# Asteroid Detector – Principle of operation

pfr

September 23, 2014

## Introduction

I only heard about the contest until it was already well under way, so I made the following decisions in order to save time:

- The algorithm is based on a Bayesian maximum-likelihood approach derived from an a-priori model, so that it requires little tweaking and no training.
- I made no attempt to exploit the bonus points awarded for flagging Near-Earth Objects.

For the sake of concision, many details have been left out.

## 1 Processing pipeline

### 1.1 Glow removal – `computeUnglow()`

This step discards the contribution of the atmosphere to the image, in order to make the image of a dark sky as close as possible to zero. This is necessary in order to make comparison of signal levels between different frames meaningful. We simply decompose each frame in small blocks, compute the median of each block to infer the glow at the center of the block, and subtract the interpolated glow from the image. Each frame is processed independently because there is a lot of temporal variation.

### 1.2 Layer separation – `computeLayers()`

This step separates the image into three layers:

- **star**: a constant layer, with varying position determined by the position of the stars
- **dark**: a constant layer, with varying position determined by the position of the CCD sensor
- **mov**: the movement layer, i.e. the remainder of the image after **star** and **dark** subtraction

We would like to find the minimum of the convex energy functional

$$E = \sum_{i,j} |\mathbf{star}_{ij}| + \sum_{i,j} |\mathbf{dark}_{ij}| + \sum_{i,j} \sum_{f \in F_{ij}} |\mathbf{mov}_{fij}|$$

which we approximate by exact minimization of **star** and **mov** when **dark** is fixed to 0, followed by exact minimization of **dark** and **mov** when **star** is fixed to the previously found value. Each exact minimization boils down to a simple median. Bicubic sampling is used for image transformations.

## 1.3 Filtering – `computeFiltered()`

### Offset removal

This step refines the glow removal already performed: we can do more aggressive filtering now that each layer has been separated. By modeling the offset according to a certain prior distribution, we obtain the following robust linear filter:

$$\mathbf{mov}'_f = \mathbf{mov}_f - \frac{(\mathbf{mov}_f \cdot \mathbf{w}_f) \star B}{\mathbf{w}_f \star B}$$

where

- $\mathbf{w}_{fij}$ is the inverse of the standard deviation $\sigma^{\text{offset}}_{fij} = \sqrt{\sigma_0^2 + k^2 \mathbf{dark}^2_{ij} + \mathbf{mov}^2_{fij}}$
- $\star B$ denotes convolution with a box filter
- $\cdot$ denotes pointwise multiplication.

We apply the same processing to the **dark** layer to obtain $\mathbf{dark}'$.

### Pre-detection

This step does as much filtering work as possible so that the detection step will only have to examine a single pixel per frame to evaluate the likelihood of an object trajectory.

For each pixel in the movement layer, we will want to compare the likelihood of the hypothesis that there is an object at this point with the likelihood of the null hypothesis.

In the null hypothesis, each point is simply white noise: $\mathbf{mov}_{fij} \sim N(0, \sigma^2_{fij})$.

In the asteroid hypothesis, in addition to noise we will observe a point source convolved by the telescope's point spread function. Because our approach is robust to errors in the estimation of the PSF, we assume for simplicity that the PSF is a Gaussian kernel $K$ with radius $\mathbf{blur\_radius} = 0.95$.

Suppose that we are investigating the presence of an object around $(0, 0)$. In the asteroid hypothesis, there exists some brightness $b \geq 0$ such that

$$\mathbf{mov}_{fij} - bK_{ij} \sim N(0, \sigma^2_{fij})$$

Assuming $\sigma_{fij} \approx \sigma_{f00}$ close to the object center, the maximum-likelihood value for $b$ is

$$b = \max\left(0, \frac{(\mathbf{mov}_f \star K)_{0,0}}{(K \star K)_{0,0}}\right)$$

and we have the likelihoods

$$\begin{aligned}
-\log \mathcal{L}_{\text{null}} &= \sum_{i,j} \frac{\mathbf{mov}^2_{fij}}{2\sigma^2_{fij}} - \log \mathcal{L}^0_{\text{null}} \\
-\log \mathcal{L}_{\text{asteroid}} &= \sum_{i,j} \frac{(\mathbf{mov}_{fij} - bK_{ij})^2}{2\sigma^2_{fij}} - \log \mathcal{L}^0_{\text{asteroid}} \\
&= \sum_{i,j} \frac{\mathbf{mov}^2_{fij}}{2\sigma^2_{fij}} - \max\left(0, \frac{(\mathbf{mov}_f \star K)^2_{0,0}}{2(K \star K)_{0,0}\, \sigma^2_{f00}}\right) - \log \mathcal{L}^0_{\text{asteroid}}
\end{aligned}$$

Therefore the log likelihood ratio is

$$G_{f00} = \log \frac{\mathcal{L}_{\text{asteroid}}}{\mathcal{L}_{\text{null}}} = \max\left(0,\ \frac{(\mathbf{mov}_f \star K)^2_{0,0}}{2\|K\|^2\,\sigma^2_{f00}}\right) + G^0 = \frac{\max(0, z_{f00})^2}{2} + G^0$$

where

$$z_{fij} = \frac{(\mathbf{mov}'_f \star K)_{ij}}{\|K\|\sigma_{fij}}$$

$$\sigma_{fij} = \frac{\sqrt{\|K\|^2\sigma_0^2 + e^2_{\text{star}}(\mathbf{tstar}_f \star K)^2_{ij} + e^2_{\text{dark}}(\mathbf{dark}' \star K)^2_{ij}}}{\|K\|}$$

$e_{\text{star}}$ and $e_{\text{dark}}$ reflect the fact that **star** and **dark** slightly evolve with time, due to atmospheric and sensor conditions respectively.

Pixels close to points where $z_{fij}$ is abnormally low are masked out by setting $z = 0$.

For a whole trajectory, we have

$$G = \log \frac{\mathcal{L}_{\text{asteroid}}}{\mathcal{L}_{\text{null}}} = \sum_f \frac{\max(0, z_{fij})^2}{2} + G^0$$

which accomplishes our goal of reducing the likelihood computation to a single pixel lookup per frame.

### 1.4 Detection – `detectBright()`

We're now looking for object trajectories with constant speed $\Delta$. The basic idea is to start from the log likelihood ratio defined in the previous section, augmented by a speed likelihood term:

$$G = \log \frac{\mathcal{L}_{\text{asteroid}}}{\mathcal{L}_{\text{null}}} = \sum_f \frac{\max(0, z_{fij})^2}{2} + \frac{\|\Delta - \Delta_0\|^2}{2\sigma^2_\Delta} + G^0$$

However, we want to be resilient to the case where one or more of the $z_{fij}$ along the trajectory is an outlier. Therefore we sort the values of $\max(0, z_{fij})^2/2$ into $x_1 \le x_2 \le x_3 \le x_4$, and weigh them appropriately to obtain the score $s$:

$$s = \sum_{l=1}^{4} x_l w_l + \frac{\|\Delta - \Delta_0\|^2}{2} w_\Delta$$

for certain weights $w_1, w_2, w_3, w_4$ and $w_\Delta$. $w_i$ can be interpreted as the inverse of the raw moment of order 2 of $x_i$, with a normalization factor applied so that $w_3 = 1$. Since $x_4$ is very likely to be an outlier, it has infinite variance and therefore we set $w_4 = 0$.

### 1.4.1 Tracking

We enumerate all constant-speed trajectories passing through points where $z_{fij}$ exceeds a certain threshold at any given frame, with speed resolution 1/3 pixel per frame and maximum speed 14 pixels per frame. We use weights $(w_1, w_2, w_3) = (4, 2, 1)$ and $w_\Delta = 0$.

When trajectories are close to one another, we only keep the one with the best score. We also limit the number of trajectories to a reasonable number.

### 1.4.2 Scoring

We estimate the mean speed $\Delta_0$ of the image set by computing the median of speeds of trajectories above a certain score, and re-score the trajectories, this time with the final weights $(w_1, w_2, w_3) = (52, 8, 1)$ and $w_\Delta$ set to a value depending on the inter-frame time interval. We also penalize trajectories lying close to the border of the frame, because there is a much greater likelihood of artifacts.

The reason we don't use the same weights for tracking and scoring is that we want to penalize artifacts, which tend to have high temporal variance but are often flexible enough to maximize any scoring function (by reducing their variance if variance is strongly penalized). On the other hand, the tracking of true asteroids is mostly unaffected by weights, making them much less flexible. Using mismatched weights turns flexibility from an advantage into a disadvantage. A cleaner Bayesian approach would explicitly measure and penalize flexibility, at the cost of greater complexity.

## 2 Candidate output – `finish()`

Since the ground truth contains duplicate objects, each object is emitted twice, once with its original score and once with a reduced score. An early submission used a fixed penalty, and a later submission offered a minor improvement by modeling the score-probability curve and the probability that an object is duplicated given its score. Specifically, instrumenting the program on the training set revealed that the rank obeys the approximate relation $r = \sqrt{M/s}$ for some constant $M$, and that the probability that a candidate matches at least 1 object is well fit by an affine function $p(r) = p_0(1-\frac{r}{r_p})$, while the probability that a candidate matches 2 objects given that it matches at least 1 is well fit by $q(r) = q_0(1-\frac{r}{r_q})$. Therefore, the optimal rank $r'$ for the second candidate is such that $p(r') = p(r)q(r)$, and the optimal score $s'$ is

$$s' = \frac{M}{p^{-1}\left(p\left(\sqrt{M/s}\right) q\left(\sqrt{M/s}\right)\right)^2}$$

The constants were derived by observation of the graphs, but since scaling coefficients cancel each other out, there are few degrees of freedom and trial-and-error would easily give equally good coefficients. (All other constants in the code were from a priori knowledge, direct image observation and/or adjusted through trial and error.)

Once this is done, candidates are ranked by decreasing score and printed out.

# 3 Results

I chose this particular method because looking at the data, I saw that detecting very faint objects would be the number one factor in achieving a good score, much more so than being able to detect very slow or very fast objects. The layer separation was designed with this goal in my mind. So I didn't suspect that this method would be adapted for image sets where objects are moving slowly and are therefore hard to distinguish from stars, but it did prove surprisingly efficient, because while much of the energy of the object is transferred to the **star** layer during layer separation, there can still be enough energy left in the **mov** layer to trigger a detection, especially in the first and last frames.

The Bayesian approach also proved to work right from the first time, with very little tweaking required. Most tweaks had to do with imperfections in artifact modeling.

### An embarrassing mistake

Based on observing occasional null scores from my competitors, I incorrectly assumed that throwing an exception in my submission would result in a null score. I therefore decided to validate basic assumptions such as frame size and inter-frame displacements by liberally throwing exceptions, and assumed that since I had a good score no exceptions were being thrown. I did consider silently catching all exceptions for my final submission, but decided against it in order to minimize the risk inherent to any code change and because I knew the final testing would operate on the same input files as the provisional testing.

Unfortunately, a few image sets did throw exceptions. This problem only affected 1% of image sets but corrupted the whole test case, causing a hefty 19% score reduction which could have easily been prevented almost entirely by catching exceptions in `testingData()` (a two-line fix), and prevented completely by increasing inter-frame displacement limits. According to my estimates, my algorithm would have placed first with 77% probability with the first fix and 95% probability with the second fix.

This was a disappointment, but it was the first time I participated in that kind of contest so I'm still glad to have made the top 3 despite my rookie mistake.

# 4 Future work

The solution described above can still be improved on the following aspects:

- sensitivity to faint objects
- sensitivity to slow-moving objects
- rejection of artifacts

I had completed theoretical work for a greatly improved solution based on a much more accurate and theoretically sound Bayesian approach but didn't have time to implement it for the contest, even though it doesn't add much complexity. Preliminary testing shows very promising results.

Once the Bayesian approach has been fully exploited, smaller performance improvements could be gained by applying machine learning techniques such as random forests in order to improve artifact classification. Machine learning shines at artifact rejection, but a sound Bayesian framework is likely

to be the most powerful way to achieve sensitivity to faint objects, and the planned Bayesian approach should also be capable of handling most artifacts.