Edge-based Approach to Estimate the Drift of a Helicopter During Flight

Alexander Gatter

Institute of Flight Systems
German Aerospace Center, DLR, Lilienthalplatz 7
38108 Braunschweig, Germany

ABSTRACT

The Institute of Flight Systems at the German Aerospace Center (DLR) site in Braunschweig Germany has set its goal into making helicopter flying as safe as possible. The new DLR research project "Rettungshubschrauber 2030" addresses the topic of aiding helicopter rescue missions. Research will be conducted to increase the safety of these missions as well as to enable the conduct of missions in circumstances where nowadays a helicopter would not be allowed to operate. One aspect of this research is to increase or maintain the situational awareness of the pilot by processing data from camera images. The presented paper will focus on the field of visual odometry. Most of the publications on this topic use techniques that are only working with satisfying reliability in a very restricted environment, i.e. in good weather conditions. It shall be surveyed, if an edge-based approach for extracting features is a possible alternative or addition to established feature extractors. In the following paper, two algorithms for edge-extraction will be compared: An algorithm that is based on Hough transform and an algorithm that is based on the Douglas-Peucker-Method. They will be tested on their ability to detect a sufficient amount of features in camera images as well as on their computational complexity. Then, their ability to detect the drift of a helicopter will be surveyed on recorded data from flight tests with the Advanced Control Technology/Flying Helicopter Simulator (ACT/FHS) of the DLR. Their performance will be tested on the base of reference data from the ACT/FHS which have been recorded by the use of a highly accurate INS/DGPS system. Finally, a short outlook in form of a first comparison of well established feature extractors and the presented algorithms will be shown on a recorded scene with raindrops covering the lens of the camera.

Keywords: Computer Vision; Visual Odometry; Drift Estimation; Degraded Visual Environment; Helicopter Landing; IR

1. Introduction

The Advanced Control Technology/Flying Helicopter Simulator (ACT/FHS) is a highly modified EC 135 that is operated by the German Aerospace Center (DLR) in Braunschweig. It is equipped with a large set of sensors in order to enable the research of new technologies that improve the safety of helicopter flying. Some of these sensors are

- a commercial off-the-shelf visible light camera that operates at a rate of 25 Hz and has a resolution of 640×480 px,
- a thermal infrared bolometer camera from the company MaxViz Inc. that operates at a rate of 30 Hz,
- the H-74 ACE INS from Honeywell which is a highly precise coupled GPS/INS system, and
- a radar altimeter.

Figure 1 shows an image of the ACT/FHS. The combination of these sensors enables the research of new methods for visual odometry. The Institute of Flight

Figure 1: DLR's modified EC135: The ACT/FHS

Systems of the DLR currently pursues the research project “Rettungshubschrauber 2030” (Rescue Helicopter 2030). One goal of this project is to improve the safety of helicopters in rescue missions, especially in adverse operating situations. To this purpose, a vision-based approach shall be developed that enables the compensation of possible GPS failures. Many publications exist that substitute GPS data with visual data like for example in [1] and [2]. These approaches focus mainly on the localization of the helicopter. The presented algorithm is designed to detect high lateral velocities in landing approaches of helicopters. Many accidents occur because a helicopter has an exceeding (often lateral) movement speed when touching the surface [3]. This can cause the landing gear of the helicopter to get stuck on an obstacle on the surface. This changes its lateral movement to an angular movement with the obstacle being its angle point. A so-called “dynamic rollover” results from that if the pilot does not manage to counter-react in time. In many cases the pilot can estimate the speed of the helicopter by observing the landing zone and the relative movement of the helicopter to it. In degraded visual environment (DVE) situations, this task gets more difficult to perform. From that, the need arises to provide an alternative way of estimating the speed of the helicopter. Most feature extractors that are nowadays n use are designed to operate in a non-disturbed environment. The edge-based approach that will be presented in this paper tries to give an alternative to commonly used feature extractors which is more robust in certain types of DVE scenarios. Two different edge extractors have been implemented and tested. Their working principles as well as their performances will be presented in this paper.

2. Pre-processing

This section gives an outline of the two processing steps that are conducted in order to enable the later extraction of the edges. These steps consist of the elimination of distortion effects and the projection of images into an orthographic view of the scene.

2.1. Distortion

Distortion effects would cause edges to have a bent appearance. This would lead to the loss of many possible edge segments for further processing. The camera that is used by the ACT/FHS shows strong barrel distortion effects. Because of that, the camera images are rectified by using functionalities of the OpenCV library [4] which base on the algorithm of Zhang [5]. Figure 2 shows the difference between an unmodified camera image with the horizon and the helipad appearing bent and an image that has been undistorted.

(a) Unmodified image

(b) Undistorted image

Figure 2: Visualization of the effect of distortion

2.2. Image Transformation

The camera is mounted forward-looking and is tilted by approximately 20 degrees. Due to this setup, the appearance of the edges is dependent on their position in the image. This means, that an edge, which would appear vertical in the middle of the image, would appear tilted when being located near the right or left border of the image. In case of a lateral movement of the camera, this would lead to a rotation of the edge over time. Algorithmically subtracting out this effect is possible. However, it is more convenient to transform the camera image in a fashion that prevents this behavior. This is done by taking a plane earth assumption and by then calculating a homography of the image over the tilt angle of the camera. A plane earth assumption presumes that the ground that is observed by the camera is flat. With this assumption and the knowledge of the orientation and the altitude of the camera, it is possible to assign a depth value to every pixel in the camera image. This is needed to be able to measure the absolute speed of the camera.

Now, all pixels of the image are projected into camera coordinate system with

$$X = Z\frac{x - c_x}{f_x} \text{ and} \tag{1}$$
$$Y = Z\frac{y - c_y}{f_y}$$

where X and Y are the coordinates of a pixel in helicopter-space. Z is the estimated distance of this point to the camera, based on the plane earth assumption and the knowledge of the altitude of the helicopter. x and y are the coordinates of a pixel in camera-space. c_x and c_y represent the center of the image. f_x and f_y are the focal lengths of the camera. To achieve an orthogonal representation of the camera image, a rotation of the projected points is then carried out via

$$R_1 = \begin{pmatrix} \cos^2\phi\cos\theta' + \cos\theta & \cos\phi\sin\theta\cos\theta' & \sin\phi\sin\theta \\ \cos\phi\sin\phi\cos\theta' & sin^2\cos\theta' + \cos\theta & -\cos\phi\sin\theta \\ -\sin\phi\sin\theta & \cos\phi\sin\theta & \cos\theta \end{pmatrix}. \tag{2}$$

In this formula, ϕ stands for the roll angle of the helicopter. θ stands for the pitch angle. $\cos\theta'$ represents $\cos(1-\theta)$. The estimation of the movement of the edges can be simplified even further by regarding orientation changes in the roll axis of the helicopter. This is done by applying a rotation

$$R_2 = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

to the result of the earlier transformation. Next, a rotation is applied to correct the mounting angle of the camera and additionally adapt the pixels to camera coordinate system:

$$R_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Delta\theta & -\sin\Delta\theta \\ 0 & \sin\Delta\theta & \cos\Delta\theta \end{pmatrix}. \tag{4}$$

Here, $\Delta\theta$ stands for the angle between the orientation of the camera and the longitudinal axis of the helicopter. The application of all of these transformations

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \cdot R_1 \cdot R_2 \cdot R_3 \tag{5}$$

results in the transformed points X', Y', and Z' in helicopter coordinate system. Finally, these transformed points are projected back into the image coordinate system by calculating

$$x' = f_x\frac{X'}{Z'} + c_x \text{ and} \tag{6}$$
$$y' = f_y\frac{Y'}{Z'} + c_y \tag{7}$$

where x' and y' are the projections of x and y in camera space. The application of all these transformations ensures that only changes in heading and movement of the helicopter cause shifts in the image. To provide that the projection results in a coherent image without holes, occurring gaps are filled by use of bi-linear interpolation. A comparison of a non-modified image and an image that has been transformed by the application of the above mentioned formulas is presented in Figure 3.

(a) Original camera image

(b) Transformed image

Figure 3: Image of a helipad before and after its transformation

3. Edge Extraction

Two different edge extractors have been tested on their applicability on detecting the drift of the helicopter: The Hough transform [6] and the Douglas-

Peucker-Algorithm [7]. In the current section, all processing steps will be treated, that are necessary to receive a set of lines that is suitable for estimating the drift of the helicopter. Both algorithms return the lines via their starting points x_a, x_b and end points y_a, y_b. In addition, the central point (x, y), the angle according to the x-axis Φ, and the length l of the lines is stored for further processing. It is

$$
\begin{aligned}
x &= \frac{x_a + x_b}{2}, \\
y &= \frac{y_a + y_b}{2}, \\
\Phi &= \arctan \frac{y_b - y_a}{x_b - x_a}, \\
l &= \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}.
\end{aligned} \tag{8}
$$

3.1. Hough Transform

Tests on camera images that have been recorded during flight tests have shown that the unmodified Hough transform tends to produce a large number of falsely detected edges. Since the Hough transform is computationally expensive, these false positives significantly increased the amount of computation time. In addition, the drift estimation of the helicopter was often heavily disturbed by the false positives. This resulted in strong aberrations of the estimated drift from the reference drift that has been provided by the GPS/INS of the helicopter. Because of that, several modifications have been applied to the native Hough transform.

In order to maintain a large amount of extracted edges while reducing miss-classifications, an orientation map is created that contains the dominant orientation of the gradient of each pixel. Based on [8], these orientations are calculated by applying Gabor filters which detect contrast orientations of a set amount of orientations. By use of this orientation map, an anisotropic filter is applied to the original image [9]. This is done in order to reduce noise and maintain the sharpness of potential edges. A gradient image of the anisotropic filtered image is then created via convolution with a Sobel filter. Then, a Canny algorithm [10] is applied in order to improve the robustness of the following feature extraction. This algorithm determines the most promising edge estimations which resulted from the Sobel filtering. The Hough transform is now conducted on the computed set of edges. Every potential line that was detected by the Hough transform is crosschecked pixel-wise with its corresponding entries in the orientation map. If the orientations of the gradients of these pixels differ more than a set value, the potential line is discarded. If the orientation difference is below that threshold and its length is above a set minimum, the line is stored for further processing.

3.2. Douglas-Peucker-Method

The second tested algorithm is the Douglas-Peucker-Method which approximates a given line segment with a reduced set of line segments. This algorithm tends to work considerably faster than the previously presented algorithm, but it turned out that it produces a smaller amount of potential lines than the Hough transform. In order to increase the set of potential lines, an unsharp masking has been conducted. For this purpose, the original image is convolved with a low pass filter. The resulting smoothed image is then weighted and subtracted from the original image. This is done with

$$
I'(x, y) = 1.5 \cdot I(x, y) - 0.5 \cdot U(x, y) \tag{9}
$$

where I and U denote the original image and its low-pass filtered unsharp mask, I' denotes the resulting image, and (x, y) denotes the image coordinates. Equivalent to section 3.1, a Sobel filter is then applied to calculate the gradients of the image. The result is thinned out with use of the Canny algorithm. Unlike the Hough transform, the Douglas-Peucker-Method needs a set of coherent potential lines to work with. The coherence information is computed with an modified version of [11]. Unlike the original implementation, the used method does not try to compute a contour which starts and ends with the same pixel by running adjacent to the contour. It rather directly searches on top of a line until it reaches an end and then stores the result as a coherent segment. In line crossings, the method prefers a straight oriented continuation of a line to a continuation that would produce a sharp angle. Without this, a line crossing would produce "x"-like structures instead of the desired two intersecting lines. Another alteration to the original principle is, that pixels which already have been used for creating a coherent line segment cannot be used for starting a new line segment, yet they can be used for continuing a line. The final change to the original principle is, that a pixel can only be the starting point of a line extraction, if it has only adjacent pixels that are located in one vicinity to the pixel (i.e. top, left, right, and bottom). This constraint prevents that the line finding algorithm can start in the middle of a potential line instead at its ends. A visualization of the differences to [11] is presented in Figure 4.

An exemplary image of a scene, with all found lines of a time step being visualized, is presented in Figure 5.

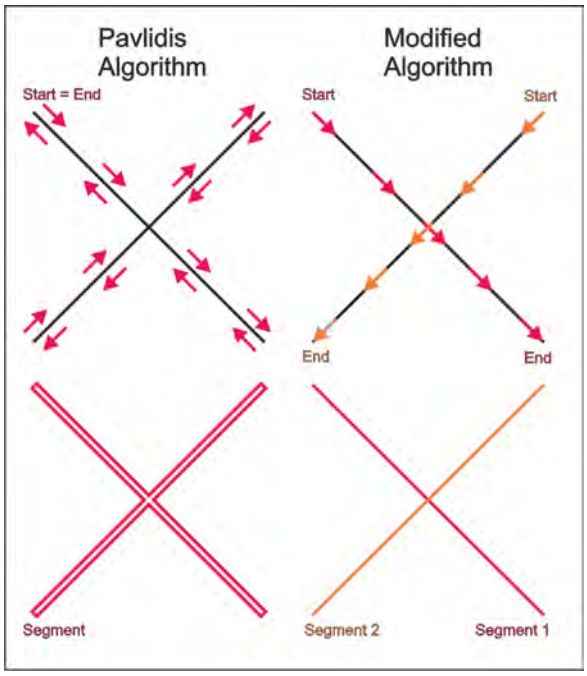


Figure 4: Difference between the algorithm of Pavlidis and its modification on the example of two intersecting lines

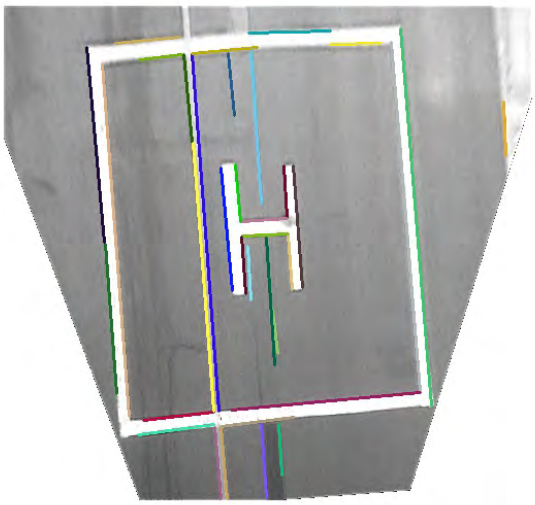


Figure 5: Orthographic view of a helipad. Lines that have been found by edge extraction are visualized

4. Edge Tracking

In order to be able to estimate the drift of the helicopter, the found lines have to be tracked reliably over time. The tracking algorithm uses a prediction of the position of a line in the current time step t by estimating the position change in $t-1 \rightarrow t$ by using the current velocity estimation as well as the changing of the heading angle $\Delta\Psi$ and the changing of the height of the helicopter Δh. Velocity data of the INS is used, if no velocity information is available yet (i.e. at the beginning of the algorithm).

The displacement of the helicopter position is calculated by computing the intersections of the found lines and comparing the changes in position of the resulting intersection points over time. All the formulas that are used in this chapter are working in the image space that is introduced in section 2. This means that the image coordinate system is set up in a way so that it lies on a plane which is parallel to the helicopter fuselage with the camera looking downwards onto that plane. That way, the position of the helicopter (neglecting altitude) can be regarded as a point in this coordinate system. The intersection (x_s, y_s) of lines m_1 and m_2,

$$m_1 = \frac{y_{b1} - y_{a1}}{x_{b1} - x_{a1}}, \tag{10}$$

$$m_2 = \frac{y_{b2} - y_{a2}}{x_{b2} - x_{a2}}, \tag{11}$$

is calculated with

$$x_s = \frac{y_{a1} - y_{a1} - m_1 x_{a1} + m_2 x_{a2}}{m_2 - m_1} \text{ and} \tag{12}$$

$$y_s = y_{a1} + m_1(x_s - x_{a1}) \tag{13}$$

where $x_{a1,b1}$, $y_{a1,b1}$, $x_{a2,b2}$, and $y_{a2,b2}$ denote the starting and the ending points of two line segments in the image coordinate system. The next step is to convey the linear speed $v_{x,y}$ of the helicopter into pixel-space:

$$v_x = \frac{v_x}{h/f_x}, \tag{14}$$

$$v_y = \frac{v_y}{h/f_y}. \tag{15}$$

The altitude of the helicopter is h. The focal length of the camera is given by $f_{x,y}$. Additionally, the image point (x_m, y_m) that represents the point that is directly beneath the camera in the world coordinate system has to be moved to the image center (c_x, c_y). This is done by subtracting (x_m, y_m) from every point of the image.

The next step is to calculate the movement of the segments that is caused by changes of the altitude of the camera:

$$x_t = x_{t-1}\frac{h_{t-1}}{h_t}, \tag{16}$$

$$y_t = y_{t-1}\frac{h_{t-1}}{h_t}. \tag{17}$$

These formulas however, only cover linear movement. Most of the times, movement consists of a mixture of translational and rotational movements. Changes in the orientation Ψ result in displacements of the intersection points of the lines that have to be subtracted out. The separation of both kinds of movement and the final extraction of the aspired translational movement will be treated in the following formulas. The

radius r of a curve on which the camera is moving can be calculated with

$$r = \frac{\Delta t \sqrt{v_x^2 + v_y^2}}{\Delta \Psi}. \tag{18}$$

Since the orientation of the camera does not necessarily have to be identical with its movement direction, the angular difference between these two orientations has to be regarded with

$$\beta = atan2 \frac{v_x}{v_y}. \tag{19}$$

In this formula, β is the angle between the orientation of the camera and its movement direction. Also, the displacement between helicopter orientation and camera orientation has to be regarded. However, the camera was mounted with the same orientation as the helicopter's longitudinal axis in the current setup. Therefore, this calculation is neglected. Finally, the displacement (m_x, m_y) that results from angular movement can be calculated by the following formula:

$$m_x = r \cdot \cos \beta, \tag{20}$$

$$m_y = r \cdot \sin \beta. \tag{21}$$

Summing it all up, the prediction for the displacement (x_t, y_t) of the lines can be calculated via

$$x_t = (x_t - m_x) \cdot \cos \Delta\psi + (y_t - m_y) \cdot \sin \Delta\psi + m_x \tag{22}$$

and

$$y_t = (x_t - m_x) \cdot \sin \Delta\psi + (y_t - m_y) \cdot \cos \Delta\psi + m_y. \tag{23}$$

For the actual tracking of the line segments, their difference d_Φ in terms of orientation is first computed with

$$d_\Phi = |(\Phi_{t-1} - \Phi t)|. \tag{24}$$

In this formula, Φ stands for the angle of a line in reference to the x-axis of an image. The distance (d_x, d_y) of both segments is then calculated, if d_Φ is sufficiently small with

$$d_x = |(x_t - x_{t-1}) - (y_t - y_{t-1}) \cdot \tan(\frac{\pi}{2} - \Phi_{t-1})| \text{ and} \tag{25}$$

$$d_y = |(x_t - x_{t-1}) \cdot \tan \Phi_{t-1} - (y_t - y_{t-1})|. \tag{26}$$

The coordinate of the center of a line at time step t is represented with (x_t, y_t). Accordingly, (x_{t-1}, y_{t-1}) is the coordinate of the center of a line at time step $t-1$. If all the computed distances are below the set limits, the parameters of the line are stored (see (9)). Together with any lines that have been found in previous time-steps, these lines are the basis for the further movement estimation.

In the next step, the errors have to be eliminated which result from movements of the image that are not caused by a translational movement of the camera. Influences from rotary movement and camera setting are already eliminated by the image transformation in 2.2. Altitude changes of the camera have no negative influence because the found segments are projected back onto the object space, which lies parallel to the transformed image space. The distance between these two spaces is given by

$$x = \frac{h}{f_x}(x - x_m) \text{ and} \tag{27}$$

$$y = \frac{h}{f_y}(y - y_m). \tag{28}$$

Due to the parallel setup of both planes, changes in the height of the camera do not cause a displacement of the segments. Next, the error has to be estimated that is caused by changes of the orientations of the segments. Therefore, two error values e_1 and e_2 are calculated. For better understanding, a sketch of these two errors is given in Figure 6. x and y represent the image coordinates. Both depicted triangles consist of two intersecting segments a and b. The third element of these triangles is the horizontal distance d_x between the angular point of the segment which changes and the constant segment. Since the angular point of this segment is not known, an approximation is taken by using the midpoint of the changing segment. This assumption can be taken due to the fact that the segment has obviously been tracked, the angular point must be close to the midpoint. The error e_1 results from a change of the first segment b by the angle $\Delta\alpha$, with the second segment a being constant. The error e_2 results from an angular movement $\Delta\beta$ of the segment a with b being constant. The angles α and β can be calculated by use of the stored angles of the segments. With this knowledge, both error values can then be calculated with simple mathematical formulas. To compensate for the found error $e = e_1 + e_2$, e is subtracted from the measured displacement of the segments.

5. Movement Estimation

For the final estimation of the movement speed, first the length of the flown arc has to be determined. This length is computed by the length of the displacement of the intersections Δl as well as the intersections of the circles on which the intersections rotate. Figure 7 (a) depicts how to get to the angular point (m_x, m_y), around which the helicopter is turning and

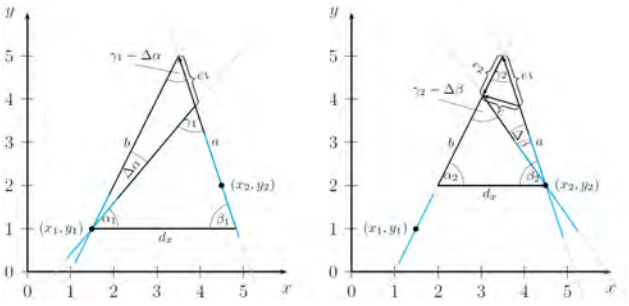


Figure 6: Visualization of the error elimination of angular movement

the length r_s of the arc that is spanned by the displacement of the intersection (x_s, y_s) at the time steps 1 and 2 and the angular point. The coordinate system is centered at the position of the helicopter with x and y being the axes which go through the longitudinal and the lateral axis of the helicopter. The altitude of the helicopter is neglected. r_s can be calculated by

$$r_s = \sqrt{\frac{\Delta l^2}{2 \cdot (1 - \cos \Delta\psi)}}. \tag{29}$$

The radius of the arc that the helicopter is flying can be calculated by

$$r = \sqrt{m_x^2 + m_y^2}. \tag{30}$$

With this radius, the absolute value of the speed of the helicopter can be calculated by

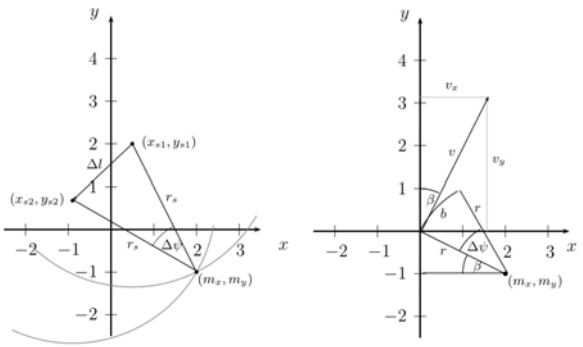
$$v = \frac{r \cdot \Delta\psi}{dt}. \tag{31}$$

v_x and v_y can be calculated by

$$v_x = v \cdot \sin \beta \text{ and} \tag{32}$$

$$v_y = v \cdot \cos \beta \tag{33}$$

with v_x representing the lateral speed of the helicopter and v_y representing the directional speed of the helicopter. Figure 7 (b) depicts the process of calculating v_x and v_y with the coordinate system being identical to the coordinate system in Figure 7 (a). This calculation is repeated for all intersections in a scene. Those are then inserted into a coordinate system with the flight course on the x-axis and v on the y-axis. The final movement estimation is then calculated with a RANSAC approach. This prevents strong outliers that can result from violations of the plane earth assumption to have an influence on the estimation.



(a) Calculation of (m_x, m_y) (b) Calculation of (v_x, v_y)

Figure 7: Movement speed estimation

6. Tests

The aim of the conducted tests was to evaluate if the presented feature extraction and feature tracking algorithms are able to measure the ground speed of a helicopter with sufficient precision. A formulation of the maximal tolerable error in the estimation of the movement speed is a nontrivial task. The maximal error depends on the maximal drift speed, with which a helicopter can land without being endangered of causing a dynamic rollover. And this speed depends on the surface on which the helicopter wants to land. Pilot surveys have yielded a maximal tolerable drift speed of 1 m/s when landing on mostly plane landing places, like meadows. On rocky surfaces, pilots tend to tolerate a drift speed up to only 0.5 m/s. Consequently, these two values present the demand on the needed precision of the tested algorithms. To be able to provide a landing aid on arbitrary surfaces, a maximal error of 0.5 m/s is aspired. The drift speed that is calculated by the presented feature tracking algorithms is averaged over a time of 3 s to account for a noisy short-time behavior of the features. This is acceptable, since only the slowly increasing error of the INS has to be detected. High frequency changes in the helicopter speed can still be detected by an INS. Additionally, discretization effects are lessened with this procedure.

The algorithms have been tested on a synthetic scene as well as on real flight data. In the following, the test results on the synthetic scene, as well as test results of several flight tests, are presented. Results will be shown for the lateral drift only. The magnitude of error for directional drift speed is similar to the magnitude of error in lateral drift speed. All tests have been conducted on a Windows PC with a quad-core i5-2500 processor that has a clock speed of 3.30 GHz (without utilization of multiprocessing) and 8 gigabyte working memory.

6.1. Scenario 1: Synthetic Data

A synthetic scene has been created to evaluate which results can be achieved in ideal environments. The surface in this scene is completely plain and consists of an endless grid. An image of this test scene is shown in Figure 8. The results of both algorithms are shown in Figure 9.

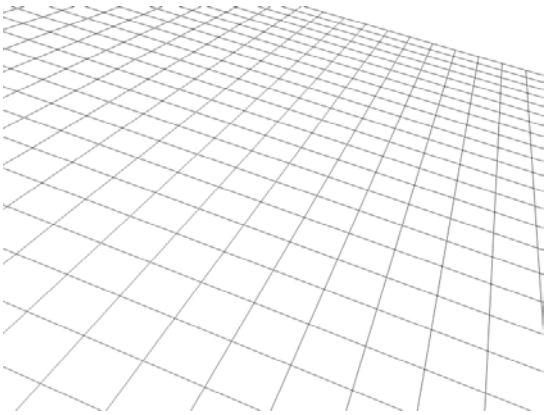
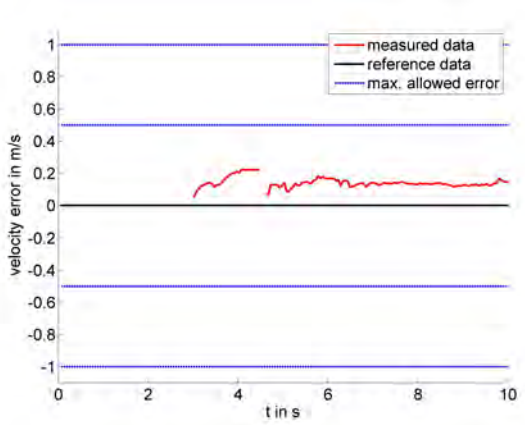
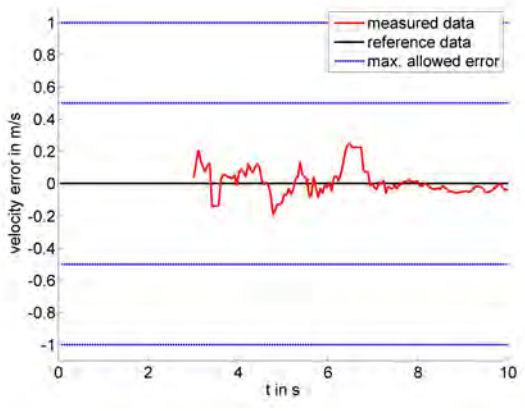


Figure 8: Recorded image of the setting in scenario 1: Synthetic data



(a) Computed difference with Hough



(b) Computed difference with Douglas-Peucker

Figure 9: Evaluation of test scenario 1: Synthetic data

The parameters of the synthetic camera are known. In the evaluated test, the camera performs a constant curved motion with a known height over this grid. During this motion, the altitude of the camera decreases linearly from a height of 40 m over the grid down to a height of 10 m. The test has a duration of 10 s. In figure 9, the estimated difference between reference data on the x-axis and measured data from the Hough-based algorithm and from the Douglas-Peucker-based algorithm is presented. It can be observed, that on these synthetic data, the results are significantly below the set limit of 0.5 m/s. This implies, that both algorithms are able to estimate the drift of a helicopter with sufficient accuracy to be used for preventing dynamic rollovers. A difference of more than 0.2 m/s between reference data and measured data is trespassed only occasionally and then just for a short time. The drift speed estimation of the Hough-based algorithm shows a very calm behavior which seems to jitter around an offset of approximately 1 m/s to the reference data. The average computing time for a frame with the Douglas-Peucker-based algorithm was 87 ms. The Hough-based algorithm needed 501 ms.

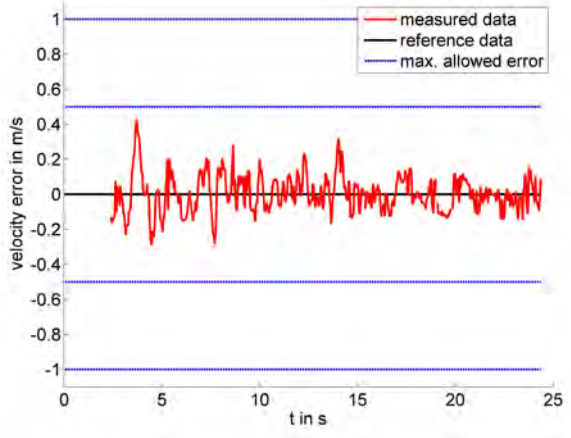
6.2. Flight Test Data

The algorithms have been tested on several flights with a set of different undergrounds. Also, one test will be shown that has been conducted on infrared data. The tests have been conducted with the ACT/FHS. Precise GPS/INS data have been used as reference data.

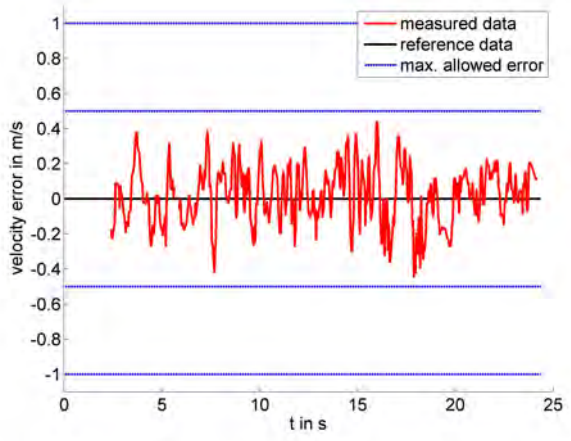
6.2.1. Scenario 2: Low-Level Lateral Flight

In the first presented flight test, the helicopter performs small translational movements over a landing zone that is marked with an "H" that is surrounded with a white rectangular border. At least one border of the rectangle is always visible during the whole test. The evaluated part of the flight has a duration of approximately 25 s and the helicopter is flying at an altitude of 3 m. According to the reference data, the directional speed is around zero and the lateral speed ranges from 0 m/s to ±0.7 m/s. An image of this test scene is shown in Figure 10. The results of both algorithms are shown in Figure 11. In this figure, the estimated difference between reference data on the x-axis and measured data from the Hough-based algorithm and from the Douglas-Peucker-based algorithm is presented. As it can be seen, the errors in both algorithms stay below the 0.5 m/s demand. The Hough-based algorithm shows better results in this test. Over the complete test, this approach yields an error of more than 0.4 m/s only once, while the Douglas-Peucker-based approach breached that limit several times. The

Figure 10: Recorded image of the setting in scenario 2: Low-level lateral flight



(a) Computed difference with Hough



(b) Computed difference with Douglas-Peucker

Figure 11: Evaluation of test scenario 2: Low-level lateral flight

average computing time for a frame with the Douglas-Peucker-based algorithm was 87 ms. The Hough-based algorithm needed 469 ms.

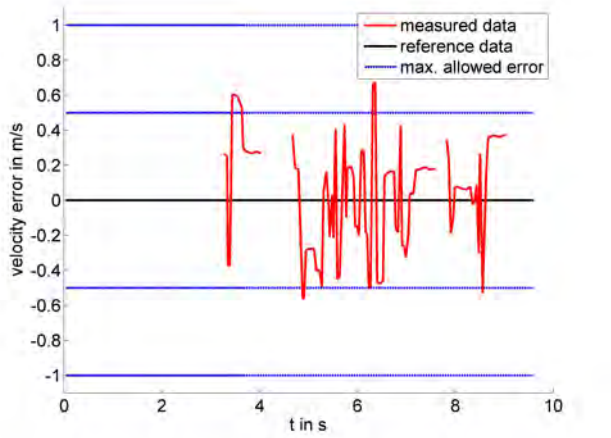
6.2.2. Scenario 3a: Urban Setting - Visible Light Camera

The next scenario is placed in an urban setting. Man-made objects build an ideal basis for the presented algorithms because they mainly consist of straight edges. Therefore, the amount of found edges is to be expected very high and the tracking of these edges should be easy. On the other side it is possible that the altimeter of the helicopter returns erroneous altitude measurements when flying over buildings. In the current scenario, the helicopter flies over the city of Wolfsburg at an altitude between 130 m and 160 m with a directional speed of approximately 26 m/s and a lateral speed of around zero. The evaluated test scene has a duration of approximately 9 s. After that time period, the helicopter flies over a cloud of smoke emitted by a large factory chimney where further speed estimations cannot be conducted anymore. An image of this test scene is shown in Figure 12.

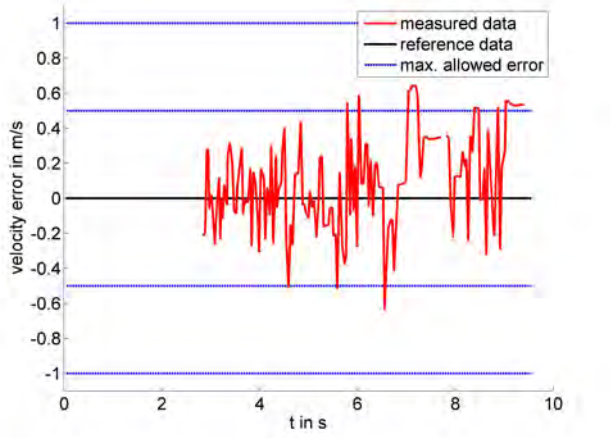
Figure 12: Recorded image of the setting in scenario 3a: Urban setting - Visible light camera

The results of both algorithms are shown in Figure 13. In this figure, the estimated difference between reference data on the x-axis and measured data from the Hough-based algorithm and from the Douglas-Peucker-based algorithm is presented. As it can be seen, the set limit of 0.5 m/s is slightly trespassed several times with both algorithms. However, there are several reasons why this result can still be regarded as satisfying. First, the helicopter flies at an altitude of around 150 m which increases the problem of discretization by a large amount. Second, the plane earth assumption is violated significantly due to the fact, that the setting consists of many houses, some of them several floors high. The main goal of the presented approach is to detect the drift speed in low altitudes and mostly plain surfaces. This test in an urban setting was performed in order to draw conclusions about the ability of the presented algorithms to work in scenarios other than close to the ground directly before

the touchdown. The presented test shows that, even when these two demands on the settings are breached, a nearly satisfying estimation can still be achieved.



(a) Computed difference with Hough



(b) Computed difference with Douglas-Peucker

Figure 13: Evaluation of test scenario 3a: Urban setting - Visible light camera

6.2.3. Scenario 3b: Urban Setting - Infrared Camera

Scenario 3b is basically the same setting as scenario 3a. This time however, the images are recorded with the infrared camera of the helicopter. This aggravates the extraction of features considerably, since infrared images show weaker contrasts than images that have been recorded by visible light cameras. However, the use of an infrared camera potentially extends the applicability of the vision based drift estimation to operations in the night. The infrared pendant to the presented figure in scenario 3a is shown in Figure 14. The results of both algorithms are shown in Figure 15.

In this figure, the estimated difference between reference data on the x-axis and measured data from

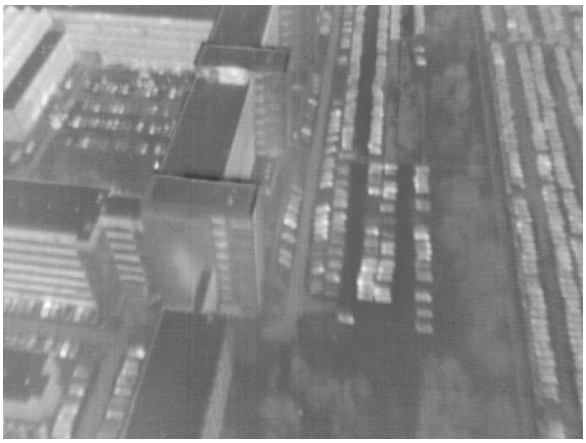
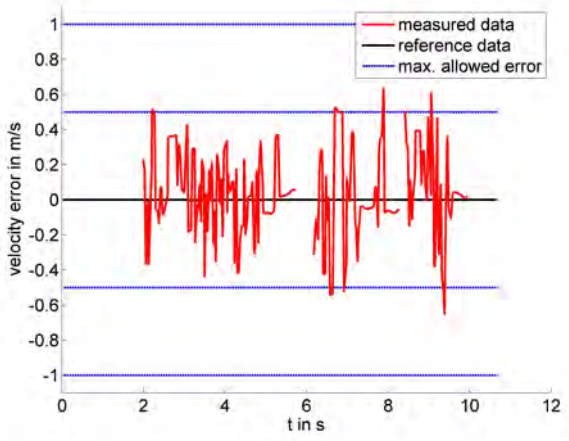
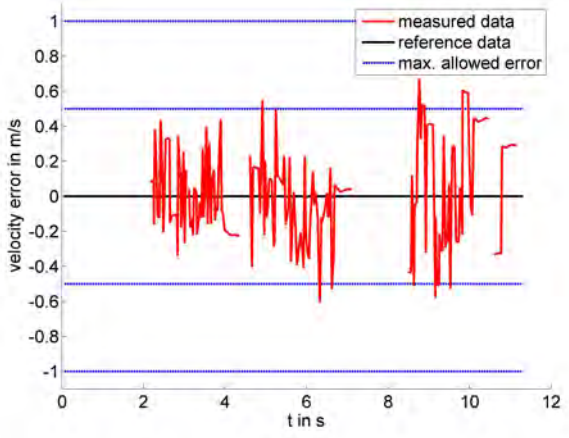


Figure 14: Recorded image of the setting in scenario 3b: Urban setting - Infrared camera



(a) Computed difference with Hough



(b) Computed difference with Douglas-Peucker

Figure 15: Evaluation of test scenario 3b: Urban setting - Infrared camera

the Hough-based algorithm and from the Douglas-Peucker-based algorithm is presented. The results are very similar to the results of scenario 3a. Con-

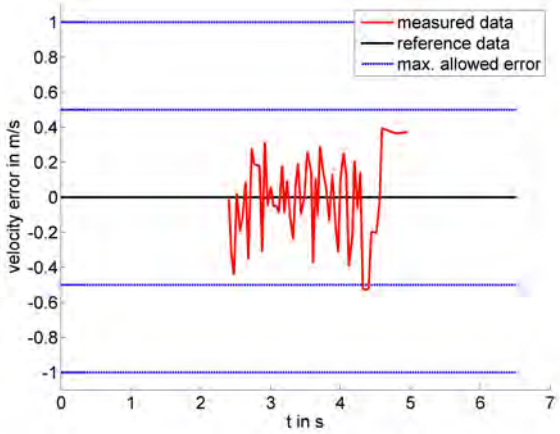
sequently, it can be assumed, that the algorithms are able to work with IR images of sufficient quality. However, it must be stated that the algorithms have been applied on several scenarios that have been recorded with IR camera. On most of these, no sufficient amount of lines could be extracted. Two potential reasons for this have been identified. First, because the used IR camera cannot compete with nowadays high-end IR cameras. Second, in most if the test settings the helicopter was flying very close to the ground. Often, there were not enough differences in the heat of the surface to be able to extract features (e.g. when hovering over a landing pad).

6.2.4. Scenario 4: Flight over Low-Contrasted Landscape

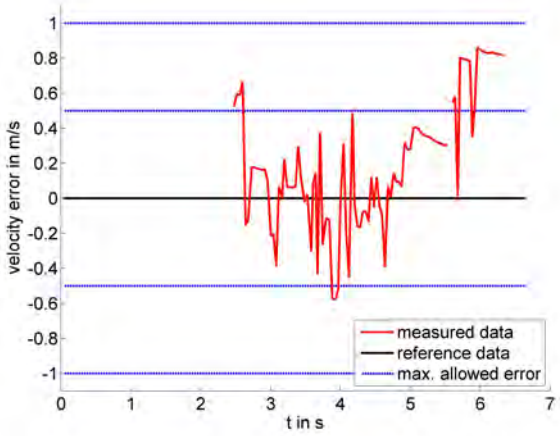
The last test shows a scenario where the algorithms reach their limits. The helicopter flies at an altitude of 50 m with a longitudinal speed of 36 m/s, approaching the landing field of the airport of Braunschweig. The surface below the helicopter consists of several fields and a road. The contrasts in the scene are relatively smooth and most of the contrasts are curved. The test has a duration of approximately 10 s. An image of this test scene is shown in Figure 16.

Figure 16: Recorded image of the setting in scenario 4

The results of both algorithms are shown in Figure 17. In this figure, the estimated difference between reference data on the x-axis and measured data from the Hough-based algorithm and from the Douglas-Peucker-based algorithm is presented. Near the end of the evaluated time-span, the Douglas-Peucker-based algorithm reaches a critical error of up to 0.8 m/s. Only then it finally stops giving an estimation of the drift speed. The Hough-based algorithm shows a better behavior. Here, the limit of 0.5 m/s is slightly breached in only a few consecutive time steps. Unlike the other method, it stops to give drift estimations nearly two seconds earlier, potentially preventing a bad estimation of nearly 1 m/s. Both algorithms however are not able to reestablish their drift estimation until the landing field with its sharp contours fills out a considerable amount of the image (which is not included in the presented scenario). The average computing time for a frame with the Douglas-Peucker-based algorithm was 88 ms. The Hough-based algorithm needed 529 ms.



(a) Computed difference with Hough



(b) Computed difference with Douglas-Peucker

Figure 17: Evaluation of test scenario 4

7. Conclusion

In this paper, two alternative feature extraction algorithms have been presented and compared. Additionally, a method to estimate the drift of a helicopter has been presented. The developed algorithms are intended to aid popular feature extraction algorithms. Also, they can be used to increase the amount of features of any visual odometry in general and to calculate redundant movement estimations. Tests have shown that both algorithms are able to extract and

track lines in an image. Also, the advantages and disadvantages of the presented algorithm to estimate the movement speed of a helicopter have been discussed. It has been shown, that the maximal velocity errors of the algorithms are below the set limits when flying over flat ground with a sufficient amount of edge features. Also, it has been shown, that both algorithms can work on infrared images up to a certain extent. In general, the algorithms can be of use in any setting that shows a certain amount of man-made environment, which can often be found in populous regions. The algorithms are stretched to their limits, when flying over a low-structured surface which contains only few and low-contrasted edges. Of the both algorithms, the Hough-based algorithm did show slightly better performance at the cost of a more than five times higher computing time. With nowadays computing power, a real-time operation of this algorithm is not realizable with hardware that can be used in flight tests. So it has to be concluded, that at the moment the Douglas-Peucker-based algorithm is the better choice for a utilization as a redundant feature tracker.

In the future, a fusion of the Douglas-Peucker-Method with another feature tracker is planned. Hence, different DVE scenarios shall be surveyed to verify the anticipated benefit from adding a line-extracting algorithm to a conventional approach. Scenarios with raindrops covering the lens of the camera are of special interest in this context.

8. Acknowledgment

The author wants to thank Airbus Defence and Space which has partially funded the research that was conducted in this paper. Also, thanks to Mr. Rico Dötsch who did help with creating the presented algorithms in the course of writing his Bachelor thesis and to Mr. Michael Zimmermann who did provide the synthetic test scenario.

References

[1] F. Andert, N. Ammann, J. Pueschel, and J. Dittrich, "On the Safe Navigation Problem for Unmanned Aircraft: Visual Odometry and Alignment Optimizations for UAV Positioning," *Proceedings of the 2014 International Conference on Unmanned Aircraft Systems*, pp. 734–743, 2014.

[2] R. Madison, G. Andrews, P. DeBitetto, S. Rasmussen, and M. Bottkol, "Vision-Aided Navigation for Small UAVs in GPS-Challenged Environments," *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, May 2007.

[3] M. Couch and D. Lindell, "Study on rotorcraft safety and survivability," in *Proceedings of the 66th American Helicopter Society Forum*, 2010.

[4] G. Bradsky and A. Kaehler, *Learning OpenCV - Computer Vision with the OpenCV Library*. O'Reilly, 2011.

[5] Z. Zhang, "A flexible new technique for camera calibration," in *Proc. of IEEE Transaction on Pattern Analysis and Machine Intelligence 22*, pp. 1330–1334, 2000.

[6] P. Hough, "Method and means for recognizing complex patterns," in *U.S. Patent*, p. 3.069.654, 1962.

[7] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," in *Cartographica, The International Journal for Geographic Information and Geovisualization*, pp. 10:112–122, 1973.

[8] J. Zhou, W. Bischof, and A. Sanchez-Azofeifa, "Extracting lines in noisy image using directional information," in *Pattern Recognition, 18th International Conference on*, pp. 215–218, 2006.

[9] J.-M. Geusebroek, A. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," in *Image Processing, IEEE Transactions on 12 (8)*, pp. 938–943, 2003.

[10] J. Canny, "A computational approach to edge detection," in *Pattern Analysis and Machine Intelligence 8*, pp. 679–698, 1986.

[11] T. Pavlidis, *Algorithms for Graphics and Image Processing*. Computer Science Press, 1982.