An Edge Computing Algorithm for Onboard Processing of Electro-Optical Imagery

Dr. Matthew C. Britton, Dr. Christopher Griffith, Dr. Brian Baptista

The Aerospace Corporation

ABSTRACT

This paper presents techniques enabled by a real-time embedded algorithm for onboard processing of electro-optic rate-track imagery acquired by ground and space sensors. These techniques offer many advantages: 10^4 reduction in data volume to be transferred off-sensor, elimination of confusion between RSOs and stars, multi-target tracking, multi-frame integration of both stars and RSOs at the single pixel-level, a variety of electro-optic feedback control options, and absolute metric and photometric calibration. These advantages are available in real time at rates exceeding video frame rates. These techniques promote sensor autonomy and alleviate the restrictions of communication bandwidth and processing latency that arise when transferring raw imagery off-sensor. This paper demonstrates these techniques using data collected on a GEO target from a 30cm ground telescope and through an ongoing embedded development effort using a Zybo Z7-20 FPGA development board. These techniques are patent pending and available for licensing from The Aerospace Corporation for government and commercial use.

1. INTRODUCTION

In the fields of space traffic management (STM), space domain awareness (SDA), and space control, electro-optical (EO) sensors obtain two-dimensional images of resident space objects (RSOs). These images are acquired sequentially in time, and are used in applications such as conjunction analysis for collision avoidance, remote proximity operations, light curve estimation and orbit determination. EO imagery are routinely acquired from sensors on both ground telescopes and space vehicles for processing in data centers.

The number of EO sensors and the quantity of data they generate are increasing significantly. The exponential increase in the number of government and commercial RSOs and the desire for sensor resiliency and observations on tactical timescales are precipitating a proliferation of EO sensors in ground and space. At the same time, modern focal plane arrays (FPAs) with 100 Megapixels or more operating at frame rates up to 100 frames per second yield data rates of order 1 GB/sec. The implications of more sensors generating data at these rates are shown in the upper right panel and described in the caption of Figure 1. The increase in communication bandwidth required to transport these images to data centers and the resulting processing latency drive costs and limit opportunities for autonomous decisions by the ground telescope or space vehicle hosting the EO sensor.

Limits on communication bandwidth and processing latency have a negative impact on EO sensor design and their concept of operations (CONOPS). Limited communication bandwidths force a compromise among sensor field of view, angular resolution, and frame rate. Large processing latencies preclude real-time electro-optical feedback for tracking and guidance control: the imagery may be acquired at rates of 100 frames per second or more, but feedback requires data transport and processing at the data center that occur on

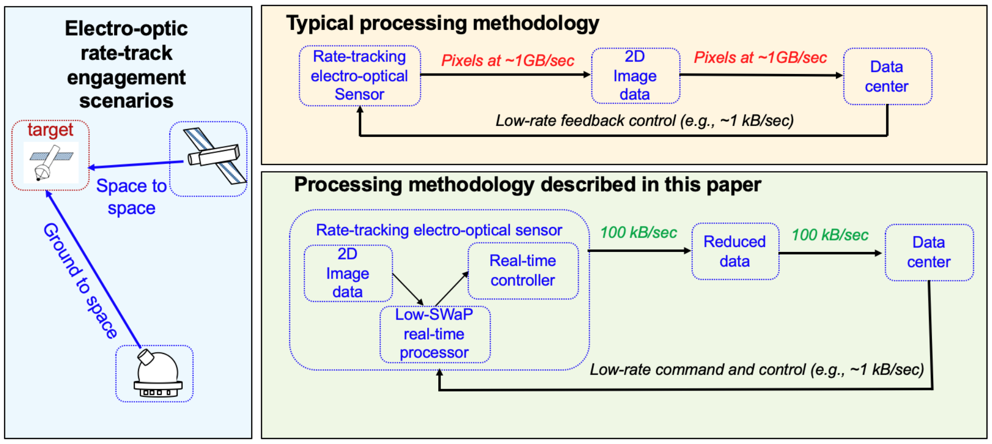

Electro-optic rate-track engagement scenarios
target
Space to space
Ground to space
Typical processing methodology
Rate-tracking electro-optical Sensor
Pixels at ~1GB/sec
2D Image data
Pixels at ~1GB/sec
Data center
Low-rate feedback control (e.g., ~1 kB/sec)
Processing methodology described in this paper
Rate-tracking electro-optical sensor
2D Image data
Real-time controller
Low-SWaP real-time processor
100 kB/sec
Reduced data
100 kB/sec
Data center
Low-rate command and control (e.g., ~1 kB/sec)

Fig. 1: The left panel displays examples of ground-to-space and space-to-space engagement scenarios for electro-optical (EO) telescopes operating in rate-track engagement scenarios. In rate-track mode, the EO telescope slews to track the target. The upper right panel shows a typical processing methodology for these EO sensors. Image data are transferred to a data center, processed, and resulting control commands are transferred back to the sensor. Image data can be acquired at rates of 1GB/sec or more. The lower right panel shows the processing methodology described in this paper. Image data are processed onboard to enable low-latency, real-time feedback control for autonomous decision support on the space vehicle. The reduced data are transferred to the data center at a much lower data rate. This paper demonstrates a reduction in data rate of order 10^4.

timescales orders of magnitude longer. Image processing at the data center splits decision support across the sensor and the data center, creating an interface that may be difficult to maintain and upgrade, while limiting opportunities for the ground telescope or space vehicle to operate autonomously. These negative impacts cascade to yield a functionally compromised sensor design and a more rigid, enterprise-level architecture that may be slow to deliver actionable information on tactical timescales.

This paper presents techniques enabled by a real-time embedded algorithm for onboard processing of electro-optic imagery acquired in rate-track mode from EO sensors hosted by ground telescopes and space vehicles. This paper uses on-sky data from a ground telescope and initial results from an embedded field programmable gate array (FPGA) development effort to demonstrate how this algorithm:

1. reduces data volume to be transferred off-sensor by a factor of order 10^4
2. differentiates between photons from RSOs and photons from stars, thereby eliminating confusion
3. performs multi-frame integration on stars and on the RSO at the pixel-level
4. performs multi-target tracking, with independent specification of integration time for each RSO
5. provides multiple EO feedback control options, including "beaconless" autoguiding to allow tracking on a user-supplied ephemeris referencing background stars
6. processes in-frame calibrators to provide real-time photometric and metric estimates in absolute units (e.g., irradiance, RA, DEC)

Alternative techniques for onboard processing of rate-track imagery rely on exceedance detection[13, 5], iterative, GPU-based neural networks[12, 14, 15] or image segmentation[19] to perform blob detection[16] of stars and RSOs. These techniques operate on a single image at a time, yielding single-image sensitivity limits. In comparison, the algorithm described in this paper processes and integrates over a continuous stream of images at the single-pixel level. This approach improves sensitivity limits as $\sqrt{N_{img}}$ for both stars and RSOs. Conceptually, the algorithm is an extension of the time and delay integration technique[18] generalized to compensate for motion in arbitrary directions on the focal plane. The algorithm is specifically structured to be embedded on an FPGA, so that radiation-hardened parts are available for flight programs.

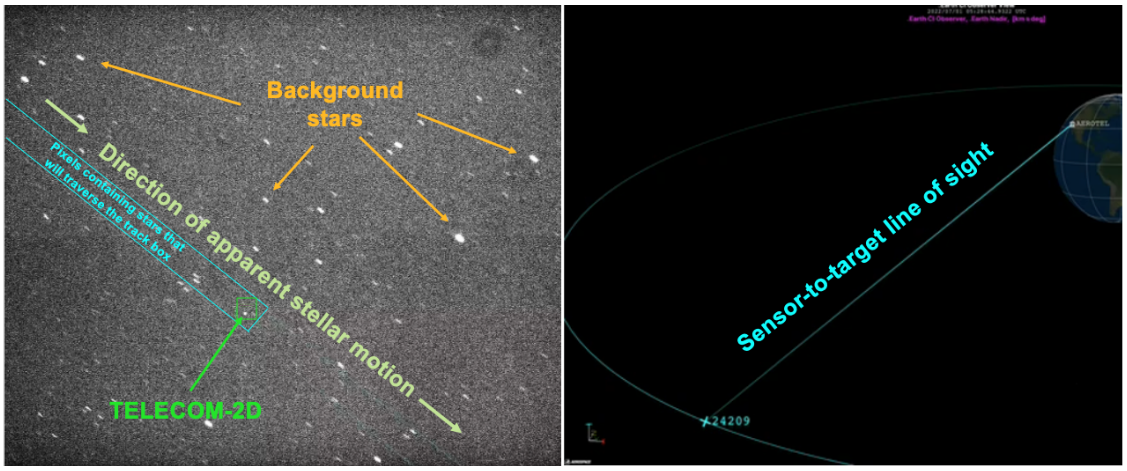

Background stars
Direction of apparent stellar motion
Pixels containing stars that will traverse the track box
TELECOM-2D
Sensor-to-target line of sight
24209

Fig. 2: The left panel: A single image from a sequence of 64 images acquired in rate-track mode on the GEO target TELECOM-2D (NORAD ID 24209) with a 30cm telescope located in El Segundo, CA. All images in this paper are presented on a clipped log stretch. The exposure time for this image was 1 second. The target is boxed in green, while stars in the field of view appear as streaks. This stellar streaking effect arises from the telescope's change in pointing direction to maintain tracking on TELECOM-2D during the time that the camera is acquiring the image. This tracking rate differs from that of the sidereal rate: the rate at which the stars appear to move for a ground observer. The right panel: The modeled geometry for the collect is shown, which predicts the evolution of the sensor-to-target line of sight during the exposure. From this prediction, one may infer the direction of apparent stellar motion as marked on the left panel. The sensor exposure time determines the length of the stellar streaks in the frame, while the time between successive images determines the streak offset between images. All stars within the field share the same streak length and offset. The modeled streak direction, length and offset suffice to specify a matched filter used by the algorithm to search the image for stars.

The imagery produced by EO sensors consists of time-sequences of images. Paper publications are not well suited to communicating the spatio-temporal effects displayed by these image sequences. In particular, Figures 2 and 4 - 9 are better viewed in color and as timeseries animations rather than single frames. A companion briefing containing animations, audio track descriptions, and written notes is available for download via the QR code at right or via URL[6].

This paper exercises the algorithm on observational data acquired 1 July, 2022 on the GEO satellite TELECOM-2D using a 4-Mpix visible camera mounted on a 30cm ground telescope in El Segundo, CA. The collect consisted of 64 sequential images at 1 second exposure time. A C++ emulation of the algorithm was developed to process these images and obtain the results shown in Sections 2 - 3.6. At the time this paper was written, development of the first embedded implementation of the algorithm was underway but not yet complete. This first implementation demonstrates hard real-time processing in the FPGA component of the algorithm on two independent targets in 4-Mpix images at a rate of 50 frames per second. Section 4 reports status of this embedded development effort.

2. INFERENCES FROM RATE-TRACK ELECTRO-OPTIC IMAGERY

The character of EO imagery acquired in rate-track mode is highly structured. The algorithm described in this paper capitalizes aggressively on this structure to maximize efficacy and minimize latency. Figure 2 and the companion briefing[6] display an example of rate-track imagery, while the caption of Figure 2 describes the structure inherent in all such imagery. The algorithm is structured through a sequence of inferences itemized below. Each item lists the sections of this paper that describe how the algorithm realizes the inference in an embedded implementation.

A. The limited number of pixels that contain stellar streaks which traverse the track box lie upstream of the track box. The remaining pixels may be discarded to reduce data volume. (Sections 3.1 and 3.3)

B. Stars move in one dimension along diagonals with an arbitrary angle depending on sensor and target engagement geometry. With an advantageous representation, stellar streak detection reduces to a set of one-dimensional calculations along diagonals rather than a two-dimensional calculation. Each one-dimensional calculation is independent of the others. (Section 3.2)

C. A model of the stellar streak orientation, length and offset between images is available for matched filtering in real-time. (Caption of Figure 2 and Section 3.2)

D. Stars upstream of the track box are imaged many times before entering the track box. Accumulating stellar streaks over N images improves stellar sensitivity by a factor $\sqrt{N}$, ensuring stellar detection before the star enters the track box. (Section 3.2)

E. Each star upstream of the track box enters and traverses the track box during a specific set of images. Stars are detected upstream of the track box and move at an offset between images that is predicted by the model. Therefore, the set of images in which the star is resident in the track box may be predicted prior to the star entering the track box. (Section 3.3)

F. Based on the previous inference, track box pixels affected by stellar photons are known at the time each image is acquired. This classifies photons from stars and from RSOs with no latency penalty. These pixels may be masked to eliminate confusion between stars and the RSO. (Section 3.3)

G. Masking pixels affected by stellar photons isolates the RSO, eliminating bias from background stars. This permits unbiased centroiding of the RSO for real-time electro-optic feedback control and multi-frame integration of the track box. (Sections 3.3, 3.6)

H. Multi-frame integration of the track box further reduces the volume of data to be transferred off-sensor. (Section 3.3)

I. The processing pipeline for extracting pixels upstream of a track box may be replicated for many track box locations within the image. This enables multi-target tracking. (Section 3.4)

J. Calibration stars traversing the field may be extracted in real time for absolute photometric and astrometric calibration of the RSO via catalog lookup. (Section 3.5)

3. THE EMBEDDED PROCESSING ALGORITHM

The algorithm described in this paper capitalizes on the structure of EO rate-track imagery and the inferences itemized in Section 2 by splitting the processing steps across an FPGA and a CPU. This divides the labor between brute-force computation better suited to an FPGA and higher level logic better suited to a CPU. Figure 3 displays the signal flow from the point at which the camera writes the image data into CPU RAM through its transfer into the FPGA where raw pixel data are processed and back out to RAM where the CPU can perform higher level calculations. To help orient the reader, Figure 3 is reproduced at the left edge of Figures 4 - 9 with the algorithmic step under discussion highlighted in cyan.

The division of labor between CPU and FPGA is tailored for system-on-chip (SoC) devices[17] such as Zynq[3] and Versal[2] SoCs. These devices offer ARM processors, RAM and programmable FPGA logic on a single device. The SoC form factor minimizes size, weight and power (SWaP) while providing opportunity to implement the algorithm within a self-contained software development environment. Section 4 describes the design and preliminary results of an embedded SoC implementation on a Zynq-7000 device.

3.1 Scatter-Gather Direct Memory Access Transfer from CPU to FPGA

Section 2 notes the finite region of the image containing background stars that will impinge upon the track box. In the example shown on the left panel of Figure 2, stars that traverse the track box enter the cyan-bounded region at upper left and move along diagonals towards bottom right. The cyan boundaries

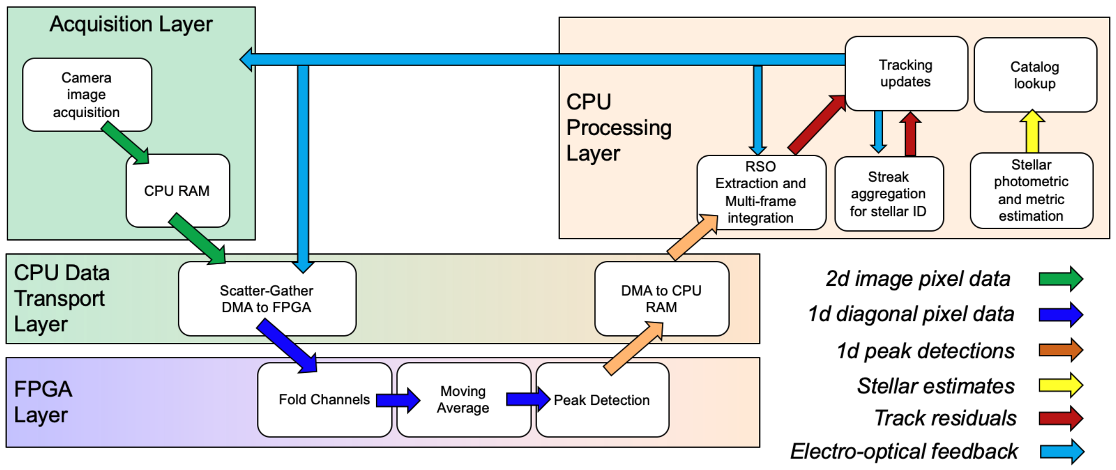
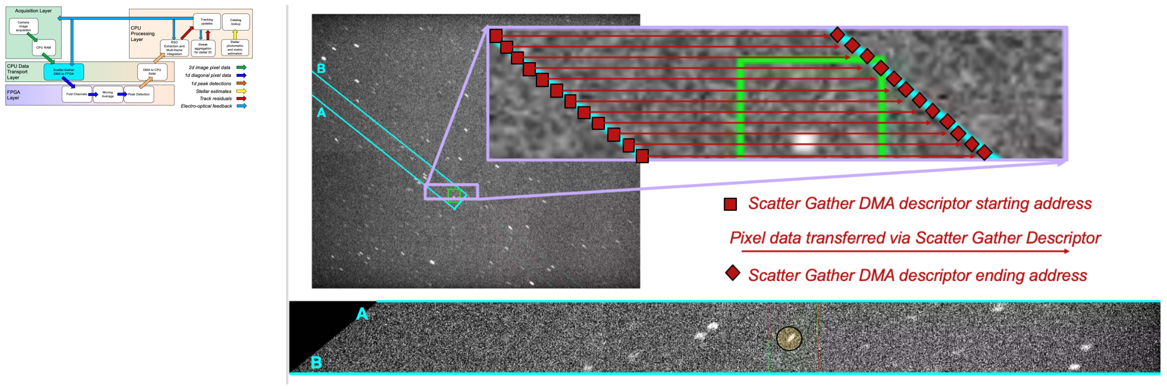


Fig. 3: The diagram displays the flow of data from acquisition in host RAM through a system-on-chip (SoC) device containing both a CPU and an FPGA. In the Acquisition Layer, data are transferred from the camera into CPU RAM. Once resident in CPU RAM, data are transferred into the FPGA programmable logic via direct memory access. The FPGA performs the computationally intensive pixel-level processing and reports calculations back to the CPU. For the on-sky collect described in this paper, the FPGA reduces the data volume by a factor of order 500. This data volume reduction enables the CPU to perform the higher-level operations shown in the upper right panel of the figure on a greatly reduced volume of data. This division of labor between the FPGA and the CPU is a key feature of the real-time processing algorithm presented in this paper.



Scatter Gather DMA descriptor starting address
Pixel data transferred via Scatter Gather Descriptor
Scatter Gather DMA descriptor ending address
A
B

Fig. 4: Illustration of the scatter-gather DMA transfer technique. Pixels bounded in cyan contain stars that will impinge upon the green track box. These pixels are to be transferred into the FPGA for processing. The two-dimensional pixel data are stored contiguously in RAM, and selection of the pixels bounded in cyan is performed via a set of scatter-gather descriptors. Each scatter-gather descriptor is demarcated by a starting and ending address, and specifies a contiguous region of memory to be transferred. The purple box in the figure enlarges a subset of the cyan region, in which descriptors are denoted in red. There is one descriptor per image row, and in this example the number of rows covering the entire cyan region is 1048. This set of 1048 scatter-gather descriptors constitutes a scatter-gather descriptor table, which the CPU creates based on the geometric model of the engagement scenario. The FPGA then serially transfers the memory via DMA to obtain the "pixel ribbon" shown in the bottom panel. The RSO is marked in orange on this panel. The reconfigurable scatter-gather descriptor table and subsequent DMA transfer generalizes the technique of time delay and integration to arbitrary directions of motion on the focal plane.

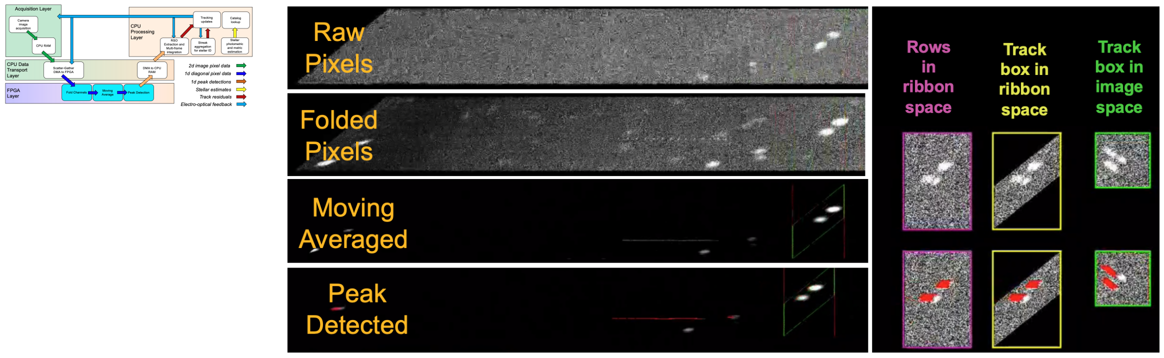


Fig. 5: Pixel ribbon processing steps performed within the FPGA. The pixel ribbon is transferred via DMA into a serialized stream shown at upper left. The pixel data are folded into an accumulation of prior pixel ribbons shifted by the modeled stellar streak offset between images, as shown in the second panel from upper left. The accumulations are then convolved with the modeled matched filter, as shown in the third panel from upper left. This is inference C in Section 2. The fourth panel from upper left shows the result of applying a user-defined peak detection threshold and marking pixels surrounding the peak with a red streak of length equal to the modeled streak length. These red streaks mark the stars detected by the FPGA component of the algorithm. The right panels show chip-outs of the section of pixel ribbon encompassing the track box (outlined in purple), the pixel ribbon masked by the track box (outlined in yellow) and the track box in image space (outlined in green). The upper right panel shows these chip-outs without marking the streak detections, while the lower right panel marks streak detections in red that have been identified in earlier images and projected into the track box via the model offset between images.

demarcate the set of pixels to be transferred into the FPGA. This paper uses "pixel ribbon" as a short, descriptive term for this region, as illustrated in Figure 4 and described in its caption. The pixel ribbon depends upon the geometry of the engagement scenario, the size and location of the track box within the image, and the pixel dimensions and orientation of the focal plane array (FPA). Given these dependencies, the CPU computes a scatter-gather descriptor table and writes this table to RAM.

During image acquisition, the camera writes to CPU RAM, the FPGA reads the scatter-gather descriptor table from CPU RAM, and then the FPGA direct memory access (DMA) transfers the pixel ribbon into the FPGA. During image acquisition, the CPU does not access or manipulate the memory. This frees the transfer process from the risk of CPU interrupts and frees the CPU for other processing tasks. This is the key advantage of DMA. This scatter-gather DMA transfer realizes the one-dimensional representation stated as inference B in Section 2.

3.2 FPGA Processing Layer

The pixel ribbon is transferred into the FPGA and presents as shown in the bottom panel of Figure 4 and via animations in the companion briefing[6]. The algorithm searches for stars within each row of pixels through the three steps shown in Figure 5 and described in its caption. The search over each row within the pixel ribbon proceeds independently and may be performed in parallel. The FPGA reports peak detections via DMA transfer back to CPU RAM. Each peak detection contains the pixel coordinates of the peak and an amplitude determined by the coaddition of accumulations formed from folding successive pixel ribbons and by the output of the matched filter.

As each successive image is acquired and processed, the FPGA reports peak detections. Each star within the image generates a sequence of these peak detections. In the left panel of Figure 2, stars progress from upper right towards bottom left at a modeled offset between images, and the detected peak amplitude increases linearly with the number of images containing the star. In the Figure 2 example, stars appeared in 53 successive images before reaching the track box, and the modeled streaks were of length 18 pixels. This yielded a signal-to-noise (SNR) improvement of $\sqrt{53 * 18} = 31$ over the single-image, single-pixel SNR. Obtaining this sensitivity in a single-image, single-pixel detection would require an aperture with 31 times the diameter. With this technique, the algorithm realizes inference D in Section 2 assuming a high performance low noise FPA.

Table 1: Data volume calculations from the first 10 frames of the sample collect shown in Figure 2. Column 1 reports the image index, while columns 2 and 3 report the raw pixel dimension of the images and their data volume. Columns 4 and 5 report the number of streaks detected by the FPGA and the required storage. The streak detection algorithm adopted a requirement of detection in four sequential images, so no streaks are listed in the table until image four. Each reported streak is assumed to occupy 12 bytes: 4 bytes for row index, 4 bytes for column index, and 4 bytes for amplitude. The track box dimensions and its data volume are listed in Columns 6 and 7. The volume of data occupied by the streaks and the track box is listed in Column 8. The ratio of data volume in Column 3 to the data volume in Column 8 is listed in the final column. This ratio is a function of the number of background stars and the track box dimensions, and in this example is always greater than 500.

Col. 1 Image Index	Col. 2 Image Dimens (Pixels)	Col. 3 Image Data Volume (bytes)	Col. 4 # of Streaks Detected	Col. 5 Streak Storage (bytes)	Col. 6 Track Box Dimens (Pixels)	Col. 7 Track Box Data Volume (bytes)	Col. 8 Total Output Data Volume (bytes)	Col. 9 Data Volume Ratio (Col. 3 / Col. 8)
0	2560x2160	11059200	0	0	96x96	18432	18432	600
1	2560x2160	11059200	0	0	96x96	18432	18432	600
2	2560x2160	11059200	0	0	96x96	18432	18432	600
3	2560x2160	11059200	16	192	96x96	18432	18624	593
4	2560x2160	11059200	33	396	96x96	18432	18828	587
5	2560x2160	11059200	57	684	96x96	18432	19116	578
6	2560x2160	11059200	82	984	96x96	18432	19416	569
7	2560x2160	11059200	79	948	96x96	18432	19380	570
8	2560x2160	11059200	80	960	96x96	18432	19392	570
9	2560x2160	11059200	42	504	96x96	18432	18936	584

In performing the compute-intensive calculations on the pixel ribbon, the FPGA provides benefits in both processing latency and data volume reduction. To illustrate the processing latency benefit, consider the example shown in Figure 2 in which the FPGA processes a pixel ribbon of dimensions 96 x 1048. Assuming the algorithm can be pipelined to require two clock cycles per pixel and an FPGA clock speed of 50 MHz, a single row can be processed in 20.8 microseconds. If all rows can be processed in parallel, the processing time required per image is also 20.8 microseconds. This corresponds to a processing rate of 24 kilo-frames per second. While attaining this frame rate for an EO imaging system is challenging for many reasons, this simple calculation demonstrates that the FPGA processing will not be the latency bottleneck.

Table 1 illustrates the data volume reduction benefit using the example collect shown in Figure 2. In performing and reporting the peak detection calculations illustrated in Figure 5, the FPGA has discarded all pixel data and has reported only the peak detections arising from stellar streaks within the pixel ribbon. The final column of this table lists the data volume reduction ratio. This factor represents the first $\sim$2.5 orders of magnitude data volume reduction of the 10^4 value stated in inference A of Section 2. An additional $\sim$1.5 orders of magnitude reduction is described in Section 3.3. This first data volume reduction factor allows the CPU to perform higher-level analysis on less than 1% of the data volume in the raw image.

3.3 RSO Extraction and Multi-frame Integration

As images are acquired and peak detections are reported, the CPU analyzes these peak detections to aggregate peak histories that arise from stars as they move through successive images. Consider the history of peak detections arising from a single star. The first peak detection occurs when the peak amplitude exceeds a threshold above the noise floor. In each subsequent image the measured peak amplitude increases and its measured coordinate shifts by the modeled offset. This distinctive pattern allows the algorithm to identify peaks from successive images as having arisen from a single star. This paper uses the term “streak history” to denote the timeseries of associated peak detections generated by a single star within a single row of the pixel ribbon. The streak history contains a timeseries of coordinate estimates that identify the star’s offset and rate of approach to the track box. This determines the set of future images in which the track box will contain the star. Examples of these predictions are marked on the lower right panels of Figure 5. This is inference E in Section 2.

As an added benefit, the history of peak detections follows a distinctive amplitude and offset pattern that extends over successive images. This pattern is not exhibited by stray light gradients or by image defects such as hot pixels, cosmic ray detections, and some focal plane array artifacts. Consequently, the streak detection algorithm is relatively robust against these types of spurious effects.

The streak history allows the algorithm to predict pixels within the track box that are affected by background stars. With this foreknowledge, the CPU can prepare a track box mask to eliminate these pixels prior to processing the track box data. This mask eliminates confusion between stars and the RSO, and has the effect of classifying stellar photons from RSO photons with no latency penalty. A more aggressive variant on this approach is to subtract the estimate of stellar photons formed from the streak history and retain the residuals for processing. This variant was not evaluated in this paper. Either variant eliminates confusion between RSO and stellar photons: inference F in Section 2.

Elimination of confusion permits estimation of the RSO centroid without introducing biases from background stars. Figure 6 displays results of fitting a Gaussian to the track box pixels after application of the mask. For each image, pixel data within the track box were fit to a 5-parameter 2D Gaussian using a Levenberg Marquardt nonlinear optimizer. The five parameters were Gaussian amplitude, x and y centroid, width of the Gaussian, and the mean background. These unbiased centroid estimates are available in real time for use in EO feedback control: inference G in Section 2.

The masked track box images may be integrated over multiple images to obtain deep exposures on the RSO that do not suffer from stellar confusion. Figure 7 displays examples of these multi-frame integrations. The centroids estimated in the Levenberg-Marquardt fit may be used to co-align each image with the running accumulation of track box images. This multi-frame integration at the pixel-level permits the EO sensor to achieve limiting sensitivity that scales as $\sqrt{t_{int}}$, as shown in the right panel of Figure 7: inference G in Section 2 assuming a low noise FPA.

This feature of the algorithm can enhance opportunities for sensor autonomy. For example, scheduling a

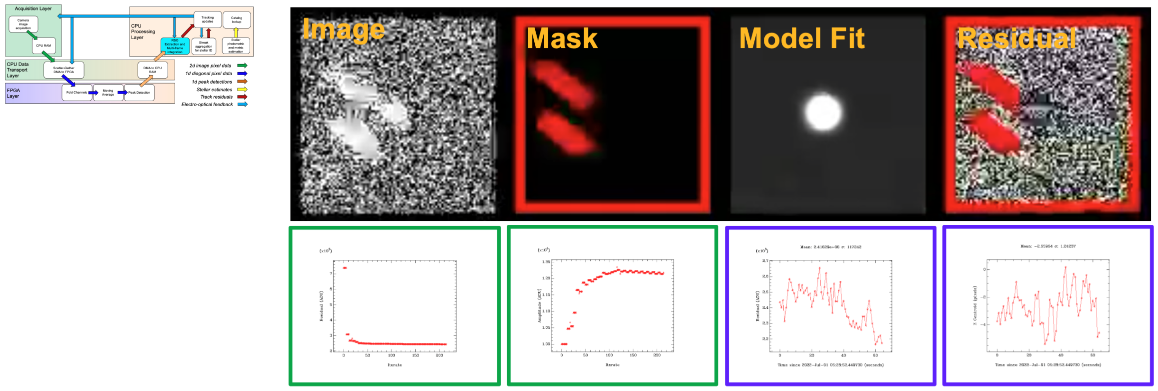


Fig. 6: Levenberg-Marquardt algorithm for fitting a Gaussian to masked pixel data in the track box for the on-sky collect on TELECOM-2D. The upper panel displays the track box image, the mask predicted from prior images, the model fit, and the residual. The lower right panels outlined in green display the fit residual and amplitude history over 250 iterations of the Levenberg-Marquardt fitter applied to the track box within a single image. The lower right panels outlined in purple display the fit residual and x-centroid over the 64 one-second exposures. These plots display two of the 5 parameters from the Levenberg-Marquardt fit: amplitude, x and y centroid, Gaussian width, and mean background. The stability of the x-centroid history as stars move through the track box illustrates the benefit of applying the mask so that background stars do not bias the estimate.

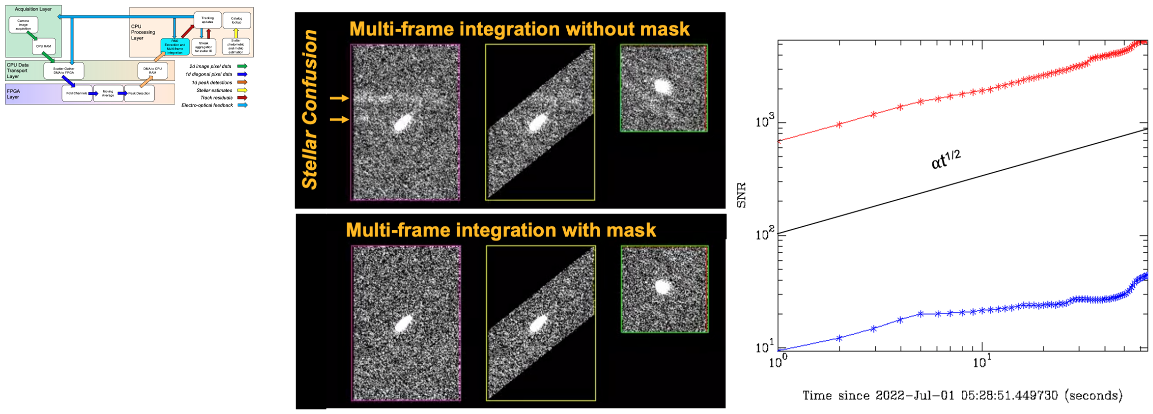


Fig. 7: Multi-frame integration of the track box. The upper left and lower left panels display multi-frame integration without and with application of the mask, respectively. The color outlines are as described in the caption of Figure 5. Integrations in the upper left panel display streaks from background stars, which have been masked in the lower left panel. The right panel shows the RSO signal-to-noise ratio (SNR) vs integration time. The blue trace quotes an SNR computed from peak Gaussian, while the red trace quotes an SNR computed from the matched filter. Roughly speaking, these correspond to whether the existence of the RSO must be discovered when the peak exceeds the noise floor, or whether its existence is known and may be found by a Gaussian matched filter. The predicted scaling law $\alpha t^{1/2}$ is also drawn for reference. This plot demonstrates how the algorithm performs multi-frame integration at the pixel level while rejecting background stars.

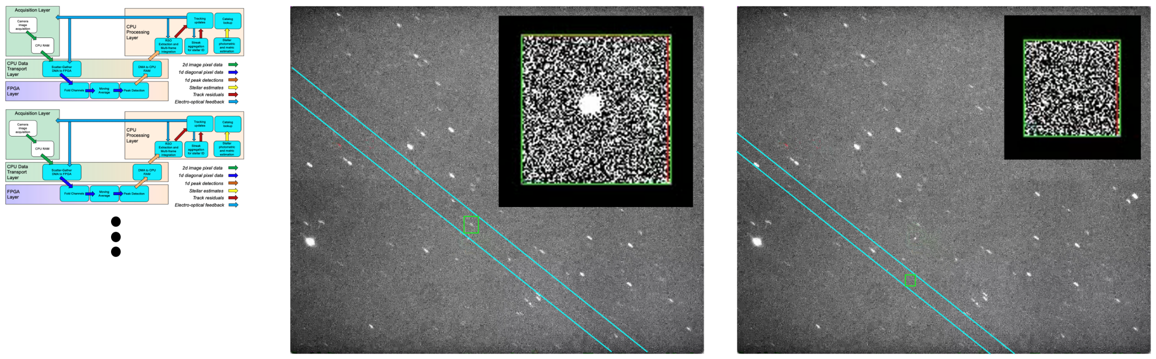


Fig. 8: Independent tracking of multiple targets. The left panel displays the track box shown in Figure 2, while the right panel displays a second track box of different size positioned elsewhere in the image. Once the image is written by the camera into CPU RAM, the raw pixel data are available to be read concurrently via multiple scatter-gather descriptor tables. Multiple processing pipelines within a single FPGA or single processing pipelines within multiple FPGAs can perform the scatter-gather DMA transfer and analyze multiple track boxes independently and concurrently. Each track box analysis may be configured independently in pixel dimension through its scatter-gather descriptor table and through the number of frames to integrate as described in Section 3.3. Each track box analysis generates RSO centroid estimates that are available for multi-frame integration and/or for EO feedback control. The vertical ellipsis at far left expresses this multi-target tracking opportunity.

sensor to observe an RSO often involves allocation of a block of time during which the observation will be attempted. Due to uncertainties in the target ephemeris, the sensor may not successfully acquire the target. Alternatively, solar lighting conditions may make the RSO appear fainter or brighter than anticipated, so that the time allocation is poorly matched to lighting conditions. If the data are to be post-processed, then no mitigating steps are possible during the observation. The onboard processing algorithm permits mitigation in real time, enabling on-board decision support. For example, to overcome RSO ephemeris uncertainty the sensor could be instructed to perform multi-frame integration to a limiting sensitivity. If no RSO is found within the track box, then the sensor could shift the track box to a new location and repeat the integration to the same limiting sensitivity. This CONOPS enables area search to a user-defined sensitivity limit. As a second example to overcome uncertainty in RSO brightness, the sensor could be instructed to terminate the multi-frame integration after the RSO signal reaches half-well depth on the focal plane array. As a third example, EO feedback control loops benefit from trading control loop rate against measurement noise to minimize the residual disturbance spectrum. This onboard algorithm permits this optimization to occur dynamically by adjusting the image frame rate based on the measured brightness of the RSO to achieve optimal control.

As an additional benefit, multi-frame integration further reduces the data volume to be transferred off-sensor: inference H in Section 2. In the TELECOM-2D collect, multi-frame integration over 64 images reduces the data volume by a factor of 64: another ∼1.5 orders of magnitude over the values listed in Table 1. In the TELECOM-2D collect, the data volume to be transferred off-sensor was greater than $500 \times 64 = 39000$. This is inference A in Section 2.

3.4 Multi-target Tracking

The combination of larger dimension focal planes and a larger RSO population drive towards conditions in which multiple RSOs may be present in the same image. Examples include groups of geostationary satellites and strings of LEO satellites whose orbital parameters differ only in mean anomaly. A natural question to pose is whether this onboard processing algorithm can track multiple RSOs simultaneously. The question becomes more complicated if the multiple RSOs exhibit relative motion. (i.e., spreading out or coalescing as the sequence of images is acquired). The onboard algorithm is well-suited to tracking multiple RSOs as long as relative motion does not carry the RSOs out of their track boxes during the collect. Figure 8 displays an example in which two independent track boxes are selected from the image, while the caption of this figure describes how the track box analysis can be performed concurrently. This is inference I in Section 2.

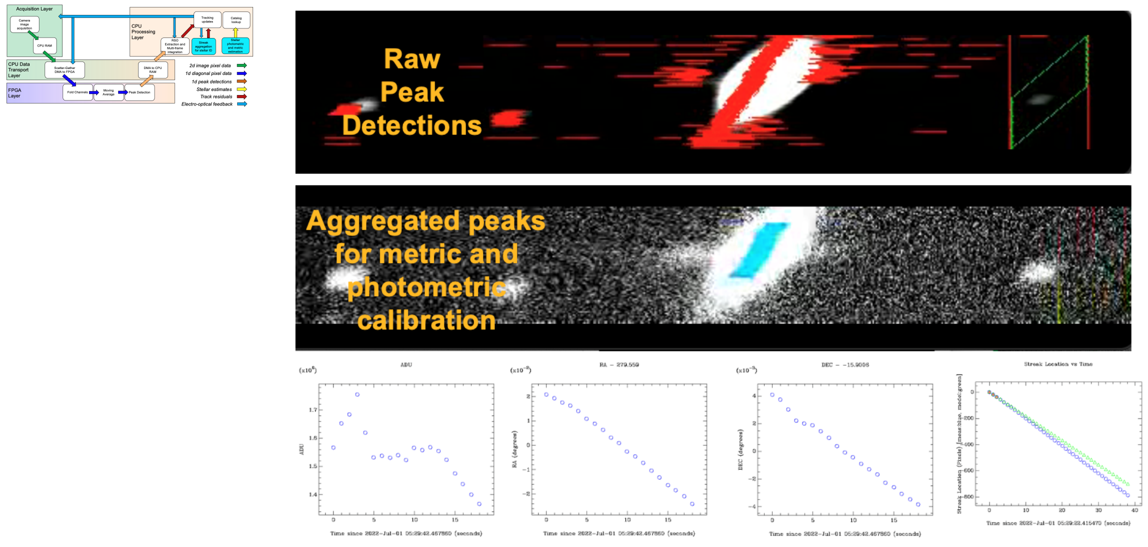


Fig. 9: Illustration of in-frame stellar calibration using the star HD172032 acquired during the TELECOM-2D collect. This star traversed the sequence of images at a location displaced by hundreds of pixels from the RSO TELECOM-2D. The upper panel shows the extracted pixel ribbon and peak detections marked in red. These peak detections are associated into the streak histories described in Section 3.2. To compute an estimate of the stellar brightness and pixel coordinates of HD172032, the streak histories are aggregated into "stellar histories" that encompass the vertical dimension of the lower panel. These stellar histories aggregate the streak histories by testing adjacency within the pixel ribbon. The resulting aggregate is marked in cyan on the middle panel. Recall that all images are shown on a clipped log stretch, and the bright pixels not marked in cyan contribute very little to the estimate. The plots at bottom show the timeseries of estimates of HD172032 brightness in ADU based on the aggregation of peak detection amplitudes, the timeseries of RA, and DEC estimates based on the aggregation of peak detection coordinates and reported telescope pointing coordinates, and the discrepancy between these RA and DEC estimates and catalog values. These plots are discussed in the text of Section 3.5.

Concurrent, multi-target tracking introduces additional CONOPS possibilities. The search to a user-defined sensitivity limit described in Section 3.3 may be prosecuted more quickly using multiple track boxes. As a second CONOPS, one track box could be assigned to an RSO and one or more additional track boxes could continuously search an area around the RSO in a patrolling mode. The pixel dimensions and sensitivity limit for each track box may be specified independently, so the sensitivity limit for patrol is independent of that for the RSO. A third multi-target CONOPS to perform absolute metric and photometric calibration using background stars is described in the next section.

3.5 Metric and Photometric Calibration

The amplitude and centroid estimates reported in Section 3.3 are in uncalibrated units: amplitudes are reported in Arbitrary Data Units (ADU), while centroids are reported in units of pixels measured from a reference point on the focal plane array. Conversion of RSO metric and photometric observations to absolute units of irradiance, right ascension (RA) and declination (DEC) requires a calibration step. This calibration may be performed through a separate observation of a calibrator star, or via indirect means such as attitude estimation from onboard star trackers or inertial measurement units.

The multi-target tracking feature of this algorithm presents an alternative approach to absolute calibration. This approach uses calibrator stars that serendipitously traverse the focal plane array during the RSO collect. In this CONOPS, a second track box is allocated for extraction of calibrator stars. When operating in rate-track mode, the sensor typically knows to high accuracy the laydown of the focal plane array on the sky. With this knowledge, the CPU can perform a catalog lookup of stars that will traverse the field. The CPU can write a scatter-gather descriptor table to extract the pixel ribbon containing this calibrator, the FPGA can process the ribbon to perform peak detections, and the CPU can aggregate to obtain streak histories. A final step in this calibration CONOPS is to aggregate streak histories into stellar histories, as described in the caption of Figure 9. The stellar catalog irradiance, RA, and DEC are then used to establish conversion factors from uncalibrated to calibrated units. These calibration factors are applied to the RSO measurements to obtain metric and photometric observations of the RSO in absolute units. This procedure may be applied repeatedly over the course of the collect, since the second track box may be deployed whenever calibrator stars serendipitously traverse the field. This is inference J in Section 2.

The lower panel of Figure 10 displays timeseries of the uncalibrated estimates for HD172032. The leftmost plot shows the estimated ADU vs time, which is stable to about 20% over the 64 image collect. This value would be converted to irradiance via catalog lookup and filter color correction. The left-center and right-center plots display the RA and DEC computed from the reported telescope pointing coordinates and the centroid of HD172032 estimated from the stellar history. The RA drifts by ±72 arcsec = 350 μrad, while the DEC drifts by ±7.2 arcsec =35 μrad.These drifts arise from errors in the reported telescope pointing coordinates. The RSO metric observations may be calibrated independently of the reported telescope pointing coordinates by reference to the catalog coordinates of HD172032. Repeated extraction of calibrator stars throughout the collect would provide cross-validation. This technique utilizes calibrators acquired contemporaneously on the same focal plane as the RSO. This degree of commonality may reduce bias errors relative to other calibration techniques.

3.6 Methods for Electro-optic Feedback Control

The algorithmic steps described in Sections 3.2 - 3.5 produce centroid estimates that may be applied for EO feedback control. In EO feedback control, a centroid estimate is used for pointing corrections that control tracking errors. Tracking errors can originate from noisy measurements, telescope mechanical jitter arising from stochastic random processes, or from systematic pointing bias errors. Either hardware or software can effect the control. Repeated cycles of measurement and control are required to reject these errors. These repeated cycles must be performed in real time at latencies that are small compared to the disturbance timescales.

This algorithm may present opportunities for EO feedback control. Telescope jitter is a disturbance that affects all targets in the field identically. Measurements of the RSO centroid described in Section 3.3 may be fed back to the telescope pointing system to correct these jitter errors. This pointing system can be the telescope gimbal, a tip/tilt secondary mirror, or another fast steering mirror in the optical train. These are examples of applying EO feedback control in hardware. In a multi-target tracking scenario, bias or jitter detected in the Levenberg Marquardt fit for each individual RSO can be corrected by shifting each track box prior to the multi-frame integration of that RSO. This is an example of applying EO feedback control in software. Alternatively, a common mode correction computed by averaging the centroids from all RSOs can be fed back to the telescope pointing system and the residual differential for each RSO may be absorbed during its multi-frame integration. This method combines hardware and software feedback control.

“Beaconless tracking” is another interesting EO feedback control method with parallels to star trackers. In this method, the algorithm guides on the background stars to stabilize tracking on a user-supplied ephemeris. The lower right panel in Figure 9 illustrates discrepancy between the modeled vs measured pixel offset in the stellar calibrator from the TELECOM-2D collect. To control this offset and ensure a particular rate, pointing corrections would be applied to increase or decrease the rate. Likewise, centroiding on stars may reveal motion of their centroids in the transverse dimension of the pixel ribbon. EO feedback can control these errors through pointing corrections. These pointing corrections may be applied in hardware by supplying commands to a gimbal or steering mirror. Alternatively they may be applied in software by frame-stacking to absorb rate error, and by rewriting the scatter-gather descriptor set to absorb transverse error. An important practical consideration lies in the relative cost between flowing pointing requirements to hardware vs. performing corrections in software: often the latter is more cost-effective.

Beaconless tracking may be useful in applications that require initiation of tracking prior to RSO detection. An example arises in satellite laser ranging, when an EO telescope must identify when the laser has illuminated the RSO. Prior to illumination the RSO is not visible to the EO sensor, but the EO sensor must initiate track and wait until the laser is on-target. Beaconless tracking allows the EO sensor to stabilize tracking against pointing disturbances prior to RSO detection. A second example arises in space-to-ground optical communication, when a ground terminal is scheduled against a comm space vehicle. With beaconless tracking, the ground sensor can stabilize tracking before the space vehicle initiates its broadcast. This can be

Table 2: Single image timing results for C++ emulation performed on a Mac Pro and preliminary timing results for the embedded implementation performed on a Zynq-7000 device operating with a 50 MHz clock. Each row of the table breaks out timing results for the steps in the algorithm. Generation of the pixel ribbon on the CPU was dominated by I/O of the FITS files storing images from the TELECOM-2D collect, while the scatter-gather DMA operation completed in 20 milliseconds. C++ emulation of the FPGA processing required 119 milliseconds, while the FPGA pipelines this operation with an effective latency less than 1 μsec.

Processing Step	C++ emulation on Mac Pro	Embedded Implementation
Generation of pixel ribbon	1000 milliseconds	20 milliseconds in each of two scatter-gather DMA channels
Pixel-level calculations shown in Figure 10	119 milliseconds	Pipelined latency less than 1 μsec
Formation of streak histories described in Section 3.2	12 milliseconds	pending
Levenberg-Marquardt fitter	560 milliseconds	pending
Multi-frame integration	14 milliseconds	pending
Formation of stellar histories described in Section 3.5	11 milliseconds	pending
Total processing time per image:	1716 milliseconds	pending

particularly important when the ground terminal is mounted on moving vehicles like ships, trucks or planes.

4. EMBEDDED SYSTEM DESIGN

This section contains status of an ongoing effort to develop the first embedded implementation of this algorithm. A Zybo z7-20 development board[8] hosting a Zynq-7000 part was selected for this effort. Software development incorporated compilation of a PetaLinux kernel, custom linux DMA driver based on the Xilinx DMA example[20], and a Vivado design. Figure 10 shows a schematic design for the custom intellectual property (IP) core that implements the operations shown in Figure 5. At the time this paper was written, this custom IP was being developed and tested in Vivado simulation and had not yet been tested on the Zybo board.

Compilation of the FPGA bitfile, PetaLinux kernel, custom Linux drivers, and custom Linux application software represents a complex software toolchain encompassing many steps. This development project used an SVN repository polled by a Jenkins continuous integration, continuous deployment (CI/CD) server[10] to compile the full toolchain on a nightly basis. Nightly builds ensured the software integrity and made the build products available to stakeholders in a timely manner. This approach also allowed stakeholders to commit changes and retrieve the resulting build products online without replicating the full build environment on a local workstation.

Figure 11 displays a screenshot of the Vivado design that incorporates two independent scatter-gather DMA channels to enable the multi-target tracking CONOPS described in Section 3.4. Each channel supports bidirectional DMA between the ARM processor and the programmable logic via an AXI interface. The DMA engines operate on memory mapped regions and signal the ARM processor via interrupt upon completion. The two blocks labeled myip in Figure 11 contain the custom IP performing the operations in Figure 10.

Table 2 presents timing results for the C++ emulation and the embedded implementation of this algorithm. The first two rows of the table encompass scatter-gather DMA transfer into the FPGA and the low-level pixel processing shown in Figure 5. The 20 millisecond/image processing duration supports the assertion that the FPGA component of the algorithm performs at 50 frames per second. Timing results for the remaining embedded processing steps were not available at the time this paper was written. Of these, the Levenberg Marquardt fit is the most time-consuming processing step performed by the CPU, requiring 560 milliseconds. If fitting is to be performed on the ARM Cortex processors, this processing step will require acceleration to

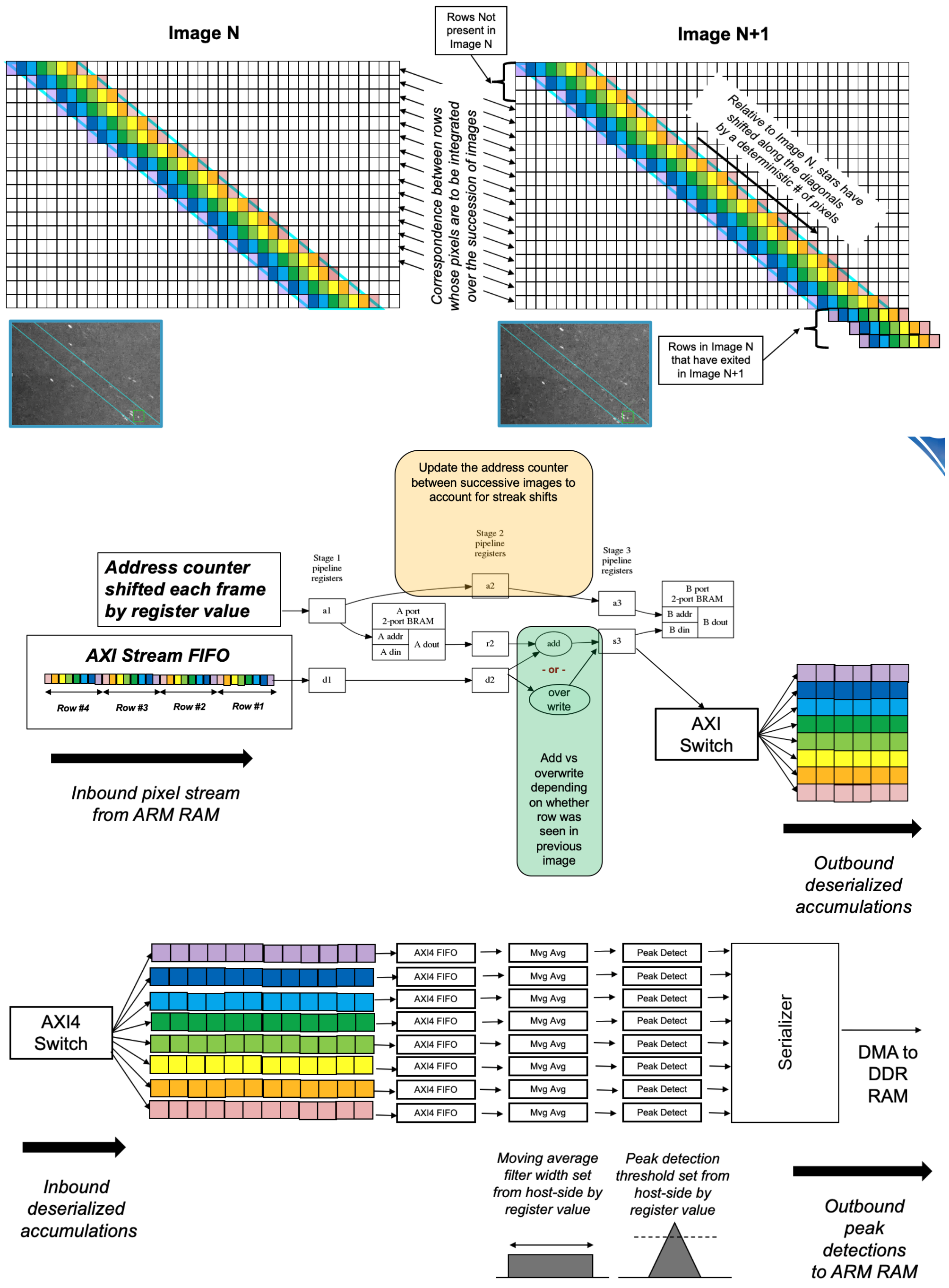


Image N
Image N+1
Rows Not present in Image N
Correspondence between rows whose pixels are to be integrated over the succession of images
Relative to Image N, stars have shifted along the diagonals by a deterministic # of pixels
Rows in Image N that have exited in Image N+1
Update the address counter between successive images to account for streak shifts
Stage 1 pipeline registers
Stage 2 pipeline registers
Stage 3 pipeline registers
Address counter shifted each frame by register value
AXI Stream FIFO
Row #4
Row #3
Row #2
Row #1
Inbound pixel stream from ARM RAM
a1
a2
a3
d1
d2
r2
s3
A port 2-port BRAM
A addr
A din
A dout
B port 2-port BRAM
B addr
B din
B dout
add
- or -
over write
Add vs overwrite depending on whether row was seen in previous image
AXI Switch
Outbound deserialized accumulations
AXI4 Switch
AXI4 FIFO
Mvg Avg
Peak Detect
Serializer
DMA to DDR RAM
Inbound deserialized accumulations
Moving average filter width set from host-side by register value
Peak detection threshold set from host-side by register value
Outbound peak detections to ARM RAM

Fig. 10: Schematic design for the FPGA portion of the embedded algorithm. The upper panel displays the pixel ribbon in image space, with boundaries in cyan. Pixels are colorized to show how they will be sorted into rows of pixels in ribbon space. In successive images the pixel ribbon shifts downwards: new pixels enter at the top and the oldest pixels fall off the bottom. The center panel displays the serialized pixel stream generated by the scatter-gather DMA transfer being folded into an accumulation, as shown in Figure 5. For each pixel in the stream, the corresponding aggregate is retrieved from FPGA Block RAM, added to or overwritten by the incoming pixel value, and rewritten into Block RAM. An AXI Switch deserializes these values into rows. In the bottom panel, the deserialized rows are processed through the moving average filter and peak detector. Values that exceed a user-defined threshold are transferred back to CPU RAM via DMA.

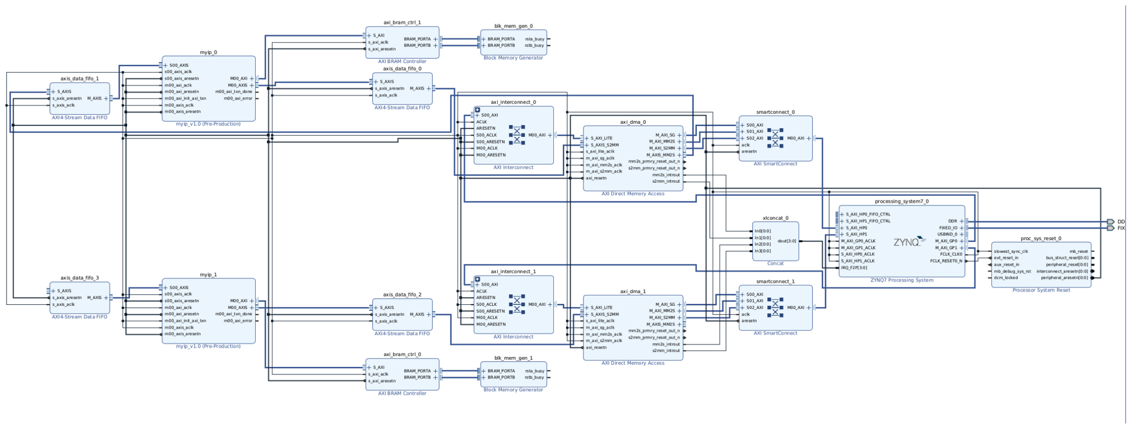


Fig. 11: Screenshot of the Vivado design incorporating two independent DMA pathways to enable multi-target tracking on a Zybo Z7-20 development board.

bring down the compute time to values comparable to the other algorithmic steps in Table 2. The Levenberg-Marquardt fitter[9] used in the C++ emulation was not multithreaded and proceeded from a cold start in each image. These simplifying choices drove up the compute time. A straight centroid computation could be performed to obtain the estimate. These represent potential opportunities for acceleration on the CPU. The Levenberg Marquardt fitter has been embedded on an FPGA for other applications[1, 11]. An embedded fitter could be used in a second FPGA acceleration stage through a DMA transfer of the track box and projected streak histories. These are future trades.

An important limitation in this embedded implementation arises from the available quantity of Block RAM on the FPGA. This FPGA resource is necessary for storage of the folded pixel ribbons, as single-cycle access to these values is required to implement the signal flow shown in the center panel of Figure 10. The Zybo Z7-20 hosts the XC7Z020-1CLG400C part that contains 4.9 Mb of Block RAM. Assuming 3 bytes per accumulated pixel, this device can store a maximum of 214,084 accumulations. In the above design these accumulations are to be allocated across two pixel ribbons. For example, the Block RAM could be evenly partitioned between the two DMA channels, with storage for two pixel ribbons of dimension 96 × 1115. This particular part offers a relatively small quantity of BRAM, and Zynq and Versal SoC's with larger Block RAM capacity are available. For example, the ZU19EG part provides 34.6 Mb of Block RAM[4].

5. CONCLUSIONS AND FUTURE DIRECTIONS

This paper concludes by correlating the six assertions from Section 1 against the material presented above. Table 3 lists these six assertions against results from the on-sky TELECOM-2D collect. These results were generated using a C++ emulation of the embedded processing algorithm, while Section 4 provides status on the first embedded implementation of the algorithm. In combination, the on-sky demonstrations and the status of the embedded implementation indicate that there are no technical impediments to realizing this embedded algorithm in hardware. Additional work remains to complete the Verilog development for the custom IP shown in Figure 10 and implement the pending rows in Table 2. With a complete embedded implementation it will be possible to evaluate the algorithm's performance against simulated focal plane array data and in on-sky collects exercising EO feedback control. This will provide quantitative results in real time to assess the algorithm under operational conditions. Ground telescopes offer the most expedient platform for initial trials.

Collectively, the six assertions in Table 3 enable autonomous decisions at the sensor and shift the Observe, Orient, Decide, Act (OODA) loop from centralized data centers to the space vehicle or ground telescope hosting the EO sensor. Edge computing and sensor autonomy provide a way to ensure that the sensor constellation architecture scales as the number of sensors grow and as the volume of data collected by each sensor increases. This argument applies more strongly to EO space sensors, where the considerations of

Table 3: Assertions from Section 1 and their correspondence to material in this paper.

Assertion from Section 1	Supporting Material in this Paper
1. Reduces data volume to be transferred off-sensor by a factor of order 10^4	Table 1 and final paragraph of Section 3.3 demonstrate data compression of 39000 over sixty-four 4-Mpix images acquired in the TELECOM-2D collect. Data volume compression ratio scales as [focal plane array size] / [pixel ribbon size], which increases linearly with the dimension of the focal plane array. Consequently, data volume compression will grow as focal plane array dimensions increase.
2. Differentiates between photons from RSOs and photons from stars, thereby eliminating confusion	Section 3.2 demonstrates the method for detecting stars before they enter the track box, while Section 3.3 demonstrates application of a pixel mask to eliminate the contribution of stars prior to RSO processing.
3. Performs multi-frame integration on stars and on the RSO at the pixel-level	Section 3.2 demonstrates multi-frame integration on stars through the folding operation shown in Figures 5 and 10. Section 3.3 demonstrates application of a Levenberg-Marquardt fitter to determine centroid offsets. These offsets are used to frame-stack the track box to the nearest pixel, yielding a multi-frame integration corrected for pointing errors. Figure 7 shows the expected scaling behavior of SNR$\propto t^{1/2}$ for the sixty-four 4MPix images in the TELECOM-2D collect.
4. Performs multi-target tracking, with independent specification of integration time for each RSO	Section 3.4 describes an embedded implementation for a multi-target tracker that deploys one processing pipeline per target, with all pipelines operating concurrently. Section 3.3 describes how a single processing pipeline controls integration time by coadding a number of track box images. The number of coadditions may be specified independently for each processing pipeline. Section 4 demonstrates a dual-target embedded design.
5. Provides EO feedback control options, including "beaconless" autoguiding to allow tracking on a user-supplied ephemeris by referencing background stars	Section 3.6 discusses EO feedback control to compensate for pointing disturbances. Options include applying RSO centroid measurements to control hardware or to frame-stack track box images. Section 3.3 presents RSO centroid measurements from the Levenberg-Marquardt fit, which can be supplied to an EO feedback control loop. Beaconless tracking uses background stars to drive the sensor so as to stabilize on a user-defined ephemeris. The lower left panel of Figure 10 shows detected drift relative to the reported telescope coordinates. This differential can be supplied to an EO feedback control loop for beaconless tracking.
6. Processes in-frame calibrators to provide real-time photometric and metric estimates in absolute units (e.g., irradiance, RA, DEC)	Section 3.5 demonstrates selection of a serendipitous calibrator star HD172032 present in a subset of the images from the TELECOM-2D collect. This section demonstrates how estimates of HD172032's uncalibrated brightness and centroid are extracted from the imagery using the same real-time processing pipeline used for the RSO in Section 3.2. Catalog lookup allows conversion to calibrated units, and this conversion may then be applied to calibrate the RSO metric and photometric observations.

off-sensor data transport, autonomous operation, and real-time decision support are far more stringent than for ground telescopes.

LICENSING INFORMATION

The techniques described in this paper are patent pending. The Aerospace Corporation may offer royalty-free licenses to non-profits, Federally Funded Research and Development Centers (FFRDCs), universities, government contractors working under federal contracts, and government agencies[7]. For licensing information, please contact ip@aero.org.

ACKNOWLEDGEMENTS

The authors thank Don Blaty and Tim Wilkinson for thoughtful peer reviews. The motivation for pursuing this line of inquiry arose from a conversation with Justin Fletcher in January 2022.

REFERENCES

[1] A. Abba, F. Caponio, A. Geraci, and G. Ripamonti. Implementation of High Efficiency Non-linear Least-squares in FPGA Devices for Digital Spectroscopy. In IEEE Nuclear Science Symposuim and Medical Imaging Conference, pages 1371–1376, 2010.
[2] AMD. Versal, July 2023.
[3] AMD. Zynq 7000 SoC, July 2023.
[4] AMD. Zynq UltraScale+ MPSoC Product Tables and Product Selection Guide, July 2023.
[5] Nicholas P. Bertrand, Derrick Cheung, Joy Gu, Chris McNamara, Camille Saidnawey, and Ernie Zenker. NGSatSentry: On-Orbit Detection System for Space Domain Awareness. AMOS Conference Technical Papers, 9 2021.
[6] Dr. Matthew C. Britton. Demonstration of Real-time Photons to Obs Algorithm at GEO with AeroTel 30cm Telescope. https://www.dropbox.com/scl/fi/ydi15qerw2c60zp0l83uz/aerospaceEdgeComputingAlgorithmOnboardProcessing.pptx, December 2022.
[7] The Aerospace Corporation. Technology Transfer at Aerospace, July 2023.
[8] Digilent. Zybo Z7 - Digilent Reference, July 2023.
[9] Eigen3, July 2023.
[10] Jenkins, July 2023.
[11] Myung-Jin Lee and Yong-Jin Jung. FPGA Implementation of Levenverg-Marquardt Algorithm. Journal of the Institute of Electronics and Information Engineers, 51:73–82, 11 2014.
[12] Michael Lim, Mousavi Payam, Jelena Sirovljevic, and Huiwen You. Onboard Artificial Intelligence for Space Situational Awareness with Low-Power GPUs. AMOS Conference Technical Papers, 9 2020.
[13] T. Alan Lovell, David Zuehlke, and Troy Henderson. Processing of Space Object Data from Optical Observers for Space Domain Awareness. In 2021 IEEE Aerospace Conference (50100), pages 1–11, 2021.
[14] Dominique Low, Christos Koulas, and Domenico Di Giovanni. Training Neural Networks to Detect Resident Space Objects using Space Based Optical Payloads and Low-SWaP Onboard Processing. AMOS Conference Technical Papers, 9 2022.
[15] Kyle Merry, Matthew Dykstra, and Brian Hacsi. Assessment of Onboard Processing Algorithms for Cislunar Space Domain Awareness. AMOS Conference Technical Papers, 9 2022.
[16] Wikipedia. Blob Detection, July 2023.
[17] Wikipedia. System on a chip, July 2023.
[18] Wikipedia. Time Delay and Integration, July 2023.
[19] Douglas Woodward, Celeste Manughian-Peter, Timothy Smith, and Elizabeth Davis. Pixelwise Image Segmentation with Convolutional Neural Networks for Detection of Resident Space Objects. AMOS Conference Technical Papers, 9 2021.
[20] Xilinx. Linux DMA From User Space 2.0, July 2023.