2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Bioinspired object motion filters as the basis of obstacle negotiation in micro aerial systems

Rui Zhou and Huai-Ti Lin

***Abstract*— All animals and robots that move in the world must navigate to a goal while clearing obstacles. Using vision to accomplish such task has several advantages in cost and payload, which explains the prevalence of biological visual guidance. However, the computational overhead has been an obvious concern when increasing number of pixels and frames that need to be analyzed in real-time for a machine vision system. The use of motion vision and optic flow has been a popular bio-inspired solution for this problem. However, many early-stage motion detection approaches rely on special hardware (e.g. event-cameras) or extensive computation (e.g. dense optic flow map). Here we demonstrate a method to combine an insect vision inspired object motion filter model with simple visual guidance rules to fly through a cluttered environment. We have implemented a complete feedback control loop in a micro racing drone and achieved proximal-distal object separation through only two object motion filters. We discuss the key constraints and the scalability of this approach for future development.**

Keywords*— *motion vision, obstacle avoidance, visual guidance

I. Introduction

Obstacle avoidance has been an important topic in mobile robots as one of the fundamental challenges in real-world operation [1]. The task can be broken down into two components: obstacle detection and avoidance manoeuvring. For obstacle detection, the top-down approach consists of object identification and classification [2]. The agent would then react based on an interpretation of the identified objects in the world. The bottom-up approach requires mapping. The simultaneous localization and mapping (SLAM) processes solves both obstacle detection and avoidance manoeuvring problems at the same time [3]. On the other hand, recent deep learning approaches neither identify the objects nor map the environment [4][5]. They translate the sensory input directly into motor commands given sufficient model training.

While all these above approaches can achieve obstacle avoidance under certain conditions, most of them require high computational power and/or expensive instrument. These requirements impose a challenge for small aerial systems with limited computational power and payload capacity. Is there an alternative that delivers adequate performance with little computational/instrumentational cost? Specifically, can we implement a basic guidance strategy via one low-cost camera which most small aerial systems have onboard already? While the majority of research in the field has focused on developing methods based on more complex systems, such as stereo camera and camera with sensors, there are a few solutions based on monocular vision. For example, there is the appearance-based method on ground vehicle using floor color

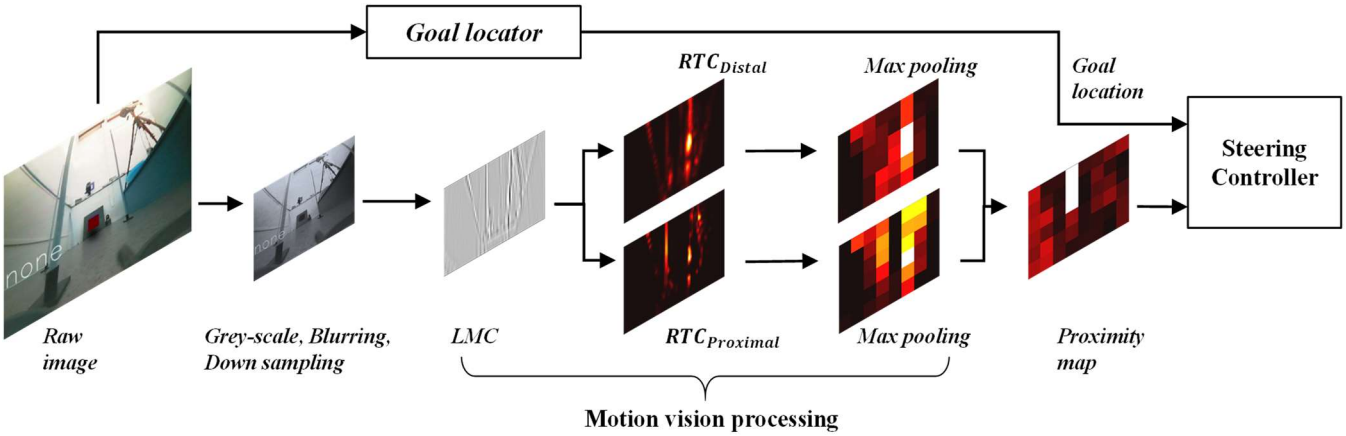

Fig. 1: Flowchart of the proposed **real-time vision framework**. After resampling the raw image, the motion vision processing stage consists of a model of lamina monopolar cell (LMC) and a model of rectifying transient cell (RTC) in the insect visual system. LMC model performs spatio-temporal filtering to emulate the early visual processing of insect eyes. Two model RTCs extract foreground and background motion via different spatial kernels and temporal dynamics. Collectively this bio-inspired motion vision implements object motion filters for obstacle detection. Finally, we pool the signal and take the ratio of the two model RTCs to construct a proximity map. An independent color-based goal locator produces a general directional signal which is combined with the proximity map to inform the steering controller. To provide a reference, we substituted the motion vision processing stage with a classic optic flow algorithm and generated another proximity map.

The authors are based at Neuromechanics and Bioinspired Technologies (NBits) Laboratory, Imperial College London, United Kingdom.
h.lin@imperial.ac.uk

as reference [6]; optic flow method for obstacle detection [7][8]; bioinspired method using LGMD model [9]; machine learning approach using low cost CNN [10]; SIFT-based method using the size-ratio factor between frames [11] and mono-SLAM approaches such as ORB-SLAM [12].

Insect vision has provided numerous inspirations in efficient visual guidance [13][14]. It operates with milliwatts of power to achieve complex visual tasks compared to kilowatts for many modern machine vision systems. The success can be attributed to three principles: **1) minimal & efficient sampling; 2) operating on changes; 3) using visual motion heuristics**. To exploit such an architecture, we focus on the second principle (motion detection) in conjunction with the third principle (visual motion heuristics). For **motion vision processing**, we adapted previous work [15] modelling the lamina monopolar cells (LMCs) and the rectifying transient cells (RTCs) in the insect medulla (**Fig 1**). The LMC model contains spatio-temporal filters to mimic the contrast enhancement of insect eyes. The RTC model implements motion detection with parameters suitable for constructing motion filters of objects moving in the real-world (i.e. visual size, speed, and relative motion). This study explores how such an approach might simplify the visual computation for a micro unmanned aerial system (μUAS) in the context of obstacle negotiation.

II. Motion Vision Processing

A. Initial image pre-processing

In line with the cost argument, we used a standard analog video stream (640 × 480, 29fps). We down-sampled to an image size for our application (80 × 60), and converted the color image into grayscale (**Fig 1**). Then we implemented the LMC model by applying a temporal band-pass filter to denoise pixel changes and spatial high-pass to enhance edges within the image. This process removed unnecessary information in the image and further reduced the data volume. In insect eyes, much of this is done by the optics of the compound eyes and the processing of the early visual layers [16][17].

B. Construction of an object motion filter

The Reichart-correlator type Elementary Motion Detector (EMD) is perhaps the most popular motion vision model [18]. In this model, on- or off- responses from neighbouring pixels are convolved after some time delay, forming a strong downstream integrated response. The RTC model is a variant of EMD which correlates the on-responses to the off-responses at the same pixels via a temporal filter [19]. If a pixel experiences an on-response followed by an off-response (or vice-versa) separated by a specific time, we assume motion at that pixel location. This approach is generic and yields robust responses to the leading and trailing edges of a moving object.

To turn a model RTC into an object motion filter, we employed lateral inhibition to filter the size of the expected moving objects. These spatial kernels can range in sizes as well as shapes. In general, we had the best response when the moving object matches the size and shape of the kernel (**Fig 2A-B**). This formed the first part of the object motion filter. The second part consisted of an infinite impulse response filter to perform temporal low-pass filtering, which effectively

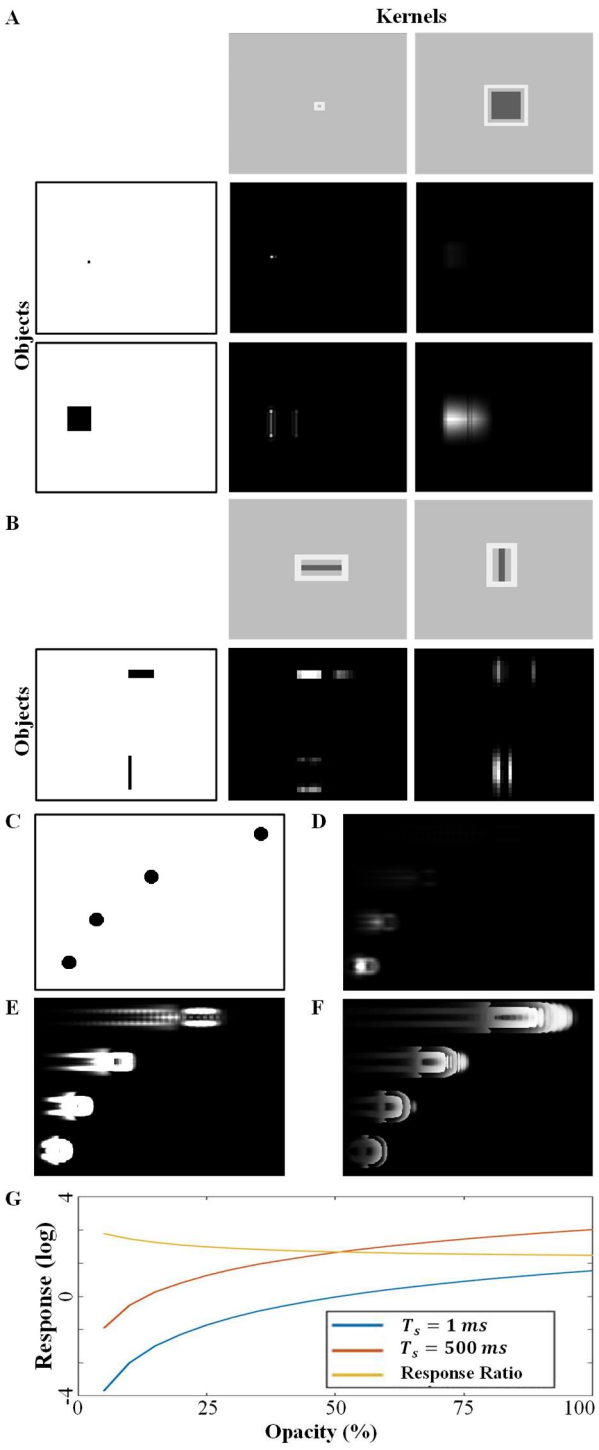

Fig. 2: The effect of spatial kernels and sampling time *Ts* to the response of a model RTC. (A) The responses of a model RTC to different object sizes using different sizes of spatial kernel as illustrated in grayscale. (B) the responses to different shapes of object using horizontal and vertical spatial kernels. (C) Four identical objects (from top to bottom) moving at the velocity of 8°/s, 4°/s, 2°/s and 1°/s respectively. A model RTC using large (D) and small (E) value of *Ts* respectively. (F) The ratio between the responses generated from these 2 model RTCs. (G) The influence of object contrast (simulated via opacity) on the responses of model RTCs. While the individual RTCs respond stronger to higher contrast, the response ratio maintains stable contrast invariance.

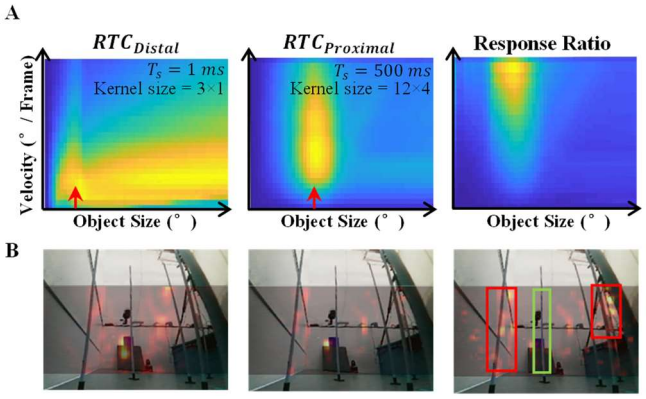

Fig. 3: Size-speed tuning and motion detection. (A) The response to a moving object with different size and speed using distal (left) and proximal (middle) model RTCs. The red arrows indicate sizes of spatial kernel used. The ratio between these 2 RTCs (right) has a specific size-speed tuning that we can engineer. (B) The transparent masks illustrate the responses to objects in visual field by RTC_{Distal} (left) and $RTC_{Proximal}$ (middle) during the drone flight through the obstacle forest. The ratio between these 2 channels provides the velocity profile of the current scene (right). The red boxes mark areas with high response (i.e. near obstacles); the green box marks the object in far distance, which does not induce noticeable response due to low relative velocity.

introduced the time constant needed for the temporal correlator [16]. In our implementation, this correlation was achieved by altering the sampling time T_s in the filter coefficient, which altered the time constant in the transfer function. As shown in **Fig 2C-E**, the response of a model RTC towards fast moving objects increased with increasing T_s, thus altering the speed sensitivity. For each model RTC, we could assign a specific spatial kernel and T_s to match moving objects of interest.

III. Size-speed tuning and object detection

A. Size-speed tuning

By choosing a specific combination of spatial kernel and T_s in a model RTC, we can design its size-speed tuning. **Fig 3A** shows the response of a model RTC with selected pairs of spatial kernel and T_s, induced by moving black objects with different size (in pixel) and speed (in degree per frame) on a white background. At small T_s, the RTC shows stronger response towards both large object and object matched to the kernel size, but the response decays significantly with the increase of speed. On the other hand, the peak response in RTC using a large T_s covers a much larger range of speed but shows a high sensitivity towards the size of moving object.

B. Proximal-distal object separation

It is well-known from motion parallax that nearby objects produce more translational visual motion than distant objects. Using this principle, one can produce a "nearness map" purely from visual motion such as optic flow [20][21]. Here, we take advantage of the size-speed tuning of different model RTCs to produce a "proximity map". In essence, we expect the distant objects to have small features with slow visual motion while the near-by objects to exhibit larger features with higher visual motion (**Fig 3B**). To implement this, the LMC output was fed into two independent model RTCs, referred to as RTC_{Distal} and $RTC_{Proximal}$. The first one had a T_s of 1ms with a kernel size of 3 × 1 and the latter one had a T_s of 500ms with a kernel size of 12 × 4. This effectively screened for small slow objects and large fast objects respectively. The proximity map is the ratio between these two RTC outputs:

$$Proximity = \frac{RTC_{Proximal}}{RTC_{Distal} + e}$$

where e is a very small number to avoid division by zero. The choice of kernels and T_s were based on the image statistics from a small 29fps camera moving at ~1.5 m/s. This proximity map can be tuned for different image resolution and is robust against contrast variations. We applied a constant threshold to the proximity map to extract relevant nearby objects, effectively implementing **obstacle (proximal object) detection**.

This comparative approach also solves an issue most EMDs suffer in application of motion detection: inhomogeneous contrast across the visual scene which affects the response in a quadratic way [22]. This issue was explored using simulated moving object with different opacity levels, and the result is shown in **Fig 2G**. The RTC response increases with increasing object contrast. However, RTCs with different speed tuning (T_s) share a similar trend. Consequently, when we took the ratio between 2 separate RTCs, the response maintained relatively constant across contrast levels (**Fig 2G**). This provided a good contrast compensation and reduced the tracking error in the output, shown in **Fig 2F**.

IV. Micro-UAS Implementation

A. Hardware setup

The proof-of-concept experiments were carried out using a commercial micro racing drone (Mobula 7; 44g flying weight) (**Fig 4C**), with a built-in wide angle (120°) camera and video transmitter. A video receiver and analog-to-digital converter provides low-latency (27ms) live feed for the computer. Then our bio-inspired visual guidance model derived steering commands and controlled the drone via the trainer port of the radio controller. The entire sensorimotor feedback loop clocks ~47ms total latency. We implemented the bio-inspired visual guidance system in Python on a laptop for convenience (**Fig 4B**). The actual computation is simple and can be easily accomplished onboard at a later stage.

B. Flight-control heuristics

For proof-of-concept, the steering control implementation modulated only the roll angle of the drone as lateral steering. The steering command was further discretized into three fixed level roll signals: left, none and right. A constant forward pitch signal was applied outside the control loop to keep the drone moving forward at a steady speed of ~1.5m/s. The throttle was set to a constant level to allow the drone to maintain neutral buoyancy. The use of steady command signal in all controlled channels ensured simple flight gestures while maintaining roughly constant speed.

This study focused on obstacle negotiation, thus we provided a fixed goal for the drone's general navigation with a rectangular red goal. We applied an HSV threshold on the

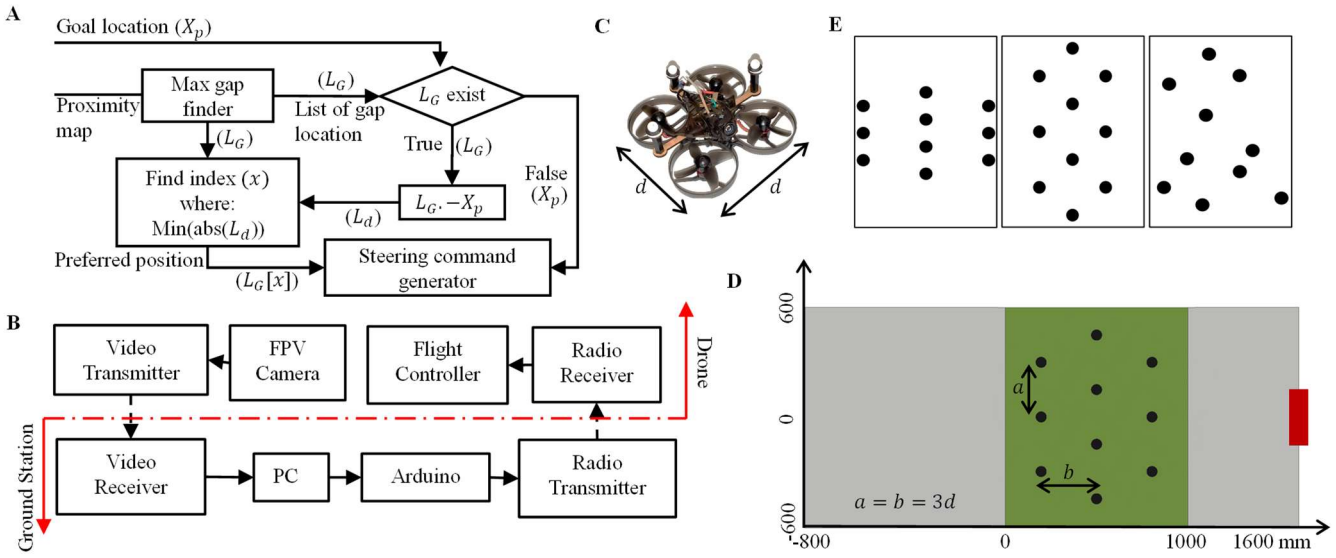

Fig. 4: Steering controller, Micro UAS hardware, and flight experiment implementation. (A) Control system flowchart after motion vision processing. (B)(C) Micro UAS hardware implementation with a complete visual close-loop. The drone with dimension $d \times d$ carries three retroreflective markers for kinematics analyses. (D) The flight arena with reference obstacle layout. The obstacle forest zone is marked in green and the goal location is represented by a red block. In the reference layout, the latitudinal spacing a and longitudinal spacing b between pylons are $3 \times d$. (E) Example obstacle forest layouts with lattice (left: $a = 2, b = 4$; middle: $a = 4,\ b = 2$) and a pseudo-random configuration (right).

colour band-pass filter to recognize the goal. The basic goal-directed behaviour was implemented as a binary control toward the red goal via lateral steering as described. Example trajectories in **Fig 5A** show how the drone converged to the goal despite different starting locations.

Given the basic goal-directed behaviour, the drone must weave through an obstacle field using the bio-inspired object motion filter algorithm proposed. To avoid conflicting avoidance commands from multiple obstacles, we took the gap aiming approach after previous work [23]. Based on the proximity map described above, the drone steered toward the maximum gap between obstacles in the visual field. Specifically, we summed up the proximity map along the horizontal axis into a 1-D array and found potential obstacle location via a proximity threshold (**Fig 4A**). This process transformed the proximity map into a binary array where ones represented obstacles and zeros represented gaps. Then we found the position of the centre of the largest gap, or gaps L_G if multiple gaps were the same size. Both the goal position and gap positions were passed into an evaluation process to decide the appropriate command. If the gap list was empty, the steering command would be based on the goal location X_p only. Otherwise, an element-wise subtraction operation occurred between L_G and X_p, generating a list of difference values. The preferred direction would be selected as the gap in the list closest to the goal direction. Based on the selected gap location, the drone made fixed lateral steering either to the left or to the right in a discrete control scheme. We did not attempt to construct any sophisticated flight controller as this was sufficient for our proof-of-concept evaluation of the motion vision system.

To provide a benchmark, we substituted the entire motion vision processing with a standard dense optic flow algorithm [22] to generate the proximity map. This allowed the comparison of different motion vision methods while maintaining the same steering heuristics.

C. *Obstacle course*

The experiment was carried out in an enclosed rectangular flight area (2400 × 1200mm), forming a linear corridor (**Fig 4D**). Ten pylons were placed in a 1200 × 1000mm area in the middle of the arena as the obstacle forest. We had 800mm and 600mm of clear space before and after the obstacle forest respectively as buffer areas. A motion capture system (Qualisys M5, Qualisys AB, Sweden) was set up above the arena to record the drone trajectories at 100fps for analyses.

D. *Drone flight test*

The drone was first tested to fly in the empty arena with the red goal only to validate the effect of visual navigation, and to collect baseline trajectories (**Fig 5A**). Two groups of obstacle forest layout were designed to test our obstacle negotiation system. To test the effect of obstacle density, we designed a lattice layout (**Fig 4E**) which allowed us to vary the obstacle density in either the lateral or longitudinal axes of the arena. We used the drone body-size as the unit to vary the pylon density between 2 and 5 (**Fig 4D**). The lattice with 3 drone size spacing was our reference layout. To test how the system copes with un-structured obstacle arrangements, we derived pseudo-random pylon translations from the reference layout to make sure the forest density was close to the reference layout (example in **Fig 4E**). We subjected the drone with one random forest 20 times to test the repeatability (**Fig 5D-F**) and then 20 different random forests once each to test the robustness (trajectories not shown). We only captured valid flight trials as defined by the drone passing through the obstacle forest instead of around it.

V. Obstacle Negociation Analysis

A. *The effect of obstacle density*

To determine the performance of our object motion filter as an obstacle detection mechanism, all 3D flight trajectories were collapsed to 2D top-down view for the purpose of this analysis. The baseline trajectories demonstrated the underlying navigation toward a distant goal (**Fig 5A**). **Fig 5B** shows the success rate of clearing the lattice forest of different obstacle densities with and without the proposed object motion filter based obstacle detection activated. The result clearly shows a notable improvement of the success rate with the obstacle detection activated. As the obstacle spacing increased homogenously from 2 to 5, the success rate went from 0.1 to 0.85 with the obstacle detection activated versus 0.2 to 0.5 without. For the reference layout ($a = b = 3d$), the obstacle detection doubled the success rate from 0.3 to 0.6.

B. *Random forest trajectory analysis*

We further tested our system using pseudo-random obstacle layouts. To provide the reference performance, we overlaid the baseline trajectories in **Fig 5A** on top of a random obstacle layouts and quantified the obstacle clearing success rate. Then we sent the drone through the same random obstacle forest without obstacle detection given the same starting point. This estimated how well the drone can miss the obstacles by chance under only the goal-directed guidance with varying starting points (**Fig 5C**), and with the same starting point (**Fig 5D**). The success rates were 5/20 and 6/20 respectively. We then flew the drone through the obstacle forest 20 times with the object motion filter method (**Fig 5E**) and with the optical flow method (**Fig 5F**). The success rates were both 10/20. Finally, we challenged the drone with 20 different random obstacle forests (trajectories not shown) given no obstacle detection, optic flow method, and object motion filter method. The success rates were 4/20, 11/20, and 9/20 respectively. The results suggest that our motion vision steering control implementation improved the obstacle flight success and the object motion filter method produced similar performance as the optic flow method.

VI. Conclusion and Discussion

In general, the proposed object motion filter steering controller doubled the success rate in passing the obstacle field compared to chance. Given the same control heuristics, it achieved similar performance as a standard optical flow process (i.e. Farneback) [24], but with less computational complexity. The Big O notation has been often used to quantify the scaling of computational complexity given the input data size **n**. For the motion vision processing part (**Fig 1**) where we could substitute an optic flow algorithm, the object motion filter has O(n) [15] which is the minimum possible complexity for an image processing algorithm. A classic sparse optic flow method Lucas-Kanade would require $O(n^2\ N+\ n^3)$ [25][26] and a state-of-the-art target tracking algorithm Kernelised correlation filter (KCF) would need $O(n^3)$ [27]. Even an efficient dense optic flow algorithm can easily scale in quadratic rate depending on the kernel setup [24]. Finally, object motion filter method could be less susceptible to visual noise as it has built-in selectivity to moving objects we expect. Comparing the control signal generated by different methods would be useful if we manage to normalise the visual inputs from different trajectories.

There are several issues we must consider for future studies. Firstly, a properly tuned flight controller is needed if we want to resolve performance details of motion vision. In the current implementation, the motion vision steering

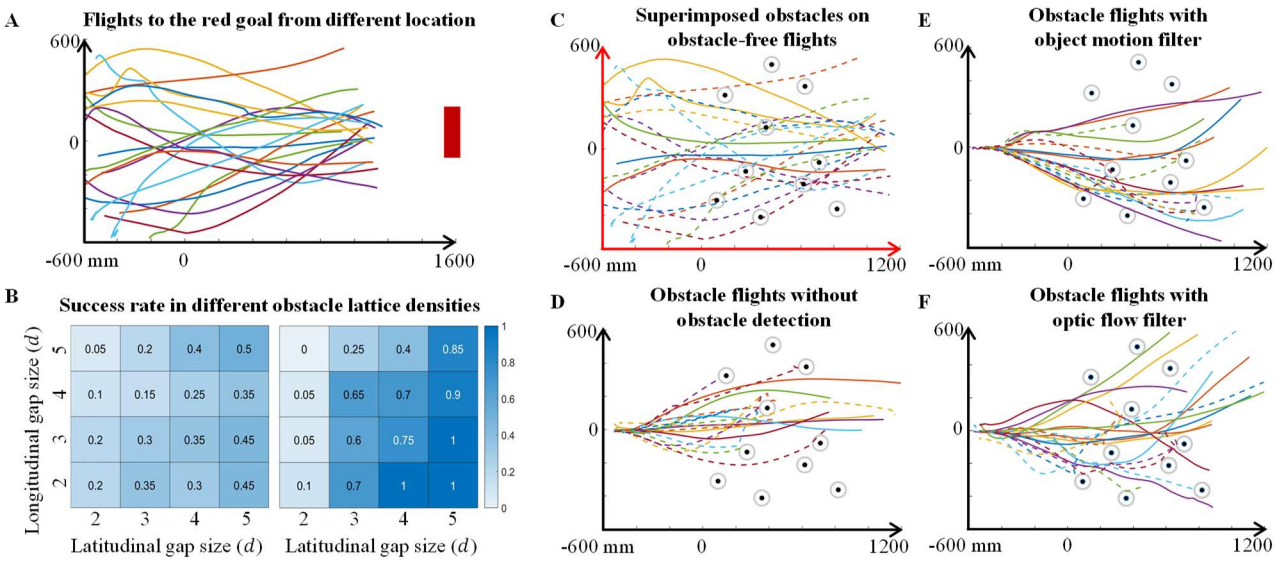

Fig. 5: Obstacle flight analyses. (A) The drone flew to the red goal given different starting positions in the obstacle-free arena. Most of the time it maintained tracking of the red goal with a few exceptions. (B) Obstacle flight success rate in lattice layout without (left) and with (right) object motion filter activated. We recorded 20 valid flight trials per condition (i.e. the drone flew through the obstacle field instead of around it). (C) A pseudo-random obstacle forest layout (black dots represent obstacles in the work space while grey circles mark the drone half-width margin around each obstacle) superimposed on trajectories in (A) provided a reference scenario in which the drone flies through the random forest from different starting positions. (D) Trajectories of the drone flying through the random obstacle forest without obstacle detection implemented, (E) with object motion filter and (F) with optic flow method from the same starting position.

control cannot cope with precision steering. Consequently, it performed worse than chance at high obstacle density, as any control signal increased the chance of obstacle collision. Not surprisingly, the easiest obstacle density configuration was with the lowest latitude density and highest longitudinally density. Intuitively, this forms a sparsely spaced obstacle array with very shallow depth, for which precision steering is not needed. Scrutinizing the unsuccessful obstacle flights through random forests reveals that the most common problem was overshooting the control setpoint as determined by the largest gap. The drone often steered strongly into a gap and failed to direct its inertia to move toward a gap in the second layer of obstacles. This was somewhat unavoidable given a rudimentary flight steering controller with a fixed pitch offset for forward flight. This can be addressed with more formal steering controller in the future.

The second issue we should consider is scaling/tuning the model RTCs. Due to on- and off- channel coupling, the trailing edge of an object inherently invokes higher responses as the leading edge has primed the pixels. However, this effect depends on the visual size and speed of the object. Such characteristics resulted in a delay in detection and locating of fast-moving objects. The ratio of two RTCs provides some compensation to this issue by pushing the response toward the leading edge closer to the actual object location. Whether such compensation is sufficient as we speed up the drone remains to be determined. In addition, we are currently employing only two model RTCs to generate proximal-distal object separation. The fixed threshold operation does not allow adaptive control and thus more continuous identification of obstacles. Future implementations should incorporate a small population of model neurons (instead of two) to cover a wider range of motion conditions. By taking different combinations of RTCs, we could implement detections of different motion features in the world.

The last issue is the quantification of performance metrics. While there are several standard benchmark datasets for computer vision, there is no such framework for assessing the perception-action implementation of obstacle negotiation. In our current experimental setup, the initial path of the trajectories largely depended on the initial visual position of the goal, with some influence from the drift of onboard inertia measurement unit (IMU) for self-stabilization. As a result, no trails shared the same initial visual motion even when the drone started from the same position. Is it fair to compare trajectories with very different sensory inputs? We did notice that entering the obstacle forest from the same gap and similar angle generated similar trajectories inside the forest. Should we compare only these trajectories across different motion vision algorithms? These are questions we should address in the future as perception and action are highly coupled in a mobile system.

In conclusion, our object motion filter vision architecture with its low computational complexity achieved notable performance to validate the basic concept. In fact, even in the perfect world of simulation, the steering heuristics of aiming for the largest visual gap typically predict 70~80% of the successful obstacle flights by birds given similar obstacle scenarios [23]. The 50~60% success rate we achieved with a rudimentary steering controller is promising.

There are several obvious directions we should go for future development. Firstly, having a better state estimation and flight controller will eliminate some failures simply due to the loss of control. It should also allow us to fly at a larger range of speed. Secondly, incorporating self-motion compensation (e.g. forward model) for the yaw control could reduce the noise in motion detection in the system. Indeed, proximity estimation via visual motion works the best when the self-motion is tightly controlled [28]. Finally, we have so far used simple colour segmentation for the goal-directed behaviour in order to focus on the obstacle negotiation task. Since the RTC model was originally designed for target tracking applications [15], we should be able to leverage a population of RTCs to perform both goal-directed behaviour and obstacle negotiation simultaneously. This would require integration of all motion information and compare with an internal model of the body state similar to what insects do for challenging visual guidance tasks [29].

ACKNOWLEDGEMENTS

This work was supported by European Research Council Horizon 2020 ERC Starting Grant to Dr Huai-Ti Lin. We would like to thank Dr Joseph Fabian and Prof Holger Krapp for their discussions about insect motion vision. We would like to thank two anonymous reviewers, Dr Jiaqi Huang, Mr Daniel Ko, and members of **NBits Lab** for comments on the manuscript.

REFERENCES

[1] J. Borenstein, and Y. Koren "Real-time obstacle avoidance for fast mobile robots." *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187, 1989.

[2] M. Geyer, and J. Eric, "An integrated top-down approach to 3d obstacle avoidance for unmanned aerial vehicles." in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6825.

[3] N. Rehman, and K. Kundan, "Implementation of an autonomous path planning & obstacle avoidance UGV using SLAM." in *International Conference on Engineering and Emerging Technologies (ICEET)*, 2018, pp. 1-5.

[4] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning." in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252-6259.

[5] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun. "Off-road obstacle avoidance through end-to-end learning." in *Advances in neural information processing systems*, pp. 739-746. 2006.

[6] I. Ulrich, and I. Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision." in *Proceedings of the International Conference on AAAI/*IAAI, 2000, pp. 866-871.

[7] K. Souhila, and K. Achour, "Optical Flow Based Robot Obstacle Avoidance." in *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, p. 2, Mar 2007.

[8] M. Blanchard, F. Rind, and P. Verschure, "Collision avoidance using a model of the locust LGMD neuron." in *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 17-38, Jan 2000.

[9] J. Serres, and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots." in *Arthropod structure & development*, vol. 5, no. 11, 2017.

[10] D. S. Levkovits-Scherer, I. Cruz-Vega, and J. Martinez-Carranza, "Real-Time Monocular Vision-Based UAV Obstacle Detection and Collision Avoidance in GPS-Denied Outdoor Environments Using CNN MobileNet-SSD." in *Martínez-Villaseñor L.,*

Batyrshin I., Marín-Hernández A. (eds) Advances in Soft Computing. MICAI, Lecture Notes in Computer Science, vol 11835. Springer, Cham, 2019.

[11] S. Badrloo, and M. Varshosaz, "Vision based obstacle detection in UAV imaging." in *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 21, 2017.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." in *IEEE Transactions on Robotics,* vol. 31, no. 5, pp. 1147-1163, Oct 2015.

[13] J. S. Humbert, and A. M. Hyslop. "Bioinspired Visuomotor Convergence." in *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 121-130, Feb 2010.

[14] M. V. Srinivasana, J. S. Chahla, K. Weber, S. Venkatesh, M. G. Nagle, and S. W. Zhang, "Robot navigation inspired by principles of insect vision." in *Robotics Auton. Syst.,* vol. 26, no. 2-3, pp. 203-216, 1999.

[15] Z. M. Bagheri, S. D. Wiederman, B. S. Cazzolato, S. Grainger and D. C. O'Carroll, "Performance of an insect-inspired target tracker in natural conditions." in *Bioinspiration & biomimetics*, vol. 12, no. 2, p. 025006, Feb 2017.

[16] D. C. O'Carroll, and S. D. Wiederman, "Contrast sensitivity and the detection of moving patterns and features." in *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 369, no. 1636, p. 20130043, Feb 2014.

[17] Y. Tian, W. Hu, H. Tong, and J. Han, "Phototransduction in Drosophila." in *Sci. China Life Sci*, vol. 55, no. 1, pp. 27-34, Jan 2012.

[18] H. Yang, and T. Clandinin, " Elementary Motion Detection in Drosophila : Algorithms and Mechanisms." in *Annual Review of Vision Science*, vol. 4, pp. 143-163, Sep 2018.

[19] S. D. Wiederman, P. A. Shoemaker, and D. C. O'Carroll, "A model for the detection of moving targets in visual clutter inspired by insect physiology." in *PloS one*, vol. 3, no. 7, 2008.

[20] O. J. N. Bertrand, J. P. Lindemann, and M. Egelhaaf, "A bio-inspired collision avoidance model based on spatial information derived from motion detectors leads to common routes." in *PLoS computational biology*, vol. 11, no. 11, Nov 2015

[21] R. S. A. Brinkworth, and D. C. O'Carroll, "Robust models for optic flow coding in natural scenes inspired by insect biology." in *PLoS computational biology*, vol. 5, no. 11, Nov 2009.

[22] M. Egelhaaf, A. Borst, and W. Reichardt, "Computational structure of a biological motion-detection system as revealed by local detector analysis in the fly's nervous system," in *J. Opt. Soc. Am. A*, vol. 6, no. 7, pp. 1070-1087, Jul 1989.

[23] H. Lin, I. G. Ros, and A. A. Biewener. "Through the eyes of a bird: modelling visually guided obstacle flight." in *Journal of The Royal Society Interface*, vol. 11, no. 96, p. 20140239, Jul 2014.

[24] G. Farnebäck. "Two-frame motion estimation based on polynomial expansion." in *Proceedings of the 13th Scandinavian conference on Image analysis (SCIA'03)*, Springer-Verlag, Berlin, Heidelberg, Jun. 29, 2003, pp. 363–370.

[25] B. D. Lucas, and T. Kanade. "An iterative image registration technique with an application to stereo vision." in *Proceedings of the 1981 DARPA Image Understanding Workshop*, April, 1981, pp. 121-130.

[26] S. Baker, and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework." in *International Journal of Computer Vision,* vol. 56, pp. 221–255, 2004.

[27] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters." in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37 no. 3, pp. 583-596, 2014.

[28] J. V. Huang, Y. Wei, and H. G. Krapp, "A biohybrid fly-robot interface system that performs active collision avoidance." in *Bioinspir. Biomim.,* vol. 14, pp. 065001-065001, 2019.

[29] M. Mischiati, H. Lin, P. Herold, E. Imler, R. Olberg, and A. Leonardo "Internal models direct dragonfly interception steering." In *Nature*, vol. 517, no. 7534, pp. 333-338, 2015.