Asteroid Detection using a Single Multi-Wavelength CCD Scan

Jonathan Melton
Multidimensional Image Processing 12/5/15

Abstract

This paper details an algorithm used to find asteroids in a single CCD scan. The algorithm works by finding the centroids of the points of light in the first component of an RGB image and comparing these to the centroids in the last component. As there is a 5 minute delay between the two components, an object moving with sufficient speed will be in two different places in the two images. This algorithm was used on a set of 49 images obtained from the Sloan telescope in New Mexico, and performed reasonably well when used on images with a known asteroid content.

1) Background - Why is asteroid detection important?

The recent meteor strike in Chelyabinsk, Russia, raised public awareness of the danger of orbital debris impacting the Earth. However, asteroid detection has been a NASA priority since the late 1990’s and a large amount of work has been performed both on the hardware and software of sky survey telescopes to better locate and characterize asteroids. In addition to finding potentially dangerous asteroids, space mining startups like Planetary Resources and Deep Space Industries are searching for asteroids that may contain valuable resources for use in space or back on Earth. These corporations have an interest in finding and cataloging asteroid positions and types to find which ones may be best suited for mineral extraction.

Though it has been publicly popularized by recent events, asteroid hunting has been going on for quite some time. There are large datasets of sky images available from transient sky surveys like NEAT (Near Earth Asteroid Tracking) catalogue, Pan-STARSS and the Sloan Digital Sky Survey. These data sets were searched for asteroids and other stellar objects before being archived, but recent advances in image processing have made it possible to comb through the images using improved algorithms to find potential asteroids that may have been missed before.
Where do you find Asteroids?

There are 3 large collections of asteroids in the inner solar system, categorized by where they appear (Figure 1). Main belt asteroids (MBAs) are arrayed between the orbits of Mars and Jupiter, and comprise the largest number of asteroids in the solar system. Most MBAs have stable orbits that keep them a safe distance from the Earth. A subset of MBAs is NEAs, which can have highly elliptical orbits that bring them close to the Earth at some point in their orbit. These asteroids are often MBAs that have been pushed from their original orbit by a gravitational perturbation from one of the more massive planets like Jupiter [1].

![Figure 1: Positions of largest groups of asteroids in the inner solar system. Asteroid positions based on information from the JPL and Minor Planet Center, image retrieved from https://en.wikipedia.org/wiki/Asteroid_belt](image)

A second variety is the Jovian Trojan and Greek asteroids, which occur in the orbit of Jupiter in clusters leading and following the planet at 60 degree increments. These asteroids lie near the L4 and L5 Lagrangian points of Jupiter, where the interaction between the gravity fields of Jupiter and the Sun balance out and make an extremely stable pocket [2].

Hildas asteroids orbit the center of the solar system in a triangular pattern caused by gravitational resonance with the motion of Jupiter. While they are a consequence of the large planet’s gravitation, they never approach close enough to be fully deflected by Jupiter.

It is an interesting consequence of the formation of our solar system that the belts of asteroids are extremely dependent upon the position of the largest and most massive planet in our
system, Jupiter. At times (like during the late heavy bombardment) Jupiter has acted as a slingshot to direct asteroids and other debris into the inner solar system, pummeling the inner planets. However, it is also theorized that after stabilization of the solar system Jupiter has acted as a shield, deflecting comets and other Oort cloud objects away from the inner planets, and thus protecting the Earth.

The most common type of asteroid is the main belt asteroid, and they are also the most interesting to us here on Earth. Not only are they close enough to find and potentially profit from, they are also the most likely to have an orbit that may intersect that of the Earth at some point in the future. [1]

2) Image Processing Methods used in this Paper

Image Subtraction as a Method of Finding Moving Objects

Image subtraction is a convenient method to find objects that may be moving between images. If two images of the same scene, such as Figure 2, are subtracted, the result should be a matrix of zeros. However, if something moves between the two images the object will appear twice in the subtracted image; once in its original position as a negative, and again in the new position as a positive.

![Figure 2: Example of moving object detection using image subtraction. The blue arrow signifies the direction of movement.](image)

While there is little apparent motion in the first and second images, moving objects can be easily identified by subtracting the two images. In addition, one can determine the direction of motion using the sign of the artifacts in the image. Black indicates position in the first image, while white indicates position in the second image.

This applies to astronomical photos as well in principal. If two images are taken of the same star field with a slight delay in time between the images, subtraction will remove all the static stars.
Moving objects such as asteroids should remain as a series of dots. In reality this is harder than it sounds; astronomical instruments are extremely precise and try to image extremely dim objects, which can lead to a low signal to noise ratio. Figure 3 shows the result of subtracting two registered images in the data set I retrieved. Even though the two photos were only taken 5 minutes apart, there is a considerable amount of noise in the shape of the objects.

![Figure 3: Result of subtracting two components of the same image from the Sloan data set. The subtraction contains a substantial amount of noise.](image1)

While not an insurmountable problem, this can make it difficult to build a program to automatically find asteroids with 100% accuracy.

**Connected Component Labeling**

When objects are displayed against a black background, connected component labeling can be used to find details about the objects in an image. For example, a threshold is applied to the coins in Figure 4. A connected component labeler is then used on the binary image to find the centroids of these points.

![Figure 4: Example of a threshold and connected component labeling](image2)

**Histogram Analysis**

A histogram can be used to analyze the intensity of an image’s components over a certain region of the image. Histograms are generally used to determine the quality of light distribution over the allowable range of the image, and to determine how best to change the distribution to
improve contrast. In this algorithm histogram analysis was used to determine if an object moved between images. This process will be explained in more detail later in the paper.

3) Methods used by Professional Telescopes

The NEAT asteroid detection system is housed at the Maui Space Surveillance Site on the island of Hawaii. The telescope and system operates autonomously, surveying the sky once per month and relaying the results to the Jet Propulsion Laboratory in Pasadena, CA. The NEAT camera instrument is a 4096x4096 CCD that is attached to the focus of the Maui telescope [3].

The telescope operates in a “survey mode,” which means it scans the sky sequentially in an effort to discover moving objects. A search field is determined before observing begins, and through the night the telescope passes over each area of the field 3 times to get a set of 3 images of the same night sky, called a “triplet.” The program attempts to monitor as much of the sky as possible, but areas close to the galactic plane are avoided because the high density of stars leads to false positives.

At the same time as the observations, an onsite computer runs the images through a series of image processing programs to look for moving objects. The first program, REMDARK, subtracts a baseline reading for darkness taken with the shutter on the camera closed from the images to normalize them. STARCAT finds objects in the triplet. The program registers the three images in the triplet, then averages and subtracts the individual images to look for movement. If the program finds moving objects larger than a specified size it records information such as intensity, pixel count and orbital parameters. Several other programs then perform checks to ensure the objects are not noise due to weather, that their positions do not correspond to the location of known objects, and that the orbital parameters are within reason. If the object successfully passes all these checks it is passed on to the scientists at Pasadena for a visual check for object confirmation.

Objects that pass the computer program are organized into 9 “patches.” The position of the object is cropped out of each image in the triplet set in which it appears, as well as a similar crop of the remaining two images in the position in which the asteroid appeared in the third. These patches are then analyzed by a human to confirm the object (Figure 5).
The patches can be used to confirm the object; if the object is in a different frame in each column it is indeed moving through space, and is not some sort of other artifact. The human checker would then submit this object to the Minor Planet Center for classification after the orbital parameters have been determined and checked to ensure it is a new object.

Pan-STARRS is another wide field sky scanning survey that uses twin 1.8m telescopes with 7 degree fov to image the sky using 1.4 gigapixel cameras. This setup can image the entire sky visible from the summit of Mauna Kea in less than a week [4]. The telescope takes multiple pictures of the same patches of sky as it moves, which allows for transients to be discovered. The camera uses six passbands to search for objects of interest at different wavelengths of light.

To find asteroids in these images, the Pan-STARRS consortium developed a custom program called the Moving Object Processing System (MOPS). This process starts with by processing images received from the telescope, then proceeds to object detection, orbit determination and automatically submits potential discoveries. The creators of the program wanted to build an entire ecosystem to determine the efficiency of the asteroid detection process from beginning to end. The Pan-STARRS system was an important contributor to NASA’s goal of discovering 90% of NEAs, which most likely happened in 2011.

The image process starts by obtaining a four exposure ‘quad,’ 4 images of the same location in the sky with ~15 minute gaps between exposures. The images in the quad are subtracted and morphology is used to determine moving objects. A point of light that has more than 3 different
positions in the 4 images is considered to be a potential object. Every potential object is vetted by a human to ensure it is correct before being submitted to the MPC.

The human is needed in the image processing loop because the number of false detections can be quite high. While it inconvenient and time intensive to weed through images looking for false detections, it means that small, dim and otherwise hard to find asteroids can be found. **Figure 6** shows asteroids and false detections as a ratio of the signal to noise ratio of the object in the image and the magnitude (intensity of light) of the object for the MOPS program. Grey dots are asteroids, black dots are false detections. A definite arc of positive detections can be seen, with a tail of false detections interspersed with a few real detections. This illustrates the difficulty in building a completely automated process to find asteroids, as for low signal to noise ratios it can be difficult even for humans to tell the difference between what is real and what may be an artifact.

![Figure 6: Graph of asteroids and false positives from the Pan STARRS survey. At low signal to noise ratios asteroids and false positives can be difficult to discern.](image)

**3) Open Questions and Active Research**

Most telescopes used for sky surveys are small because of budgetary constraints on the use of large telescopes. As a result, there is a lower limit on the size and distance of objects that can be seen because of either pixel size or luminosity of the object. The number and composition
of small distant asteroids in the main belt and Trojans is currently an open question. These small asteroids do not reflect much sunlight and are difficult to detect for current moving object survey telescopes. While these small main belt asteroids generally do not pose a threat to the Earth because their orbits are quite stable, their positions are important as they could have valuable resources that could be mined or pose a problem for navigation of scientific equipment through the asteroid belt.

The Subaru Telescope is an 8m wide field optical telescope on the summit of Mauna Kea, Hawaii. During the commissioning of this world class instrument in 2000 a series of test images of the asteroid belt were taken before the telescope was fully operational. 27 potential asteroids were found in the test images by subtracting the test images and manually examining them for moving objects. After determining the orbital parameters of the objects and cross-referencing the Minor Planet Service Asteroid Database, all 27 objects were previously undiscovered [5].

After making a series of assumptions scientists at the Subaru telescope found a negative linear relationship between asteroid size and population size. Because of gravitational accumulation of larger asteroids and some of the consequences of collisions between smaller members of the belt, they theorized that there may be a small number of sub-km asteroids in comparison to larger bodies. As a result, this type of investigation using large, high powered telescopes may be unnecessary to find most asteroids in the main belt. However, this investigation was not statistically robust because of the small sample size. A more through follow-up investigation with another large telescope could bring interesting results.

4) My method

While the most common professional method of asteroid detection is subtraction of images separated by large amounts of time (>10m), I thought it would be interesting to try something slightly different. Sky scanning telescopes such as Pan-STARRS and Sloan search the sky at several different wavelengths of light. Because of how these wavelengths are recorded, objects can be seen to move within a single image, not just between multiple images. Sky scans could be sped up considerably if moving objects could be discovered in a single pass, instead of multiple passes as are currently normal. To try this method, instead of subtracting different images at the same wavelength I used images from the Sloan Digital Sky Survey taken during a single pass with several wavelengths of light. I then used connected component labeling to try to find movement between the images created at different wavelengths. The details of my method follow.
How the Sloan Digital Sky Survey Works

For this project I tested the asteroid detection approach of looking between time differentiated frames of the same patch of sky for moving objects. I wrote a program in MATLAB to analyze a series of images retrieved from the Sloan Digital Sky Survey website. The Sloan survey uses 30 2048x2048 pixel CCD cameras in a rectangular array (Figure 7). 6 cameras are aligned with each of 5 filters, r’, i’, u’, z’ and g’. Each filter has a passband for a different wavelength of light, as detailed in the chart in Figure 7. The camera works by scanning across the sky in the row direction, which means each point in the scan sequentially passes through each of the 5 filters. Light is integrated across the filter during transit of an object, and the time each object spends in the field of view of a filter is 54.1s. Passage of an object over the entire array is ~5.7 minutes. These sky scans are connected, and specific parts can be analyzed. [6], [7]

![Figure 7. Cartoon of the configuration of the Sloan Telescope filter array and the corresponding wavelength bands of each filter. Retrieved from http://cas.sdss.org/](http://cas.sdss.org/)

While distant objects such as stars and galaxies remain relatively stationary during the telescope scan, closer objects such as asteroids move quickly enough to be picked up in separate locations by different filters. As can be seen in Figure 7, the r’ and g’ filters are on opposite sides of the instrument. The r’ filter images the sky between t=0s and t=54.1s, while the g’ filter images the sky between t=287.9s and t=342s. This provides a separation of 4.8 minutes between the images taken by each filter. Moving objects in the image can be found by
comparing the images taken by the $r'$ and $g'$ filters; if something is moving it will not be in the same position in both images.

**Image selection and pre-processing**

To test this, I retrieved 49 images from the Sloan Digital Sky Survey from the 14$^{th}$ survey during the 5$^{th}$ data release. An example image from the dataset is shown below (Figure 8). The images used in this project come from the $r'$, $i'$, and $g'$ filters. For ease of use, the images will be referred to as green ($r'$) red ($i'$) and blue ($g'$). As with many things in astronomy, the colors used in the image do not necessarily match the naturally occurring colors, but this is not necessary for analysis.

![Figure 8: Night sky image retrieved from the Sloan Digital Sky Survey](image)

The $r'$ and $g'$ filters have the most time between them, so these two filters are stripped out of the 3 dimensional image for further processing. These images are slightly noisy at low intensities so a threshold was applied to remove background noise. While this threshold may remove smaller, dimmer objects, the main purpose of this program is to look for larger objects that could be of interest for mining or planetary protection. Next, very large objects such as galaxies and extremely bright stars are also removed. Finally, the image is morphologically opened using two 2 pixel structuring elements (one vertical, one horizontal) to remove noise left over from the pre-processing. This leaves two binary images of mid-sized objects (Figure 9).
This step is repeated twice in the program, once at a threshold of 50, and again at a threshold of 20. This double threshold allows the program to find objects that have different levels of luminosity, or that might be noisy or removed in the other threshold. By using this double threshold method the number of potential objects discovered in the image set was increased by 42%. This result is examined in greater detail in the Experiment section.

Next, the centroids of all the objects in each image are found using a connected component labeler, “regionprops()” (Figure 10). The Euclidean distance between each centroid is found by passing the coordinates of the centroids past each other in a loop, and any two points in the two images separated by more than 3 and less than 20 pixels are saved as a potential objects. In the figure green dots denote the centroids of the first image, blue dots signify centroids of the second image, and red dots indicate points at which there is a difference between the two centroids indicating a possible object.
Next, a check is run to ensure that the objects in the two images are unique. That is to say the point exists in the r’ image, but not in the g’ image, or vice versa. This ensures that two stars which are close together and appear in both images are not picked up as a potential object. A first pass checks to ensure the coordinates of each object is unique in list of coordinates for both objects. After similar points have been discarded, a line is generated between each of the remaining sets of potential objects (Figure 11). A histogram is taken along this line, and the program analyzes the histogram for movement (Figure 12).

Figure 11: Lines connecting two objects of interest in the image. A histogram is taken over this line allowing more analysis to be performed on the objects.

Figure 12: Two histograms showing the difference between an asteroid and a set of stars. The program would automatically reject the star histogram.
Figure 12 shows the histogram results for an object and a non-object. A moving object has a profile in the r’ and i’ filters (green and red), and a secondary spike in the g’ (blue) filter. A non-object has similar intensities in all 3 filters. The second object was most likely picked up because the intensity of the g’ filter in one object is small, and was most likely filtered out in the pre-filtering process. The secondary check ensures this kind of object is not recorded as a potential object by the program.

Finally, now that the potential objects in the image have been identified and checked a final image is produced by the program showing the locations of the potential objects outlined with a red box (Figure 13).

![Figure 13: Final image showing potential objects boxed in red. The pop out shows what an asteroid looks like in the original image.](image)

Two moving objects were found in Figure 6. As can be seen from the object in the pop-out, the location of the maximum intensity of the object in the r’ filter (green spot) and g’ filter (blue spot) are separated by a number of pixels when the two images are combined. This object is moving with considerable speed relative to the observer, as a difference in location can be noticed even with the i’ filter.

Program Organization

This program was written as 2 separate functions and a control script. The control function, ‘Asteroid_hunter_control’ loads the list of objects, then calls the ‘asteroid_hunter’ function to analyze each image individually. The functionality of the ‘asteroid_hunter’ was explained above in this section. Once ‘asteroid_hunter’ has found the potential objects in an image it prints the
original image with red boxes to file and passes the number of potential objects and the number of rejected objects found in the image back to the control program. The control program develops a list of the images with the number of objects in discovered which is then printed to file. The final function is a check function called by ‘asteroid_hunter’ as part of the secondary object check.

5) Experimentation

Because these images are quite noisy a threshold was used in an attempt to improve the signal to noise ratio. However, the results of the program are highly reliant upon the chosen thresholding level used in the images. Table 1 shows the variation in objects found when the threshold is applied at several different values.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>40</td>
<td>6</td>
</tr>
<tr>
<td>45</td>
<td>12</td>
</tr>
<tr>
<td>50</td>
<td>14</td>
</tr>
<tr>
<td>55</td>
<td>7</td>
</tr>
<tr>
<td>70</td>
<td>8</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: Number of objects found using different values for the original threshold

A threshold of 50 finds the most objects and only returns one false hit. This level of thresholding seems to have the greatest balance between trying to find dim objects and removing enough noise to clearly differentiate the on the image. As a compromise, the final program runs two separate thresholds, one at 20 and one at 50. This allows for a larger range of objects to be correctly identified, but also increases the number of false positives.

6) Results

This program was able to identify 20 moving objects in the set of 49 images, and only found 3 false positives. In images in which I knew the number of potential objects, the accuracy of the program is 100%. A list of the images in the data set and the number of moving objects discovered in each image follows as Table 2.
Asteroid Hunter Output

<table>
<thead>
<tr>
<th>Image</th>
<th>Potential Objects</th>
<th>Rejects</th>
<th>Image</th>
<th>Potential Objects</th>
<th>Rejects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sloan_example1.jpg</td>
<td>1</td>
<td>2</td>
<td>Sloan_example27.jpg</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Sloan_example3.jpg</td>
<td>0</td>
<td>13</td>
<td>Sloan_example28.jpg</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Sloan_example4.jpg</td>
<td>2</td>
<td>6</td>
<td>Sloan_example29.jpg</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Sloan_example5.jpg</td>
<td>2</td>
<td>3</td>
<td>Sloan_example30.jpg</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Sloan_example6.jpg</td>
<td>1</td>
<td>5</td>
<td>Sloan_example31.jpg</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Sloan_example7.jpg</td>
<td>0</td>
<td>7</td>
<td>Sloan_example32.jpg</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Sloan_example8.jpg</td>
<td>0</td>
<td>6</td>
<td>Sloan_example33.jpg</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Sloan_example9.jpg</td>
<td>0</td>
<td>10</td>
<td>Sloan_example34.jpg</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Sloan_example10.jpg</td>
<td>1</td>
<td>7</td>
<td>Sloan_example35.jpg</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Sloan_example11.jpg</td>
<td>0</td>
<td>9</td>
<td>Sloan_example36.jpg</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Sloan_example12.jpg</td>
<td>1</td>
<td>8</td>
<td>Sloan_example37.jpg</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Sloan_example13.jpg</td>
<td>0</td>
<td>6</td>
<td>Sloan_example38.jpg</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sloan_example14.jpg</td>
<td>0</td>
<td>12</td>
<td>Sloan_example39.jpg</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Sloan_example15.jpg</td>
<td>0</td>
<td>4</td>
<td>Sloan_example40.jpg</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Sloan_example16.jpg</td>
<td>0</td>
<td>1</td>
<td>Sloan_example41.jpg</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Sloan_example17.jpg</td>
<td>0</td>
<td>1</td>
<td>Sloan_example42.jpg</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Sloan_example18.jpg</td>
<td>1</td>
<td>4</td>
<td>Sloan_example43.jpg</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Sloan_example19.jpg</td>
<td>0</td>
<td>4</td>
<td>Sloan_example44.jpg</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Sloan_example20.jpg</td>
<td>0</td>
<td>6</td>
<td>Sloan_example45.jpg</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Sloan_example21.jpg</td>
<td>0</td>
<td>8</td>
<td>Sloan_example46.jpg</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Sloan_example22.jpg</td>
<td>0</td>
<td>6</td>
<td>Sloan_example47.jpg</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Sloan_example23.jpg</td>
<td>0</td>
<td>7</td>
<td>Sloan_example48.jpg</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Sloan_example24.jpg</td>
<td>0</td>
<td>5</td>
<td>Sloan_example49.jpg</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sloan_example25.jpg</td>
<td>0</td>
<td>6</td>
<td>Sloan_example50.jpg</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Sloan_example26.jpg</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: List of results from the set of test images. Rejected objects are objects initially flagged as interesting in the program, but then rejected by the check function.

Table 2 also shows the check program is working properly. In each image there were many places in which a possible object was discovered, but by examining the histogram difference a large number of objects could be automatically rejected without human intervention.

7) Discussion

This program seems to work reasonably well for finding large, bright and fast-moving objects in astronomical images. Moving objects can be found by comparing images taken by different wavelength filters separated by a period of time, and looking for differences between the
images. By looking for centroids in the binary images it was simple to find repeat objects and the method seems to be fairly robust to error. Every asteroid the author was able to find by visual analysis was also picked up by the program, though there are most certainly more objects in the images.

It would not appear that this is a better method of moving object detection than simple subtraction of images with a longer time difference between them. A few of the weaknesses of this method are discussed below.

First, the thresholding used at the beginning of the pre-processing removes extremely dim objects. Because the images used were JPEGs instead of the standard astronomical FITS image container they should have already been dark subtracted (the process of removing sensor noise by subtracting an image with the lens closed). However, the images were very noisy. The thresholds used helped to remove this noise and increase the robustness of discoveries, but the number of potential finds were sacrificed in an attempt to keep the ratio of finds to false hits high.

The wide field of view of the images is also a liability when searching for smaller objects. Large numbers of bright objects are picked up in each image, which have the effect of washing out dimmer objects. A survey with a smaller field of view may be able to pick out dimmer objects because of greater contrast between the asteroids and dim stars.

Because the images used are not taken at exactly the same wavelength, differences in object composition could cause a difference in the luminosity between different images. If the object does not reflect as much light in the range of the g’ filter as it does in the r’ filter, a potential object could be missed because it looks as though it is not moving.

Another potential issue is the short time delay between images. The maximum time delay between the r’ filter and g’ filter is 4.8 minutes. While this allows enough time for a fast moving object to be clearly determined, slower moving objects or very dim objects do not display much movement between frames. For example, while the r’ filter is quite distinct for the asteroid in Figure 14, the g’ filter is slightly blurred, meaning that the centroid does not appear to have moved between the two images. Using a set of images with a greater time separation would show greater movement for dim objects such as this one. Originally my program could not pick up the dim object in Figure 14, but by running multiple thresholds I was able to increase the ability of the program to find objects such as this one. There may be more small objects hiding in the images that the program was unable to find.
8) Conclusions

Looking for asteroids by using image subtraction is effective, but prone to errors due to the simple fact that the objects one is looking for are often barely brighter than the background. Institutional searches have access to full uncompressed images of the night sky and intimate knowledge of the telescope parameters, leading to an effective asteroid detection process that has discovered over 90% of estimated NEAs. However, even with this technological prowess humans are still needed to manually check each image to ensure it is an actual asteroid.

The program developed in this paper uses a single compressed JPEG image with light at 3 different frequencies and a time delay between the first and third frequencies of ~5 minutes. A threshold is applied to the first and third slices of the images and the centroids of the remaining objects in the two slices are compared to find moving objects. This program attempts to use a single image to find objects instead of a series of images to speed up transient surveys. A larger percentage of the sky could be imaged each night if only one image was required to find moving objects.

However, while this function is effective at finding bright, fast moving objects, the combination of thresholding and the small time delay between the images make it a less effective tool for asteroid detection than the method currently employed by sky surveys.
References:


Code:

```matlab
%% Asteroid hunter control program
% Written by Jonathan Melton

% This is a control program for the asteroid hunter function. It reads a
% list of image names from at text file (the images need to be in the
% matlab path as well) then runs the asteroid hunter program in a loop
% through the requested images. The program saves a picture with potential
% moving objects highlighted, and outputs the number of moving objects. The
% number of objects in each image is saved to a text file.

%Clear workspace
close all
clear, clc

%Read list of images for program to run through from .txt file
image2 = textread('C:\Users\chilly\Google Drive\Mines Google Drive\Matlab\Asteroid Project\Asteroid Input\Image_list.txt', '%s', 'delimiter', '
');
%image2 = {'Sloan_example48.jpg'};
obj = zeros(size(image2,1),2);

%Create array for potential objects

%Run program for each image
for i = 1:size(image2,1)
    %Program output is number of potential objects discovered in each
    %image, place in a matrix to keep track
    [obj(i,1),obj(i,2)] = asteroid_hunter( char(image2(i)) );
    clc
    disp('% Complete: '), disp( round(i/size(image2,1)*100,0))
end

%Write data to a table
T = table(image2, obj);
 writetable(T,'C:\Users\chilly\Google Drive\Mines Google Drive\Matlab\Asteroid Project\Asteroid Output Images\Object_Count.txt','Delimiter','	')

sprintf('Number of Potential Objects Found: %f', round(sum(obj(:,1)),0))
sprintf('Number of Images Examined: %f', size(image2,1))

function [ num_obj, hits ] = asteroid_hunter( I_name )
```
%Asteroid Hunter: Input name of image to be analyzed, output number of
%objects
% Processes photos retrieved from the Sloan Digital Survey website to
% look for moving objects. Uses a multi-step process detailed below
%
% 1) Takes color image and strips out colors, thresholds to remove noise,
%    then removes large objects from the images by using an opening
%    morphological operator. (This assumes the large objects are not
%    moving in the image)
% 2) Finds the centroids of the remaining objects in each image, then
%    compares the locations of centroids in the blue and green images.
%    This is because the blue image has a slight delay compared to the
%    red and green images. If an object is moving in the foreground
%    (asteroids, aliens, secret military projects) there will be a blue
%    spike in intensity trailing the red and green spikes. The program
%    finds locations at which a green and blue spike are slightly offset.
% 3) Next a line is built between the two objects and an intensity
%    histogram is taken along the line, allowing a comparison to be made
%    between the two spikes in intensity. If the two spikes follow a set
%    of conditions, the object is added to a list of potential objects.
%    The conditions are: The green and blue spikes must be sufficiently
%    far from one another, the red and green spikes must lie sufficiently
%    close to each other, and the blue spike should be at least twice as
%    large as the red and green values in that area. (this is to prevent
%    two objects that are not moving but have funny looking intensity
%    curves from being picked up as potential objects.)
%
% If the object fulfills the conditions its location is placed in a matrix,
% and rectangles are superimposed on the original image to show the
% locations of the possible objects. This image is saved to file, and the
% function terminates.
%
% The output variable (num_obj) of this function is the number of asteroids
% found by the program.

%% Processes an asteroid photo to bring out the stars and reduce noise

%I1 = imread('Sloan_example.jpg');
[I1, map] = imread(I_name);
IR = I1(:,:,1);
IG = I1(:,:,2);
IB = I1(:,:,3);

%Threshold images
thresh = 50;
Ibt = IB > thresh;
Igt = IG > thresh;

%Remove large objects (Assuming they are not asteroids)
Ibt = logical(Ibt - bwareaopen(Ibt,50)); %Open image with large disk to
remove large objects
Igt = logical(Igt - bwareaopen(Igt,50)); %Open image with large disk to
remove large objects

%Open image with a small line to remove tiny objects
Ibt = imopen(imopen(Ibt,strel('line',2,0)),strel('line',2,90));
Igt = imopen(imopen(Igt,strel('line',2,0)),strel('line',2,90));
%imshow(Ibt)

%Find the centroids of the remaining objects in the image
b = regionprops(Ibt,'centroid');
g = regionprops(Igt,'centroid');

%Create matrix of centroid positions
blue_centroids = cat(1, b.Centroid);
green_centroids = cat(1, g.Centroid);

%Create matrix of possible matches
p_match = zeros(50,3);
hits = 0;

%Find Similar points within 10 pixels of one another
for i=1:size(blue_centroids(:,1),1)
    for j = 1:size(green_centroids(:,1),1)
        %Find distance between centroids
        dist = sqrt((blue_centroids(i,1) - green_centroids(j,1))^2+(lue_centroids(i,2) - green_centroids(j,2))^2);

        %Search Array for similar hits
        %If distance is more than 3 and less than 20, it is a possible asteroid
        if dist > 3 && dist < 20
            %Search Array for similar hits
            c1 = isempty(find(green_centroids(j,1) <= blue_centroids(:,1)+2 &
                           green_centroids(j,1) >= blue_centroids(:,1)-2,1));
            c2 = isempty(find(green_centroids(j,2) <= blue_centroids(:,2)+2 &
                           green_centroids(j,2) >= blue_centroids(:,2)-2,1));

            if c1 == 1 || c2 == 1 %The locations of the centroids are unique
                %Determine intensity curves near the potential asteroids
                c = polyfit([blue_centroids(i,1),green_centroids(j,1)], [blue_centroids(i,2),
                             green_centroids(j,2)],1);
                x = linspace(blue_centroids(i,1)-20,green_centroids(j,1)+20);

                %Decrease line length
                x1 = x(50) + 20/sqrt(1+c(1)^2); %X coordinate of 20 pixel line
                x = linspace(x1,x(100) - (x1 - x(50)));
                y = c(1)*x + c(2);

                %Create 3 color intensity profile along the line connecting two points
                figure, improfile(I1,[x(1),x(100)],[y(1),y(100)]) %Displays a plot of the profile
                [CX,CY,C] = improfile(I1,[x(1),x(100)],[y(1),y(100)]);
                check = check_hist( CX, CY, C );
                hits = hits+1; %Record number of non-object hits
```
% Make sure the peaks of the function are well distributed

if check == 1
    % If so, the object passes the tests, it is a potential asteroid. Record position in the p_match array
    p_match(i,:) = [dist, green_centroids(j,1), green_centroids(j,2)];
    % Show the histogram as a visual check
    figure, improfile(I1, [x(1), x(100)], [y(1), y(100)])
end
end
end
end

% Find the number of moving objects found in the image
num_obj = nnz(p_match)/3;

% Plot objects
figure, imshow(Ibt)
hold on
plot(blue_centroids(:,1), blue_centroids(:,2), 'b*') % Blue Centroids
plot(green_centroids(:,1), green_centroids(:,2), 'g*') % Green Centroids
plot(p_match(:,2), p_match(:,3), 'r*') % Potential objects
hold off

% If the program identified objects, print the picture with objects highlighted
if num_obj > 0
    figure, imshow(I1, []) % Plot original Picture
    size_rec = size(I1,1)/100; % Make the rectangle 1/100 of the image
    for i = 1:size(p_match,1) % Put red boxes around objects
        rectangle('Position', [p_match(i,2)-size_rec/2, p_match(i,3)-size_rec/2, size_rec, size_rec], 'EdgeColor', 'r');
    end

    % Record picture with objects circled
    print(sprintf('C: \Users\chilly\Google Drive\Mines Google Drive\Matlab\Asteroid Project\Asteroid Output Images\A_%s', I_name), '-dpng')
end

close all

end
function [ check ] = check_hist( CX, CY, C )
%check_hist Checks histogram to ensure it's a moving object
% This function receives the histogram taken along a line between two
% objects from the 'asteroid hunter' program and checks to make sure it
% fulfills a number of conditions to be considered as a potential object:
% 1) the blue peak and green peaks of the object (corresponding to the g'
% and r' filters) need to be separated by at least 3 pixels
% 2) The blue peak must be at least 1.5 times greater than the green and
% red peaks to be considered as an object. This ensures that objects in
% which all three peaks are similar are rejected.
% The program then returns a true or false (1 or 0) as to whether the
% object has passed the check.

% Set check to false
check = 0;

% Separate Histogram into color components
[yr, xr] = max(C(:,:,1));
[yg, xg] = max(C(:,:,2));
[yb, xb] = max(C(:,:,3));

% Make sure the peaks of the function are well distributed
if xb > xg + 3 || xb < xg - 3 % Is the blue peak more than 3 pixels away?
    if yb > C(xb,:,2)*1.5 && yb > C(xb,:,1)*1.5 % Is the blue peak the highest by 1.5?
        check = 1; % If the preceding arguments are true, set check to true
    end
end