

Development of EO Image Processing Applications using the GEDAE tool

David Humphreys, Graeme Harvey, Mark Daniell

BAE SYSTEMS Avionics

Sensor Systems Division

Edinburgh, U.K.

+44(0)131 343 5019

david.humphreys@baesystems.com

graeme.harvey@baesystems.com

mark.daniell@baesystems.com

ABSTRACT

Preliminary workstation designs have been developed using GEDAE for target cueing and tracking algorithms. These algorithms rely heavily on image processing techniques and are similar in complexity to those found within typical EO surveillance systems. This work has shown that GEDAE can be used to implement both the image and data processing aspects of these algorithms. The designs have used a blend of custom and library primitives and it has been found that porting to a target system can be achieved with ease. It has been seen that the choice of primitive type can have a significant effect on algorithm performance. This will be an important issue for the design of real-time target system applications.

1. INTRODUCTION

In recent years, image processing technology has been widely deployed within both military and security surveillance systems. This technology has been used to provide essential, automated surveillance functions such as target detection, tracking and recognition using image data sourced in real-time from a variety of Electro-Optical (EO) imaging sensors. Typical systems employing this technology include airborne and ground based surveillance systems.

The continued growth in complexity of these systems and the corresponding rise in software development costs are driving the need for system level design tools, with software auto generation support, that can be used to implement both the control and processing aspects of these systems. GEDAE has been identified as a possible tool for this task.

This paper presents preliminary results from the first phase of a study designed to explore GEDAE's suitability for implementing processing function designs similar to those typically found in these systems. This initial phase has focused in particular on the development and assessment of GEDAE workstation designs for a Target Cuer and a Target Tracker. These algorithms are well defined and are considered to be similar in complexity to those deployed in a practical EO surveillance system.

Section 2 provides an overview of the architecture, main functions and sub-systems of an example EO surveillance system and outlines the tasks performed by a typical Cuer and Tracker within such a system. Section 3 then provides a top-level discussion of the GEDAE designs produced for these algorithms and provides a first estimate of design performance. Initial views on GEDAE's suitability for EO processing implementation are then given in section 4. Future work aimed primarily at porting the Tracker and Cuer designs, and additional new designs, to a

multiprocessor target system to assess GEDAE's target system design tools is outlined in section 5.

2. Example EO System

An example EO Surveillance System that would utilise a Cuer and Tracker is shown in Figure 1.

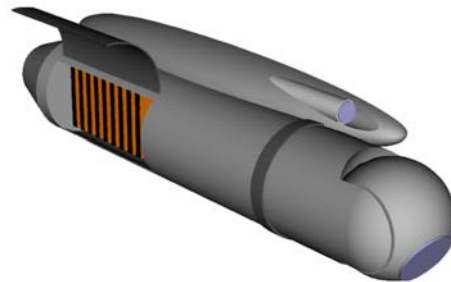


Figure 1: Example EO Surveillance System

Such a system would typically be carried on a fast jet platform and may provide the following functions:

- Navigation support using a forward looking sensor
- Ground or air target detection, tracking and recognition

The system may consist of the imager and processing sub-systems shown in Figure 2. This example system uses two imagers:

- Wide Field of View (WFOV) Imager

This imager may be used as a navigation aid and to provide imagery for the Target Cuer. It would be nominally aligned with the aircraft longitudinal axis.

This sensor could be a Focal Plane Array (FPA) Infra-Red (IR) camera with 640 x 480 pixel resolution with a field of view greater than 10°.

- Narrow Field of View (NFOV) Imager

The NFOV imager would be pointed at targets of interest found within the WFOV and the high-resolution imagery it generates used for tracking and recognition. This sensor could also be a FPA camera with array size similar to the WFOV sensor, but with a smaller field of view typically $<1^\circ$.

In a typical scenario, the Cued would detect multiple targets of interest using the WFOV imagery. The Recognition process would then interrogate these and a single target would be selected for tracking.

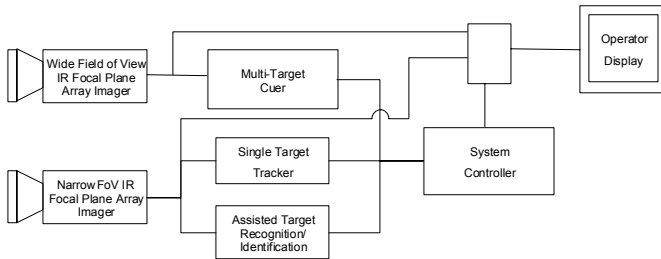


Figure 2: Surveillance System Block Diagram

3. Algorithm Designs

Preliminary GEDAE designs were produced for typical Cued and Tracker algorithms and a first estimate of design performance made using GEDAE's trace table facility. Initial porting of these designs to a target system was then undertaken. An overview of this work is presented in this section.

3.1 Target Tracker

The block diagram for the tracker is shown in Figure 3. This particular tracker is known as a centroid tracker and uses the position of the target's centroid as the basis for tracking.

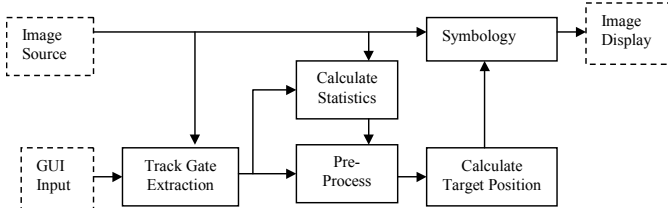


Figure 3: Tracker Block Diagram

3.1.1 Algorithm Overview

The tracker consists of the following major processing and support functions.

Processing Functions:

These are primarily signal processing, with some elements of data processing and control.

- Track Gate Extraction

Extracts an area of the image expected to include the target to be tracked.

- Calculate Statistics

Statistics for the image and track gate are calculated. These statistics are used to determine appropriate threshold levels for the pre-processing.

- Pre-Process

The image is pre-processed to extract the target from the background and clutter.

- Calculate Target Position

The target centroid is calculated and it's absolute position in the image determined.

- Symbology

Symbols representing the positions of the target and tracking gate and a text line indicating algorithm status are generated and overlaid on the source image.

Support Functions:

For I/O and configuration.

- Image Source

A sequence of greyscale images sourced from either disk, live video or a simulation generated in real time.

- GUI Input

A user interface to provide mode selection options and manual acquisition gate placement.

- Image Display

The display function generates an X-Windows image display and outputs the current image with overlaid symbology on the screen.

3.1.2 GEDAE Implementation

The algorithm was decomposed into 15 functions. These include a range of image and data processing elements. The image processing elements are primarily matrix operations on full and sub images.

Initial prototype designs were produced for these functions. As GEDAE has a large library of matrix and vector primitives it was possible to develop designs for many of these relatively quickly. It is expected however that the need to design basic image processing functions such as image threshold and centroid calculation using low level primitives will lessen as image processing library functions become available.

The data-processing functions used are relatively simple in terms of computation. However, they employ complex logic in order to achieve the correct functionality. After consultation with BHDS it was decided to implement these functions using custom primitives with encapsulated C code.

An initial, full design for the tracker was created using the above approach. This was then benchmarked on a Windows NT workstation using the GEDAE trace table tool. The specification of the workstation used was:

- Intel P4 2.0 GHz processor
- 1 GB PC800 RDRAM
- ATI Rage 128 Ultra graphics card
- MS Windows NT 4.0

Initial benchmarking indicated that the total processing time per frame, for 320 x 256 pixel images, was approximately 36ms using the first full version of the algorithm. This included 18.7 ms to display the image using the GEDAE `m_MA` and `m_disp` primitives. The actual processing time of the algorithm was then approximately 17ms per frame of data.

This processing time was considered to be excessive and was observed to be due to scheduling the large number of low-level primitive boxes used in this initial design; such as single operation primitives and logic boxes. It was therefore decided to modify the implementation of the algorithm in an attempt to improve the execution speed. This resulted in a second prototype workstation version of the algorithm for which the following execution times were observed.

Table 1 Tracker Execution Times
[P4 2.0 GHz Workstation]

Function	Execution time (µs)
Track Gate Extraction	132
Calculate Statistics	787
Pre-Process	278
Calculate Target Position	240
Symbology	310
Total	1747

It can be seen that a considerable increase in performance has been achieved over the original 17ms execution time. This was due to a combination of factors, including removal of dynamic queues and reducing the number of simple primitives by the use of ‘higher-level’ custom primitives. This illustrates that the choice of design approach can significantly influence the performance of a design. A specific example highlighting the two approaches considered is presented in Section 3.1.3.

Table 2 highlights the number of custom and GEDAE primitives used in the second design.

Table 2 Tracker Function Breakdown

Box Type	Number
Total flow-graphs	31
GEDAE Primitives	104 of which 10 are type conversions and 48 are GUI functions
Custom Primitives	30
Typedefs	37
Total	202

While custom primitives account for nearly one quarter of the total number of processing blocks the actual development time for these was minimal. This was due to the concise and modular nature of each primitive and use of modified GEDAE primitive code.

Considerable use was also made of typedef boxes in order to simplify the flow-graph wiring. Similarly, parent flow-graphs were used extensively in order to make the algorithm canvas more readable.

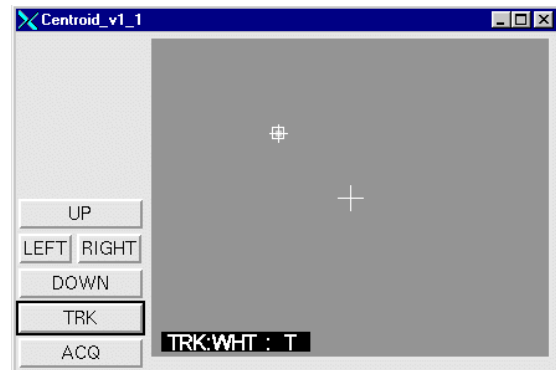


Figure 4: Tracker Screen shot

An example GUI screenshot of the completed Tracker is shown above. This shows a single white-hot target being tracked by the algorithm.

3.1.3 Alternative Implementations

In order to illustrate the alternative design approaches discussed in the previous section, the designs produced for the centroid function are briefly discussed below. This function is part of the calculate-target-position function of the tracker and is defined by the following equations derived from Ref. [1]

$$X_c = \frac{\sum_{i=0}^{COLS} ((\sum_{j=0}^{ROWS} P(i, j)) \times i)}{\sum_{i=0}^{COLS} \sum_{j=0}^{ROWS} P(i, j)} \quad (1)$$

$$Y_c = \frac{\sum_{i=0}^{ROWS} ((\sum_{j=0}^{COLS} P(i, j)) \times i)}{\sum_{i=0}^{COLS} \sum_{j=0}^{ROWS} P(i, j)} \quad (2)$$

where: X_c = centroid X position
 Y_c = centroid Y position
P = pixel intensity value

The original flow graph design for this function is shown in Figure 5 and comprises 19 GEDAE primitives, of which 7 are data and token type conversions.

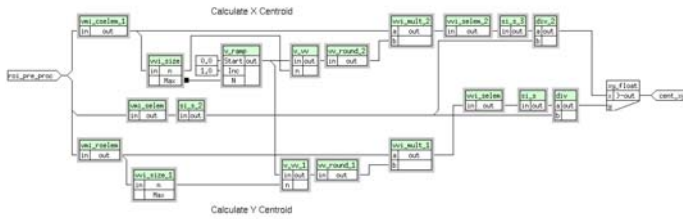


Figure 5: Original Centroid Function Flow-Graph

The single custom primitive shown below replaces this flow-graph in the second implementation of the algorithm. This is comprised of approximately 35 lines of C code.

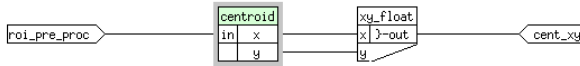


Figure 6: Custom Centroid Function Flow-Graph

In order to confirm the results observed during workstation tests a simple benchmarking exercise was performed on a single 500 MHz G4 processor of a target system. The target system is described in more detail later.

Inspection of the trace-tables revealed that the custom primitive executes an order of magnitude faster on the target system than the original flow-graph. The execution time for the custom primitive is approximately 15µs compared with more than 200 µs for the original flow-graph. Thus confirming the workstation results and observations. Similar results were seen for other functions that were implemented in this manner. It should be noted that at present the centroid custom primitive does not make use of e_ functions. This may further improve performance.

3.1.4 Initial Target System Benchmarking

As an initial exercise in using GEDAE on a target system, the improved workstation algorithm was ported to a target system in order to assess the ease of this task in GEDAE. The architecture of target system used for this work is outlined in Figure 7.

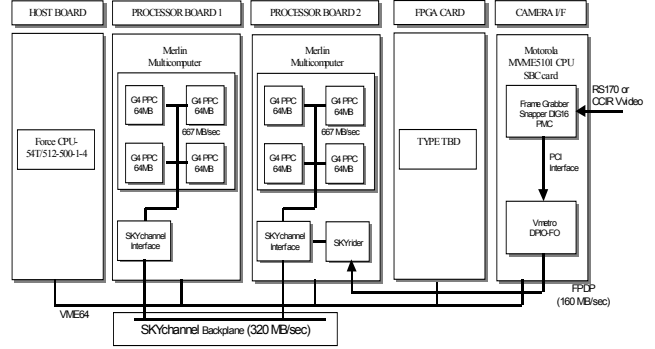


Figure 7: Target System Architecture

Its key features are:

- A SKY Computers SDS-2 7 slot, 6U VME unit
- Force CPU-54T/512-500-1-4 host controller
- Two Merlin quad PowerPC™
- MOTOROLA MVME5101 Single Board Computer with Camera Interface Board and interface to Merlin card.

An initial mapping of the Tracker design to the host and a single processor on the target system has been performed. This proved to be a trivial task, taking approximately one hour to transfer all the files, modify a single custom primitive that wouldn't operate on the target and select the appropriate Group Settings.

Benchmarking was performed to allow a comparison with the workstation performance results. This gave an execution time of 6690 µs for 640 x 480 imagery compared to 3565 µs on the workstation. These times exclude host display and VME transfer of the images to the host. This would produce a theoretical processing rate of 149 Hz, more than adequate for real-time operation with a typical video source, such as 30Hz RS-170 or 25Hz CCIR. However future work is planned to investigate the capabilities of the processor mapping tools in more detail.

3.2 Target Cuer

The second of two algorithms investigated was the Target Cuer. This algorithm detects potential targets, forms temporal track files of these targets over a number of image frames and highlights high priority tracks as cues. The Cuer has two sub-algorithms: Multiple Target Detection (MTD), and Multiple Target Tracker (MTT) as shown in Figure 8.

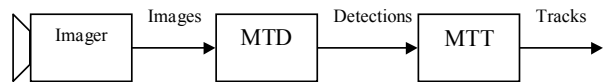


Figure 8: Cuer Structure.

In the following sections the MTD and MTT algorithm implementations are briefly discussed and a first estimate of performance given.

3.2.1 MTD Algorithm Overview

The block diagram of the MTD is shown in Figure 9.

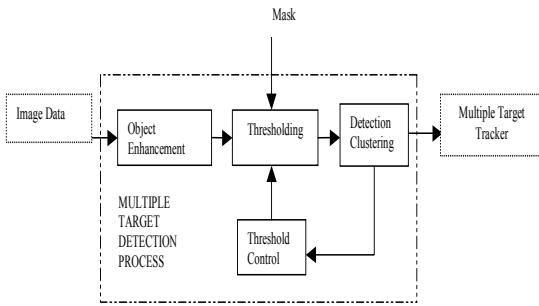


Figure 9: MTD Top Level Block Diagram.

The four processing blocks of the MTD algorithm are:

- Object Enhancement: A 2-D spatial filter is applied to each frame.
- Thresholding: A threshold is applied to the enhanced image to highlight potential target pixels.
- Detection Clustering: Adjacent potential target pixels are grouped into clusters. The centre of a cluster is calculated and the co-ordinates passed to the MTT algorithm as a detection.
- Threshold Control: An adaptive threshold is calculated.

3.2.2 MTD GEDAE Implementation

An initial GEDAE flow graph of the MTD algorithm has been created, and is shown in Figure 10. Note that the structure of the algorithm closely mirrors the main function blocks illustrated in Figure 9. The additional functions shown in the flow-graph were to support test and demonstration.

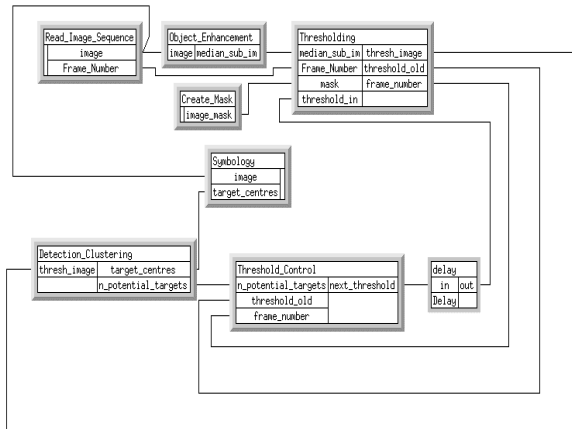


Figure 10: MTD Top Level GEDAE Flow Graph.

The design approach for this algorithm was similar to that for the Tracker. Likewise, the current design is primarily composed of custom primitives with use of GEDAE primitive functions where possible.

Sample results from initial workstation benchmarking with 640 x 480 imagery are shown in Table 3. These give an indication of the expected relative execution times for each main sub-function of the design.

Table 3: MTD Function Execution Times
[P4 2.0 GHz Workstation]

Function	Execution Time (ms)
Object Enhancement	331
Thresholding	29
Threshold Control	<1
Detection Clustering	58

These results indicate that this design would not operate in real-time on a single processor of the target system using the preliminary implementation described above. Techniques for improving the speed of operation of this design will need to be investigated to produce a suitable target system design. These may include image data partitioning, parallel processing and use of optimised e_ functions.

3.2.3 MTT Algorithm Overview

The top-level block diagram for the MTT is shown in Figure 11. The Track Association and Track File Update functions process the track database. Track files are created and updated based on the detection data received and high-priority tracks are output as cues.

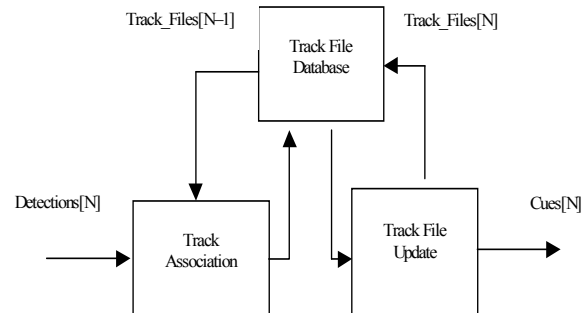


Figure 11: Multiple Target Tracker Block Diagram.

3.2.4 MTT GEDAE Implementation

Figure 12 shows the top-level GEDAE flow-graph for the MTT algorithm. Also shown is the test bench flow-graph used to test the design.

The design consists of hierarchical flow-graphs and application specific data processing primitives produced to capture the design hierarchy and required data processing. In the absence of a dedicated library of data processing primitives, custom GEDAE primitives were produced. Initial workstation benchmarking indicates that the algorithm executes in less than 1 ms with 12 targets. Future work will investigate the performance of this algorithm on the target system.

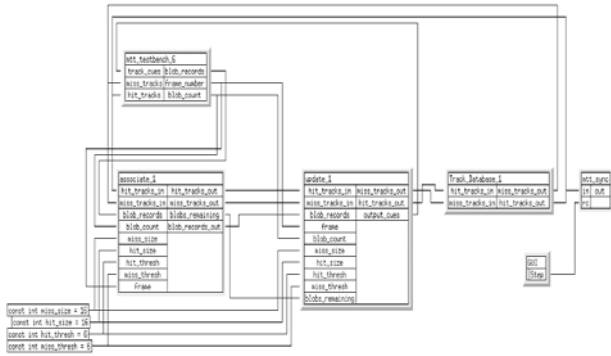


Figure 12: MTT Top Level GEDAE Flow Graph

4. Conclusions

Preliminary GEDAE designs have been created for typical Cuer and Tracker algorithms. These designs have employed a mix of custom and embedded library primitives. This work has shown that image processing based EO algorithms can readily be implemented using the available GEDAE toolset. It has been observed that the choice of primitive type can have a significant impact on algorithm performance. This issue will have a bearing on achieving real-time performance and will be explored further when developing target system designs.

The results obtained from this initial design work have enabled a first impression of GEDAE to be formed. A more extensive view will be obtained in follow-on work that aims to produce practical target system designs. Initial comments and observations on GEDAE are given below:

- The majority of the image processing functions studied could be produced using GEDAE primitives, however they were almost exclusively found to be less efficient than hand-coded custom primitives.
- The ability to use legacy and custom C code provides excellent flexibility and in some cases can allow more rapid algorithm development than use of the library functions.
- The target system mapping, scheduling and analysis tools have been found easy to use.
- The availability of application-specific libraries and an input type-independent core library would make the tool considerably easier to use.
- GEDAE’s block diagram data-flow representation allows for easy understanding of algorithm structure and makes replacement of existing, or addition of new function blocks a trivial task
- GEDAE has a number of minor usability issues that are frustrating to the novice user.

5. Future Work

The next phase of the work will primarily focus on exploring GEDAE’s target system design tools and will involve the following three tasks:

- Target System Designs

The Cuer and Tracker designs will be fully ported to the target system to enable a more detailed investigation of GEDAE’s target system mapping and optimisation facilities.

- Additional Target System Designs

New designs will be produced for processing functions not included in the cuer and tracker algorithms to allow further exploration of GEDAE’s capabilities for EO processing implementation.

- Integrated EO System Model

An EO System model based on that shown in Figure 13 will be created and ported to the target system to allow further exploration of GEDAE’s capabilities. The existing Cuer and Tracker designs will be integrated into this model.

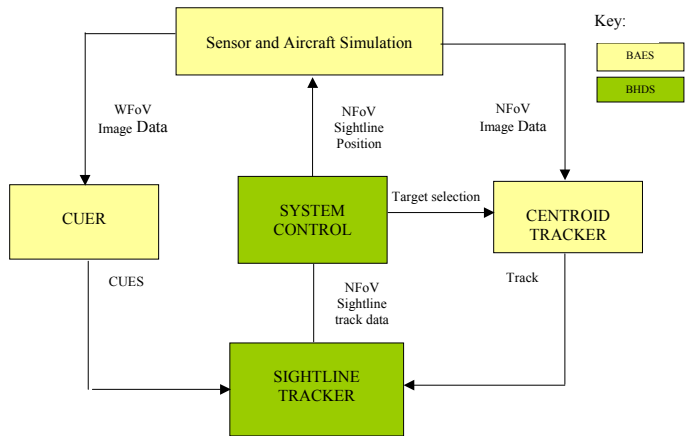


Figure 13: Basic EO System Model

6. References

[1] Sonka, M., Hlavac, V. and Boyle, R., Image Processing, Analysis, and Machine Vision, 2nd. Ed., PWS Publishing 1998