ALLFLIGHT - BLOB-BASED APPROACH TO DETECT DANGEROUS DRIFT VELOCITIES DURING HELICOPTER LANDING APPROACHES

Alexander Gatter[a], Hans-Ullrich Doehler[b], Thomas Lueken[b], Ralf Stadelhofer[c]

[a]Institute of Flight Systems and [b]Institute of Flight Guidance
German Aerospace Center, DLR, Lilienthalplatz 7
38108 Braunschweig, Germany

[c]Avionics Systems - COEYA 2
Cassidian Electronics, Claude-Dornier-Str. 1
88039 Friedrichshafen, Germany

ABSTRACT

The Institute of Flight Systems at the German Aerospace Center (DLR) site in Braunschweig is dedicating much effort to the goal of making a helicopter flying safe. This paper is concentrating on decreasing the danger of accidents caused by so called “dynamic rollovers” which may result from undetected lateral velocities during the landing approach. Especially in degraded visual environment (DVE) situations when pilots cannot evaluate visually the horizontal movement many accidents happen. While under normal circumstances the combination of INS and GPS is usually sufficient to detect critical lateral drift velocities during the landing process, the INS drift may reach a dangerously high value quite soon after the GPS signal has been lost (which may happen e.g. in canyons, cities, or due to jamming).

The vision algorithm that is proposed here uses a blob-based approach to solve this problem. Blobs are regions in an image that fit together bound by a specific criterion. In this paper we will describe how an image is segmented, possible blobs are selected, characterized, how they are tracked, and how the velocity of the helicopter is calculated. Finally, the accuracy of this algorithm will be analyzed using data recorded from flight tests that have been conducted by the DLR’s flying helicopter simulator EC135 (ACT/FHS).

1. INTRODUCTION

The project ALLFlight (Assisted Low Level Flight and Landing on Unprepared Landing Sites) from the German Aerospace Center is focusing on decreasing the workload of the helicopter pilot and on increasing his situational and mission awareness. To that purpose, a couple of sensors have been attached to the ACT/FHS. This highly modified helicopter is equipped with a couple of sensors: a camera that is operating at a rate of 25 Hz and is sending images with a recorded resolution of 640x480 pixels, a thermal infrared bolometer camera from the company MaxViz, Inc. that is operating at a rate of 30 Hz and is also sending images with a resolution of 640x480 pixels, the laser scanner system HELLAS from the company Cassidian that is delivering data with a rate of 2 Hz and a resolution of 95x200 pixels, and the millimeter wave impulse radar system AI-130 from the company ICX. Additionally, the helicopter possesses the H-764 ACE INS from Honeywell which is a coupled GPS/INS system that shows an error in velocity accuracy of under 0.05 m/s when coupled with GPS and an error of under 1.0 m/s without coupling with GPS. INS data during the flight tests have been recorded at a rate of 10 Hz. Figure 1 shows the sensor mount which is equipped with the cameras, the ladar, and the radar. For our algorithm we used the radar altimeter, the camera, and the INS.

Figure 1: Picture of the sensors that have been attached to the ACT/FHS.

Since dynamic rollovers are responsible for a large amount of helicopter crashes and therefore also for immense costs, many companies are trying to find a reliable solution to this problem. Most of them are setting their expectations on expensive sensor equipment to overcome this dangerous situation as can be seen in a NATO report [1] about the brown-out problem from 2010. We however provide a low-cost solution, that in case of a brown- or white-out situation exploits the circumstance that during the last seconds of the landing approach a space below the helicopter is created that is relatively free from dust. This is called "donut effect". Some first approaches to a low cost solution already have been made in [2] and [3].

There is no absolute value to be found that states when lateral velocity is getting dangerous, but pilot surveys provided a maximal velocity between 0.5 m/s and 1 m/s

Our goal is to develop a solution for the dynamic rollover problem that is also working in situations with much disturbance (e.g. small particles flying around) or at night and therefore needing infrared images. This intention prohibits the use of established algorithms for image movement detection like the KLT from Lucas and Kanade [4] that is focusing on corners with high contrast. Also, the MSER algorithm [5] that has the highest similarity of all blob-detectors to our algorithm is only performing really well on images with sharp contrasts and else homogeneous areas (see [6]) which we did verify by applying this algorithm to our real test-scenes. Another famous blob-detector is the SIFT algorithm [7] that calculates gradients in several directions for regions and therefore also needs high contrasts. The SURF algorithm [8] roughly works similar to the SIFT algorithm while only approximating the gradients.

The algorithm that has been implemented to overcome the problem of undetected lateral velocities is based on finding regions in an image (we often will refer to them as so-called "blobs") that can be tracked reliably over a longer period of time (another blob-based algorithm for detecting movement can be found in [9]).

In the following sections we will show the complete computation pipeline which starts with the segmentation of an incoming image by grouping regions together that are similar in their brightness, the creation of blobs, the selection of blobs that are suitable for tracking, the extraction of features from those blobs, the tracking of them over time, estimating movement through the blob displacement over time with assistance from INS data, and as a last step merging the information of several blobs to get the final estimation of the movement and velocity of the helicopter. The quality of the algorithm is proven with reference data from GPS/SBAS (Satellite Based Augmentation System).

2. ALGORITHM

Now let us take a look into the processing chain of the algorithm. Before the blob detection starts, some beforehand computation has to be done. A low-pass filter is applied to reduce high frequencies since recorded images tend to suffer from noise. Therefore, a median filter is used since it maintains contrasts better than another low pass filter like e.g. a Gaussian filter. The correct adjustment of the filter size is very important since blobs could result from heavy noise in otherwise homogeneous areas while a too big filter size could destroy vital information. Tests on real data have shown that a filter size of 5×5 px is a good tradeoff between keeping important original image data and reduction of noise.

Another important preprocessing step is the rectification of incoming images. The TV camera used by the ACT/FHS tends to produce distorted images with mainly barrel effects. We use the OpenCV library [10] together with a common chessboard calibration pattern (see [11] and [12]) to eliminate those distortions. Results of this rectification can be seen in figure 2.

(a) Distorted image. Strong curvature of the horizon

(b) Rectified image. The horizons curvature is visibly reduced.

Figure 2: Image before and after rectification

2.1. Image Segmentation

After the preprocessing is done, the next step is to segment an incoming image into useful regions. We decided for a region growing based initial flood fill algorithm that groups pixels together according to their intensity. Beginning from the top left pixel of the image, initial blobs are created by taking the intensity of the first pixel that is regarded as the seed point and grouping neighboring pixels into a newly created blob if their intensity differs only up to a certain extent. That procedure alone is unfortunately not very effective when it comes to finding good regions that show a high contrast to their surroundings. That's why the flood fill algorithm is only used to create a presegmentation of the image by using a large acceptable difference in intensity of neighboring pixels and therefore creating a subdivision into mostly more or less homogeneous larger regions. Regions that already fulfill the desired size for a blob despite the large tolerance in intensity are stored as a new blob without further computation. For very small regions a check is performed with which it is tested if they are similar to an adjacent region that could become a possible new blob. If this is the case, both regions are fused. In a following step, histograms of those regions are calculated that are above a certain size and sections of intensity values identified from which only few pixel exist. To those seldom occurring pixels another flood fill is applied in order to check if they are forming connected regions. They are stored as a new blob if the size of those thereby found regions lies within certain boundaries and following requirements are met:

- they have an acceptable size
- they do not touch any border of the image
- their position is below the horizon (detectable using INS-Data and mounting angle of the camera)
- difference in intensity between blob and surrounding is large enough

With that procedure we get features that are likely distinct to their surroundings. Due to that 2-step segmentation approach, our algorithm usually achieves an even distribution of blobs containing regions that show an overall high contrast and regions that differ from an otherwise quite homogeneous surrounding. The whole process of segmenting an image is illustrated in figure 3.

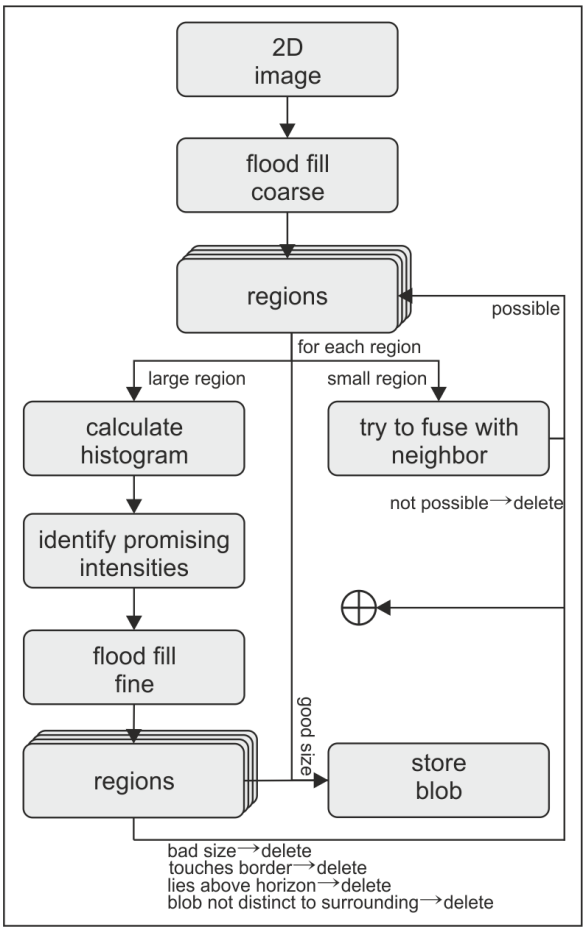


Figure 3: Processing pipeline for the image segmentation

2.2. Structure of a blob

In this section we want to give a short overview of all the components that belong to a blob.

They can be grouped into three different categories:

- internally used data
- data that is used to describe a blob
- data that is used to estimate movement

A list of all this data is presented in figure 4. Later in this paper we will go more into detail about some of this elements. Supplementary to the stored blobs, some arrays exist that handle indexing of blobs. First of all, an array was created that contains the indices for all blobs. The list of adjacent neighbors that every blob possesses is in fact a list of pointers to entries of that array. Pointers have been chosen because of the fact that when a blob is deleted or fused with another blob, it is sufficient to change the underlying index instead of having to iterate through all blobs in order to correct the neighborhood relations. In order to get a fast overview of the image segmentation, a two dimensional array in the size of the image has been created that contains for every pixel the index for addressing the data.

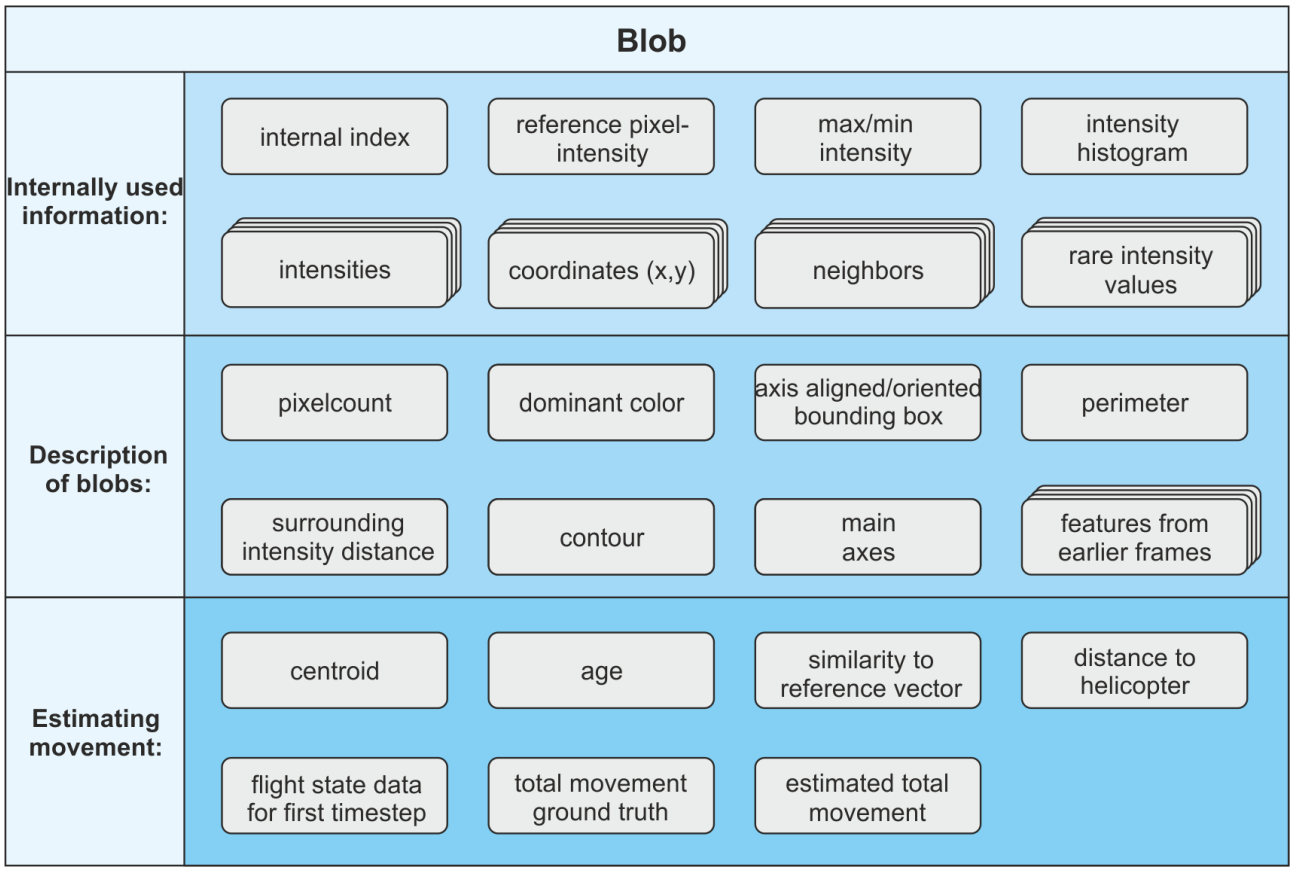


Figure 4: Components of a blob

2.3. Blob Tracking

The actual tracking of blobs is organized in a couple of processing steps. First of all we need to create a region in which we want to search for the blob. Therefore we create a square area around the former blob center. Since we do not want to delete a blob the first time that we have not been able to find it in a subsequent image there also exist situations in which we have no center to place this square around. In that cases the center of the last time that the blob has been found is used and the search area is increased accordingly. Within this search square we now apply flood fills with the minimal and maximal intensity values of the original blob as parameters to create new blobs with the same intensity values. Therefore all connected regions of satisfying size in this area are stored as blobs. This grants a high chance that the original blob is found and selected as a successor again. Only if regions lie within this area that have nearly identical shape as the regarded blob the algorithm will not be able to discriminate between those. In order to make the algorithm robust to lighting changes we also alter the intensity field and perform the flood fill with the new altered intensity values until the whole square is treated (blobs may of course exceed the boundaries of that square). If neither the original parameters nor the altered ones are able to create a new blob, the processing is repeated one more time with stronger deviating intensity values. If this also does not produce a new blob, then the current blob is marked as not being trackable in this time step (and ultimately deleted after it has not been found again for a longer time).

Provided that we found a couple of blobs during the preceding computations, we now have to find the blob that shows the highest resemblance to the reference vector of features. This vector consists of the calculated features of the origin blob if this blob is tracked for the first time or out of a mixture of features from a number of older blobs that have been marked as reliably tracked. All that information have been chosen as features are listed in figure 5.

For the comparison we calculate the features that we have mentioned above for all newly found blobs. We implemented a fuzzy-based voting system that rates the similarity to the origin blob according to their sizes to account for different error-behavior for differently sized blobs. Every feature initially contributes to the score with “1” if it matches the compared feature perfectly and gives lower scores the more distinct the compared features are. After that, a weighting is applied to the single features. The votes of the surround-

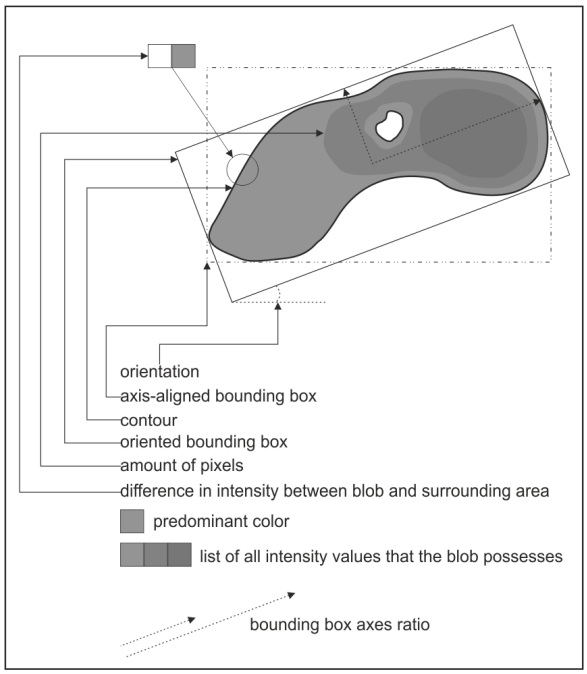


Figure 5: Sketch of a blob with all data that are used for description

ing distance and the area of the oriented bounding box (which is calculated with [13]) are tripled and the votes for pixel amount as well as perimeter (which has been calculated with the algorithm of Pavlidis [14]) are doubled. The blob that achieves the highest score is selected to be the successor to the current blob. This yields good recovery results as long as there are no very similar regions in the direct neighborhood of the regarded blob.

After having identified the blob that is most likely the original blob in the new time step, we are not done yet. Unfortunately we only have found the best fit so far. That the best fit may be completely different from the original blob has not been treated yet. Therefore we now estimate the quality of the tracking by applying certain limits that the new blob may maximally differ from its predecessor and convey these into one number. This number decides if a blob will be used to estimate the helicopter movement in a later step. Finally, all vital information of the old blob is transferred to the new blob which is going to replace the old one in the blob list. That includes a set of features of a couple of predecessing blobs including the features of the current blob. These features are organized in a deque structure which means that older features are only stored up to a certain age until they are replaced by new ones. The process that is described above is also visualized in figure 6.

In order to keep the processing pipeline filled with a sufficient amount of blobs we have to segment new incoming images to extract further blobs. This is not done in every time step since it would increase the list of blobs to track to a huge size. Only if the amount of blobs in the list is dropping under a lower limit an image is taken for a new segmentation. Compared to the initial explanation of segmentation a few more things have to be considered during resegmentation. First of all we do not want blobs to overlap during segmentation and therefore mark regions that already contain a blob in the index map as not valid. Furthermore such a resegmentation step may not increase the amount of blobs to an amount that is over the lower limit of blobs. If this happens, the intensity region with which the blobs are searched during segmentation is set to a finer level and the image is segmented once more. If this also does not lead to the desired amount of blobs the whole process is repeated another time and only then the segmentation ends whether we got enough blobs or not.

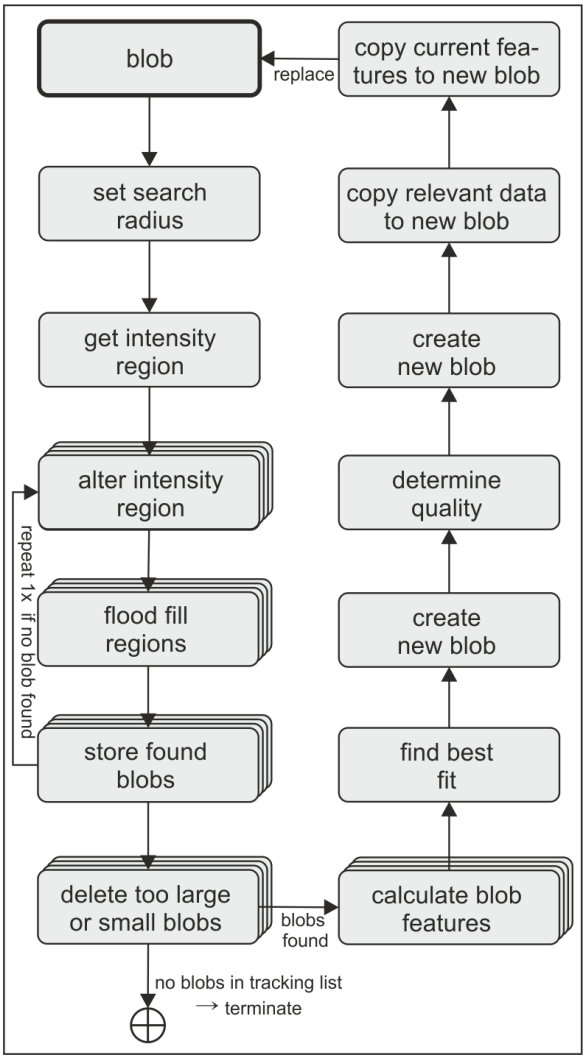


Figure 6: Processing pipeline for tracking blobs

2.4. Movement Estimation

Now that we have treated the way blobs are created and tracked we have to take a look at how to extract actual movement estimation from the information that we have got so far. This estimation is based on the detected shift of the centroids of the blobs between two time steps (time of the blob creation and current time step) and the following projection of this relative distance into world coordinate system by taking a flat earth assumption without slope. To make this assumption valid, we postulate that the landing zone has been selected and has been evaluated to be flat and without slope before the landing approach starts.

This flat earth assumption is assigning a distance from the helicopter to each pixel in the image. Therefore we take the data from the radio altimeter of the ACT/FHS to get the camera's height above ground together with the data of the INS and the knowledge over mounting angle and place of the camera system and the camera intrinsics in reference to the helicopter coordinate system center. Together with the flat earth assumption we then have enough information to assign each pixel a depth value Z by calculating intersection points between the virtual rays emitted by the camera and the ground plane. The absolute movement estimation of the helicopter for one blob is calculated by solving the formula for optical flow for ego-motion

(1)

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} -f & 0 & x \\ 0 & -f & y \end{pmatrix} \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix} + \frac{1}{f} \begin{pmatrix} xy & -(f^2+x^2) & fy \\ (f^2+y^2) & -xy & -fx \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}$$

for its lateral translational component (the horizontal part v is not needed and therefore left out in the following):

(2)

$$T_X = \frac{T_Z x}{f} - \frac{uZ}{f} + \frac{Z}{f^2}\left(xy\omega_1 - (f^2+x^2)\omega_2 + fy\omega_3\right)$$

T_X and T_Z stand for the absolute lateral and directional movement estimation of the helicopter. T_Z may be calculated by the change of estimated distance on the ground between blob and helicopter over time and a plausibility check but we take this information from the INS so far. ω_1, ω_2, and ω_3 are the changes in orientation between two time steps within the camera coordinate system, f is the focal length, x, and y the coordinates within the image (relative to the optical center), and finally u and v stand for the lateral and horizontal displacement in pixels of the centroid between two time steps. This gives us one lateral movement estimation per blob in reference to the orientation of the helicopter at the time when the blob was created (=reference coordinate system). While one blob would already be sufficient to give us all translational information that we need we take as many blobs as we can detect for the drift calculation to lessen the effects of wrong tracking results of single blobs. Since blobs from different time steps have different reference coordinate systems we have to create a global reference system into which all local reference systems are transformed. Since the orientation of the helicopter will not vary much during the last seconds of a landing approach we have taken an average over all orientations of the last seconds.

In order to receive reference data for comparison reasons we also performed a movement calculation with the velocity data from our GPS-aided INS by transforming and summarizing the incoming signals into the reference coordinate system of each blob and compare them to the results of our algorithm.

For the final determination of the validity of a blob's measured movement we also take the information of the INS but here we only take the maximal accuracy of the INS without GPS backing. The INS that is built into the ACT/FHS is delivering raw INS data with an error of maximal one meter per second. This builds the upper limit for acceptance of a movement estimation. If a blob reports a movement that exceeds this upper limit then it is not taken into account for determining the movement of the helicopter. With this we also are able to back up our assumption of a flat earth because blobs that lie on structures that are significantly elevated over the ground will produce estimations that exceed the maximal INS error and therefore will not be included for movement measurement. The final result is then low pass filtered to further suppress high frequencies that are caused by jittering of the centroids. Over those smoothed values we calculate the velocity by simply taking the moved distance over the last second.

Figure 7: Visualization of all blobs that have been found in an image at two different timesteps during the tracking process

3. TEST ON REAL DATA

The algorithm has been applied to real test data that have been recorded during flight tests at the airport of Braunschweig. In order to be able to survey the ability of the algorithm to detect lateral movement we took T_Z from the reference data for the evaluation of this flight tests.

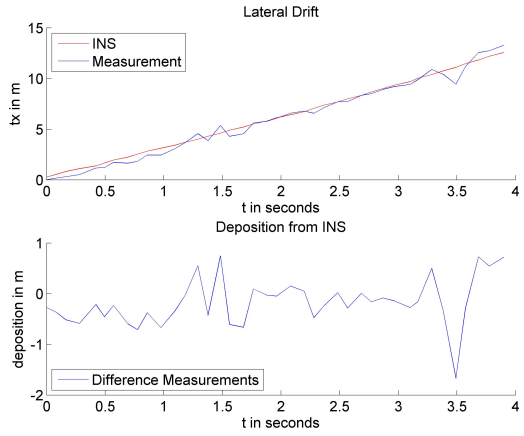
In figure 7 we present the visualization of two time steps during the tracking process. All blobs that have been detected at the current time step have been marked with a green (= actual blob borders) and white (= contour) border. The regions that are filled with red are the blobs that have been kept tracked successfully after a couple of time steps. In this example we can see that all except of two remaining blobs have been tracked correctly. The lower of those falsely tracked blobs can easily be identified by a comparison between measurement and INS data. The blobs are distributed quite evenly and despite having the strongest contrasts lying behind the taxiway these regions contain no blobs because the possible blobs that have been found there all touch the horizon and therefore have been deleted.

3.1. Test scene 1

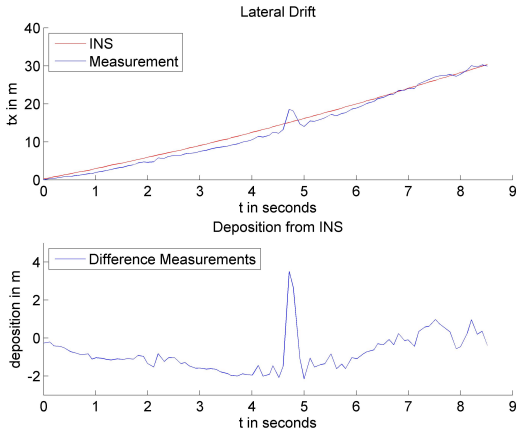
Some background information to this test:

- lateral flight at the airport Braunschweig
- height above ground is 3 meter
- mounting angle of the camera is forwarded and approximately 16 degrees downwards
- lateral speed of the helicopter is approximately 3 meter per second
- total length of regarded test file is 35 seconds
- The camera sends data at a rate of 25 Hz
- INS information is available at a rate of 10 Hz

In figure 8, we present the behavior of selected blobs to visualize the possible accuracy in absolutely moved distance that we can achieve. Due to the fact that the blobs form did quiver a bit over time because of small lighting changes and discretization effects the measured moved distance jumps slightly around the



(a) Over the tracking time of four seconds the helicopter moved sideways with a speed of approximately 3 meter per second. The error peak at second 3.5 is induced by a wrong detection of the blob.



(b) Over the tracking time of 8.6 seconds the helicopter moved sideways with a speed of slightly over 3 meters per second. The error peak at second 4.8 is induced by a wrong detection of the blob over 2 time steps.

Figure 8: Tracking of selected blobs in test scene 1

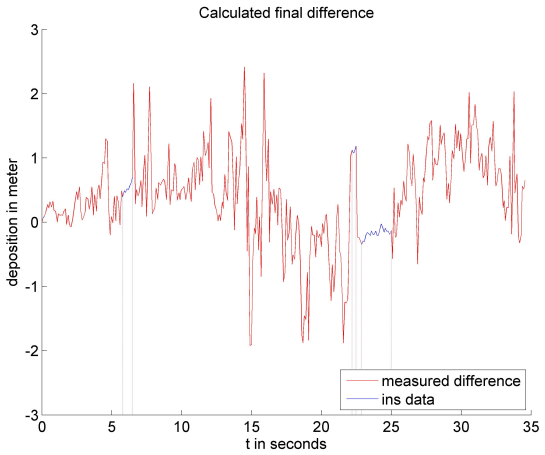
reference data but the error that is induced by that does not increase over time.

After having presented the performance of selected blobs we now will show the complete processing of a scene. Therefore we start the computation at the same time step as in figure 8(a).

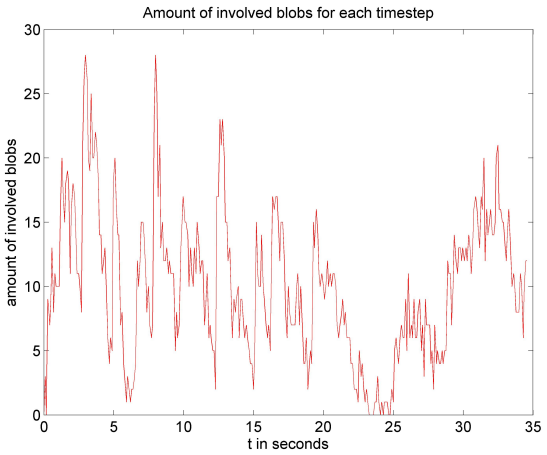
Explanation for the evaluation-plots of the tests: The red segments in the top left image of figure 9 represent the calculated shift of the movement of the helicopter in comparison to the reference data. Since this algorithm is only interested in drift velocities and not in absolute deposition we did not accumulate any moved distances here. The measured values are just the differences between reference data and measurements of every blob in its own coordinate system at every time step. The blue parts are normally distributed random values based on the maximal error of the unaided INS that have been used instead of the measurements of the blobs because at these time steps too few blobs were able to give a vote (see the picture on the top right side that depicts the amount of blobs that took part in the voting for each time step). The bottom left picture shows the averaged measured values and the bottom right picture shows the estimated difference in velocity in meter per second.

Figure 9 shows all important data that have been evaluated. The averaged shift of all blobs in test scene 1 reaches error peaks of up to 2.5 meter shift from the INS data but lies under 1 meter most of the time. The measurements also do not tend to drift away from the reference data. The amount of blobs that did vote for the movement estimation ranges between 0 and 28. Whenever only 3 or less blobs did give a voting then their vote was discarded and instead we used INS data that we modified with a normally distributed error to simulate an accuracy of an INS that is not coupled with GPS.

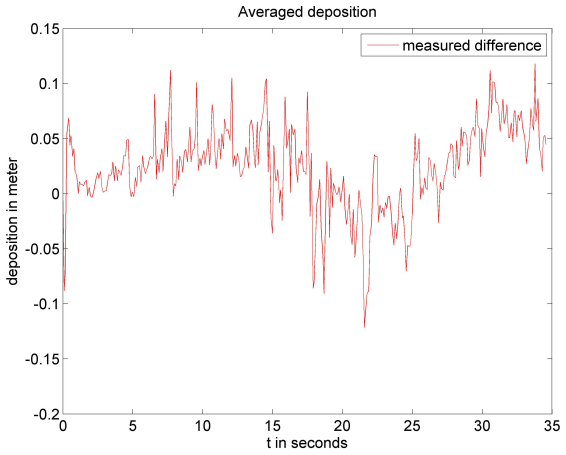
This did happen in test scene 1 between seconds 5 and 6, at second 23, and between seconds 23 and 25. These time segments are visualized in figure 9(a) by changing the color of the graph to blue and by framing those segments with blue dotted lines. Under the assumption of a quite uniform landing approach and the nonexistence of undetected aggressive pilot inputs we decided to take the average of measurements over the last 3 seconds and determine the velocity estimation for the helicopter with these. This led to an estimated error in velocity of 0.1 meter per second in test scene 1. Shortening the area of averaging by 1 or 2 seconds would result in an increase of calculated resulting velocity by 0.1 or 0.2 seconds in its error peaks. Even a shortening of the sampling range to 1 second would therefore not violate the requirements specification.



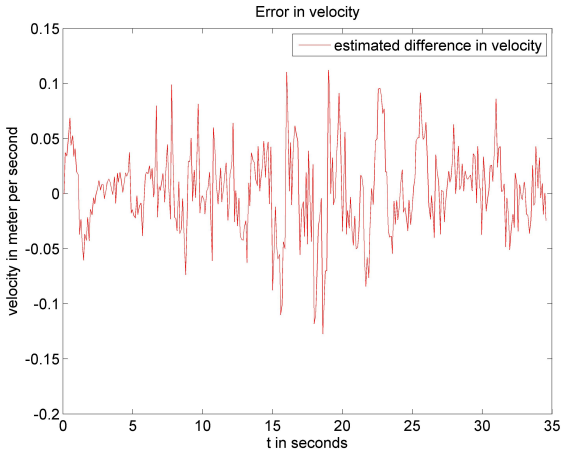
(a) Position error between measured data and reference data



(b) Amount of blobs that took part in the voting



(c) Mean value of position error data over 3 seconds



(d) Velocity error between reference and measured data

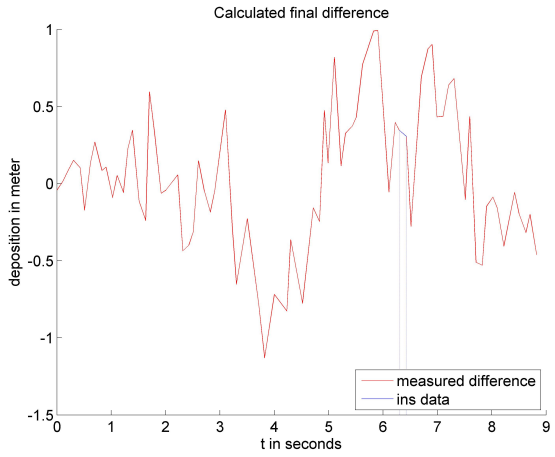
Figure 9: Evaluation of test scene 1

3.1.1. Test scene 2

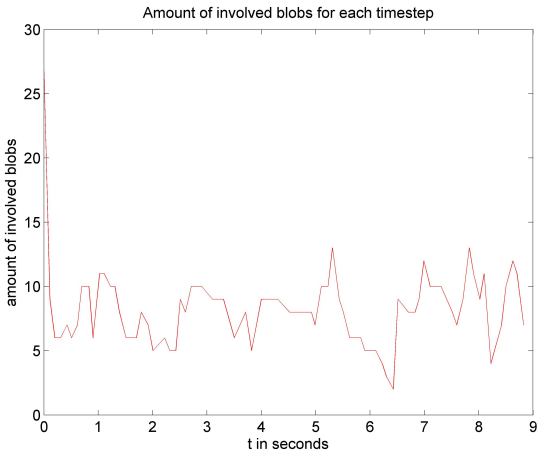
Some background information to this test:

- landing approach at the airport Braunschweig
- height above ground goes from 3 meter in the beginning to 1 meter
- mounting angle of the camera is forwarded and approximately 16 degrees downwards
- speed of the helicopter is approximately 3 meter per second
- total length of regarded test file is 9 seconds
- The camera sends data at a rate of 25 Hz
- INS information is available at a rate of 10 Hz

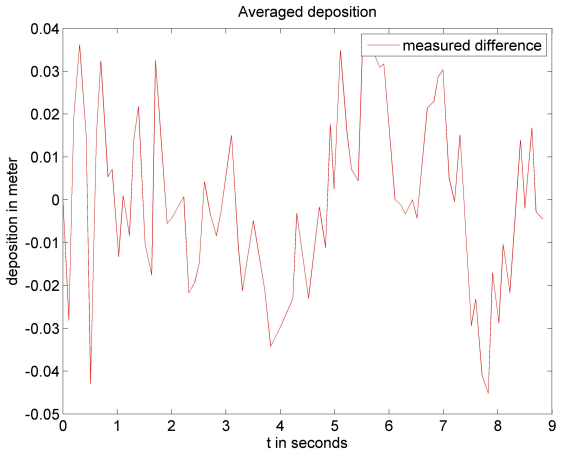
Figure 10 shows all important data that have been evaluated for test scene 2 and is structured identical to the evaluation of test scene 1. Test scene 2 shows up error peaks of only about 1 meter. The measurements also do not tend to drift away from the reference data. The minimal amount of 3 blobs has only been breached once between seconds 6 and 7. Averaging the data in the same manner as in test scene 1 led to an estimated error in velocity of 0.06 meter per second. Shortening the area of averaging by 1 or 2 seconds would also result in an increase of calculated resulting velocity by 0.1 or 0.2 seconds in its error peaks.



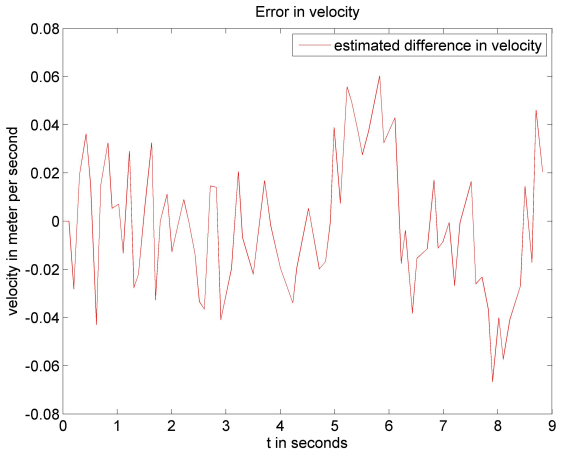
(a) Position error between measured data and reference data



(b) Amount of blobs that took part in the voting



(c) Mean value of position error data over 3 seconds



(d) Velocity error between reference and measured data

Figure 10: Evaluation of test scene 2

4. CONCLUSION

In this paper we have presented an alternative approach for the estimation of movement speed of a helicopter with a camera and the backup from an INS. This algorithm is aimed at adverse contrast situations and scenarios where parts of the image or even the whole image may sometimes be disturbed over several time steps. Following single features over several time steps while judging if the tracked region is still identical to the original one enables estimation of movement that can reduce discretization problems by a large amount and prevent drifting of the measurements up to a certain extent. As well it enables us to recover features that have been obscured for a short time. Missing depth information has been estimated by a flat earth assumption and the suitability of the algorithm for detecting dangerous drift velocities has been evaluated by the use of real test data, showing an error in velocity estimation of about 0.1 meter per second on averaged input data. This error is significantly smaller than the most restrictive demand of a drift velocity of under 0.5 meter per second and therefore fulfills the required performance. In the future we want to improve the algorithm with following points:

- the principal movement direction will in future be calculated dependent on the relation between orientation and absolute movement in a direction,
- the square based search method will be advanced by additionally using the information that we get from the INS (like in [15]),
- the algorithm shall be enhanced by additionally taking features of different kind to overcome scenarios where the blob based algorithm does not return a sufficient, amount of blobs
- the capability of coping with adverse weather and lighting situations shall be tested on virtual and real image data.

As a final conclusion it can be summarized that this algorithm possesses the potential to aiding pilots during landing approaches by reliably detecting lateral velocities that could be responsible for dangerous accidents and therefore is contributing to the goal of making a helicopter flying more safely.

COPYRIGHT STATEMENT

REFERENCES

[1] NATO Science and Technology Organization, "Rotary-wing brownout mitigation: Technologies and training." http://www.cso.nato.int/Pubs/rdp.asp?RDP=RTO-TR-HFM-162, 2010.

[2] H.-U. Doehler and N. Peinecke, "Image-based drift and height estimation for helicopter landings in brownout," in *ICIAR 10 Proceedings of the 7th international conference on Image Analysis and Recognition - Volume Part II*, pp. 366–377, 2010.

[3] M. Hebel, K. Bers, and K. Jäger, "Imaging sensor fusion and enhanced vision for helicopter landing operations," *SPIE*, vol. 6226, pp. 62260M:1–10, May 2006.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, pp. 121–130, 1981.

[5] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," *Procedings of the British Machine Vision Conference 2002*, pp. 36.1–36.10, 2002.

[6] T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2007.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision 60*, pp. 91–110, 2004.

[8] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proceedings of the 9th European Conference on Computer Vision*, pp. 404–417, 2006.

[9] L. Goormann, "Objektorientierte Bildverarbeitungsalgorithmen zum relativen Hovern eines autonomen Helikopters," Diplomarbeit, Fachhochschule Braunschweig/Wolfenbüttel, 2004.

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

[11] D. C. Brown, "Close-range camera calibration," in *Photogrammetric Engineering 37*, pp. 855–866, 1971.

[12] 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.

[13] S. Gottschalk, *Collision Queries using Oriented Bounding Boxes*. PhD thesis, Chapel Hill, 2000.

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

[15] M. Veth and J. Raquet, "Fusion of low-cost imaging and inertial sensors for navigation," *Proceedings of the ION meeting on Global Navigation Satellite Systems*, 2007.