Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-1 (2018) 480–485

Vision-based Control for Aerial Obstacle Avoidance in Forest Environments

Sumedh Mannar * **Mourya Thummalapeta** **
Saumya Kumaar Saksena *** **S N Omkar** ****

* *Project Assistant, Department of Aerospace Engineering, Indian Institute of Science (e-mail: sumedh.mannar@gmail.com)*
** *Student, Department of Aerospace Engineering, University of Petroleum and Energy Studies (e-mail: mouryathumplet@gmail.com)*
*** *Project Assistant, Department of Aerospace Engineering, Indian Institute of Science (e-mail: saumya@aero.iisc.ernet.in)*
**** *Chief Research Scientist , Department of Aerospace Engineering, Indian Institute of Science (e-mail: omkar@aero.iisc.ernet.in)*

Abstract: The increasing application of unmanned aerial vehicles (UAVs) in unstructured, natural environments raises the demand for robust obstacle avoidance systems that work in real-time. In view of this problem, we present a control algorithm for quadrotors based on monocular vision that is specifically designed for obstacle avoidance in forest environments. The algorithm presented is an enhancement of an existing algorithm, originally proposed and tested for usage with rovers, that uses a weighted combination of texture features to compute distances to nearest obstacle in various longitudinal strips of the image frame. The weights are pre-computed by means of supervised learning of correspondences of the features to ground-truth distances processed on frames derived from a simulated forest environment. The modifications proposed on the original algorithm show significant improvement in its obstacle-distance estimation accuracy and computational efficiency as observed from actual autonomous flying experiments carried out on an off-the-shelf quadrotor. The modified algorithm works at 15 frames/sec for a frame size of 160 x 120 pixels as profiled on an Odroid XU4 mini-computer. Results from both simulated images and real videos have been presented.

Keywords: Vision-based control, autonomous navigation, obstacle avoidance, unmanned aerial vehicles, quadrotor, supervised learning

1. INTRODUCTION

With the advent of unmanned aerial vehicles (UAVs) in the civilian domain for applications in agriculture [1], search and rescue [2], monitoring and security [3] etc., there is an increased demand for autonomy in such systems. Robotic systems designed to assist the human work force can be very productive if they exhibit a certain degree of autonomy. Most of the autonomous missions in case of UAVs rely on the Global Positioning System (GPS) for navigation. However, in applications like search and rescue missions, the target environments are unknown to the pilot and the UAV needs to maneuver close to the ground. In such cases, the UAV needs to follow the GPS coordinates of the target location and avoid any possible collisions also. Collision avoidance is one of the fundamental threads that runs in the human brain whether on foot or in a vehicle, which makes it an indispensable feature in robot autonomy.

Collision avoidance has been a prime sub-domain of research in robotics, very well implemented in context of robotic manipulators. Attempts to implement collision detection and avoidance in computer controlled manipulators date back to as early as 1977. Udapa [4] has discussed problem of planning safe trajectories for computer controlled manipulators with two movable links and multiple degrees of freedom. However little was known at that time about modifying trajectories dynamically based on any sensory data the system may acquire during execution, so not much could be achieved.

Nine years down the line, Khatib *et al.* [5] demonstrated real time collision avoidance with links and moving obstacles using PUMA 560 and a Machine Intelligent Vision Module. They were able to achieve a servo-control rate of 225 Hz in COSMOS which is an experimental manipulator programming system, using the operational space formulation approach.

Later in 1997, Fox *et al.* [6] demonstrated collision avoidance in mobile robots for indoor navigation at speeds up-to 95cm/s using motion equations for synchro-drive robots. Chakrabarthy *et al.* [7] came up with a novel collision cone approach for collision avoidance in 1998, ideally suited for indoor or mobile robots, but their was no real time implementation of the same. Till now, not much research was done in designing autonomous UAVs capable of collision detection and avoiding. However, in 2002 there

⋆ The work has been conducted in the UAV Laboratory, Department of Aerospace Engineering, Indian Institute of Science, Bangalore.

10.1016/j.ifacol.2018.05.081

was a big development in this field when Richards *et al.* [8] designed and published a MILP (mixed-integer-linear program) technique for path planning and collision avoidance for multi-aircraft systems. The approach was novel, however there was no real-time implementation of it yet. A lot of earlier research advances are mentioned by Vahidi *et al.* [9] in intelligent detection and adaptive cruise control.

In case of UAVs, most of the research has been focused on coordinated flight and collision avoidance in this regard is considered as one UAV crashing into another. Beard *et al.* [10] have discussed cooperation constraints like communication maintenance and collision avoidance. They attempted to implement a cooperative search problem using UAVs. Sigurd *et al.* [11] from MIT, implemented a total field sensing approach of magnetic nature for trajectory design and path planning for UAVs. In this scenario, each vehicle is associated with a magnetic field and the same field is the frame of reference for the particular vehicle. They calculated the gradient of the total field generated, hence the actual position is not needed to be known. A very interesting application of the popular proportional navigation (PN-law) for missile guidance was used to design an optimal collision avoidance system for UAVs by Han *et al.* [12], where they suggested to guide the velocity vector to the collision avoidance vector. Park *et al.* [13] discussed a PCA (Point of Closest Approach) technique to evaluate the worst conflict condition between two UAVs. The paper presented one resolution maneuvering logic called 'Vector Sharing Resolution'. Wang *et al.* [14] offered a MPC (model predictive control) algorithm for trajectory planning of UAVs. The complexity of their algorithm is on O(n), where 'n' is the number of obstacles. They achieved a speed of 25m/s in simulation testing.

The above mentioned techniques were suggested theoretically with very minimal practical support to their claims. Coming to the techniques that were actually implemented in real time, Green *et al.* [15] could achieve autonomous flight to an extent with an aircraft that weighed only 26 gms and at a speed of 2m/s. They however faced issues like optical flow characterization, wide angle view etc. Ross *et al.* [16] were successfully able to avoid 680 trees in a stretch of 3km, at a speed of 1.5m/s and 4m altitude. They carried out their experiments in a cluttered forest environment using imitation learning, which mimics human learning techniques. Fasano *et al.* [17] deployed a multisensor approach to collision avoidance, wherein they designed a system (1 radar + 4 cameras + 2 computers), incredibly suitable for defense applications but not that well suited for commercial uses. Along the same lines, Liao *et al.* [18] have demonstrated a LIDAR based collision avoidance approach for a tandem flock of small-scale unmanned aerial vehicles. The leader-UAV generates an occupancy grid map that allows navigation through obstacle rich environments and the follower-UAVs are made aware of the leader-UAV's position only by an on-board camera. The simulation results have proven the strategy to be effective, although there has not been a real-time implementation of the same.

The work presented in this paper is inspired from an algorithm proposed by Michels *et al.* [19]. Their approach estimates distances to nearest obstacle in various longitudinal stripes of the input image by using a weighted combination of texture features. We have closely studied their algorithm from a theoretical perspective and have identified various major and minor means to enhance the algorithm's performance. We have validated the modifications incorporated into the original algorithm in a simulated environment as well as in actual autonomous flight experiments.

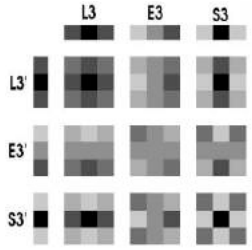


Fig. 1. Law's Masks for Texture Energy

2. ORIGINAL ALGORITHM

In this section, the original algorithm proposed by Michels *et al.* [19] is discussed in brief. In unison, the enhancements proposed on the various components of the algorithm are discussed from both design and practical implication points of view.

2.1 Feature Vector

The feature vector comprises of two types of texture features - (1) Texture Energies, and (2) Texture Gradients, which are spread over the spatial domain of the image frame. The input image frame is divided into 16 vertical stripes that correspond to 16 steering directions that the vehicle could take based on corresponding distance estimates by the algorithms. Additionally, each stripe is subdivided into 11 overlapping windows (with spatial overlap of 50 %). To form the feature vector for a stripe, the set of features described below are computed for each window in the stripe and in the two stripes directly adjacent to the stripe in question, thus generating a total of 33 sets of features per stripe.

Texture Energies : Laws Texture Energy computations are commonly used in the field of image segmentation. Their working principle is that regions of the image corresponding to objects differing in their surface textures are associated with distinctive texture energy signatures. In the context of forests, tree trunks shows remarkably lower texture energy measures when compared to that of the foliage. For this reason, texture energy measures serve as effective cues in quantifying the area covered by tree trunks in each window. Computation of texture energies is carried out by convolution of image regions with Laws Masks for Texture Energy described in Fig. 1.

Convolution is carried out in YCbCr color space. Convolution masks M_1 to M_9 are applied to Y channel (intensity channel) and M_1 is applied to Cb and Cr matrices (color channels). A total of 11 texture energy (F_n) measures are generated per window as described in Equations (1-3).

$$F_n = I_Y * M_n, \qquad n = 1, 2, \ldots, 25 \tag{1}$$

$$F_{10} = I_{Cb} * M_1 \quad (2)$$
$$F_{11} = I_{Cr} * M_1 \quad (3)$$

Texture Gradients : Texture Gradients, commonly referred to as 'edges', serve as an effective cue in perception of depth in humans. As such, Michels *et al.* [19] have proposed using them as effective features in this context. As discussed in their paper, density of edges in various orientations could be calculated using two popular techniques - (1) Radon Transform, (2) modified version of Harris Corner Detector. From our observation, using Harris corner detector for the same delivers better obstacle-distance estimation accuracy when compared to that obtained using Radon. Also, as indicated in the reference paper and also confirmed by experimental verification from our side, using Radon and Harris together does not show any significant improvement in the accuracy. Thus we used the Harris Corner Detector in its modified form for texture gradient feature computations in our experimentation.

The steps involved in computing texture gradient features for each processing window are discussed below. Firstly, prepare 15 equally spaced bins corresponding to angles spread over 0° to 360°. Then, a sliding Harris window of size 5×5 is traversed across the processing window with the following set of steps performed on each Harris window -

- Generate 2×2 matrix of intensity gradient covariances
- Compute eigenvalues and eigenvectors of the gradient covariance matrix
- Calculate angle represented by both eigenvectors and identify to which bin they belong to
- Store the eigenvalues corresponding to each of the eigenvectors in their respective bins

After execution of the above steps, for each window, the sum of the eigenvalues in each of the 15 bins are calculated and stored as the texture gradient feature values for that window.

Thus the feature vector for each window comprises of 11 Laws and 15 Harris features. For a total of 11 processing windows in a stripe and for the 22 windows in its directly adjacent stripes, this gives us a feature vector of length 858.

2.2 Estimation Model

The distance to nearest obstacle in each stripe is modelled as a linear function of the feature vector computed for the stripe. Let D be the distance to nearest obstacle in a particular stripe. Let the features discussed above be represented by F_n, where n = 1,2,3...858. Let coefficients of the linear equation be represented by c_n, where n = 1,2,3...859. The estimation model defined for this system is described in Equation (4).

$$\ln(D) = c_1F_1 + c_2F_2 + \cdots + c_{858}F_{858} + c_{859} \quad (4)$$

2.3 Training

Given the estimation model, the problem boils down to arriving at estimates for c_n (where n = 1,2,3...859). Thus we have to solve a prepared set of simultaneous linear equations for 859 unknowns using linear regression based on least square fit. In ideal conditions, where every equation is a perfectly accurate representation of true measures of each of the constituent parameters, the number of equations required to solve this problem would be 859. Taking into consideration the imperfections in training data occurring due to human error and the variance in the possibilities of tree structures occurring in training images, the number of equations required would be at least 10 times that of the ideal scenario in order to ensure convergence of the estimates. This implies that if the algorithm is used as it was proposed by Michels *et al.* [19], ground truth values of distances to nearest obstacle would have to prepared and marked for a total of 8590 stripes.

3. PROPOSED ENHANCEMENTS

The specifics of our proposed modifications are discussed in this section.

3.1 Major Enhancement: Improved Choice of Feature Vector -

It is important to note that the algorithm proposed by Michels *et al.* [19] was originally presented and tested for use with ground vehicles. As the first step towards implementing the same for obstacle avoidance in UAVs, we have investigated its suitability and applicability in this context. Through our theoretical analysis, we figured that the core principle driving the algorithm is the spatial spread of signatures of tree trunk textures in a stripe in question.

Before we understand the technical role played by spatial spread in identifying distance to trees, it is important to have a look at a couple of principles discussed before - texture energies of computed from processing windows that hold tree trunks show remarkable distinction from the ones that hold foliage, which lie more towards brighter regions of the color space. In a similar fashion, edges (texture gradients) in the case of tree trunks shows high densities in the in the angular proximity of the vertical direction (close to 0° and to 180°) by virtue of the nearly upright posture of typical tree trunks. So, based on these principles, for each stripe being processed, the algorithm virtually knows all windows in the stripe where tree trunks are present and to what extent (represented by the magnitudes of the various features). Let us consider two cases - (1) a tree situated a few feet in front of the camera, and (2) a second tree situated a few metres in front of the camera. Let us consider, for example, that the nearer tree falls within stripe-4 of the image and the farther tree falls within stripe-12 of the image. The farther the tree, the more the distance of the base of the tree's trunk from the bottom of the image. As a result, the lower windows of the stripe-4 show feature signatures corresponding to tree trunks which would not be the case with stripe-12. Thus, from this theory, we understand that lower windows have a dominant role to play in determining the distance to nearest obstacle in a stripe. Putting it in simpler theoretical terms, the distance of the tree is estimated by this algorithm by identifying from what height (represented by window position) in the stripe the tree trunk begins to rise up from.

With this in mind, let us consider the role played by windows in the upper half of the image in distance estimation. As per the theory discussed above, they do not contribute to estimating the distance of the tree from the camera. In fact, they have a negative impact on the estimation model, as explained here - Consider a training image that has trees of varying heights but situated at the exact same distance from the camera. Consider, for example, that stripe-5 has a tree that extends up to about 50% of the height of the image (covers windows 5 to 10). Consider that stripe-6 has a tree that extends up to about 70% of the height of the image (cover windows 3 to 10). The feature signatures from stripe-5 and stripe-6 differ in significantly in windows 3 and 4 because of the present of trunk textures in those regions in stripe-6 but not in stripe-5. So, in this scenario, we observe that though both the trees are situated at the same distance from the camera, there is a remarkable difference in their features vectors. It is clear that such scenarios will have a negative impact on the training of the estimation model for the simple reason that the output is the same for significantly different inputs. Also they do not have any recognized positive impact on improving the training or estimation accuracy.

Based on our above analysis, we have conducted training experiments using feature vectors of reduced lengths by neglecting feature information from windows in the upper half of the image and arrived at an optimal solution for the same. We have concluded that neglecting feature extraction from windows 1 to 5 helps generate better estimation accuracy without any induces losses. Thus we propose the use of windows 6 to 11 in feature vector computations. As a result, the length of the feature vector drops from 859 to 469 with an added improvement in accuracy and zero negative impact. Additionally, this reduces the computational cost by 45% of that of the original algorithm by drastically reducing the number of features to be computed.

3.2 Major Enhancement: Fusion-processing at Dual Image Resolutions -

The original algorithm was tested with input frames of resolution 352×288. In the process of testing the performance of the modified algorithm for various resolutions, we encountered some interesting observations.

At the resolution of 320×240 (we maintained a standard 4:3 aspect ratio in our trials), with a Harris window-displacement of 2 pixels, the trade-off between accuracy and computational efficiency was well balanced when compared to any other resolutions, as illustrated in the results section. On observing results from 160×120 and 80×60 resolutions with Harris windows displacement set to 2 pixels and 1 pixel respectively, we noted that the algorithm had its drawbacks in a few scenarios at both resolutions but in most cases, one of them did better where the other failed to perform as expected. An illustrative example for the same is provided in Fig. 2, where in two shown cases it can be seen that the output for a few stripes at 80×60 resolution is better than that at 160×120. In majority cases, 160×120 provides a better performance when compared to 80×60, but the examples such as the ones represented in Fig. 2 have been identified in significant numbers.

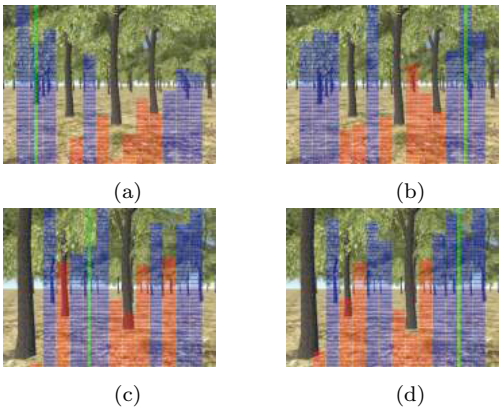


Fig. 2. Fusion Processing at Dual Image Resolutions - (a), (c) are outputs of the algorithm on two different input images at 160×120 and (b), (d) are outputs of the algorithm on the same pair of images at 80×60

As a result, we learned that there could be a better performance optima for the algorithm if processing at multiple resolutions is considered. So we generated outputs of fusion between 160×120 and 80×60 resolutions by means of weighted combination of distance estimates - we used different pairs of weights ranging from (0.1, 0.9) to (0.9, 0.1). We arrived at an optima at (0.6, 0.4). Thus, we recommend running this algorithm at multiple resolutions in order to achieve a high performance to computational cost ratio.

3.3 Minor Enhancement: Control Logic -

The algorithm is designed to send only high-level commands to the drone - (1) forward translation speed, and (2) (heading / yaw) rotational velocity. Forward Velocity is maintained constant throughout the period of functioning of the algorithm. This algorithm allows the user to choose a forward velocity magnitude based on factors such as density of trees in the test area, constraints on velocity because of mission requirements (frame rate of surveillance algorithms, for example) and so on.

Before we get into the details of yaw velocity control, let us review the number of possible output directions from the algorithm. This number comes to two less than the number of stripes, which comes to 14. The last 2 stripes are excluded because each of them have only one adjacent neighbor, and as discussed before, the feature vector of each processed stripe should essentially hold the feature values of its two adjacent stripes according to the design of the system. Thus the algorithm's outputs, 14 in number, belong to a discrete set of angles corresponding to valid stripes and valid stripe junctions. Fig. 3 indicates how the angle corresponding to each of these directions was calculated based on the field-of-view of camera, focal length of camera, and width of the image in pixels.

In Fig. 3, A_1 - A_8 represent the angular positions of stripe borders. Lets say, for example, the best stripe as decided by the algorithm is stripe-5 (starting from the top). The angular position of the stripe that would be considered for steering commands is calculated by taking an arithmetic mean of the angles corresponding to its two borders, i.e.

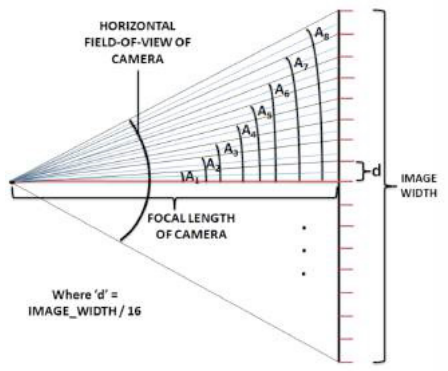


Fig. 3. Correspondence of Stripes to Steering Angles

in the case of stripe-5, it would be calculated as the mean of A_4 and A_3.

Direction of yaw rotation velocity is determined based on the longitudinal half in which the direction output of the algorithm lies. Coming to the magnitude of yaw velocity, the only value that is set is an upper limit, again according to the user's requirements. This maximum value is assigned to the pair of farthermost stripes from the longitudinal center of the image, i.e. when the target direction decided by the algorithm is either of the farthermost stripes, the magnitude of the yaw velocity command sent to the drone equals the upper limit that was set by the user. Linear proportionality is applied for all other cases as described by Equation 5. In Equation 5, θ_s stands for angular position of an arbitrary stripe decided by the algorithm, θ_{max} stands for the angular position of the farthermost stripe, ω_{max} stands for the set upper-limit value for yaw velocity, and ω stands for the angular velocity computed for the decided stripe.

$$\omega_s = \left(\frac{\theta_s}{\theta_{max}}\right) \times \omega_{max} \tag{5}$$

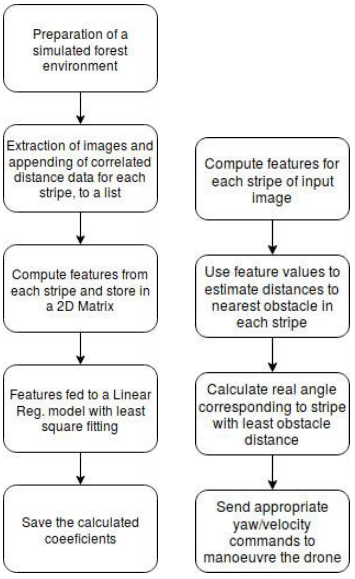
4. EXPERIMENTATION

4.1 Hardware and Software Platforms used -

The hardware platform used in our experimentation was the Parrot AR Drone 2.0. It is an off-the-shelf quadcopter compatible with ROS. Low level controls of the drone are handled by a driver for ROS named as ardrone¬autonomy. This driver package is based on official AR-Drone SDK version 2.0.1.

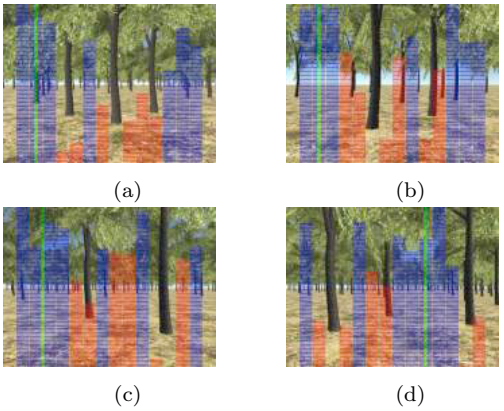
4.2 Algorithm Control Parameters -

- Image Resolution(s): 160×120 pixels and 80×60 pixels
- Number of Valid Stripes: 14
- Number of Windows: 6 (where top 5 windows from original 11 overlapping windows are neglected)
- Vertical Overlap between Windows: 50 %
- Harris Window Size: 5×5 pixels
- Harris Window Displacement: 2 pixels for 160×120 resolution and 1 pixel for 80×60 resolution (along both horizontal and vertical directions)



(a) Training Process Flow (b) Testing Process Flow

Fig. 4. Process Flow Diagrams



(a) (b) (c) (d)

Fig. 5. A few Results from Graphics-basic / Graphics-1 Dataset

5. RESULTS

The algorithm has been tested on two datasets generated from simulations of forest environment as well as on a Parrot AR Drone 2.0. A few results from each have been illustrated in Fig. [5-7]. A video captured from one of our real-time experiments on the Parrot could be viewed ***here***.

Test Data	E_{depth}	Rel. Depth	E_θ	Hazard Rate
Graphics-1	0.34	0.38	0.47	N/A
Graphics-2	0.48	0.57	0.78	N/A
Real-time	0.53	0.44	0.51	1.85

Table I. Test Error on - Graphics-basic, Graphics-advanced, and Real-time datasets using common training on separate Graphics-advanced dataset

Error metrics proposed for the original algorithm by Michels *et al.* [19] have been calculated in each case and have been tabulated in the Table I above.

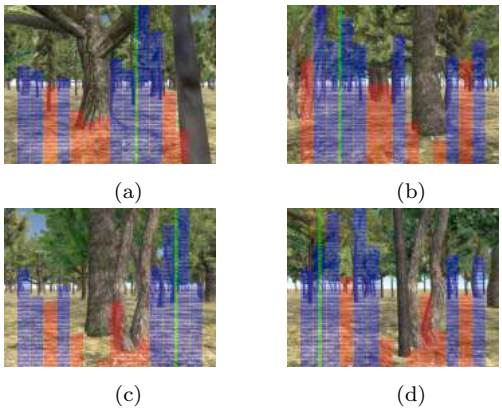


Fig. 6. A few Results from Graphics-advanced / Graphics-2 Dataset

Fig. 7. Real Time results from AR Drone

6. CONCLUSIONS & FUTURE WORK

Results indicate significant improvement in the performance of the algorithm by virtue of enhancements incorporated into it. Though the algorithm was introduced back in 2005 and uses a technique as basic as linear regression for training, from our understanding, the real driving factor here is the robustness of the features used. Thus we intend to explore the potential of these features in classification of obstacles in forest environments as a step towards development of a more advanced system.

REFERENCES

[1] Bastiaan A. Vroegindeweij, Sjoerd W. van Wijk, and Eldert J. van Henten, Autonomous Unmanned Aerial Vehicles for Agricultural Applications, presented at the 2014 Int. Conf. Agricultural Engineering, Zurich, Switzerland.

[2] Sonia Waharte and Niki Trigoni, Supporting Search and Rescue Operations with UAVs, presented at the 2010 Int. Conf. Emerging Security Technologies, Canterbury, UK.

[3] Konstantinos Kanistras, Goncalo Martins, Matthew J. Rutherford, and Kimon P. Valavanis, A Survey of Unmanned Aerial

[4] Udupa, Shriram Mahabal. "Collision detection and avoidance in computer controlled manipulators." PhD diss., California Institute of Technology, 1977

[5] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." The international journal of robotics research 5, no. 1 (1986): 90-98

[6] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance." IEEE Robotics & Automation Magazine 4, no. 1 (1997): 23-33

[7] Chakravarthy, Animesh, and Debasish Ghose. "Obstacle avoidance in a dynamic environment: A collision cone approach." IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 28, no. 5 (1998): 562-574

[8] Richards, Arthur, and Jonathan P. How. "Aircraft trajectory planning with collision avoidance using mixed integer linear programming." In American Control Conference, 2002. Proceedings of the 2002, vol. 3, pp. 1936-1941. IEEE, 2002

[9] Vahidi, Ardalan, and Azim Eskandarian. "Research advances in intelligent collision avoidance and adaptive cruise control." IEEE transactions on intelligent transportation systems 4, no. 3 (2003): 143-153

[10] Beard, Randal W., and Timothy W. McLain. "Multiple UAV cooperative search under collision avoidance and limited range communication constraints." In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on, vol. 1, pp. 25-30. IEEE, 2003

[11] Sigurd, Karin, and Jonathan How. "UAV trajectory design using total field collision avoidance." American Institute of Aeronautics and Astronautics (2003)

[12] Han, Su-Cheol, and Hyochoong Bang. "Proportional navigation-based optimal collision avoidance for UAVs." In 2nd International Conference on Autonomous Robots and Agents, pp. 13-15. 2004

[13] Park, Jung-Woo, Hyon-Dong Oh, and Min-Jea Tahk. "UAV collision avoidance based on geometric approach." In SICE Annual Conference, 2008, pp. 2122-2126. IEEE, 2008

[14] Wang, Xiaohua, Vivek Yadav, and S. N. Balakrishnan. "Cooperative UAV formation flying with obstacle/collision avoidance." IEEE Transactions on control systems technology 15, no. 4 (2007): 672-679

[15] Green, William E., Paul Y. Oh, and Geoffrey Barrows. "Flying insect-inspired vision for autonomous aerial robot maneuvers in near-earth environments." In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 3, pp. 2347-2352. IEEE, 2004

[16] Ross, Stphane, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. "Learning monocular reactive uav control in cluttered natural environments." In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pp. 1765-1772. IEEE, 2013

[17] Fasano, Giancarmine, Domenico Accardo, Antonio Moccia, Ciro Carbone, Umberto Ciniglio, Federico Corraro, and Salvatore Luongo. "Multisensor based fully autonomous non-cooperative collision avoidance system for UAVs." AIAA Infotech@ Aerospace (2007): 7-10

[18] Liao, Fang, Jianliang Wang, Rodney Teo, Yuchao Hu, Shupeng Lai, Jinqiang Cui, and Feng Lin. "Vision-based autonomous flocking of UAVs in unknown forest environment." In Control and Automation (ICCA), 2016 12th IEEE International Conference on, pp. 892-897. IEEE, 2016.

[19] Michels, Jeff, Ashutosh Saxena, and Andrew Y. Ng. "High speed obstacle avoidance using monocular vision and reinforcement learning." In Proceedings of the 22nd international conference on Machine learning, pp. 593-600. ACM, 2005.