Railway Track Following with the AR.Drone Using Vanishing Point Detection

Előd Páll, Koppány Máthé, Levente Tamás, Lucian Buşoniu

***Abstract*— Unmanned aerial vehicles are increasingly being used and showing their advantages in many domains. However, their application to railway systems is very little studied. In this paper, we focus on controlling an AR.Drone UAV in order to follow the railway track. The method developed relies on vision-based detection and tracking of the vanishing point of the railway tracks, overhead lines, and other related lines in the image, coupled with a controller that adjusts the yaw so as to keep the vanishing point in the center of the image. Simulation results illustrate the method is effective, and are complemented by vanishing-point tracking results on real images.**

***Index Terms*— Unmanned aerial vehicle, vanishing point, flight control, railway system.**

I. Introduction

The usage of unmanned aerial vehicles (UAV) in civilian applications is increasingly being investigated. Famous recent examples include online giant Amazon's research into using UAVs for package delivery [1], and Deutsche Bahn's exploration of UAVs to curb graffiti spraying [2], but more classical applications have long been considered, such as search and rescue [5], [23]. In this wider context, our end-goal is developing automated UAV-based procedures for railway surveillance and maintenance, and is therefore related to [2]. UAV-based inspection aims to be a low-cost alternative that does not require stopping the rail traffic, and can work in areas not easily accessible to human operators.

As a first step, in this paper we present a technique to control a UAV so that it automatically follows the rail track. We focus on the AR.Drone, a low-cost, lightweight UAV widely used in robotic research [14], [22], [6]. Our method relies on two main components. First, image processing is used to detect the vanishing point (VP) in the image, and Kalman filtering is applied to track the VP over subsequent frames in the presence of noise. The second component is the controller, which uses a PD regulator measuring the VP displacement to adjusts the yaw of the drone and keep the VP in the center of the image. Together with a constant forward velocity, this leads to the rail track being followed.

Vanishing point detection is based on edge detection with Laplacian filtering [20] and line detection is performed with the probabilistic Hough Transformation [13]. We additionally apply a selection procedure to eliminate unneeded lines, due to the building edges. Then, the densest area of crossing points between the detected lines has a high probability to be the VP of the rail tracks, and is therefore used as the observation input in the Kalman filter.

We present simulation results investigating the performance of the VP tracking as well as that of the overall controller, using Gazebo, a 3D simulation environment, integrated in the Robotic Operating System (ROS). The dynamical model of the AR.Drone [9] is already implemented and simulated in this environment, in which we additionally created a simulation of the railway and altered the start-up position and velocity to fit our scenario. We also present VP detection and tracking results on real images captured with the drone cameras. The real-time control of the drone is the first step of our future work.

The fields of control [12] and vision-based state estimation [21] for UAVs are very well developed, as well as robotic navigation [8], [7], [16]. For instance, close to our work are the lane marker detection method in [3], the VP-based road following method of [15], and vision-based object detection on railway tracks in [19]. Some early ideas were presented by [18] for the related problem of power line inspection. However, our work is one of the first to consider UAV in railways, and is novel especially in its focus on control. New features and challenges arise in this context and must be taken into account, such as the possibility to always rely on the presence of the track and overhead lines, or the problem of discriminating sleepers from the track.

Next, Section II briefly introduces the AR.Drone, its simulation environment, and the existing methods for line detection and filtering that we use. In Section III, we describe the technique developed, including VP detection, filtering-based tracking, and control. The experimental results are given in Section IV, and Section V concludes the paper.

II. Background

A. Hardware and Software

We are using the AR.Drone presented in Figure 1, a four-rotor, fixed-pitch aircraft. It has built-in orientation and altitude sensors, one camera facing to the bottom, one to the front, a micro controller, and a WiFi module. It is a commonly used quadrotor in research projects, the dynamic model of the drone is known [17]. The model has a twelve-dimensional state space, which consists of the linear and angular velocities and accelerations in the three dimensional space.

We are processing the images from the front camera, which has a wide angle lens, specifically 92^o. The resolution is 1280×720 pixels for recording, 640×480 pixels for WiFi

The authors are with the Department of Automation, Technical University of Cluj-Napoca, Memorandului 28, 400114 Cluj-Napoca, Romania (Pall.Elod@gmail.com, Koppany.Mathe@aut.utcluj.ro, Levente.Tamas@aut.utcluj.ro, Lucian.Busoniu@aut.utcluj.ro). The work of E. Páll, L. Buşoniu, and T. Levente was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PNII-RU-TE-2012-3-0040.

Fig. 1: Parrot AR.Drone

streaming and this frames are sent at a speed of 30fps. In spite of the high frame rate, the speed can drop in case the drone is destabilized, because the stabilization procedure has higher priority than the video broadcasting.

The drone is supplied with its own software, which is not designed to be modified, but different drivers are implemented in ROS to communicate with the drone via WiFi.

We used a 3D simulation tool integrated in ROS, called Gazebo. The state space model of the AR.Drone is implemented in tum-ardrone ROS package [11] and the 3D model is created in Gazebo. The same driver can be used to control and communicate with the drone, as with the real device. The Gazebo is easy to use for building 3D scenarios such as a railway track. Moreover, all the states ground truth values are known and we can add disturbances, for example wind.

B. Methodology

In this section, we review the methods from the literature that we employ and build our approach on.

1) Line detection: Line detection generally needs low-level pre-processing of the image such as: smoothing, sharpening, erosion, dilation, and edge detection. In this field many algorithms are used for edge detection like Canny, Sobel, and Laplace algorithms [20].

The Laplacian method is part of the gradient filters of edge detection, which also includes the Sobel method. It is based on the fact, when the first derivative of a function is at a maximum then the second order derivative is zero. An edge on a 2D image can bee seen as a jump of intensity between the two surfaces. Therefore, the Laplacian edge detector searches for zero crossings in the second order derivative of the image. The filter is applied on a gray-scale image and for the two dimensional function $f(x, y)$ representing the intensity, it can be written as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{1}$$

An approximation of this derivative is obtained by implementing a discrete convolution with a mask, given below:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

The Sobel method performs a 2D spatial gradient on the image, hence the high spatial frequency regions are highlighted. The Sobel algorithm uses two masks, one for the horizontal lines and one for the vertical lines. The two convolution masks calculate the approximations of the derivative along the horizontal and vertical directions on the image. The masks are shown below:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \tag{2}$$

The Canny algorithm is a multi-step method. First, it smooths the image with a Gaussian filter, and then finds the intensity gradient of the image by using the same masks as the Sobel algorithm. Finally, edge thinning and thresholding is applied.

Sharp and long edges can be seen as lines. The Probabilistic Hough Transformation (PHT) [13] is one of the most commonly used algorithms in perspective vision. The algorithm is based on the parametric representation of a line:

$$\rho = x \cos\theta + y \sin\theta$$

where ρ is the perpendicular distance from the origin to the line and θ is the angle between the horizontal axis and the this perpendicular.

The family of lines going through a given point (x_0, y_0) can be written as a set of pairs of (ρ_θ, θ). This set of lines can be represented as a sinusoidal, if $\rho > 0$ and $\theta \in (0, 2\pi)$. The algorithm searches intersections of sinusoidal curves. If the number of curves in the intersection is more than a threshold, then the pair of (ρ_θ, θ) is considered to be a line on the image. The algorithm takes a random subset of points for line detection, thus optimizing the procedure.

2) Estimation: The Kalman filter (KF) [10] is frequently applied in the mobile robotic field for position estimation and tracking. It is an optimal estimator when the dynamics are linear and the model and measurement noises are uncorrelated and have Gaussian distributions. In general the KF estimates the state $\mathbf{x} \in \mathbb{R}^n$ of a linear discrete-time system:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{y}_k &= \mathbf{H}\mathbf{x}_k + \mu_k \end{aligned} \tag{3}$$

where $\mathbf{F}$, $\mathbf{B}$, $\mathbf{H}$ are the system matrices, $\mathbf{y}$ is the measurement, $\mathbf{w}$ and μ are the process and measurement noises. These noises are assumed to be independent and Gaussian distributed, $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$ and $\mu_k \sim \mathcal{N}(0, \mathbf{R})$. The algorithm estimates the state $\mathbf{x}_k$ recursively and it has two phases: prediction and update. First, an initial state, $\mathbf{x}_0$ and initial covariance of the state, $\mathbf{P}_0$ is chosen.

In the prediction phase, KF calculates the prior state estimate, $\mathbf{x}_k^-$ and the prior error covariance, $\mathbf{P}_k^-$:

$$\begin{aligned} \mathbf{x}_k^- &= \mathbf{F}\mathbf{x}_{k-1}^+ + \mathbf{B}\mathbf{u}_k \\ \mathbf{P}_k^- &= \mathbf{F}\mathbf{P}_{k-1}^+ + \mathbf{Q} \end{aligned} \tag{4}$$

The update phase estimates the current state based on the prior estimate and the observed measurement, with a weighted average:

$$\begin{aligned} \mathbf{x}_k^+ &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k^-) \\ \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R})^{-1} \end{aligned} \tag{5}$$

where $\mathbf{x}_k^+$ is the posterior state estimate, $\mathbf{P}_k^+$ is the posterior error covariance, and $\mathbf{K}_k$ is the Kalman gain calculated at each step, so it minimizes the trace of the error covariance matrix. See [10] for further details, for example how to compute $\mathbf{K}_k$.

3) Control: The most commonly used industrial control method is the proportional-integral-derivative (PID) controller. The discrete time equation of this controller is:

$$\begin{array}{rcl} u_k & = & e_k \cdot K_p + \frac{e_k - e_{k-1}}{\delta_k} \cdot K_d + e_k^i \cdot K_i \\ e_k^i & = & e_{k-1}^i + e_k \cdot \delta_k \end{array} \quad (6)$$

where e_k is the current error, δ_k is the sampling time, e^i is the integrated error, and K_p, K_i, and K_d are respectively the proportional, integral, and derivative gains.

The tuning parameters are the three gains. In case the model of the system is unknown or poorly approximated, but a simulator or the real system is available for online tests, then the Ziegler-Nichols or the Åström-Hägglund [4] methods can be used to tune the regulator.

III. Approach

A. Image Processing

We are processing the images of the drone's front camera. The frames are analyzed based on perspective clues. In our case, the rail tracks viewed in perspective appear to converge to a point, called vanishing point. Recall from the introduction the idea of using line detection in the images in order to find the VP. In the detection phase of the project, we faced difficulties to eliminate the noises on the images taken in an outdoor environment, close to urban areas. We experienced higher noise on outdoor images than on indoor.

In order to reduce this noise and to prepare the image for further processing we blur the image and convert from color to gray-scale.

Next, we use an edge detection method, see Section II-B.1 on the pre-processed image. The result is sharpened by thresholding the image to enhance the intensity of the strong edges.

Afterwards, we search for lines with the Probabilistic Hough Transform method. The outdoor urban scenes have horizontal and vertical edges e.g. because of the surrounding buildings, and these lines are detected. Since these lines do not converge to the VP and are not useful in detecting it, they are filtered in order to have a more accurate VP detection. In order to make the detection robust, we separate out the lines based on their orientation angle: we neglect the lines which are approximately horizontal and vertical. The majority of the remaining lines are from the rail tracks, so the most dense area of crossing points of these lines will be the vanishing point of the tracks. The neglected lines' orientation angles, see Figure 2, are tuning parameters that can be changed if needed. We tuned these parameters from real case tests and these are: $\theta = 10^\circ$ and $\lambda = 10^\circ$.

B. VP Tracking

The vanishing point is tracked with the Kalman estimator, see Section II-B.2. In our case, ideally, a twelve dimensional state-space model and a mapping would be necessary between the 3D world model and the 2D plane of the image. This is impractical to use, so we chose a simplified model to describe the behavior of the vanishing point. This is the constant velocity model, presented below:

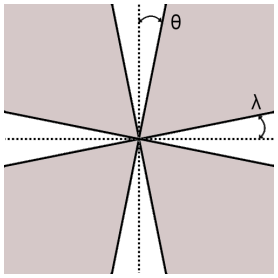


Fig. 2: The recognized lines with orientation angles between $\pm\theta$ and $\pm\lambda$ are neglected in the VP detection procedure.

$$\begin{cases} x_k & = & x_{k-1} + v_{k-1} \cdot \delta_k + w_{k-1}^x \\ v_k & = & v_{k-1} + w_{k-1}^v \end{cases} \quad (7)$$

where x_k is the current position of the vanishing point on the X axis measured in pixels,[1] v_k is the velocity of x_k, w_k is the process noise, and δ_k is the current sampling interval, more exactly it is the duration between video frame $k-1$ and k. Recall the observation of the variable frame rate, which can be caused by destabilization of the drone. Hence, the model covariance matrix (8), model transition matrix (7), and the PID controller (6) are influenced by the varying δ_k and must be recalculated for each k. The model presented above is a random walk model for the velocity, meaning we do not assume a dynamics for the velocity and just rely on measurements.

The KF can estimate past, present, and future states, hence it can be used when the vanishing point is not detected. If no observation can be done, the KF is used in open loop, which means that the prediction is used as estimation and the update phase is omitted.

The tuning parameters are obtained from different real scene flight tests. The chosen model and measurement covariance matrices are shown below:

$$Q = \begin{bmatrix} \delta_k^4/3 & \delta_k^3/2 \\ \delta_k^3/2 & \delta_k \end{bmatrix} \cdot \sigma_w^2; \; R = \sigma_v^2 \quad (8)$$

where $\sigma_w = \sqrt{70}$ and $\sigma_v = 10$ are the standard deviation of the process and the measurement. We obtained these tuning parameters from video feeds of real railway tracks.

C. Control

We want to maintain at zero the horizontal distance between the vanishing point and the center of the image. We implemented a PID control in order to achieve this. The controlled output is the VP position x, and the reference signal x_r is equal to half of the image width, 320 pixels. The control input is the yaw angular velocity while the drone is flying forward with a constant linear velocity, as shown in Figure 3.

[1]Note that x_k is just the first component of the two-dimensional state $\mathbf{x}_k$ in (4).

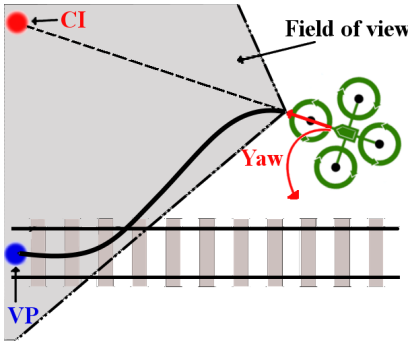


Fig. 3: Drone controlled with the yaw angle velocity, in function of the distance between the vanishing point (VP) and the center of the image (CI). By turning toward the VP and flying forward with a constant velocity, we navigate back to the rail track and remain there.

	P	PD	PID
K_p	0.004275	0.00684	0.00513
K_i			0.00128
K_d		0.00684	0.00513

TABLE I: PID controller gains tuned with Ziegler-Nichols closed loop method.

We tested the Ziegler-Nichols closed loop PID tuning method in the simulation and we found the ultimate gain, $K_u \approx 8.55 \cdot 10^{-3}$[px] and the ultimate period $T_u \approx 8$[sec]. The controller gains are calculated with the Ziegler-Nichols calculation formulas, see Table I.

Because of the poor experimental results with the above mentioned controller parameters, we implemented a grid-based PD tuning. The chosen set of parameters are $K_p \in [2 \cdot 10^{-3}, 9 \cdot 10^{-3}]$ and $K_d \in [1 \cdot 10^{-3}, 9 \cdot 10^{-3}]$. We measured the geometric mean of e_k for a 30 second flight and for each combination of (K_p, K_d). After repeating the test multiple times, we chose the pair with the lowest geometric mean, namely $K_p = 2 \times 10^{-3}$ and $K_d = 4 \times 10^{-3}$.

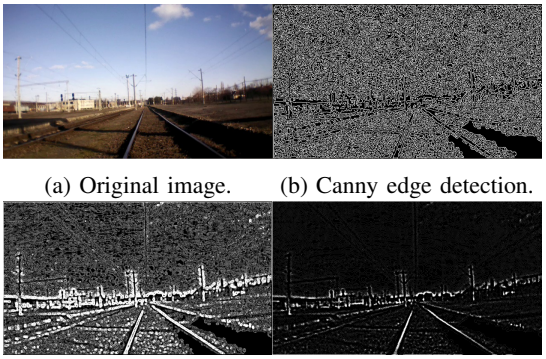
The reason we do not use the integrator of the PID controller is that the model already contains an integrator, from the yaw velocity to the yaw angle and from this geometrically to the horizontal VP position on the image.

IV. Experimental results

In this section we are going to present the experimental results of the detection and tracking methods based on real data sets. Moreover, we show the outcome of the PD controller in the 3D simulation environment.

A. Detection

We tested the edge detection methods mentioned in Section II-B.1. The results show that the Laplacian filter performs the best noise filtering and edge detection compared to the Sobel and the Canny methods, see Figure 4.



(a) Original image. (b) Canny edge detection.

(c) Sobel edge detection. (d) Laplacian filtering.

Fig. 4: Comparison of edge detection methods.

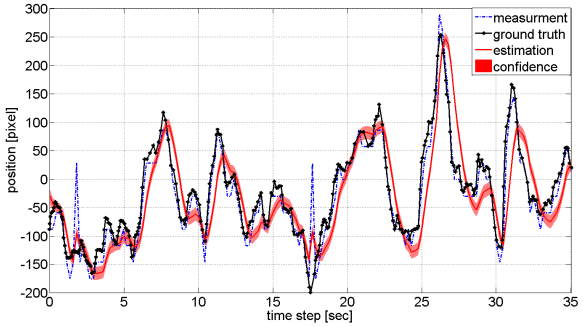


Fig. 5: Detection and KF estimation analysis on frequently used tracks. The dashed (blue) line is the measurement, the dotted (black) line is the ground truth labeled by hand frame by frame, the continuous (red) line is the estimation and the shaded (red) surface is the 95% confidence interval on the estimation.

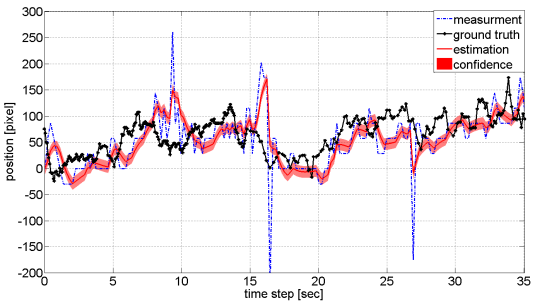


Fig. 6: Detection and KF estimation analysis on rarely used tracks.

B. Tracking on real images

The implemented tracking algorithm was tested on the field and in the simulation environment. The Kalman estimator filters the high peaks but it has a delay of one measurement, presented in Figures 5 and 6, showing real data processing. A weaker precision in observation and estimation is observed on the rarely used tracks, due to the characteristics of the unused steel, which is rustier. Figure 7 shows the detection results in simulation, where errors are of course smaller. Note also that different from the real-image

results, here the VP position is controlled to 0.

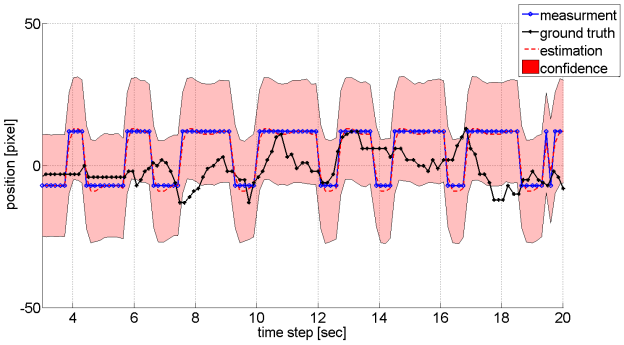


Fig. 7: Detection and KF estimation analysis in 3D simulated environment.

C. Control in simulation

Figure 8 presents a screenshot of the simulation environment we used to validate the controller. Our observation, regarding the Ziegler-Nichols tuning parameters, is that only the proportional controller keeps the system stable, while the PD and PID made the system unstable.

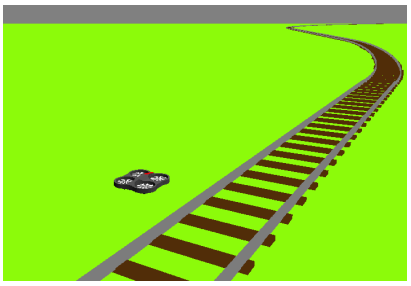


Fig. 8: AR.Drone in the simulation environment.

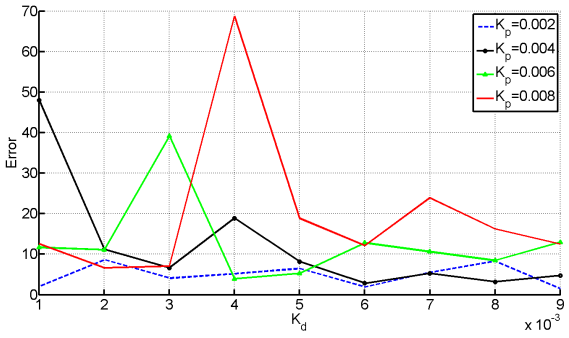


Fig. 9: PID tuning results based on the grid, on the Y axis is the error, on the X axis the values for K_d, and the different colored lines correspond to different values of K_p.

The result of the grid-based tuning method are presented on Figure 9. Based on this, we chose the K_p and the K_d as in Section III-C, and ran flight test in the simulation environment, for straight tracks and with turns. In both cases, the trajectory of the UAV remains between the railway tracks, see Figures 10 and 11, so our vision-based controller is successful.

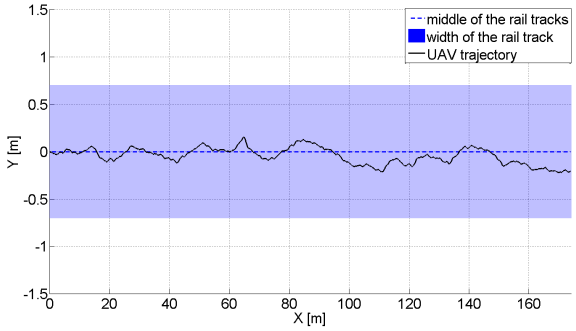


Fig. 10: Trajectory of the drone in the simulation following a straight rail track, where the black line is the trajectory, the blue line is the middle of the track and the blue shade is the width of the track.

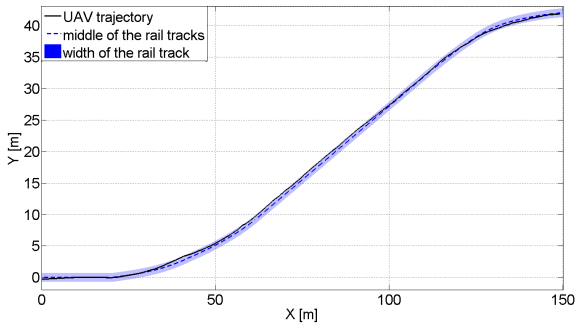


Fig. 11: Trajectory of the drone in the simulation, following a rail track with turns.

V. Conclusions

We developed a method to control a quadrotor UAV along a railway track, using images from its front camera. This can be used e.g. as a core component in a future UAV-based approach for railway monitoring or maintenance. Our method employs line detection to find the vanishing point (VP) of railway tracks and other lines in the image, and Kalman filtering to track this vanishing poing over subsequent frames. An optimized PD controller is then used to stabilize the VP to the center of the image, thereby ensuring the railway is followed. The overall methodology was successfully validated in a simulation environment, and we additionally validated the vision component on real frame sequences acquired with the UAV.

The validation of the controller in field tests is the first step of our future work. Additionally, a better model for the VP dynamics will be derived and combined with nonlinear filters, in order to improve the tracking performance and thereby the control.

References

[1] "Amazon testing drones for deliveries," BBC News, 2013. [Online]. Available: http://www.bbc.co.uk/news/technology-25180906

[2] "German railways to test anti-graffiti drones," BBC News, 2013. [Online]. Available: http://www.bbc.co.uk/news/world-europe-22678580

[3] M. Ali, "Real time detection of lane markers in urban streets," in *Proceedings of the 2008 IEEE Intelligent Vehicles Symposium*, Eindhoven, the Netherlands, June 2008, pp. 7–12.

[4] K. J. Astrom, "Pid controllers: theory, design and tuning," *Instrument Society of America*, 1995.

[5] C. Beard, Z.-Q. Chen, V. Kumar, Y. Lee, W. D. Leon-Salas, and P. Rao, "Saveus: Saving victims in earthquakes through unified systems," *IJCNDS*, vol. 10, no. 4, pp. 402–420, 2013.

[6] A. Benini, A. Mancini, and S. Longhi, "An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 4, pp. 461–476, 2013.

[7] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," in *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China*. IEEE, May 2011, pp. 5776–5783.

[8] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent Robotics Systems*, vol. 53, no. 3, pp. 263–296, 2008.

[9] T. Bresciani, "Modeling, identification and control of a quadrotor helicopter," Department of Automatic Control, Lund University, Sweden, Master's Thesis ISRN LUTFD2/TFRT--5823--SE, oct 2008.

[10] H. Durrant-Whyte, *Introduction to Estimation and the Kalman Filter*. ACFR, 2006.

[11] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*. IEEE, 2012, pp. 2815–2821.

[12] M.-D. Hua, T. Hamel, P. Morin, and C. Samson, "Introduction to feedback control of underactuated VTOL vehicles: A review of basic control design ideas and principles," *Control Systems, IEEE*, vol. 33, no. 1, pp. 61–75, 2013.

[13] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic hough transform," *Pattern recognition*, vol. 24, no. 4, pp. 303–316, 1991.

[14] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "Ar-drone as a platform for robotic research and education," in *Research and Education in Robotics-EUROBOT*. Springer, 2011, pp. 172–186.

[15] S.-P. Liou and R. C. Jain, "Road following using vanishing points," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 1, pp. 116–130, 1987.

[16] A. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza, "Mav urban localization from google street view data," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan*. IEEE, November 2013, pp. 3979–3986.

[17] G. Martin, "Ar.drone system identification," University of Camberra, Tech. Rep., 2012.

[18] 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 1st International Conference on Applied Robotics for the Power Industry (CARPI 2010)*, Montreal, Canada, 6–7 October 2010, pp. 1–7.

[19] Y. Rubinsztejn, "Automatic detection of objects of interest from rail track images," Master's thesis, School of Computer Science, University of Manchester, 2011.

[20] S. Saluja, A. K. Singh, and S. Agrawal, "A study of edge-detection methods," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, 2013.

[21] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation for autonomous rotorcraft mavs in complex environments," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*. IEEE, May 2013, pp. 1758–1764.

[22] P. Stephane, B. Nicolas, E. Pierre, and D. H. Frederic, "Ar.drone developer guide," May 2012.

[23] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, 2012.