Tracking control of a UAV with a parallel visual processor

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

***Abstract*— This paper presents a vision-based control strategy for tracking a ground target using a novel vision sensor featuring a processor for each pixel element. This enables computer vision tasks to be carried out directly on the focal plane in a highly efficient manner rather than using a separate general purpose computer. The strategy enables a small, agile quadrotor Unmanned Air Vehicle (UAV) to track the target from close range using minimal computational effort and with low power consumption. To evaluate the system we target a vehicle driven by chaotic dual-pendulum trajectories. Target proximity and the large, unpredictable accelerations of the vehicle cause challenges for the UAV in keeping it within the downward facing camera's field of view (FoV). A state observer is used to smooth out predictions of the target's location and, importantly, estimate velocity. Experimental results also demonstrate that it is possible to continue to re-acquire and follow the target during short periods of loss in target visibility. The tracking algorithm exploits the parallel nature of the visual sensor, enabling high rate image processing ahead of any communication bottleneck with the UAV controller. With the vision chip carrying out the most intense visual information processing, it is computationally trivial to compute all of the controls for tracking onboard. This work is directed toward visual agile robots that are power efficient and that ferry only useful data around the information and control pathways.**

I. INTRODUCTION

The requirements of agile platforms indicate a need to be efficient not only in actuation but increasingly in dealing with visuo-control tasks. In this respect, efforts should be directed to find architectures for sensing, processing and filtering just the relevant information. Recent developments with dynamic vision sensors (DVS), offer some insight into the possibilities for developing more efficiently perceiving robotic systems. Examples for UAVs include works such as [1] for evasive maneuvers, [2] for agile visual odometry and [3] for landing from optic flow. When using a DVS, however, both basic and more visually complex tasks such as target tracking and/or combinations with lower visual competences require separate and more conventional processing architectures. Tasks such as semi-dense mapping with a DVS can be achieved using GPUs [4] but these can consume significant power, up to hundreds of Watts. The contemporary work of [5] and [6] use conventional cameras together with onboard processing to detect a target and estimate agile trajectories for crossing a window gap at speed.

*This work was supported by the Bristol Robotics Laboratory

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

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

In this work, we explore a sophisticated parallel visual architecture based on the SCAMP vision chip [7] that allows for on-sensor agile target tracking, while maintaining high frame rates (>1000 fps) and low energy consumption (< 1.3W). The vision sensor can perform image computations directly at source, in parallel, outputting meaningful processed information such as the x,y coordinates of a target, without requiring any other processing hardware.

II. TRACKING FROM A UAV

UAVs have been used in the past to track ground vehicles with both fixed wing as well as rotary aircraft. Broadly speaking, this work can be split into two categories, one that is concerned with following a target from a distance of 10-100m and another from close range. Tracking ground vehicles from a distance has applications such as surveillance and has been considered by many, including [8], [9] and [10]. At these distances the UAV control often falls into the category of guidance.

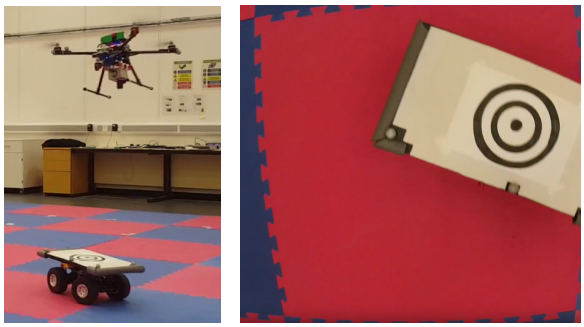

Fig. 1: Quadrotor closely tracking target vehicle and "quadrotor's eye view".

Developments concerned with tracking targets from close range typically focus on maintaining a close relative position and velocity to the target. A popular challenge is to use the visual target for landing such as in [11], [12] and [13] where the authors managed to program the UAVs to land on moving targets. These previous works, however, track cooperative targets that are either fitted with beacons and or GPS informed telemetry links to aid the UAV in acquiring an accurate lock onto the landing zone. Additionally, the target vehicles are often driven at a constant velocity and/or a constant turn rate - making the control also cooperative. In the work presented here we do not yet attempt to command the UAV to land on the tracked vehicle, but rather focus on the non-cooperative nature of the target. The target is non-cooperative in the control; sensing is made possible

by a passive target on the surface of the tracked vehicle; no active beacons or telemetry link is provided between UAV and the vehicle. To illustrate both the usability of the onboard parallel visual processor and to evaluate tracking under agile conditions the vehicle features a visual pattern that can be detected and re-detected at very high framerates with the visual parallel processor. Figure 1 shows the setup we use. The vision system could be programmed to perform more generic template matching and track arbitrary patterns, although this is left to future work.

There have been some very interesting presentations of small UAVs tracking non-cooperative targets in a similar manner. Teuliere et al. [14] have demonstrated a system for commanding a quadrotor to track over a Radio Controlled (RC) car, including the ability to handle visual occlusions whilst the car drives underneath a chair. Gomez-Balderas et al. [15] also demonstrate a method for visually tracking an RC car from a UAV. The visual identification and tracking in these methods, however, rely on a ground PC for computation and so the update rate is limited and the controller must be able to handle communication delays.

Prior work demonstrating visual-based tracking has shown the potential of solely visual based tracking [16] (this algorithm operates on a restricted number of pixels from a frame at 10fps) and pose estimation [17] (designed to operate at 15fps). Related visual based UAV cueing has shown that visual odometry has been achieved at 25fps on a two-camera setup with a 1.5GHz Core 2 Duo processor [18].

The purpose of utilizing a parallel vision sensor is to enable increased control loop speeds by virtue of latency reduction and frame rate increase; the work described here is a step in that direction.

III. Flight Hardware

A 'Tarot FY650 (TL65B01)' quadrotor UAV, shown in Figure 2, was used for tracking a ground rover. The quadrotor was controlled by the flight hardware shown in Figure 3. The vision chip controller streams program instructions to the SCAMP vision chip which acquires and processes the visual information internally - the end result being the image coordinates that represent the bounding box of the target. Only the resulting 4-byte data is read-out from the vision chip and transferred to the ODROID single board Linux computer, which maintains the target state estimation. On the basis of this estimation, instructions are sent to the Pixhawk flight controller to change vehicle x-y position. The height of the quadrotor is controlled via a Vicon motion capture system; the Vicon system additionally permits safe operation of the quadrotor during development of control systems, constraining flight to within a pre-determined space. The SCAMP vision system is fitted directly to the underside of the aircraft pointing downwards. Given the SCAMP-sourced data, the onboard controller is tasked with moving the quadrotor directly above the target. The addition of a gimbal would add mass and physical complexity that is not desired. Indeed, given the capability of the SCAMP it is easily capable of visual tracking during high pitch rates without stabilisation. The lack of gimbal does however, make the controller's task of tracking a target from close range more challenging.

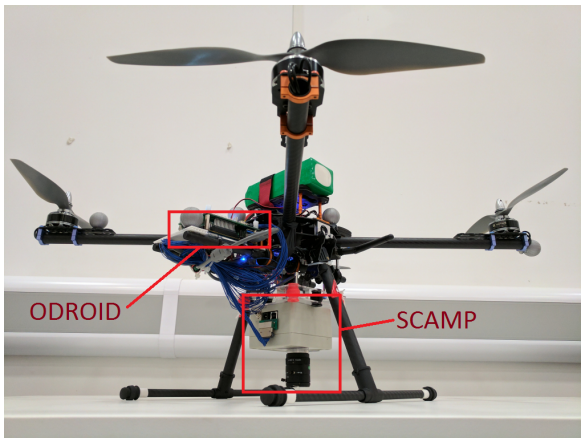

Fig. 2: Photograph of quadrotor used in experiments. Note: The ODROID interfaces the visual processor with the flight controller and **does no** further visual processing.

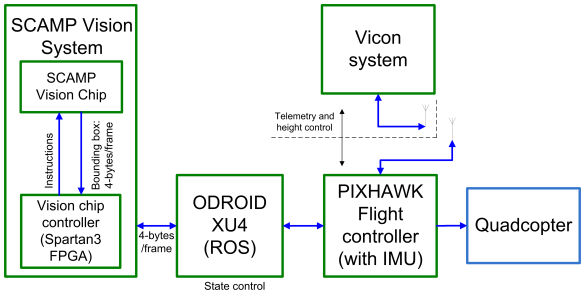

Fig. 3: Block diagram of flight control hardware. Note: The ODROID interfaces the visual processor with the flight controller and **does no** further visual processing.

As the rover moves away, the quadrotor is forced to tilt in order to catch up. This tilt changes the direction in which the camera is facing and in turn causes the location of the target to move across the image plane. For example, if the rover moves forwards then the target would travel up in the image; then, as the quadrotor pitches forwards to catch up the target, the target will move even further up in the image. If the target moves quickly and the quadrotor reacts too quickly, then the target could even leave the image frame. The combination of the non-cooperative movement of the rover and the hard mounting of the camera lead to a difficult control problem.

The rover being tracked was a traditional RC truck measuring about 60cm in length, carrying a 20cm diameter visual target on top. The target and visual tracking are described in Section IV-B. In addition, while tracking markers for a Vicon motion tracking system were attached to enable automated control of height above the rover, this motion tracking system was not used to help the visual target tracking. The rover controller that was developed for this work made it possible to command the rover to drive predefined paths automatically

for quantifying and testing the repeatability of the tracking. A chaotic (double-pendulum) trajectory was also encoded to demonstrate the tracking during unpredictable manoeuvres.

A. The SCAMP and interfacing hardware

Identifying and tracking the target visually is achieved by using the SCAMP vision system; this enables fully onboard information processing rather than via uplinking to a remote workstation. The SCAMP-5 vision chip is a general purpose vision sensor and processor capable of being programmed for a diverse range of tasks [7], [19], [20]. The chip comprises an array of 65,536 processor elements (PEs) - each processor element incorporating a photosensor, local memories and ALU. It is programmed as a single instruction multiple data (SIMD) computer. The processors can carry out basic computational tasks in a parallel manner enabling sophisticated computer vision algorithms to be carried out on chip. Due to the co-location of processors and sensors, these algorithms may be run at extremely high rates, such as 100,000 fps [7].

An ODROID XU4 computer was fitted to the UAV as a development tool. The ODROID is capable of running Robot Operating System (ROS), which enables rapid prototyping of control algorithms and ease of integration with the Pixhawk autopilot. Due to the computer vision algorithms all being executed by the SCAMP, the computational effort actually utilized on the ODROID was trivially minimal. With further work on integration it would be possible to run the control algorithms presented here directly on the Pixhawk and remove the ODROID entirely.

The SCAMP system locates the target in the image frame as described later in Section IV-B and then returns the location, height and width of the target over a Serial Periphery Interface (SPI) link. This Region Of Interest (ROI) data describing the target in the image plane is received by the onboard ODROID computer, which was used to prototype the algorithms described here. Control inputs are computed on the ODROID and sent to the Pixhawk autopilot over a serial link. Data from the autopilot, such as from the Inertial Measurement Unit (IMU) is also retrieved over this serial link.

IV. METHOD

A. Control Architecture

Figure 4 gives an overview of the control structure used by the quadrotor to maintain station over the moving target. First, the ROI data provided by the SCAMP is combined with IMU data from the Pixhawk to compute a relative position between the quadrotor and the rover - as described in Section IV-C. Then the state observer described in Section IV-D is used to estimate the relative position and velocity of the rover before control inputs may be generated using the controller presented in Section IV-E. Finally, these control inputs are sent from the ODROID to the Pixhawk.

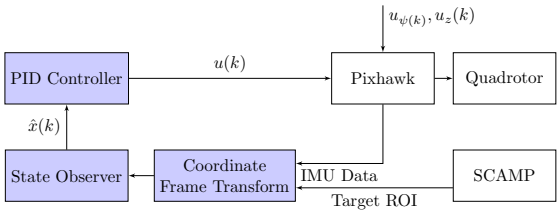

Fig. 4: Control Architecture. Nodes on ODROID shaded blue

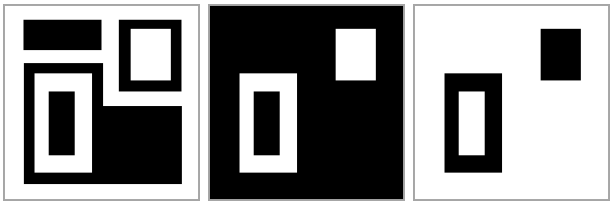

Fig. 5: Flood fill and inversion of a binary image. (a) shows an original image; (b) shows the result after performing a flooding operation from the image boundaries; (c) shows the inversion of (b), making it ready for further flooding.

B. Vision Algorithm

The parallel nature of the SCAMP vision sensor allows various basic image processing tasks to be conducted with minimal computational overhead[1]. Efficient asynchronous flood fill of a binary black and white image is one such task, and it is this capability which is exploited in the target tracking algorithm implemented for this work. Figure 5 illustrates the result of flooding such an image from its boundaries and then inverting the resulting image. It can be seen that this results in the removal of both any solid black shapes and black outer boundaries within the image. Effectively this process eliminates any shapes within the image which are not fully enclosed by the boundary of some larger shape. This process can be repeated iteratively to eliminate any image content which is not fully enclosed within a given number of boundaries. Thus by using a visual target such as shown in Figure 1, consisting of a given number of enclosed shapes, this process of iteratively flooding and inverting can be employed to eliminate all image content but the inner content of the given target itself. This is further illustrated in Figure 6. Once all image content but the target has been eliminated by this process the bounding box of the target, consisting of four bytes, is output from the image plane and communicated to the ODROID using SPI.

Pseudo code for this process is listed in Algorithm 1, in which the latest camera image I is thresholded to a binary image B after which N flood fill and inversion iterations are performed to ensure all image content is removed but the visual target. The target used in this work as shown in Figure 1 required four such flood and invert iterations. Finally the bounding box (described by the left, top, right and bottom

[1]A simulator for the SCAMP architecture is available at http://personalpages.manchester.ac.uk/staff/p.dudek/scamp-sim/

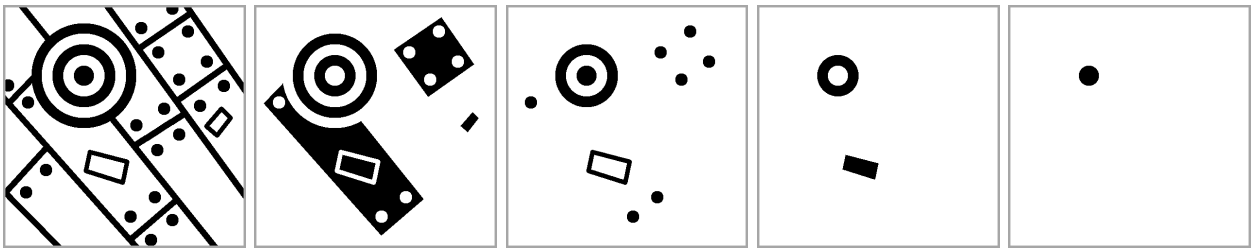

Fig. 6: Tracking Algorithm Stages, showing the result of performing 4 "flood and invert" operations starting from an initial image on the left.

pixel boundaries) of any remaining content is extracted from B and returned over SPI to the ODROID.

Each of these steps can be implemented in a simple and efficient manner on the parallel architecture of the SCAMP hardware. It should be noted that it is possible that the visual target is not the only shape within the camera image contained within N distinct boundaries, which would cause the algorithm to return an invalid bounding box location for the visual target. In most scenarios however this is an unlikely prospect, further the number of boundaries in the visual target used can be increased to increase robustness.

The proposed algorithm requires no prior estimation of the target's location within the image, and will instantly acquire the target given it is fully within the camera image.

This algorithm can be computed efficiently on the parallel visual processor while using $<$1.3 Watts and at rates of $>$1000 FPS under 4000 Lumens (a slightly brighter overcast day at mid-day). In the lower light levels found indoors good performance was found at frame rates of 286 FPS, although this could be increased if desired though gain or using different lenses. The latency was measured, finding that after the end of light integration image processing time to isolate the centre of the target takes $82\mu s$, followed by $6.4\mu s$ to extract the target position and place the data packet in the vision system SPI buffer. The ODROID (as SPI master) then requires an average of $333\mu s$ to transfer the data.

Algorithm 1 $Get\_Target(I, T, N)$
Extract Visual Target Location From Camera Image

INPUT: I *// Camera Image*
T *// Threshold Value*
N *// Flood and Invert Iterations*

OUTPUT: $(p_x^l, p_y^t, p_x^r, p_y^b)$ *// Bounding Box Of Visual Target*

$B = Threshold\_Image(I, T)$
for $n = 0$ to N **do**
  $Flood\_From\_Boundaries(B)$
  $Invert\_Image(B)$
end for
$(p_x^l, p_y^t, p_x^r, p_y^b) = Scan\_Bounding\_Box(B)$

return $(p_x^l, p_y^t, p_x^r, p_y^b)$

C. Coordinate Frames

The quadrotor must be able to estimate the location of the rover in order to track it. The location is estimated relative to the UAV, *i.e.* in the UAV's body frame. It is assumed that the location of the quadrotor is not known and no input is taken from GPS or motion tracking - the camera's output alone is used for measuring horizontal displacements. The height and yaw of the quadrotor weren't considered in this work and so were stabilised independently in a separate controller to the tracking.

Figure 7 shows the pertinent angles of the tracking problem. It would not be sufficient to form a controller around the pixel error in the image plane alone as a non-minimum phase problem would result. Any pitching during the initial forward acceleration, for example, would increase the error in the forward direction. Instead one must use the measured angle of the quadrotor in resolving the distance error.

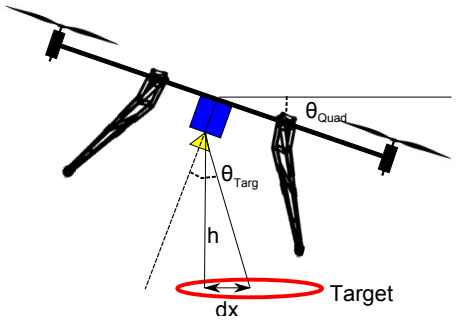

Fig. 7: 2D depiction of quadrotor bank angle and angle to target

The onboard Pixhawk autopilot estimates the quadrotor's roll and pitch angle as part of its stabilisation loop. This angle data is easily accessible via the mavros ROS node. The angles to the target are considered for both the aircraft's x and y axes, but for simplicity just one of those dimensions is described here. If the centre of the target bounding box is reported to be p_x pixels out of the full 255 pixels from the left of the image then the angle to the target (θ_{targ}) may be calculated by

$$\theta_{targ} = \left(2 \times \frac{p_x}{255} - 1\right) \times \frac{FoV}{2} \quad (1)$$

given a horizontal field of view FoV. If the quadrotor banks by θ_{quad} then the horizontal distance to the target (δx) can

be calculated by

$$\delta x = h \tan(\theta_{quad} - \theta_{targ}) \quad (2)$$

D. Target State Estimation

As confirmed in previous work [11] about landing on a moving platform, maintaining position over the target being tracked benefits significantly from measuring the velocity of the target. As the target's velocity cannot be directly observed in the problem described here it must be estimated. A state observer is used to generate velocity estimates, which also has the benefit of providing a filtered position estimate.

The state observer assumes a constant velocity model of the target's dynamics relative to the quadrotor, where the state vector describes the distance and velocity in the horizontal plane at time step k, *i.e.*

$$x(k) = \begin{bmatrix} \delta x \\ \delta y \\ \dot{\delta x} \\ \dot{\delta y} \end{bmatrix} \quad (3)$$

The dynamics of the tracking can therefore be described by

$$x(k+1) = Ax(k) + Bu(k) \quad (4)$$

$$y(k) = Cx(k) \quad (5)$$

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$B = [0] \quad (7)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (8)$$

The rover's control input is unknown and ignored, simply the observations $y(k)$ of position from the camera are used. The state observer is then constructed from

$$\hat{x}(k+1) = A\hat{x}(k) + L(y(k) - \hat{y}(k)) \quad (9)$$

$$\hat{y}(k) = C\hat{x}(k) \quad (10)$$

where the observer gain L is tuned using MATLAB's dqle command and $\hat{x}(k)$ represents the estimated state at time step k.

E. Feedback Control

The quadrotor is driven to maintain position above the target using a standard PID controller. The controller takes advantage of the existing attitude control mode built into the Pixhawk autopilot software, which enables attitude setpoints to be commanded whilst internally stabilising the vehicle from IMU data.

In tuning the controller it was important to find a balance of having a controller that is aggressive enough to keep up with the target, but not so aggressive that the target repeatedly leaves the FoV due to sharp control corrections. The preferred solution found was a fairly aggressive controller with attitude setpoint input saturations of just under fifteen degrees. If the units of error are measured in metres and the output in degrees then the continuous time PID control gains were $[P, I, D] = [18, 4.5, 13.5]$. Whilst it would be possible to design a more sophisticated tracking controller, the immediate motivation here is to demonstrate the effectiveness of integrating a parallel visual processor.

F. Rover Control

The rover was driven using a typical RC remote control connected to a desktop computer. An Arduino connected to the trainer port on the remote control enabled a ROS node to command rover's motor speed, motor direction and steering angle. Waypoint following was achieved by implementing the nonlinear guidance control law presented by Park et al. [21].

V. RESULTS

A. Straight Line Performance

Experiments were carried out to test how well the quadrotor would track the target from a standing start. The quadrotor was commanded to track the rover from stationary up to a constant forward velocity. Upon the quadrotor settling to a steady hover above the target, the rover was given a step input to motor speed and accelerated to a constant speed. Figure 8 shows the resulting time history of the tracking during the experiment.

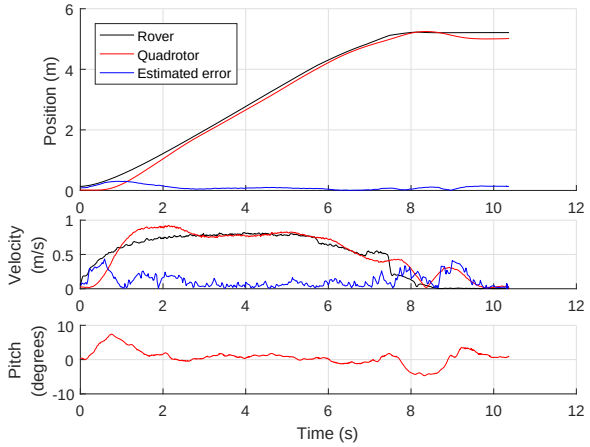

Fig. 8: Straight line tracking performance, step input applied to rover's motor speed

The first subplot compares the rover's position along the path against the quadrotor's; additionally the onboard estimated position error between the two is plotted in blue. The second subplot shows the velocity time history along with the onboard estimated velocity difference. There is a short delay between the rover moving forward and the quadrotor moving forward whilst the errors build up, but also due to the way in which the quadrotor is able to accelerate forward. The third subplot shows the pitch angle of the quadrotor, which must increase first before it can start accelerating.

The quadrotor's control input was limited to reduce the likelihood of the target leaving the camera's FoV during sharp accelerations; this control saturation in turn limits the

maximum lateral acceleration. Figure 9 shows a similar time history of the quadrotor tracking the rover, but at a higher rover motor speed setting during which both the quadrotor's control saturates and the target leaves the FoV. Figure 10 shows snapshots from a computer visualization using ROS's rviz, which illustrates the FoV during the experiment. Once the quadrotor catches up with the rover by frame 3, the target re-enters the FoV the quadrotor snaps to the reverse pitch angle (frame 4) and starts decelerating. Shortly, the target leaves the camera's FoV again. Ultimately, the quadrotor settles above the target (frame 5); a wider field of view lens and further adjustment of the maximum pitch angles could result in uninterrupted tracking of the target.

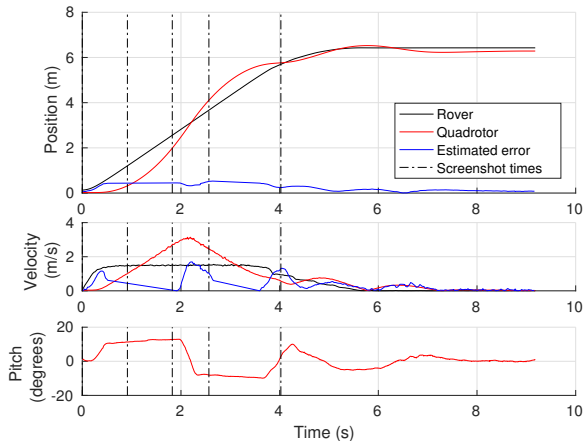

Fig. 9: Straight line tracking performance, step input applied to rover's motor speed

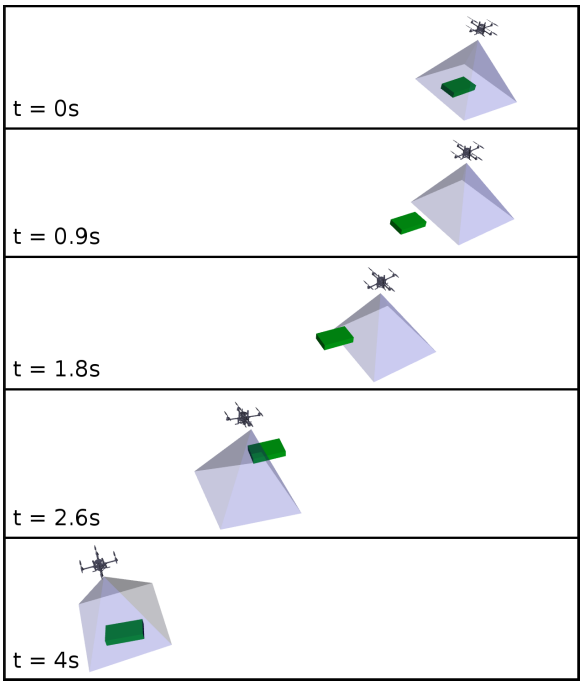

Fig. 10: Visualization of tracking state in Figure 9. Starting at t=0, corresponding frame timings denoted by vertical lines in Fig. 9

B. *Tracking Performance*

For the tracking demonstration, the quadrotor was commanded to follow the non-cooperative rover, which was in turn automatically following a set of waypoints. Two different waypoint paths were evaluated and the quadrotor was programmed to maintain an altitude of just one metre above the target. The first path was constructed of six repeating waypoints that caused the rover to track an oval like shape. The second path was constructed from sampling a simulation of a double pendulum, resulting in a chaotic path. Whilst tracking the double pendulum path, the rover was commanded to change direction if the next waypoint (sampled point) of the path required a turn of more than 90°. The changing of direction meant that the quadrotor had an even harder time tracking due to the starting and stopping of the rover.

Figure 11 shows the paths followed by both the rover and the quadrotor on the oval path. The rover was commanded to pass through six waypoints that caused it to drive under a tunnel that was constructed to occlude the rover from the quadrotor's viewpoint for around half a second. Occlusion from the motion tracking system during the experiment also shows up on the figure as glitches in the reported path taken by the rover. The path taken by the quadrotor does diverge from the rover's during the occlusion due to no further information being available, but at the end of the tunnel tracking resumes and the quadrotor snaps back over the target. This data shows excellent consistency in the tracking around the path for the five laps along with the ability to quickly re-aquire the target after brief moments of occlusion.

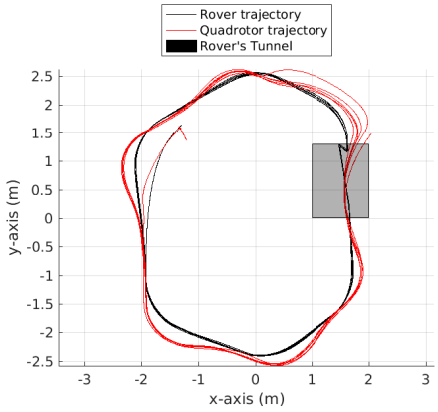

Fig. 11: Tracking performance following rover around an oval pattern. Rover is occluded when it drives through tunnel.

The second trajectory used was evaluated in order to fully demonstrate the non-cooperative tracking concept. By simulating a chaotic system, the path taken by the rover could not be predicted and so the quadrotor's ability to track is clearly just from observations of the target. Figure 12 shows the path taken by both the rover and the quadrotor. Due to the dynamics of the rover it would be impossible for it to follow the double pendulum path exactly, but the chaotic effect is represented well. The tracking control only sees the target

as a point target and does not take into consideration that it could not drive sideways for example. The track shown represents a two minute experiment. The key feature here is that during some of the forward and reversal transitions that take place at around $[0, -1]$ the quadrotor is able to keep up with the rover despite the FoV limitations.

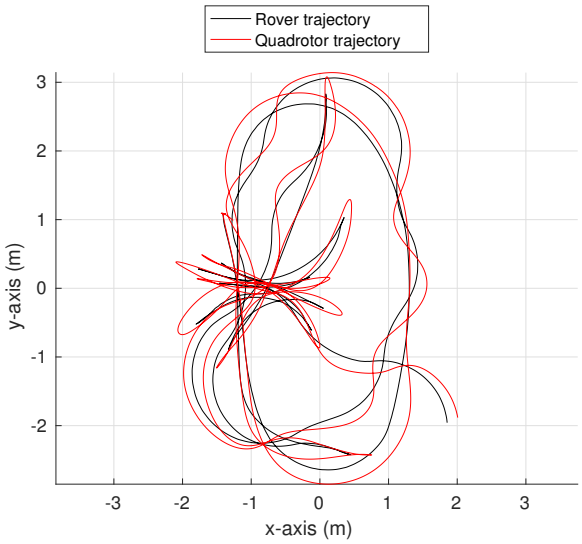

Fig. 12: Tracking performance following rover around a chaotic pattern constructed from simulating a double pendulum

VI. CONCLUSIONS

This work has demonstrated a novel visual parallel processor device applied to agile control of a UAV. Vision sensors are promising for moving agents due to their low energy consumption and ability to compute and forward high level information. This reduces the clogging of the control and information pathways in the rest of the system. We develop a method for tight target tracking on a visuo-control task. The SCAMP vision sensor system gives fast, low latency tracking without further onboard computer vision processing. Our trials suggest that visual targets can be tracked at over 1000 FPS.

The flight hardware framework has been shown to be capable of controlling the UAV's flight through tracking the land target. The UAV - with strapped-down image processing system, allows for a migration path to smaller, more agile air systems. The gimbal-less system showcases platform agility but presents challenges for the control requiring fast reactions to maintain the target in view. The parallel visual processor can be reprogrammed to perform many sophisticated algorithms and we are exploring further visual competencies. Considerable excess processing power exists within the existing framework allowing an expansion in capability in the future.

REFERENCES

[1] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza, "Towards evasive maneuvers with quadrotors using dynamic vision sensors," *European Conference on Mobile Robots (ECMR)*, 2015.

[2] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[3] B. J. Pijnacker Hordijk, K. Y. W. Scheper, and G. C. H. E. de Croon, "Vertical Landing for Micro Air Vehicles using Event-Based Optical Flow," *ArXiv e-prints*, 2017.

[4] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *European Conference on Computer Vision*. Springer International Publishing, 2016, pp. 349–364.

[5] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *IEEE International Conference on Robotics and Automation*, 2017.

[6] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.

[7] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, "A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array," in *VLSI Circuits (VLSIC), 2013 Symposium on*. IEEE, 2013, pp. C182–C183.

[8] M. Zhang and H. H. Liu, "Vision-based tracking and estimation of ground moving target using unmanned aerial vehicle," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 6968–6973.

[9] P. Theodorakopoulos and S. Lacroix, "A strategy for tracking a ground target with a uav," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1254–1259.

[10] K. B. Ariyur and K. O. Fregene, "Autonomous tracking of a ground vehicle by a uav," in *2008 American Control Conference*, June 2008, pp. 669–671.

[11] T. S. Richardson, C. G. Jones, A. Likhoded, E. Sparks, A. Jordan, I. Cowling, and S. Willcox, "Automated vision-based recovery of a rotary wing unmanned aerial vehicle onto a moving platform," *Journal of Field Robotics*, vol. 30, no. 5, pp. 667–684, 2013.

[12] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 221–238, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10846-010-9473-0

[13] C. Hui, C. Yousheng, L. Xiaokun, and W. W. Shing, "Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision," in *Control Conference (CCC), 2013 32nd Chinese*. IEEE, 2013, pp. 5895–5901.

[14] C. Teuliere, L. Eck, and E. Marchand, "Chasing a moving target from a flying uav," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4929–4934.

[15] J. E. Gomez-Balderas, G. Flores, L. R. García Carrillo, and R. Lozano, "Tracking a ground moving target with a quadrotor using switching control," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 65–78, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10846-012-9747-9

[16] G. R. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald, "A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera," *Remote Sensing*, vol. 4, no. 4, pp. 1090–1111, 2012.

[17] I. F. Mondragón, M. A. Olivares-Méndez, P. Campoy, C. Martínez, and L. Mejias, "Unmanned aerial vehicles uavs attitude, height, motion estimation and control using visual systems," *Autonomous Robots*, vol. 29, no. 1, pp. 17–34, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9183-2

[18] R. Strydom, S. Thurrowgood, and M. Srinivasan, "Visual odometry: autonomous uav navigation using optic flow and stereo," in *Proceedings of Australasian Conference on Robotics and Automation*, 2014.

[19] J. N. Martel, L. K. Müller, S. J. Carey, and P. 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*. IEEE, 2016, pp. 1430–1433.

[20] J. N. Martel, L. K. Mueller, S. J. Carey, and P. Dudek, "A real-time high dynamic range vision system with tone mapping for automotive applications," *CNNA 2016*, 2016.

[21] S. Park, J. Deyst, and J. How, "A new nonlinear guidance logic for trajectory tracking," in *AIAA guidance, navigation, and control conference and exhibit*, 2004, p. 4900.