7 Debris Detection Program Details

Introduction

This chapter covers the Debris Detection (DDT) program starting with a broad overview of the program idea, followed by an in depth discussion.

7.1 Program Initial Ideas

Faced with the data format produced by the Herstmonceux camera system detailed in chapter 6, recognition of debris by analysis of the debris image in one frame would be extremely difficult. Stars, noise pixels and any debris that survived the thresholding process would all look alike - no information about the origin of the centroids is stored from frame to frame. Effectively the processed data is just a list of positions of centroids, or “dots”. Many ideas were thought of initially and rejected; e.g. fast fourier transforms would work well if bright debris were the only target for the algorithm; for faint debris however, the threshold level of the system would be necessarily set to a low SNR and in such a case, the faint spike caused by a regularly moving object would be swamped by the noise.

7.2 Program Final Form

Given a sequence of noisy frames consisting of randomly positioned dots, one intuitive method for detecting a debris object’s image would be to examine the frame series for an object moving with uniform speed in a straight line in the image plane (any slight warping effects caused by the system optics distorting a straight line into a curve could be modelled mathematically and incorporated into the detection system). The probability of a straight track of regularly spaced dots being due to random noise would decrease with the increasing number of dots forming that track, and therefore increase the probability of the track being due to a real moving object in the FOV. A probability threshold would eventually be crossed, above which the track could be reasonably considered as being real.

This technique is similar to the Particle Imaging Velocimetry (PDV) and Particle Displacement Tracking (PDT) techniques as outlined by Wernet and Edwards (1990),

Wernet (1991), and Wernet and Pline (1993), but has to operate in a different regime in terms of the speed and number density of objects within the field of view. In this case, the debris images always enter the field of view from its edge, cross the field of view at an arbitrary angle, and exit at a different edge. That is, no debris tracks originate and/or terminate within the field of view, and because of this, there are several extra subtleties to be considered when adapting this method to the case of space debris identification.

The philosophy behind this program was therefore to sort through frames entering the front end of the system and detect any dots amongst the noise that appeared to be moving in a regular manner across the FOV.

Debris could enter the FOV at any point in the frame stream coming from the front end. For this reason the most recent N frames are analysed in a moving bracket fashion (Figure 7.1). In the forthcoming discussion, frames analysed in order in the moving bracket will be referred to as “frame 1”, “frame 2”, etc. The numbering refers only to the position within the moving bracket, which will be different from its position in the entire night’s stream of data.

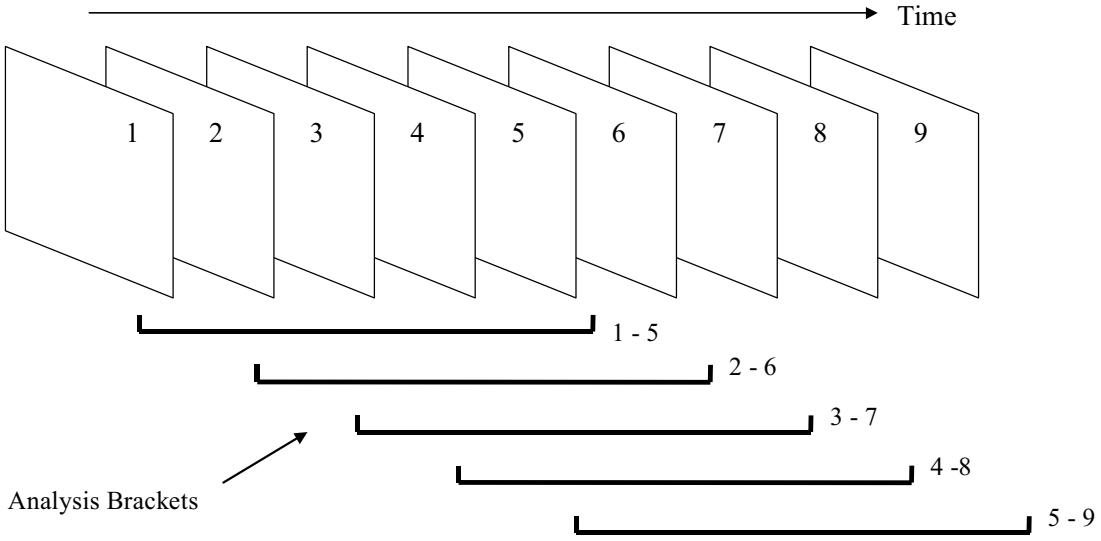


Figure 7.1: Diagram illustrating the moving bracket principle of frame analysis for a 5-frame bracket.

Debris would obviously enter the FOV from its edge, moving further from the edge with time. For this reason there is no point searching the middle of frame 1 for the beginning of a possible debris stream. Rather, only an “active border” sensitive to new

debris entering the FOV around the edge of the frame need be examined (see Figure 7.2 and Figure 7.3). The width of this border corresponds to twice the maximum apparent angular velocity of debris expected, being the diameter of a circular "search area" employed in frame 2 to try and associate dots within it there to dots in frame 1 within the active border.

A candidate debris stream starts with a dot selected in this active border region (Figure 7.2). A circular search area centred on the same x,y position of the first dot is then scanned in frame 2 to produce pairs of dots that may or may not be the first two dots of a debris track across the FOV. The search area is initially circular (modifications to the initial shape are outlined later on), its radius being the maximum apparent angular velocity of space debris the search program will be sensitive to.

A frame 2 dot within this radius (i.e. "near" to the frame 1 dot) could be the next in a possible debris track. Its focal plane displacement relative to the first dot is noted and used to extrapolate forwards into frame 3 to create a "predicted position" (PP), another small search area with a radius defined to account for any offsets in the centroiding algorithm.

This PP is then scanned to determine if the third in this possible debris stream is present. If not, the scan returns to the next dot in the search area in frame 2 and another stream interrogated. If a third dot is present, the chances of it being a real debris track are more likely, another PP produced and a fourth frame examined, and so on recursively until the desired accuracy is reached.

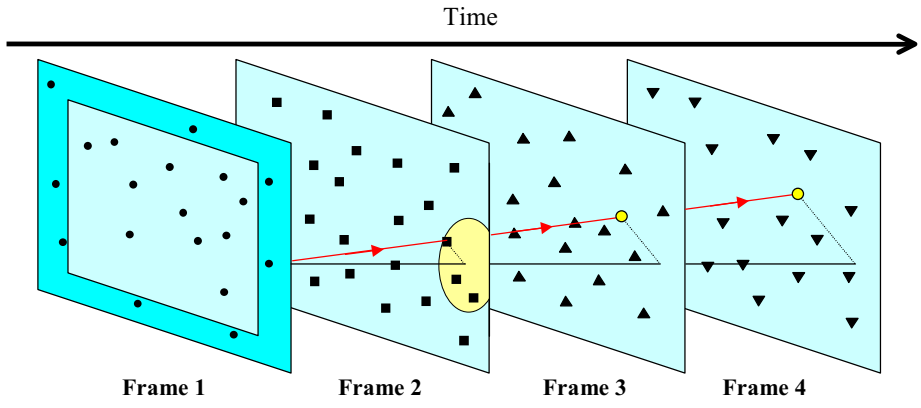


Figure 7.2: Illustration of the basis behind the search algorithm. A pair of dots are extrapolated into two or more frames by creating predicted positions in those frames. Those small areas are then scanned to see if any further dots exist. If so, the probability of those dots being the track of a real object passing through the FOV is high.

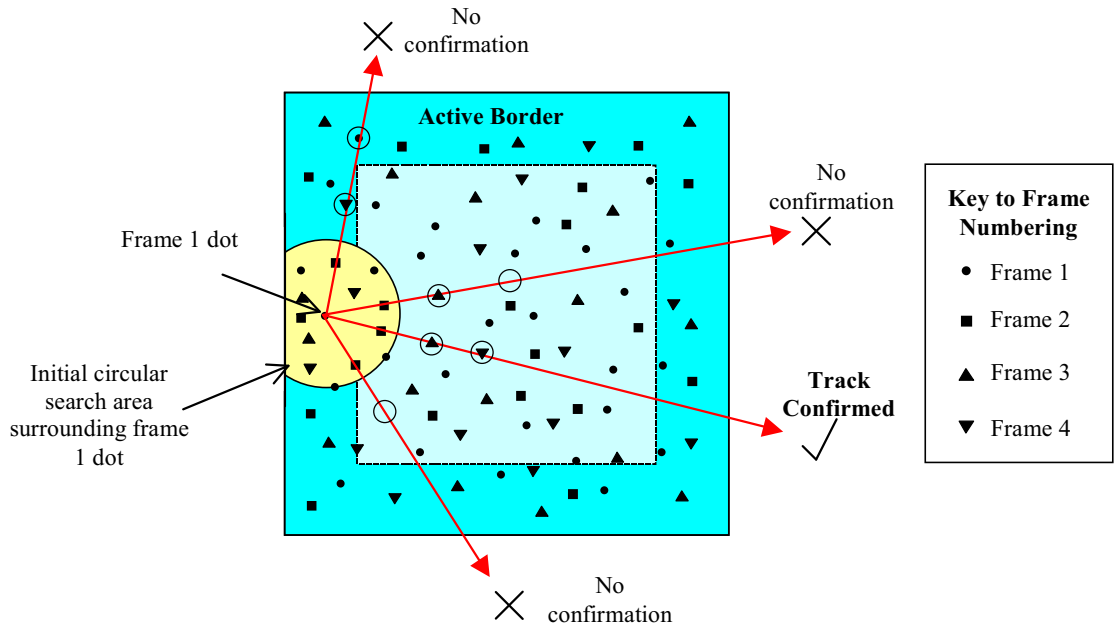


Figure 7.3: View looking down the axis of the space-time "corridor" of Figure 7.2, to illustrate the time independent methodology of the search algorithm, that of sorting through a field of coded dots to confirm or refute the presence of a particle's track.

7.3 Search Area & Predicted Position Theory

7.3.1 Reason for Search Area

It is not necessary to pair every frame 2 object with every frame 1 object. Owing to the deliberately short frame interval, a debris object will only have had time to move partway across the FOV between frames. For a given frame interval t_f and maximum expected topocentric angular velocity ω_{max}, the maximum distance R moved across the FOV between frames therefore defines a "search radius", R_S, beyond which it is pointless to make pairs. For this general case, there is therefore a circular "search area", A_S, for every frame 1 object around its corresponding point in frame 2 (Figure 7.4) The picture is not as simple as this in practice however, for reasons given in section 7.3.3.

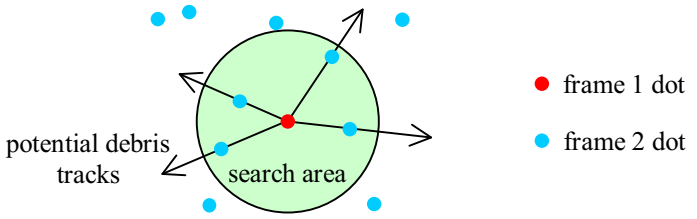


Figure 7.4: Introducing the concept of the search area placed around each frame 1 dot analysed. Only those frame 2 dots within the search area are considered for pairing.

As the search area radii are defined by the maximum angular displacement of the debris between frames, then if N dots per line are required to minimise ambiguity, the search area radius is therefore given by:

$$R_S = \frac{L}{(N-1)} \tag{7.1}$$

where L = length of side of CCD frame (a square CCD will be assumed throughout).

7.3.2 Search Philosophy & Creation Of Active Border

As introduced in section 7.2, once the observing run is under way, any new debris object entering the FOV will enter at the frame edge and move away from that edge into the frame. This highlights the region around the edge of the frame as sensitive to the formation of new debris tracks. The shape and maximum extent of this sensitive region into the frame can be obtained by considering the path of a debris particle moving at maximum considered angular velocity, in a direction perpendicular to the frame edge. If in frame 0 its image was an infinitesimal distance outside the frame, then in frame 1 it would be at a distance R_S into the frame and by frame 2, at $2R_S$ (Figure 7.5). The maximum width of the Active Border then is R_S, extending into the frame as a rectangular border around the frame edge. This defines the maximum intrusion region a new debris particle can make into the frame, and the search for all possible new debris tracks must be concentrated within the active border. This has the beneficial effect of reducing the number of image objects to make pairs with since the central remainder of the frame is not used for pair production.

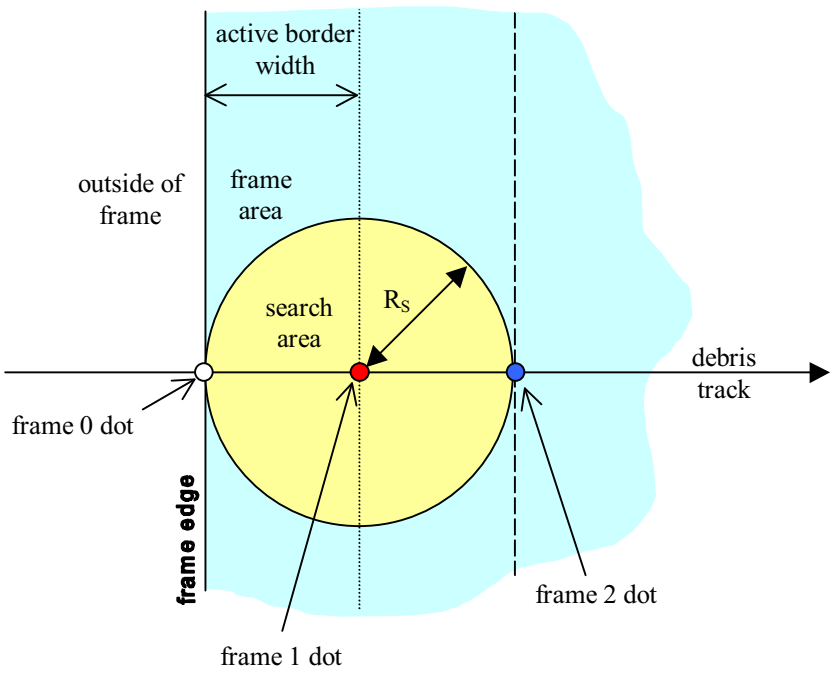


Figure 7.5: Positioning of frame 2 search area around the frame 1 dot and its relation to the active border.

7.3.3 Search Area Geometry

In the case of the perpendicular travelling debris in Figure 7.5, the only place necessary to look for candidate frame 2 objects to pair with the frame 1 object would be at the exact point a distance $2R_S$ into the frame, in line with the frame 1 object. In this case the search area is not a circle but a point at a specific spot in relation to the frame 1 object. The number of frame 2 objects lying within this search area would be small, and the number of pairs produced small in consequence.

The search area for debris moving slower than the maximum angular velocity is more complex in nature. Consider a debris object in the image frame moving at an angular velocity such that it moves a distance $r < R_S$ in t_f, and in frame 1 its image is a distance $d < r$ from the frame edge. Initially one wouldn't know its angular velocity, so a circular search area of radius Rs is tentatively put around its corresponding position in frame 2 at first (dashed circle around red frame 1 dot in Figure 7.6).

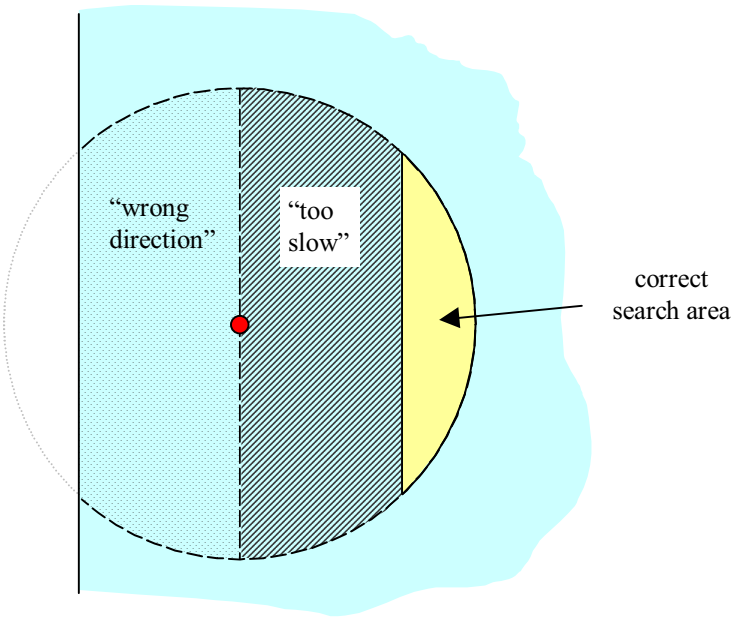


Figure 7.6: Geometry of search area once geometric factors take effect.

All frame 2 objects lying within R_s, but to the left of a line parallel to the edge running through the frame 1 position can't be candidates for pairing, because their consideration would imply that the object was moving out of the frame. This in turn implies that the object has spent some time previous to the current frame traversing the field of view, and so should have been detected earlier on. Therefore it cannot be part of a new debris track and that zone can therefore be ignored (dotted zone in Figure 7.6).

All frame 2 objects lying within the area delineated by the first parallel, the search area boundary, and a second line parallel to the first and the same distance in from the edge (striped zone in Figure 7.6) cannot be considered for pairing either. This is because their inclusion would imply that they would move a distance less than d in the interval t_f and would therefore be possible first, not second, dots in new debris tracks, if debris objects they are. These objects in effect are moving "too slow" to be considered. The only frame 2 objects worthy of consideration then lie in the remaining segment of the circular search area (yellow segment in Figure 7.6), it being the real search area for this case.

Another way of determining the real search area is to think that by definition the

frame 0 object accompanying the frame 1 object (for which the search area has been constructed) must have been outside the frame during integration of frame 0. Within the bounds of Rs, the segment of the circular search area outside the frame must therefore contain all possible positions of the object in frame 0. The area denoting the only places the object can be in frame 2 therefore is that obtained by mapping the frame 0 area through the frame 1 position, giving the same area described in Figure 7.6 (Figure 7.7).

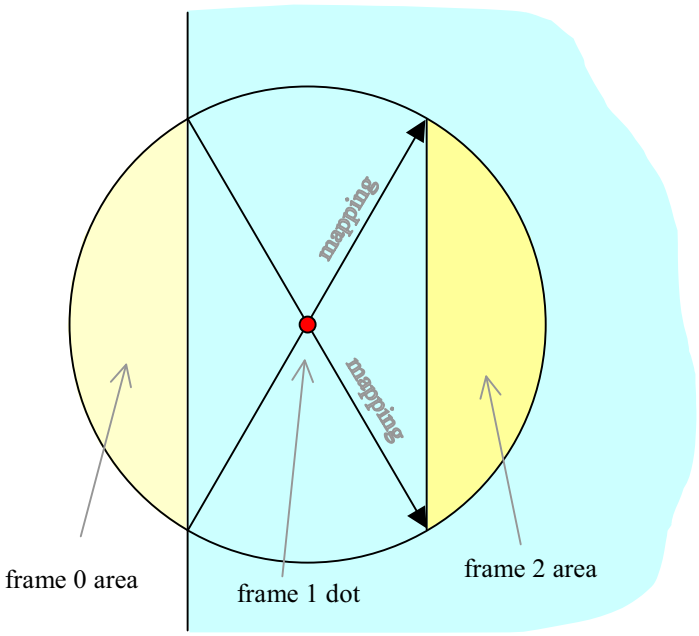
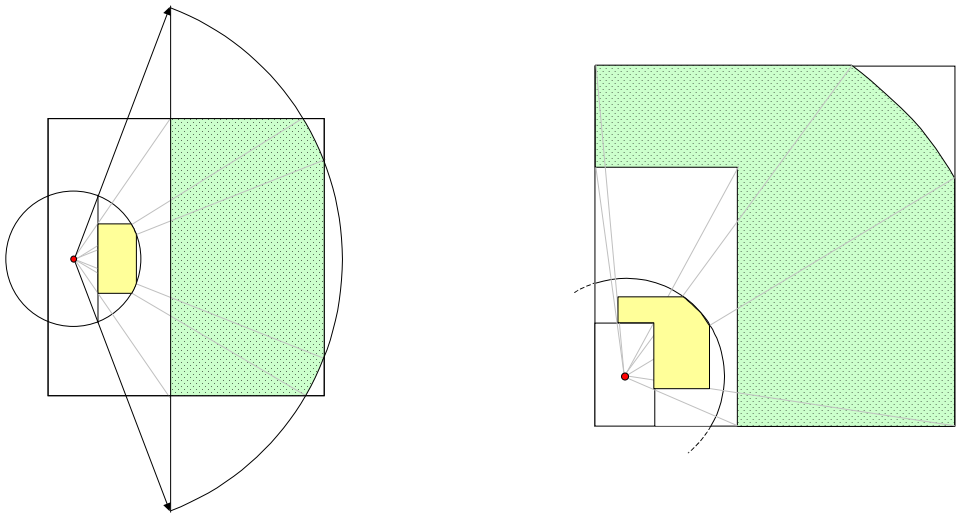


Figure 7.7: Mapping geometry of range of possible positions of previous frame's dot position, through frame 1 dot, into frame 2.

A further geometric effect to be taken into account however is to consider a debris track that would leave the FOV before the requisite number of dots in the track have been confirmed, determined by position of the frame 1 dot. For tracks that would overshoot the FOV, there is no point creating line pairs in the search area. Thus the search area may be truncated to imitate the edge of the frame (Figure 7.8).



(a) (b)

Figure 7.8: (a) Edge truncation effect on frame 2 search area. The initial segment of the search area shown in Figure 7.7 is truncated in a pattern (pale yellow) matching the overlap of the frame and the "end zone", the region where all possible debris tracks would finish (striped). Grey lines show the match from the search area to the end zone. (b) Corner geometry effect on the search area. The initial search area, neglecting truncation, is shown uncoloured with a solid border. Superimposed on top of this is the truncated search area (pale yellow), which matches the end zone (striped).

It can be seen that no matter the configuration (sides or corners), the search area decreases the further the frame 1 object is from the frame edge, until at a distance $d = R_S$ the search area is vanishingly small and for $d > Rs$, is zero. Thus only a region, or "active border" around the edge of the frame need be analysed. The effective search area of the Active Border therefore is not simply the area of the whole frame minus the area of the central unused part, but some more complex function that decreases with increased "depth" into the frame, thus being less than a simple subtraction of areas.

7.3.4 Predicted Positions

The predicted positions that are projected forwards in the time stream from any paired frame 1 and 2 dots are extrapolated from the relative vector between them, and have a finite size to allow for uncertainties during centroiding caused by atmospheric blurring. The predicted position generating and scanning subroutine in the program is accessed again recursively each time a successful "hit" is detected until the desired probability threshold mentioned earlier is surpassed. At this point a real debris track is deemed to have been detected and its track attributes noted. The search algorithm then

retraces its steps to the last branch point and continues the search. Thus the algorithm searches all possible debris tracks sequentially, but not all the dots in the bracket are examined.

7.4 Extra considerations

7.4.1 Star Background

If the telescope is fixed with respect to the horizon, stars will drift across the FOV. Given the location of the observing site and the pointing direction of the camera, the magnitude ω_{SD} and direction θ_{SD} of sidereal motion with respect to an altazimuth grid (see Figure 7.9) can be calculated as:

$$\omega_{SD} = \frac{2\pi}{T_{SD}} \left[1 - \left(\sin a \sin \phi + \cos a \cos \phi \cos A\right)^2\right]^{\frac{1}{2}} \text{ rad s}^{-1} \qquad (7.2)$$

$$\& \quad \sin \theta_{SD} = \cos \phi \sin A \left[1 - \left(\sin a \sin \phi + \cos a \cos \phi \cos A\right)^2\right]^{-\frac{1}{2}} \text{ rad}, \qquad (7.3)$$

where a = Altitude (radians above horizon), A = Azimuth (radians), ϕ = topocentric latitude (radians), T_{SD} = length of one sidereal day (86164.091 s).

From equations (7.2) and (7.3) the expected motion of stars through the FOV can be calculated. Given the small size of the FOV, fluctuation of the sidereal vector through the FOV is minimal. Hence if a pair of dots from frames 1 and 2 yield a displacement vector matching the sidereal vector, they could be flagged to be the first two probable dots of a star track. Further frames would be used to confirm this, then all further points across the FOV would be calculated to the point where the star leaves the FOV, in order that they are not included in the search for debris.

An alternative method of locating stars in the FOV could be to use a lookup table of star positions to determine where and at what time stars would enter and leave the FOV. If however the telescope was siderostatic (tracking star motion), stars would not drift through the FOV at all, and the same lookup table could be used to mask out the stationary stars from the image (i.e. direct the computer to ignore those centroids).

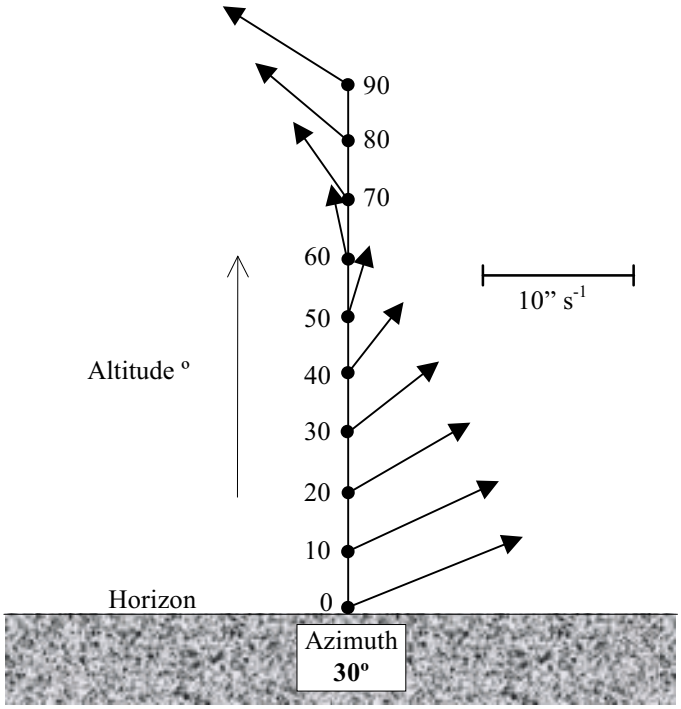


Figure 7.9: Example of variance of sidereal motion vector with altitude. The observer is located at 50º North latitude, looking at azimuth 30º. The vector arrows are drawn to scale.

7.4.2 Noise (camera/sky - twilight)

The point of the investigation is to detect small and therefore faint debris, which necessitates thresholding close to the noise levels. It helps therefore to keep noise levels to a minimum. This problem can be addressed by cooling the detector to liquid nitrogen temperatures to reduce electron shot noise, and ensuring good manufacture of the CCD to minimise 1/f noise (Mclean, 1989).

7.5 Implications on minimum height detectable

Given that the minimum number of dots to make an unambiguous straight line is three, this puts constraints on the maximum topocentric angular speed ω_{top} a piece of debris can have when passing through the FOV. Too fast, and the debris will be out of the FOV again before the system can take three images. The parameters controlling this are therefore the platescale and the size of the CCD (which give the FOV), the duty cycle time T_{cyc} which determines how many exposures can be taken in a given time, and the angular speed of the debris, ω_{top}.

It can be seen therefore that the maximum value of ω_{top} to ensure N dots in a line is given by:

$$\omega_{top_{max}} = \frac{FOV}{T_{cyc}(N-1)} \tag{7.4}$$

which for N=3,

$$= \frac{FOV}{2T_{cyc}} \tag{7.5}$$

.

Equation (7.5) therefore dictates the maximum ω_{top} detectable for the system given. In chapter 6 the relation between ω_{top} and height for tracking and non-tracking systems was reproduced; equation (7.5) can therefore be related to a minimum (circular orbit) height using the information in chapter 6. For example, consider a telescope system with a FOV of 0.5º and a cycle time T_{cyc} of 1s. The maximum possible ω_{top} is therefore:

$$\omega_{top_{max}} = \frac{1800''}{2 \times 0.5} = 1800'' s^{-1}, \tag{7.6}$$

which by Figure 6.2 implies a height of ≈ 800km for both tracking and non-tracking systems.

7.6 Implementation

7.6.1 Choice of C over Fortran

C was chosen as the programming language for the DDT algorithm for two reasons:

1. The software for the camera "front end" system at Herstmonceux was written in C, so it was deemed wise to maintain compatibility.
2. The algorithm lends itself readily to a recursive approach when progressing down the lines of inquiry of a debris stream, as after the initial pair production using the dots of frames 1 & 2, the calculation and production of predicted positions is repeated for however many times are necessary. Recursion is a feature supported by C, but not by Fortran 77.

7.6.2 Picture data I/O

A cyclic frame counter is adopted to control the order in which frames are analysed within the analysis bracket. It would be time-consuming to completely reload all

pertinent frames into the bracket in a different order, so instead the newest picture overwrites only one bracket frame at a time, and analysis simply follows a different order each time (Figure 7.10).

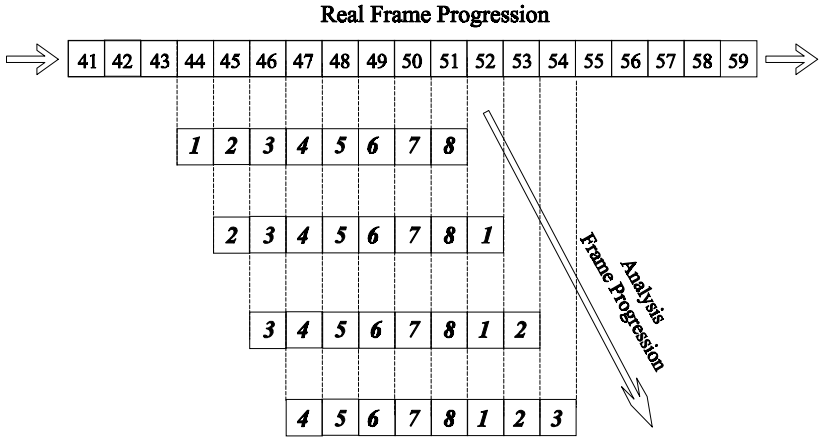


Figure 7.10: Illustration of how the moving bracket corresponds to the absolute picture progression.

7.6.3 Search Area Scanning

Each picture's data is read into the corresponding frame in the moving bracket as per the cyclic counter. The moving bracket consists of a block of memory the same size as the total number of pixels for the number of frames in the bracket (Figure 7.11), e.g. 10 frames corresponding to images from a 1024×1024 pixel CCD is approx. 10.5×10^6 memory locations. The memory is allocated as RAM at the beginning of the observing run once only, and filled with zeros automatically (using the C "calloc" function).

Rather than use arrays for this purpose, the memory is created as a block which is accessed using pointers. This has the advantage that access of the memory is faster; the alternative, of using index numbers to refer to elements of an array, is a relatively high level feature of C and therefore requires more time (Seigel, 1989).

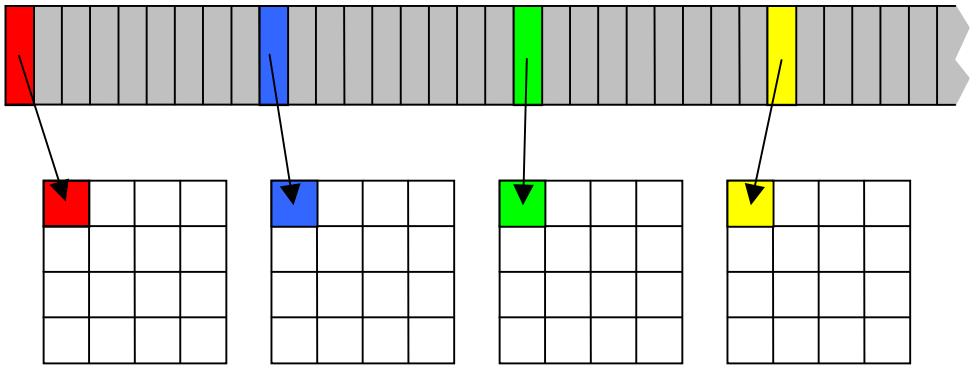


Figure 7.11: Schematic showing how the memory allocated (grey) relates to the "virtual frames" of the moving bracket. First cells in each of four simplified frames are shown with their corresponding positions in the allocated memory.

The frames are filled by reading in the x,y values from each picture, calculating the memory location of each dot and writing a "1" in that memory cell. When that frame needs to be filled with a later picture, it must first be "flushed", i.e. the old data must be erased or it would be confused with the new data. This is accomplished by reading in the x,y values again from the picture file, and using that data to write "0" to the required memory cell.

A further enhancement of the program would be to write different numbers to each memory cell corresponding to the picture number from which the data is extracted, thus removing any confusion about which dots were from which picture data, and obviating the necessity of flushing the frame.

The version of the DDT coded for this thesis uses rectangular search areas (SAs) and predicted positions (PPs) rather than the more exact shapes described in section 7.3.3. It was thought that such areas, while not as true to the nature of the problem as the more exact ones, would require much less CPU time to calculate their shape, and to scan. Plus, coding of the software to calculate and scan the exact areas was difficult due to time constraints.

The SAs and PPs are scanned in a raster-scan fashion; pixels with no dot have a value of 0 while those that do have a value of 1.

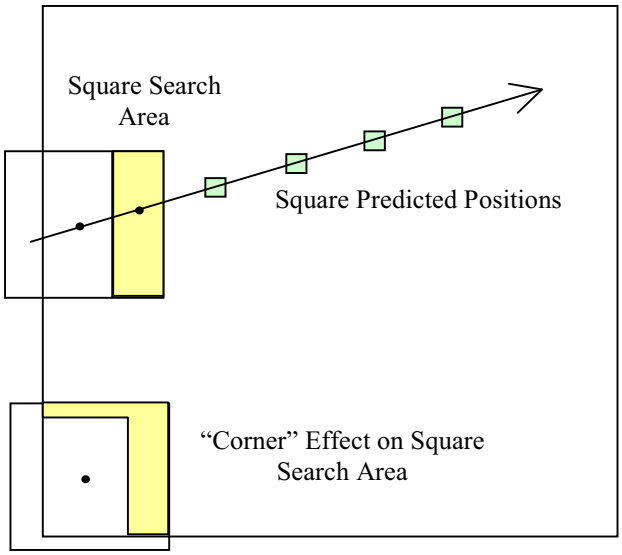


Figure 7.12: Illustration of the rectangular search areas used for the tests of the debris searching program. Yellow areas show the parts of the initial search areas that are actually searched. The small yellow squares represent the smaller predicted positions set up to confirm or refute the presence of a possible debris track (arrow).

7.7 Algorithm code

The full code listing may be found in Appendix 4.

7.8 Summary

- The structure and form of the DDT program was presented as a response to the nature of the data to be analysed.
- The program uses a moving bracket technique to analyse the most recent few frames.
- Full working of the program was presented in detail - geometric effects on the search method were described.
- Implementation of the algorithm into code was described.