Towards drone racing with a pixel processor array

Colin Greatwood[1], Laurie Bose[1], Thomas Richardson[1], Walterio Mayol-Cuevas[1], Robert Clarke[1]
Jianing Chen[2], Stephen J. Carey[2] and Piotr Dudek[2]

Abstract

Drone racing is an interesting scenario for an agile MAV due to the need for rapid response and high accelerations. In this paper we use a Pixel Processor Array (PPA) demonstrating the marriage of perception and compute capabilities on the same device. A Pixel Processor Array (PPA) consists of a parallel array of processing elements, each of which features light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the sensor itself. This paper presents the use of a PPA for gate detection and location in a typical drone racing scenario. Conventional sensing techniques typically require significant processing overheads on separate hardware, resulting in lower frame rates and higher power consumption than is possible to achieve with a PPA. The results given here demonstrate gate detection and location with real-time planning to account for uncertainty in the gate location. Additionally, the PPA only needs to output specific information such as the estimated target location variables, rather than having to output entire images. This significantly reduces the bandwidth required for communication between the sensor and on-board computer, further enabling a high frame rate, low power operation.

1 Introduction

Autonomous Drone Racing (ADR) requires a Micro Air Vehicle (MAV) to fly with high speed, agility and accuracy. This agile control also requires suitable perception that can enable flight through small racing gates and around obstacles. An aircraft with such capabilities would be of great use in many future robotics applications. In such a vehicle it is important to minimise the size, mass and power consumption of on-board components such as sensors and processors. Image sensors must also be able to cope with scenes that move at high speed without suffering from motion blur. Sensing must be rapid, accurate, robust and take place with minimal delay in order to allow for the rapid control decisions required for precision flying and obstacle avoidance. Pixel Processor Arrays (PPAs) are a great fit for ADR and many other robotics applications due to their size, speed and low power requirements.

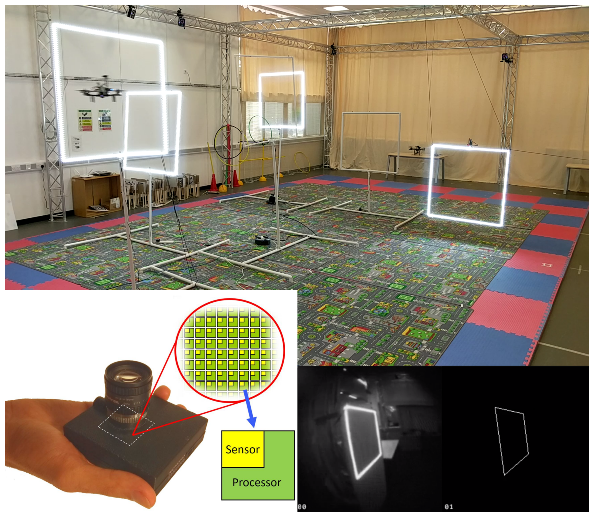

Figure 1: Racing layout; PPA structure; and the SCAMP-5 system tracking a single gate.

Autonomous navigation for aerial robotics has historically leveraged Simultaneous Localisation and Mapping (SLAM), where powerful single board computers or FPGAs process information from vision sensors such as stereo cameras and RGB-D cameras e.g. [1–5]. More recently in ADR competitions however, focus has shifted to high frame rate estimation of the platform's motion, rather than rapidly generating a detailed map of the environment. This rapid frame estimation allows for agile control and manoeuvring of the vehicle and enables accurate map generation to take place, both at a lower frame rate and through post flight data processing.

Another direction recently under development is the use of delta sensors or dynamic visual sensors such as the DVS, reporting only pixel event locations that changed intensity. Such sensors have very low latency and energy usage, however their data often needs to be deeply processed and events pooled to construct whole images before being usable for navigation [6].

Targeting the current interest in ADR, this paper presents an autonomous drone racing strategy using a Pixel Processor

*This work was conducted at the Bristol Robotics Laboratory

[1]Faculty of Engineering, Aerospace and Computer Science, University of Bristol, Bristol, England
thomas.richardson@bristol.ac.uk

[2]School of Electrical and Electronic Engineering, The University of Manchester, Manchester, England p.dudek@manchester.ac.uk

Array (PPA) camera to estimate gate locations at an average frame rate of 500 Hz. The approach taken here is similar to the previous competition winning method used at IROS Autonomous Drone Racing 2018 [7]; A key difference being the onboard sensing. Where the authors of 2018 [7] used deep-learning to estimate relative gate pose at 10 Hz on an Intel UpBoard, this work instead uses a PPA sensor to detect the racing gates and provide information to the control systems.

In contrast to many conventional image sensors, PPA sensors, for example the SCAMP-5 system used in this work, are capable of high frame rate, low latency, vision processing workloads. This has the potential to greatly reduce the workload exacted on any associated on-board computer. PPA sensors consist of a parallel array of processing elements, each featuring light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the sensor [8,9]. Crucially, the PPA is capable of outputting only required information such as the estimated location of a target, rather than having to output entire images. This vastly reduces the bandwidth required per frame in communication between the sensor and on-board computer, enabling high frame rate, low power sensing.

Figure 1 shows the overall layout of the gates and their brightness relative to the environment. The lower portion of the figure shows the raw SCAMP-5 system image captured alongside the binarized image containing the extracted gate.

The following section outlines the strategy taken for gate detection and vehicle control. Section 3 provides the experimental setup in the flight arena at the Bristol Robotics Laboratory; and Section 4 provides the test results from autonomous flight tests of the drone with the on-board SCAMP-5 system sensing the gates, with real-time planning to allow for uncertainties in their position.

2 Method

The system presented here follows recent autonomous drone racing strategies, splitting the problem into perception, and combined planning and control. The novelty, contribution and focus of this paper is in the programming and use of the SCAMP-5 system for high frame rate detection and localization of the racing gates. Detected gates given in the vehicle's frame of reference are combined with the vehicle's state estimate to produce a filtered estimate of the gate poses. The planning and control is subsequently performed for the flight tests using the Perception Aware Model Predictive Controller (PAMPC) framework presented by Falanga et al. [10].

2.1 *Gate Sensing using the SCAMP-5 System*

Gate detection is performed using a SCAMP-5 system attached to the front of the vehicle. The system is programmed to detect potential gates within each frame, sending only their size and location within the image frame back to the on-board computer, hence consisting of only a short stream of bytes per frame. This significantly reduces both the computation overhead in the controller, and bandwidth required for communication with the sensor. Additionally the gate detection algorithm is designed to exploit the parallel features of PPAs, which when combined with the small data transfer per gate, allows the algorithm to be performed at frame rates of up to 1500 Hz. Performing detection at this frame rate has the added benefit of allowing for a short exposure time, effectively eliminating motion blur, and improving gate detection accuracy under rapid camera motion.

2.2 *PPA Algorithms*

The approach used to detect gates makes heavy use of two different pieces of functionality on the SCAMP-5 system. First the ability to locate a set pixel (ie. pixel of value of +1) within a binary image stored upon the PE array, and secondly the ability to perform a parallel flood fill upon such a binary image. A second binary image is used to control this flood fill operation, restricting flooding propagation to only pixels which are set in this control image.

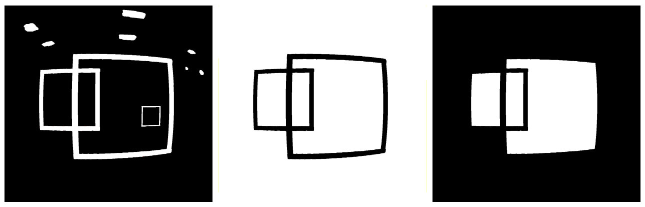
Figure 2: Left to right, a white shape is extracted from the binary camera image, inverted, and then flooded black from the image bounds leaving only the contained shapes within.

Using these features gate detection proceeds by extracting separate shapes from within a binary thresholded camera image, and then extracting the shapes contained within each of these as shown in Figure 2. Different combinations of the shapes contained within each extracted shape are the tested to determine if they constitute a potential gate. This simply involves extracting approximations of the four corners of the gate and evaluating how well the polygon spanning these vertices fits the shapes as illustrated in Figure 3. Pseudo Code for this algorithm is listed in Algorithm 1.

2.3 *Gate Positioning*

The PPA's output is transformed into an estimated gate pose relative to the drone through knowledge of the camera's intrinsic parameters. The vehicle's state estimate is then used to transform these relative gate poses into the world frame. The world frame positions are compared with a set of prior estimates provided to the vehicle before flight. Only gate updates that fall close enough to the existing gate predictions are accepted as valid gate updates used to drive the flight path.

2.4 *Control*

The control architecture follows that of Falanga et al. [10]. With the PAMPC controller, there are two objectives; the first is to have the vehicle follow a reference trajectory; and the

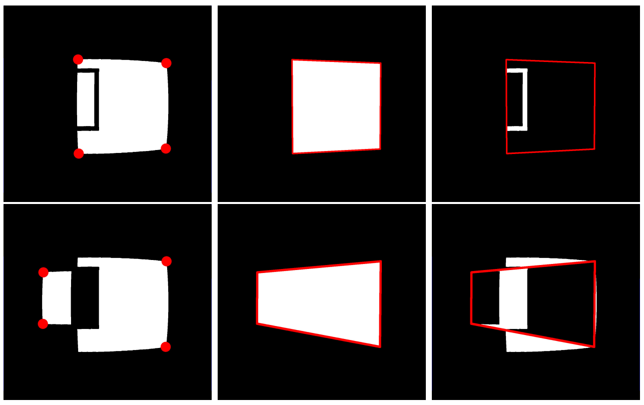
Figure 3: Two examples of combining and testing contained shapes from Figure 2 as potential gates. Approximate gate corners are extracted and a filled polygon fitted to these vertices. An XOR is then performed between these two images, with the number of remaining pixels indicating how closely this polygon fitted the shape.

second is to bring the next gate into the field of view of the SCAMP-5 system. In this work, the controller acts over a 2 s horizon.

The reference trajectory is a simple linear progression along a set of way-points, and for this work maintains a constant speed along the whole path. This can result in physically unrealisable accelerations at the transitions between each pair of way-points. This can in turn limit the maximum speed that the reference trajectory can be generated for while having the vehicle successfully follow it. The way-points are generated in pairs either side of each gate, such that the reference trajectory passes through the centre of each gate, to ensure as tight tracking as possible to the centre of each gate. The offset of the way-points either side of the gate was set at 1.5 m, which worked well with the chosen reference trajectory speed. In addition to these generated way-points, a start and end way-point were added to the overall list.

In this work, the PAMPC controller's point of interest, which controls where the vehicle attempts to point the PPA, is set to be the way-point after the next gate. This provides the fastest possible acquisition of the next gate of interest using the onboard PPA. These gate updates are subsequently used in real-time to update the reference trajectory by updating the endpoint for the reference trajectory generation. This rather coarse approach to the control results in some sharp discontinuities in the reference trajectory when a gate update is incorporated. However, this was found to be sufficient for these tests. Future control work will focus on smoothing out the transitions in the reference trajectory when a gate estimate is incorporated, and generating a smoother reference trajectory accounting for the gate topology.

Algorithm 1 $Extract\_Gates(A, \alpha)$

```
Input and Output
$A$    //Binary Camera Image
$\alpha \in \mathbb{N}$    // Gate Error Threshold In Number Of Pixels
$G$    // List of detected Gates as corners

while $Global\_OR(A)$ do
    $n = 0$  //Reset inner Shapes Counter
    $B = Extract\_Shape(A)$  //Extract White Shape
    $C = Flood\_From\_Edges(NOT(B))$
    $B = NOT(OR(B, C))$  //Get inner shapes
    while $Global\_OR(B)$ do
        $S_n = Extract\_Shape(B)$  //Extract inner Shape
        $n++$  //Increment inner Shape Counter
    end while
    for $i = 0$ to $n$ do
        for $j = 0$ to $n$ do
            $S = OR(S_i, S_j)$  //Combine inner shapes
            $corners = Extract\_Gate\_Corners(S)$
            $C = Draw\_Filled\_Polygon(corners)$
            $C = XOR(C, S)$  //Generate Error Image
            $Err = Count\_Set\_Pixels(C)$
            if $Err < \alpha$ then
                $G = G \cup \{corners\}$  //Add corners List $G$
            end if
        end for
    end for
end while
return $Gates$
```

3 Experimental Setup

3.1 SCAMP-5 System

Gate detection was performed using SCAMP-5 system [9, 11, 12] specifically programmed for the task. No other device was used in directly processing visual data. The SCAMP-5 system integrated circuit features an array of 256×256 processing elements (PEs), each capable of light capture, storage and processing of visual data - effectively putting a small "microprocessor" inside every pixel of the sensor array. The pixels feature a photosensor, local analogue and digital memory, and the ability to perform various logic and arithmetic operations. The SCAMP-5 system is attached to the front of the vehicle as shown in Figure 4. Each PE may also communicate with its four neighbouring elements in the array, making it possible to transfer register data across PEs. A programmable controller chip issues identical instructions to each PE, which then all perform said instruction simultaneously. In this way processing follows the standard single instruction multiple data (SIMD) approach and allows for efficient parallel processing. Vision algorithms can then be performed directly upon the pixel array, without ever transmitting the images out of the sensor.

By only sending the meaningful data such as the values relating to gate locations, there is a significant decrease in the bandwidth and hence power required during operation. This approach allows many visual tasks to be conducted at very high frame-rates (such as at 100 000 fps in [9]), something typically not possible using the standard visual processing pipeline. SCAMP-5 system is also low power, requiring below 2 W, which compares well with GPU-based approaches that while parallel, require 10s-100s of Watts.

3.2 *Flight Hardware*

A custom quadrotor, shown in Figure 4, was designed and built to carry the SCAMP-5 system, it weighs 1kg with the PPA installed and measures 400 mm diagonally between rotors. In the work presented in this paper, the sensor was mounted facing forwards with a 4.5 mm lens providing a 107° field of view.

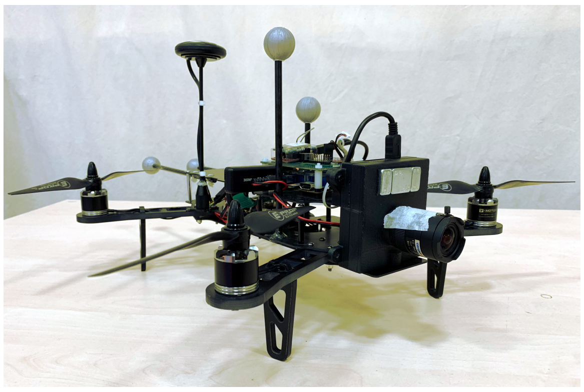
Figure 4: Custom quadrotor used for experiments. SCAMP-5 system facing forwards for gate sensing.

An ODROID XU4 single board Linux computer is fitted to the top of the quadrotor and enables the SCAMP-5 system and 'Pixhawk' autopilot to both be integrated within the Robot Operating System (ROS) for rapid development and system testing. Data is passed from the SCAMP-5 system over USB to the ODROID, whilst flight data from the Pixhawk is sent via a serial UART link. These communication links are summarised in Figure 5. If the ROS system was not used, the SCAMP-5 system has an M4 processor that could be used to carry out the computations currently programmed on the ODROID, and it could talk directly to the Pixhawk with the available serial link. The final overall mass of the system could therefore be reduced significantly if the requirement for rapid development were removed.

The outer loop control system runs on the ground, with only the low-level attitude controller running on-board the vehicle. Control inputs are sent over a Laird RM024 whilst data from the SCAMP-5 system is sent back to ground over WiFi using TCPROS. Position information for the vehicle is provided by a series of Vicon cameras and associated tracking software. The same system is used to track the gate positions, thereby providing a ground truth for the gate position estimates.

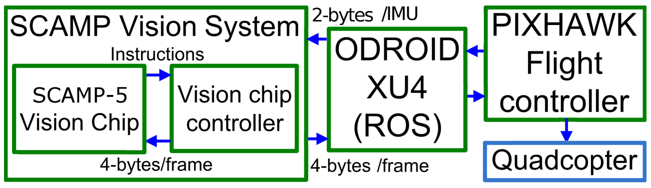

Figure 5: Block diagram of hardware. ODROID is used for rapid development and debugging with ROS and passes data between flight controller, SCAMP-5 system and the ROS system. It does not do any further computation.

3.3 *Initial Testing*

Six square gates were constructed with an array of LEDs around the outer edges and a width and height of 1 m. These gates were then positioned at various heights and locations to form a course, representative of the one used in IROS 2018. No particular consideration was given to the visibility of the gates while traversing the course. In this work, they were always vertical, although the method could be extended to deal with inclined gates. Figure 6 shows a top-down map of the course, highlighting the overall size, direction and numbering of each of the gates.

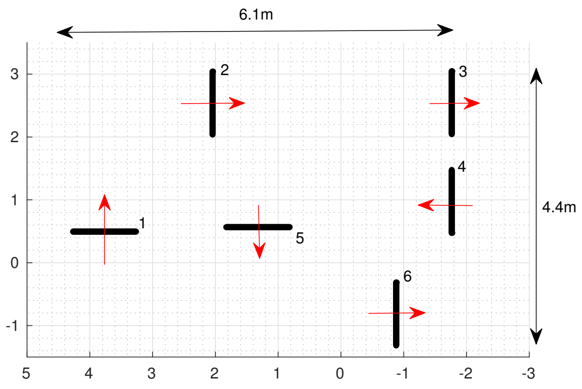

Figure 6: Map of the course

Initial testing of the system was carried out using simulated SCAMP-5 system output feeding back to the the PAMPC controller. This simulator used the output of the Vicon tracking to generate idealised inputs for the gate estimation. The system was found to be highly susceptible to lag, increasing the motivation for the low latency image processing pipeline provided by the SCAMP-5 system. In parallel with the testing of the controller, the SCAMP-5 system output was compared to the ground truth reference points measured using Vicon

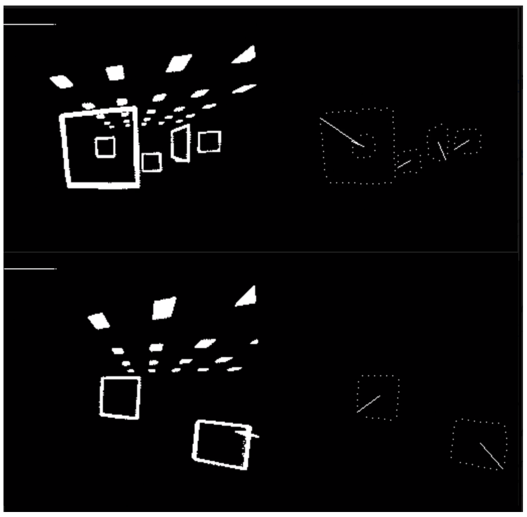

Figure 7: SCAMP-5 system simulated output for multiple gates

Figure 7 shows some examples of simulated binarized frames captured by the SCAMP-5 system used for testing. From this image, the algorithm running on the system detects any gates present and transfers their image location and size to the on-board computer. This information is then utilised by the controller to update the vehicle's estimate of the gate positions. In these examples a number of gates are visible along with the ceiling lights of the arena, whose shapes are rejected by the gate detection algorithm.

4 Results

Initial experimentation with the SCAMP-5 system output feeding into the control loop was carried out with the speed of the reference trajectory set to $1.5\,\mathrm{m\,s^{-1}}$. The gates were within $5\,\mathrm{cm}$ of the positions given in the prior estimates provided to the system. The vehicle successfully traversed the course completing it in approximately $17.5\,\mathrm{s}$. Figure 8 shows the reference trajectory and the measured vehicle position relative to the prior estimate and measured gate positions for this $1.5\,\mathrm{m\,s^{-1}}$ run. Of note are a number of sharp discontinuities in the reference trajectory caused by the gate updates shifting the end way-points for the linear trajectory generation. As mentioned previously, future work will focus on smoothing the effect of incorporating the updated gate estimates. There is very little shift between the measured gate positions and those provided as prior estimates as seen by the close correspondence of the dotted and solid gate positions.

For the following set of results, selected gates were moved prior to the flights. Three of the gates in the course were shifted laterally by up to 75% of a gate width, namely gates 2, 4 and 6. The gates remained in approximately the same plane as their pre-shift positions, though it should be noted that this is not required for the control strategy selected. The reference speed was also increased to $2.3\,\mathrm{m\,s^{-1}}$. With

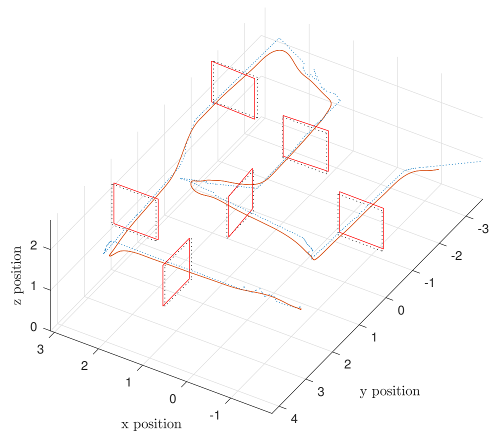

Figure 8: Plot showing track taken by drone through gates. No intentional shift of gates, reference speed $1.5\,\mathrm{m\,s^{-1}}$. Reference trajectory and prior gate estimates dotted, measured trajectory and positions solid.

the current control implementation, this allowed for robust and repeatable completion of the course. Figure 9 shows successful completion of the course for this reference speed in approximately $12.5\,\mathrm{s}$. The shift in gate positions can be seen by the offset between the prior estimates provided (dotted) and those measured by the Vicon system (solid). The reference trajectory is noticeably noisier than that seen in Figure 8 which was at a refernce speed of $1.5\,\mathrm{m\,s^{-1}}$, but future smoothing of the gate updates will solve this problem. Variable reference trajectory speed and way-point offsets will also allow the overall speed to be increased.

Table 1 provides the maximum speed along all three axes and the maximum overall velocity during this run. The maximum roll and pitch angles experienced by the vehicle are also given.

Value	Absolute Maximum
x-velocity	$3.04\,\mathrm{m\,s^{-1}}$
y-velocity	$2.83\,\mathrm{m\,s^{-1}}$
z-velocity	$1.72\,\mathrm{m\,s^{-1}}$
Total velocity	$3.14\,\mathrm{m\,s^{-1}}$
Roll	$39.4°$
Pitch	$44.0°$

Table 1: Maximum values reached during the run shown in Figure 9

Figure 10 shows the location of the vehicle as it passes through each gate relative to the mean estimated gate position. The plot spans the overall cross-section of the gate. It can be seen that these are closely grouped, indicating that the vehicle is consistently passing through the gate at the targeted

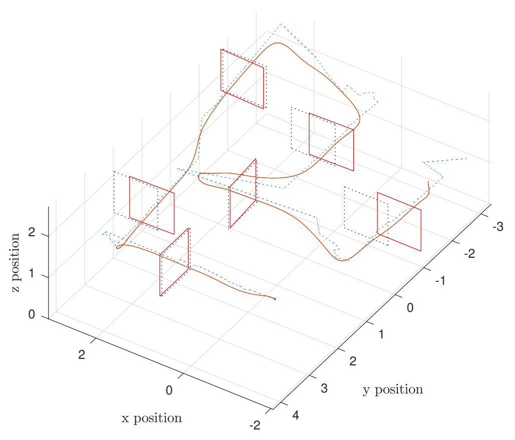

Figure 9: Plot showing track taken by drone through gates. Gates 2,4 & 6 shifted, reference speed $2.3\,\mathrm{m\,s^{-1}}$. Reference trajectory and prior gate estimates dotted, measured trajectory and positions solid.

location, as identified by the SCAMP-5 system tracking estimate. The vehicle trajectory has also been included for $0.5\,\mathrm{m}$ on both sides of each gate pass-through, indicating that the vehicle is consistent both in the approach and the departure for each gate. The pass-through locations for all six gates are within a $20\,\mathrm{cm}$ square, which together with the possible improvements identified above, indicate that there is still significant improvement possible in terms of maximum speed, acceleration and the minimum time to complete the course.

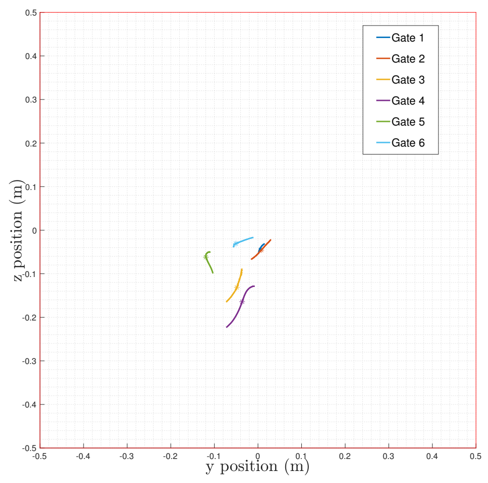

Figure 10: Accuracy of flight path relative to the estimated gate positions. Gates 2, 4 & 6 shifted. Plot spans overall size of gate, $0.5\,\mathrm{m}$ of the trajectories either side of the gate are plotted.

Figure 11 shows the location of the vehicle as it passes through each gate in terms of the measured position from the Vicon cameras. This is shown as a ground truth, and it shows little difference in terms of the grouping when compared to Figure 10. From these two plots, it can therefore be concluded that the dynamic SCAMP-5 system driven estimate of the gate position and the vehicle control are both sufficiently accurate for further speed increases. The small difference between the two plots, Figure 10 and Figure 11 could be due to a number of factors, namely a small offset in the orientation and/or position of the SCAMP-5 system on the vehicle; an error in the state estimate of the gate positions; or an error in the measurement of the true gate position. The combined errors though are very small and are not currently the limiting factor with regards to overall vehicle performance.

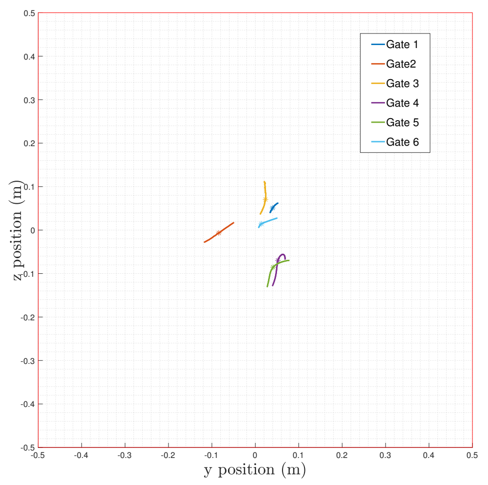

Figure 11: Accuracy of flight path relative to the measured gate positions. Gates 2,4 & 6 shifted. Plot spans overall size of gate, $0.5\,\mathrm{m}$ of the trajectories either side of the gate are plotted.

5 Conclusions

This work has shown that a novel Pixel Processor Array (PPA) device can be used to correct imperfect knowledge of a drone racing course in real-time. The high frame rate achievable with the SCAMP-5 system - i.e. an average of $500\,\mathrm{Hz}$ - has been shown to provide robust and reliable estimates of the true gate positions. This has been carried out on a representative drone racing course, with rapid and significant changes in vehicle trajectory. For the results shown, the position estimate based on the PPA sensing was not found to be the overall limiting factor for speeds and accelerations experienced.

Key limitations in the control strategy used have been identified and these will be addressed in future work to find the limits in terms of speed and acceleration for the scenario considered. These results show the PPAs are likely to be one of a suite of sensors used on future small agile drones when manoeuvring rapidly in an unknown environment.

Acknowledgements and data access statement

Supported by UK EPSRC EP/M019454/1 and EP/M019284/1. The nature of the PPA means that the data used for evaluation in this work is never recorded.

References

[1] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems*, volume 1. Berlin, Germany, 2013.

[2] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 298–304. IEEE, 2015.

[3] Changhong Fu, Adrian Carrio, and Pascual Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 957–962. IEEE, 2015.

[4] Reuben Strydom, Saul Thurrowgood, and Mandyam V Srinivasan. Visual odometry: autonomous uav navigation using optic flow and stereo. In *Australasian Conference on Robotics and Automation (ACRA)*, pages 1–10. Australian Robotics and Automation Association, 2014.

[5] Roberto G Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Ström, and Jizhong Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5233–5238. IEEE, 2014.

[6] A.I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *CVPR*. IEEE, 2018.

[7] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. *arXiv preprint arXiv:1810.06224*, 2018.

[8] Alexey Lopich and Piotr Dudek. A SIMD cellular processor array vision chip with asynchronous processing capabilities. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(10):2420–2431, 2011.

[9] Stephen J Carey, Alexey Lopich, David RW Barr, Bin Wang, and Piotr Dudek. A 100,000 fps vision sensor with embedded 535gops/w 256$\times$ 256 simd processor array. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C182–C183. IEEE, 2013.

[10] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[11] Julien NP Martel, Lorenz K Müller, Stephen J Carey, and Piotr Dudek. Parallel hdr tone mapping and auto-focus on a cellular processor array vision chip. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1430–1433. IEEE, 2016.

[12] Julien NP Martel, Lorenz K Mueller, Stephen J Carey, and Piotr Dudek. A real-time high dynamic range vision system with tone mapping for automotive applications. *CNNA 2016*, 2016.