Recognition of a Predefined Landmark Using Optical Flow Sensor/Camera

Galiev Ilfat, Alina Garaeva, Nikita Aslanyan
The Department of Computer Science & Automation, TU Ilmenau
98693 Ilmenau
ilfat.galiev@tu-ilmenau.de; alina.garaeva@tu-ilmenau.de; nikita.aslanyan@tu-ilmenau.de;

***Abstract*— In this paper vision-based approaches for autonomous landing of Unmanned Aerial Vehicle (UAV) are considered. The aim of this work is to analyze and compare existing approaches in landmark recognition techniques and implement one of them. We have reviewed several approaches for solving the problem, such as recognition using Haar Features Cascade Classifier, Speeded-Up Robust Features (SURF) with Fast Library Approximate Nearest Neighbor Search (FLANN), contour analysis. All these approaches we implemented in Python programming language with the help of OpenCV library.**

***Index Terms*— Haar Cascade Classifier, SURF, FLANN, OpenCV, quadrocopter, UAV, raspberry pi, image processing, autonomous landing, helipad, landmark.**

I. Introduction

Nowadays UAVs are widely used for solving a wide range of the problems, where participation of human can be dangerous, expensive or impossible. Examples of such problems are exploration of regions and places that are hard to reach, aerial survey, inspection of power lines, e.t.c. For some problems, such as patroling or recovering stationary nodes in communication network, UAVs with autonomous navigation may be required. Autonomous control of UAV includes following tasks: takeoff, landing, object tracking, flying along a predetermined trajectory, e.t.c. Sometimes navigation using Global Positioning System (GPS) or Inertial Navigation System (INS) cannot be used due to low accuracy or availability aspects. In such cases, computer vision approach for navigation can be applied.

The aim of this study is to analyse and compare different vision-based algorithms, which allow recognizing several different landmarks, and use results to implement it in UAV to solve problems of hovering or landing autonomously.

This paper consists of two main parts: II Vision based approaches and III Implementation of Approaches. Section II describes three algorithms which were considered in this work, their basic logic and main features. Section III covers practical part of the work, where we implemented each of these algorithms and compared them according to accuracy and performance rate.

II. Vision based approaches

In this work the task of recognition of predefined landmarks is considered (Fig. 1). The recognition process consists of the following subtasks: detection of landmark in the video

Fig. 1. Landmarks example

stream and recognition of the type of this landmark. For landmark recognition problem we considered the following approaches:

A. Haar cascade

The first vision-based approach we considered was Haar Classifier [6]. Basically, Haar Classifier, also known as Viola-Jones algorithm, is used for face recognition problem, but in recent years it is also used for solving wide range of computer vision problems.

Main principles of Haar Classifier Algorithm are following:

1) Integral representation of image. On the first step, it is necessary to get integral representation of the image transforming it into the from of a matrix, which has the same size as initial image. Each element of matrix is a sum of nearby pixels brightness. To calculate this sum, pixels, which are located above and on the left of the corresponding element, are considered. Elements of the matrix are calculated using (1):

$$L(x,y) = \sum_{i=1,j=1}^{i<=x,j<=y} I(i,j), \quad (1)$$

where $I(i,j)$ - the brightness of a pixel of the original image.
Suppose, that the image is a rectangle ABCD (e.g. Fig. 2). The sum of the pixels of this rectangle can be expressed through the difference of adjacent rectangles, by using (2):

$$S(A,B,C,D) = L(A) + L(C) - L(B) - L(D) \quad (2)$$

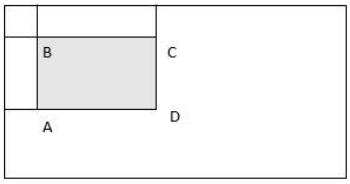

Fig. 2. Rectangle for integral sum calculation

2) Haar-Like Features.
Haar-Like Features is a set of rectangles, which called Haar primitives (e.g. Fig. 2). Each rectangle is defined by the weight, which is calculated as a sum of pixels brightness values (these pixels are covered by this rectangle).
Feature value could be calculated by the usage of following equation $F = A - B$, where A - sum of brightness of pixels, which covered by light part of the feature, B - sum of brightness of pixels, which covered by dark part of the feature rectangle.
Haar Classifier is looking for the landmark and landmark features using general approach of scanning window. Scanning window moves across the input image; in each windows position a huge number of features (for different features shapes, sizes and location) are calculated.

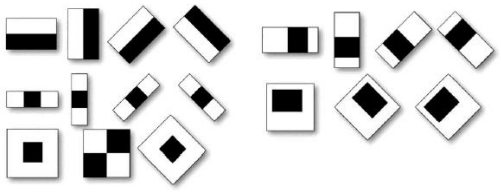

Fig. 3. Examples of Haar primitives

3) Boosting.
Boosting is a set of methods, which improves the accuracy of analytic model. In machine learning boosting means a procedure, which sequentially construct a composition of machine learning algorithms, where each of following algorithms tries to improve and compensate weaknesses of all previous algorithms compositions.
Boosting is a greedy algorithm of algorithms composition construction. The main purpose of boosting approach is finding a local optimal solution on each stage. Boosting on decision trees is one of the most effective method for classification issue.
Improvement of Boosting Algorithm is AdaBoost [7] approach, which uses a various number of classifiers and provides a learning stage which uses only one testing set, applying them in turn to the various steps.

B. Speeded-Up Robust Features (SURF)

SURF [4] consists of two main steps - finding specific points of the image and creating descriptors that are invariant to scale and rotation. Last means that the description of the key points will be the same, even if the sample's size will be changed or it will be rotated. Search for points is also invariant.

The method searches for specific terms by Hessian matrix. The determinant of the Hessian matrix reaches extremum at the point of maximum change in the gradient of brightness. So algorithm can easily detect spots, corners and edges.

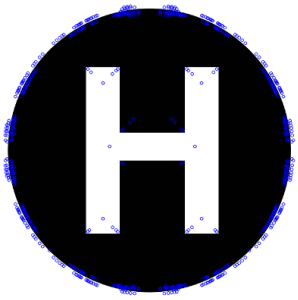

Fig. 4. SURF keypoints example

Hessian is invariant under rotation, but do not scale invariant. Therefore SURF uses various scale filters for finding the

Hessians.

For each key point the direction of maximum brightness change (gradient) and scale, taken from a scale factor of Hessian matrix, are calculated.

The gradient at the point is calculated using Haar filters. After finding the key points, SURF forms their descriptors. The descriptor is as a set of 64 (or 128) numbers for each key point. These numbers reflect the fluctuations of the gradient around the key point. The key point is the maximum of the Hessian matrix and it ensures that the neighborhood should be areas with different gradients. Thus, SURF ensures a variance (difference) of descriptors for different key points.

Fluctuations of gradient's neighborhood key points are calculated relative to the direction of the gradient around the point (near the key points). Therefore, invariance under rotation handle is achived. The size of the field, where descriptor is calculated, determined by the scale of Hessian matrix, which provides scale invariance. Fluctuations gradients are also calculated by the Haar features (Fig. 3).

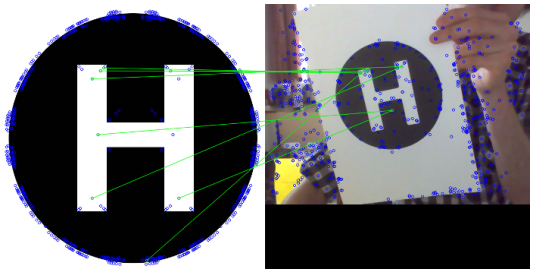

Fig. 5. SURF matching example

For matching keypoints of SURF we used Fast Library Approximate Nearest Neighbor Search (FLANN) algorithm.

FLANN [5] is a library for performing fast approximate nearest neighbor search (NNS). NNS is used to find among a set of elements, arranged in a metric space, those, that are close to the target element, according to some predetermined proximity function that defines that metric space.

C. Contour analysis

The last approach - based on contour processing. The idea is to try finding a contour, that describes our landmark and determines the type of the mark. To determine a label on the image we used Canny edge detector [3]. It finds contours on the image preconverted into a binary form. For binarization different thresholds can be used. It can help to reduce influence of the landmark illumination on the recognition accuracy.

1) Canny operator: Canny studied the mathematical problem of obtaining the filter: the optimal criteria for selection, localization and minimization of multiple responses of object's borders. This means that the detector should react to the border, but ignore the false detections and accurately determine the boundary line (without its fragmentation) and respond to each border once to avoid the perception of broad borders brightness changes as a set of boundaries. Canny introduced the concept of Non-Maximum Suppression (non-maxima suppression), which means that the points are declared for boundaries of the pixels at which a local maximum of the gradient is achieved in the direction of the gradient vector.

Detector uses filters which are based on the first derivative. First derivative is very sensitive to noise, that is why detector is recommended to use for only preprocessed images. The original images have to be smoothed with a Gaussian filter. The boundaries of the image may be in different directions, so Canny algorithm uses four filters to detect horizontal, vertical and diagonal borders by using the edge detection operator(e.g, Sobel operator) which gets a value for the first derivative in the horizontal direction (Gy) and vertical direction (Gx)

Using this gradients we can obtaine the angle of border direction:

$$Q = arctan(\frac{G_x}{G_y}) \tag{3}$$

The angle of border direction is rounded to one of the four corners representing the vertical, horizontal and two diagonal (e.g., 0, 45, 90 and 135 degrees). Detector checks the value of gradient is it local maximum in this direction or not.

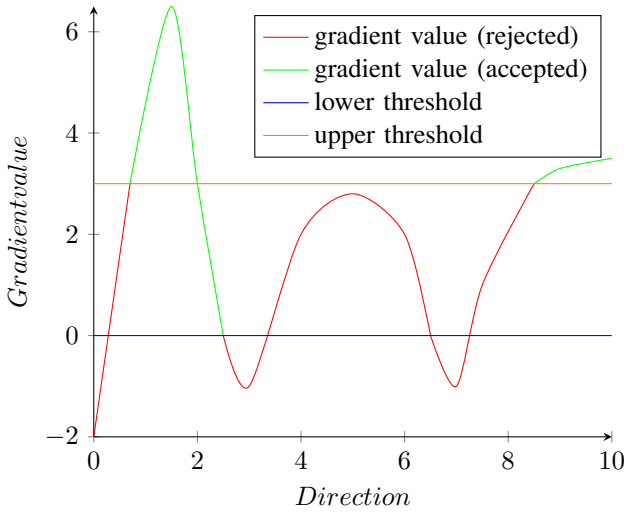

Fig. 6. Graphics of gradient value with two threshold

The algorithm of border detection is not limited by the computation of the smoothed image gradient. Points of the maximum gradient of the image are saved in the borders contour, but close points to the border will be removed. It also uses information about the direction of the border to remove points which are close to the border and do not break the border near the local maxima of the gradient. Then, using two thresholds weak boundaries are removed.

A fragment of the border at the same time is processed as a whole. If the value of the gradient somewhere on traced fragment exceeds the upper threshold, then this fragment is also "permitted" and also in the points of borders where the gradient value falls below this threshold, until it falls below the lower threshold. If on the entire fragment there is no

point with a value greater than the upper threshold, then it is deleted. (See Fig. 6)

This allows to reduce the number of breaks in the output limits. Considering noises in Canny algorith leads to increased stability of the result on the one hand, but on the other hand - increases the computational cost and leads to distortion and even the loss of detail borders. For example, such algorithm is used for rounding corners of objects and braking borders at joint-points of object.

III. Implementation of Approaches

A. Implementation of Haar Cascade Classifier

For implementing Haar Cascade Classifier algorithm we used OpenCV library[8]. First, we created a training set based on landmark images by using *opencv_createsamples* tool, which allows to create multiple images with different background and different rotation angles out of source image.This images set is used for training of Haar Classifier. Examples of such training set is presented in Fig. 7.

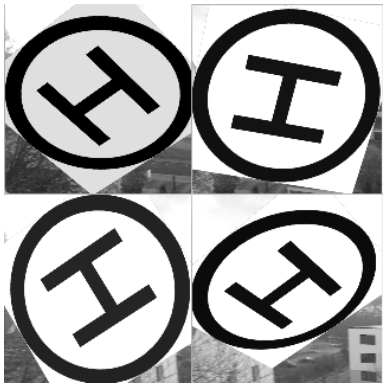

Fig. 7. Example of images from training set

The second step is training of Haar Classifier by using created training set. Different number of training set images and different learning accuracy was used, e.g. 100, 1000, 10000 images. Increasing number of images in training set and learning accuracy leads to increasing recognition accuracy, but it is also slowing down the performance of the algoritm. Haar Classifier is based on gradient features of the image. Because of landmarks are represented with simple images with low gradient properties (not enough unique features are present) this results into a lot of false detections of landmark. In order to improve performance of this algorithm, we need additional post-processing procedures, that will be checking results of the recognition step.

B. Implementation of SURF

SURF algorithm was implemented in OpenCV library. The first step in SURF algorithm is finding key points (descriptors) of the template image, which are used in recognition step by matching them with descriptors of input images. For matching key points FLANN algorithm is used. As a result of applying this algorithm low recognition accuracy was obtained, because the template image has many simple features (right angles, straight lines) which lack of unique image information.

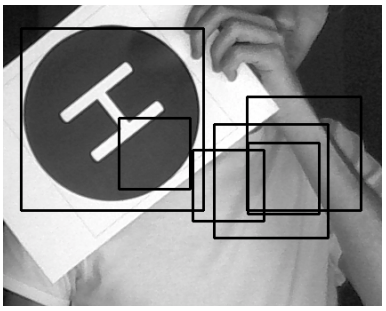

Fig. 8. Haar Classifier false detection example

C. Implementation of Contour analyse

1) Contours processing: The first step in contour analysis is finding a contour of input image using Canny operator. Then by processing each contour, circle on a landmark image is found by following algorithm:

1) Calculation of contour square (number of pixels bounded by this contour)
2) Calculation of contour perimeter (number of pixels on the boundary of contour)
3) Calculation of compactness of figure, which defined as a ratio of perimeter squared to the square of the contour.

$$Compactness = \frac{Perimeter^2}{Square} \tag{4}$$

4) For circle compactness value is defined as

$$\frac{PI * R^2}{(2 * PI * R)^2} = \frac{1}{4 * PI} = 0.079577 \tag{5}$$

which allows to decide whether it is circle or not.

After all circles on the source image are found, for each circle processing of internal contours is performed, which are obtained by using OpenCV library function.

For each internal contour, the minimum bounding rectangle is found. In this rectangle main control points 1, 2, 3, 4, 5, 6 are defined by the following principles: firstly the position of these points are calculated relatively to boundary of rectangle (See Fig. 9,10,11), then additional points 7 and 8, which identify particular landmark are calculated. Then brightness of pixels of the lines in binary image between main control points and additional near-by points are checked.

2) Determinig H-mark: following conditions must be satisfied: white pixels for lines 1-3, 4-6, 2-5 , black for pixels near additional points 7 and 8. See Fig. 9.

3) Determinig E-mark: lines 1-3, 3-4, 1-6, 2-5 are white, lines 7-9, 8-10 are black. See Fig. 10.

4) Determinig Z-mark: lines 3-4, 4-6, 1-6 are white, lines 5-7, 2-8 are black. See Fig. 11.

The final result of implementing analyse a is shown in Fig. 12

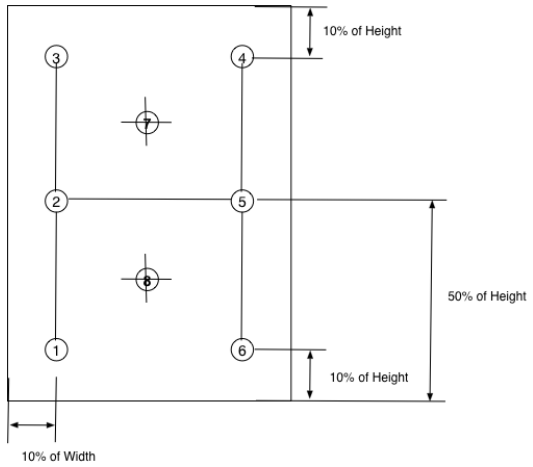

Fig. 9. Defining main control points for h-landmark recognition

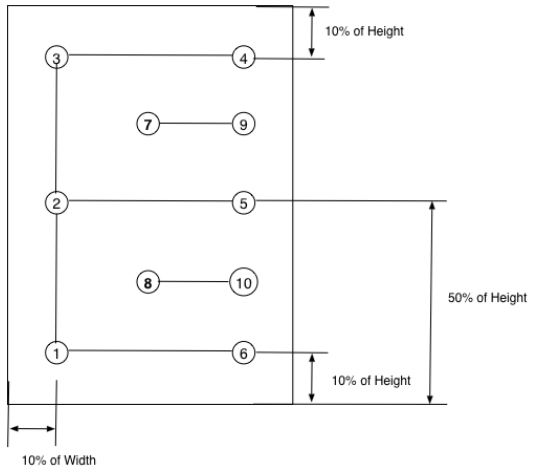

Fig. 10. Defining main control points for e-landmark recognition

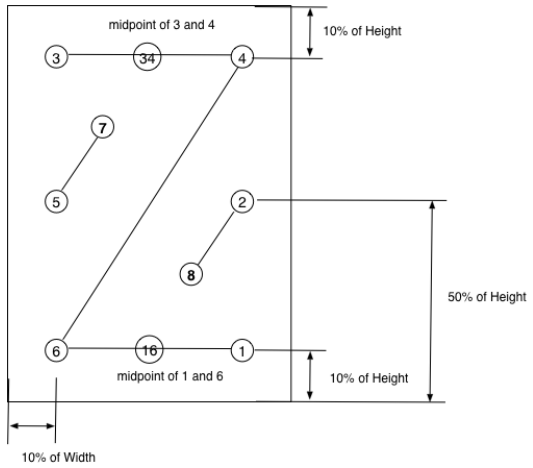

Fig. 11. Defining main control points for z-landmark recognition

IV. Conclusions

As a result of this work, thee vision-based approaches were considered and implemented. Comparing result of these algorithms showed us that the optimal solution for our landmark recognition problem was applying the contour analysis. This method has shown high recognition accuracy and performance in contrast to other two methods considered. Also this algorithm shows more stable performance in case of large distance between optical sensor/camera and landmark, comparing to other algotithms.

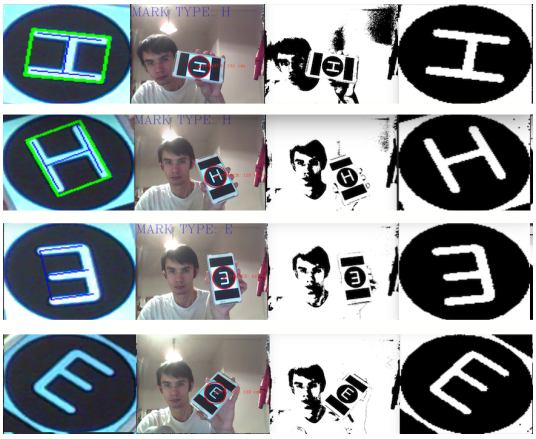

Fig. 12. Final result of the work. Landmark recognition in action

V. Future work

In this work solution of navigation problem(landing, hovering, e.t.c.) was not considered. Using information about recognized landmark UAV should perform some corresponding actions: land on the landmark, hover above the landmark and so on. That part is a navigation problem, which will be a task of the next step of this work.

References

[1] Marius M., David G.L. Fast Approximate Nearest Neighbors With Automatic Algoritm Configuration//Computer Science Department, University of British Columbia, Vancouver.-2009. 10 .

[2] Lowe, D. G., Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[3] Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

[4] Bay, H. and Tuytelaars, T. and Van Gool, L. SURF: Speeded Up Robust Features, 9th European Conference on Computer Vision, 2006

[5] Marius Muja, David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, 2009

[6] Paul Viola, Michael J. Jones, Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 2004.

[7] P. Viola ,M.J. Jones,Rapid Object Detection using a Boosted Cascade of Simple Features, proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001

[8] The OpenCV Documentation.// `http://docs.opencv.org/2.4/`

[9] Krasilnikov N. N. Digital processing of 2D- and 3D-images: tutorial. St.P.: BHV-Petesburg, 2011. 608 p.: img. (Textbooks for high schools) ISBN 978-5-9775-0700-4

[10] A. Cesetti E. Frontoni A. Mancini P. Zingaretti S. Longhi //A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks// Springer Science + Business Media B.V. 2009