Detecting Space Debris using Phase Congruency

Student: Clive Trenton
Supervisor: Associate Professor Andrew Lambert
University of New South Wales at Australian Defence Force Academy
23 October 2017

Abstract - Modern technology depends upon the safe use of satellites to provide effective and comprehensive global communications coverage. Ongoing space operations are under constant threat of impact with space debris. This report investigates the viability of Phase Congruency image processing at high speeds, in conjunction with a FPGA, to produce a fast and accurate space debris detection tool. Phase congruency can process images, at high speeds (up to 500 fps), to facilitate feature detection on high-speed CMOS sensor imagery. This project advances the investigation of debris detection using high speed imagery in conjunction with FPGA’s with a view to determining the debris’ trajectory. The effectiveness of phase congruency is successfully confirmed on space imagery and a pre-existing VHDL map is modified to permit high speed full-frame image retrieval for off-line analysis. The effectiveness of configurable phase congruency is corroborated and its potential to process a full-frame image, with offline analysis, permits the project to be developed to fruition.

I. PROJECT OUTLINE
   A. PROJECT STATEMENT
   B. GOALS
   C. PROJECT PLAN

II. INTRODUCTION
   A. DEBRIS TRACKING
   B. DEBRIS TRACKING ANALYSIS
   C. PHASE CONGRUENCY
   D. EXISTING HARDWARE & SOFTWARE

III. PROJECT DEVELOPMENT
   A. SYSTEM DEVELOPMENT
      III.A.1 Image Storage
      III.A.2 Phase Congruency Analysis
   B. RESULTS
   C. DISCUSSION

IV. CONCLUSION

V. FUTURE WORK

VI. REFERENCES

NOMENCLATURE

FPGA = Field Programmable Gate Array
SSN = Space Surveillance Network
USA = United States of America
RAID = Redundant Array of Inexpensive Discs
VHDL = VHSIC Hardware Description Language
FTN = Falcon Telescope Network
RAM = Random Access Memory
USB = Universal Serial Bus
PCB = Printed Circuit Board
SSD = Solid State Drive
CMOS = Complimentary metal-oxide semiconductor

Engineering Project Report 2017, UNSW Canberra at ADFA
I. PROJECT OUTLINE

A. PROJECT STATEMENT

This report investigates the feasibility of using phase congruency, in conjunction with FPGA technology, to process high-speed CMOS imagery. Although beyond the scope of this report the final assembly will be used with controllable optics, thus providing a low-cost alternative for detect space debris.

B. GOALS

The following list details the goals to achieve the project statement:
1. Investigate the feasibility of phase congruency
2. Determine the effectiveness of phase congruency upon a sample image
3. Establish operation of existing project model
4. Store & recall an image to PC via USB (Full-frame) at fast rates
5. Process real-time phase congruency in the FPGA
6. Investigate a control technique to provide motor control output

C. PROJECT PLAN

The aim of this project is to investigate the potential of phase congruency, when used in conjunction with high speed imagery, to assist in the detection of space debris. We utilise a Spartan 3A FPGA interfaced with a PB-1024 high speed CMOS sensor onto a prefabricated PCB which incorporates RAID data output and USB connectivity [1] [2]. The VHDL provided with the project includes a Visual Basic software interface capable of uploading camera settings and retrieving decimated image data. The project also includes FPGA code for phase congruency and a VHDL project.

Improvements to this system, from this project, include positive confirmation that phase congruency can successfully process an image and prepare aforementioned image for feature detection techniques and the VHDL map now possess’ an alternate save functionality thereby enabling the retrieval of a full-frame image to a PC for offline analysis.

II. INTRODUCTION

There are a large number of active satellites in earth orbit, the US SSN currently records over eighteen thousand active satellites and in excess five hundred thousand pieces of debris. This debris presents a significant hazard to ongoing space operations and most of this debris is present within a LEO, between 800 km and 1000 km.

As reliance upon space operations rises the number of active satellites, and quantity of debris, steadily increases. Increased space operations increase the probability of an impact, the now popular use of constellations presents increased risk of impact now estimated at over five percent [3]. If left unchecked the danger posed by debris will increase, rendering ongoing space operation expensive and difficult.

To reduce the threat posed by debris large maps of debris are maintained; these debris maps are maintained by organisations with a vested interest in space operations (e.g. NASA, ESA etc) [4]. By identifying, classifying, and recording the location of known debris, active satellites are manoeuvred to reduce the likelihood of collision. The mapping process is a non-exhaustive task and is an ongoing expensive effort.

Identification and classification of debris is a difficult and laborious ongoing task that strives to maintain a comprehensive map of debris. This data enables the risk of a collision to be mitigated using collision avoidance and will enable future debris removal techniques to effectively remove debris.

A. DEBRIS TRACKING

Due to the range of physical characteristics the debris can possess it presents in a variety of sizes and states ranging from molecular to large debris. Large debris is classified as being greater than 10 cm; debris in this category possesses energy comparable to an anti-take round due to its high velocity. Most of the debris is, in the event of a collision, large enough to render a satellite inoperative and produce a significant amount of debris from the collision. The United States Space Surveillance Network (US SSN) estimates there to be more than 21,000 objects larger than 10 cm orbiting the earth. Debris presents a significant hazard and the large amount of debris within the densely populated LEO region would, if ignored, render future space operations untenable [4].

Existing detection techniques fall into two broad categories, defined by their dependence upon illumination. Active detection imitate’s targets to actively enhance their signature and detects a return from the object. Illumination can be provided in many forms (e.g. RADAR or LIDAR) but this is expensive and difficult implement.
These methods have the advantage of increasing reflected energy and permits the detector to be tuned to the illumination source, thus facilitating easier detection and reduced errors.

Passive methods, categorised by their lack of artificial illumination, utilise natural sources of radiated energy to assist in detection. Our system will employ passive detection, utilising the sun and reflected light as a target painter to assist in illuminating the debris. Energy reflected towards the detector is therefore dependant on several factors (e.g. spin, colour, presentation, attitude etc) and the effectiveness of this method is dependent upon factors beyond our control; experimental verification is required to determine the optimal settings for the system.

B. DEBRIS TRACKING ANALYSIS

Through analysis of the phase congruency output image, Figure 6, we can carry out line and point feature detection to identify ‘streaks’ and ‘dots’; each feature is representative of the debris/star map or debris. Assuming a telescope is stationary and a CMOS FPS of approximately 500 a fast-moving piece of debris will present as a ‘streak’ on the image and a star will present as a ‘dot’. due to atmospheric affects the image these outputs will be blurred.

Utilising our knowledge of the optics trajectory motors an algorithm will analyse the output images and instruct motors pointing the telescope to incrementally adjust their rate of operation. This process will be carried out iteratively, with each image proving a small graduated increase to the tracking motors until the tracking algorithm is successful; a successful track will be when the output image presents debris as a ‘dot’. Through analysis of known data regarding the telescopes position and slew rate the debris’ approximate location and altitude can be ascertained; Figure 5 provides a typical image of debris, in this case satellites, against a noisy and starry background.

Debris tracking is considered successful once the debris presents as a ‘dot’ on the optical image. In conjunction with the known data regarding the optics azimuth the approximate attitude and altitude information relating to the debris will be recorded and passed to the FTN for further analysis [6].

C. PHASE CONGRUENCY

Phase congruency performs edge detection, at real time, on the full-frame images at the optical output frame rate. We will need to determine the maximum frame-rate that can be realised as it is constrained by the limitations of the CMOS sensors sensitivity to the low illumination and FPGA operating speeds. The maximum frame rate is determined through experimentation as the number of variables involved in the calculation of light returned to sensor are too great e.g. object orientation, atmospheric disturbance, colour etc.

Phase congruency assumes that edges within an image are perceived as points where the Fourier components of an image are maximally in phase. By correlating between a locally engineered finite range of frequencies \( f_1, f_2, f_3, \ldots f_n \) and the image, the resulting processed output provides an effective tool for isolating edges. This produces a robust form of edge detection and, of import to this project, the product is invariant to contrast, illumination and is highly customisable.

Phase congruency is defined by the following equations:

\[
P_{C_1(x)} = \frac{\sum_n A_n(x) \cos(\phi_n(x) - \overline{\phi(x)})}{\sum_n A_n(x)}
\]

\[
P_{C_2(x)} = \frac{\sum_n W_n(x) [A_n(x) \cos(\phi_n(x) - \overline{\phi(x)}) - \sin(\phi_n(x) - \overline{\phi(x)})] n - T}{\sum_n A_n + \epsilon}
\]

Equation 1 defines phase congruency as the sum of functions of the cosine of the deviation of each component from the local mean (shown at Figure 1), this provides relatively bad localization and is sensitive to noise. Kovessi developed an alternate model, equation 2, which redefines phase congruency as the magnitude of the cosine minus the magnitude of the sine, this increases sensitivity to localization [5]; this is achieved by introducing a weighting function \( W_n(x) \) and a noise compensator \( T \). \( T \) represents the estimated noise value of the image, only energy values exceeding \( T \) are valid; this removes noise below the input threshold \( T \).

The individual frequency products of phase congruency are composed of complex values which possess both amplitude \( A_n(x) \) and phase angle \( \phi_n(x) \). Figure 1 demonstrates phase congruency as a sum of the Fourier components at several frequencies \( n = 4 \).
D. EXISTING HARDWARE & SOFTWARE

This project investigates the feasibility of incorporating a high-speed camera and real-time phase congruency; to achieve this quickly we will recycle an existing hardware imaging system. The existing hardware package consists of a Xilinx Spartan 3A FPGA, high-speed PB-1024 CMOS sensor and FTDI 245 USB interface embedded onto a PCB [2] [1] [7]. The provided FPGA model consists of multiple VHDL process blocks, each providing specific functionality. Figure 2 shows an abstracted diagram of the existing VHDL model and the additional image save functionality.

![Polar diagram demonstrating phase congruency as the sum of cumulative Fourier components](image1)

**Figure 1 - Polar diagram demonstrating phase congruency as the sum of cumulative Fourier components**

The CMOS PB-1024 image receptor is embedded onto the PCB and can provide 500+ fps at a resolution of 1024x1024 pixels at 8 bits per pixel; this equates to a high maximum output data rate of +500Mb/s. The sensor is configurable and permits configuration of many internal setting (i.e. windows size/location, output multiplexing, frame rate, data rate etc).

Configurations settings for the PB-1024 are established on start-up by FPGA; configurations settings are provided by applying a voltage to the applicable input pins of the CMOS sensor. Startup settings are established and can be changed, on-the-fly, using a PWM controlled DC output voltage pins connected to the PB-1024. A series of clocks are internally generated by the FPGA; these are connected to the CMOS sensor, C-processor, VGA output module and RAID based data save output and dictate the operation performance of each.

A C-processor emulator is embedded into the VHDL model; it acts as an interface between the USB connection and VHDL controller and provides a simple to use/modify interface. This connection provides the means to pass instructions to the VHDL controller (i.e. adjust camera settings, frame rate etc). Additionally, it enables a decimated
image (128x128) to be passed, via USB, to a PC from the FPGA. A second process, working in conjunction with the clock and decimated image output, provides a continues VGA output for viewing on an external monitor.

An additional VHDL script manages transferral of the high-speed CMOS imagery to an external RAID array for offline storage and analysis; the array consists of sixteen conventional magnetic drives. These are connected using four ethernet connections embedded onto the PCB. A customised array, housing the discs, operates using a disk striping principle; this allows relatively slow memory to store data at a high output rate. Due to technical issues with the array this feature was not functioning.

III. PROJECT DEVELOPMENT

The current FPGA can implement phase congruency, to enable implementation of the real-time phase congruency and line detection techniques the existing Altium model requires significant rework. This presented a significant problem due to the obsolescence of the software and faulty customised external storage array.

A. SYSTEM DEVELOPMENT

We utilise the PB-1024 CMOS sensor; it is configurable and allows the sensor to be configured to maximise the likelihood of debris detection. Phase congruency will be used to process the image in preparation for feature detection. The phase congruency method employs many variables, which are configurable. By configuring phase congruency variables, it can increase the likelihood of debris detection but will conversely produce a noisy image and produce false positives. It will therefore be necessary to adjust the configurable settings to maximise the overall effectiveness of the system.

To maximise phase congruency’s effectiveness, enable error correction and determine image coherence a high-speed, full-frame, images will need to be analysed offline. The current project relies on an unserviceable RAID array and it was not possible to extract an image to memory. To overcome this deficiency the existing system is modified to include the ability to store a single image, taken at an exceptionally high frame rate. Due to memory size constraints imposed by the RAM embedded onto the FPGA it is only capable of storing a single 1024x1024 image [2].

Implementing this functionality will enable the image to be analysed both pre- and post-phase congruency processing, this will enable phase congruency to be configured to provide optimal images suitable for feature detection methods. A Matlab script that performs the required mathematical manipulation is available online and updated by the its creator.

III.A.1 Image Storage

This project’s main contribution is implementing a full-frame save capability that can operate at the speed of the CMOS sensor. The VHDL script generated can save a full-frame image and, after a little manipulation. The project’s model was produced upon an aged software version of Altium designed to run on a MS Windows ME operating system, additionally it relied upon Altium Designer plug-ins to compile the model. Due to the age of the FPGA, support for it has been removed from modern version of Altium Designer, this necessitated the use of Altium Designer Version 14.3. This presented significant compatibility issues as it inhibited our ability to compile or upload and modifications to the Altium Model.

Compatibility issues were further exacerbated by the software’s reliance upon Windows Visual Basic, which is obsolete. Through a timely and meticulous process of driver and software testing these problems were eventually overcome; after extensive iterative testing, software and driver configuration a PC that enabled both Microsoft Visual Basic and Altium Designer Version 14.3 to operate concurrently was determined.

Image storage is achieved using a VHDL process added to the original FPGA map, denoted as ‘Full-frame image save function’ in Figure 2 and utilises FPGA’s onboard RAM [8]. Due to memory constrains the memory can store only a single image of 1024x1024 pixels. Interfacing with the process is managed via instructions provided by a Matlab script. Instructions are passed to the FPGA and image retrieval, once a save has been initiated, is returned via USB, enabling analysis of imagery and configuring of PB-1024 settings.

Image retrieval is initiated by a sending an instruction to the C processor, via USB; the C Processor responds by setting a retrieve instruction flag high. The retrieve process responds to this flag instructs the C-processor to revert the flag to zero on completion. A retrieve flag rising edge is detected by the VHDL retrieve process which extracts the contents of the RAM using a 20kHz clock; a slow clock is used to ensure error free image communication. Every 20kHz clock pulse the VHDL process extracts the contents of a memory address from both RAM modules and sequentially transmits them to the PC via the C-processor; this is graphically demonstrated at Figure 4.
The storage/retrieval process is initiated by an instruction sent from a PC, via USB, to the FPGA embedded ‘C’ code processor. The embedded processor passes the save/retrieve request to the VHDL image storage process by means of an internal flag. The FPGA identifies the rising edge of this flag and waits until the next rising-edge of the PB-1024 ‘Frame Sync’ output. This ensures that the image stored in memory is complete and entire, thus ensuring no image tearing.

An abstract flow diagram outlining the image storage/retrieval process is shown at Figure 3.

![Abstract VHDL overview of the image save/retrieval process](image)

The save and retrieve operations operate independently and are initiated via USB commands sent to the FPGA via the embedded C processor. Instruction are sent to the ‘C’ processor, from a PC via USB, which in-turn passes instruction signals to initiate the save/retrieve process; the requested process will operate autonomously until complete. Internal ‘hand-shake’ signals are used to initiate/stop dependant process; processes are stopped once complete and the system reverts back to initial conditions. This two-way protocol is represented by the double-ended arrows in Figure 3. A save operation is halted by the ‘Memory Address’ process and the retrieval process is halted once a complete image has been uploaded to the PC via USB.

The VHDL process is dependent on three system clocks, the FPGA has the Samsung K6R4016V1D-TC10 embedded and the memory write speed dictates the maximum CMOS sensor output data. The clock speeds of 100 MHz and 50 MHz for data writing and memory address increasing respectively as this is the maximum rate of operation that the RAM can sustain and ensures stable operation [8].

The CMOS sensor output data bus is 64 bits wide, each module of memory is 16 bits wide; to facilitate image retrieval the memory address is incremented at 100 MHz and the CMOS output data is transferred into memory at 2x16 bit elements into each RAM module every 50 MHz. Image data is transferred to the memory on the 100 MHz rising edge; to ensure the memory address is input correctly the memory address is incremented on the falling edge of the 50 MHz clock. A third clock is utilised, at 20 kHz, to transfer data from the FPGA memory to PC via USB at a sustainable transfer speed.

Image retrieval begins by sending an instruction to the FPGA via USB; the C-processor responds by raising a retrieve flag to high. The image retrieval process proceeds by setting the memory to read-mode and zeroing the address. A 20 kHz clock is used to extract the image from memory; upon each rising edge 16 bits of memory is extracted from the memory and passed to the C-processor for USB upload. Each rising edge of the 20kHz clock initiates a 16 bit data extraction from one alternating memory modules. The memory address is incremented every other falling edge. The process is halted once the memory address is its maximum.
III.A.2 Phase Congruency Analysis

Phase Congruency can be customised to suit individual images or styles. For example, performing phase congruency calculations at a multitude of phase angles (e.g. 0°, 30°, 45°, 60° & 90°) increases the sensitivity of the calculation and the output is less dependent upon the angle of the feature. Additionally, by varying the number of correlations and the frequency separation the sensitivity can be adjusted to better suit the images. As the output image must be capable of highlight stars and debris whilst simultaneously ignoring background noise the process must be configured to best suit the situation.

Phase congruency is applied to the CMOS images to isolate the features of interest (e.g. stars and debris); it is imperative that this is carried out at, or faster than, the optical sensors frame rate to improve the accuracy, effectiveness, and concurrency of the image. Phase congruency characteristics are manipulated by variation of the number of angles, the noise threshold and the frequency spread.

To determine the effectiveness of phase congruency a test image with a known debris pattern will be used to enable its viability to be quantitively determined. The process is tested using a Matlab script provided by modified script provided by Koveki [6]. The final produce will incorporate an alternative method of implementing phase congruency that still facilitates customisation.

The movement rate and azimuth heading of the telescope are provided by the motors and using this information the approximate trajectory and location of the tracked debris can be determined (e.g. by applying a Kalman filter); this information is suitable to be passed to the FTN for debris categorisation and mapping [4] [6].

B. Results

Figure 6 shows the produce of Phase Congruency when applied to the image at Figure 5. It was produced through an iterative process.
C. DISCUSSION

System set-up was difficult and highlighted a significant issue with the current hardware and software configuration. Due to obsolescence of both hardware, software and a plethora of operating system compatibility issues the existing configuration is untenable. The software configuration enabling VHDL upload and edit capability requires Microsoft Basic Studio and Altium Designer Version 13.

Replacing the Microsoft Visual Basic software with Matlab scripts enabled amended VHDL code to upload full-frame images to a PC for analysis and Phase Congruency calibration. Additionally, it was confirmed that the existing FPGA hardware is unable to facilitate embedded FPGA Phase Congruency; the project therefore requires a hardware upgrade to facilitate further development. Upgrading the hardware would also remove the compatibility issues present due to the dependency upon old system software.

Phase congruency functions as an edge detection tool and can, if properly configured, produce processed images suitable for line detection methods. The intricacies of phase congruency require the mathematical model to be calibrated, thus producing an optimal output.

The requirement to analyse images to ensure their quality both pre- and post- phase congruency application necessitated the inclusion of full-frame save/retrieve functionality. The incorporation of this functionality required the FPGA map to modified to include this functionality. Including this functionality was difficult due to software versioning problems and the necessity to save at very high output data rates.

VHDL script was generated and enabled a full-frame image to be extracted to a PC, as shown at Figure 3. Although the complexity of the VHDL script is abstracted the FPGA memory has the capacity to store a single full-frame image; this image can be extracted, via USB, to a PC for analysis. This enables calibration of the phase congruency process whilst also enabling processed and unprocessed images to be examined and adjusted to ensure optimum settings.

The processed image at Figure 6 successfully demonstrates the ability of phase congruency to produce an image and accurately remove background noise in a high contrast environment. The output image is suitable for line detection methods but demonstrates the need to configure the phase congruency method. The processed image displays four ‘dots’ and two ‘streaks’ whilst the source image has two clear ‘streaks’ and five clear ‘dots’. Furthermore, the image utilised for the phase congruency test includes additional cube-sats that are not visible; this demonstrates the inherent problem with all detection methods, due to object orientation or illumination problems the debris may not present upon the image and will remain unmapped until such a time that its orientation/position permits it to be detected.

Although beyond the scope of this project the output image at Figure 6 would be suitable for a feature detection. Experimentation with the VHDL model confirms the production of a variable duty cycle, suitable for driving optical motors, is achievable. Thus, with the appropriate coding, and FPGA upgrade, development of a real-time system capable of implementing high-speed phase congruency and feature detection is theoretically is possible pending an FPGA upgrade. Furthermore, real-time implementation of line detection upon the output image will enable generation of a negative feedback loop driving optic motors is feasible.
IV. CONCLUSION

Phase congruency functions as an edge detection tool and can, if properly configured, produce processed images suitable for line detection methods. The intricacies of phase congruency require the mathematical model to be calibrated, thus producing an optimal output.

The requirement to adjust CMOS settings to maximise its effectiveness necessitated the requirement to save an image for offline analysis; incorporation of this functionality required the FPGA map to be modified. This was difficult due to a lack of experience with VHDL coding and FPGA usage and was significantly hindered by difficulties in producing an operating system and software configuration that facilitated modifying of the VHDL map.

Having implemented phase congruency upon a test image, discrepancies between the number of identifiable objects in the phase congruency output and a known image exemplified the difficulty in detecting debris and the necessity to calibrate the technique. Configuring phase congruency to increase the number of positive matches is achieved by decreasing the sensitivity however this conversely increases the number of false positives. Optimisation of the process therefore necessitates calibration of both phase congruency and camera settings and, in the absence of an automatic calibration, validates the need to configure the method offline using an extracted image.

In summary, the technique for phase congruency is suitable for image processing and produces images suitable for line-detection method analysis. Although incomplete, this project has incorporated necessary additions to a platform upon which a debris detector, utilising high speed imagery and phase congruency, can be built.

V. FUTURE WORK

We have confirmed the ability of phase congruency to be used as an image processing tool and implemented features essential to configuring the process, therefore incorporating essential tools for future advancement on the project.

The current FPGA is incapable of implementing phase congruency onto the current VHDL map. To enable further development the project requires the current FPGA to be replaced with a newer model with sufficient free architecture to implement phase congruency VHDL code. This will also address the compatibility issues that plague the current FPGA due to software and hardware obsolescence.

Additionally, the project requires a line detection method to be implemented to process the images once phase congruency has been applied. This would enable the development of a tracking algorithm that outputs commands to the mounted optics that adjust for the different coordinate systems used by the telescope mount and likely debris trajectory. To reduce potential re-work and tracking of known objects it may be necessary to include a real-time link with the known debris map, thus reducing the likelihood of previously identified debris being tracked.

Increasing the effectiveness of phase congruency is beneficial and could be achieved by increasing the light provided to the camera from a light source i.e. debris or stars. The inclusion of an image intensifier would increase the signature of debris and objects. This would in-turn manifest as a stronger centralisation of light provided thus increasing the efficiency of the phase congruency technique and debris detection ability.

Due to the large output data rate of the existing system, a storage array system consisting of sixteen magnetic hard-discs in a disk striping is required to save full-frame imagery files. This storage solution was developed before the development of large capacity SSD’s. They have undergone significant technical advances and offer significant advantages the current storage array. If coupled with the upgrade of the FPGA, incorporation of a SSD will require the placement of a single pre-made VHDL scripting block and industrial interface. SSD’s can store over 14 hours of imagery, occupy significantly less volume and can continuously save at the required rate [8].

The PB-1024 CMOS sensor is configurable, to ensure the image is ideal for phase congruency the sensor variables must be set to optimum settings. Currently, this technique requires analysis of the image and iterative adjustment of the settings. The inclusions of an automatic technique that performs an automatic calibration would enhance the quality of the detection process by ensuring the sensor is set at optimal configuration.
VI. REFERENCES


