NEAR EARTH BODY DETECTION AND LINKING

P. Rajan[1] *P. Burlina*[1,2] *M. Chen*[2] *B. Jedynak*[3] *N. Mehta*[2] *A. Sinha*[4] *G. Hager*[1]

The Johns Hopkins University
[1]Dept. of Computer Science, [2]Applied Physics Laboratory
[3]Dept. of Applied Mathematics and Statistics [4]Dept. of Electrical and Computer Engineering

ABSTRACT

Most asteroid population discovery has been accomplished to date by Earth-based telescopes. It is speculated that most of the smaller Near Earth Objects (NEOs), up to 140 meters in diameter, whose impact can create substantial city-size damage, have not yet been discovered. Many asteroids cannot be detected with an Earth-based telescope given their size or the location of the Sun. Our objective is therefore to develop an efficient asteroid detection and linking algorithm that can be hosted on-board a spacecraft. By having on-board algorithms, the system would also minimize the need to downlink entire images taken by a space-based telescope. We describe an image processing algorithm that we have developed for onboard asteroid detection and characterize its performance.

***Index Terms*—** Asteroids detection, identification.

1. INTRODUCTION

NASA has a congressional mandate to discover all Near-Earth Objects (NEOs) at least 1 kilometer in diameter. Fortunately, 95% of the NEOs larger than 1 km that have been discovered are likely not to impact Earth. Near-Earth Object search programs [1] are currently almost exclusively accomplished by Earth-based telescopes such as MIT's LINEAR [2] project, the NEAT [3] program, the Catalina Sky Survey, or Pan-STARRS [4]. An exception is JPL's spacecraft-based WISE telescope brought out of hibernation to characterize NEOs in the 3.4 and 4.6 micron infrared bands [5] (called NEOWISE).

It is notable, however, that the NEO that impacted Chelyabinsk, Russia on 15 February 2013 was only about 17 meters. The impact of a 50 meter asteroid that caused the Tunguska Event of 1908 could have destroyed an entire city or metropolitan area. It is estimated that only a relatively small fraction of those so called "city-killing" asteroids, particularly objects less than 140 meters in diameter, have been discovered to date. Because of their size, atmospheric effects and the location of the sun, some of these NEOs cannot easily be detected with an Earth-based telescope.

This work was supported by an Early Stage Innovations grant from NASA's Space Technology Research Grants Program NNX14AB04G.

Our focus here is therefore on developing an algorithm that can be hosted on-board a spacecraft. Understanding that there are processing and resource (memory) constraints on-board, our algorithm design needs to not only meet the performance objectives of detecting and identifying asteroids using a space-based telescope, but it also needs to have a small footprint to be feasible with the limited on-board resources. This paper describes an agile algorithm candidate that is being investigated as well as our use of representative real and simulation imagery for testing it.

Notable prior work includes [4], describing a reference processing system for detection and identification of asteroids named the Pan-STARRS Moving Object Processing System (MOPS). This pipeline aims at identifying moving objects in our solar system and linking those detections within and between night observations. It attributes those detections to known objects, calculates initial and differentially corrected orbits for linked detections, recovering detections when they exist, and orbit identification. Most proposed pipelines for Earth-based detection include a step to combine images into a high S/N static-sky image that is subtracted from the current master image to obtain a difference image containing only transient sources. Examples include [6], where a shift-and-add technique is used to improve signal to noise ratio and then synthetically creating long exposure images to facilitate the detection of trajectories. A related shift-and-add method using a median image rather than an average image is reported in [7]. A match filter is used for asteroid detection and matching in [8]. In [9, 10, 11], tree based searches (including KD-trees) are used for efficient linking of successive asteroids detections and finding sets of observation points that can be fitted with an inherent motion model, through an exhaustive search for all possible linkages that satisfy the model constraints.

This paper describes a small-footprint pipeline (with regard to memory and CPU usage) that can be deployed on a flight-qualified hardware. When compared to prior work, the salient features of this study are its focus on space-based applications. In particular, (a) we use a combination of Principal Component Analysis and 2d-trees to efficiently link detections into trajectories (see Section 2 describing our approach); (b) we employ a simulation engine relying on principled optics models to characterize performance on realistic space-based imagery, along with a combination of real image datasets (NEAT and CSS) (Section 3).

2. APPROACH

The main components of our image processing pipeline addressing detection and linking of asteroids and taking as input a sequence of time-lapse images acquired from space-based platforms is now described (also detailed in Algorithm 1):
Image Pre-Processing Median filtering is applied to reduce impulsive noise and artifacts that depend on the acquisition.
Image Registration Image registration is used to bring the images into alignment with a common image of reference so that the stars in the background line up in all images. In the case of a triplet of images, the second image is used as reference. We use a similarity transformation (translation, rotation, scaling) and/or skew (full affine transformation) to align all images in the sequence. Our registration uses mutual information [12] to estimate the transformation parameters.

Image Logical Differencing A global thresholding is applied to the registered image for detecting asteroids and suppressing background noise. As asteroids are typically very faint compared to the surrounding stars, the selection of the detection threshold impacts false alarm rate. Thresholding yields binary detection images. The set of all binary images is then used to generate an intersection image that contains objects that occur in at least two images in the sequence. This is followed by logical differencing whereby we produce a set of difference images by intersecting the corresponding binary image with the negative of the common intersection image. This operation provides a list of candidate detections (movers) for each image in the sequence. While the logical differencing results in good detections, additional artifacts such as crater-like formations (see Fig. 5) are seen as a result of some celestial bodies being over-exposed. To mitigate this artifact, we also find connected components and perform filtering of hollow objects as well as filtering based on object size.

Trajectory Linking The list of centroids of moving objects obtained from image differencing defines a set of candidate rectilinear trajectories. The goal is to find a subset of centroids that fit a linear model. The set of filtered centroids potentially has a high number of noisy points (falsely detected movers), and the cardinality of the set of all candidate trajectories increases exponentially with the number of detections, thus requiring subsequent pruning. We therefore combine PCA and 2d-trees to efficiently find trajectories. Unlike MOPS [4] and CSS[13], we do not set an upper limit on the velocity of the asteroid, and hence do not risk missing fast movers. Given a sequence of images, we form all the possible trajectories connecting the detections in the first and last frames. There are $O(n^2)$ such trajectories, where n is the number of detections per image. We then calculate the points of intersection of each of these trajectories with all the frames in the middle. A binary space partitioning 2 dimensional tree is constructed from the candidate detections for each of the middle frames(excluding the first and last frames), and we perform a range search on the trees to find a subset of points that lie near the point of intersection in each frame. This query can be done in $O(\log(n))$ time on average. Once we find a collection of such points that potentially form linear trajectories, we perform PCA and compute the ratio of eigenvalues to develop a line confidence score for each candidate trajectory, and choose lines that have high confidence. Using the candidate trajectories thus found, we then enforce temporal ordering by using the sign of the projection on the principal eigenvector. As a final step, we eliminate false positives by ensuring that the distance between projections is proportional to the time interval between images. Compared to a brute force line search of $O(n^3)$ for a triplet of images, our algorithm takes $O(n^2 \log(n))$ time.

Algorithm 1 Detection and Linking Algorithm for a sequence of images

```
1: Median filter all images in the sequence.
2: Align all the images to a reference frame (e.g. the second image
   for a triplet) using Mutual Information based registration.
3: Perform thresholding on the registered images using a pre-
   defined threshold to generate binary detection images.
4: Form the logical intersection image from all the binary detection
   images, then perform logical differencing between each detec-
   tion image and the negative of the intersection image.
5: Subsequently filter out hollow objects to produce a set of candi-
   date moving objects on each image.
6: Form the set of all trajectories connecting the candidate moving
   object detections from the first and last frame.
7: Construct 2d-trees for the difference images corresponding to
   the frames in the middle, excluding the first and last frames.
   Each node of the tree stores the bounding box of all its children.
8: for all trajectories connecting the first and last frames do
9:     Find the points of intersection of each trajectory with the
       middle frames.
10:    Perform range search on the 2d-trees to find a list of detec-
       tions near this point of intersection within each difference image.
11:    Perform PCA on this subset of trajectories connecting points
       from all frames in the sequence. Find the ratio of eigenvalues,
       $\lambda_1/(\lambda_1 + \lambda_2)$. Reject trajectories for which this ratio is below
       a line confidence threshold. Ensure that the distance between
       projections is proportional to the time interval between images.
12: end for
```

3. EXPERIMENTS

We detail the experiments performed using a range of simulated space-based imagery generated using the JHU/APL developed *Renderer and Camera Emulator* (RCE) as well as real imagery from the NEAT dataset.

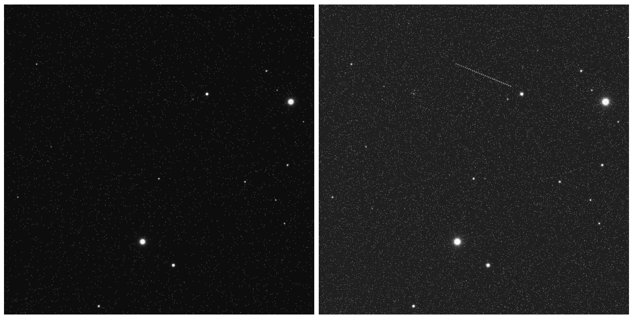

Fig. 1. Left: An image simulated by RCE. Right: 31 simulated MWIR images super-imposed in order to visualize the trajectory of the asteroid in a single image. The true trajectory can be seen as a faint line towards the top center of the image

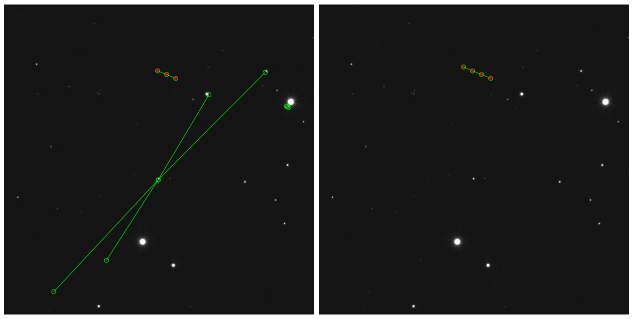

Fig. 2. The trajectories found by the pipeline are shown in green. The true location of the asteroid is marked in red. Left: Trajectory Detection on a simulated triplet. Right: Trajectory Detection on a quadruplet. Adding one more image to the triplet eliminates the false positives.

3.1. Simulated Space-Based Imagery

Using RCE, asteroids are modeled as spherical blackbody-like emitters, with a cross-sectional area that approximates the sizes of actual asteroids and surface temperatures typical of sun-illuminated asteroids in an Earth-like orbit. Similar to the way stars are modeled, the radiation emitted is modeled using a form of Planck's equation:

$$B_\lambda(T) = \epsilon \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda k_B T}} - 1} \tag{1}$$

where $B_\lambda(T)$ is the spectral radiance at a given wavelength λ and temperature T The value ϵ is the emissivity of the asteroid, which essentially converts the blackbody spectral radiance into spectral irradiance. The constant, h is the Planck's constant, c is the speed of light, λ is wavelength, k_B is the Boltzmann constant, and T is the temperature.

The asteroids are assumed to have a nominal temperature of 200 K due to solar heating and emissivities in the range from 0.9 to 0.98. The RCE uses stellar data available as part of the Two Micron All Sky Survey (2MASS), a stellar survey that scanned the entire sky in three IR bands (centered at 1.25 µm, 1.65 µm, and 2.17 µm, respectively). The 2MASS catalog also incorporates data in two visible bands from other surveys. An example of the simulated MWIR image and the ground truth trajectory derived from one set of simulated imagery is shown in Fig. 1. Fig. 2 shows the final trajectories detected by our algorithm for one triplet and one quadruplet of the simulated MWIR dataset. As is shown in Fig. 2, using trajectory verification on a greater number of images in the sequence allows us to quickly disambiguate and reject false trajectories. In this case this trend is readily apparent when going from a triplet to a quadruplet of images.

We characterize the algorithm with regard to detections at each of the following three successive stages: the initial detection of objects, the detection of moving objects and the detection of trajectories. In Fig. 3, we plot the number of detected objects at each step of the pipeline for each selected threshold value. We can verify that the number of detections monotonically decreases when the threshold increases. Note that we have found that this not always the case, especially since a higher number of detections at the thresholding stage may induce more cancellations at the logical differencing. Additionally, we note in the right plot in Fig. 3 that the use of quadruplets allows for a single trajectory to be found irrespective of the value of the threshold used (and hence the number of detections found at the first stage), echoing the results displayed in Fig. 2. Last, we show the Receiver Operating Characteristic (ROC) for simulated space-based imagery in Fig. 4.

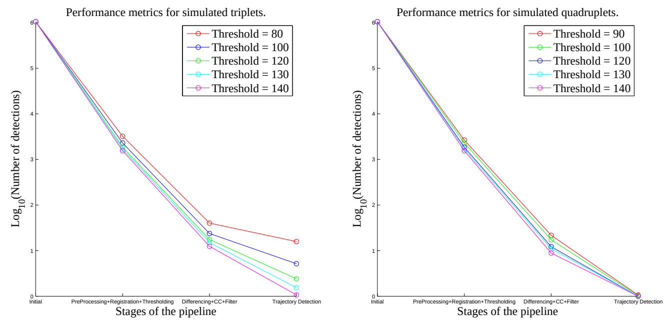

Fig. 3. Left: The number of detections at various stages of the pipeline for triplets of images. Right: The number of detections at various stages of the pipeline for quadruplets of images.

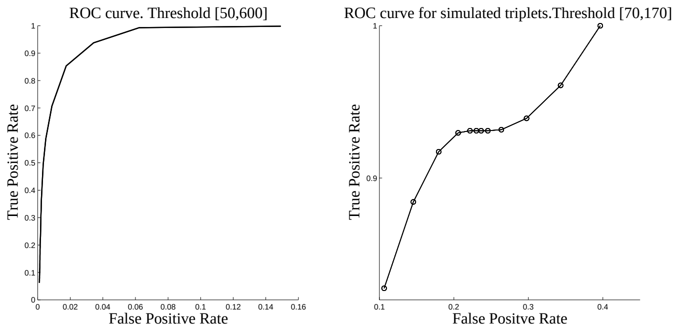

Fig. 4. Left: ROC curve for the detection of stars and asteroids after the Image thresholding stage of the pipeline. Right: ROC curve for asteroid detections at the final stage of the pipeline.

3.2. Real Imagery

3.2.1. NEAT

Near Earth Asteroid Tracking (NEAT) [3] is an earth-based program run by NASA from 1995-2007 to discover NEOs. Fig. 5 and Fig. 6 show the results at all stages of the pipeline for one triplet of images of the 2002-CY46 asteroid obtained from the NEAT system archive.

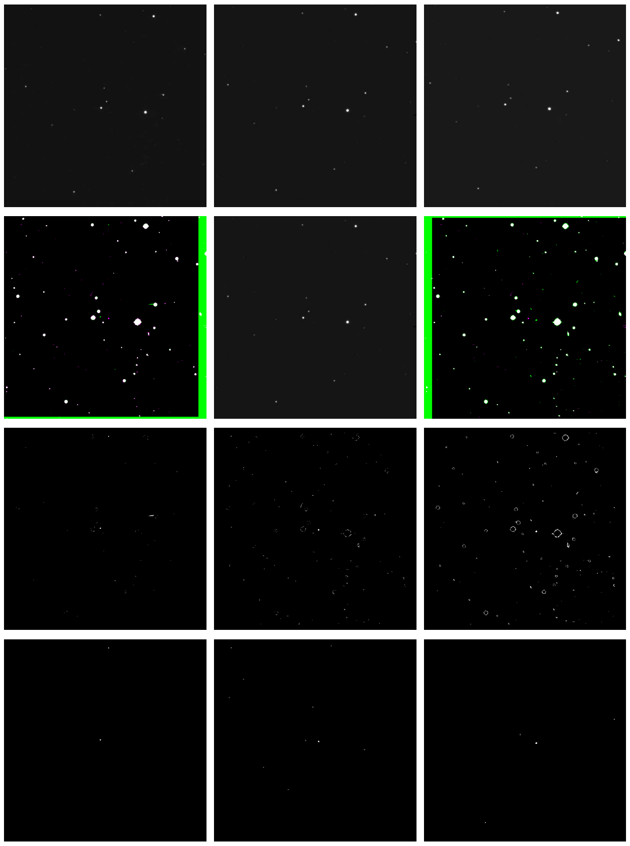

Fig. 5. Image Processing Pipeline results shown on 2002 NEAT data. (row 1): input image triplet (CY46) taken approx. 10 minutes apart. (row 2): Image Registration. Left: Image-1 registered to Image-2. Right: Image-3 registered to Image-2. (row 3): Image Differencing: Artifacts such as crater-like formations are seen in the difference images above. This is the result of some celestial bodies being overexposed.) (row 4): Image Differencing: Filtered centroids shown in each image of the sequence.)

3.2.2. CATALINA

The Catalina Sky Survey (CSS) [14, 15] is intended to discover NEOs, specifically potentially hazardous asteroids that pose risk to earth. Fig. 7 shows the final result for one set of earth-based images from CSS.

3.3. Implementation

This pipeline has been developed and tested in MATLAB and has also been jointly implemented in C++ on a recent Intel

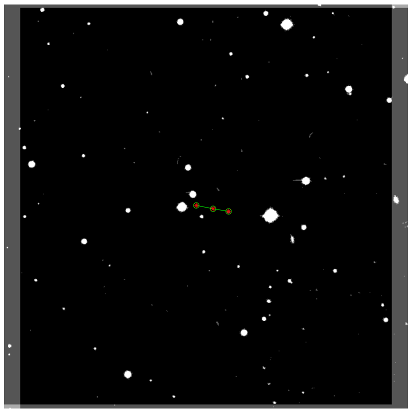

Fig. 6. Trajectory Detection for the NEAT CY46 Triplet. Asteroid trajectory detected is shown in green. True location is in red. 3 Images of the triplet are super-imposed here after registration and thresholding for ease of visualization.

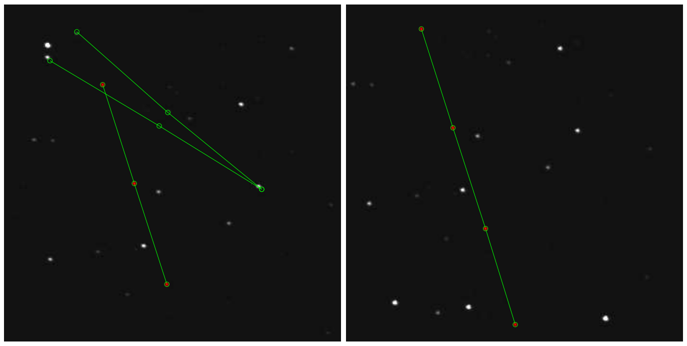

Fig. 7. Image Processing results shown on CSS data.The registered images are super-imposed to visualize the trajectory in a single image.Asteroid trajectory detected is shown in green, true location of the asteroid is in red. Left: Trajectories detected on a triplet. Right: Trajectory detected on a quadruplet.

processor. Preliminary benchmarking on this processor has shown that peak memory usage is of the order 92 MB. Additional preliminary studies for extrapolating CPU and memory usage on an MCP750 – a good proxy for BAE's RAD750, an operational spacecraft processor – have led to metrics that seem promising for a possible deployment on a spacecraft architecture. We are now conducting additional work to validate these figures including activities involving the deployment of this algorithm onto a MCP750 in conjunction with FPGAs.

4. CONCLUSION

Detecting smaller *city-killer* asteroids in space-based surveys is important, as these are not easily seen in Earth-based surveys. We describe a pipeline for on-board NEO detection and characterize its performance on simulated space-based as well as real imagery and show that quadruplets offer much higher performance with little incidence on complexity. Performance results and preliminary computational requirements are promising with regard to the possible future deployment of this algorithm on spacecraft processors such as the RAD750.

5. REFERENCES

[1] Grant H Stokes, Jenifer B Evans, and Stephen M Larson, "Near-earth asteroid search programs," *Asteroids*, vol. 3, pp. 45–54, 2002.

[2] Jenifer B Evans, Frank C Shelly, and Grant H Stokes, "Detection and discovery of near-earth asteroids by the linear program," *Lincoln Laboratory Journal*, vol. 14, no. 2, pp. 199–215, 2003.

[3] NASA, "The NEAT survey," 2014.

[4] Larry Denneau, Robert Jedicke, Tommy Grav, Mikael Granvik, Jeremy Kubica, Andrea Milani, Peter Vereš, Richard Wainscoat, Daniel Chang, Francesco Pierfederici, et al., "The pan-starrs moving object processing system," *Pan*, vol. 125, no. 926, pp. 357–395, 2013.

[5] NASA, "The NEOWISE survey," 2014.

[6] Michael Shao, Bijan Nemati, Chengxing Zhai, Slava G Turyshev, Jagmit Sandhu, Gregg Hallinan, and Leon K Harding, "Finding very small near-earth asteroids using synthetic tracking," *The Astrophysical Journal*, vol. 782, no. 1, pp. 1, 2014.

[7] Toshifumi Yanagisawa, Atsushi Nakajima, Ken-ichi Kadota, Hirohisa Kurosaki, Tsuko Nakamura, Fumi Yoshida, Budi Dermawan, and Yusuke Sato, "Automatic detection algorithm for small moving objects," *Publications of the Astronomical Society of Japan*, vol. 57, no. 2, pp. 399–408, 2005.

[8] Peter S Gural, Jeffrey A Larsen, and Arianna E Gleason, "Matched filter processing for asteroid detection," *The Astronomical Journal*, vol. 130, no. 4, pp. 1951, 2005.

[9] Jeremy Kubica, Joseph Masiero, Andrew W Moore, Robert Jedicke, and Andrew Connolly, "Variable kd-tree algorithms for spatial pattern search and discovery," *Robotics Institute*, p. 253, 2005.

[10] Jeremy Kubica, Andrew Moore, Andrew Connolly, and Robert Jedicke, "A multiple tree algorithm for the efficient association of asteroid observations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 138–146.

[11] Jeremy Kubica, Larry Denneau, Tommy Grav, James Heasley, Robert Jedicke, Joseph Masiero, Andrea Milani, Andrew Moore, David Tholen, and Richard J Wainscoat, "Efficient intra-and inter-night linking of asteroid detections using kd-trees," *Icarus*, vol. 189, no. 1, pp. 151–168, 2007.

[12] Paul Viola and William M Wells III, "Alignment by maximization of mutual information," *International journal of computer vision*, vol. 24, no. 2, pp. 137–154, 1997.

[13] NASA, "The Catalina Sky Survey," 2014.

[14] S Larson, J Brownlee, C Hergenrother, and T Spahr, "The catalina sky survey for neos," in *Bulletin of the American Astronomical Society*, 1998, vol. 30, p. 1037.

[15] AJ Drake, SG Djorgovski, A Mahabal, E Beshore, S Larson, MJ Graham, R Williams, E Christensen, M Catelan, A Boattini, et al., "First results from the catalina real-time transient survey," *The Astrophysical Journal*, vol. 696, no. 1, pp. 870, 2009.

5.1. Appendix I: details on Memory and CPU usage characterization

We detail here information regarding CPU and memory usage and their implications for deployment on a reference proxy spacecraft platform such as the MCP750.

The pipeline was developed and tested on MATLAB initially and then implemented also in C++ for doing a benchmark analysis using certain performance metrics. The C++ code was run on a 8-core Intel processor (3rd generation) with clock speed of 2.3 GHz and an Intel VTune Amplifier XE 2015 profiler was used. The peak CPU usage was 12%, peak private memory and peak working set memory usages were 99,212 kB and 84,740 kB respectively. The CPU time observed was 1.267 sec (spin time = 0.465 sec, effective time = 0.802 sec). A single CPU core was used during the 0.802 sec "effective time", and the processor was idle during the spin time. Therefore "spin time" has been used in the subsequent calculations. Some additional measurements made for the Intel processor are as follows: Instructions executed = 2,868,100,000, CPI (cycles per instruction) = 0.581, CPU frequency ratio(ratio between actual and nominal CPU frequency) = 0.573. These metrics were then extrapolated for a reference processor having maximum clock speed of 466 MHz (MCP750). We estimate performance time using the relation:

$$\tau = I \times IPC^{-1} \times C^{-1} \tag{2}$$

$$IPC = 1/CPI \tag{3}$$

where τ is the performance time, I is the total number of instructions executed, IPC is the instructions per cycle, C is the clock rate of the processor, CPI is the cylces per instruction. Here we assume that the total number of instructions executed by both processors are same. The performance time on the MCP750 can be predicted as

$$\tau_{MCP} = \frac{\tau_{Intel} \times IPC_{Intel} \times C_{Intel}}{IPC_{MCP} \times C_{MCP}} \tag{4}$$

Using $\tau_{Intel} = 0.802$ sec, $IPS_{MCP,max} = 525 \times 10^6$, we get,

$$IPC_{MCP,max} = \frac{IPS_{MCP,max}}{C_{MCP}} = \frac{525 \times 10^6}{466 \times 10^6} = 1.13 \tag{5}$$

If we assume 100% processor utilization on MCP750 then, $IPC_{MCP} = 1.13$. Thus,

$$\tau_{MCP} = \frac{0.802 \times 0.12 \times 0.581^{-1} \times 2.3 \times 10^9 \times 0.573}{1.13 \times 466 \times 10^6} = 0.414 sec \tag{6}$$

This could further be extrapolated to find performance time on MCP750 for other values of CPU utilization.

In the above calculation the CPU frequency ratio measured on the Intel processor has been multiplied to make use of the nominal CPU clock frequency on that processor. We predict the memory usage to remain same on MCP750 as that in our test. This is acceptable for our application given MCP750 has up to 256 MB dynamic RAM and that we can still have sufficient memory available for additional necessary processes such as navigation, guidance, communication etc onboard. Here the specifications of the MCP750 have been referred from the processor **??** and the measurements on the Intel processor were done using 'Open Hardware Monitor v.0.7.1-beta','Speedfan v.4.5' and 'Pc-Wizard v.2014.2.13'.

% CPU Utilization	Performance time(sec)
50	0.828
12	4.968
10	4.14

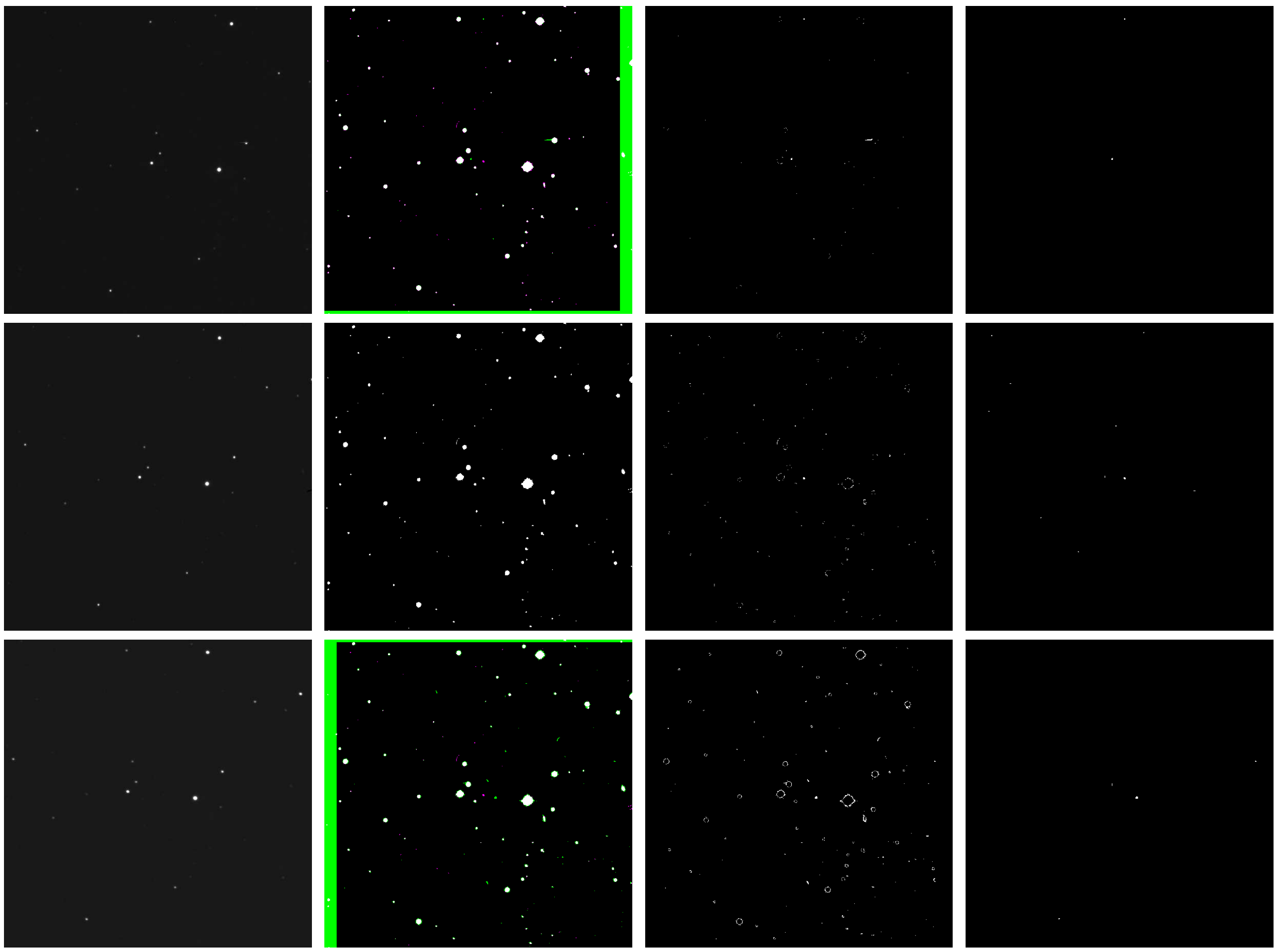

Fig. 8. Image Processing Pipeline results for the CY46 Triplet. (column 1): 2002 CY46 Triplet images taken approximately 10 minutes apart. Near Earth Asteroid Tracking (NEAT) system archive. (column 2): Image Registration results for the CY46 Triplet. Top: Image-1 registered to Image-2. Middle: Image-2 Bottom: Image-3 registered to Image-2. (column 3): Image Differencing results for the CY46 Triplet. (Artifacts such as crater-like formations are seen in the difference images above. This is the result of some celestial bodies being over-exposed.) (column 4): Image Differencing results for the CY46 Triplet. Filtered centroids in each image of the sequence.)