

Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array

Yanan Liu^{1,2} | Laurie Bose² | Colin Greatwood¹ | Jianing Chen³ | Rui Fan⁴  | Thomas Richardson¹ | Stephen J. Carey³ | Piotr Dudek³ | Walterio Mayol-Cuevas^{2,5}

¹ Bristol Robotics Laboratory, The University of Bristol, Bristol, UK

² Visual Information Laboratory, The University of Bristol, Bristol, UK

³ Department of Electrical and Electronic Engineering, The University of Manchester, Manchester, UK

⁴ The Department of Computer Science and Engineering & The Department of Ophthalmology, The University of California, San Diego California, USA

⁵ Amazon, Seattle, Washington, USA

Correspondence

Yanan Liu, Bristol Robotics Laboratory, The University of Bristol, Bristol, BS16 1QY, UK.
Email: yanan.liu@bristol.ac.uk

Funding information

UK EPSRC EP/M019454/1, EP/M019284/1, EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems: FARSCOPE and China Scholarship Council (No. 201700260083)

Abstract

This paper presents an agile reactive navigation strategy for driving a non-holonomic ground vehicle around a pre-set course of gates in a cluttered environment using a low-cost processor array sensor. This enables machine vision tasks to be performed directly upon the sensor's image plane, rather than using a separate general-purpose computer. The authors demonstrate a small ground vehicle running through or avoiding multiple gates at high speed using minimal computational resources. To achieve this, target tracking algorithms are developed for the Pixel Processing Array and captured images are then processed directly on the vision sensor acquiring target information for controlling the ground vehicle. The algorithm can run at up to 2000 fps outdoors and 200 fps at indoor illumination levels. Conducting image processing at the sensor level avoids the bottleneck of image transfer encountered in conventional sensors. The real-time performance of on-board image processing and robustness is validated through experiments. Experimental results demonstrate the algorithm's ability to enable a ground vehicle to navigate at an average speed of 2.20 m/s for passing through multiple gates and 3.88 m/s for a 'slalom' task in an environment featuring significant visual clutter.

1 | INTRODUCTION

Vision-based mobile robot navigation technology plays an important role in the field of intelligent transportation systems and robotics, being increasingly used across many fields of industry, such as, driverless cars [1], assisted living, logistics [2] and domestic applications [3]. However, implementing a mobile robot to perform navigation tasks rapidly, robustly and energy efficiently can be challenging especially in cluttered environments [4]. A reactive navigation methodology is a solution to this problem that combines sensory information with robot control directly [5] through processing of sensor data [6].

A number of approaches for visual reactive ground navigation have been proposed in recent and past years. In [7], Penin

et al. utilised a Unmanned Aerial Vehicle (UAV) to perform visual-based reactive navigation, track targets and keep a fixed distance from the target at a speed of 1.00 m/s. Galluppi et al. [8] developed an autonomous mobile robotic platform using two dynamic vision sensors (DVS) and a SpiNNaker computing system. According to their work, the average navigation speed is less than 0.5 m/s. Many of the existing platforms require expensive computation to perform reactive navigation, which limits reaction time to changing environments and the mobile robot's top speed [9–11]. More recent vehicles geared for learning visual navigation, for example, DeepRacer¹ or Audi driving cup cars,² are slow in processing and are yet to demonstrate

¹ <https://aws.amazon.com/deepracer/>

² <https://www.audi-autonomous-driving-cup.com/>

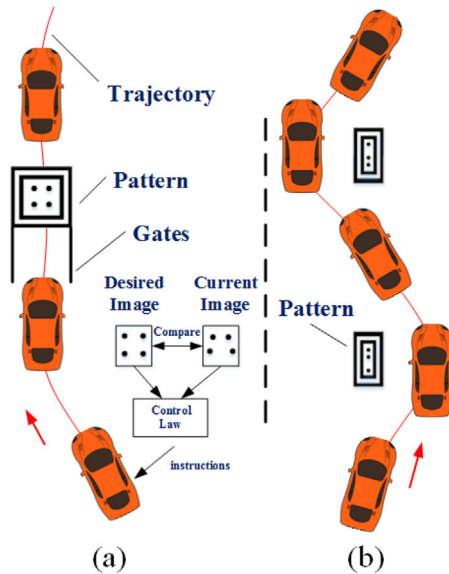


FIGURE 1 Examples of the proposed ground vehicle agile navigation. (a) Passing through the gates. The width of the car body and the gate are 18 cm and 36 cm, respectively; (b) the 'slalom' navigation. Experimental video can be seen at: <https://youtu.be/e85q-yoBuSk>

agile behaviours. Dedicated sensors for car line following do allow faster operation but are tuned for the task, and thus less generic.

In agile visual reactive navigation, a vision sensor with a high frame rate is necessary to enable the mobile robot to react to the environment quickly. Various types of visual sensors, such as CCD/CMOS cameras [12] and RGB-D sensors [13] are widely used for map building, robot localisation, navigation and other vision guidance applications [12, 14]. However, these sensors transmit entire images to a computer for processing, a process that requires relatively significant time and power, decreasing the response time of the robot [15]. The key contribution of this paper compared to our previous work [16] includes improving iterative flooding operations to enhance its robustness when extracting disks, designing a filtering algorithm based on the neighbourhood pixel values to de-noise the image, exploiting the PID control to complete the navigation task and integrating the improvements mentioned above to directly link the visual information from the image plane to robot control instructions enabling agile reactive navigation. Compared to a traditional visual sensor, each pixel in the pixel processor has storage and processing abilities, which support fast and lower power parallel computation directly on the visual sensor. The pixel processor array (PPA) is capable of performing general-purpose vision algorithms at several thousand frames per second due to its parallel computing features [17, 18]. This paper presents a novel agile reactive navigation strategy using the PPA and a mobile robot with a low-priced single-board computer and a simple control structure. In our applications, a car-like mobile robot self-navigates, running through multiple gates at an average speed of 2.20 m/s according to the guidance provided by visual information extracted from the pre-designed patterns pasted on the gates (see Figure 1(a)). The control instructions

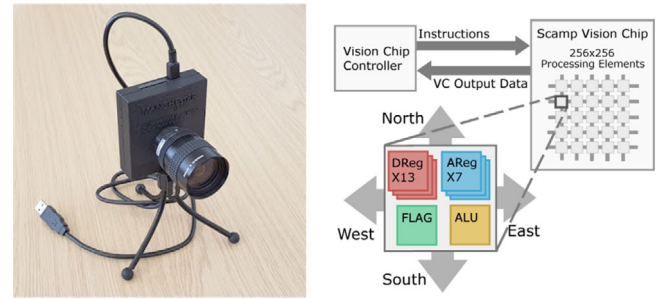


FIGURE 2 SCAMP-5d (left) and the processing elements in pixel processor arrays (right) [18]

for the mobile robot are generated by comparing the desired image and the current image. The mobile robot also achieves a highest speed of 3.88 m/s during a 'slalom' process (Figure 1(b)). In particular, the mobile robot is directly instructed by the coordinates of feature points on the patterns themselves, hence there is no need to perform transformation calculations among the camera coordinate system, image coordinate system and ground vehicle coordinate system.

The remaining sections of this paper are organised as follows: Section 2 describes the proposed vision system including the introduction of the SCAMP-5 vision system and image processing algorithms. In Section 3, we present the robot control system which consists of hardware and software structures. Section 4 illustrates the agile reactive navigation method. Experimental results are illustrated and the proposed system performance is evaluated in Section 5. Finally, Section 6 summarises the paper and provides recommendations for future work.

2 | VISION SYSTEM DESCRIPTION

This section introduces the SCAMP-5 vision system, compares it with the traditional visual sensor and illustrates the image processing algorithms based on the SCAMP-5.

2.1 | SCAMP-5 vision system

SCAMP-5d (see Figure 2) is the latest version of the programmable, general-purpose, power-efficient and high-performance pixel processor vision chip family developed by the University of Manchester [17]. The PPA of the SCAMP-5 vision chip integrates a massively parallel Single Instruction Multiple Data (SIMD) processor array into the image sensor pixels. It is the combination of the image sensor's pixels and processor granting it the ability of both a traditional visual sensor and a programmable processor. The vision sensor of the SCAMP-5 system consists of a grid of 65,536 (256×256) processing elements (PEs). Each PE in the image array consists of 7 analogue registers, 13 digital registers and 1 digital 'FLAG' register, which enables this vision system to store 7 grey-value images, 13 binary images and to support conditional execution among frames. Moreover, arithmetic operations, logic

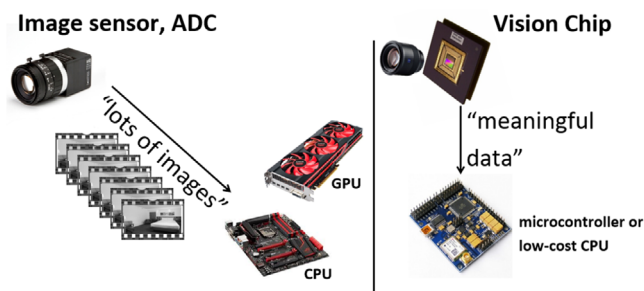


FIGURE 3 Comparison of the workflow between the traditional vision system and the SCAMP-5 vision system

operations and neighbourhood operations on the image plane are available.

Unlike the conventional image sensor which only contains the light sensitive pixel circuits, SCAMP-5 features the integration of image sensors and processors on a single silicon die. By doing so, the image processing happens directly on the image plane. Thus, the image processing results rather than the image itself can be transferred to other controllers (see Figure 3). For most tasks, the data/calculations resulting from this image processing are far smaller than the raw image data, which greatly reduces the transfer time between SCAMP-5 and the controller compared to a standard camera. The vision chip controller is based on the ARM-Cortex-M, it supports programming and general-purpose computation tasks. The image processing algorithms can be developed using a standard C++ compiler on the vision chip controller. After compiling the program, the image processing kernels for acceleration on the processor array are sent to each PE to perform processing tasks. These PEs are able to perform basic computation tasks in parallel, hence enabling fast-speed on-sensor image processing. Finally, the image processing results are sent back to the chip controller and then transferred to next-level hardware, such as the single board computer through GPIO or USB interface. Moreover, the SCAMP-5 system enables real-time image processing on the focal plane while consuming minimal power (between 0.1 W and 2 W depending on the algorithm), which is suitable for embedded systems in the field of robotics [16, 18–21].

2.2 | Gate pattern recognition algorithm

This subsection describes the SCAMP-5 algorithm used to detect the pre-designed patterns. Patterns are a common means to give instructions to human drivers or even to autonomous cars such as in line following. The challenges include tolerance to noise, clutter and fast enough processing. The patterns we use are primarily aimed at demonstrating agile visuo-control behaviours.

For the gate's task, each pattern contains four black disks surrounded by two black concentric squares. The disk coordinates in the image plane are utilised to adjust the rover's position and orientation, guiding it to go through multiple gates at high

speed. Figure 4 and Figure 5 show the image processing procedure conducted on SCAMP-5. High frame rate real-time image processing is typically challenging to conduct with a standard camera and computer setup due to the limited rate of image capture and delay in data transfer that such devices have. In this work, the target extraction algorithm is designed to exploit SCAMP-5's fast image flooding ability to minimise computation time per image and improve the image processing robustness. Iterative flooding operations are used to extract centres of four disks out of the cluttered background [16]. This method fills regions with value '1's except these areas closed by value '0's. Hence those areas that are out of the closed regions would be filled with '0's and with this method, the background of the pattern is easily eliminated within several iterations. As shown in Figure 4, after the first flooding iteration, the background is eliminated. Therefore, the time cost for this kind of operation is much less than that of the traditional camera, because this flooding operation is carried out in all PEs in parallel and asynchronously rather than pixel by pixel for traditional image processing.

The extraction of four points in the pattern relies on the presence of two black concentric boundaries. However, issues, such as target occlusion, bright light reflections or simply the target being distant from the camera, may break or merge together those two boundaries, causing failure of the dot extraction process. For example, in Figure 5, the pattern in the source image is partly outside the view field and the boundaries broken on the right-hand side for demonstration. In this case, the direct flooding method is not useful since four disks are not enclosed by '0's. To improve the robustness of the pattern extraction, the prior knowledge of where the dots were located in the previous frame is used whenever the two black boundaries are not present. This is illustrated in Figure 5, when there is no object remaining after two flooding and inversion operations. Firstly, the inversion of the current binary image is performed. Then, a point from the last detected disk centre is loaded into the current frame. Since the frame rate of the SCAMP-5 is set to 200 fps or more, the shift between two consecutive frames is small, and thus, we assume that the loaded point falls into the current corresponding disk. Then, flooding is conducted outwards from this point using the current inverted image as a mask. If the point from which flooding was performed existed within the disk, the resulting image contains only that disk which can then be easily extracted. After getting rid of the first extracted disk from the inverted image, the location of the remainder of the disks can be obtained using the loading point and flooding method iteratively. The detailed algorithm description can be seen in Algorithm 1.

Noise in an image is caused by various factors, such as overexposure, cluttered environment and quantisation. In this scenario, the light spots caused by overexposure or reflection are the main sources of noise. To eliminate the noise, an image filtering method based on the neighbour communication is conducted upon SCAMP-5. Since each PE can communicate with its four neighbours (north, south, west and east), a given pixel can access its neighbours' data directly. Four new pictures are obtained after moving the source image into four directions

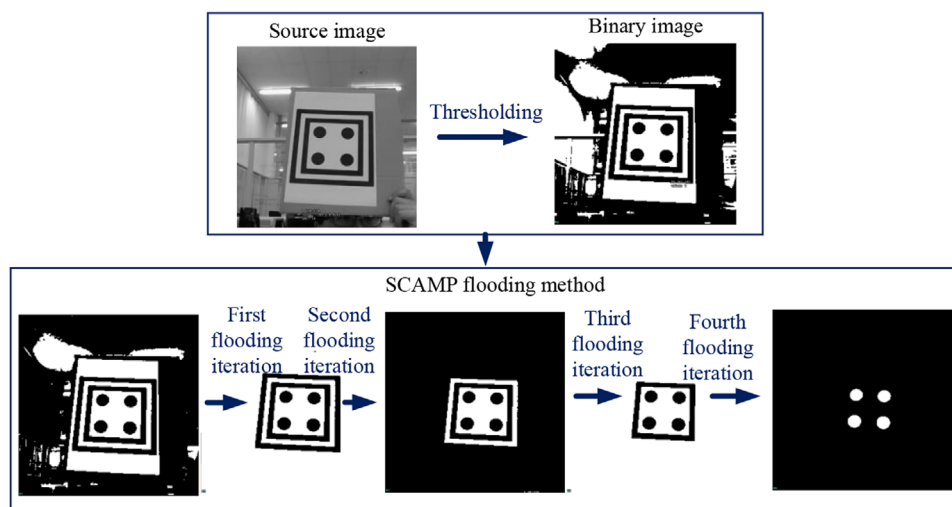


FIGURE 4 Image processing procedure. The four extracted dots in the final result form a quadrangle

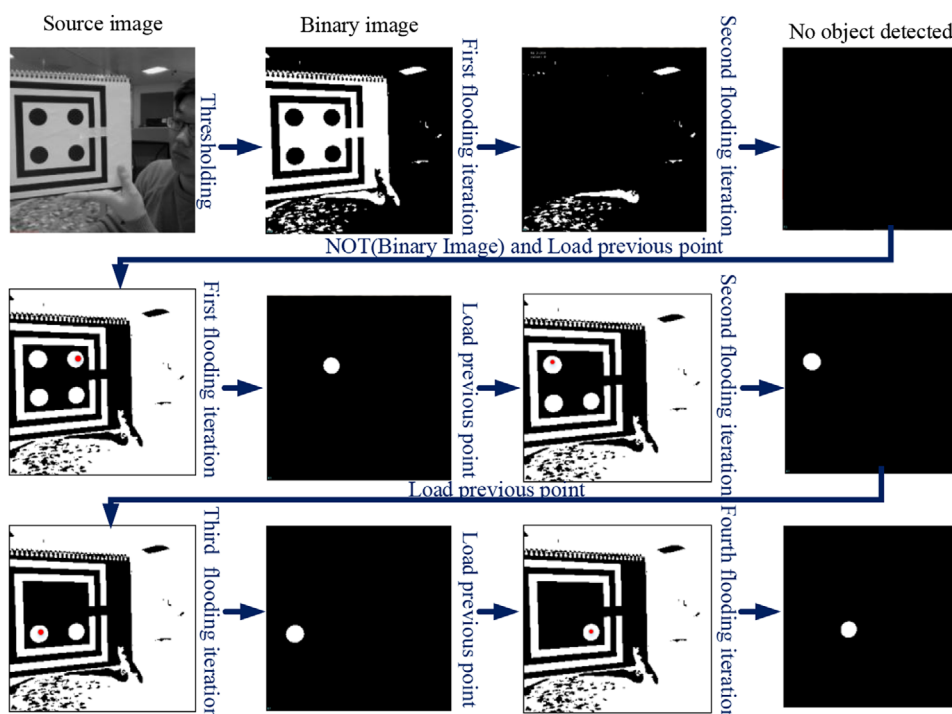


FIGURE 5 Image processing procedure in which the squares are broken

with a pixel distance P_{step} . Then, 'AND' operator is used to add these four images together on the image plane directly. In Figure 6, noisy area whose radius is smaller than shrink radius will be eliminated. The pseudo codes are given in Algorithm 2.

We now discuss the way of getting each dot's centroid in the image plane. First an inbuilt function *scan_event* of the SCAMP-5 is used to find the location of a white pixel in the image. This white pixel must be located within one of the disks we wish to extract. We then perform a flooding operation originat-

ing from this extracted point, using current image as a mask, resulting in an image consisting of the entire disk the point was within, as shown in the second image of Figure 7. The bounding box of all white pixels in the image (which in this case are those of the disk) is then extracted using another built-in SCAMP-5 function *scan_boundingbox*. After that, the disk is removed from the original binary image using a NOT operation. With this method, one white disk can be extracted. This process is similar to that used in Figure 5 where squares are not intact. The *scan_boundingbox* function outputs the centre position

Algorithm 1 Disk extraction based on the SCAMP-5**INPUT:***Threshold* // used to get binary image*Iterations* // iteration times for flooding and inversion operation*Disc_num* // pre-set number of disks**OUTPUT:***Coordinates* // coordinates of four disks' centres**while true do***R5* = *Scamp5_get_image()**R6* = *Scamp5_threshold(R5, Threshold)***for** *n* = 0 to *Iterations* **do***R7* = *Scamp5_flood(R6)**R6* = *AND(R6, NOT(R7))**R6* = *NOT(R6)***end for***Image filtering* (see Algorithm 2)*Num* = *Object_detect(R6)***if** *Num* == *Disc_num* **then***Coordinates* = *Scamp5_scan_boundingbox(R6)***else***R7* = *Scamp5_load_point(previous_point)**R7* = *Scamp5_flood(R6)**R8* = *R6 XOR R7**R6* = *R8**Coordinates* = *Scamp5_scan_boundingbox(R7)***end if***previous_point* = *Coordinates***end while****return** *Coordinates***Algorithm 2** Image filtering based on the SCAMP-5**INPUT:** P_{step} **OUTPUT:** *R10***while true do***R5* = *Scamp5_get_image()**R6* = *Move R5 to the north with P_{step}* *R7* = *Move R5 to the south with P_{step}* *R8* = *Move R5 to the west with P_{step}* *R9* = *Move R5 to the east with P_{step}* *R10* = $R6 \cap R7 \cap R8 \cap R9$ **end while****return** *R10***TABLE 1** list of hardware adopted for the navigation system

Hardware name	Hardware configuration
Subaru RC Car	SKU H94123-12344
SCAMP-5 Vision System	SCAMP-5d (gray-level image, 256×256 array)
Raspberry Pi	Version 3 Model B+, CPU: 1.4GHz
Subaru power unit	ANSMANN, RC-Racing Pack
Raspberry Pi power unit	HUAWEI Colorphon 5
Remote control	TARANIS

can be easily performed at more than 2000 fps with enough light illumination.

3 | ARCHITECTURE OF ROBOT SYSTEM

A Subaru rover chassis, shown in Figure 8, was utilised for the agile navigation experiment. The rover was controlled by both the visual data from the SCAMP-5 and the remote control shown in Figure 9. Specifically, the visual data controls the steering while the speed of the car is set to a fixed value using the remote control. As shown in Figure 9, images are captured and processed on the vision chip and the processing results are sent back to the chip controller. Finally, these coordinates are transferred to the Raspberry Pi, which handles read-out data and performs mobile robot navigation tasks. The detailed hardware and its hardware configurations can be seen in Table 1.

4 | AGILE REACTIVE NAVIGATION

4.1 | Object tracking

The SCAMP-5 extracts the four disks located within any visible targets which are then utilised to steer the mobile robot through the gate with the observed target. Controls are generated by comparing the four disks extracted from a pattern in view

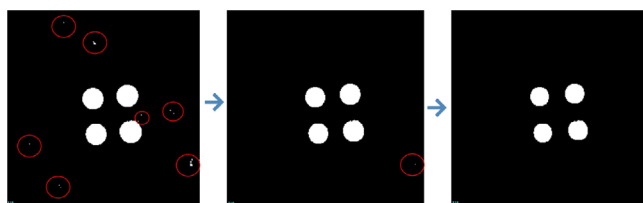


FIGURE 6 Image de-noising, the shrink radius is 2 pixels. Noisy pixels are circled in red. All noisy pixels are eliminated in the third image

of the white region. After performing this type of operation for four iterations, all centre positions of the disks can be extracted separately.

The image processing time of all algorithms mentioned above is recorded by the SCAMP-5 application that receives the image processing information from the SCAMP-5 vision system through a USB cable. In this scenario, the frame rate is set to 200 fps given the illumination conditions in the indoor arena. It is noteworthy that the proposed image processing algorithm

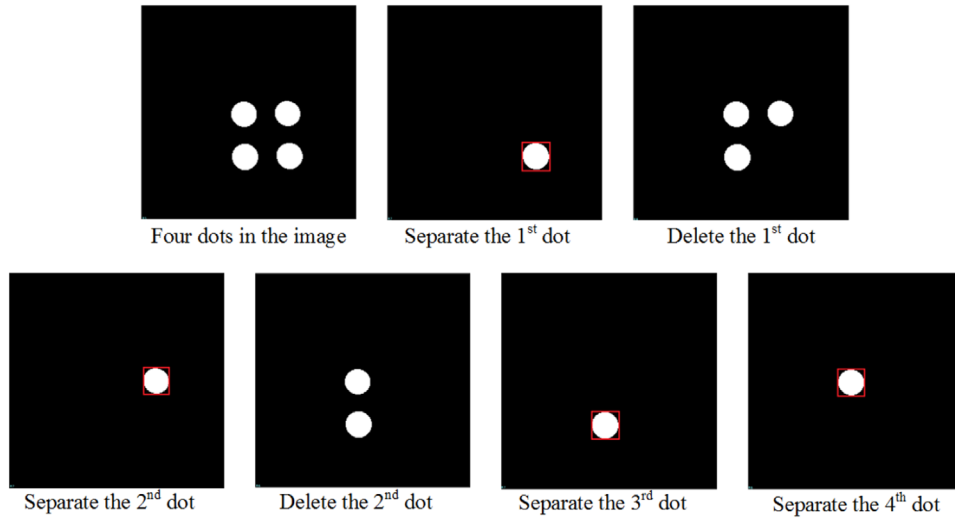


FIGURE 7 An example of getting coordinate of each dot's centre

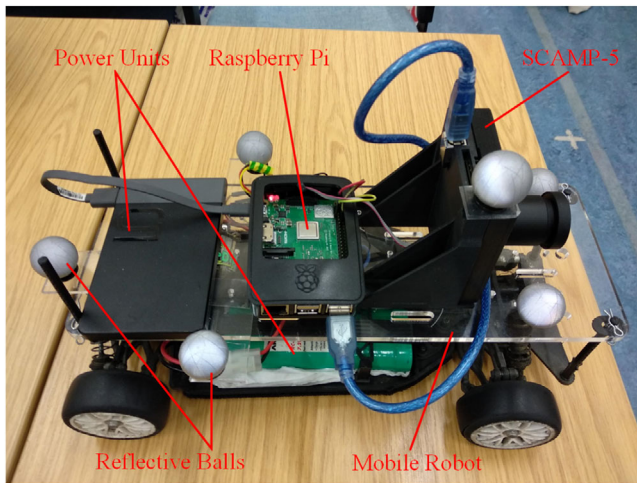


FIGURE 8 Subaru RC car (1:10 scale model) and hardware used in experiments

against four reference disks for a pattern at a known distance and angle. However, when carrying out the disk extraction, the order of these four dots is unknown because the relative position between the camera and the gate can be random. Hence, a method is developed to make these eight points correspond to each other before making a comparison.

As shown in Figure 10, the centre of these four dots P_{cc} can be obtained by $P_{cc} = \sum P_{ci}/4$. By comparing the difference between P_{cc} and P_{ci} , these four dots can be allocated into the corresponding quadrant. The current picture is divided into four quadrants each of which will contain a specific disk which can then be compared to the corresponding disk in a reference pattern.

$$Sign = P_{ciy} - P_{ccy}, \quad i \in \{1, 2, 3, 4\}. \quad (1)$$

where $Sign$ represents the sign symbol of the difference between P_{ccy} and P_{ciy} along y axis.

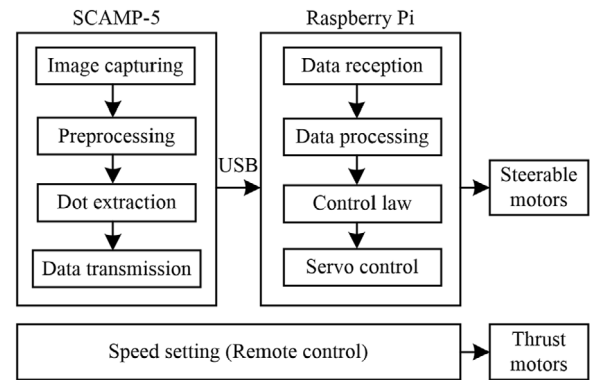


FIGURE 9 Control architecture of the navigation system

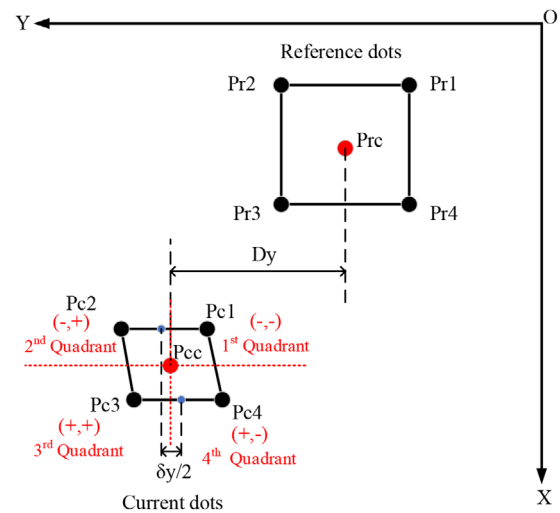


FIGURE 10 Quadrant allocation for four current dots and its difference from the reference dots. These dots represent the centres of extracted disks and they form a rectangle in reference image and a quadrangle in current image, respectively

4.2 | Mobile robot reactive navigation

This subsection compares the difference in both position and shape between the reference rectangle and the currently processed quadrangle to generate instructions for the mobile robot to run through gates vertically. The reference image is defined with an image that was pre-set by putting the ground vehicle in front of the gate at a distance which is the minimum distance that is in sight of the sensor. The reference rectangle is compared to the currently processed quadrangle in SCAMP; from this comparison, controls are generated for the RC rover which will bring the observed rectangle closer to the reference, guiding it towards the gate. As the pose of the rover gets closer to that at which the reference image was taken, the similarity between the quadrangles correspondingly increases. This control guides the rover through the gate without collision enabling the rover to see the next gate.

During reactive navigation, the distance D_y between P_{cc} and P_{rc} and the deformation δy along y axis (see Figure 10) act as the input for the PID control.

$$D_y = P_{cc} - P_{rc}, \quad (2)$$

where D_y is utilised to adjust the angle of the front wheels towards the target. δy describes the deformation of the pattern along the y axis caused by the relative position between the pattern and the rover. In this scenario, δy is the difference between top side centre and bottom side centre in the quadrangle.

$$\delta y = P_{c3y} + P_{c4y} - (P_{c1y} + P_{c2y}), \quad (3)$$

where δy is used as a compensation to slightly change the wheel angle and attempt to take the rover straight through the gate rather than going through at an angle. The adopted control method is as follows:

$$Output = Output_1 + Output_2, \quad (4)$$

$$Output_i = K_P e(k)_i + K_I \sum_{j=0}^k e(j)_i + K_D [e(k)_i - e(k-1)_i], \quad (5)$$

$$e(k)_1 = D_y, \quad (6)$$

$$e(k)_2 = \delta y. \quad (7)$$

where, $i \in \{1, 2\}$, K_P , K_I and K_D are coefficients adjusted experimentally here using Ziegler-Nichols [22]. $e(k)_i$ is the error between D_y , δy and 0. As we can see from the equation, the visual information is slightly processed before generating control instructions for the mobile robot. The aim of this PID control is to minimise both D_y and δy to 0. During this process, the ground vehicle is moving towards the gate while adjusting its pose to enter the gate head on.

5 | EXPERIMENTAL RESULTS

Experiments are carried out to test how agile the ground vehicle would track the target (the gate) and run through the gate in heavily cluttered environment. The mobile robot is set at a constant speed using the remote control. In Figure 11, there are eight gates placed in the robot arena taking into consideration space limitations and the vehicle's turning radius. Beside each gate, there are some other gates with a random or similar pattern to the real pattern acting as a disturbance to show the robustness of this navigation system to the cluttered environment.

The views in front of the rover are recorded by a GoPro camera. As we can see from Figure 12, the images captured by the GoPro are blurred because of the high speed and vibration of the chassis during navigation. However, the SCAMP is able to get clear images and output-effective visual data. When the mobile robot is running through these gates, its trajectory is recorded by the VICON capture system. Figure 13 shows the rover's paths and the layout of eight gates and the disturbance. As we can see, the rover runs through these gates, which indicates the effectiveness of the proposed control method and the robustness of the image processing. Figure 14 shows the changes of the front wheel angle when the rover is approaching a gate and the steering angle is asymptotically close to 0. The extracted dots are recorded during the navigation and plotted. As shown in Figure 15, the relative position of these four dots is changing when the rover is running towards a gate. The difference between the reference image and the current image is used for controlling the motion of the mobile robot. Consequently, in the image plane, their difference should be getting increasingly smaller during the navigation process. The expected phenomenon can be seen in Figure 15, where the current pattern shape is becoming increasingly closer to the reference pattern, in terms of both position and shape.

Although all the patterns are designed identically, their positions on the gate are slightly different. In addition, the view of the SCAMP-5 is possibly not horizontal because of its suspension system during high-speed motion. As a result, the final image when the mobile robot is about to pass through the gate could be slightly different from the reference image.

The agility of this reactive navigation can be seen from Figure 16. The average navigation speed is calculated as 2.20 m/s and the maximum speed reaches 2.50 m/s according to the

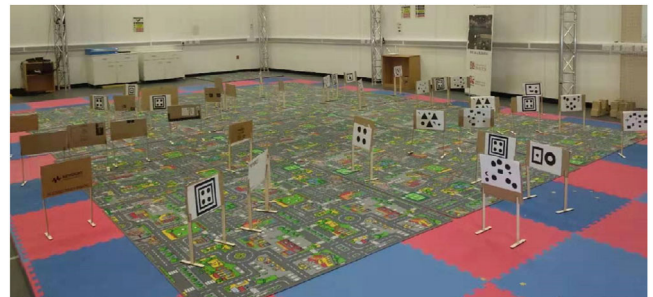


FIGURE 11 Gates and clutter layout in the arena

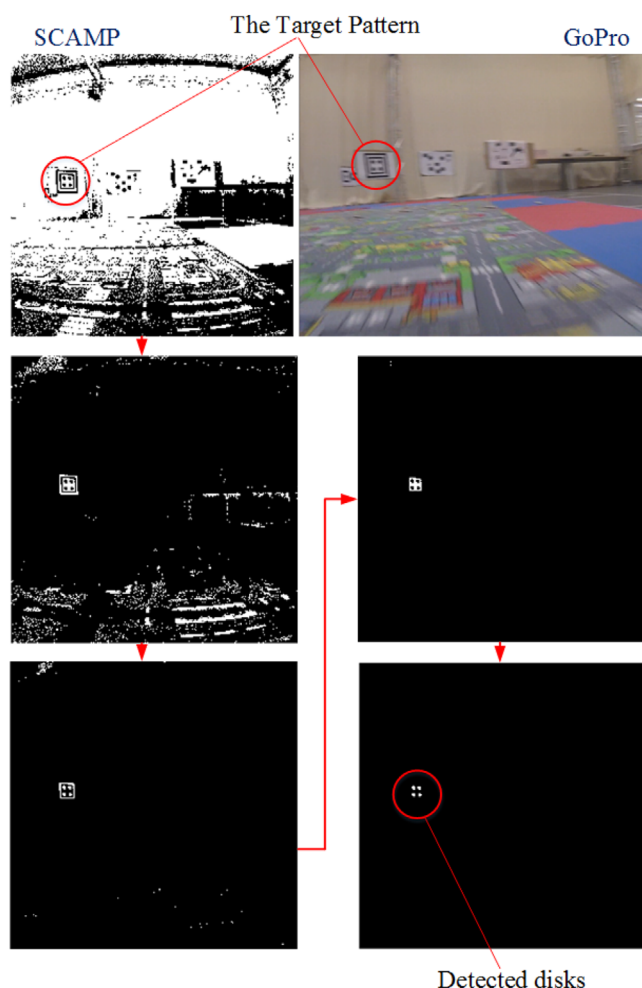


FIGURE 12 Image captured by the SCAMP-5 and GoPro during high-speed navigation. The SCAMP-5 detects four disks during high-speed navigation in a cluttered environment

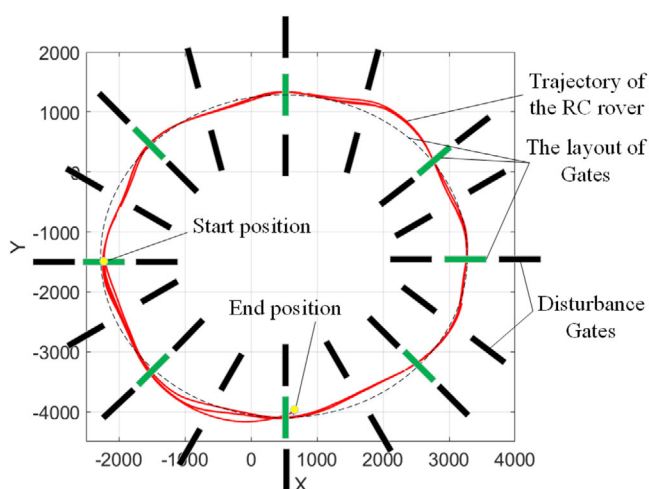


FIGURE 13 Trajectory in the arena

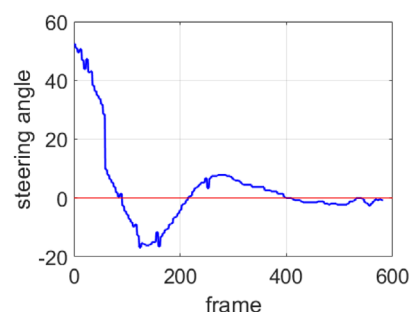


FIGURE 14 The angle change when the rover is approaching a gate

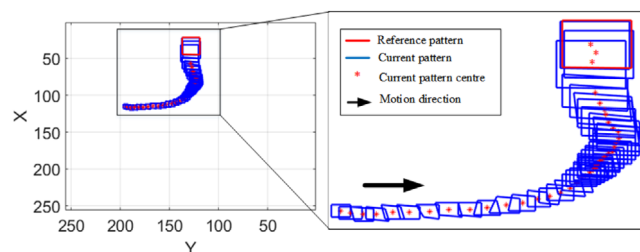


FIGURE 15 Trajectory in the image plane

VICON tracking data, which shows the rover response to the environment quickly with the guidance of the SCAMP-5.

The time cost for each procedure of the whole control system, including that in SCAMP-5 and Raspberry Pi, is measured and listed in Table 2. Our SCAMP-5 pipeline is able to perform at over 2000 fps according to Table 2. This accounts for the image and PID processing but does not include the image exposure time which will depend on environmental illumination conditions.

To further explore the agility of the robot system based on the SCAMP-5, we also proposed a 'slalom' application with some pre-designed visual patterns (see Figure 17) encoding turning and angle information which are extracted using the SCAMP-5 vision system. By placing these patterns in specific positions, the rover weaves left and right around the patterns according to the information encoded on each, in a similar manner to slalom skiing. The image processing method is similar to the one used for four disks extraction.

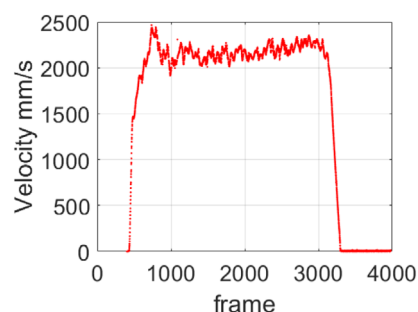


FIGURE 16 Velocity curve for the whole navigation process recorded by VICON

TABLE 2 Time cost for different sections of the system

Processing steps	Time cost (μ s)
Image capturing and thresholding	52
Flooding method to extract dots (including normal and broken squares)	63
Image de-noising	4
Getting coordinates of four dots' centre	328
Coordinates transmitted to Raspberry Pi	≈ 24
PID control in Raspberry Pi	≈ 1
Total	≈ 472

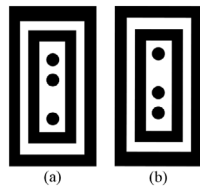
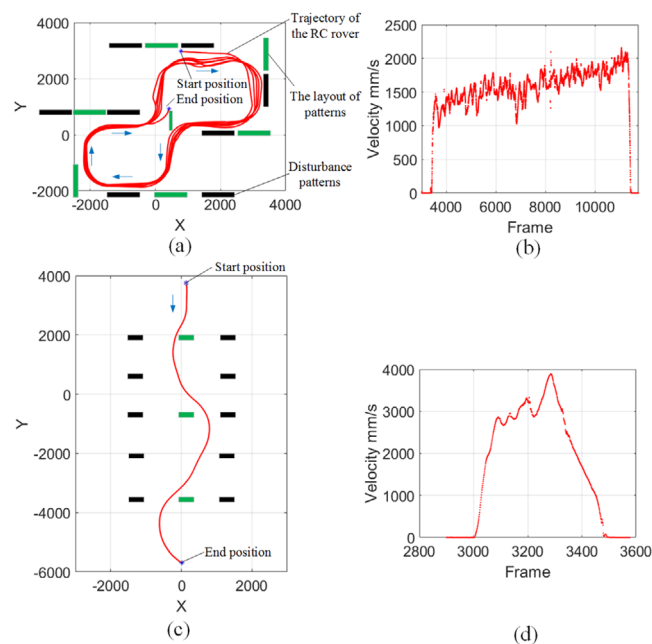
**FIGURE 17** 'Slalom' pattern. (a) Turn right for certain degree; (b) turn left for certain degree

Figure 18 shows the trajectory and velocity of the rover during navigation. In Figure 18(a), there are eight patterns placed in the arena, six of them guide the rover to turn right and two to turn left. The robot turns then continues towards the next closest pattern it encounters. Figure 18(b) shows the average speed of 1.70 m/s for this scenario. By placing these gates in a line, a high-speed slalom can be achieved. In Figure 18(c), the distance

**FIGURE 18** Trajectory and velocity curve for the 'slalom' process. (a) trajectory of number '8'; (b) velocity curve for number '8'; (c) trajectory of a fast 'slalom'; (d) velocity curve for the fast 'slalom'

between gates are set 2.40 m apart, and the rover turns right or left at a distance 0.80 m from the pattern. Its velocity curve (see Figure 18(d)) shows the highest speed reaches 3.88 m/s which translates to 139 km/h for the real-sized car.

6 | CONCLUSION AND FUTURE WORK

This paper focuses on real-time visual information processing using a PPA and agile robot navigation with information derived from the visual sensor. The SCAMP-5 is capable of processing high-frame rate visual information as the agile navigation requires [10]. We implemented the high-speed reactive navigation and enable a Subaru mobile robot to run through multiple gates at an average speed of 2.20 m/s and a highest speed of 3.88 m/s for a 'slalom'. Moreover, the developed vision algorithms based on the SCAMP-5 can reach up to 2000 fps with enough illumination. We proposed a direct link between the image information and the angle of the steerable servo motor in the mobile robot. To the best of our knowledge, the speed of the agile ground vehicle achieved is the highest among the systems using non-conventional visual-processor pipelines.

The multi-gate test shows high-speed navigation for an agile reactive navigation system and shows the effectiveness of the control method under motion disturbance and vibrations. However, the navigation speed and performance are still limited by many factors including low control frequency of the servo motor or loss of targets especially when the fast-moving mobile robot is far from the target. In our future work, we aim to improve the system adaptability in both indoor and outdoor environments. We believe that the development of novel vision-inspired processing hardware architectures, is key to the progress of agile and responsive robotics that are required to operate in a complex and uncertain world. This work explores the case of agile reactive navigation and our future emphasis will include extending the capabilities beyond known target detection and into classification and place recognition while navigating.

DATA ACCESS STATEMENT AND ACKNOWLEDGEMENTS

This work was supported by UK EPSRC EP/M019454/1, EP/M019284/1, EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems: FARSCOPE and China Scholarship Council (No. 201700260083). The nature of the task and PPA means that the SCAMP-5 images in this work are not recorded.

ORCID

Rui Fan  <https://orcid.org/0000-0003-2593-6596>

REFERENCES

- Ozgunalp, U., et al.: Multiple lane detection algorithm based on novel dense vanishing point estimation. *IEEE Trans. Intell. Transp. Syst.* 18(3), 621–632 (2017)

2. Ni, H., et al.: Sorting system algorithms based on machine vision for delta robot. *Robot* 38(1), 49–55 (2016)
3. Sotelo, M.A., Rodríguez, F.J., Magdalena, L.: Virtuous: vision-based road transportation for unmanned operation on urban-like scenarios. *IEEE Trans. Intell. Transp. Syst.* 5, 69–83 (2004)
4. Liu, Y., et al.: Mobile robot localisation and navigation using lego nxt and ultrasonic sensor. In: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1088–1093. IEEE, (2018)
5. Mujahed, M., Fischer, D., Mertsching, B.: Safe gap based (sg) reactive navigation for mobile robots. In 2013 European Conference on Mobile Robots (ECMR), pp. 325–330. IEEE, (2013)
6. Tobaruela, J.A., Rodríguez, A.O.: Reactive navigation in extremely dense and highly intricate environments. *PloS One* 12(12), e0189008 (2017)
7. Penin, B., Giordano, P.R., Chaumette, F.: Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions. *IEEE Rob. Autom. Lett.* 3(4), 3725–3732 (2018)
8. Galluppi, F., et al.: Event-based neural computing on an autonomous mobile platform. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 2862–2867. IEEE, (2014)
9. Lopez, B.T., How, J.P.: Aggressive 3-d collision avoidance for high-speed navigation. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5759–5765. IEEE, (2017)
10. Hwu, T., et al.: A self-driving robot using deep convolutional neural networks on neuromorphic hardware. In 2017 International Joint Conference on Neural Networks (IJCNN), pp. 635–641. IEEE, (2017)
11. Pan, Y., et al.: Agile autonomous driving using end-to-end deep imitation learning. In *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, RSS, (2018)
12. Liu, Y., et al.: Design of a pc-based open industrial robot control system integrated with real-time machine vision. In 2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA), pp. 1–6. IEEE, (2018)
13. Kong, Y., Fu, Y.: Discriminative relational representation learning for RGB-D action recognition. *IEEE Trans. Image Process.* 25, 2856–2865 (2016)
14. Fan, R., Dahmoun, N.: Real-time stereo vision-based lane detection system. *Meas. Sci. Technol.* 29(7), 074005 (2018)
15. Greatwood, C., et al.: Perspective correcting visual odometry for agile mavs using a pixel processor array. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 987–994. IEEE, (2018)
16. Greatwood, C., et al.: Tracking control of a UAV with a parallel visual processor. In IEEE International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, pp. 4248–4254. IEEE, (2017)
17. Carey, S.J., et al.: A 100,000 fps Vision Sensor with Embedded 535GOPS / W 256x256 SIMD Processor Array. Symposium on VLSI Circuits, Kyoto, Japan, pp. 182–183. IEEE, (2013)
18. Bose, L., et al.: Visual odometry for pixel processor arrays. *Proceedings of the IEEE International Conference on Computer Vision* 2017, 4614–4622 (2017)
19. Martel, J.N., et al.: A real-time high dynamic range vision system with tone mapping for automotive applications. In *Proceedings of CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–2. VDE, (2016)
20. Chen, J., Carey, S.J., Dudek, P.: Feature Extraction using a Portable Vision System. In IEEE International Conference on Intelligent Robots and System (IROS) Workshop, Vancouver. IEEE, (2017)
21. Chen, J., et al.: Proximity estimation using vision features computed on sensor. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2689–2695. IEEE, (2020)
22. Ziegler, J.G., et al.: Optimum settings for automatic controllers. *Trans. ASME* 64(11), 759–765 (1942)

How to cite this article: Liu Y, Bose L, Greatwood C, Chen J, Fan R, Richardson T, Carey SJ, Dudek P, Mayol-Cuevas W. Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array. *IET Image Process.* 2021;15:1883–1892. <https://doi.org/10.1049/ipr2.12158>