TOWARDS THE IMPLEMENTATION OF VISION-BASED UAS SENSE-AND-AVOID SYSTEM

Luis Mejias, Jason J. Ford, John Lai
Australian Research Centre for Aerospace Automation (ARCAA)
Queensland University of Technology, QLD 4000, Australia

Keywords: *Detection algorithms, Filtering techniques, Filter banks, Obstacle avoidance, sense-and-avoid.*

Abstract

Machine vision represents a particularly attractive solution for sensing and detecting potential collision-course targets due to the relatively low cost, size, weight, and power requirements of the sensors involved. This paper describes the development of detection algorithms and the evaluation of a real-time flight ready hardware implementation of a vision-based collision detection system suitable for fixed-wing small/medium size UAS. In particular, this paper demonstrates the use of Hidden Markov filter to track and estimate the elevation (β) and bearing (α) of the target, compares several candidate graphic processing hardware choices, and proposes an image based visual servoing approach to achieve collision avoidance.

1 Introduction

The problem of unmanned aerial vehicle (UAV) collision avoidance or sense-and-avoid has been identified by the Office of the Secretary of Defense [1] and DeGarmo [2] as one of the most significant challenges facing the integration of UAVs into the national airspace. The full potential of UASs can never be realised unless sense-and-avoid issues are adequately addressed. A survey of potential 'sense and avoid' technologies for unmanned aerial vehicle (UAV) is presented in Karhoff *et al*. [3]. In their analysis, machine vision emerged as a promising means of addressing the sense and detect aspects of collision avoidance. Interestingly, vision is the primary means by which many organisms perceive the world and serves as a basis for their interactions therein. However, some studies have shown (BASI [4]) that it is actually difficult to detect and avoid a collision using the human visual system. Hence, it must be recognized that sensing by vision is not appropriate for all circumstances, for instance in very low light conditions affect adversely visible spectrum cameras. In addition, many challenges still need to be overcome, including: to guaranteed solution robustness, to achieve the required computational speed, to resolve various ambiguities, etc.

From the last three decades of work a two-stage processing paradigm has emerged for the simultaneous detection and tracking of dim, sub-pixel sized targets. Examples of this two-stage approach include works by Gandhi *et al*. [5, 6], Arnold *et al*. [7], and Barniv [8], Carnie [9] and Lai [10]. These two stages are: 1) an image pre-processing stage that, within each frame, highlights potential targets with attributes of interest; and 2) a subsequent temporal filtering stage that exploits target dynamics across frames. However, one of the key limitations of previous detection work relates to the inability of these previous approaches to provide higher-level target information, such as the target's relative heading in the image frame.

A second key limitation of this previous work is that the feasibility of the presented algorithms

have not yet been tested in real-hardware platforms and this remains a key challenge that faces any vision-based sense and avoid system. To date, public domain hardware implementation of vision-based sense and avoid systems have been limited to a small number, with the most significant developments being made by Utt et al. [11]; here, a combination of field programmable gate array chips and microprocessors using multiple sensors, was tested in a twin-engine Aero Comander Aircraft. Motivated by these hardware implementation challenges, this paper aims to exploit the capabilities of data-parallel arithmetic architectures such as graphics processing units (GPUs) which can outperform current CPUs by an order of magnitude. As discussed in Owens *et al*. [12], GPUs have been proven to be very capable parallel processing systems and have been demonstrated in applications ranging from medical image analysis [13], video processing applications [14, 15], image processing and robotics [16, 17, 18, 19], through to high performance processing applications [20].

For these above reasons, this paper describes the extension of our previous work through the implementation of our detection techniques on GPU devices and the addition of relative bearing and elevation estimation capabilities. Specifically this paper presents 1) a hidden Markov model (HMM) based filtering approach for the detection of aerial targets, 2) a control strategy for collision avoidance based on target dynamics and estimation of target relative bearing/elevation angles, 3) an implementation of our HMM detection algorithm on GPU-based hardware for real-time target detection, and 4) the characterisation of data processing speeds on various candidate GPU devices.

This paper is structured as follows: Section 2 describes morphological pre-processing and HMM temporal filtering techniques. Section 3 describes our proposed control strategy and a technique for estimating target heading information. Section 4 provides a description of our GPU implementation and the optimisation strategies undertaken. Section 5 presents experimental results illustrating processing rates on a variety of GPU devices. Finally, some discussions and conclusions are presented.

2 Detection approach

This paper considers an image pre-processing approach that exploits grayscale morphological operations to highlight potential targets, and a temporal filtering approach to detect and track persistent features (targets). Next, we describe the details of these two approaches.

2.1 Morphological pre-processing

In the first stage of processing we use a CMO morphological filter for low level detection. The CMO filter is based on operations known as top-hat and bottom-hat transformations (see [21] for more details) which at the same time are based in two basic image processing operations called dilation and erosion. Here, a pair of CMO filters using orthogonal 1D structuring elements is implemented. One CMO filter operates exclusively in the vertical direction, while the other operates exclusively in the horizontal direction. The vertical and horizontal structuring elements of the CMO morphological pre-processing filter are given by $s_v = [1,1,1,1,1]$ and $s_h = [1,1,1,1,1]$, respectively. Our implementation of the CMO filter procedure can be summarised as follows:

for $i = 1$ to n **do**
$v = D(E(image_i, s_v), s_v)$
$h = E(D(image_i, s_v), -s_v)$
$img_v = h - v$
$v = D(E(image_i, s_h), s_h)$
$h = E(D(image_i, s_h), -s_h)$
$img_h = h - v$
$result_i = min(img_v, img_h)$
end for

where D and E are the two fundamental image processing operation called dilation and erosion, respectively (see [21] for more details).

2.2 Temporal filtering

We consider the target detection as evaluating the likehood of two complementary hypotheses, H_1 and H_2, where H_1 is the hypothesis that there is

a single target in the field of view of the camera, and H_2 is the hypothesis that there is no target. Our filtering approach assume that under the hypothesis H_1, the target resides on a 2D discrete grid, that is the image plane, such as $I = \{(i,j) \mid 1 \leq i \leq N_v, 1 \leq j \leq N_h\}$, where N_v and N_h are the vertical and horizontal resolution of the 2D grid (image height and width, respectively). Let $N = N_v \times N_h$ be total number of grid points, and the measurements provided by the sensor be Y_k.

In our target detection problem, we represent a unique HMM state by the target pixel location (i,j) in the image, when present. Using a standard vector representation of an image, let any HMM state m be represented as $m = [(j-1)N_v + i]$, when the target is at location (i,j). In addition, let x_k denote the state (target location) at time k. The *HMM transition probabilities* (i.e likelihood between state transitions) is described by $A^{mn} = P(x_{k+1} = state\ m | x_k = state\ n) \forall (m,n) \in [1,N]$. In addition, *initial probabilities* $\pi^m = P(x_1 = state\ m)\ \forall m \in [1,N]$ are used to specify the probability that the target is initially located in state m. Finally, to complete the parametrisation let the *measurement probabilities* $B^m(Y_k) = P(Y_k | x_k = state\ m)\ \forall m \in [1,N]$ be used to specify the probability of obtaining the observed image measurements $Y_k \in [1,N]$ (see [22] for more details about the parameterisation of HMMs)

The HMM detection is achieved propagating recursively an un-normalised probabilistic estimate ($\alpha_k^i = P(Y_1, Y_2, ..., Y_k | x_k = state\ m)$) of the i target state (x_k^i) over time (see [23]). The procedure can be summarised as follows:

for $m = 1$ to N **do**
initialisation: $\alpha_1^m = \pi^m B^m(Y_1)$
recursion: for $k > 1$

$$\alpha_k^m = \left[\sum_{n=1}^{N} \alpha_{k-1}^n A^{mn} \right] B^m(Y_k)$$

end for

Two probability measures that facilitates the detection of the target are used: 1) the probability of measurement up to time k assuming H_1

$$P(Y_1, Y_2, ...Y_k | H_1) = \sum_{m=1}^{N} \alpha_k^m \tag{1}$$

and 2) the conditional mean filtered estimate of the state state m given measurements up to a time k assuming H_1

$$\begin{aligned} \hat{x}_k^m &= E\left[x_k = state\ m | Y_1, Y_2, ..., Y_k, H_1\right] \\ &= \frac{\alpha_k^m}{\sum_{n=1}^{N} \alpha_k^n} \end{aligned} \tag{2}$$

where $E[.|.]$ denotes the mathematical conditional expectation operation (see [24] for more details). Equation 1 may be interpreted as an indicator of target presence and equation 2 as a indicator of likely target locations. For computational efficiency, equation 2 can be evaluated directly from the following expression (see [22] for more details):

$$\hat{x}_k = N_k B_k(Y_k) A \hat{x}_{k-1} \tag{3}$$

where N_k is a scalar normalisation factor; $B_k(Y_k)$ is a $N \times N$ matrix such as $B_k(Y_k) = diag(B^m(Y_k))\ \forall m \in [1,N]$; A is a $N \times N$ matrix with elements A^{mn}; and $\hat{x}_k$ is a $N \times 1$ vector with elements $\hat{x}_k^m\ \forall m \in N$. In addition, note the following relationship between the normalisation factor N_k and the probability of measurements up to time k assuming H_1:

$$P(Y_1, Y_2, .., Y_k | H_1) = \prod_{l=1}^{k} \frac{1}{N_l} \tag{4}$$

For a HMM based detection approach we will let η_k denote a test statistic for declaring the presence of a target which is given by the following exponentially weighted moving average filter with a window length of L:

$$\eta_k = \left(\frac{L}{L+1}\right) \eta_{k-1} + \left(\frac{1}{L+1}\right) log\left(\frac{1}{N_k}\right) \tag{5}$$

We found that $L = 10$ offered good detection performance and smoothed out the transient resulted from noisy behaviour in the state transition. When η_k exceeds a predefined threshold, the HMM detection algorithm considers the

target to be present and located at state $\gamma_k = arg\ max_m(\hat{x}_k^m)$ at time k. The definition of η_k and γ_k is motivated by the filtering quantities discussed earlier.

A total of four independent filters operating over the same pre-processed image data were implemented [25]. This filter bank approach is less well characterised than the standard single HMM filter, and its application has not been prevalent in the context of dim-target detection from imaging sensors. The transition probability parameters of each filter in the HMM filter bank are designed to handle a range of slow target motion. These type of target motions correspond to transition probability matrices that only have non-zero probabilities for self-transitions and transitions to states nearby in the image plane (all other transitions have zero probability). Furthermore, it is important to note that the implemented HMM filter exploits the following probabilistic relationship between target location x_k and the pre-processed measurements Y_k:

$$B^m(Y_k) = \frac{P(Y_k^m|x_k = state\ m)}{P(Y_k^m|x_k \neq state\ m)}, \forall m \in [1,N] \quad (6)$$

In equation 6, we can note that $P(Y_k^m|x_k = state\ m)$ and $P(Y_k^m|x_k \neq state\ m)$ can both be determined on a single-pixel basis (rather than requiring the probability of a whole image, representing a computational advantage). In order to construct the measurement probability matrix $B_k(Y_k)$, estimates of the probabilities $P(Y_m^k|x_k = state\ m)$ and $P(Y_m^k|x_k \neq state\ m)$ are required. The latter describes the prior knowledge about the distribution of pixel values in the absence of a target (i.e. the noise and clutter distribution), while the former captures the prior knowledge about the distribution of values at pixels containing a target. The required probabilities for $B_k(Y_k)$ are trained directly from sample data. The probability $P(Y_m^k|x_k \neq state\ m)$ is estimated as the average frequency that each pixel value resulted from a non-target location. Using a similar procedure, $P(Y_m^k|x_k = state\ m)$ is estimated as the average frequency that each pixel value measurement resulted from a target location.

3 Control task and target dynamics estimation

Once the target is detected and sequentially tracked, we extract the target dynamics using a standard projective model. Using this model, and with due consideration of distortion models, we attempt to find the unit vector r and its orientation α (bearing) and β (elevation) (Figure 2a and b, respectively). The value of these quantities in successive stages can be used to infer properties of the target motion. Our motion model for the camera is shown in Figure 1 and this is used to present the following relationships. The point $T = (X,Y,Z)$ in camera coordinates is projected into the image (x,y) by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (7)$$

where (x,y) is the location in the image of the projected point and $f > 0$ is the focal length. More realistic models could take into consideration all the camera intrinsic parameters, e.g, the coordinates of the principal point (c_v, c_u) and ratio of pixel dimension γ, but these models are prone to accumulated error in the parameters. Let us denote the velocity of a observed target point, $V_i = \dot{T}$, in camera coordinates as $\dot{T} = -(V_i + \omega \times P_i)$, where $\dot{P} = (\dot{X}, \dot{Y}, \dot{Z})$ and $\omega = (\omega_x, \omega_y, \omega_z)$. Note that $V_i = -V$ is the velocity in body frame, if and only if the object is static w.r.t the body frame. If we develop the equation for $\dot{T}$ and combine it with the derivative w.r.t time of equation 7, we obtain

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = L \begin{bmatrix} V \\ \omega \end{bmatrix} \quad (8)$$

where

$$L = \begin{bmatrix} -\frac{f}{Z} & 0 & \frac{u}{Z} & \frac{uv}{f} & -(\frac{u^2}{f} + f) & v \\ 0 & -\frac{f}{Z} & \frac{v}{Z} & (\frac{v^2}{f} + f) & -\frac{uv}{f} & -u \end{bmatrix} \quad (9)$$

and

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{10}$$

This is the optical flow equation that relates the feature velocities $(\dot{x}, \dot{y})$ with the camera (aircraft) translational and angular velocities (V, ω). The matrix L in (eq. 9) that relates these magnitudes is sometimes called *interaction matrix* [26]. The above model takes into consideration the linear and angular velocities and is applicable in most cases where it is desired to control 6 d.o.f. motion. This model presents non-linearities in the interaction matrix and depends on the unknown features depth that cannot be measured directly. This represents a classical problem in IBVS (Image Based Visual Servoing), the estimation or approximation of the Image Jacobian [27][28]. In practice it is useful to linearise this model and use an approximate inverse of this matrix L^+ (or pseudoinverse [29]) or alternatively, a cylindrical representation could be used [30].

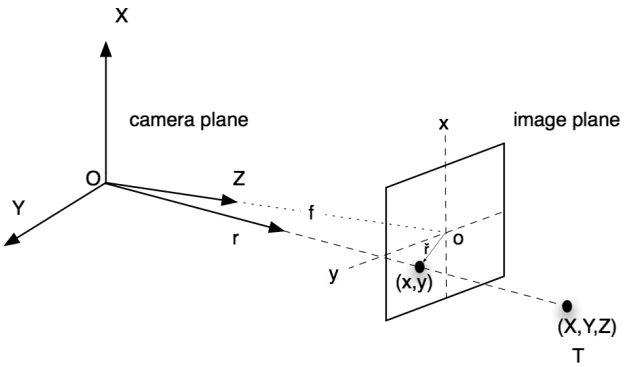

Fig. 1 Camera projection model.

Here, we will use a hybrid approach (i.e. both cylindrical and cartesian). Our control task is to move the actuator (camera/aircraft) away from the features (target). We achieve this using an exponential control law. Let's define $\hat{f}(s)$ as the error vector between current feature point $s = (x, y)$

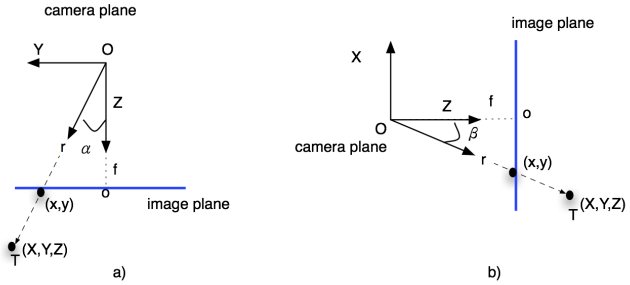

Fig. 2 Camera geometry for estimation of bearing (α) and elevation(β). a) shows the top view of the plane Y-Z for estimation of α. b) shows the side view of the plane X-Z for estimation of β

and the desired feature point $s^* = (x^*, y^*)$, such that $\hat{f} = (s - s^*)$. Using equations 8 and 9, we can express the control law such as

$$\delta = -\lambda L^+ \hat{f} \tag{11}$$

where λ is a positive gain and L^+ is the pseudoinverse of L. The error function is defined as exponential function such as

$$\hat{f} = \pm \phi_{max} e^{k(s-s^*)} \tag{12}$$

where k is a positive gain such that $k = k_1, \forall\, s \in [0, width/2]$ and $k = -k_2$ elsewhere, and ϕ_{max} is the commanded maximum heading. In our case, $s^* = width/2$ which correspond to the point $s^* = (0, width/2)$. The error function described by Equation 12 has the form shown in Figure 3.

The value of α (Figure 2a) and β (Figure 2b) can be computed as follows. From Figure 2a, target relative bearing can be defined as

$$\alpha = \tan\left(\frac{y - \frac{width}{2}}{f}\right) \tag{13}$$

and target relative elevation (Figure 2b) as

$$\beta = \tan\left(\frac{x - \frac{height}{2}}{f}\right) \tag{14}$$

where (x, y) is the location of the target in the image plane, $f > 0$ is the camera focal length, and width and height are the image dimensions. In

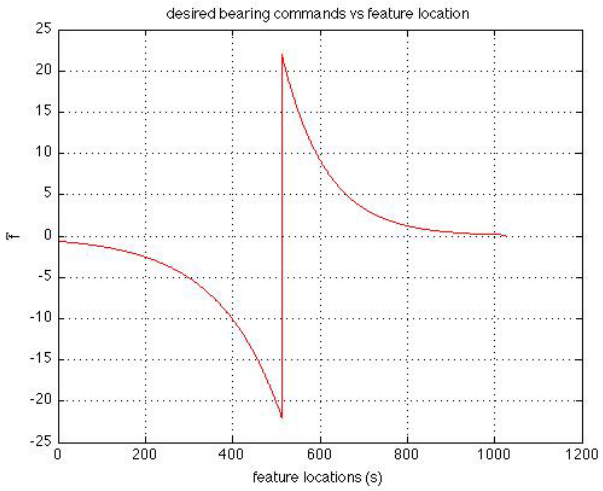

Fig. 3 Control signal in the feature space

order to infer the evolution of target in the image plane we use the $\dot{\alpha}$ and $\dot{\beta}$. This type of information is useful in determining whether the target represents a real collision threat or not. Furthermore, note that α and β have a role in relation to Equation 12 through the target location (x, y).

The above control approach is currently under lab testing and will soon be tested on a real platform.

4 Algorithm implementation

As mentioned in section 2, our HMM based detection filter involves two processing stages. The first stage, the morphological processing stage implementation involves a mathematical compute-intensive task that requires little flow control. The second stage, the temporal filtering stage is again compute-intensive. In the work here, the HMM filter has been implemented using a SIMD (Single Instruction, Multiple Data) approach to allow flight ready real-time operation. In our implementation, we have used the Compute Unified Device Architecture (CUDA [31]) which is a Nvidia application programming interface (API) that allows us to exploit the parallelism of a GPU device.

The implementation flow is sequential and begins with the CPU host transferring the current image to the GPU device memory. The GPU host then schedules a parallel set of operations. After the GPU device operations have been scheduled, the CPU is free to perform other tasks while it waits for the image processing operations to complete. During this CPU 'wait time', the GPU executes the scheduled operations in the order requested but will perform these operations in parallel and will finish these operations significantly sooner than if these operations were instead performed on the CPU. Once the GPU has completed its operations, the CPU then requests the resultant output and stores this in RAM. The program flow for the detection filter, as described above, is shown in Figure 4.

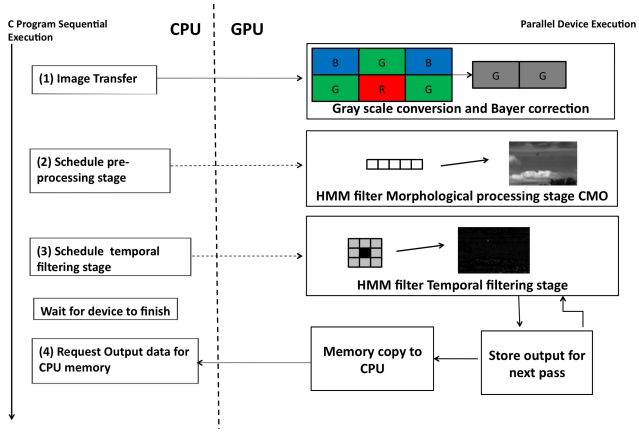

Fig. 4 HMM filter GPU implementation architecture.

With the motivation of understanding the scalability to future hardware and how to achieve real-time filter implementation on cheaper graphics processing units, we have examined some optimisations strategies, as well as different GPU models. We have implemented the filter using CUDA Kernels, which are a special type of C function that are executed N times in parallel by N different CUDA threads [31]. Threads are grouped into blocks, and should communicate only with threads in the same block using quick access L1 cache type memory.

The block size (and therefore the number of threads per block) is limited and can be optimised to suit the task, the amount of cache memory required, and the particular GPU limit. Therefore,

	GTX280	8800GTS	9500GT
Number of Multiprocessors	30	12	4
Compute Capability	1.3	1.0	1.1
Average Frame rate	150Hz	53Hz	11Hz
Optimised Threads per Block	1024	768	768
Optimised min no. of Blocks	60	24	8
Improvement over CPU-based computing	x20	x7	x1.5

Table 1 Performance results in three GPU hardware versions

we observed the following: a) to avoid un-utilised warps and to maximumise utilisation of blocks, the number of threads per block should always be a multiple of 32, b) To optimise block utilisation we must ensure, at least, an equal number of blocks as multiprocessors. c) Finally, we should choose as high as possible number of threads per-block, obviously limited by the compute capability and the available registers.

Table 1 shows the optimisation choices in terms of No. of blocks and threads for different GPU models and also the performance increase over standard CPU implementation for 3 different CUDA enabled GPU devices. We have approached the optimisation in terms of trying to understand the working principles of the GPU hardware and CUDA API parallelism to maximise its potential for the task we are dealing with, rather than optimising the detection algorithm for this particular GPU hardware. Future potential improvements include efforts to limit the use of conditional branches in the temporal filtering stage. In the next section, we will examine the processing rate of our proposed detection algorithm on the three candidate GPU devices.

5 Filter performance on GPU Hardware

In this section we consider the processing rates that can be achieved by various GPU hardware when running our proposed HMM detection algorithm. We express the processing rates in terms of the average frame rate obtained when processing 1024-by-768 pixel images encoded at 8 bits per pixel. For comparison purposes, we have established that a standard C implementation of the detection algorithm running on a CPU-based computing device[1] is able to process image frames at an average rate of approximately 7.58 frames per second (these figures are provided purely for illustrative purposes and do not reflect a fully optimised C implementation). This will serve as a baseline for comparison with processing rates on GPU hardware.

The CUDA implementation of our detection algorithm (as described in an earlier section) was tested on three candidate NVIDIA GeForce GPUs: 1) GTX 280; 2) 8800 GTS; and 3) 9500 GT. Table 1 illustrates the frame rates achieved on these three candidate GPUs.

These results demonstrate that GPUs have the potential to greatly accelerate the rate at which image processing tasks are performed. We also highlight that faster processing rates were achieved on GPUs with more multiprocessors. Motivated by this, we have selected a NVIDIA GeForce 9600 GT GPU (low-power version) for use in our flight-ready hardware configuration (this hardware is currently being integrated onto a UAV platform). This GPU offers a good balance between processing performance, power consumption and size. It has 8 multiprocessors, a compute capability of 1.1, and consumes only 59 Watts of power.

Figure 5 illustrates the relationship between processing rates and the number of multiprocessors based on data from Table 1. We can use this figure to roughly estimate the processing rate that can be achieved by the 9600 GT GPU (with-

[1] Intel Pentium IV processor @ 3.2 GHz with hyper-threading; 1 Gb SDRAM @ 666 MHz; Linux Ubuntu 32-bit operating system

out the need to perform extensive simulations). Given that the GPU has 8 multiprocessors, we anticipate that it will be able to process image information at a rate of approximately 30 Hz, which will be sufficient for real-time target detection. We highlight that there is scope for further improvement of this rate, as we have yet to exploit advanced GPU code optimisation techniques (such as pipelining and dynamic memory allocation methods).

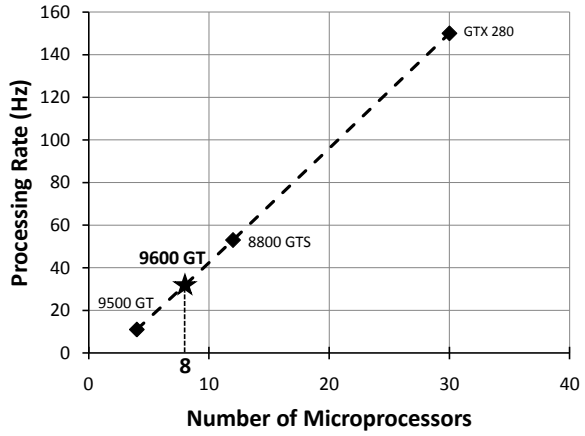

Fig. 5 Relationship between processing rate and number of multiprocessors.

6 Acknowledgements

This research was supported under Australian Research Council's Linkage Projects funding scheme (project number LP100100302). Engineering and flight testing carried out in support of this research was provided by the Smart Skies Project, which is funded, in part, by the Queensland State Government Smart State Funding Scheme

7 Conclusions

In this paper we have proposed a HMM-based target detection algorithm, a target heading/elevation estimation approach, and a collision avoidance control strategy. Furthermore, we have described an approach for implementing our detection algorithm on GPU hardware, and demonstrated the advantages of GPU devices over traditional CPU-based computing hardware in achieving real-time processing speeds. We are currently engaged in the integration of flight-ready hardware onto a suitable UAV platform, and plan to conduct closed-loop collision avoidance flight trials in the near future to verify our proposed detection algorithm and control strategy.

References

[1] Office of the Secretary of Defense, "Unmanned systems roadmap," Tech. Rep., Department of Defense, 2007.

[2] M. T. DeGarmo, "Issues concerning integration of unmanned aerial vehicles in civil airspace," Tech. Rep., The MITRE Corporation, 2004, MP 04W0000323.

[3] B.C. Karhoff, J.I. Limb, S.W. Oravsky, and A.D. Shephard, "Eyes in the domestic sky: An assessment of sense and avoid technology for the army's warrior unmanned aerial vehicle," April 2006, pp. 36–42.

[4] BASI, "Limitations of the sense and avoid principle," Tech. Rep., The Bureau of Air Safety Investigation, 1991.

[5] T. Gandhi, M.-T. Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless, "Performance characterisation of the dynamic programming obstacle detection algorithm," *IEEE Trans. Image Process.*, vol. 15, pp. 1202–1214, May 2006.

[6] T. Gandhi, M.-T. Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless, "Detection of obstacles in the flight path of an aircraft," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, pp. 176–191, Jan. 2003.

[7] J. Arnold, S. W. Shaw, and H. Pasternack, "Efficient target tracking using dynamic programming," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 29, pp. 44–56, Jan. 1993.

[8] Y. Barniv, "Dynamic programming solution for detecting dim moving targets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-21, pp. 144–156, Jan. 1985.

[9] R. Carnie, R. Walker, and P. Corke, "Image processing algorithms for uav "sense and avoid","

in *IEEE Int. Conf. on Robotics and Automation*, Orlando, May 2006.

[10] J. Lai and J. J. Ford, "Relative entropy rate based multiple hidden markov model approximation," *IEEE Trans. Signal Process.*, vol. 58, pp. 165–174, 2010.

[11] J. Utt, J. McCalmont, and Mike Deschenes, "Development of a sense and avoid system," *American Institute of Aeronautics and Astronautics (AIAA)*, 2005.

[12] J. D Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. E. Lefohn, and T. J. Prurcell, "A survey of general-purpose computatioin on graphics hardware," Tech. Rep., Eurographics 2005, Styate of the Art Reports, August 2005.

[13] Lei Pan, Lixu Gu, and Jianrong Xu, "Implementation of medical image segmentation in cuda," in *Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on*, May 2008, pp. 82–85.

[14] Wei-Nien Chen and Hsueh-Ming Hang, "H.264/avc motion estimation implmentation on compute unified device architecture (cuda)," in *Multimedia and Expo, 2008 IEEE International Conference on*, 23 2008-April 26 2008, pp. 697–700.

[15] P. Kehoe and A.F. Smeaton, "Using graphics processor units (gpus) for automatic video structuring," in *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on*, June 2007, pp. 18–18.

[16] P. Michel, J. Chestnut, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-Nov. 2 2007, pp. 463–469.

[17] P. Carr, "Gpu accelerated multimodal background subtraction," in *Computing: Techniques and Applications, 2008. DICTA '08.Digital Image*, Dec. 2008, pp. 279–286.

[18] S. Fukui, Y. Iwahori, and R.J. Woodham, "Gpu based extraction of moving objects without shadows under intensity changes," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, June 2008, pp. 4165–4172.

[19] J. Fung and S. Mano, "Openvidia: parallel gpu computer vision," in *Proceedings of the 13th annual ACM international conference on Multimedia*, 2005.

[20] Zhe Fan, Feng Qiu, A. Kaufman, and S. Yoakum-Stover, "Gpu cluster for high performance computing," in *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*, Nov. 2004, pp. 47–47.

[21] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing using MATLAB*, chapter Morphological Image Processing, pp. 334–377, Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

[22] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov Models: Estimation and Control*, Springer-Verlag, Berlin, 1995.

[23] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[24] P. Billingsley, *Probability and Measure*, Wiley, New York, 3rd edition, 1995.

[25] J. Lai, J. J. Ford, P. O'Shea, and R. Walker, "Hidden markov model filter banks for dim target detection from image sequences," in *Digital Image Computing: Techniques and Applications (DICTA)*, 2008.

[26] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," in *IEEE Transaction on Robotics and Automation*, June 1992, vol. 8, pp. 313–326.

[27] A. C. Sanderson and L. E. Weiss, "Adaptative visual servo control of robots," *In Robot Vision (A. Pugh, ed)*, pp. 107–116, 1983.

[28] Lee Weiss, *Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach*, Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 1984.

[29] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82 –90, dec. 2006.

[30] M. Iwatsuki and N. Okiyama, "A new formulation of the visual servoing based on cylindri-

cal coordinated system," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 2, pp. 266–273, April 2005.

[31] NVIDIA, "Cuda (compute unified device architecture) programming guide 2.0," 2009.

Copyright Statement