Proximity Estimation Using Vision Features Computed On Sensor

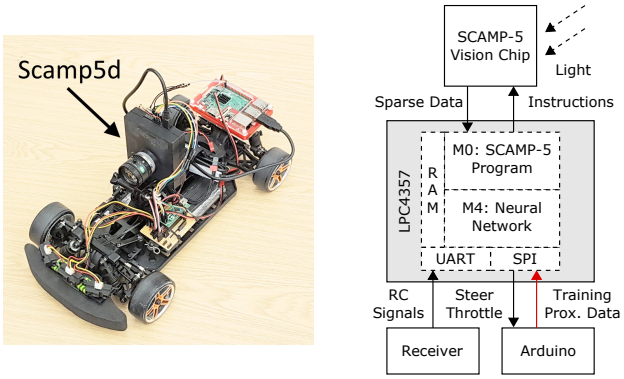
Jianing Chen[1], Yanan Liu[2], Stephen J. Carey[1] and Piotr Dudek[1]

***Abstract*— This paper presents a monocular vision based proximity estimation system using abstract features, such as corner points, blobs and edges, as inputs to a neural network. An experimental vehicle was built using a vision system integrating the SCAMP-5 vision chip, a micro-controller, and an RC model car. The vision chip includes image sensor with embedded 256×256 processor SIMD array. The pixel processor array chip was programmed to capture images and run the feature algorithms directly on the focal plane, and then digest them so that only sparse feature description data were read-out in the form of 40 values. By logging the vision output and the output from three infrared proximity sensors, training data were obtained to train three fully connected layer-recurrent neural networks with fewer than 700 parameters each. The trained neural network was able to estimate the proximity to the level of accuracy sufficient for a reactive collision avoidance behaviour to be achieved. The latency of the control system, from image capture to neural network output, was under 4 ms, enabling the vehicles to avoid obstacles while moving at 0.64 m/s to 1.8 m/s in the experiment.**

I. Introduction

For the problem of proximity estimation, the use of vision cues has several advantages over dedicated distance sensors as long as good visibility is assumed: as a passive mechanism, vision based solutions will not interfere with each other in an environment with numerous agents; the surface properties of the obstacles are less likely to hamper the performance of a vision based solution; it offers better scalability and potentially greater operating range of distances; the requirements for additional hardware is reduced, as on-board cameras can be used for other tasks. Stereo-vision based depth perception is reliable, but the dependency on significant computing power and dual full resolution video streams could limit its application in systems that have to be low power and compact size. In this work, we are attempting to achieve proximity estimation through monocular vision, using methods suitable for embedded vision systems. In particular, we are interested in a system which efficiently extracts certain visual features, containing depth cues, and combines them using a trained neural network.

Convolutional neural network (CNN) based methods can provide end-to-end solutions when the computational power allowed on-board the vehicle is sufficient [1]. However, with limited on-board processing power, we consider a system where the low level feature extractors are engineered, providing their efficient implementation using the SCAMP-5 vision chip, and only employ a neural network for final layers of computation. In particular, three types of features that contain monocular depth cues were studied: (a) spatial frequency filtering, (b) motion parallax, and, (c) corner features. The receptive field [2] method was used to digest these feature images into a vector of scalars to be processed by a fully connected neural network, whose output was the proximity distance to any nearby obstacles. In this way, the efficiency of near-sensor parallel processing hardware can be exploited, while still retaining the effectiveness of the neural network based approach.



(a) (b)

Fig. 1. Hardware overview. (a) The 1/10 scale RC model car used for experiments. The 'Scamp5d' vision system (black box) integrates an ARM MCU and the SCAMP-5 vision chip that contains a pixel-parallel SIMD processor array; the red box is a Raspberry Pi 3, which was not a part of the control loop but used to relay data from the vision system over Wifi. (b) The hardware block diagram (see text for description)

II. Related Work

Proximity estimation could be viewed as a subset of the depth mapping problem. Monocular depth estimation can be achieved by a camera on a moving vehicle [3], [4]. However, such methods usually require powerful computing hardware. Moreover, the latency in these systems can be too high for an agile vehicle. For avoiding obstacles, which only requires close range depth information, it is possible to not compute an explicit depth map but use optical flow as input [5], [6], [7]. Control of a ground vehicle through end-to-end learning based on human driving behaviour has been applied to obstacle avoidance problems [8], and demonstrated in simulations of autonomous driving on road [9]. These methodologies use high-resolution RGB/gray-scale video as input, and required relatively powerful computing hardware. This limitation also exists for stereo-vision based approaches [10]. Moreover, relying on stereo vision parallax may have limited scalability

This work has been supported by EPSRC Grant EP/M019284/1.

[1]Department of Electrical & Electronic Engineering, The University of Manchester, Manchester M13 9PL, United Kingdom.

[2]Bristol Robotics Laboratory, Faculty of Engineering, University of Bristol, Bristol, BS16 1QY, United Kingdom.

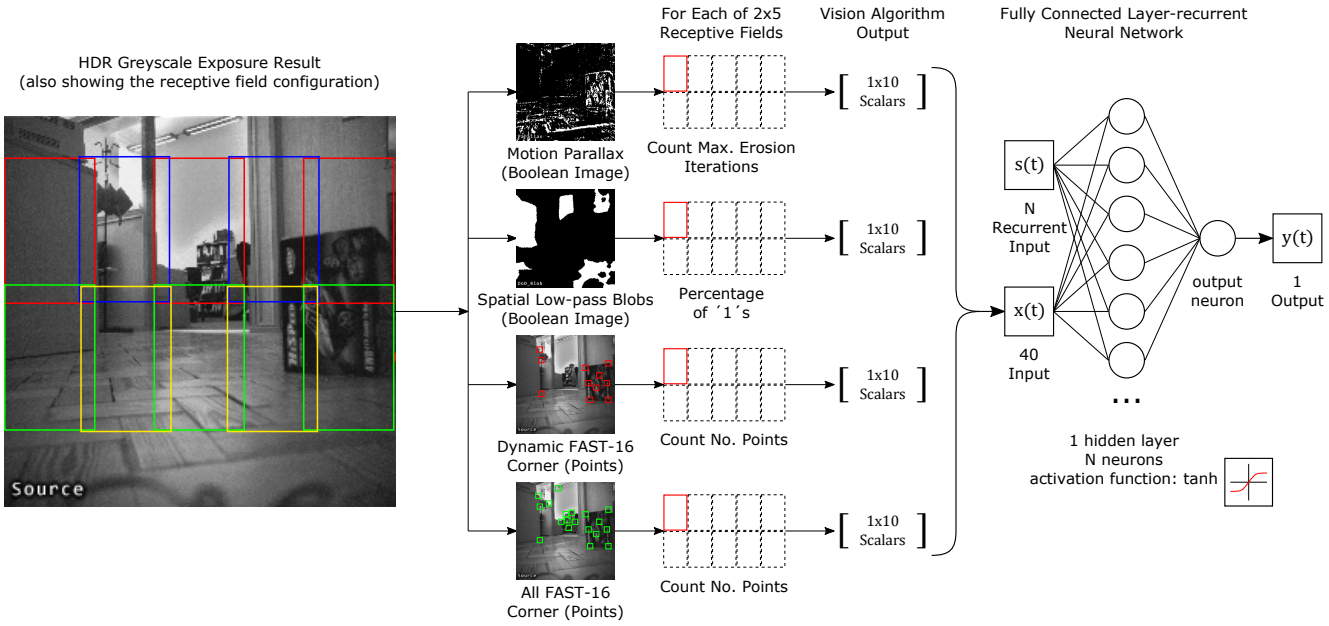


Fig. 2. Overview of the vision based proximity estimation system. The vision chip produces 4 types of feature images. Receptive Fields are used to digest these images into a vector of 40 scalar values. The layer-recurrent neural network is executed on the MCU to mimic a proximity sensor using only the input vector and the output of all neurons from previous time-step ($S(t)$). The horizontal field of view of the vision system is approximately 60°.

as the distance between the two cameras is critical to the effectiveness of such method.

In nature, many agile flying insects can evade obstacles using their compound eyes. Because the nervous systems behind them are unlikely to be advanced enough for explicit cognitive tasks, such evasive behaviours are likely carried out reflexively. Several biologically-inspired vision systems have been proposed, for example in [11] a very simple processing algorithm running on an MCU was used to control an ultra-light indoor aircraft in a high contrast artificial environment. In [12] an obstacle avoiding solution based on neuromorphic sensor-processor hardware has been presented. In [13], a vision based navigation was implemented for a nano drone weighing 27 g. The processor onboard the drone has a moderate level of optimization for simplified CNN algorithms, achieving a frame rate of 18 Hz.

III. SYSTEM DESCRIPTION

A. Hardware Platform

The hardware platform used in this work is shown in Fig. 1. An electric RC car was modified to be the vehicle platform which can move relatively fast (over 3 m/s). For training data acquisition, it was equipped with three IR proximity sensors (Sharp GP2) with a effective range of 80 cm, pointing at 30° left, center and 30° right. An Arduino board was used to sample the sensors and generate PWM signals. The control algorithm of the vehicle is computed by the "Scamp5d" vision system [14]. The vision system is an integration of a LPC4357 micro-controller (MCU) and a SCAMP-5 sensing/processing vision chip [15]. The MCU has two cores, an ARM Cortex M4 and a M0, sharing 136 KB of RAM while both of the cores are running at 204 MHz. The M0 core issues instructions to the vision chip. The M4 core has hardware floating point unit, and thus is suited for implementing several types of commonly used neural network architectures. The SCAMP-5 vision chip has 256×256 processing elements (PE). The M0 core issues instructions to all PEs of the vision chip to realize SIMD parallel computation. Each of the PEs contains a light sensor (image pixel), 12 binary registers and 7 analog registers. Logic and arithmetic operations can be performed on local memory, and using the four adjacent PEs. Basic arithmetic operations, such as addition, subtraction, divide-by-2, are executed with approximately 8-bit precision. Analog register data can be binarized based on a given threshold, realising 1-bit analog-to-digital conversion. Program branching across the PE array can be realized through masking all write operations is selected PEs. Mathematical morphology, such as erosion and dilation, can be executed efficiently across the binary registers.

Further hardware functionality on the vision chip includes:

- $Blur$: An image stored in an analog register array can be blurred (spatial low-pass filter) with one instruction. This blur can be repeated for a blur with a higher strength.
- $EventReadout$: The coordinates of the first '1' along a given direction in a binary register array can be acquired immediately (without outputting the entire array).
- $GlobalOr$: The event readout capability also means it is possible to determine whether there is at least one '1' in a binary register array.
- $GlobalSum$: an approximate sum of the analog values of a given region or the entire image can be obtained.

Using these global operations, results of image processing

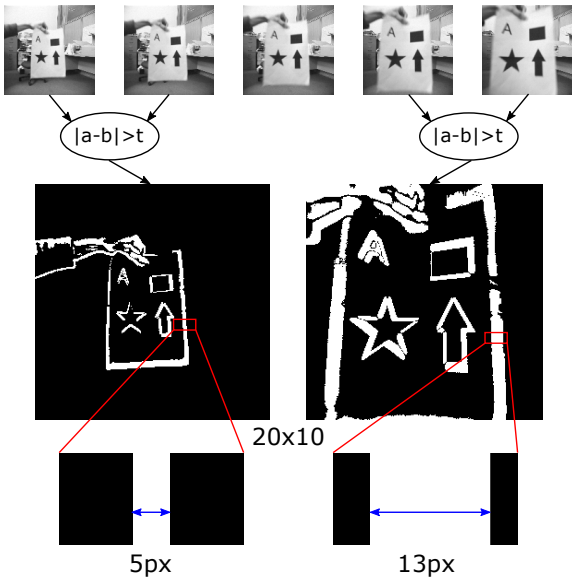


Fig. 3. Motion Parallax extraction was used to detect objects' relative motion in the image, which produces an image similar to an edge map. In a given window, the approximate magnitude of motion can be quantified by measuring the width the widest edge in the window. The algorithm to achieve this measurement is shown in Algorithm.1.

algorithms computed in the sensor-processor array can be reduced to scalar values, that can be read-out to the MCU.

B. System Architecture

The overall architecture of the proximity estimation system is illustrated in Fig. 2. The SCAMP-5 vision chip captures and stores an analog grey-scale high dynamic range (HDR) image. Then the vision chip proceeds to compute three types of vision algorithm, which were chosen because: (a) they contain monocular depth cues; (b) their computation is highly efficient for the vision chip. The details about these vision algorithms will be introduced in next section. To gather and send this information off the vision chip, a 2×5 array of receptive fields are defined across the image plane. Each receptive field is essentially a window used to digest the results of vision algorithms into scalar values. The receptive-fields produce a vector of 40 scalar values as the final output of the vision algorithms. This vector is then input to a fully connected layer-recurrent neural network which emulates the response of an IR proximity sensor. For each of the three proximity sensors, a separate network was used. The scalar results from the vision algorithms are noisy and have a temporal structure. Hence, a well-trained recurrent neural network is required to process such signals.

C. Vision Algorithms

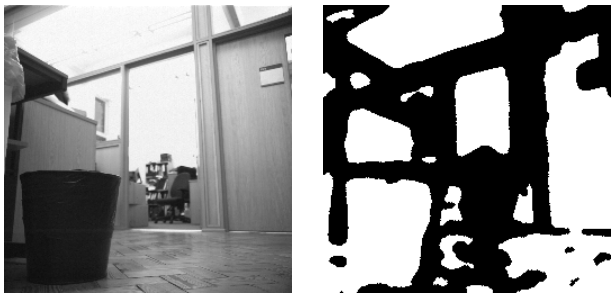
1) Motion Parallax Extraction: Motion parallax can be observed in the image of a moving camera, where closer objects move faster. A Boolean motion parallax image can be obtained by applying a threshold to the absolute difference between the current frame and the last frame. On such an image, wider edges are more likely belong to closer objects (see Fig. 3). The approximate maximum width of these edges can be measured through iterative erosion. The detail of the algorithm is shown in Algorithm 1. In the algorithm, from Line 12 to Line 18, any pixels that have at least one '0' neighbour will be cleared to '0'. Collective-wise, any blobs of '1's in the image will be eroded in a way that their width/height are reduced by 1 pixel. Through iterating such erosion until all '1's are gone, the number of erosion steps is the width of the widest edge in the region.

Algorithm 1 Motion Parallax Extraction

1: **procedure** MOTIONPARALLAXEXTRACTION
2: $A \leftarrow$ current input image.
3: $B \leftarrow$ last input image.
4: $t \leftarrow$ a threshold.
5: $R_w \leftarrow$ receptive field window Boolean image.
6: *init*:
7: $R_a \leftarrow |A - B| > t$.
8: $R_b \leftarrow R_a \wedge R_w$.
9: $steps \leftarrow 0$.
10: *loop*:
11: **if** $GlobalOr(R_b)$ **then**
12: $R_c \leftarrow \neg R_b$.
13: **if** *steps* **is odd then**
14: $R_d \leftarrow North(R_c) \vee East(R_c)$.
15: **else**
16: $R_d \leftarrow South(R_c) \vee West(R_c)$.
17: **if** $R_d = 1$ **then**
18: $R_b \leftarrow 0$.
19: $steps \leftarrow steps + 1$
20: **goto** *loop*.
21: **return** *steps*

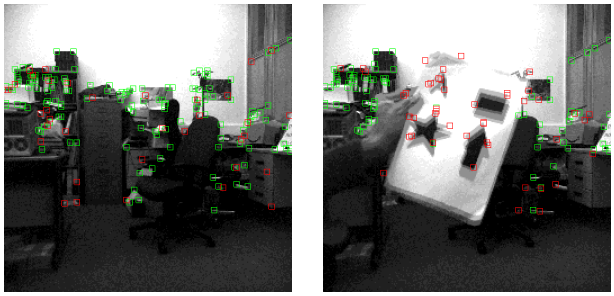
2) Spatial Band-pass Filtering: It is common (but not always) that a close object/surface on the image appears to be an area with uniform color, which has low spatial frequency. A Difference of Gaussian (DoG) process can be used to extract patterns of a certain spatial frequency [16]. On SCAMP-5, DoG-like filters can be efficiently achieved using the analog $Blur$ function. While this hardware-accelerated function does not precisely compute a Gaussian convolution kernel, its "blurring" effect is similar enough to realize the spatial band-pass effect. For the goal of proximity estimation, a high-pass DoG filter was used to binarize the image into uniform areas and feature-rich areas. The process is demonstrated in Fig. 4. After this process, the uniform areas on the image can be flagged. The percentage of flagged pixels in each of the receptive fields is estimated.

3) FAST-16 Corner Extraction: FAST-16 Corner Extraction algorithm [17] flags all the pixels that are on the corners of objects/patterns. Heuristically speaking, distant areas in a common scene are likely to contain a consistent number of corner features. Corner features on closer objects can be distinguished when the camera is mounted on a moving vehicle, because they move faster than the rest when the uniform egomotion was compensated. The FAST algorithm



(a) (b)

Fig. 4. Using Difference of Gaussian filter $|Blur(3) - Blur(0)|$ to extract plain area in the view. (a) Source image. (b) Result Boolean image after being post-processed by inversion, morphological opening and smoothing.



(a) (b)

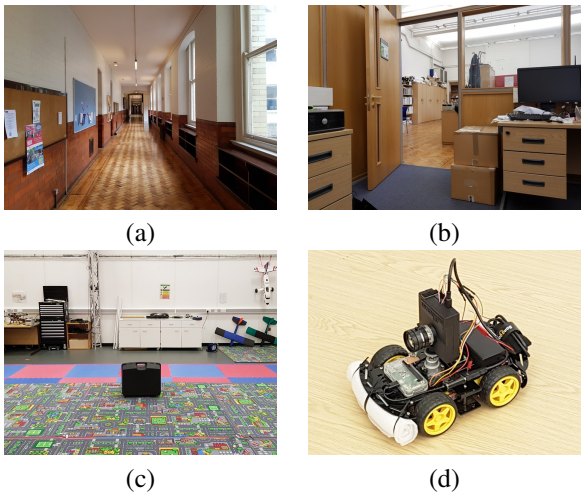
Fig. 5. FAST-16 Corner Extraction algorithm running on SCAMP-5, showing output corner features in two groups: static corners (green) and dynamic corners (red). (a) In a static scene, most corner are static. Dynamic corners, caused by noise in this case, are in the minority. (b) Moving objects in a region generate mostly dynamic corners. Uniform global motion of the entire image can be cancelled, which is explained in the text.

is inherently parallel and can be directly implemented on SCAMP-5's PE array through basic arithmetic operations and neighbour access. The result is a Boolean image, in which the '1's are the corner pixels in the image. The vision chip's $EventReadout$ was used to acquire the coordinates of these pixels for each of the receptive fields, requiring only 2 bytes per feature. Utilising the chip's $GlobalSum$ function, yaw and pitch egomotion can be estimated efficiently by searching for the global shift amount required to minimise the sum of absolute differences to the previous frame. Conventionally, tracking of feature points are done serially on a CPU (e.g. [5]). For SCAMP-5, the goal of the matching process was simplified and computed in parallel based on Boolean morphology. The Boolean image containing corner pixels from the previous frame is dilated by one step, and then used to mask the corner pixels this frame. When the frame rate is high enough relative to the motion speed and the uniform "scrolling" motion caused by the yaw motion of the vehicle is compensated, this mask process can clear majority of the static corner pixels and thus reveals the moving (dynamic) ones.

IV. Experiment

A. Neural Network Training

1) Training Data Collection: The RC model car was driven manually in an indoor environment (Fig. 6(a)&(b)).



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

Fig. 6. Experimental setup: (a) & (b) The environment used to collect the training data. (c) The environment of the systematic experiment and the obstacle, which was in a differently building and scenario. (d) The differential wheeled rover used in the experiment. The vision system and the proximity estimation neural networks on the rover were same as those on the RC model car. Only the collision avoidance controller was different, which enabled this vehicle to move fully autonomously.

The car was steered to avoid obstacles and walls. In addition, it had also been steered based on a random navigational plan. This was done to reduce the effect of visual feedback on the avoidance behaviour, which might cause the neural network to infer the presence of obstacles from the yaw motion of the car rather than motion depth cues. Vision output and proximity sensor output were collected at 50 Hz from 50 trials of 10 to 50 seconds each. Data from 25 trials was used for neural network training, while the rest was used for testing. In both cases, the data were also left-right-mirrored, which essentially created a data set of 100 sequences of 500–2500 time steps each.

2) Training Result: For each of the three proximity sensors, an independent RNN was trained. The training used Bayesian regularization back-propagation in MATLAB. A phase lead of 3 time steps was applied to the target output to: (a) cancel out the communication latency from the Arduino to Scamp5d; (b) reduce the effect of visual feedback.

Hidden layer size of the RNN and selection of the vision inputs had been studied, the result of which will be discussed in the next section.

3) Network Input and Size: The effectiveness of each of the vision features was accessed by using only a subset of them as the network input (M). The hidden layer size has been varied in $N \in 4, 6, 8, 10, 12, 14, 16$.

Fig. 7 plots the mean-squared-error (MSE) measure of the 50 test sequences for each of the M-N combination. Fig. 8 compares the proximity sensor output to the neural network output with $M = 1000$ and $M = 1111$ using $N = 12$. The $M = 1111$ group has a better performance than those groups with only one feature type. This implies that a combination of features can result in a better estimation than using only one type of feature. Among groups with two or three feature types, inconsistency can be observed: some performed as

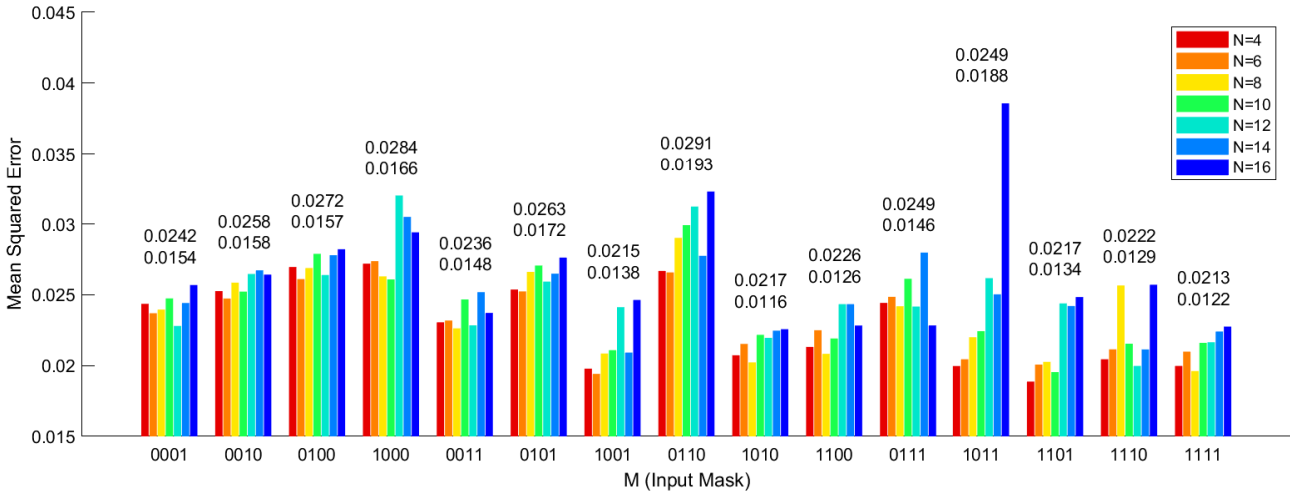


Fig. 7. Bar plot of the neural network performance. Each 'bar' represents the mean of 300 data points, which are the mean-square-error measure of both left and right RNN, three samples each, for the 50 test data sequences. The black values on top of the bars are the means of all seven bars in the group, while the blue values are the standard deviation. The input (M) was abbreviated by binary masks indicating which of the feature vectors were enabled, in the order shown in Fig. 2. For example, $M = 1100$ means only motion parallax and spatial low-pass filter were enabled.

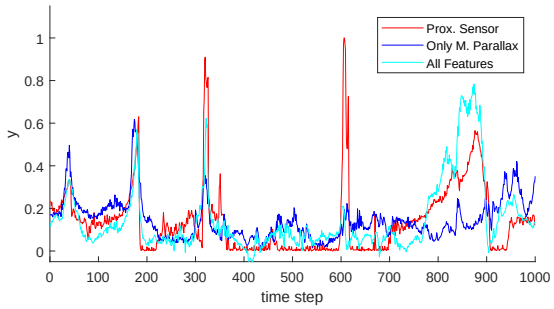


Fig. 8. Proximity distance estimated by the left neural networks when $N = 12$. The network with all four feature types ($M = 1111$) matched the proximity senor reading better than the one with only motion parallax ($M = 1000$). There is also one peak both networks failed to estimate.

poor as those with only one feature type whereas others performed as well as the $M = 1111$ group. Since the test data was also from the training environment, it could be possible that such scenario was more suited for a specific combination of features. Increasing the size of the hidden layer (N) did not improved the performance. Such a phenomenon could be caused by over-fitting because these vision features also have potential to recognize positions in the training sequences and form a case-wise classification when the neural network was given enough capacity. $N = 12$ was used for further physical experiment, which covered the influence of this phenomenon.

B. Obstacle Avoidance Experiment

1) Experiment Setup: A systematic experiment has been conducted to examine the performance of the system by using the output of the RNN to implement an obstacle avoidance behaviour. The experiment environment (Fig. 6(c)) was a different indoor location, with a different lighting condition and interior style to the training environment. The obstacle used was a container that was hard to detect for infrared proximity sensor due to it's color and surface properties. However, the vision based system should still be able to detect the container, because its features were similar to any boxes/desks in the training environment.

The experiment applied the set of left-center-right networks with $M = 1111, N = 12$, to be used as if they were the infrared proximity sensors. To realize the obstacle avoidance behaviour, a state-less reactive controller outputs the vehicle control parameters every frame. The controller was kept simple as it only used threshold-based rule and proportional control, so that the behaviour of the vehicle mostly depended on the proximity estimation system. Two vehicles were used:

1) The RC model car used in training without the proximity sensors. The reactive controller outputs the angle of the steering servo while the throttle of the car was still controlled manually through RC.
2) A differential wheeled rover (Fig. 6(d)). It was slower than the RC car, but easier to control due to its differential wheeled drive mechanism. Both the linear speed and angular speed of the rover were controlled by the reactive controller, which means this vehicle was moving fully autonomously.

2) Trajectory Tracking: For each of the vehicles, 30 trials were conducted. The experiment environment was installed with a motion tracking system (VICON). The trajectory plots of the vehicles in each of the trials are shown in Fig. 9. The RC model car collided with the obstacle in 2 out of the 30 trials, but the avoidance manoeuvre could still be observed from the trajectories. The rover avoided the obstacle successfully in all of the 30 trials. It can also be observed that the vehicles remained moving on a relatively unaltered course in the mostly obstacle-less environment until being close enough to the obstacle, and regained a straight course after clearing the obstacle. This indicates that there were no

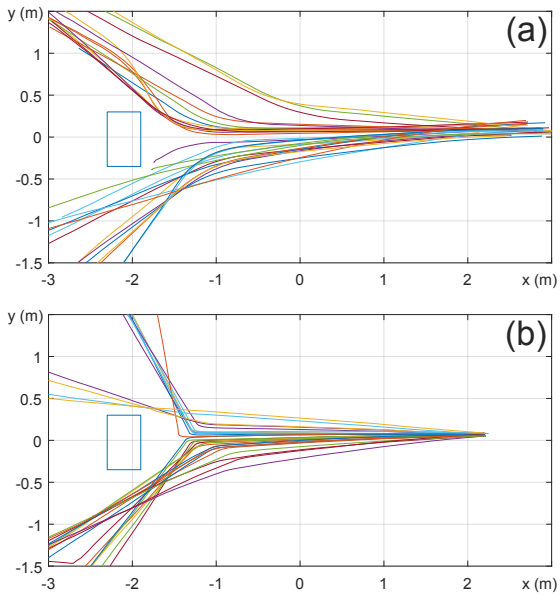


Fig. 9. Trajectories of vehicles in the obstacle avoidance experiment. The blue rectangle represents the obstacle. The speed of the vehicles in $x \in (-1.0, 1.0)$ were measured. (a) The RC model car. The reactive controller using the neural network output as the proximity input, and outputting the angle of the steering servo. The throttle was controlled manually. The minimum, maximum and average speed of the car were 0.90 m/s, 1.87 m/s and 1.15 m/s correspondingly. (b) The fully autonomous differential wheel vehicle. The reactive controller using the neural network output as the proximity input, and outputting the speed of the wheels. The minimum, maximum and average speed of the car were 0.64 m/s, 0.74 m/s and 0.71 m/s correspondingly.

significant false positives.

The speeds of the two vehicles were different. In theory, it would be optimal to change the frame rate if the networks applied were kept the same. However, the tracking system interferes with the camera when frame rate is not 50Hz. Nevertheless, the system performed robustly enough in the experiment.

C. Free Roaming Experiment

To assess the effectiveness of the system when the environment is more complex, the vehicles were put in an uncontrolled environment and roamed while avoiding collisions. The video of these trials can be found in the video attachment. Some failure modes can also be observed in the video. For example, chair legs, especially swivel chairs, can hardly be detected by the proximity sensors used in the training process. As a consequence, the trained vision based system failed to detect such structures. The vision based estimation system also failed to detect obstacles that were inherently hard to "see" from the perspective of the camera such as low steps on the floor.

D. Timing and Latency Test

The time cost for each of the procedures in the vision processing program was measured and listed in Table I. The computation time cost for the three neural networks running on the M4 core was 220 μs when $N = 12$. The reactive obstacle avoidance controller and the control signal

TABLE I
VISION ALGORITHM TIME COST

Processing Steps	Approx. Time Cost (μs)
Exposure Result Processing	30
Image Global Motion Estimation	702
FAST-16 Corner Detection	600
Corner Points Scanning	680
Motion Parallax	10
Motion Parallax Erosion Step Count	54–1000
Spatial Band-pass Filter	46
Spatial Band-pass Area Fill-rate Readout	360
Total	2500–3500 μs

transmission takes negligible amount of time. Thus, the control latency of the system was under 4 ms.

The vision algorithm has also been implemented on a Raspberry Pi 4 using OpenCV and a 320×240 webcam. With SIMD and multi-thread optimization, the time cost per-frame was around 30 ms while the power consumption was over 6 watts. While running the vision algorithm at 250 fps, the Scamp5d system consumes around 2.1 watts, which means $8\times$ higher performance using a third of the power consumption.

V. CONCLUSIONS

A vision based proximity estimation system was proposed and implemented using the Scamp5d embedded vision system. The vision chip captured the image and ran three types of feature extraction algorithms on the focal plane. The only output of the vision chip was scalar data digested from the vision features. The MCU in the vision system then ran three layer-recurrent neural networks to estimate the proximity distance similar to proximity sensors. Training data for the neural network were recorded when the car was driven manually. In the the obstacle avoidance experimental trials, the general effectiveness of the system was demonstrated.

The power consumption (<2.1 W) and the latency (<4 ms) of the system are low, because: (a) the feature extraction algorithms exploits the hardware acceleration provided by the vision chip, and thus can be executed highly efficiently while no image needs to be transferred, (b) the neural networks running on the MCU operate on abstract feature vectors, and thus are relatively simple.

The work described here provides a proof-of-concept demonstration. Further studies can look into a more thorough investigation of other types of vision features and receptive field configurations, as well as a variety of neural network sizes and architectures. The combination of on-sensor extraction of abstract features with neural network based post-processing may also be used to mimic other types of depth sensor. Future work will also be focused on exploiting vision feature data to achieve other tasks at the same time as depth estimation. The current iteration of the SCAMP vision system hardware is still relatively large, but a System-on-Chip integration of the system electronics, which is technically feasible, would lead to a powerful miniature embedded vision system that could be applied in many robotic vision applications.

REFERENCES

[1] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, ser. ICRA '17, May 2017, pp. 2043–2050.

[2] T. Lindeberg, "A computational theory of visual receptive fields," *Biological Cybernetics*, vol. 107, no. 6, pp. 589–635, Dec 2013.

[3] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," in *2011 IEEE Int. Conf. Robotics and Automation*, May 2011, pp. 5776–5783.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *2017 IEEE Conf. Computer Vision and Pattern Recognition*, ser. CVPR '17, July 2017, pp. 6612–6619.

[5] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *2013 IEEE Int. Conf. Robotics and Automation*, May 2013, pp. 1750–1757.

[6] K. Souhila and A. Karim, "Optical flow based robot obstacle avoidance," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, p. 2, 2007.

[7] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav navigation through indoor corridors using optical flow," in *2010 IEEE Int. Conf. Robotics and Automation*, ser. ICRA '10, May 2010, pp. 3361–3368.

[8] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Proc. 18th Int. Conf. Neural Information Processing Systems*, ser. NIPS'05, 2005, pp. 739–746.

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *2015 IEEE Int. Conf. Computer Vision (ICCV)*, December 2015.

[10] A. Saxena, J. Schulte, A. Y. Ng *et al.*, "Depth estimation using monocular and stereo cues." in *Proc. 20th Int. Joint Conf. Artificial Intelligence*, vol. 7, 2007, pp. 2197–2203.

[11] J. . Zufferey and D. Floreano, "Fly-inspired visual steering of an ultralight indoor aircraft," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 137–146, Feb 2006.

[12] M. B. Milde, H. Blum, A. Dietmüller, D. Sumislawska, J. Conradt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Front. Neurorobot.*, 2017.

[13] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano–uavs," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2019, pp. 604–611.

[14] J. Chen, S. J. Carey, and P. Dudek, "Scamp5d vision system and development framework," in *Proc. 12th Int. Conf. Distributed Smart Cameras*, ser. ICDSC '18. ACM, 2018, pp. 23:1–23:2.

[15] S. J. Carey, D. Barr, B. Wang, A. Lopich, and P. Dudek, "A 100,000 fps vision sensor with embedded 535 GOPS/W 256×256 SIMD processor array," in *Proc. 2013 IEEE Symp. VLSI Circuits*, Jun 2013, pp. 182–183.

[16] T. Lindeberg, "Image matching using generalized scale-space interest points," *Journal of Mathematical Imaging and Vision*, vol. 52, no. 1, pp. 3–36, May 2015.

[17] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. Tenth IEEE Int. Conf. Computer Vision*, vol. 2, Oct 2005, pp. 1508–1515.