However, there are no autonomous systems yet that allow operation of small UAVs closer to the ground or in urban areas. The main missing element is an autonomous routine for successful collision avoidance.

There has been extensive research that does bring such collision avoidance within reach. Notably, research on larger UAVs with multi-axis, long range, high-resolution laser scanners (cf. [3, 4]) allows for successful navigation in cluttered environments. However, these UAVs weighed more than 75 kg and had to use most of their payload capability to lift the laser scanner. In this article, we focus on a sense-and-avoid system that can be applied to UAVs in the range of 500g to 1.5 kg (also referred to as Micro Air Vehicles, MAVs). Such a small system would allow larger UAVs to use their payload almost entirely for their mission. Laser scanners have been miniaturized for use on MAVs by sacrificing both resolution and sensing directions. Scanners that measure distances to obstacles in a single plane through the MAV are now part of the most successful systems for indoor flight (cf. [5, 6]). The range of these scanners is < 30 m, which may not be sufficient for outdoor flight when the UAV is moving at higher speeds. In addition, for some obstacles such as other air vehicles or power lines, sensing in a single plane is not enough.

Research on outdoor sense-and-avoid for MAVs has mainly focused on the use of a camera. It is a passive sensor and as such consumes less energy than active sensors as, e.g., laser scanners. In addition, a camera can provide information about a large part of the environment at once, including obstacles at large distances. Moreover, cameras can be made very lightweight, up to the order of milligrams. Cameras have been mainly used for *stereo vision* (c.f., [7]) and for *optic flow* (e.g., [8, 9, 10, 11, 12]). Both these techniques are likely to play a role in final obstacle-avoidance capabilities, but they also have their limitations. In particular, stereo vision has a limited range in which it can determine the distance to obstacles. This range depends on the resolution of the images and the base distance between the two cameras. The base distance is inherently limited in MAVs, so stereo vision will only be useful for detecting obstacles at a relatively short range. The main current limitation of optic flow is that it heavily relies on texture in the images. As a consequence, obstacle avoidance on the basis of optic flow fails around many human-built structures, since they can have too little texture.

Another visual cue that can be exploited for obstacle avoidance is the relation between non-sky segments in an image and the horizon line. In particular, non-sky segments located above the horizon line are possible collision threats [13, 14]. To our knowledge, the use of an obstacle's protrusion from the horizon line for obstacle avoidance by UAVs has first been suggested in [15]. However, in that study no algorithm was developed for actually using the sky segmentation to avoid obstacles. Instead, sky segmentation was applied to a UAV that had to hit a balloon. In addition, the skyline as detected in the image was taken to be the horizon line. This is incorrect, since these two are only equal in the case of a completely open terrain: an MAV with a forward-looking camera close to the roof of a building could easily be fooled to think that the edge of the roof is the horizon line, resulting in a probable crash.

The main contribution of this article is the introduction of a feasible approach to obstacle avoidance based on the segmentation of camera images into sky and non-sky regions. The approach is named the *Sky-Segmentation Approach* (SSA) to obstacle avoidance. The principal challenge for the real-world success of SSA is to automatically interpret images robustly enough for use in various environments and fast enough for real-time performance. If the segmentation performance is adequate, SSA has to interpret the resulting information in terms of directions to possible collision threats and take corresponding actions.

The remainder of the article is structured as follows. In Section 2 a theoretical example with restrictive assumptions is given that allows a rotary-wing aircraft to successfully, albeit inefficiently, avoid any static obstacle. The example sets the stage for the rest of the article, in which a strategy for SSA is devised that takes into account the violations of the aforementioned assumptions. In Section 3, the main challenge of this approach is addressed, namely, how to automatically interpret the images robustly enough for use in various environments and fast enough for real-time performance. In order to achieve robust image segmentation, machine learning is applied to a large database of images with many different types of skies and obstacles. From these images, different types of visual features are extracted, among which most of the features investigated in the literature. In the interest of execution speed and comprehensibility, decision trees are learned to map the feature values at an image location to a classification as sky or non-sky. In Section 4, the learned decision trees are used in simulation to verify the validity of the obstacle avoidance approach. Subsequently, a preliminary test on a real platform is performed in Section 5. Subsequently, we discuss the potential and the limitations of SSA in Section 6. Finally, the conclusions of the current study are drawn in Section 7.

2 Idealized Sky Segmentation Approach

In this section, we start with a theoretical example of how the ideal implementation of SSA is able to avoid any obstacle in the path of the MAV. It will be explained how a given behavior and five strong assumptions will result in a system that can avoid any static obstacle while going from place A to B. This purely theoretical example serves to introduce the main components and challenges of SSA behavior.

The assumptions are that:

1. The MAV is able to hover in place, yaw, and ascend and descend without going forwards or backwards.

2. The MAV is able to fly forwards while staying at the same height.

3. There are no obstacles directly above the MAV at location A.

4. The MAV can safely land when arrived at location B.

5. The MAV is able to perfectly segment images into sky and non-sky regions with real-time performance.

6. The MAV knows its state (attitude and position).

The given behavior is then as follows. At point A, the MAV lifts off and first yaws until it faces point B. Then it segments the image and uses the knowledge of its state to project the horizon line into the segmented image. It determines whether there are protrusions from the horizon line. In an iterative process, the MAV ascends and segments the image, until there are no protrusions from the horizon line anymore. This means that at the height of the MAV's camera, there are no obstacles until the visible horizon. The MAV then still ascends h_b m., which is larger than the distance from the camera to the bottom of the MAV's body. Now the MAV starts moving forwards. Arrived at location B, the MAV descends. Since after the initial turning, the MAV only moves (1) up at A, (2) forwards from A to B, and (3) downwards at B the MAV can only hit obstacles on these trajectories. The first option is excluded by assumption 3. The second option is excluded since such an obstacle should have appeared as an protrusion during segmentation and therefore excluded by assumption 5 and 6. The third option is excluded by assumption 4. As a consequence, the behavior described above will result in the avoidance of any static obstacle, given that the assumptions hold.

Of course, the above assumptions are strong, especially assumptions number 5 and 6. The main challenges involved in SSA are to cope with the violations of these assumptions in practice. In the remainder of the article, methods for segmenting and flying are discussed that take into account the violation of the assumptions.

3 Sky Detection

The segmentation of images into the two classes of *sky* and *non-sky* is a rather well-studied subject in the field of MAV research [16, 17, 18, 15, 19, 20, 21, 22, 23, 24]. The goal of the image segmentation is almost always to obtain an estimate of the horizon line, which conveys information on the pitch and roll of the air vehicle. As a consequence, most studies contain an underlying assumption that the MAV is flying quite high in the sky, with few to no obstacles protruding from the horizon line. The goal of segmentation in SSA is different, namely it is to detect obstacles that protrude from the horizon line, which itself may not be visible. As will become clear, the difference in goals between the current study and past studies will be reflected in the obtained segmentation results when the camera is not high in the sky.

The image segmentation method employed in our implementation of SSA learns the appearance of sky and non-sky of a large publicly available database of images. In Subsection 3.1 the setup of the segmentation experiments is discussed. Subsequently, the appearance features extracted from the images are explained in Subsection 3.2. Most of these features have been used in the literature before. Finally, the image segmentation results are shown and analyzed in Subsection 3.4.

3.1 Setup image segmentation experiments

As mentioned in the introduction, the main challenge of SSA is to automatically segment the images robustly enough for use in various environments and fast enough for real-time performance. With these two criteria in mind, the following approach is taken to image segmentation.

Machine learning is employed to learn the appearance of the classes of sky and non-sky in images. As an image collection, the relatively large and publicly available *labelME* database is used [25][1]. The database contains a large variety of images from different environments, in which all entities have been labeled by hand by different persons. Anyone can add images and help in the labeling process. For the experiments in this article, the labelME database was downloaded on the 22nd of September 2009. At that time, 7456 images contained an entity labeled as *sky*. Many of the images have been taken in different urban environments, but the database also contains considerable numbers of images in snowy mountain areas, forrests, green meadows, seas, etc. Images have been taken under a wide variety of

[1]`http://labelme.csail.mit.edu/`

light conditions at different times of day and in different weather conditions. From this image set, 6085 images were selected which corresponded to the idea of sky as employed in SSA. Figure 1 shows eight example images (top row) together with their labelings (bottom row - white represents sky). The two rightmost images in Figure 1 were excluded, since the first labels the branches and leaves of trees as sky, while the second has been taken at night, a case that is excluded for this study.

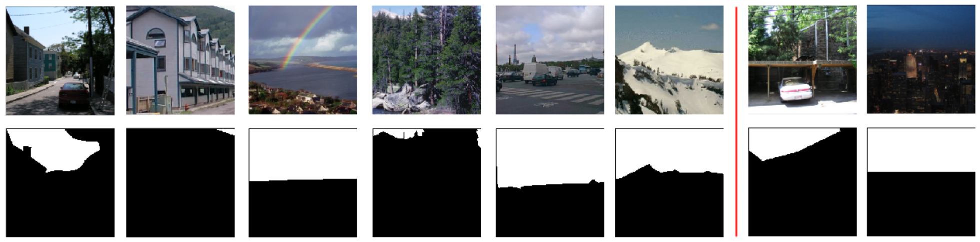

Figure 1: Eight example images from the labelME database. The top row shows the images, the bottom row the ground truth segmentations as labelled by human users of the database. The rightmost two images have been excluded from the database for the experiments.

All images are resized and cropped with the labelME function *LMimresizecrop.m* to a size of $W \times H = 120 \times 120$ pixels. The function first resizes the smallest side of the image to 120 pixels and then crops the other one. This procedure maintains the relative scaling of the axes. The image size is chosen on the basis of the final implementation on the *Surveyor SRV-1 Blackfin* camera system, which will execute the segmentation algorithm on 160×120 images.

The first 10% of the images are reserved as the test set, while the remaining 90% serve as the training set. From the images in the training set, the features explained in Subsection 3.2 are extracted. For training, $250,000$ local samples are extracted from all the images in random locations (~ 45 samples per image). The extracted data is a $250,000 \times n$ matrix, with n the number of features. This matrix is saved in the format employed by *WEKA*, a publicly available machine learning toolkit that contains many implementations of different machine learning methods [26][2].

Decision trees are learned with WEKA's *J48* implementation of the *C4.5* algorithm [27]. The choice for decision trees has three main motivations. First, decision trees are computationally efficient, involving a number of

[2]`http://www.cs.waikato.ac.nz/~ml/weka/`

comparisons equal to the depth of the path in the tree that is followed by evaluating the current local image sample. In addition, only the features tested in the current path have to be extracted, leading to a further speed-up. Second, decision trees of limited tree depth are in general quite comprehensible. One advantage of this comprehensibility is that it increases the insight into successes and failures of the image segmentation method, allowing informed improvements. Third, the proportion of sky-pixels and non-sky-pixels remaining at the leaf nodes form a measure of uncertainty about the classification at that node. Of course, C4.5 decision trees also have disadvantages. For one, they do not make linear combination of features, placing class boundaries orthogonally to the feature axes. In addition, the boundaries placed in the feature space are crisp. This means that small differences in feature values sometimes lead to entirely different classifications.

In the experiments, the J48 algorithm uses 90% of the $250,000$ training samples for training and 10% for testing. This testing is to get a first impression of the performance of the tree. For J48, the standard settings are employed, except for the confidence C, which is set to 0.05, and the minimum number of instances at leaf nodes M, which is set to 2000. Both these settings stimulate shallower trees that are consequently faster at execution time and may generalize better to unseen instances.

After training, the classifier is employed to entirely segment all test images. For all pixels in the test set, the classification is compared to the ground truth value, leading to a confusion matrix containing the counts of all true positives and false positives of both the sky and the non-sky classes.

3.2 Features extracted from the images

On the basis of previous studies [16, 17, 18, 15, 19, 20, 21, 22, 23, 24], a large group of features was selected for extraction from the images in the labelME database. In addition, some novel features are introduced in this article. In total 34 features are extracted for each pixel during the training phase. The J48 decision tree learning algorithm then selects a small subset of features for execution on the MAV platform.

Features 1-3: Pixel values from the $\mathcal{RGB}$-color space.

Features 4-6: Pixel values from the $\mathcal{HSV}$-color space.

Features 7-9: Pixel values from the $\mathcal{Y}\mathcal{C}b\mathcal{C}r$-color space.

Features 10-16: Patch features introduced by [19], plus an additional novel feature. Although originally extracted from all three channels of an image, here they are extracted from the gray-scale transformed image. The features are extracted from patches of size $l \times l$ (in the experiments 5×5), containing L pixels. The features are defined as follows in our implementation:

* Patch mean: $m = \frac{1}{L}\sum_{i=1}^{L} p_i$, with p_i as the i^{th} pixel value.
* Patch standard deviation: $s = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(p_i - m)^2}$.
* Smoothness: $q = 1 - \frac{1}{1+s^2}$.
* Third moment: $t = \frac{1}{L}\sum_{i=1}^{L}(p_i - m)^3$.
* Uniformity: $u = \sum_{b=1}^{B} P(b)^2$, where $P(b)$ is the probability for the pixel value falling into a bin b of a histogram that has the interval $[0, 1]$ (of pixel values) and in the experiments has $B = 10$ bins.
* Entropy: $e = -\sum_{b=1}^{B} P(b)\log(P(b))$.
* Texture: $x = \frac{1}{L-1}\sum_{i=1}^{L} |p_i - p_j|$, where p_j is the central pixel of the image patch. This feature is inspired by the work in [28].

Feature 17: The relative y-coordinate of the pixel in the image, $y(p)' = \frac{y(p)}{H}$.

Features 18-21: Gradient features comparable to those used in [18]. The features are:

* The absolute horizontal gradient: $dx = |p(x-1, y) - p(x+1, y)|$, with $p(x, y)$ the pixel value at image coordinate (x, y).
* The absolute vertical gradient: $dy = |p(x, y-1) - p(x, y+1)|$.
* The absolute combined gradient: $dg = dx + dy$.
* The relative absolute gradient: $rdg = \frac{dg}{\max_{\mathcal{I}}(dg)}$

Features 22-24: Gradient features of the blue spectrum. As noted in [22], when going through the image from the top down, sky pixels become brighter but less colored towards the horizon line. In the implementation, the $\mathcal{HSV}$-image is first processed to isolate the blue component. An image $\mathcal{J}$ is formed by $\mathcal{J} = \mathcal{S}(\mathcal{H} > 0.475 \wedge \mathcal{H} < 0.650 \wedge \mathcal{V} > 0.15)$. Then, the following features are extracted from this one-channel image $\mathcal{J}$:

* The horizontal gradient: $dbx = p(x-1,y) - p(x+1,y)$.
* The vertical gradient: $dby = p(x,y-1) - p(x,y+1)$.
* The absolute combined gradient: $dbg = |dbx| + |dby|$.

Feature 25-26: Relative illuminance features that relate the pixel value in a grayscale image to the pixel value at a certain percentile of the cumulative distribution of pixel values in the image:

* $i_1 = \frac{p_i}{p_m}$, where p_m in the experiments is at the 75^{th} percentile.
* $i_2 = \frac{p_i}{p_n}$, where p_n in the experiments is at the 0^{th} percentile (minimal intensity).

Features 27-28: Corner values v_h and v_n as obtained with the methods of [29] and [30]. The intuition behind the introduction of these values is that corner-like structures should rarely occur in the sky.

Features 29: Grayness feature. A measure of how far the pixel is from the $\mathcal{C}b$ and $\mathcal{C}r$ values for gray g in $\mathcal{Y}\mathcal{C}b\mathcal{C}r$-images: $g = (p_{\mathcal{C}b} - 0.5)^2 + (p_{\mathcal{C}r} - 0.5)^2$. The rationale behind its introduction is that it may be used to exclude colored pixels that are not blue.

Features 30-34: Fisher discriminant features inspired by [23], in which Fisher discriminant analysis was performed on the $\mathcal{RGB}$-space to obtain a linear recombination of the color channels that separates the classes of sky and non-sky as much as possible. Including other color spaces and applying the same method also to only the color channels of the color spaces results in the following features, of which all weight values have been obtained on the labelME training set:

* $FD_{\mathcal{RGB}} = -3.77\mathcal{R} - 1.25\mathcal{G} + 12.40\mathcal{B} - 4.62$.
* $FD_{\mathcal{HSV}} = 3.35\mathcal{H} + 2.55\mathcal{S} + 8.58\mathcal{V} - 7.51$.
* $FD_{\mathcal{SV}} = 2.77\mathcal{H} + 0.83\mathcal{S} - 1.40$.
* $FD_{\mathcal{Y}\mathcal{C}b\mathcal{C}r} = 8.60\mathcal{Y} + 25.50\mathcal{C}b - 5.10\mathcal{C}r - 15.45$.
* $FD_{\mathcal{C}b\mathcal{C}r} = 19.75\mathcal{C}b - 4.46\mathcal{C}r - 8.18$.

3.3 Segmentation methods

The following methods have been applied to the test set of images:

1. A hand-tuned classifier using the $\mathcal{HSV}$-values. It classifies a pixel p in an image $\mathcal{I}$ as sky, if $\mathcal{V}_p > 0.7 \max_{\mathcal{I}}(\mathcal{V}) \wedge \mathcal{H}_p > 0.4 \wedge \mathcal{H}_p < 0.7$. In other words, a pixel belongs to the sky if it is bright and belongs to the color spectrum from blue to purple.

2. A method that segments the image by applying *Otsu* thresholding [31] on the $\mathcal{B}$ image channel [18].

3. The same method as above, but applied to the grayscale image.

4. The method of [23], which first transforms each image pixel in the $\mathcal{RGB}$-space on the basis of Fisher linear discriminant analysis to a single value $F_{\mathcal{RGB}}$, and then searches for a good threshold on this value. The search for a threshold is done for each image's histogram of $F_{\mathcal{RGB}}$-values. It depends on the presumed proportions of sky and non-sky in the image and assumes that it is better to take a threshold value that itself does not occur too much in the image.

5. Inspired by [21], the method of [32] is applied to the images. This computationally rather expensive method extracts many different features from the image, including Leung-Malik filter bank responses [33] and information on the vanishing points of the projection. In contrast to most other methods mentioned in this article, the extracted features are not evaluated at a pixel level but at the level of color segments in the image. The algorithm of [34] is used for image segmentation at different scales, and the classifications from segments at multiple scales are combined. The algorithm also estimates the location of the horizon line and uses this in the classification. The method is applied to the test set with $k = 100$ (cf. [32]).

6. A J48 tree named 'HSV-tree', which is trained on only the $\mathcal{HSV}$-features. Its performance can be compared with the hand-tuned $\mathcal{HSV}$-segmenter.

7. A J48 tree named 'full-tree', which is trained on all features extracted from the training images.

8. A J48 tree named 'tree without y-coordinate', which is trained on all features except for feature 17, the relative y-coordinate in the image. The reason for including this tree is that the nature of the data set (many photos taken at eye-level) has as a consequence that the relative y-coordinate carries information on the relation of a coordinate

to the horizon. This tree can be regarded as segmentation without information on the MAV's state.

9. A J48 tree named 'BF-tree' trained on features that are easily extracted from $\mathcal{YCbCr}$-images available to the BlackFin processor, also excluding the relative y-coordinate. In particular, it can use features 7– 16, 18 – 21, 25 – 29, and features 33 – 34.

The learned trees with corresponding uncertainties can be found at `http://www.bene-guido.eu/guido`. To illustrate which features are most informative, Table 1 shows which features have been selected by the J48 algorithm for classifying instances as sky or non-sky. If the relative y-coordinate is allowed, it is always selected. As shown later, it makes a considerable performance difference, indicating that on the long run the state estimate of the MAV should be an input to the image segmentation process. Furthermore, the Fisher-Discriminant features seem very informative, as does the novel patch feature that measures the texture. Since the BF-tree is also used in the simulated and real-world experiments, it is shown in Figure 2.

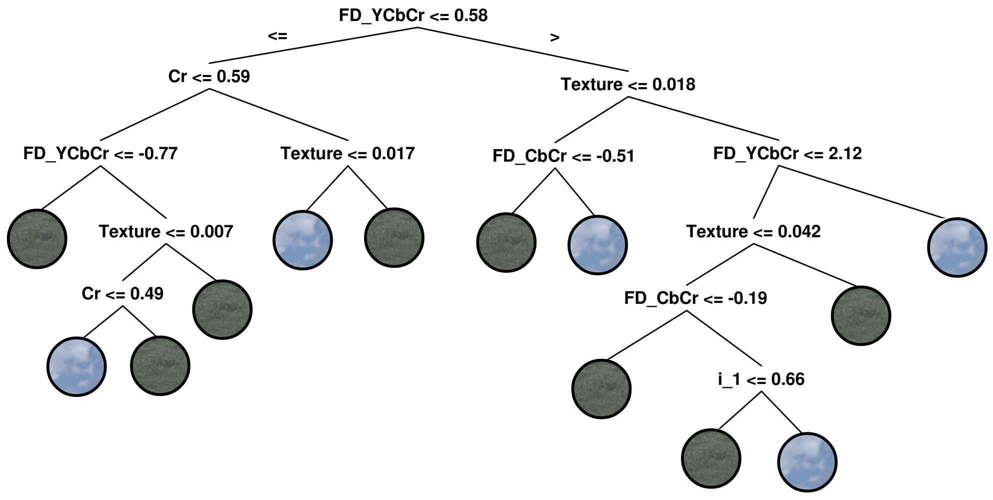

Figure 2: The decision tree learned for implementation on the BlackFin. All features are based on the $YCbCr$ color space. The left branch represents the variable check being true ($\leq$), the right branch represents the check being false ($>$).

The segmentations of each method are compared with the ground truth segmentations, leading to a confusion matrix per method. Such a confusion

Table 1: Features used in the different trees.

Tree	**Features used**
HSV-tree	$\mathcal{H}$, $\mathcal{S}$, $\mathcal{V}$
Full tree	$\mathcal{FD}_{\mathcal{RGB}}$, $\mathcal{FD}_{\mathcal{HSV}}$, $y(p)'$, patch texture and uniformity
tree w/o y-coordinate	$\mathcal{FD}_{\mathcal{RGB}}$, $\mathcal{FD}_{\mathcal{HSV}}$, $\mathcal{FD}_{\mathcal{Y}\mathcal{C}b\mathcal{C}r}$, dg, patch texture, standard deviation, and uniformity, i_1, g
BF-tree	$\mathcal{FD}_{\mathcal{Y}\mathcal{C}b\mathcal{C}r}$, $\mathcal{FD}_{\mathcal{C}b\mathcal{C}r}$, patch texture, $\mathcal{C}r$, i_1

matrix leads to one point on a graph that plots the precision of the sky-classifications vs. the recall of sky pixels. If possible, we modify parameters of the methods to generate ROC curves (cf. [35]). In the case of the method of [32] the certainty threshold for the sky-class is changed. In the case of the learned decision trees, the thresholds in the nodes of the tree are changed relative to the standard deviation of the corresponding variable in the training set. For example, to obtain a result with a higher precision, but a lower recall, the texture thresholds are all decreased with $m\sigma(\text{texture})$, where $m = 1$ leads to a higher precision and lower recall than $m = 0.2$. Of course, $m = 0$ represents the original tree. All other features have a similar clear relation to the precision and recall.

3.4 Results segmentation experiments

The results of the image segmentation experiments are shown in Figure 3. Of course, the goal is to have as much precision and recall as possible, so methods more to the top right perform better. The best performing method for a precision lower than 0.80 is that from [32]. It combines information at many levels and is the only method in the graph that classifies on the basis of more than local information. However, it is computationally quite an expensive method, and hence lends itself less well to the application of MAVs. Above a precision of 0.80 the method is slightly outperformed by the full tree. The next best methods are in the following order: tree with no y-coordinate, BF-tree / HSV-tree. The trained HSV-tree outperforms the hand-tuned HSV-method. The methods that adaptively search for a threshold in an image on the basis of pixel values all have a very low precision and high recall (Otsu Blue, Otsu Gray, and Thurrowgood 2009). This is to be expected, since the proportions of sky in the images of the data set can be quite low.

Figure 4 shows the proportions of sky $\in [0, 1]$ in the images in the test set. It is perhaps more fair to compare the adaptive threshold methods with the other ones only on images that have more than $\sim 25\%$ of sky. The results of this test are shown in Figure 5. The adaptive threshold methods

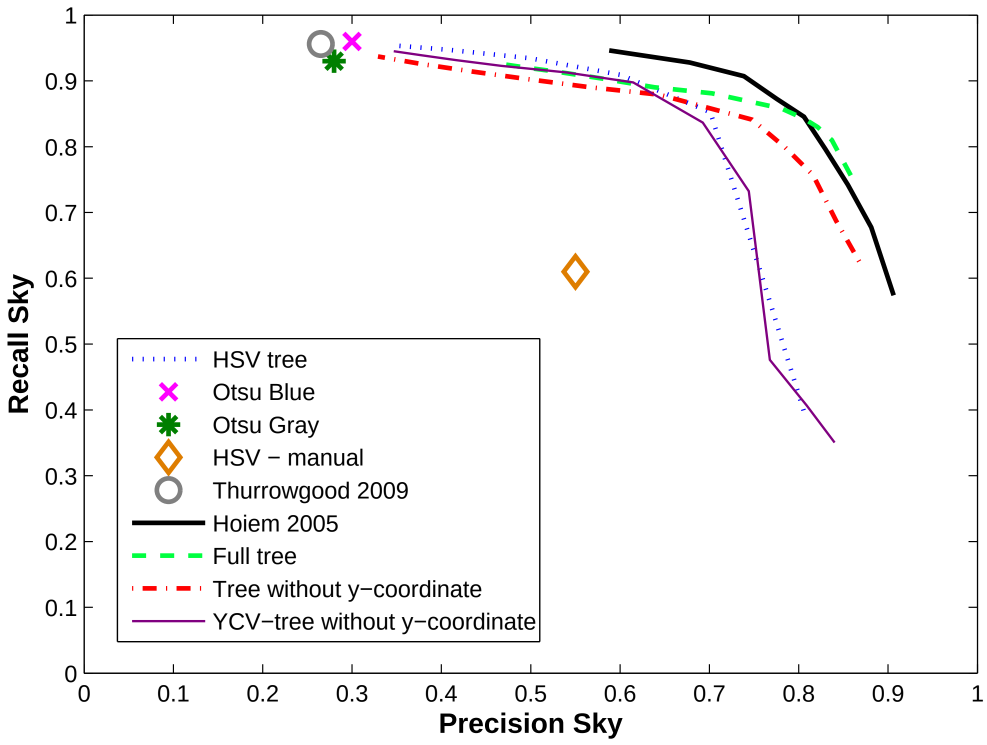

Figure 3: Precision vs. recall for the sky class. Results of all methods mentioned in Subsection 3.3

obtain a much better performance on these images, but still have a lower precision than the method from [32] and the learned decision trees.

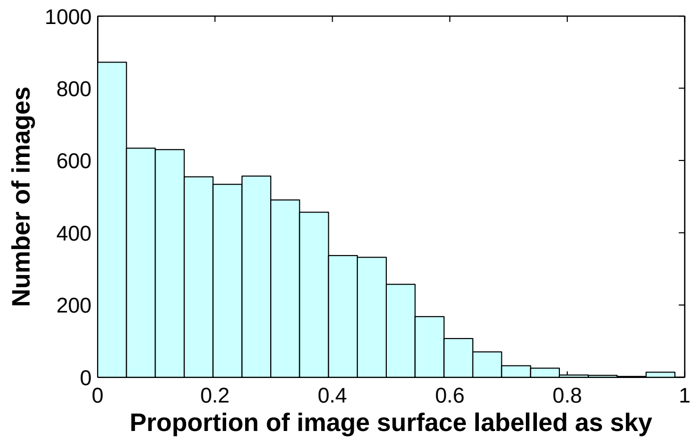

Figure 4: Histogram of the proportion of sky surface in the images of the labelME database.

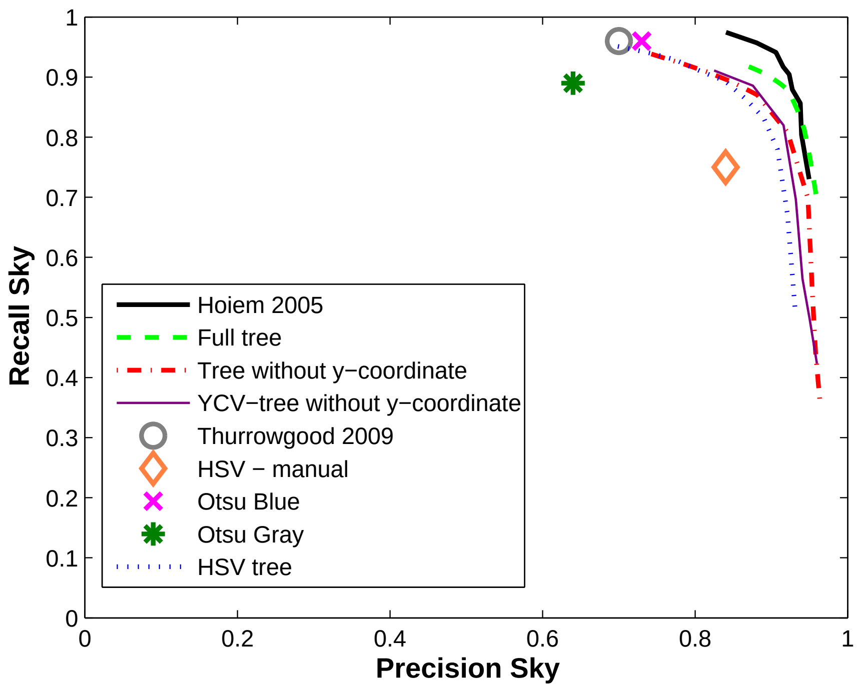

Figure 5: Precision vs. recall for the sky class. Results of all methods mentioned in Subsection 3.3 on images with more than 25% sky.

4 Simulation Experiments

The simulation experiments have two main goals. The first goal is to form a first dynamic test for the devised obstacle avoidance strategy. The simulation is quite realistic concerning the movements of the MAV, sensor noise, etc. and hence may reveal possible problems with the devised strategy. Successful obstacle avoidance in simulation would be a first validation of the SSA concept. The second goal is to test the strategy under many different circumstances without endangering the real MAV or third parties. For example, the differences can be studied between rotary wing MAVs and fixed wing MAVs.

The remainder of this section is structured as follows. First a strategy for SSA with a rotary wing MAV is introduced (Subsection 4.1). Subsequently, this strategy is tested in a few environments for validation (Subsection 4.2). Finally, we simulate the system that will be used for the real-world experiment (Subsection 4.3).

4.1 Proposed strategy for a rotary wing MAV

In this subsection, a strategy is proposed for employing SSA for a rotary wing MAV. Although the most efficient strategy for SSA would be to ascend if there are obstacles in many different directions and deviate if there are only obstacles in the flight path, here the focus is on an *only-ascend* strategy. Deviation is left to future work. First, the MAV's perception is discussed, and then the manner in which perception is mapped to obstacle-avoiding behavior. Again, the context of the strategy is that the MAV has to fly from a point A to a point B.

The MAV continuously segments incoming images into sky and non-sky regions. It uses the current state estimate of the pitch and roll ($\hat{\theta}$ and $\hat{\phi}$) to project a horizon line into the image[3]. The image area above the horizon line is subdivided into bins, representing angle intervals in which obstacles can be detected. The number of obstacle pixels in each bin is summed as a measure of obstacle presence in an angle interval. The uncertainty of all classifications is summed as well. Please remark that the bins can be placed either (a) vertically, or (b) orthogonally to the horizon line (see Figure 6). This corresponds to detecting the relative angle $\Delta\psi$ to objects either in (a) a body reference frame, or (b) in a plane parallel to the ground plane

[3]Throughout the text we assume that the camera is a perfect linear camera, or that images have first been undistorted before any further processing. In addition, it is assumed that the pitch angle is equal to the flight path angle; $\theta = \gamma$.

(earth reference frame). The latter is more interesting for obstacle avoidance with SSA, since rolling and pitching down always cause the appearance of ‘obstacles’ in a body reference frame (namely the earth itself).

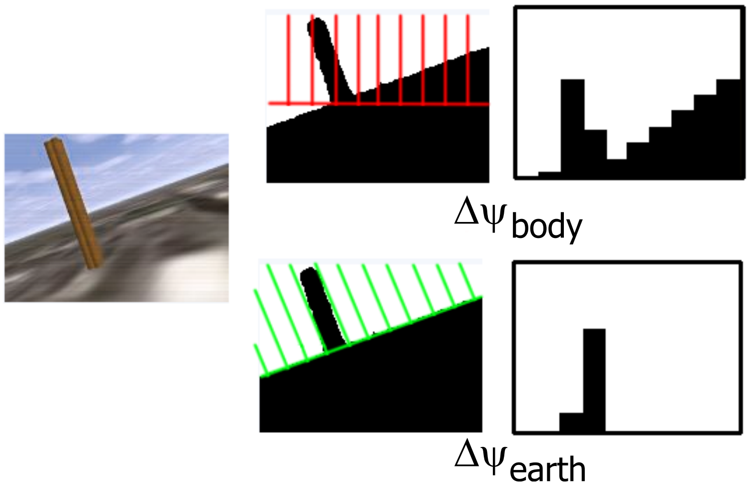

Figure 6: Left is an image from the simulator, as viewed by the MAV. In the center are two copies of the sky (white) / non-sky (black) segmentation. On the right, the result is shown of the discretization by summing the obstacle pixels in the bins. In the top row, obstacle pixels are always binned with vertical bins above half of the screen. As a consequence, the bins represent the presence of obstacles in a relative yaw-direction in a body reference frame ($\Delta\psi_{\text{body}}$). In the bottom row, obstacle pixels are binned in bins that are orthogonal to the (estimated) horizon line. As a consequence, the bins represent the presence of obstacles in a relative yaw direction in a earth reference frame ($\Delta\psi_{\text{earth}}$).

Furthermore, the bin size determines the angular resolution, while the angular interval captured by the camera depends on the roll angle. To retain the same angular resolution per bin, one needs to vary the distance between subsequent bin borders in pixels (Δx) according to the roll angle. The left part of Figure 7 shows the variables involved in determining Δx. The goal is to keep the part of the horizon line constant. In the case of the images with $W = 160$ and $B = 10$ bins, it should be kept at 16.0 pixels. On the basis of the roll angle ϕ, Δy can be expressed in terms of $a\Delta x$, with a the coefficient. Then the Pythagorean theorem can be used for determining Δx. The right part of Figure 7 shows two segmented images with their corresponding step size. The center bin with $\Delta\psi = 0$ is located at the intersection of the horizon line and the orthogonal line passing through the image center (blue lines in Figure 7).

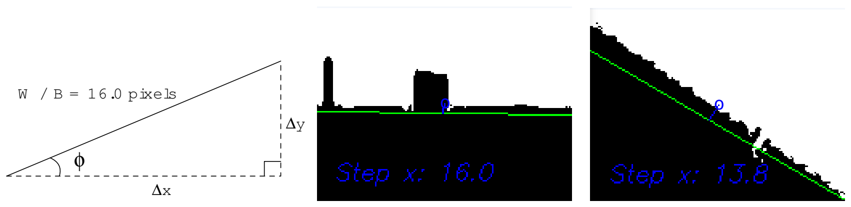

Figure 7: Left: Variables involved in the calculation of Δx. W is the image width, B the number of bins. Center and right: Two segmentation images from the simulator with different roll angles of the MAV. The yaw angle interval captured by the camera depends on the roll angle. On the left, the roll is almost 0 degrees, leading to a step size of $\Delta x = 16.0$ pixels (in the 160×120 image). On the right, the estimated roll angle is larger, necessitating a step size of $\Delta x = 13.8$ pixels to maintain the same angular resolution per bin. The image center is indicated with a blue circle. The blue line is the line through the image center orthogonal to the horizon line, correponding to $\Delta\psi = 0$.

The sums of obstacle pixels and corresponding uncertainties are used to determine the behavior of the MAV. In particular, the sum of obstacle pixels over all bins together, S_{all}, is compared with τ_{all}, a threshold that identifies cases in which there are a lot of obstacles in view. If so, the MAV should ascend. If not, the sum S_b is evaluated per bin b, with the help of a threshold τ_b. The safest strategy is to ascend if any bin is higher than the threshold. This is safest, since one segmentation provides information on which relative yaw angle intervals contain obstacles, but does not contain any information on the distance to the obstacles, nor on the width of open spaces. Still, in a large majority of cases it is reasonable to assume that obstacles are detected quite far away. If so, only obstacles in the flight path of the MAV pose a threat. When following a strategy of first yawing and then advancing, the center bins represent the angles relevant to the flight path. If any of the center bins supercede the threshold τ_b, the MAV should ascend as well. When ascending, MAV slows down to 0.2 m/s. It does not attempt to stop completely, since in the current setup the GPS-track is necessary for heading control.

Importantly, in the simulation experiments the number of obstacle pixels per bin is low-pass filtered. As a consequence, if the MAV detects an obstacle it will ascend slightly more than necessary to put the obstacle at the height of the camera. As a result, the body of the MAV does not collide with the obstacle. The low-pass filtering comes at the cost of detecting an obstacle slightly later. When not flying at too high a speed, this does not pose a problem.

Finally, the sum of uncertainties over all bins is an indication of how well the obstacle detection is working. The MAV should not fly with too high an uncertainty; it can then for example mistake clouds for obstacles, which could lead to the MAV ascending until its maximum height.

4.2 Validation of SSA in simulation

4.2.1 Experimental setup

For simulation the software gallery *SmartUAV* is used, developed by the Technical University of Delft and the DECIS-lab [36]. One of the interesting properties of SmartUAV is that it can communicate with different autopilots, such as the in-house developed *MAV-pilot* and the well-known open-source autopilot *Paparazzi*[4]. SmartUAV is set up to function both as a ground station that can be used for controlling real MAVs and as a simulator in which the components of the real MAV can be simulated. All communication between the simulated MAV and the ground station takes place in the same way as that between a real MAV and the ground station (over serial ports, UDP-connections, etc.).

For the validation of the strategy explained in Subsection 4.1, a simulated small helicopter is used. Without discussing the model and inner loop control in too much detail, it is important to note that the navigation is performed with the help of a simulated UBlox 4 Hz GPS component. Its measurements contain simulated noise, including a larger uncertainty of the height with respect to the North-East location. Furthermore it is noteworthy that the state estimate used for projecting the horizon line in the image is based on simulated gyros, accelerometers, and IR thermopiles as used in the Paparazzi system. The resulting state estimates have errors comparable to those under normal flying conditions.

In the validation experiments, two versions of the above-described strategy are tested: (1) the MAV has access to the state estimates, and (2) the MAV does not have access to the state estimates. In the latter case, the segmentation and obstacle detection assume the pitch and roll to be 0 degrees. The autopilot then only decides to further ascend if the state estimates $\hat{\theta}$

[4]`http://paparazzi.enac.fr/wiki/Main_Page`

and $\hat{\phi}$ are close enough to 0 degrees. The consequence of this scheme is that due to disturbances, the MAV will use fewer obstacle detection measurements. The inspiration for the second case comes from the fact that the real-world experiments will be performed with a fixed wing MAV, in which the segmentation algorithm does not have access to the MAV's state estimates.

The simulated experiments take place in a virtual environment of the campus of the Technical University of Delft. Figure 8 shows an overview of some of the obstacles in the environment (left) and a view generated by the simulator as the MAV's camera image (right). The setup of the experiments is to fly the MAV to a starting waypoint at which its height is forced to be 10 m. Subsequently it is ordered to move on a path through obstacles to a subsequent waypoint. After reaching this waypoint, the MAV is ordered to move to a height of 10 m again. Subsequently, it leaves for another waypoint on a path that goes through obstacles again, etc. Figure 9 shows two example way point configurations. The white crosses are the way points, the blue lines with arrows indicate the flight path and flying directions between the way points. The configuration on the left is a simple path in which the MAV has to go from A to B and back again. The path goes through multiple low buildings, the 14-story building of the faculty of architecture, and a structure consisting of four tall industrial chimneys. The configuration on the right has many more waypoints, forcing the MAV to fly over the obstacles from many different directions.

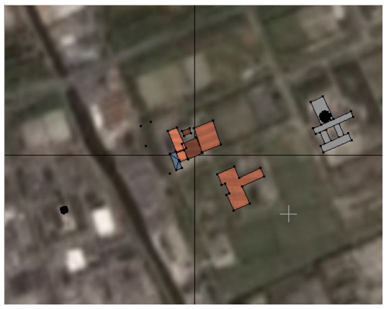
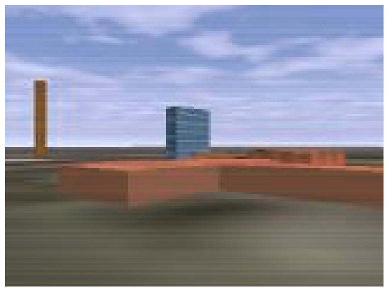

Figure 8: Left: Obstacles in the simulated environment around the faculty of aerospace engineering (the blue building). Right: View from onboard the simulated MAV.

During the experiment, the trajectory and height of the MAV are registered over time, as is the number of crashes. The performance of a strategy is determined by analyzing the number of crashes and the height of the

MAV. Both should be as low as possible, with a larger emphasis on a low number of crashes. For each strategy a control experiment is performed in which the MAV is flown at a height well above the highest obstacles (250 m). In this case, the MAV should not further ascend.

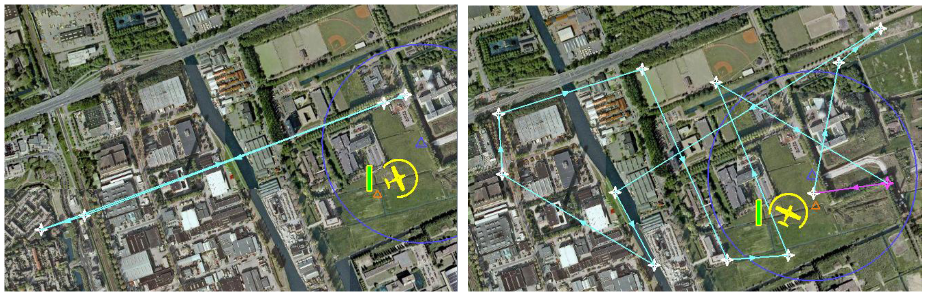

Figure 9: Two way point configurations. Way points are represented as white crosses, the flight paths by the blue lines. Left a single path through most obstacles, right a more complex patch in which obstacles are approached from various angles. Stars indicate that the MAV is forced to regain an altitude of 10 m.

4.2.2 Results

The MAV with access to its state estimate successfully avoids static obstacles. Generally, it ascends to a height slightly higher than the heights of the obstacles in its flight path. Figure 10 shows the trajectory of the MAV for the simple path (left) and the complex path (right).

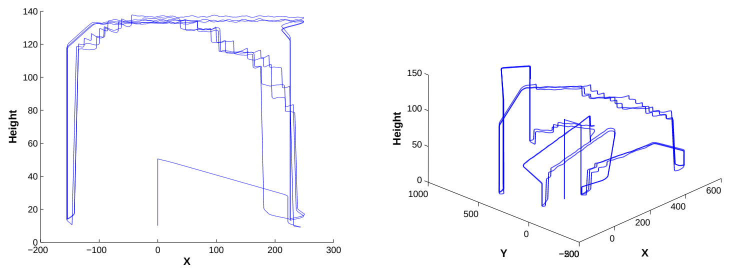

Figure 10: Trajectories of the MAV with access to its state estimate on the simple configuration (left) and on the complex configuration (right). Coordinates are given with respect to the local tangent plane. No crashes occurred.

No crashes occurred during the experiment. The MAV's height is adapted to the highest obstacle in its flight path. This can be best seen for the complex path, in which the highest altitudes are reached only when the chimneys are in sight. Please remark that the MAV detects obstacles at large distances and typically does not rise to the necessary height at once. This is because a far-away obstacle disappears under the horizon more and more until it is below noise level. If the MAV continues its path, the image surface of the obstacle grows again, until it supercedes the threshold. This explains the 'steps' in the trajectories. Furthermore, when it is tested on the control condition at 250 m (higher than all obstacles) it does not ascend further. Its average height is $\overline{h} = 249.3$ m with a standard deviation of $(\sigma(h) = 0.38)$.

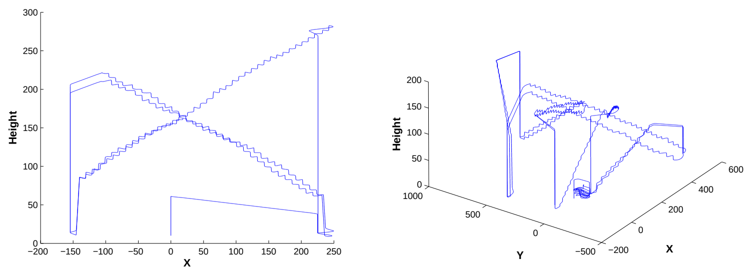

Figure 11: Trajectories of the MAV with access to its state estimate on the simple configuration (left) and on the complex configuration (right). Coordinates are given with respect to the local tangent plane. No crashes occurred.

With the same settings, the MAV without access to its state estimate does avoid obstacles, but ascends too much. It does not crash into any obstacles in either waypoint configuration, but ascends more than the MAV with access to its state (see Figure 12). This is confirmed by the control condition with the way points at 250 m: $\overline{h} = 305.6$ m, $\sigma(h) = 29.7$. The main problem for the MAV is the following. If the MAV detects an obstacle (either because there is one or because of noise), the MAV decelerates by pitching up. If it ascends far enough, the obstacle is below the horizon (or the noise has past) and the MAV accelerates back to 10 m / s. It does so by pitching down, leading to the perception of 'obstacles'. This iterative process leads to the MAV continuously ascending. One could set the thresholds for obstacle detection higher or the maximally allowed pitch and roll angles lower, but this results in a large number of crashes.

In summary, the concept of SSA is validated by the experiments in simulation. In the context of a rotary wing MAV, the use of the state estimates to project the horizon line is important for finding a good balance between avoiding obstacles and not ascending too much.

4.3 Fixed wing MAV

In this subsection a fixed wing MAV will be tested in simulation. The main reason for this is practical: the platform used for the real-world experiment is a fixed wing MAV.

4.3.1 Proposed strategy for a fixed wing MAV

The case of a fixed wing plane is less suited for SSA than the case of a rotary wing: a fixed wing will have to take into account its stall speed and be more actively concerned with its flight envelope protection. In a safety-first approach, the fixed wing MAV should evaluate the relative pitch angle to the obstacles ahead, θ_r. On the one hand, the relative pitch angle can be smaller than the maximal pitch angle of the plane $\theta_{\max}(\alpha)$, which is a function of the angle of attack α. In that case, it should start ascending according to a pitch angle $\theta > \theta_r$. Given a good segmentation, it is then certain to go over the obstacles. On the other hand, if $\theta_r > \theta_{\max}$, the MAV will have to deviate in order to avoid the obstacle. In that case, no guarantees can be given, since a single segmentation does not provide information on the distances to obstacles[5].

The strategy followed by the fixed wing MAV is identical to that of the rotary wing MAV, except that in this case the MAV does not slow down when it sees an obstacle (as in the real-world experiment). Furthermore, we remark that the real-world platform has only one-way communication between the camera and the autopilot; the segmentation and obstacle detection procedure do not have access to the autopilot's state estimates. Therefore, only the case without state estimates is investigated.

4.3.2 Results

The fixed wing MAV without access to its state estimates has much fewer false positives than the rotary wing without state estimates. The reason for this is the smaller perturbation in pitch angle. On the simple path configuration, the MAV has two crashes, and on the complex configuration

[5]The only distance cue available in sky detection alone is the variation of the height of obstacles over time, given that the MAV is oscillating in the height direction. Investigation of information over time is outside the scope of this paper.

one crash. Crashes are indicated by the red crosses. In all three cases, the plane could not avoid an obstacle that had a $\theta_r > \theta_{\max}(\alpha)$. Perhaps allowing the MAV to slow down as much as possible, might improve its performance. Still it seems that a deviation procedure is a necessary tool for making SSA work on fixed wing MAVs. That the fixed wing had fewer false positives than the MAV without access to the state estimates, is also confirmed by the control experiment at a height of 250 m: $\overline{h} = 252.3$, $(\sigma(h) = 0.70)$.

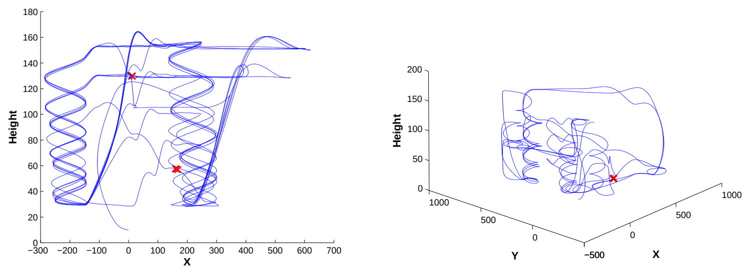

Figure 12: Trajectories of the MAV with access to its state estimate on the simple configuration (left) and on the complex configuration (right). Coordinates are given with respect to the local tangent plane. Crashes are indicated with red crosses (two separate crashes for the simple configuration and one for the complex configuration).

5 Preliminary experiment on real MAV

A preliminary experiment on a real MAV is performed to show the viability of SSA. In Subsection 5.1, the setup of the real-world experiment is discussed, and in Subsection 5.2 the corresponding results.

5.1 Setup real-world experiment

The left part of Figure 13 shows the fixed wing MAV with which the experiment is performed. It is a modified Easystar, equipped with a Paparazzi autopilot. On the nose, a Surveyor SRV-1 BlackFin camera is placed (right part of Figure 13).

The BlackFin camera has been modified so that it can communicate with a ground station via a 2.4 GHz Xbee communication module. Furthermore, the BlackFin camera have been connected to the autopilot via

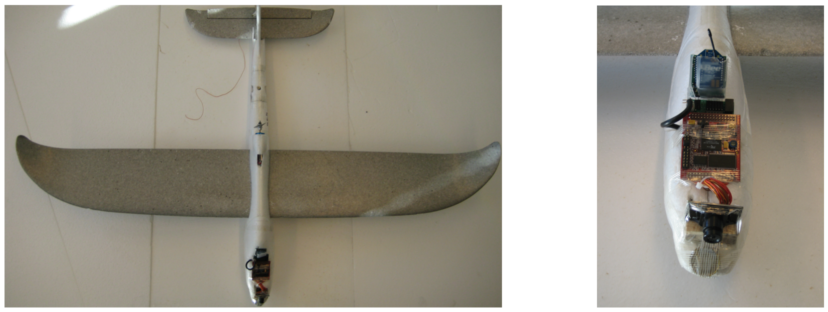
Figure 13: Left: Fixed wing MAV used for the real-world experiments. Right: modified Surveyor SRV-1 BlackFin camera mounted on the nose.

ADC-channels. In principle one could send many values over the channels by encoding them over time. However, if communication speed is essential, the setup with two ADC-channels implies that the BlackFin camera can communicate only two values to the autopilot via PWM. Currently, no communication is possible the other way around: the autopilot cannot send its pitch and roll estimates to the BlackFin camera.

During flight, the BlackFin camera continuously grabs an image and uses the $\mathcal{Y}\mathcal{C}b\mathcal{C}r$-tree shown in Figure 2 to segment it into sky- and non-sky regions. The execution frequency of the segmentation algorithm is ~ 30 Hz on the BlackFin. Since the camera has no state estimates, a pitch and roll of 0 degrees is assumed. The width of the image is divided in 10 bins, and per bin the obstacle pixels in the top half of the image are counted. The BlackFin camera sends two values to the autopilot: the maximal bin sum S_b and the total sum S_{all}. The SSA signals are in the range from 0 to 3.3 V, and the values are scaled so that they always fall in this range.

Concerning the autopilot, a module has been added to Paparazzi that receives the mentioned two values from the BlackFin camera, which in Paparazzi belong to the set $\{0, 1, 2, \ldots, 1024\}$. In the preliminary experiment, only the maximal S_b is used. If it exceeds a certain threshold *and* the roll and pitch angles are in the interval $[-5^\circ, 5^\circ]$, Paparazzi will augment the next waypoint with 0.5 m.

The experiment setup is straightforward: the MAV is ordered to descend to 30 m at the far end of a parc. Then it has to go to a waypoint at the start of the parc, where there is a group of trees. The MAV is ordered to descend to 0 m. Hence, without the obstacle avoidance method, the MAV would crash into the ground or the trees. In addition, a control experiment is performed at a height of 190 m. At that height, the MAV should not further ascend.

5.2 Results

The fixed wing MAV successfully employed SSA to avoid the trees in its flight path. Figure 14 shows a screenshot of the Paparazzi ground station of the test flight. The MAV has almost reached the first group of trees. Figure 15 shows the signal from the BlackFin camera (left). The black line indicates the threshold for augmenting the waypoint. The right part of Figure 15 shows the height of the waypoint over time.

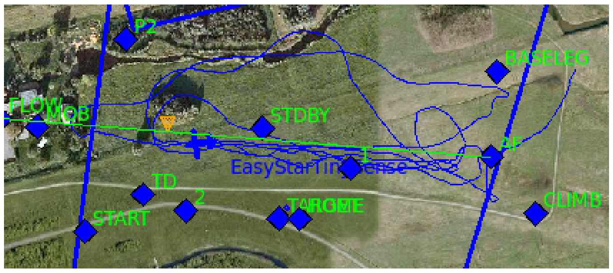

Figure 14: Screenshot of the Paparazzi ground station during the experiment.

As soon as the plane starts perceiving the trees as protrusions from the horizon, the way point starts rising. However, if the plane's altitude is still higher than the waypoint, the plane will continue descending. Therefore, the signal from the BlackFin typically increases initially. The plane starts ascending when the height of the waypoint is higher than the height of the plane. In the experiment, this happened before the plane reached the trees. Still, the experiment shows that it may be better to immediately set the next waypoint to the plane's height if obstacles are detected. In the control experiment, the MAV did not ascend further than 190 m, which confirms the validity of the results.

6 Discussion

In this article, the *Sky Segmentation Approach* to obstacle avoidance has been introduced. The current control algorithms use only the results of sky segmentation as an input for avoiding static obstacles. Here, we first discuss the limitations and potential of this approach. Then, we discuss the complementarity of sky segmentation with stereo vision and optic flow.

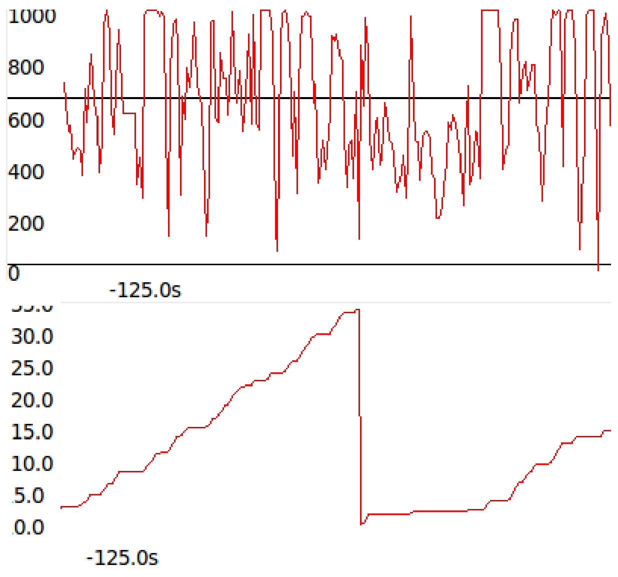

Figure 15: Left: Signal from the BlackFin over time, for two approaches of the trees. Right: Height of the waypoint over time for the two approaches.

The main limitation of SSA derives from the fact that segmentation alone does not convey any information other than the relative yaw directions to possible collision threats. The simulations show that a rotary wing using solely this cue is able to avoid most static obstacles by going over them. A fixed wing will require at least additional options, such as deviation. Even if it is likely that further study will improve the efficiency of SSA on the basis of sky segmentation alone, it will never be sufficient to achieve the optimal efficiency; the absence of distance information will in certain cases lead to unnecessary waypoint changes by the MAV. In its current form, SSA will therefore be a *safety-first* method. As a consequence, the approach using only the sky segmentation is not adequate for some applications / environments. In a truly large city such as New York or Beijing, the MAV is likely to rise to a height above most of the higher buildings, which leads to a ground resolution of a possible downward-looking camera that is too low for the intended application. Besides major cities, mountains can also be problematic when using sky segmentation alone.

The main potential of SSA derives from the fact that segmenting images into sky- and non-sky regions is a difficult, but tractable problem. Perfect segmentation is not necessary for successful obstacle avoidance. It is only necessary that the top of static obstacles leads to a number of obstacle pixels exceeding the used thresholds. Hence, it allows for an easily implementable, computationally and memory efficient strategy of reliably going over obstacles in the MAV's flight path.

Ultimately, we expect sky segmentation to be best combined with optic flow and / or stereo vision. Optic flow and stereo vision are cues that work best at relatively short distances. Sky segmentation works at much longer distances, allowing the MAV to plan a path around a tall obstacle such as the industrial chimneys in Figure 6 at an early stage. This saves the MAV from making abrupt maneuvers close to the chimneys. This will also help avoiding crashes in more cluttered environments. The work in [37] may form a starting point for the combination of sky appearance and optic flow.

7 Conclusions

We draw the following conclusions:

1. The *Sky Segmentation Approach* to obstacle avoidance (SSA) is a viable approach to obstacle avoidance. Simulation experiments show that a rotary wing MAV setup with two-way communication between the autopilot and the camera processor gives the best results. One-way communication from the camera to the autopilot leads to more false positives and fewer usable detections. Simulation experiments suggest that the negative effects of one-way communication are more detrimental for rotary wing MAVs than for fixed wing MAVs. This is due to the larger variations in pitch and roll for decelerating and accelerating for a rotary wing MAV. Finally, in a real-world experiment a fixed wing MAV successfully used SSA to avoid a group of trees.

2. The decision trees learned with the help of the labelME database compare favorably to the methods discussed in the literature. They have a higher precision than methods that employ an adaptive threshold, especially on images containing little sky. Although the decision trees typically perform slightly less well than methods such as [32], they are considerably faster, leading to an execution frequency of ~ 30 Hz on the Surveyor SRV-1 BlackFin camera.

Future work certainly includes more experiments in the real world, both for improving the image segmentation and for improving SSA control strategies. Different configurations should be tried out, such as employing fixed wing and rotary wing MAVs and having one-way or two-way communication between the camera processor and the autopilot. In addition, simulation experiments should be used to gain a more profound insight into the properties of these configurations and the results of various parameter settings. One important goal of simulation should be to determine the influence of estimation errors in the state estimates on the performance of SSA. Another should be to extend SSA to include deviating obstacles instead of only going over them. Moreover, SSA could be enhanced by combining the segmentation information with information coming from optic flow and / or stereo vision. Finally, SSA could be improved by focusing on an omnidirectional camera instead of a forward looking camera. This would further enhance the situation awareness of the MAV.

References

[1] R.W. Beard, D. Kingston, M. Quigley, D. Snyder, R.S. Christiansen, and W. Johnson. Autonomous vehicle technologies for small fixed-wing UAVs. *Journal of Aerospace Computing, Information, and Communication*, 2(1):92 – 108, 2005.

[2] K.P. Valavanis. *Advances in Unmanned Aerial Vehicles*. Springer, 2007.

[3] D.H. Shim, H. Chung, H.J. Kim, and S. Sastry. Autonomous exploration in unknown urban environments for unmanned aerial vehicles. In *AIAA GNC Conference, San Francisco*, 2005.

[4] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli. Flying fast and low among obstacles. In *Proceedings International Conference on Robotics and Automation*, 2007.

[5] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *EMAV, the Netherlands*, 2009.

[6] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *(ICRA 2009), Kobe, Japan*, 2009.

[7] R.K. Mehra, J. Byrne, and J. Boskovic. Flight testing of a fault-tolerant control and vision-based obstacle avoidance system for UAVs. In *Proceedings of the 2005 Association for Unmanned Vehicle Systems International (AUVSI) Conference, North America*, 2005.

[8] N. Franceschini, J.M. Pichon, C. Blanes, and J.M.Brady. From insect vision to robot vision. *Philosophical Transactions: Biological Sciences*, 337(1281):283–294, 1992.

[9] J.S. Humbert and M.A. Frye. Extracting behaviorally relevant retinal image motion cues via wide-field integration. In *American Control Conference*, number 14–16, page 6 pp, 2006.

[10] J. Serres, D. Dray, F. Ruffier, and N. Franceschini. A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance. *Autonomous Robotics*, 25:103 – 122, 2008.

[11] M. Hwangbo. Robust monocular vision-based navigation for a miniature fixed-wing aircraft. Ph.D. proposal, Robotics institute, Carnegie Mellon University, 2009.

[12] Antoine Beyeler, J.-C. Zufferey, and D. Floreano. Optipilot: control of take-off and landing using optic flow. In *European Micro Air Vehicle conference and competitions (EMAV 2009)*, 2009.

[13] H.A. Sedgwick. The visible horizon. Doctoral dissertation, Cornell University Library, 1973.

[14] J. J. Gibson. *The ecological approach to visual perception.* Houghton Mifflin, Boston, MA, 1979.

[15] T.G. McGee, R. Sengupta, and K. Hedrick. Obstacle detection for small autonomous aircraft using sky segmentation. In *ICRA 2005*, 2005.

[16] S.M. Ettinger, M.C. Nechyba, P.G. Ifju, and M. Waszak. Vision-guided flight stability and control for micro air vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems 2002 (IROS)*, volume 3, pages 2134 – 2140, 2002.

[17] S. Todorovic, M.C. Nechyba, and P. Ifju. Sky / ground modeling for autonomous MAV flight. In *IEEE International Conference on Robotics and Automation 2003 (ICRA)*, pages 1422–1427, 2003.

[18] T.D. Cornall and G.K. Egan. Measuring horizon angle from video on a small unmanned air vehicle. In *2nd International conference on autonomous robots and agents*, 2004.

[19] S. Fefilatyev, V. Smarodzinava, L.O. Hall, and D.B. Goldgof. Horizon detection using machine learning techniques. In *5th international conference on machine learning and applications (ICMLA'06)*, 2006.

[20] R. Carnie, R. Walker, and P. Corke. Image processing algorithms for UAV 'sense and avoid'. In *IEEE International Conference on Robotics and Automation 2006 (ICRA)*, pages 2848–2853, 2006.

[21] C. Rasmussen. Superpixel analysis for object detection and tracking with application to UAV imagery. In *3rd international conference on Advances in visual computing*, volume 1, pages 46–55, 2007.

[22] B. Zafarifar, H. Weda, and P.H.N. de With. Horizon detection based on sky-color and edge features. In W.A. Pearlman, J.W. Woods, and L. Lu, editors, *Visual Communications and Image Processing 2008 (SPIE)*, volume 6822, pages 1–9, 2008.

[23] S. Thurrowgood, D. Soccol, R.J.D. Moore, D. Bland, and M.V. Srinivasan. A vision based system for attitude estimation of UAVs. In *IEEE / RSJ International Conference on Intelligent Robots and Systems*, pages 5725–5730, 2009.

[24] I.F. Mondragón, M.A. Olivares-Méndez, P. Campoy, C. Martínez, and L. Mejias. Unmanned aerial vehicles UAVs attitude, height, motion estimateion and control using visual systems. *Autonomous Robots*, 29:17–34, 2010.

[25] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1–3):157–173, 2008.

[26] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition).* Morgan Kaufmann, 2005.

[27] J.R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[28] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[29] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[30] J.A. Noble. Finding corners. *Image and Vision Computing*, 6:121–128, 1988.

[31] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66, 1979.

[32] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In S. Ma and H.-Y. Shum, editors, *10th IEEE International Conference on Computer Vision (ICCV 2005), Beijing, China*, volume 1, pages 654–661, Washington, DC, 2005. IEEE Computer Society.

[33] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[34] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 2004.

[35] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[36] M.H.J. Amelink. *Ecological automation design, extending work domain analysis*. PhD thesis, Technical University of Delft, 2010.

[37] J. Byrne and C.J. Taylor. Expansion segmentation for visual collision detection and estimation. In *2009 IEEE International Conference on Robotics and Automation*, pages 875–882, 2009.