Vision-based Control of a Quadrotor for an Object Inspection Scenario

Koppány Máthé, Lucian Buşoniu, László Barabás, Cristea-Ioan Iuga, Liviu Miclea, Jens Braband

***Abstract*— Unmanned aerial vehicles (UAVs) have gained special attention in recent years, among others in monitoring and inspection applications. In this paper, a less explored application field is proposed, railway inspection, where UAVs can be used to perform visual inspection tasks such as semaphore, catenary, or track inspection. We focus on lightweight UAVs, which can detect many events in railways (for example missing indicators or cabling, or obstacles on the tracks). An outdoor scenario is developed where a quadrotor visually detects a railway semaphore and flies around it autonomously, recording a video of it for offline post-processing. For these tasks, we exploit object detection methods from literature, and develop a visual servoing technique. Additionally, we perform a thorough comparison of several object detection approaches before selecting a preferred method. Then, we show the performance of the presented filtering solutions when they are used in servoing, and conclude our experiments with evaluating real outdoor flight trajectories using an AR.Drone 2.0 quadrotor.**

***Index Terms*— UAV, quadrotor, object inspection, visual servoing, object detection**

I. Introduction

Lightweight unmanned aerial vehicles (UAVs) are becoming widespread in various fields such as aerial imaging [1], entertainment, or inspection and monitoring [2], [3], [4]. They are in the focus of many research groups, companies and governmental institutions.

We focus on visual inspection applications that require the observation of objects, an application domain where UAVs can present a significant benefit. Some of the common visual inspection fields are bridge [5], power line [6], pipeline [7] or building façade [8] inspection, where UAVs have already been shown to be useful tools. Traditionally, inspection tasks commonly require considerable expenses and might need to be performed in difficult-to-access or dangerous areas. In many situations, UAVs can take over the visual inspection part of the tasks, reducing the need for human intervention, and therefore save a significant amount of resources.

A less explored application field is railways, where various components of the railway infrastructure must be inspected, including tracks, catenary, poles, semaphores, etc. Here, a further advantage of using UAVs is that traffic does not have to be stopped, unlike in the case of inspection by humans. Working with lightweight platforms does not present safety issues for traffic [9], as the UAV can get off the tracks depending on the schedule or by visually detecting an incoming train, and if the UAV is light enough (under 5 kg) a potential collision is non-damaging to the train.

The objective is to use UAVs to autonomously record videos of targets such as poles, tracks, etc. with a given amount of details. In this paper, we focus on inspecting semaphores, a target for which details captured at several meters already provide significant data (for example lamps not working, or missing indicators). The detection of such details does not require precise navigation, instead the target only has to be kept in the field of view from a rough distance. Therefore, even lightweight platforms, despite their sensitivity for example to wind gusts, are suitable for such inspection tasks.

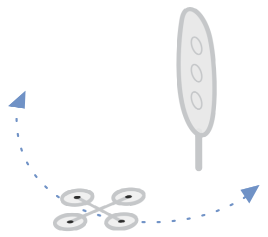
Fig. 1. Inspection scenario: fly-around

We select the AR.Drone quadrotor [10] and consider a scenario where the quadrotor autonomously navigates from a startup location to a railway semaphore. After detecting it, the quadrotor flies around the target and records a video of it for offline post-processing. We consider the last two stages of this scenario, namely visual detection of the target and the inspection flight, and assume that the application starts with the target in the neighborhood of the quadrotor. Flying to the target can be solved by GPS-based navigation, assuming no dynamic, unmapped obstacles have to be avoided. The inspection flight consists of recording a video of the target while keeping a reference distance to it and flying on an arc of $\pm 45°$ around it. The entire flight is performed autonomously.

The key parts of the approach are object detection and servoing. For detection, several methods can be used, such as feature detectors [11], classifiers [12] or template matching [13]. This paper focuses on detailed comparison of these detection methods in order to select the approach that performs the best in our scenario. On the servoing side, a simple

K. Máthé (koppany.mathe@aut.utcluj.ro), L. Buşoniu, C.-I. Iuga and L. Miclea are with the Automation Department, Technical University of Cluj-Napoca, Romania. L. Barabás is with the Mathematics Department, Babeş-Bolyai Univesity, Cluj-Napoca, Romania. J. Braband is with the Institute of Railway Systems Engineering and Traffic Safety, Technical University of Braunschweig, Germany.

This paper is supported by a grant from Siemens, Reference No. 7472/3202246859; by the Sectoral Operational Programme Human Resources Development (SOP HRD), ID 137516 (POSDRU/159/1.5/S/137516), financed by the European Social Fund and by the Romanian Government; and by a grant from the Romanian National Authority for Scientific Research, Project No. PNII-RU-TE-2012-3-0040. We are grateful to the national railway company (CFR) for offering us the semaphore panel for performing our experiments.

control strategy is developed that computes distances based on the object image and applies linear and extended Kalman filtering on the distances to obtain accurate values for the flight commands. The quadrotor keeps then a constant lateral velocity and, based on visual object detection, centers the target in the image frame and corrects the distance to it. The entire algorithm runs on a ground station that sends high-level velocity commands via Wi-Fi to the quadrotor. All these methods are tested in real, outdoor flights, using our experimental setup.

Plenty of works exist that use visual servoing for UAVs [2], [14], [15]. We do not intend to compete with these works, but rather focus on providing experimental results for a less explored application field, namely railway inspection. As also overviewed in our recent survey [16], UAV inspection applications mainly focus on power line inspection [3], [17], [6] and building monitoring [8], [18], whereas the area of railways is currently mainly discussed by companies [19], [5], [20]. Although the knowledge base is partly transferable between these application domains, experimental results are required to convincingly demonstrate the benefits of UAVs in a new domain. This is important as application results are subject to numerous conditions such as the sensor kit used and other platform parameters, or whether the tests were conducted indoors or outdoors. Except for our preliminary work [21], we found one single article [22] that deals with UAV experiments in railways. Both of these papers present only simulation results, without real flight experiments. Other articles such as [23], [24] deal only with the vision tasks from railway inspection, without using UAVs as inspection tools. In this context, making public our real outdoor flight tests and evaluations for railway inspection purposes will be useful for both the research and industrial communities.

The paper is structured as follows. Section II presents in general the methods used in the application. Section III discusses our method selection and provides implementation details. Section IV presents the experimental setup and test results and evaluation, and Section V concludes the paper.

II. Background: Object Detection, Visual Servoing, and Filtering Methods

This section introduces the methods used in the application. Several existing object detection techniques are briefly presented, namely feature detection, edge detection, machine learning and template matching-based approaches. We then briefly outline visual servoing concepts and the principles of signal filtering by probabilistic estimators.

A basic solution for object detection is the use of *feature detectors* in combination with descriptors, matching methods and transformation identifiers. A reference image (cropped to the object to be detected) is taken and features from it are matched to features from the scene image. Based on the set of matched features, the transformations between them can be determined to find the position and orientation of the reference image in the scene image. For this approach, one of the most common feature detectors is the method called features from accelerated segment test (FAST) [11]. For most feature types, the detected features require a description method that then enables the matching of the features. For feature descriptor creation, a preferred solution is the scale-invariant feature transform (SIFT) [25]. Then, feature matching is commonly achieved using the fast library for approximate nearest-neighbor (FLANN) [26] method. Finally, transformation identification can be performed with homography transforms [27]. Object detection with feature detectors offers accurate results, however it usually requires longer processing time than, for example, the classifier or template matching methods presented below. Also, the detection rate highly depends on proper tuning, and the methods are known to be sensitive to changes in, for example, light conditions or object pattern.

The above approach can be further combined with *pre-processing*. For example, the images can be transformed to edges before the features are detected. For this purpose, it is possible to use the Canny edge detectors [28]. Working with edges is most suitable for clear background scenarios (e.g. sky background or objects on a desk).

Feature detection can be combined with machine learning, which offers the possibility of taking a set of training data (features, in our case) and obtaining a better description of reference objects through optimization methods. Here, a well-known approach that can also be applied to online processing comes from Viola and Jones [12], and works with a cascade of weak feature classifiers. Two common feature types used with these classifiers are the Haar-like features [29] and linear binary pattern (LBP) [30], where the latter is known to be faster than the Haar-like features, though offering less precise detection. After an offline training of the classifiers with positive (containing the object to detect) and negative (background) examples, these *cascade classifiers* can be applied in online detection. Classifiers are primarily meant for detecting the presence of an object, and, compared for example to feature detection-based object detection, are less precise in correctly determining the size of the object.

A different approach is *template matching* [13], [31] where the entire reference image (template) is sought in the new frame based on various matching metrics. The template is slid over the scene image, similarity metrics such as the sum of squared differences or correlation coefficients are calculated, and the region with the best score is selected as the matching region. Template matching is usually employed when the template is significantly smaller than the scene image. Also, the basic form of the method will not rescale the template. Scale- and rotation-invariant adaptations already exist, like those from [32], [33], though a simple implementation of multi-scale matching is to run the algorithm multiple times for manually rescaled scene images. Template matching is known to be fast compared to the above methods, as it uses less sophisticated comparisons.

Visual servoing deals with controlling the displacement of the vehicle based on visual information, so as to obtain a desired trajectory. It requires some source of distance or position information from the environment. Based on them, geometric transformations are applied in order to determine the position of the vehicle or of other objects. Then, various control techniques guide the vehicle through a desired flight trajectory. Classic approaches are position-based and image-based visual servoing [34], [35]. Also, plenty of object tracker methods and implementations exist [36], [37], which can be used for UAV visual servoing. We implement a simple servoing logic, using the principles of image-based visual servoing, selecting a method from the object detection approaches presented before.

Distance information can be gathered from object detection results. However, these distance measurements are usually noisy. As this signal is the input of the flight control, it should be filtered in order to avoid chattering inputs. In general, not all interesting states are directly accessible via sensors, and unmeasured states must be estimated from the measured data. *Kalman filters* [38] are popular methods that estimate the signal evolution accounting for the dynamics of the system, and for noises influencing both the system evolution and the measurements. In its basic form, the Kalman filter (KF) accounts for linear system dynamics, while the extended Kalman filter (EKF) accepts nonlinear models as well, and linearizes the model at every estimation step. In our application, these filters can be used for integrating the distance calculation formulae and obtaining a smooth distance signal, filtering out noise and mitigating faulty measurements.

III. Proposed approach

This section lists the object detection methods that will be evaluated, and details the visual servoing approach proposed by us based on these alternatives.

A. Selected object detection methods

The four object detection approaches presented in Section II are evaluated based on image sets from our application. All the selected methods come with OpenCV implementations [39]. The parameter settings and evaluation results are presented in Section IV-A. The evaluation is performed in order to select a suitable method for our application, rather than to achieve an exhaustive comparison of the discussed detection methods. Nevertheless, a discussion is provided on the performance of the different algorithms as a function of different parameters, as presented in Section IV-B. The method having the best overall results will be used in visual servoing.

For feature matching-based detection, our earlier results from [16] are used, where the experimental setup had similar scenes to the ones from the current application. Based on the method evaluations from [16], we select the FAST feature detector in combination with the SIFT descriptor creator, FLANN matching and homography-based transformation detection.

For edge detection preprocessing for feature matching, the Canny algorithm is selected, one of the most popular methods, which can be quickly and easily tuned.

Classifier-based detection is performed using both Haar-like and LBP features. The classifier methods may find multiple matches in an image. The cascade search process evaluates these matches in multiple stages, discarding at each stage several matches. At the end, the match that reached the furthest stage is selected. Although the correctness of this criterion is not formally proven, experiments show that it provides good selection of the best match.

Template matching is tested with all the available similarity metrics, namely: sum of squared differences, correlation, and correlation coefficients. We develop a simple scale-invariant version of the method from an existing implementation that does not rescale the images. Scale-invariant detection is obtained by running the matching with the scene image resized at several scales, and selecting the best match from all the iterations.

The detection methods provide a bounding rectangle around the detected object, which defines its size and position in the scene image. This information is then used in performing visual servoing, as described next.

B. Visual servoing: distance calculation and trajectory tracking

Once the target object is detected, control is implemented based on a simple servoing logic: the quadrotor flies with a constant lateral velocity while rotating and keeping a desired distance to the detected target. The resulting trajectory is an arc with the target in its center. This is achieved by the following steps: raw distances are computed using the result of detection; the distances are filtered to eliminate noise; and the filtered distances are compared with the desired distances and fed to a proportional (P) controller that calculates the required velocity commands.

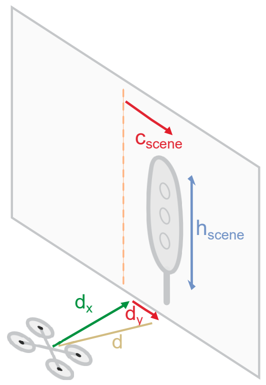

Fig. 2. Distance measurement

The first component of visual servoing is the *distance calculation*. Using the detection methods presented in Section II, a bounding rectangle is calculated around the detected object. Then, two parameters of this rectangle are used: its height in the image frame (in pixels), denoted h_{scene}, and its lateral distance (in pixels) from the center of the image, denoted c_{scene}. Denote the distance between the quadrotor and the plane of the inspected target as d_x. Also, denote the lateral distance (to the right from the center of the image frame) as d_y. Both d_x and d_y are measured in meters, and are used in obtaining the real-world distance between the quadrotor and the target, denoted d. Distances d_x and d_y are obtained from h_{scene} and c_{scene} as:

$$\begin{aligned} d_{x\text{raw}} &= \tfrac{h_{\text{ref}}}{h_{\text{scene}}} \cdot d_{\text{ref}} = \tfrac{a_x}{h_{\text{scene}}} \\ d_{y\text{raw}} &= \tfrac{h_{\text{ref}}}{h_{\text{scene}}} \cdot \tfrac{d_{\text{ref}} \cdot c_{\text{scene}}}{432} = \tfrac{a_y \cdot c_{\text{scene}}}{h_{\text{scene}}} \end{aligned} \tag{1}$$

where h_{ref} and d_{ref} denote the height (in pixels) of the reference image, and respectively the real-world distance at which the reference image was taken. The logic behind these formulae is simple: if $h_{\text{scene}} = h_{\text{ref}}$, the quadrotor is at the same d_x distance as at which the reference image was taken, i.e. $d_{x\text{raw}} = d_{\text{ref}}$. The distance decreases as h_{scene} increases. A similar logic applies to the lateral distance $d_{y\text{raw}}$, where the scaling factor 432 results from empirical tuning and is used for obtaining the real-world lateral distance from c_{scene} and d_{ref}.

The obtained distances are filtered as presented in Section III-C. Filtering is discussed separately as it is not mandatory for the servoing logic, however it improves the accuracy of distance estimation.

Finally, a P controller takes as input the filtered distances and some reference distances $d_{x\text{ref}}$ and $d_{y\text{ref}}$ and calculates control velocities to reach the reference values:

$$\begin{aligned} v_x &= P_x \cdot (\sqrt{{d_x}^2 + {d_y}^2} - d_{x\text{ref}}) \\ v_{\text{rotz}} &= P_y \cdot (d_y - d_{y\text{ref}})/d_x \end{aligned} \tag{2}$$

with P_x and P_y controller parameters. While P controllers may cause steady state errors if used exclusively, the reference values calculated by the controller above are inputs for the onboard control unit of the quadrotor, which applies further control loops to meet the reference values. Thus, this simple controller is sufficient to obtain proper velocities.

Note that the rotational velocity is influenced by the d_x distance, reducing the applied rotational command as the quadrotor moves further from the target. We have observed in our experiments that this offers more stable operation.

Besides the above controller used to correct the position and orientation of the quadrotor, a constant lateral velocity is applied to the vehicle. Also, the flight altitude is kept at a constant reference value by the internal controllers of the platform. In this manner, the quadrotor is expected to follow an arc path around the target object, continuously keeping it in the center of the field of view. The flight direction is changed based on angle measurements obtained from the inertial measurement unit of the vehicle: when the quadrotor reaches a turning of 45° compared to its takeoff position, the sign of the applied lateral velocity is changed.

This visual servoing can be used for UAV platforms that have a frontal camera, can receive high-level linear and rotational velocity setpoints as commands, and can provide navigation data such as velocities read on linear and rotational axes. We select the AR.Drone 2.0 quadrotor [10] that supports all these features, and use it in our experiments. The precision of the distance calculations and the performance of the resulting controller are evaluated in Section IV-B.

C. Signal filtering

The raw distances are filtered in order to mitigate noise and wrong measurements resulting from incorrect detections. Here two filtering alternatives are proposed, both of them based on Kalman filters: one solution, that we call 2-KF, where the d_x and d_y distances are filtered separately, using linear Kalman filters (KF); and another approach where the measurement model from (1) is integrated into an extended Kalman filter (EKF), i.e. taking into account the coupling between the evolution of the two distances.

The linear models used for the 2-KF take both the state x and the measurement y as the distance on the given axis (d_x or d_y), and have the following form:

$$\begin{cases} d_+ = d + v \cdot T_s + z \\ y = d + w \end{cases} \tag{3}$$

where d_+ marks the new discrete state of the process, d the distance on the given axis, v the corresponding velocity input, and T_s the sampling time of the process. Variables z and w are the process and measurement noises, described by normal distributions $\mathcal{N}(0, Q)$ and $\mathcal{N}(0, R)$, with Q and R as tuning parameters.

The EKF filter is based on (1), taking as state variables the two distances, $X = [d_x, d_y]^T$, as measurement variables $Y = [h_{\text{scene}}, c_{\text{scene}}]^T$, the height and lateral position of the object in the scene image, and as input the velocities on the two axes, $U = [v_x, v_y]^T$. Then, EKF uses the following nonlinear model:

$$\begin{cases} \begin{bmatrix} d_{x+} \\ d_{y+} \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + T_s \cdot \begin{bmatrix} v_x \\ v_y \end{bmatrix} + z \\ \begin{bmatrix} h_{\text{scene}} \\ c_{\text{scene}} \end{bmatrix} = \begin{bmatrix} a_x / d_x \\ a_x \cdot d_y / (a_y \cdot d_x) \end{bmatrix} + w \end{cases} \tag{4}$$

where $a_x = h_{\text{ref}} \cdot d_{\text{ref}}$ and $a_y = h_{\text{ref}} \cdot d_{\text{ref}}/432$ are constants from (1). This model combines the evolution of both distances, and is thus expected to provide a more realistic filtering.

The tuning parameters of the filters are Q and R, the process and measurement noise covariances; and the sampling time of the process, T_s. The selected parameter values are presented in Section IV-B.

If no distance measurement is available, the filters work in prediction mode, estimating the distances from the model and from the velocity inputs.

IV. Method evaluation and flight tests

In the sequel, we first evaluate offline, on test images, the detection methods presented, and use a sequence of images taken during a flight to analyze the distance measurement and filtering. Then, the online performance of the proposed visual servoing is tested with both filtering methods. The experiments are performed using the AR.Drone 2.0 quadrotor [10] and a laptop with a Core i7-4702MQ 2.2GHz processor and 8GB RAM. All processing is run on the laptop while sending high-level commands to the quadrotor for the flight tests.

A. Evaluating object detectors

The four detection approaches presented in Section II are evaluated using two sets of images. The first set consists of 100 different photos of a standard railway semaphore, see Fig. 3, a typical object that has to be inspected in the railway infrastructure. The target appears at distances varying from around 10 to 30 m, mostly in frontal view. This allows for checking the scale-invariant property of the methods. A second set of 200 frames is taken from our yard setup where only the panel of a semaphore is used, see Figs. 4 and 9. The light conditions and the background are different. Also, the target object has fewer interesting features. In this set, the target appears at distances between around 3 and 5 m, viewed from various side angles (but having the camera mostly parallel to the Earth). The different side views also check the rotation-invariant detection ability of the methods.

Both sets contain 15 frames of true negatives, i.e. images in which the target object is not present, but the background is similar to those in which it is. The first set contains 640×360 px images, whereas the frames from the second set are resized to 320×180 px size. All the frames are grayscale.

The above two image sets are useful as they present frame sequences from real scenarios. The detectors are evaluated on these sets, selecting a method that ensures overall good performance in real flights. Additionally, the detectors are tested based on different parameters, namely scale, light, view angle, and background invariance, as presented at the end of Section IV-A.

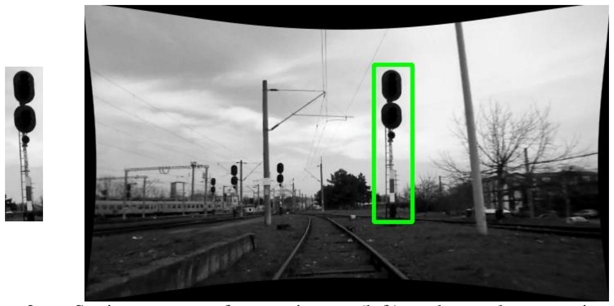

Fig. 3. Station setup: reference image (left) and sample scene image processed with template matching (right)

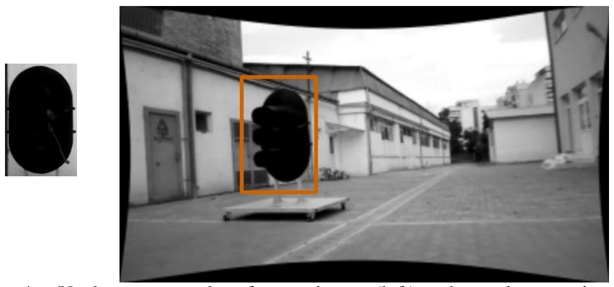

Fig. 4. Yard setup: sample reference image (left) and sample scene image processed with classifiers (right)

We use OpenCV 2.4.9 implementations [39] of the discussed methods, see the documentation of OpenCV for detailed description of the meaning of the parameters. The parameter settings result from preliminary tuning experiments conducted with each approach:

- For *feature matching*-based detection, the FAST feature detector is used with threshold 31, enabling non-max suppression; and using default parameters for the SIFT descriptor, FLANN matching and homography-based transformation detection.
- For *edge detection*-based object detection, the Canny algorithm is selected with hysteresis thresholds 100 and 200, and kernel size 3.
- The *classifiers* use different settings for the two types of objects. In both cases, the parameters presented next are used in the classifier training process [40]. For the station semaphore, the settings from [41] are used, creating 1500 positive samples of 30×120 px size with arbitrary rotations on the x, y and z axes up to $1.1, 1.1$ respectively 0.5 rad, and a maximum intensity deviation of 40. The classifiers are created over 20 training stages, searching for Haar-like features and using 1500 positive and 279 negative samples, a minimum hit rate of 0.999 and a maximum false alarm rate of 0.5. For the yard panel, 800 positive samples of 32×21 px are created with the same parameter settings. The best performance is obtained by training with linear binary pattern (LBP) feature types over 20 stages, working with 800 positive and 800 negative samples, a hit rate of 0.99 and a false alarm rate of 0.5.
- After several tests on the similarity metrics, *template matching* is performed using the correlation coefficient similarity evaluation method. A detection score threshold is manually set for each test, where, based on the reference image and the current light conditions, it is possible to determine a value below which the detections are false positives. For the station semaphore, this threshold is set to 450000, whereas for the yard panel the value 160000 is used.

Using these settings, the methods are evaluated on the two image sets. True distance information is collected by taking each frame and manually reading the height and the horizontal position of the target, if it is present in the image. Note that the manual readings may have errors, but these errors are small compared to those made by the detectors. Then, the detection methods are run and compared to the true distances. Tables I and II collect the results. For each method, the following indicators are checked:

- the *average size error* in percentage, which collects the detection height error relative to the true height. The average is calculated as the sum of the modulus of the distance errors over the number of image frames;
- the *average centering error* in pixel distance that represents the horizontal position error of the center of the detected object;
- the *number of false positives* indicating the number of frames in which the target was detected although it was not present in the frame, or had an error in the horizontal position compared to the true position. The

Method	Size err. (%)	Pos. err. (px)	False pos.	True neg.	Detect. rate (%)	Time (ms)
Feature	7.0	1.98	1	15	51	85.6
Edge	9.9	2.45	1	14	41	62.0
Classifier	11.7	1.63	0	14	99	35.0
Template	6.5	1.80	0	11	96	7.1

TABLE I

PERFORMANCE OF DETECTORS – STATION SETUP

position error tolerance is set to 80% of the width of the target for the station semaphore, and respectively 50% in the case of the yard panel, below which the detections are accepted as true positives;

- the *number of true negatives* identified correctly;
- the *detection rate* as a percentage, including true negatives;
- and the *average execution time* in milliseconds of a given method for processing a single frame. Note that, unlike the other methods, the classifiers require a long offline training (can last several hours), however this does not influence the online detection time.

Tables I and II show results when accurate size detection is not considered as a constraint, i.e the correctness of the detection is evaluated based only on the presence and the position of the target, and no sample is declared false positive based on size error.

Method	Size err. (%)	Pos. err. (px)	False pos.	True neg.	Detect. rate (%)	Time (ms)
Feature	0.8	0	0	15	8	33.1
Edge	7.5	3.10	0	15	10	23.4
Classifier	18.9	3.23	3	4	93	12.1
Template	14.4	7.41	0	15	68	8.1

TABLE II

PERFORMANCE OF DETECTORS – YARD SETUP

Looking at Table I, it can immediately be seen that the classifier and template matching methods outperform the other two. The former two detect almost all targets in the correct position. Also, the negative samples are almost all correctly identified, 14 out of 15 for classifiers and 11 out of 15 for template matching. In general, all of the methods have a low positioning error (below 3 px). This says that all the methods are able to correctly detect the position of the target, if detection is successful. The size error also reflects that, on average, all the methods detect the size of the target correctly, up to at most 12% error.

Table II confirms the previous results for the yard setup. One immediate observation is that, except for template matching, all the methods are faster, due to the smaller images used. The size detection and positioning errors are somewhat greater, however still acceptable. The target object has fewer interesting features, which determines the lower overall detection rate. The lower detection rate appears also due to the fact that the target is captured from various side views, and the methods are meant primarily for detecting

	Scale		Light		View angle		Background	
Method	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
Feature	0	90	0	10	0	0	0	0
Edge	0	65	20	10	5	0	10	0
Classifier	100	100	100	100	100	100	100	100
Template	100	100	100	100	100	50	100	100

TABLE III

DETECTION RATE (%) OF OBJECT DETECTORS: (A) SMALL OR (B) LARGE SCALE TARGET; (C) BRIGHT OR (D) SHADOW TARGET; TARGET FROM (E) FRONTAL OR (F) LATERAL VIEW; AND (G) BUILDING OR (H) SKY BACKGROUND

	Scale		Light		View angle		Background	
Method	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
Feature	–	8.3	–	12.9	–	–	–	–
Edge	–	13.4	22.4	22.2	39.7	–	22.2	–
Classifier	19.8	7.2	19	34.2	23.3	18.5	29.7	13.9
Template	11.3	6.6	19.1	32.1	20.4	10.1	28.5	10.8

TABLE IV

SIZE ERROR (%) OF OBJECT DETECTORS: (A) SMALL OR (B) LARGE SCALE TARGET; (C) BRIGHT OR (D) SHADOW TARGET; TARGET FROM (E) FRONTAL OR (F) LATERAL VIEW; AND (G) BUILDING OR (H) SKY BACKGROUND

the objects from the frontal view. However, the classifier and template matching methods still have a very good detection rate. An important remark is that in this second set the classifier method detects only four true negatives whereas the template matching method is able to identify all of them correctly. Also, template matching has less false positive detections. The perfect true negative detection in the case of the other two methods can be explained with the very low detection rate: the edge and feature matching methods are almost unable to detect the target in this set of images, therefore they have no detection even in the case of true negatives.

The detectors are further compared for scale, light, view angle and background invariance using subsets of the two image sets. Table III presents the detection rate of the methods, while Table IV shows the detection size errors in percentage. Both tables contain four pairs of columns, one for each tested parameter. For each parameter, 20 to 40 image frames are taken to test the performance of the methods in two extremes: for example, we evaluate the detectors using frames with the target appearing in frontal and in lateral view separately. Therefore, the column pairs tell how sensitive each method is to a given parameter, and which is the preferred parameter setting. The evaluation is performed based on a small number of samples, therefore limited – but still useful – conclusions can be drawn.

Table III indicates that with the chosen subsets of images the feature detectors and edge detectors fail to detect the target most of the time. On the other hand, classifiers and template matching almost always succeed. An exception is template matching for targets in lateral view, where the detection rate is only 50%. This indicates that using only frontal reference images, template matching is not able to perform rotation-invariant detection.

Summarizing the results from Table IV, in our setting, most of the methods have a more precise detection when the target appears at a large scale, is brighter, appears in lateral view, and has a more distinguishable sky background. Nevertheless, classifiers and template matching have good performance in less favourable detection circumstances as well.

Returning to the overall results of the two best-performing detectors, template matching is clearly faster than the classifiers, as can be seen from Table I. This is due to the fact that it requires only some matrix operations compared to the more complex classification procedure. However, in the case of the second set where the scene image size was reduced, the processing time of the classifiers becomes comparable, as presented in Table II. Both methods perform faster than the 30 Hz rate at which the frames are received in online flights. The more important indicator is the detection rate. While for the station setup the detection rates of the two methods are similar, the template matching performs worse with the second set. This is due to the varying appearance of the target, which is viewed also laterally. As also confirmed by Table III, template matching will fail often in lateral view. Therefore, due to the overall better detection rate, the *classifier*-based detection is selected to be used in visual servoing.

B. *Distance estimation and trajectory tracking results*

Based on the experimental evaluation from Section IV-A, classifier-based detection is selected for obtaining visual distance information. Then, the servoing logic discussed in Section III-B is used in performing inspection flights. The two distance filtering methods are evaluated offline on a set of recorded measurements from a real flight trajectory. Later, both filtering methods are applied online, in separate flights, in combination with control, and the flight performances achieved are compared.

The distance calculation is performed as presented in Section III-B, based on the detection results for each sample. For the 2-KF approach, the d_x filter uses $Q = 0.048, R = 20$, while the d_y filter has set $Q = 0.43, R = 6$. Both filters use a sampling time of $T_s = 0.04$ s, which is selected to be greater than the processing time required for the object detection and distance calculation. The EKF filter is tuned with $Q = \sigma \cdot [\delta^3/3, \delta^2/2; \delta^2/2, \delta], R = [10, 0; 0, 10]$ with $\sigma = 10$ and $\delta = 0.02$, and $T_s = 0.02$ s. The sampling time is set according to the rate at which the velocity readings are obtained. For both filters, Q and R are empirically tuned. Figs. 5 and 6 show raw data and filtering results for a set of 372 samples taken from a real flight.

Looking at Fig. 5, the raw measurements contain a 0 measurement at about sample 300, indicating that the target was not found, and some false positives with distances above 8 m. These outliers and all the noises are correctly filtered out by both filters. Also, it may be noticed that the EKF filter offers a smoother signal. The figure shows the true distance

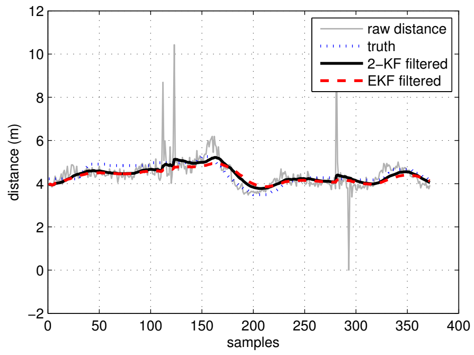

Fig. 5. Distance filtering on x axis

as well, recorded manually from the image samples. The filtered signals are well aligned with the true distances: the 2-KF-filtered signal has an average error of 0.20 m, while the EKF filter has 0.24 m average error.

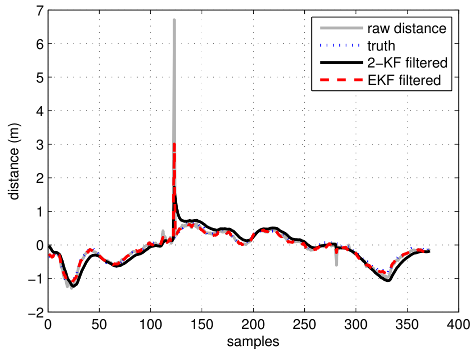

Fig. 6. Distance filtering on y axis

For the lateral distance shown in Fig. 6, the average errors are 0.12 m for the 2-KF filter and 0.05 m for the EKF filter. The difference appears mainly due to the slight delay of the 2-KF-filtered signal, which implies more errors. Also in this figure, the true distances are well estimated. However, the EKF filter has a lower filtering effect, as can also be seen from the false positive from at around sample 125.

The proposed visual servoing is evaluated in outdoor flights, showing repeatable, successful flights. Fig. 7 shows true flight paths (reconstructed from the image frames saved during the flight by calculating the distances to the target and taking into account the turning angle measurements) in the case of two flights, one with 2-KF filtering and another with EKF distance filtering applied. Both flights start at about 5 m in front of the target (coordinates $(-5, 0)$) and aim at keeping a 4 m distance to the target while flying back and forth on a $\pm 45°$ arc. The terminal positions of the trajectories are marked in green. Some further flights are shown in Fig. 8, confirming the repeatability of the autonomous inspection flights. Overall, tens of experiments were performed, with around 75% successful flights using the best settings.

Focusing on the trajectories shown in Fig. 7, the average distance-keeping error compared to the desired distance is 0.66 m when the EKF filter was used, and 0.71 m for the 2-KF-filtered measurements. Nevertheless, the difference

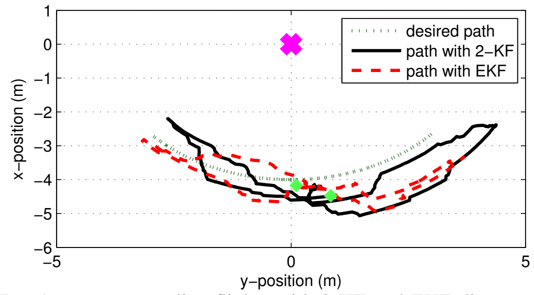

Fig. 7. Autonomous online flights with 2-KF and EKF distance filtering

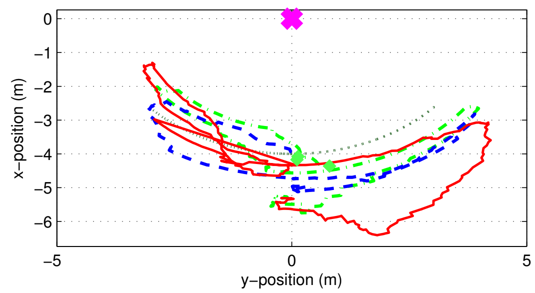

Fig. 8. Further online flight tests

between the true distance and the filter outputs is below 0.25 m for both filters. This means that, although the filters determine the true distance with a precision of 0.25 m, the controls applied by the quadrotor do not immediately correct the position and thus add further distance errors. For example, the used platform was tested and we observed that it has limitations in its internal controller regarding applying a precise control velocity, due to which a desired commanded trajectory will have further errors. Also, distance errors may appear due to the outdoor operating conditions as well, where small wind disturbances influence the flight trajectory. On the other hand, further errors may appear due to the internal sensing system of the low-cost quadrotor, and due to the manual true distance calculation. Taking into account these issues, the obtained trajectories are reasonable. The differences between the two trajectories are not significant, both methods offering similar results that are suitable for the discussed application. Fig. 9 shows sample frames captured during the inspection flight, with the detected target marked by a bounding rectangle. A video of a typical flight can be downloaded from here (please use an offline player): `http://rocon.utcluj.ro/files/semaphore_inspection_flight.mp4`.

V. Conclusions

In this paper, we presented implementation and evaluation results for a railway inspection scenario, where a quadrotor must perform visual detection of a target object and inspect it by flying around it on an arc. Four detection approaches were evaluated and a classifier-based object detection was selected to be used in visual servoing. Distances were obtained from image data. Two filtering approaches were used to smoothen the obtained distances, one that accounts for the target plane distance and the lateral distance of the target

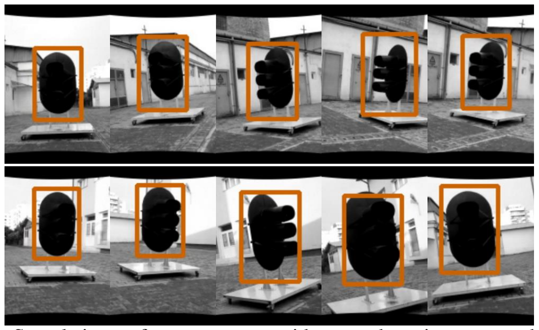

Fig. 9. Sample image frame sequence with target detection, captured during an autonomous flight of the quadrotor

separately, using simple Kalman filters, and another that combines the distances in an extended Kalman filter. Finally, online flight tests were conducted to evaluate the solution, using both distance-filtering methods. The resulting distance keeping and trajectories perform sufficiently well for the given application.

The most important open issue is obstacle avoidance for safer navigation. Also, we will work on improving object detection by obtaining better classifiers, and also by testing template matching with multiple reference images, so as to make the detection work from multiple view angles of the target. Additionally, we will also complete the scenario with GPS waypoint navigation.

References

[1] "DJI products," Available online: http://www.dji.com/products, (accessed on 26-01-2016).

[2] I. Sa, S. Hrabar, and P. Corke, "Outdoor flight testing of a pole inspection UAV incorporating high-speed vision," in *In Proceedings of the International Conference on Field and Service Robotics, Brisbane, Australia*. Springer, 9-11 December 2013, pp. 107–121.

[3] S. Montambault, J. Beaudry, K. Toussaint, and N. Pouliot, "On the application of VTOL UAVs to the inspection of power utility assets," in *Proceedings of the IEEE 1st International Conference on Applied Robotics for the Power Industry (CARPI), Montreal, QC, Canada*, 5–7 October 2010, pp. 1 – 7.

[4] T. Özaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Inspection of penstocks and featureless tunnel-like environments using micro UAVs," in *In Proceedings of the International Conference on Field and Service Robotics, Toronto, Canada*. Springer, 24–26 June 2015, pp. 123–136.

[5] "SNCF using drones for railway viaduct inspection," Available online: http://www.uasvision.com/2013/11/13/sncf-tests-infrastructure-inspection-by-uas/, (accessed on 26-01-2016).

[6] L. F. Luque-Vega, B. Castillo-Toledo, A. Loukianov, and L. E. Gonzalez-Jimenez, "Power line inspection via an unmanned aerial system based on the quadrotor helicopter," in *Proceedings of the IEEE 17th Mediterranean Electrotechnical Conference (MELECON), Beirut, Lebanon*, 13–16 April 2014, pp. 393 – 397.

[7] D. Hausamann, W. Zirnig, G. Schreier, and P. Strobl, "Monitoring of gas pipelines – a civil UAV application," *Aircraft Eng. Aerosp. Technol.*, vol. 77, no. 5, pp. 352 – 360, 2005.

[8] V. Baiocchi, D. Dominici, and M. Mormile, "UAV application in post-seismic environment," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XL-1 W*, vol. 2, pp. 21 – 25, 2013.

[9] S. G. Kiank, "Einsatz eines unbemannten Luftfahrsystem als Aufklaerer im Bahnbetrieb," Siemens Braunschweig, Tech. Rep., March 2013.

[10] "Parrot AR.Drone," Available online: http://ardrone2.parrot.com/ar-drone-2/specifications/, (accessed on 26-01-2016).

[11] E. Rosten and T. Drummond, “Rapid rendering of apparent contours of implicit surfaces for real-time tracking,” in *Proceedings of the 14th British Machine Vision Conference (BMVC), British Machine Vision Association (BMVA), Norwich, UK*. British Machine Vision Association (BMVA), 9–11 September 2003, pp. 719–728.

[12] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, Hawaii USA*, vol. 1, 8-14 December 2001, pp. 511–518.

[13] B. Zitova and J. Flusser, “Image registration methods: a survey,” *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[14] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, “Learning monocular reactive uav control in cluttered natural environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, 6–10 May 2013, pp. 1765 – 1772.

[15] G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, “Experimental validation of dynamic visual servoing for a quadrotor using a virtual camera,” in *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA*, 9–12 June 2015, pp. 1231–1240.

[16] K. Máthé and L. Buşoniu, “Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection,” *Sensors*, vol. 15, no. 7, pp. 14 887–14 916, 2015.

[17] C. Martinez, C. Sampedro, A. Chauhan, and P. Campoy, “Towards autonomous detection and tracking of electric towers for aerial power line inspection,” in *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA*, 27–30 May 2014, pp. 284 – 295.

[18] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, “A UAV system for inspection of industrial facilities,” in *Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA*, 2–9 March 2013, pp. 1 – 8.

[19] “DB about using drones against graffiti sprayers,” Available online: http://www.bbc.com/news/world-europe-22678580, (accessed on 26-01-2016).

[20] “ELIMCO industrial aerial inspection solutions,” Available online: http://www.elimco.com/eng/l_Infrastructure-Monitoring-and-Inspection_46.html, (accessed on 26-01-2016).

[21] E. Páll, K. Mathe, L. Tamas, and L. Busoniu, “Railway track following with the AR. Drone using vanishing point detection,” in *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 22–24 May 2014, pp. 1–6.

[22] C. Jarrett, K. Perry, and K. Stol, “Controller comparisons for autonomous railway following with a fixed-wing UAV,” in *Proceedings of the 6th International Conference on Automation, Robotics and Applications (ICARA), Queenstown, New Zealand*, 17–19 February 2015, pp. 104–109.

[23] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi, and L. Chen, “Automatic fastener classification and defect detection in vision-based railway inspection systems,” *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 4, pp. 877–888, 2014.

[24] B. Yang and L. Fang, “Automated extraction of 3-D railway tracks from mobile laser scanning point clouds,” *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 7, no. 12, pp. 4750–4761, 2014.

[25] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91 – 110, 2004.

[26] M. Muja and D. G. Lowe, “Fast matching of binary features,” in *Proceedings of the Ninth Conference on Computer and Robot Vision (CRV), Toronto, ON, Canada*, 28–30 May 2012, pp. 404–410.

[27] E. Dubrofsky, “Homography estimation,” Ph.D. dissertation, University of British Columbia (Vancouver), Canada, March 2009.

[28] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*. McGraw-Hill New York, 1995, vol. 5.

[29] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Proceedings of the Sixth international conference on Computer vision, Bombay, India*, 4-7 January 1998, pp. 555–562.

[30] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[31] T. Wu and A. Toet, “Speed-up template matching through integral image based weak classifiers,” *Journal of Pattern Recognition Research*, vol. 1, pp. 1–12, 2014.

[32] Y.-H. Lin and C.-H. Chen, “Template matching using the parametric template vector with translation, rotation and scale invariance,” *Pattern Recognition*, vol. 41, no. 7, pp. 2413–2421, 2008.

[33] H. Y. Kim, “Rotation-discriminating template matching based on Fourier coefficients of radial projections with robustness to scaling and partial occlusion,” *Pattern Recognition*, vol. 43, no. 3, pp. 859–872, 2010.

[34] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice, “Position based visual servoing: keeping the object in the field of vision,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA*, vol. 2, 11–15 May 2002, pp. 1624–1629.

[35] F. Chaumette and E. Malis, “2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA*, vol. 1, 24–28 April 2000, pp. 630 – 635.

[36] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Häger, G. Nebehay, and R. Pflugfelder, “The visual object tracking VOT2015 challenge results,” in *Workshop on the VOT2015 Visual Object Tracking Challenge*, 2015.

[37] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Cehovin, “A novel performance evaluation methodology for Single-Target trackers,” *Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, p. 1, 2016, to appear. [Online]. Available: http://dx.doi.org/10.1109/tpami.2016.2516982

[38] B. Ristic, S. Arulampalam, and N. Gordon, “Beyond the Kalman filter,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 7, pp. 37–38, 2004.

[39] “OpenCV,” Available online: http://docs.opencv.org/2.4.9/, (accessed on 26-01-2016).

[40] “OpenCV cascade classfier training,” Available online: http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html, (accessed on 20-01-2016).

[41] C.-I. Iuga, “Vision-based quadrotor navigation around objects,” Diploma Thesis, Technical University of Cluj-Napoca, June 2015.