The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Vision-Based Detection and Tracking of Aerial Targets for UAV Collision Avoidance

Luis Mejias, Scott McNamara, John Lai, Jason Ford

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 (as opposed to radar). This paper describes the development and evaluation of a vision-based collision detection algorithm suitable for fixed-wing aerial robotics. The system was evaluated using highly realistic vision data of the moments leading up to a collision. Based on the collected data, our detection approaches were able to detect targets at distances ranging from 400m to about 900m. These distances (with some assumptions about closing speeds and aircraft trajectories) translate to an advanced warning of between 8-10 seconds ahead of impact, which approaches the 12.5 second response time recommended for human pilots. We make use of the enormous potential of graphic processing units to achieve processing rates of 30Hz (for images of size 1024-by-768). Currently, integration in the final platform is under way.**

I. Introduction

Currently, one of the major hurdles for the integration of UAVs in the civil airspace is the detect, sense and avoid capability (see [1][2]). A survey of potential technologies for unmanned aerial vehicle (UAV) detect, sense and avoid is presented in Karhoff *et al*. [3]. Their analysis concluded that the visual/pixel based technology offers the best chances for regulator approval. To date, public domain hardware implementation of vision-based detect, sense and avoid systems have been limited to a small number, with the most significant development made by Utt et al. [4], here a combination of field programmable gate array chips and microprocessors using multiple sensor, was tested in a twin-engine Aero Comander Aircraft.

A challenge that faces any vision-based sense and avoid system is the requirement of real-time operation. Motivated by this fact, this paper exploits the capabilities of data-parallel arithmetic architectures such as graphics processing units (GPUs) which can outperform current CPUs by an order of magnitude, and which have proven to be very capable parallel processing systems as presented in Owens *et al*. [5].

Over the last three decades, 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*. [6], [7], Arnold *et al*. [8], Barniv [9] and Lai *et al*. [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. The latter is often based on a track-before-detect concept where target information is collected and collated over a period of time before the detection decision is made. In the first category, specific implementations of the morphological pre-processing approach include the Hit-or-Miss filter by Schaefer and Casasent [11], Close-Minus-Open (CMO) filter by Casasent and Ye [12], and the Top-Hat filter by Braga-Neto *et al*. [13]. Although a large proportion of research has focused on IR images, there are recent examples by Carnie *et al*. [14], Gandhi *et al*. [6], Dey *et al*. [15] of morphological processing and filters being incorporated into target detection algorithms operating on visual spectrum images.

In the second category, the temporal filtering stage that follows the image pre-processing is designed to extract image features that possess target-like temporal behaviour. For this role, there are two particular filtering approaches that have received much attention in the literature: Viterbi based approaches [16], [17] and Bayesian based approaches [18]. In this paper, we extend our previous hidden Markov models (HMM) approach [10] to implement it in a GPU. In this paper, we specifically presents 1) a demonstration of coordinated UAV flights performing a collision scenario, 2) an application of HMM for detection of aerial targets and its implementation of our algorithm in GPU-based hardware for realtime detection, 3) analysis of the detection performance in terms of range and sensitivity.

This paper is structured as follows. Section II describes the morphological and temporal techniques definitions. Section III provides a description of the GPU implementation. Section IV presents the experimental setup and algorithm evaluation. Finally, conclusions are presented.

II. 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.

A. Morphological Processing

In particular, we use the CMO morphological filter for low level detection. The CMO method is based on operations known as top-hat and bottom-hat transformations (see [19] for more details) which at the same time are based in two basic image processing operations called dilation and

The authors are with the Australian Research Centre for Aerospace Automation School of engineering Systems, Queensland University of Technology, QLD 4001, Australia {luis.mejias}, {s.mcnamara}, {js.lai}, {j2.ford}@qut.edu.au

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 [19] for more details).

B. 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 [20] 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 [21]). The procedure can be summarized 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 [22] 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 [20] 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 the HMM filtering approach, let η_k, the test statistic for declaring the presence of a target, be given by the following exponentially weighted moving average filter with a window length of L:

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

Empirically, we found that L = 10 offered better results in smoothing 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 [10]. 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.

III. Algorithm Implementation

As described in section II, the HMM filter is a two stage filter; the morphological processing stage implementation follows a mathematical compute-intensive task that requires little flow control. The temporal filtering stage is again compute-intensive however the implementation used here required flow control. 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 [23]) a Nvidia application programming interface (API) that allows to exploit parallelism of the GPU. The implementation flow is sequential and begins with the CPU host transferring the current image to process, to the GPU device memory. Then the GPU host 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 operations scheduled to it in the order requested but performs the operations in parallel, and therefore significantly faster than 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 HMM filter, as described above, has been included in Fig. 1.

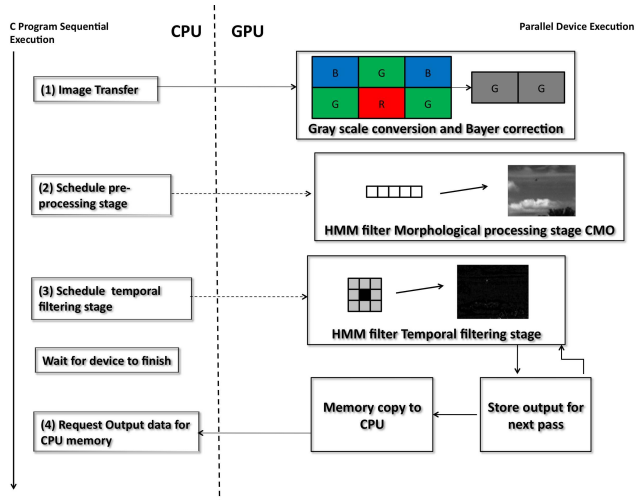


Fig. 1. HMM filter GPU implementation architecture.

In this paper, one important optimisation technique was examined with the motivation of understanding the scalability to future hardware and how to achieve real-time filter implementation on cheaper graphics processing units.

Our implementation uses CUDA kernels, which are a special type of C function that is executed N times in parallel by N different CUDA threads [23]. 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. Consequently, 1) to avoid un-utilised warps the number of threads per block should always be a multiple of 32. 2) we should ensure at least an equal number of blocks as multiprocessors. 3) Finally, the number of threads per-block must chosen as high as possible, limited obviously by compute capability and available registers.

Table I shows the appropriate optimisation choices and also the performance increase over standard C implementation in a CPU (see Section IV-B) 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, instead of shaping the HMM filter algorithm to ensure it is compute-optimised and requires minimal control flow. Future, attempts include methods to limit, for example, the use of conditional branches in the temporal filtering stage.

IV. Experimental Results and System Evaluation

A. Experimental setup

Two fixed-wing UAVs were deployed to collect suitable test data: 1) a Flamingo UAV (Silvertone UAV, 2.9m length and 4m wing span) and 2) a Boomerang 60 model airplane

a) b)

Fig. 2. Deployed UAV platforms for data collection. a) Flamingo UAV with camera pod (used as own aircraft) and boomerang (used as target). b) Actual moment of an encounter scenario showing both platfroms

(Phoenix Models, 1.5m length and 2.1 wingspan). The Flamingo was powered by a 26cc 2-stroke Zenoah engine and the boomerang by a O.S. 90 FX engine. The avionics payload of the Flamingo included a MicroPilot® MP2128g flight controller, Microhard radio modems, an Atlantic Inertial SI IMU04 inertial measurement unit (IMU) and a separate NovAtel OEMV-1 GPS device (both housed together with the camera), and an extensively customised PC104 mission flight computer. In contrast, the Boomerang only possessed a basic setup that featured a MicroPilot®MP2028g flight controller and Microhard radio modems.

In our experiments, the Flamingo served as the image data acquisition platform and was further equipped with a fixed non-stabilised Basler Scout Series scA1300-32fm/fc camera fitted with a Computar H0514-MP lens with 5mm focal length. The camera could be turned on and off remotely from the ground control station, and was configured to record 1024 by 768 pixel resolution image data at a rate of 15Hz with a constant shutter speed to maintain consistent lighting in the image frames. The captured image data was timestamped during flight so that it could be later correlated with inertial and GPS-based position logs from both aircraft in order to estimate the detection range. A solid-state hard-disk was used to store the recorded image data, as opposed to conventional mechanical disk drives which may be susceptible to vibrations during flight. Figure 2 shows the UAV platforms configured for data collection.

B. System evaluation

We have evaluated the performance of the proposed algorithm in terms of frame rate achieved processing 1024-by-768 pixel images with 8 bits per pixel, and using two types of signal-to-noise-ratio (SNR) quantities (defined later in this section). To evaluate the performance in terms of frame rate, we used as a baseline two software implementations in CPU, MATLAB and standard C, respectively. Using a total of 300 image frames, we found that MATLAB took 896 ms/per frame with an avg. frame rate of 1.12 fps and standard C took 133 ms/per frame with an avg. frame rate of 7.58 fps. These results are still far from real time, but represent a benchmark to compare against the GPU implementations. Note that the current scenarios and analysis were performed in sequences with approximately 8000 frames each.

For the GPU case, we used a baseline system consisted of a CPU Intel Pentium IV 3.2GHz, 1GB SDRAM @666Mhz, NVIDIA GTX 280 1024MB running Linux Ubuntu. Using this system we achieved processing rates of approximately 150Hz using the CUDA implementation. However, in order to test the scalability and impact of the number of GPU multicores, we tested the same implementation in two additional GPU cards, a GeForce 8800GTS (medium range) and a GeForce 9500GT (low range). Table I sums up the experimental results for the proposed algorithm on each GPU. It can be seen that power of the GPU can be measured by the number of multiprocessors. The 9500GT when tested, performed 1.5 times faster than a straight C implementation displaying that very cheap GPUs can provide substantial processing power in a compute-intensive operation and free up the CPU for performing other tasks. It is clear that the processing rates that can be achieved using GPUs are well in excess of a common 30Hz frame-rate.

Currently, we are in the process of implementing a flight-ready hardware using a mini-ITX computer based on an Intel Core Duo 800Mhz, 2GB SDRAM running Linux Debian and a low-power version GPU (Geforce 9600GT). 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. This translates to an approximate processing rate of 28.5Hz (full system tested in the lab). We highlight that there is still scope for further improvement in processing speeds, as we have yet to exploit advanced GPU code optimisation techniques (such as pipelining and dynamic memory allocation methods). This system is under testing on a Cessna 172.

The performance of the detection was evaluated in terms of SNR using the amount of image jitter as an indicator. A low amount of jitter is defined as involving apparent inter-frame background motion of between 0 and 1 pixels per frame. The metrics used to characterise the detection performance are defined as: 1) a target distinctness SNR (TDSNR), and

	GTX 280	8800 GTS	9500 GT
Number of multiprocessors	30	12	4
Compute capability	1.3	1.0	1.1
Optimised threads per Block	1024	768	768
Optimised min. no. of blocks	60	24	8
Performance increase over C implementation	20.5X	7.3X	1.5X

TABLE I

PERFORMANCE RESULTS IN THREE GPU HARDWARE VERSIONS

Scenario	Detection Range (m)	$\Delta DSNR$
1	412	38.04
2	562	31.94
3	881	19.55

TABLE II

TARGET DETECTION PERFORMANCE

2) a false-alarm distinctness SNR (FDSNR). The TDSNR provides a quantitative measure of the detection capability of the algorithm, and is defined as $TDSNR = 10log_{10}(\frac{P^T}{P^N})^2$, where P^T is the average target pixel intensity and P^N is the average non-target pixel intensity at the filtering output. In general, the more conspicuous the target is at the output of the filter, the higher the TDSNR value. Complementary, FDSNR measures the tendency of the algorithm to produce false-alarms, and is defined as $FDSNR = 10log_{10}(\frac{P^F}{P^N})^2$, where P^F is the average of the highest non-target pixel intensity and P^N is the average non-target pixel intensity at the filtering output. Strong filter responses away from the true target location will tend to increase the FDSNR value. For convenience, we will let $\Delta DSNR = TDSNR - FDSNR$ denote the difference between the two SNR metrics.

Overall, the $\Delta DSNR$ values seem to provide a reasonable indication of the detection algorithm performance. Using as a baseline the value from the low jitter scenario, which we denote by $\Delta DSNR_0$, the results suggest that as a rough rule-of-thumb successful tracking can be accomplished under a particular jitter scenario x when: $\Delta DSNR_x > 0.5\Delta DSNR_0$, where $\Delta DSNR_x$ is the $\Delta DSNR$ value corresponding to jitter scenario x.

We used data from three engagement scenarios. For illustration, we show only two encounter scenarios in Figures 3 and 4. Figures 3a and 4a show the trajectories of both aircrafts, where the asterisks denote first detection. Similarly, Figures 3b and 4b show the ROI of the frame (first detection) where the target is highlighted in a square. Furthermore, Table II shows the detection range results for the three scenarios. Overall, we obtained detection ranges that are generally consistent with the results reported in an earlier study [14]. Considering that a boomerang is 6-7 times smaller in size than a Cessna, the detection range of 6km is roughly in proportion with the ones in Table II. While we do not claim that a linear relationship exists between target size and detection distance, this comparison of detection range results reinforces the intuitive notion that larger targets should be able to be detected at greater ranges. In our final implementation we have made use of standard image stabilisation approaches to minimise the induced jitter in images.

We have used detection range as a metric for comparing the performance of the detection algorithm. However, from an operational point of view it is the time-to-impact from the point of detection that is perhaps more informative. Based on a combined closing speed of 51m/s in scenario 1, the time-to-impact estimated was 8 sec. In scenario 2 , the time-to-impact estimated was 10 sec (based on a combined closing speed of 53m/s). It is clear that these times are below the recommended 12.5 seconds [24]; however, our results must be considered in the appropriate context.

V. CONCLUSION

This paper have described a detection and tracking algorithm using morphological pre-processing and Hidden Markov Model filters. The system is implemented in Graphics Processing Units and tested using real data of encounter scenarios taken from real UAV flights. This system clearly represents a step forward with regards to visual/pixel based sense and avoid technologies for aerial vehicles, bringing this type of vehicles a step closer to its integration in civilian airspace. Implementation of this detection system in an unmanned aerial vehicle as well as in a Cessna 172 are currently underway.

VI. ACKNOWLEDGMENT

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.

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., MITRE, 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," in *IEEE Systems and Information Engineering Design Symp.*, April 2006, pp. 36–42.

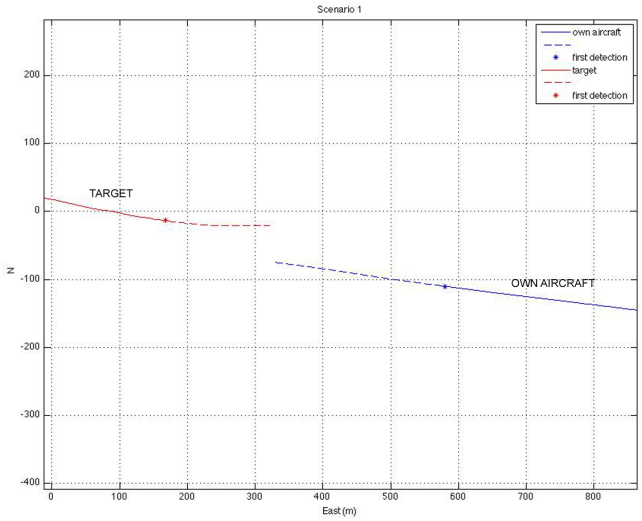
[4] J. Utt, J. McCalmont, and Mike Deschenes, "Development of a sense and avoid system," *American Institute of Aeronautics and Astronautics (AIAA)*, 2005.

[5] 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.

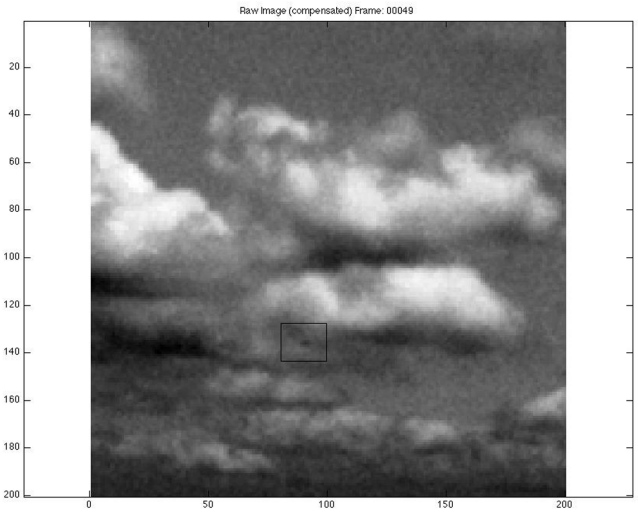
[6] T. Gandhi, M.-T. Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless, "Performance characterisation of the dynamic programming obstacle detection algorithm," *IEEE Trans. Image Process.*, vol. 15, pp. 1202–1214, May 2006.

[7] 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.

[8] 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.

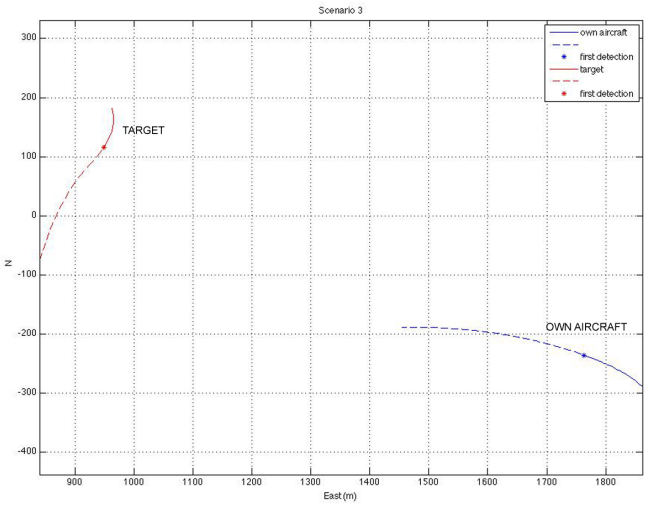


a)

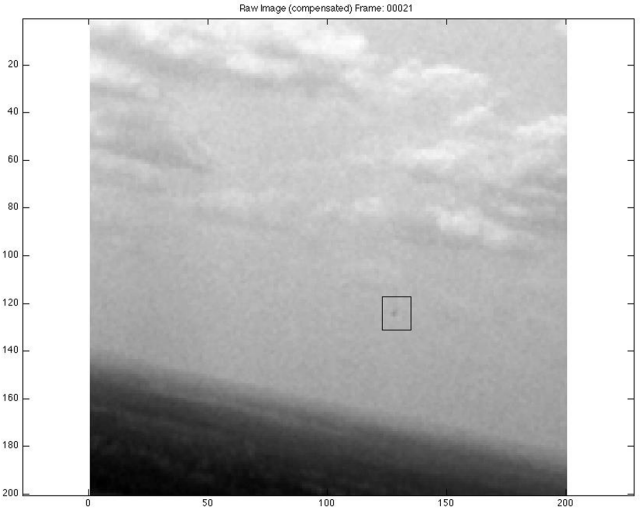


b)

Fig. 3. Encounter scenario 1. a) Show both aircraft trajectories where the asterisks denote the first detection. b) shows the region of interest of the frame with target highlighted.



a)



b)

Fig. 4. Encounter scenario 3. a) Show both aircraft trajectories where the asterisks denote the first detection. b) shows the region of interest of the frame with target highlighted.

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

[10] 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.

[11] R. Schaefer and D. Casasent, "Nonlinear optical hit-miss transform for detection," *Appl. Opt.*, pp. 3869–3882, July 1995.

[12] D. Casasent and A. Ye, "Detection filters and algorithm fusion for atr," *IEEETrans. Image Process.*, vol. 6, pp. 114–125, Jan. 1997.

[13] U. Braga-Neto, M. Choudhary, and J. Goutsias, "Automatic target detection and tracking in forward-looking infrared image sequences using morphological connected operators," *Journal of Electronic Imaging*, vol. 13, pp. 802–813, Oct. 2004.

[14] 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.

[15] Debadeepta Dey, Christopher Geyer, Sanjiv Singh, and Matt Digioia, "Passive, long-range detection of aircraft: Towards a field deployable sense and avoid system," in *Proceedings of Field and service robotics*, 2009.

[16] S. J. Davey, M. G. Rutten, and B. Cheung, "A comparison of detection performance for several track-before-detect algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, pp. 1–10, Oct. 2008.

[17] G. D. Jr. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[18] M. G. S. Bruno, "Bayesian methods for multiaspect target tracking in image sequences," *IEEE Trans. Signal Process.*, vol. 52, pp. 1848–1861, July 2004.

[19] 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.

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

[21] 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.

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

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

[24] FAA, "Pilot's role in collision avoidance," Tech. Rep. AC 90-48C, March 18 1983.