Implementation of Insect Vision Based Motion Detection Models Using a Video Camera

Andrew Budimir, Sean Correll, Sreeja Rajesh and Derek Abbott

Centre for Biomedical Engineering (CBME) and
Department of Electrical and Electronic Engineering
The University of Adelaide, SA 5005, Australia

ABSTRACT

Despite their limited information processing capabilities, insects (with brains smaller than a pinhead) are able to manoeuvre with precision through environments that are highly-crowded and contain moving objects. Their ability to avoid collisions using limited computing power forms the basis for this project, in which we attempt to simulate the motion detection ability of insects using two models – the Horridge Template Model and the Reichardt Correlation Model. In this project, the direction of motion of a moving object and its angular speed are determined by capturing visual data using a web camera focussed on a moving pattern generated by VisionEgg software. The performance of both the models is quantitatively compared and various error-reducing techniques are investigated.

Keywords: Insect Vision, Template Model, Correlation Model, Motion Detection, Bioinspired Engineering**.**

1. INTRODUCTION

Insects have highly effective visual systems that enable them to perform intricate flight maneuvers, avoiding collisions with moving objects in crowded environments (e.g. flying amongst tree branches), and reacting to visual stimuli with high speed and precision – for example, a male fly following after a female fly can adjust his course within 30 ms of the female changing her course [1]. In natural environments, they display an airborne agility which outperforms any man-made flying structures and thus are promising for aerospace and defense applications. Making their feats even more impressive is the fact that insects have a brain which is only the size of a sesame seed, more than one million times smaller than the human brain [2]. This has inspired many scientists to study the insect visual system and many models of motion detection have been proposed.

The primary aim of this paper is to improve the speed calculation methods in the Template Model [3,4], to carry out an implementation of the Correlation Model [3,4,5], and to extend the single channel detection of each model to three-channel detection using a conventional webcam prototyping platform. Thus we are able to compare the models and determine which performed better for particular metrics. The underlying goal is to measure the speed of a moving object as accurately as possible over as large a range of speeds as possible for a wide range of different visual imagery – from artificial clearly-defined shapes to more complex scenery. This is achieved by software implementation of two models that attempt to simulate the visual system of insects, particularly in the way they detect moving edges.

2. BACKGROUND

Experimentation and study into the insect visual system over many years has resulted in a better understanding of how insects so masterfully detect motion and thus avoid collisions. Several models have been developed which allow both software and hardware implementation in an attempt to simulate the process by which insects track moving object. In this paper, we have implemented the Template Model and the Correlation Model.

Further author information: send correspondence to dabbott@eleceng.adelaide.edu.au

2.1 Template Model

The Template Model has been the primary focus of this project in previous years. It was developed by G. A. Horridge in 1990 [3]. It works on the principle that a certain number of motion "templates" indicate direction of motion along a single dimensional path (e.g. leftwards or rightwards direction). These templates are derived from comparing the intensities of a pixel sampled at consecutive instances in time. So after comparing at two consecutive instances in time, any pixel intensity will be in one of three states: an increase, a decrease, or no change. These states are represented by the following symbols: *Increase* = ↑ , *Decrease* = ↓ , and *No Change*: –

When these intensities are compared for each pixel over an entire frame at consecutive time instances, there are several combinations of states which will indicate leftwards of rightwards movement [6,7,8]. This is illustrated in Figure 1:

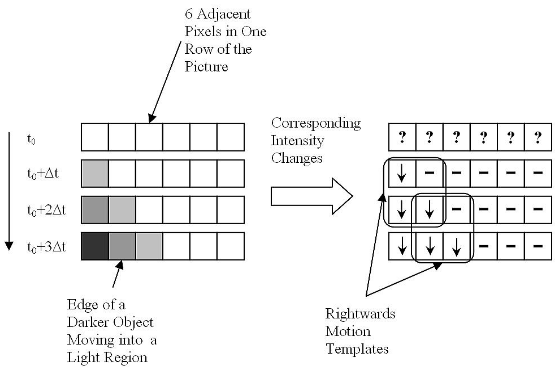
6 Adjacent Pixels in One Row of the Picture
t_0
$t_0+\Delta t$
$t_0+2\Delta t$
$t_0+3\Delta t$
Corresponding Intensity Changes
Edge of a Darker Object Moving into a Light Region
Rightwards Motion Templates

Figure 1: Dark object moving right to generate templates

This shows a particular set of six adjacent pixels as time progresses, where at time = t_0 , the pixels are all white, then at sampling intervals of Δt, an edge of a darker object progressively moves into the region. So as a moving edge enters the sampled region, by comparing the intensities at consecutive time instances for two adjacent pixels, a particular combination of these intensities is observed, and this is referred to as a "template." The template shown above is for a dark object moving into a light region. There are three other templates which indicate rightwards motion, for when a dark object moves out of a light region, or a light object moves into a dark region, or a light object moves out of a dark region. These templates are shown in Figure 2 for rightward motion and Figure 3 for leftward motion.

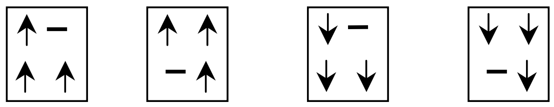

Figure 2: Templates which indicate rightwards motion

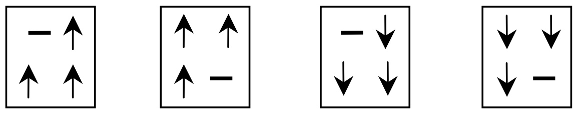

Figure 3: Templates which indicate leftwards motion

Thus there are eight templates which essentially give an indication of moving edges. By generating arrays of these templates from each set of frames obtained at two consecutive time intervals, the moving edges can be matched in any two consecutive template arrays, and thus the offset between a matched moving edge will be proportional to its speed. While the Template Model is originally based on luminance (or grayscale) information, it has been extended by Chin *et al.* [9] to include chrominance (color) information as well. By detecting changes in chrominance as well as changes in intensity, more accurate templates can be found, and thus a more reliable estimation of the motion can be made [10]. The Template Model can easily be implemented in hardware also, and multiple generations of insect vision chips based on this model have been developed by Moini *et al.* [11,12,13,14,15,16,17]. The software implementation of the Template Model is illustrated below:

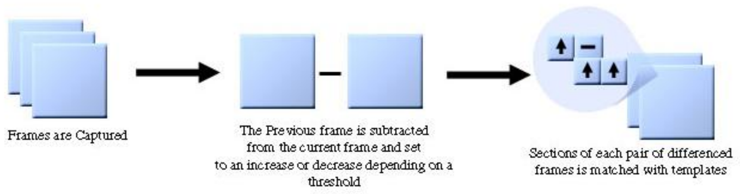
Frames are Captured
The Previous frame is subtracted from the current frame and set to an increase or decrease depending on a threshold
Sections of each pair of differenced frames is matched with templates

Figure 4: Generation of Templates in Software

So consecutive frames, containing luminance or chrominance information, are captured (e.g. from a video camera), then they are differenced to obtain an increase, decrease, or no change in the luminance (or chrominance). The differenced frames are then analyzed to find any of the eight direction-indicating templates described previously. The result is a frame which contains templates which are equivalent to the moving edges of the object.

2.2 Correlation Model

The Correlation Model, also known as the Reichardt Model, was developed by Reichardt and Hassenstein in 1956 [8]. The implementation of it in our project is based on adjacent pixels (or small, neighboring elements of a picture) being compared with one another at a previous time. So the signal level (or in our case, the luminance) of one point (or pixel) is multiplied by the signal level of an adjacent point at a delayed time. Similarly, the signal level of the second point is multiplied by the level of the first point at the delayed time. The difference of these two results then gives an indication of the angular speed in a single dimension. This process is illustrated in Figure 5:

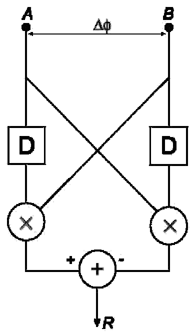

Figure 5: Correlation Model in an Analogue System

Point A represents a pixel that is adjacent to Point B. These have been sampled at two consecutive times, and the luminance of pixel A at the current time can be multiplied by the luminance of pixel B at the previous time, and then a mirror multiplication can be calculated, and the difference obtained. The Correlation Model is in theory an analogue-based model, with the rate of change of a continuous signal as the input (for spatially separated sensors, A and B), with each signal being multiplied by the delayed version of the neighboring signal to obtain a difference which gives an indication of direction. So while the analogue Correlation Model is very suited to hardware implementation, it can be modified for use in a digital system and thus fairly easily implemented in software by differencing lumance (or chrominance) values for adjacent pixels, and then performing dot products of arrays at consecutive instances in time, which is equivalent to multiplying each individual element with another element at a previous time. Thus by adjusting the offset of consecutive differenced frames, a maximum value will be encountered which indicates that the current pixels being compared are neighboring pixels. This offset will be proportional to the angular speed of the image. This process is shown in the Figure 6.

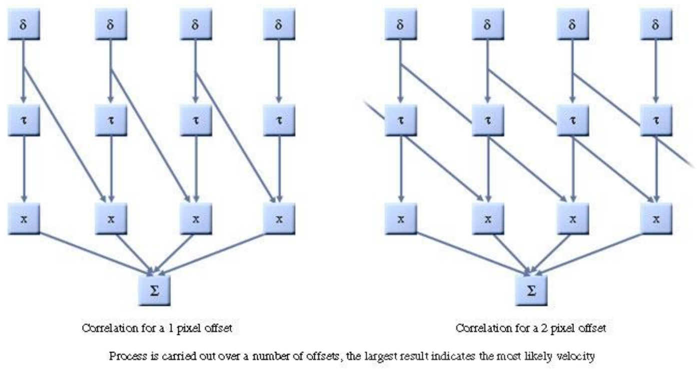
δ
τ
x
Σ
Correlation for a 1 pixel offset
Correlation for a 2 pixel offset
Process is carried out over a number of offsets, the largest result indicates the most likely velocity

Figure 6: Offsetting pixels in the Software Implementation of the Correlation Model

Correlation is performed over a defined range of offsets (which will place upper limits on the speeds which can be detected), a histogram of correlation results is obtained, with the largest sum being the peak of the histogram, which will occur at an offset which corresponds to the most likely speed. A typical histogram is in Figure 7.

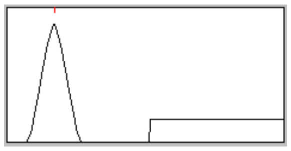

Figure 7: Typical Histogram for Correlation Model. The y-axis corresponds to the correlation sum obtained for a particular offset (the x-value).

Previously, a rotating drum with a pattern on it was used as the visual stimulus for velocity measurement. This has now been replaced by computer-generated moving patterns (VisionEgg software [16]), which have been shown to be more accurate than the original rotating drum. The Template Model was tested with a variety of different filters being applied to the captured image frames. Consequently, they obtained a set of several filters which produced more consistent results than when analyzing unfiltered frames.

3. ERROR REDUCTION TECHNIQUES

Previous approaches to reducing errors focused on pre-filtering and post-filtering of captured images for noise reduction [10]. However, we concluded that this was not the best approach to error-reduction, as the primary sources of error were not noise in the images. After obtaining some preliminary results using the software developed, we concluded that there were two primary causes of error. One was that the actual speed calculation method being used was not that effective at regularly finding correct template matches. To determine the speed, previous methods attempted to match identical templates (from consecutive frames) along a single row in the centre of the frame, giving an offset that corresponds to a speed. However, this meant that correct matches were not found very frequently (due to the limited search region), which means that this method would not be very fast in responding to changes in speed. Incorrect matches were discarded by placing upper and lower threshold values on the speed. This had the limitation of reducing the overall range over which speeds could be measured, as the maximum speed would be determined by the upper threshold, and increasing the upper threshold would result in more incorrect matches at lower speeds. More accurate template matching could be performed by increasing the search area from one row to a block of rows, but previously only a single row was used to match templates because of the other problem – frames would be lost if the speed calculation method took too long to execute. The webcam captures frames at a rate of 60 frames per second, which has been determined as the optimum rate from previous experimentation. So a time consuming speed calculation method would not return in time to analyze the next frame, meaning that a frame would be skipped, and hence inaccurate measurements would result. Our approach to this problem was to introduce a "Burst Mode." After implementing our improvements, some brief testing with filtered images confirmed that filtering did not make any improvement to our results. In some cases, we obtained the highest accuracy possible with our improvements – the accuracy being limited to a single pixel offset, as we cannot measure in fractions of a pixel. Where there were errors, we were able to identify the sources of these as hardware limitations rather than due to the effects of noise.

3.1 Burst mode

The Burst Mode enabled us to ensure that no frames were lost by capturing a set of 100 consecutive frames and then analyzing them all, rather than analyzing each frame as it is captured. Thus we do not actually perform the analysis in real-time, but essentially simulate what we would expect in real-time on a faster computer that could ensure no frames were lost. The Burst Mode could be made closer to real-time by capturing several consecutive frames (rather than 100) and then analyzing those, but there did not seem to be any value in doing this, as the purpose of the Burst Mode was to eliminate the errors obtained through dropping frames (see Section 5.1) to see how significant this problem was.

3.2 Improved speed calculation methods

In calculating the speed of the object, templates in one frame must be matched with the identical templates in the previous frame to determine how far they have moved from one frame to the next, and hence what speed the object is moving at. We tried several different methods to determine this offset between identical moving edges (templates) on consecutive frames. Of the seven "modes" which we implemented, two gave the most accurate results (Mode 4 and Mode 7), so they were the ones which we used in our testing of the Template Model. The mode used in 2003 was a pattern matching method along a single row, which is illustrated in Figure 8.

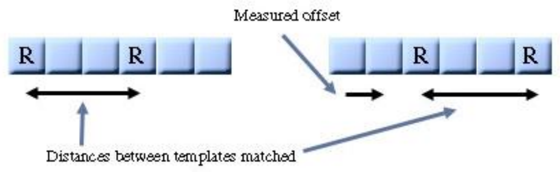

Figure 8: Pattern Matching with Templates

After generating the direction-indicating templates, a primary reference template would be found in the central row of the frame (e.g. the left-most template, "R", in the diagram above), and then a secondary reference template would be identified at a certain number of pixels away along the same row. Then this row would be analyzed in the next template frame, searching across for the primary reference template, starting at the position that this template was found at in the previous frame, and then moving along one pixel at a time. When the primary reference template is found (to ensure that it is the correct one, the secondary reference template must be located at the same relative distance away from the primary reference template), the offset from its original position gives an indication of how far the object has moved between the two frames, and hence an indication of the angular speed. Figure 9 indicates how the offset is adjusted between the two consecutive frames to determine when the correct match has been found:

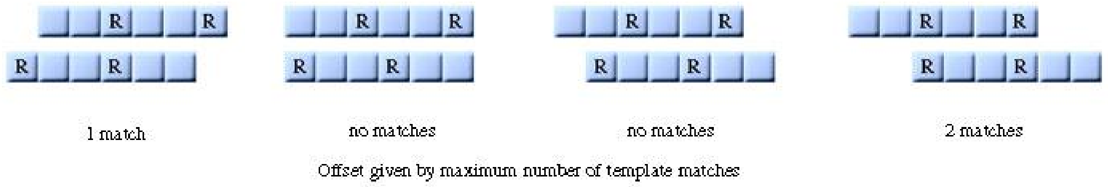

Figure 9: Calculating Velocity with Template Matching

This method had limitations in that it would not find a match very often – and hence did not respond quickly to changes in speed – and would also produce a large number of "false" matches due to noise. Because of its simplicity, matching just two templates in a single row, there is quite a high degree of error involved. In our Template Model implementation, Modes 5 and 6 are variations on this idea. Other methods used this pattern matching technique, with Mode 2 attempting to match templates along rows, using more than just a single row, and Mode 3 also offsetting templates along rows to calculate the speed, but also matching in the vertical direction, since there will be some vertical shift in the object's motion, even if it is traveling horizontally. Mode 4 attempted to match templates is shown in Figure 10.

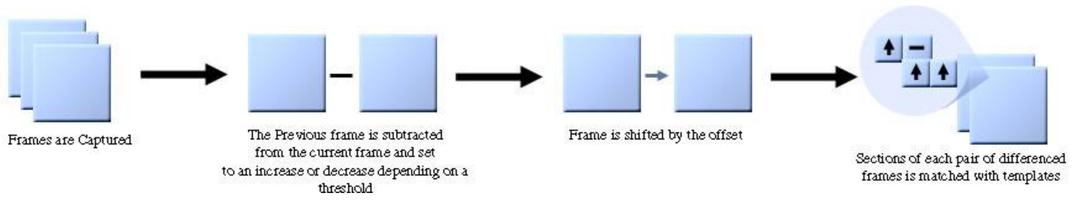
Frames are Captured
The Previous frame is subtracted from the current frame and set to an increase or decrease depending on a threshold
Frame is shifted by the offset
Sections of each pair of differenced frames is matched with templates

Figure 10: Mode 4 – Regeneration of Templates

This mode worked by regenerating templates from the differenced frames after shifting the second frame by an offset. So for each offset, a new set of templates is generated giving a direction of motion, determined by whether the majority of templates indicate rightwards motion or leftwards motion. This offset is continually shifted until the new set of generated templates indicates a change in direction. This corresponds to the second frame being shifted back one pixel further than the original frame. For example, if the object is moving to the right, the templates generated from the first and second differenced frames will indicate rightwards motion. By shifting the second frame by an offset of one pixel and then regenerating the templates, they will most likely still indicate rightwards motion (if the speed is more than one pixel/frame). So as the offset is increased for the second frame, it will eventually coincide with the original frame, so the templates generated will indicate no direction, as they will only be produced by noise. Increasing the offset by one more will generate templates that indicate leftwards motion, so when this change in direction of motion is observed, the offset minus one pixel will give a measure of the angular speed. Mode 7 worked on yet a different principle, shown in Figure 11.

For each column, the number of templates is determined, with these sums being stored in a vector. For two consecutive frames, the dot product of the two vectors is calculated for particular offsets of the second frame. The offset which results in the largest dot product will correspond to the two frames matching one another, and hence correspond to the angular speed. This method is in fact almost a mix between the Correlation and Template Models, as it uses correlation of templates to determine the angular speed.

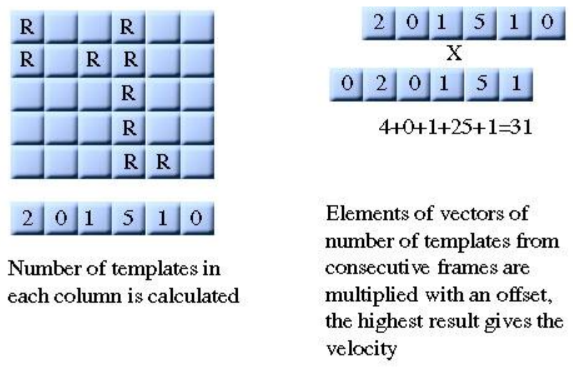
R
R
R
R
R
R
R
R
R
R
2 0 1 5 1 0
Number of templates in each column is calculated
2 0 1 5 1 0
X
0 2 0 1 5 1
4+0+1+25+1=31
Elements of vectors of number of templates from consecutive frames are multiplied with an offset, the highest result gives the velocity

Figure 11: Mode 7 – Summing Templates in a Column

4. EXPERIMENTATION, RESULTS AND DISCUSSION.

Our testing of the two models was primarily performed using two pictures. The first is a set of regular, rectangular black and white shapes with well-defined edges. Although somewhat artificial, it was used for the first round of testing, as it is an optimal scene for detecting moving edges based on luminance information, and thus was very useful in identifying and solving problems which we encountered. The black white shapes picture is shown in Figure 12.

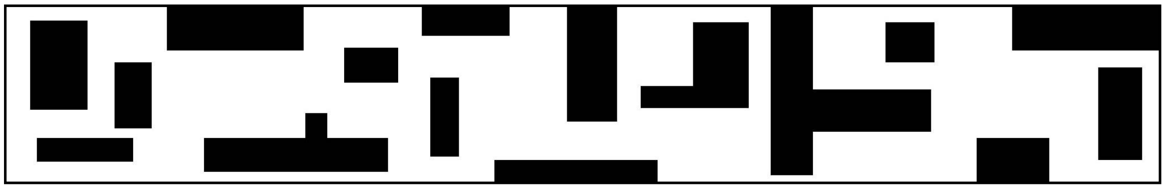

Figure 12: Black and White Shapes Picture

The other picture is a nature scene, with much less clearly defined edges, which is shown in Figure 13.

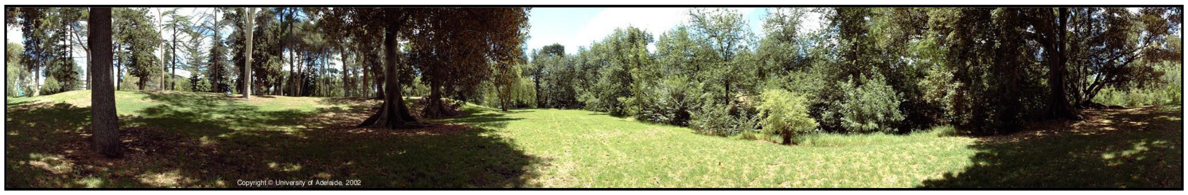

Figure 13: Nature Scene Picture

Most of the graphs are labeled with particular abbreviations to describe the mode, picture, and number of channels used for that particular test run. The key for these abbreviations is below

Key:
1-ch = Single Channel (luminous)
3-ch = Three Channels
M4 = Mode 4 used for Matching Templates
M7 = Mode 7 used for Matching Templates
BW = Black and White Shapes Scene
N = Nature Scene
BM = Burst Mode

In obtaining our results, we recorded the angular speed as pixel offsets. Consequently, a conversion factor was required to plot graphs in units of degrees/sec (these being the preferred units because that is what VisionEgg uses). This conversion factor is not fixed, since it depends on the camera's distance from the LCD monitor – the closer the camera is to the screen, the faster the object will appear to be moving. Thus we obtained the conversion factors for each test run manually by obtaining a line of best fit through the points (excluding points at higher speeds where they are no longer as accurate), obtaining the slope of this line, and then dividing the means (and standard deviations) measured by this slope. This scales them from units of pixels/frame to degrees/sec. The red dashed line on the graphs indicates the line where the y-axis and x-axis values are equal, and hence is an indication of where the measured values should lie. Error bars are plotted as $\mu \pm \sigma$, where μ is the mean of a set of 100 values of measured speed (for a particular VisionEgg speed), and σ is the standard deviation of these values.

4.1 Using the software without improved techniques

This test was performed using a Black and White shapes picture, and using the software without any of the error reduction techniques. A reasonable degree of accuracy is obtained for speeds of about 120 deg/sec to 300 deg/sec, but outside of this range there is a rapid drop-off in precision.

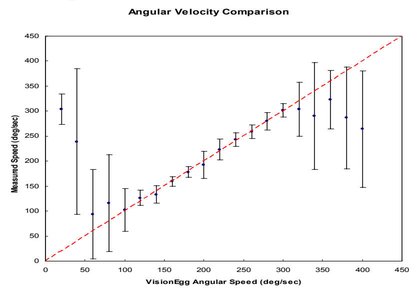

Figure 14: Testing without error reduction techniques using Black and White shapes

4.2 Template model results

Figure 15 shows the results obtained by testing the Template Model using Mode 7.

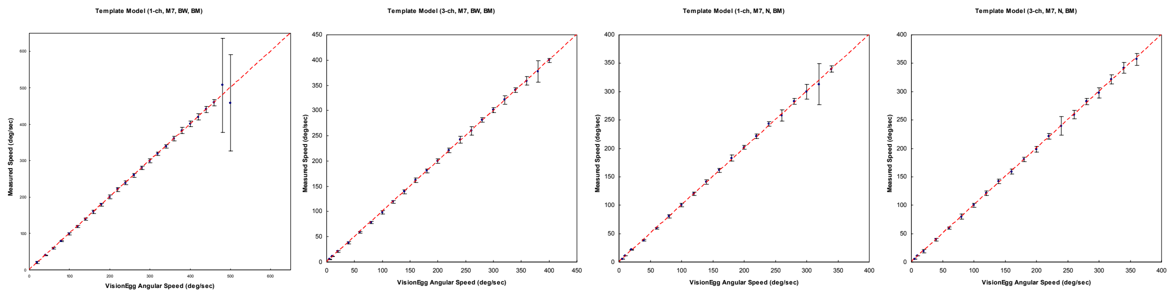

Figure 15: Testing of Template Model with Mode 7

These results indicate that a high degree of accuracy can be obtained over quite a large range of speeds using Mode 7, particularly for the Black and White shapes. With this picture, speeds as low as 5 deg/sec could be measured very accurately, and speeds up to around 460 deg/sec could also be measured accurately. Above 460 deg/sec, Mode 7 becomes very inaccurate, as the large standard deviations on the graph indicate. With the Nature scene, the upper limit of accurate speed measurement is lower, being around 300 deg/sec. This is due to the higher complexity in the picture which does not have very clearly defined edges, making this mode less effective. The graphs obtained using 3-channels indicate that this form of detection does not have a large impact on the results, although it is possible that it did increase the range of accurate detection slightly when using the Nature scene.

The results obtained with Mode 4 are shown in Figure 16. So for black and white shapes, Mode 4 can measure the speed extremely accurately up to about 450 deg/sec. Once again, this upper limit is reduced slightly for the Nature scene, down to about 400 deg/sec.

Comparing Modes 7 and 4, it appears that Mode 4 provides a higher degree of accuracy, particularly at high speeds. It can also measure speeds over a larger range for the Nature scene, reaching 400 deg/sec, compared with about 300 deg/sec for Mode 7. A primary advantage of Mode 7 (especially at lower speeds where it is very accurate) is its speed of execution – it runs 5 to 10 times faster than Mode 4 as the algorithm for Mode 7 involves much less computation. At higher speeds (above 300 deg/sec), however, Mode 7 does not find frequent matches, and thus does not respond as quickly to changes in speed, whereas Mode 4 will still respond well to changes in speed at high speeds.

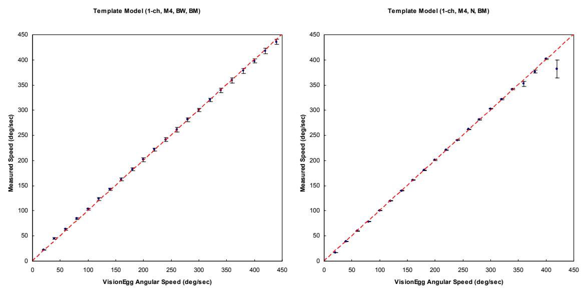

Figure 16: Testing of Template Model with Mode 4

4.3 Correlation Model results

Testing the Correlation Model gave the following results:

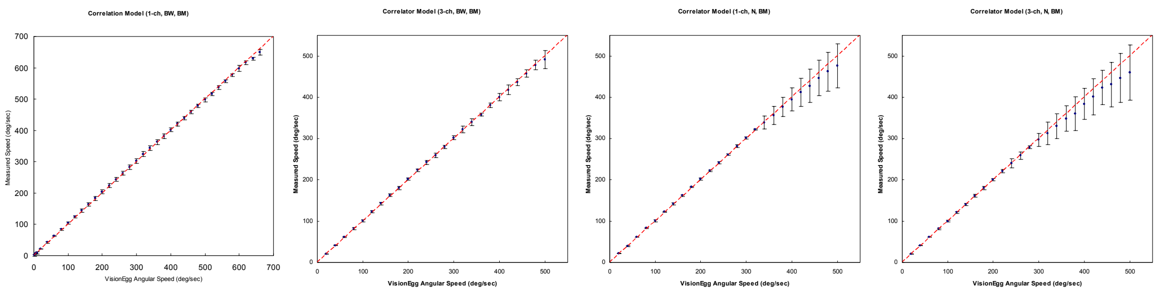

Figure 17: Testing of Correlation Model

These results indicate that the Correlation Model measures the speed very accurately over a large range of speeds, particularly for Black and White Shapes. In fact, for Black and White shapes on a single channel, the Correlation Model performed well to very high speeds (compared to any speeds which have been achieved in this project previously). The following graph indicates that it still maintained a fair degree of accuracy for speeds as high as 1000 deg/sec.

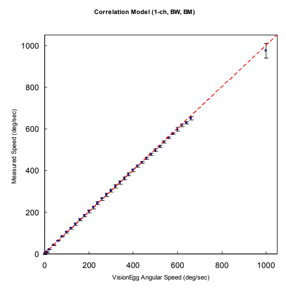

Figure 18: Correlation Model at high speeds

With the Nature scene, the Correlation Model seems to measure speeds very accurately up to about 320 deg/sec, and then the mean values drop below the ideal line, and the standard deviation increases. This is caused by a problem known as "Image Ghosting," which is discussed in Section 9.3. Once again, 3-channel detection did not have much of an impact on the results, primarily because there is enough information coming in the luminance channel to produce good results regardless, so combining it with the other two channels does not really add much information. Overall, the Correlation Model appears to be significantly more accurate than the Template Model (particularly if the "Ghosting" problem can be eliminated). One disadvantage of it is the time that it takes to execute, as when analyzing an entire frame, it is about four times slower than Mode 4 on the Template Model.

4.4 Speeding up the Correlation Model

To speed up the Correlation Model, we implemented a mode which skipped pixels to reduce the overall area which is analyzed. So a defined constant, "Detection Area" can be set to any value from slightly larger than 0.0 to 1.0, where 1.0 will result in the full frame being analyzed, and anything less will be a proportion of the full frame. To determine how small this could be, we plotted the standard deviation against the fractional detection area (as well as observing the mean values of speed), and obtained the following graphs:

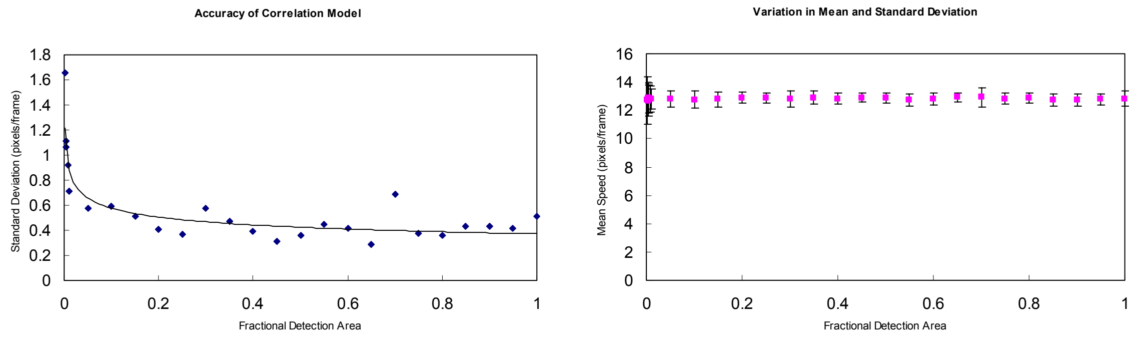

Figure 19: Effect on Mean and Standard Deviation of Changing Detection Area at 80 deg/sec

So at moderately low speeds, this indicates that the Correlation Model still works very accurately for the Detection Area parameter set to 0.05, which corresponds to about 20 times the reduction in method execution time (from when the entire frame is analyzed). The mean does not appear to be affected by the change in detection area. Performing a similar analysis at high speed (300 deg/sec) gives:

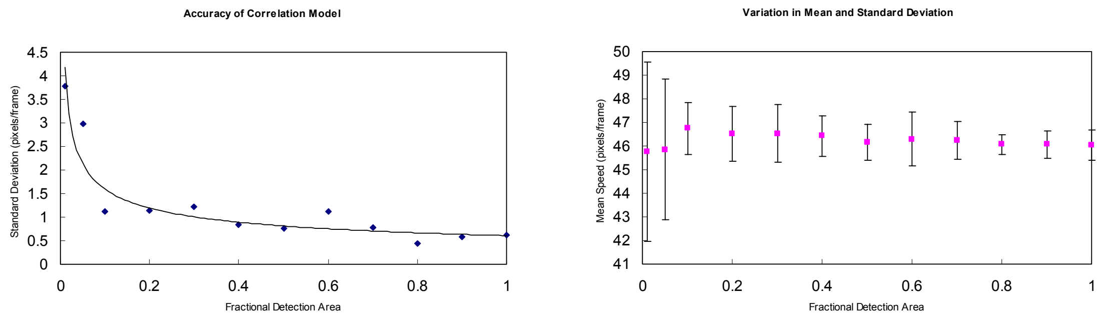

Figure 20: Effect on Mean and Standard Deviation of Changing Detection Area at 300 deg/sec

This suggests that the "Detection Area" parameter at high speeds should be a bit higher, at least 0.1, which still corresponds to a reduction in method execution time to about 1/10th of the time taken for a full frame analysis.

5. ISSUES ENCOUNTERED

5.1 Frame Dropping

The largest source of error was identified as resulting when frames are lost. As the frames are being captured by the camera at a fixed rate (in our case, 60 fps), every frame must be obtained by the software, or the calculations will no longer be meaningful (as the distances moved will be compared between frames that do not have a constant time difference). The main cause of frames being dropped was the slow processor speed and low resources of the primary PC. To ensure that all consecutive frames were captured, we introduced a "Burst Mode," to capture a set of frames first, and then analyze them. The effect of frames being dropped on the graphical output is shown below (the two spikes):

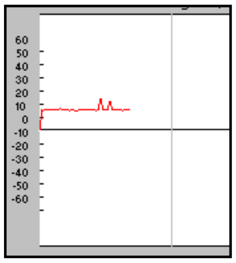

Figure 21: Spikes resulting from Dropping Frames. This is a screenshot which shows Angular Speed vs. Time

5.2 VisionEgg Jitter

Occasionally VisionEgg would not run smoothly, as the picture would move in "jerks," resulting in incorrect speed measurements. This seemed to occur when some other process was using a large percentage of the CPU resources, or if the PC had been left on for a long time and a large percentage of memory was being used (possibly due to memory leaks). So this problem was resolved by restarting the PC and not running too many other processes at the same time as VisionEgg.

5.3 Image Ghosting

This was the primary problem encountered with the Correlation Model at high speeds when using the Nature scene. This is probably due to the exposure time of the camera being too long and due to latency in the LCD display. At low speeds, the Image Ghosting does not create problems because it appears as motion blur. However, at high speeds, the multiple frames separate enough to be distinct, and as a result, there are several peaks which result from the Correlation Model histogram (with a peak also being obtained when the image matches up with the ghosted image). The ghosting effect is shown in Figure 22:

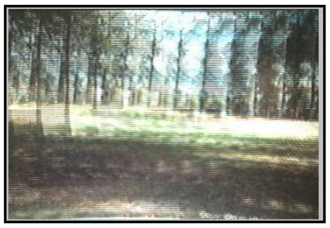

Figure 22: Image Ghosting

The resulting histogram from the Correlation model is shown below (on the left) and compared with the normal histogram (on the right) obtained when speeds are low enough for ghosting not to be a problem:

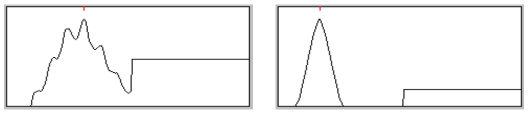

Figure 23 (left): Histogram when Image Ghosting is occurring (far left) and normal histogram (right). The y-axis corresponds to the Correlation sum for particular offsets (the x-values).

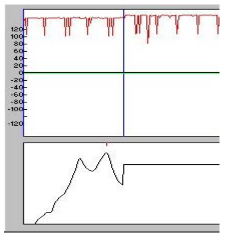

Figure 24 (right): Spikes resulting from Image Ghosting. This is a screenshot showing Angular Speed vs. Time (upper graph) and a Correlation Histogram (lower graph).

So occasionally one of the secondary peaks will end up being larger than the primary peak, resulting in about a 20% drop in the speed, which is shown on the graphical output:

This is why the mean values of speed calculated for the Correlation Model at high speeds (with the Nature scene) drop below the ideal line on our results (and obviously the standard deviation will increase). Due to shortage of time, we did not find a solution to this problem, although it is most likely that a camera with a shorter exposure time or lower latency on the LCD would correct it, or it might be worth investigating Motion Blurring on VisionEgg.

5.4 Frame Ripping

Occasionally we obtained patches on the graphical output which had a large number of glitches. We discovered that these were caused by frame ripping, an example of which is shown below:

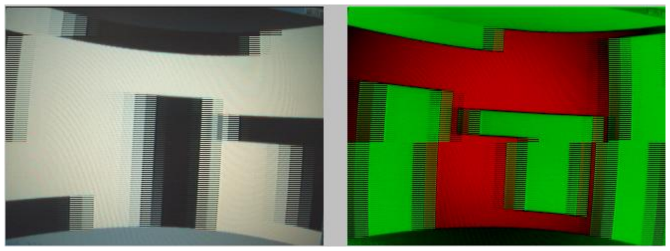

Figure 25: Frame Ripping

This appears to be because of a lack of synchronization between the camera feed and the capturing of the frames. We did not find an effective solution to this, and because it did not impact the results very frequently, we were able to ignore it for the purpose of obtaining results. A new camera with an ability to synchronize frame capture with the software would most likely fix this problem.

5.5 Initialization Issue

When the program is loaded, the first set of frame captures (in Burst Mode) are usually interrupted with other program activity, meaning that frames are lost and the results for the first set of 100 frames are not good at all and have to be discarded. We believe this is probably caused by memory allocation in the first stage of frame capture which should probably be performed as an initialization routine when the program is attempt to rectify it.

5.6 Screen Reflection

The group identified screen reflections as a significant problem in 2003, as reflections from light sources in the room would result in bright patches on the captured image, hiding some of the edges and making it more difficult to analyze. Their solution was to place a shield over the entire setup and thus remove any effects of other light sources. We found that we could eliminate this problem by adjusting the angle of the LCD monitor such that it no longer reflected light directly into the camera.

5.7 Camera Contrast Adjustment

For scenes with large changes in contrast (such as the Nature scene, which has regions which are very light, and regions which are very shadowy), the camera attempts to auto-adjust the contrast, but cannot respond quickly enough to the moving pattern, and ends up making large bright patches on the captured image which fade in and out. In general, this did not have a significant effect on the accuracy of speed measurement. However, upgrading the camera would probably solve this problem.

6. CONCLUSIONS

We identified several ways to significantly improve the accuracy of speed measurements. One of these was to ensure that a set of consecutive frames were captured from the video camera, as significant errors were resulting from frames being lost. Another was to increase the accuracy of pattern matching in the Template Model using better matching techniques. Consequently, we obtained results which were far more accurate (having lower standard deviations) than obtained in previous years, and also were able to measure speed to much higher values, achieving a high accuracy for speeds of as high as 700 deg/sec for the Correlation Model (which we implemented as a new feature in this project), and speeds as high as 450 deg/sec for the Template Model (this is compared to speeds of about 300 deg/sec obtained in 2003). For lower speeds, we obtained highly accurate measurements for as low as 5 deg/sec using the Template Model. We were also able to obtain much better responses to change in speed than in previous years (as correct matches were found much more frequently). In general, the Correlation Model performs better than the Template Model, as the results indicate that it gives better accuracy and can measure speeds over a larger range. However, it is slower in analyzing the frames (although various ways of improving the speed of method execution have been investigated with some degree of success). Also, for nature scenes it does not perform as well at higher speeds due to the image ghosting effect. So significant improvements in the accuracy and range of speed measurements have been made by implementing the Template and Correlation Models in this project. So how do our results compare with the insects upon which the models are based. Flies can detect motion at up to 2000 deg/sec! And the best insects can detect speeds as low as 0.1 deg/sec. That corresponds to 2 - 3 times faster than our highest speed, and 50 times slower than our lowest speed.

7. ACKNOWLEDGEMENTS

Funding from the Sir Ross & Sir Keith Smith Fund and Australian Research Council is gratefully acknowledged. Acknowledgement goes to the following people for their assistance and past contributions, David O'Carroll, Andrew Straw, Ben Keipert and Yi Fei and Hung Nguyen.

8. REFERENCES

1. C. Wieland, "Why a fly can fly like a fly," *Creation Ex Nihilo Technical Journal*, **Vol. 12,** Issue 3, pp. 260-261, 1998.
2. M. Lehrer, M. V. Srinivasan, S. W. Zhang and G. A. Horridge, "Motion cues provide the bee's visual world with a three dimension," *Nature,* **Vol. 332***,* pp.356-357, 1998.
3. G. A. Horridge, "A template theory to relate visual processing," *Proc. of the Royal Society of London B,* **Vol. 239**, pp. 17-33, 1990.
4. P. Sobey, "Determining range information from self motion – the template model," *Proc. SPIE*, **Vol. 1382,** Intelligent Robots and Computer Vision, pp. 123-131, 1990.
5. B. Hassenstein and W. Reichardt, "Structure of a mechanism of perception of optical movement", *Proc. of the 1st International Conference on cybernetics*, pp. 797-801, 1956.
6. X. T. Nguyen, Smart *VLSI Micro-Sensors for Velocity Estimation Inspired by Insect Vision, Ph.D.* Thesis, EEE Dept., University of Adelaide, 1995.
7. X. T. Nguyen, A.Bouzerdoum, R. E. Bogner, A. Moini, K. Eshraghian and D. Abbott, "The stair-step tracking algorithm for velocity estimation," *Proc. Australian New Zealand Conference on Intelligent Information Systems*, pp. 412-416, Perth, Australia, December 1993.
8. X. T. Nguyen, A. Bouzerdoum, R. E. Bogner, A. Moini and K. Eshraghian, "Feature Representation of motion Trajectories," *Proc. IEEE Int. Conf. Neural Networks*, **Vol. 6**, pp. 2922-2927, Perth, Australia, 27 November-1 December.
9. K. Chin and D. Abbott, "Motion Detection Using Colour Templates," *Proc. SPIE*, **Vol. 3893**, pp. 314-321, 1999.
10. H. X. Nguyen, S. Rajesh and D. Abbott, " Motion detection algorithms using template model", *Proc. of the SPIE Electronics and structures for MEMS*, **Vol. 4591,** pp. 78-90, 2001.
11. A. Moini, A. Bouzerdoum, A. Yakovleff and K. Eshraghian, "A two dimensional motion detector based on insect vision," *Advanced Focal Plane Arrays and Electronic Cameras*, Berlin, Germany, pp. 146-157, 1996.
12. A. Moini, A. Bouzerdoum, K. Eshraghian, A. Yakovleff, X. T. Nguyen, A. Blanksby, R. Beare, D. Abbott and R. E. Bogner, "An Insect Vision Based Motion Detection Chip," IEEE *Journal of Solid State Circuits*, **Vol. 32**, No.2, pp. 279-284, 1997.
13. A. Moini and A. Bouzerdoum, "A Biologically motivated Imager and Motion Detector with Pixel Level Image Processing," *Proc. Australian Microelectronics Conference*, Melbourne, pp. 180-185, 1997.
14. A. Moini, A. Bouzerdoum, A. Yakovleff, D. Abbott, O. Kim, K. Eshraghian, and R. E. Bogner, "An analog implementation of early visual processing in insects," *Proc. 1993 Int. Symp. VLSI Technology, Systems, and Applications*, Taipei, Taiwan, May 1993, pp. 283–287.
15. A. Yakovleff and A. Moini, "Motion perception using analog VLSI," Journal of Analog Integrated Circuits and Signal Processing 2, pp. 1-22, 1997.
16. D. Abbott, A. Moini, A. Yakovleff, X. T. Nguyen, A. Blanksby, G. Kim, R. E. Bogner and K. Eshraghian, "A new VLSI smart sensor for collision avoidance inspired by insect vision," *Proc. SPIE, Intelligent Vehicle Highway Systems*, Boston, **Vol. 2344**, pp. 105-115, 1994.
17. A. Yakovleff, A. Moini, A. Bouzerdoum, X. T. Nguyen, R. E. Bogner, K. Eshraghian and D. Abbott, "A micro-sensor based on insect vision", *Proc. Computer Architecture for Machine Perception Workshop (CAMP'93),* New Orleans, USA, pp. 137-146, 1993.