The LSST Moving Object Processing Pipeline

Kobus Barnard[a] Andrew Connolly[b] Larry Denneau[d] Alon Efrat[a] Tommy Grav[d]
Jim Heasley[d] Robert Jedicke[d] Jeremy Kubica[c] Bongki Moon[a]
Scott H. Morris[a] Praveen R. Rao[a]

[a]Department of Computer Science, University of Arizona, U.S.A.
[b]Physics and Astronomy Department, University of Pittsburgh, U.S.A.
[c]School of Computer Science, Carnegie Mellon University, U.S.A.
[d]University of Hawaii Institute Astronomy, U.S.A.

Corresponding author: Kobus Barnard, Computer Science Department, the University of Arizona, kobus@cs.arizona.edu.

The LSST Moving Object Processing Pipeline

Kobus Barnard[a] Andrew Connolly[b] Larry Denneau[d] Alon Efrat[a] Tommy Grav[d]
Jim Heasley[d] Robert Jedicke[d] Jeremy Kubica[c] Bongki Moon[a]
Scott H. Morris[a] Praveen R. Rao[a]

[a]Department of Computer Science, University of Arizona, U.S.A.
[b]Physics and Astronomy Department, University of Pittsburgh, U.S.A.
[c]School of Computer Science, Carnegie Mellon University, U.S.A.
[d]University of Hawaii Institute Astronomy, U.S.A.

ABSTRACT

We describe a proposed architecture for the Large Synoptic Survey Telescope (LSST) moving object processing pipeline based on a similar system under development for the Pan-STARRS project. This pipeline is responsible for identifying and discovering fast moving objects such as asteroids, updating information about them, generating appropriate alerts, and supporting queries about moving objects. Of particular interest are potentially hazardous asteroids (PHA's).

We consider the system as being composed of two interacting components. First, candidate linkages corresponding to moving objects are found by tracking detections ("tracklets"). To achieve this in reasonable time we have developed specialized data structures and algorithms that efficiently evaluate the possibilities using quadratic fits of the detections on a modest time scale.

For the second component we take a Bayesian approach to validating, refining, and merging linkages over time. Thus new detections increase our belief that an orbit is correct and contribute to better orbital parameters. Conversely, missed expected detections reduce the probability that the orbit exists. Finally, new candidate linkages are confirmed or refuted based on previous images.

In order to assign new detections to existing orbits we propose bipartite graph matching to find a maximum likelihood assignment subject to the constraint that detections match at most one orbit and vice versa. We describe how to construct this matching process to properly deal with false detections and missed detections.

Keywords: Asteroids, moving object detection, graph matching

1. INTRODUCTION

An important science goal for the Large Synoptic Survey Telescope (LSST) is to find and track fast moving objects such as asteroids. There is a particular interest in potentially hazardous asteroids (PHA's). As these tend to have unusual orbits, it is important that the process for finding moving objects be set up to properly manage the data in a statistical sense.

In this paper we describe a proposed architecture based on a similar system under development for the Pan-STARRS project.[6] The system links detections that are identified as from rapidly moving sources, and over time refines and validates the linkages, thereby constructing a moving object catalog. Our overall approach is probabilistic, allowing uncertainties to be provided for all answers. For example, the system constantly refines its belief that a proposed linkage is in fact an orbit, and the degree to which each detection is explained by it.

The general strategy is to break the problem into two parts. The first stage proposes potential linkages by associating individual point detections based on a simple motion model. The second stage refines these linkages based on the implied trajectories (few linked detections) and fitted orbits (sufficient number of linked detections). To accomplish this we compute a matrix of probabilities for the associations between candidate orbits and detections. We further model false detections and missed detections in the matrix. We then compute a maximum likelihood explanation for each image, given the proposed linkages, using bipartite graph matching.

Corresponding author: Kobus Barnard, Computer Science Department, the University of Arizona, kobus@cs.arizona.edu.

As new images arrive, their detections will either add evidence to previously proposed linkages or potentially seed new candidate linkages. In addition, since the list of candidate linkages will change, we propose processing images from the past. As a result of these activities, candidate linkages can gain or lose detections, and become either more or less certain as valid orbits. Eventually, poor candidates will be culled. Similarly, as more data become available, some orbits will be recognized as the same one, and they will be merged.

1.1. Input

The processing described in this paper relies on the coordinates of possibly moving objects derived from image differencing results. We assume that this is available not only for the current image group being processed, but for all previous images. Furthermore, the proposed system can make use of the following if available:

1. Error estimates for the detections
2. The source brightness and filter information
3. The detection efficiency as a function of expected source brightness, based on the image location and viewing direction.

1.2. Output

The main output is a catalog of linkages. Information that will be stored, pointed to, or computed on demand includes:

1. The detections and corresponding observation information
2. The degree to which we believe that each detection belongs to a given linkage
3. The degree to which we believe that the linkage is valid
4. The fitted orbit
5. Additional catalog information if it exists (e.g. it is a known, named object)

A key capability that the system will provide is, for any linkage (orbit), the maximum likelihood predicted position, and a characterization of the corresponding distribution of predicted positions for a given time. This is a critical component for internal use by the system, but it is likely useful externally as well. Note that this projection can be done approximately without a fitted orbit by using curve fitting.

2. PROPOSED APPROACH

Given all known detections at a given point in time, we would like to partition them into disjoint collections corresponding to orbits and a noise category. Probabilistically, we seek the posterior probability over the partitions, given the data. However, evaluating all possibilities is computationally intractable. Hence we want to build a system that behaves as though it has a good approximation of that posterior, especially in regards to inference. For example, we would like to be able to predict future detections of a proposed object. As a second example, we would like to evaluate the data for potentially hazardous asteroids (PHA's) based on inferred orbits.

Despite the fact that finding the complete solution is intractable, the nature of the data makes it possible to focus on promising candidates for linking. Our approach for doing so has two main components. One produces a feed of candidate linkages based on a short time span — "tracklets". This is possible because asteroids move relatively little between repeat visits of the same area of sky in the case of the LSST cadence. However, naive tracking does not do very well due to the large amount of noise and ambiguity. The approach taken to deal with this is described in further detail below (§3).

The second component of the system validates and improves the proposed linkages using all data, continuing to do so as additional data arrives. Intuitively, linkages will gain or lose detections as the computed orbit becomes more precise and more detections become available. Linkages can also be merged or refuted.

The key observation is that the LSST cadence provides sufficiently redundant information that, with intelligent subsequent processing, the initial linking described above potentially needs to succeed only once. Further, the proposed linkages need not be overly conservative, as false linkages will be pruned.

Related work. Our weight matrix serves a similar role as the (generalized) assignment matrix in multiple-target tracking.* Our approach has some elements in common the the work of Oh et al.,[10] although that approach seems computationally less suited to our problem because we need to consider many more trajectories. Finally, our approach to computing the probabilities of association seems most closely aligned with that of Virtanen et al.[15]

3. FINDING CANDIDATE LINKAGES

Asteroid linkage is a track initiation task that consists of finding new trajectories from point detections. Since linkage serves as a pre-filter for orbit fitting, we need to find detections from 3 or more different nights and covering a sufficient time span. The primary difficulty is uncovering these associations when time gap between observations of the same object may be large.

Below we briefly describe the asteroid linkage process and algorithms. These algorithms were originally developed for the Pan-STARRS survey to find initial associations.[6]

3.1. Linkage as a Three Stage Process

Asteroid linkage is treated as a three stage process designed to complement the basic cadence of future surveys. Each stage constructs longer and more accurate associations. Specifically:

- **Intra-night Linking** - We associate detections collected on the same night into small sets, called *tracklets*, that provide partial trajectory estimates (angular position and velocity).
- **Inter-night Linking** - We associate tracklets from different nights into larger sets including tracklets from 3 or more distinct nights. These larger sets can be used to more accurately estimate the object's motion.
- **Orbit Fitting and Tracking** - Once we have a sufficient number of detections from different nights, we can fit an orbit to these detections. The resulting orbit can then be treated as a tentative new object.

Because we cannot estimate an object's full orbit, and thus actual trajectory, until we have a sufficient number of detections, we must use a different model to perform initial asteroid linkage. For example, an asteroid's trajectory can be approximated as roughly linear for intra-night linkages.[12] Further, we found that a quadratic trajectory provides a good approximation over a time span of a few weeks and can be used as an effective inter-night filter.[4]

3.2. The Use of Spatial Structure

We can use spatial structure within the data to make linkage algorithms tractable on large data sets by "pruning" away large portions of the search space. The key intuition is that we often need to find detections "near" a given query point. For example, we may be interested in finding detections near a tracklet's predicted position. Thus we can often limit the search by only testing points in targeted spatial regions.

KD-trees are one type of spatial data structure that can be used to make linkage tractable.† Formally, KD-trees are hierarchical data structures that partition space by recursively splitting it along axis-aligned hyper-planes.[1] As shown in Figure 1, each node in the tree represents a region of space and (explicitly or implicitly) a set of data points. The hierarchical structure makes queries efficient because we can often rule out an entire branch of the tree by only examining the top-most node in that branch. For example, as shown in Figure 1.C, if we are searching for points that lie within r of point q, we can prune the subtree at node 8 (and all of its points) because the entire node falls outside of our search radius.

*For a good introduction to the assignment problem in data association, see Blackman and Popoli.[2]

†KD-trees have previously been used to accelerate track/observation association in multiple target tracking.[14]

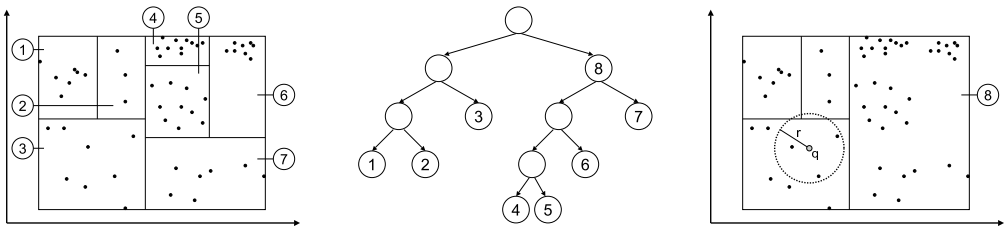

Figure 1. A KD-tree built from a set of two dimensional points (A and B). During a spatial search we can use the tree's structure to prune entire subsets of points that *cannot* fall within proximity r of the query point q, such as *all* of the points owned by node 8 (C).

3.3. Intra-Night Linking Algorithm

Our approach to intra-night linkage is to use prior velocity bounds to find potential associations by searching for detections that fall within the valid range of movement from each detection.[4] These associations can be found efficiently using a KD-tree that incorporates both angular position and time as dimensions. The potential matches can then be built into longer tracklets by using a branching search over associations, such as a simple form of multiple hypothesis tracking.[13]

3.4. Inter-Night Linking Algorithm

Our approach to inter-night linkage is to search for sets of detections that fit a quadratic motion model, by searching over all quadratic models that are defined by pairs of tracklets.[4] In addition, we efficiently determine whether each model is "supported" by additional tracklets from other nights. This algorithm uses a novel approach to searching tree-based spatial data structures that uses a *variable* number of tree nodes to adapt the search representation to the current search state.[5] Again, the key to making this search tractable is the ability to efficiently prune out large regions of the search space.

4. FROM CANDIDATE LINKAGES TO ORBITS

Given a set of detections from differenced images and a set of known orbits, we want to find the best association between the detections and the orbits. Our approach attempts to accomplish this task by applying *bipartite graph matching*,[7] which is a well-known technique in graph theory. A bipartite graph is a graph whose vertices are partitioned into two disjoint sets where vertices in one set are adjacent only to vertices in the other set. A matching assigns vertices of one set to vertices of the other set. For the purpose of matching detections with orbits, detections are represented as vertices in one set of a bipartite graph, and the orbits are represented by vertices in the other set of the bipartite graph. A weighted edge between a detection and an orbit indicates how likely the detection belongs to the orbit.

The overview of the proposed approach is summarized in Figure 2. We assume that linkages of detections are created by the process described in Section 3. The algorithm below describes the overall steps that produce a consistent set of long-term linkages (with estimates for validity) from a set of short-term linkages (or tracklets).

Algorithm 1: To Produce Long-term Linkages

for *ever* **do**

1: Choose data from an image. We consider new images as they are available, and images from the past. Consecutive streams of images can make projection of orbits faster, but streams can be interleaved (Section 4.1).
2: Compute the probability that each detection in image is explained by each orbit (Section 4.2).
3: Match detections to orbits with allowance for outliers (Section 4.3).
4: Merge any linkages that need to be merged (Section 4.4).
5: Update data structures (Section 4.5).

endfor

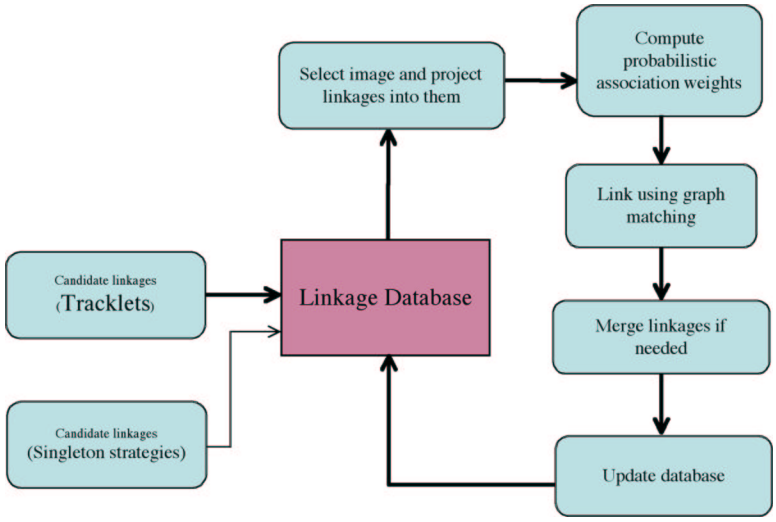

Figure 2. An overview of the proposed system described in the text. We envision two main subsystems. One provides candidate linkages of detections, and the other validates and improves the linkages, continually improving the linkage database over time. High certainty linkages and the corresponding orbital parameters are exported to user oriented data bases (not drawn).

4.1. Projection

We will process at least two streams of images. One is the new incoming images. We also need to re-processes one or more streams of old images. If images can be processed in sky coverage size batches, some efficiency can be reaped (explained shortly). The projection process provides a stream of images together with a list of linkages that *may* project into the associated image.

One method for computing which linkages may project into which images is as follows. For simplicity consider a roughly sky coverage temporal block (3-4 days) of images of known position. The detection position does not change much over this time because asteroids do not move very fast. Thus a generic position for the time window can be used to estimate the position over the entire window. We can use the generic position to consider a small set of images that might have this detection, accounting for errors due to the time approximation. Since we will only need to consider a small number of individual times (candidate images) for each object, we can expend more time computing where the object can appear more precisely. Each linkage is associated with the image(s) in which it might appear.

Ideally the projection process should be notified about changes and additions to the orbit data structures. For significant changes, the data for the images that have not yet been taken from the queue should be edited to adjust for incoming information as soon as possible.

4.2. Weight Matrix

This process computes a weight matrix that stores the negative log probability that a given detection is associated with a given linkage. It also has fictitious (phantom) orbits to absorb false detections, and fictitious (phantom) detections to model missed detections. The computation of values for the matrix are discussed further in Section 5. Conceptually, with all phantoms added, the overall matrix is a square table with one dimension for linkages (orbits) and the other for detection. However, we represent the matrix sparsely so that zero, or very low probability entries are explicitly encoded as missing. This is helpful because negative log probabilities have no stable representation in finite precision floating point arithmetic when probabilities tend towards zero. The weight matrix is passed into a graph matching algorithm that computes the minimum weight match, corresponding to the maximum likelihood assignment.

4.3. Matching

We use a global matching technique that maximizes the probability of all assignments for detections and orbits. As described above, we construct a bipartite graph with one set of vertices for orbits (including phantom orbits), and one set

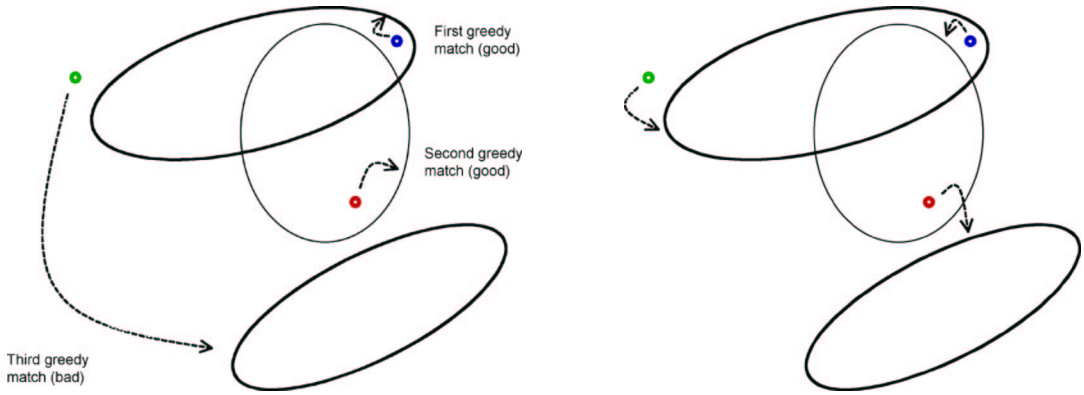

Figure 3. A simple example showing the difference between a greedy approach to matching and the graph matching approach. The ovals represent level curves in the probability density function of detection for three orbits. The small circles represent detections. The greedy approach (left) finds the best match, commits to it, finds the next best, and so on. Graph matching (right) finds the global optimal solution that minimizes the overall total cost. In the figure, this means that the large expense of the third greedy match is avoided. Instead, the approach incurs a little more error in the match for the bottom detection, in turn for a much less overall error. Recall that in this application the cost is set to the negative log probability and graph matching finds the maximum likelihood linkage, subject to the 1-1 constraint.

of vertices for detections (including phantom detections). We set the weights according to the weight-matrix. In principle, a bipartite graph can be constructed as a complete graph, because an orbit can potentially be associated with any detection. However, as described above, we represent the matrix sparsely to avoid precision problems during matching. An equally important benefit is that we can save significant computational cost by having fewer edges, as can be arranged by setting ones that are sufficiently unlikely to missing. Even though the graph is not particularly sparse due to the inclusion of phantoms, the computational saving is significant because the matching cost increases rapidly with the number of edges.

This calls for an implementation of bipartite graph matching that supports sparse graphs. Jonker and Volgenant's sparse matrix implementation[3] of the matching algorithm is one of the well known code bases which gave us the best performance compared to two others. We translated the sparse version of Jonker and Volgenant's algorithm from Pascal to C++ and applied it to our semi-sparse matrices. Preliminary experiments suggest that thresholding low probability edges to missing edges results in significantly reduced computational load in the case of expected LSST full image detection densities with 50% false detections.

4.4. Merging Linkages

The weight matrix and other data needs to be examined to look for linkages that may need merging. A candidate for merging are linkages that are fighting a detection. If further examination reveals that they would both like to claim the same set of detections, then the two linkages can be merged. Because we model missing detections when we estimate the quality of a linkage (§5.6), merges can be confirmed by noting an increase in quality. Quality increases for good merges because the combined linkage will have fewer missing detections.

4.5. Update

The matching process gives the new linkages which are then applied to database. In addition to the standard, obvious updates, the processing cycle for each image may require the following updates:

- Injection of any new probable linkages (e.g. from tracking) into data structures. This is "out of band" in that it is not a function of the matching process.
- If this is a re-processed image, then some old linkages may be need to be revised, and this should be explicitly tracked and exported to interested processes. Similarly, linkages that imply significant predictive changes need to be exported.

- The probability that modified linkages are believed needs to be recomputed and added to the data structures.
- Due to revisiting images in the past, some linkages might be whittled down to only a handful of detections. Naturally, these linkages have very small probability of existence. At some point, these need to be explicitly deleted from the database. This event may need to be advertised to processes that are interested in further analyzing unexplained detections.

5. WEIGHT MATRIX COMPUTATIONS

In the above algorithm, we consider matching a subset of orbits with a set of detections. We can compute the relative probability of linking a particular detection to a particular orbit, but this does not take into account the global constraint that each orbit generally gives rise to at most one detection. As already discussed, we propose using bipartite graph matching to integrate this global information. To do this effectively, we augment the matching problem with phantom detections (modeling missed detections) and phantom orbits (modeling false detections). The phantoms are simply a convenient book-keeping device, and we propose using sufficient numbers of them that every real entity has reasonable opportunity to match to one as described further below.

Regardless of whether matches are to real entities or phantoms, the cost of matching is set to the negative log probability of the match. This means that the minimum cost match provides a maximum likelihood estimate, subject to our constraint. We provide details for the matching costs for a detection and an orbit, a phantom detection and an orbit, a detection and a phantom orbit, and between phantoms.

5.1. Preliminaries

In what follows we use $p()$ for probability density. We keep this notation even when the expression can be equated to a real probability value.

We equate an orbit O_i with a set of linked detections, L_i. We abstractly consider our goal to be a partitioning all detections into disjoint linkage sets, L_i, and a set of detections attributed to noise. We use O_i to explicitly denote the hypothesis that the detections in L_i are the result of the same object. From these we can fit orbital parameters, or more generally, estimate the distributions over the orbital parameters and other potentially observable quantities. In particular, we assume that we can compute the appropriate density $p(s|O_i, L_i)$, of the signal, s, due to the orbit. The signal, s, has associated angular coordinates and brightness. We use d_s to denote a detection of s, and assume that we have a good understanding of the detection efficiency, $p(d|s)$, which is expected to be a function of viewing conditions. We also assume that we have the false detection density on a comparable scale, $p(d_s|N)$, where N represents noise.

We consider brightness because it can provide an important estimate as to how likely a detection is. For example, if a detection is expected to be near the detection limit, then missing it should not have much negative impact on our belief that the orbit exists. Brightness is a function of the position, size, and makeup of the object combined with viewing conditions and the filter used. Its distribution as part $p(s|O_i, L_i)$ can be estimated by combining factors computable from orbital parameters and fitting previous detections normalized for the other two factors.

For a given linkage of detections, L_i, we have a (possibly empty) set of missed detections, $\tilde{L}_i$. Technically, $\tilde{L}_i$ is a distribution because L_i implies a distribution over orbital parameters, but for now we assume that the distribution is tight enough that a simple list of missed detections can suffice.

5.2. Matching a Detection to an Orbit

The probability density of a match between a detection, d_s, and the linkage, L, under the assumption of the orbital relation (represented by O_i) is given by:

$$p(d_s|O_i, L_i) = \int p(d|s)p(s|o)p(o|L_i)do$$

Conceptually, one can imagine producing the probability distribution by projecting numerous instantiations of orbital fits to L_i produced by sampling from the appropriate distributions over the detections (e.g. "Virtual Asteroids"[8] — see Figure 4). However, to reduce computation time, we propose characterizing this projection process sufficiently well so that it can be quickly approximated. Notice that as we gather data and become confident about many asteroids, we can validate and further tune this process.

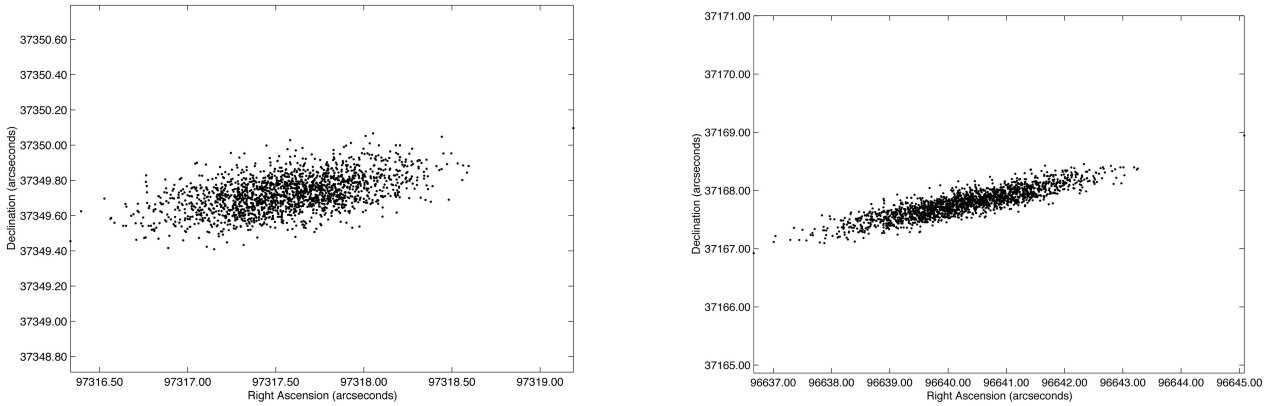

Figure 4. Example virtual asteroids generated three days out (left) and six days out (right) from the last observation (scales on the two graphs are not the same). These were produced by sampling detections with simulated Gaussian error with standard deviation set to 0.1 arc-seconds . While the distribution increases with the time from the last observation, it has clear structure that can be exploited in developing fast approximations for it as a function of the orbit. Orbit fitting was done with OrbFit[11] software.

5.3. Matching a Phantom Detection to an Orbit

We introduce phantom detections to account for missed detections. The main idea is that we can avoid errors that would occur if we forced a linkage to claim a incorrect detection when the real detection is likely to be missed. Each orbital candidate links to all phantom detections with the same value, namely, the probability that its detection will be missed.

Since we do not know the characteristics of the missed detection, we propose estimating the needed density conservatively by the maximum likelihood signal, $\hat{s}$, given by:

$$\hat{s} = argmax(p(s|O_i, L_i))$$

We then estimate the density of a missed detection by considering the detection efficiency at $\hat{s}$:

$$p(\tilde{d}|O_i, L_i) \approx (1 - p(d|\hat{s}))p(\hat{s}|O_i, L_i)$$

We propose that the number of these phantoms should be set by conservatively estimating the expected number of candidate linkages that will not have corresponding detections.

5.4. Matching a Detection to a Phantom Orbit

We introduce phantom orbits to absorb false detections. The weight of matching a particular detection to all phantom orbits is the same, namely the probability that the detection is a false detection, $p(d_s|N)$. Since each detection has different characteristics, the likelihood that it comes from noise can vary, which is modeled in this approach. The number of these phantoms should be set to make the cost matrix square after the previous set of phantoms has been added.

Notice that we have the machinery in place to develop improving estimates for false detection rates. As we become very confident of a large number of orbits, the unexplained detections can be mined for better characterizations of false detections as a function of the relevant parameters.

5.5. Matching a Phantom Detection to an Phantom Orbits

The cost of matching phantom detections to phantom orbits is always zero. This ensures that phantoms are not used unless needed, as they otherwise match to each other.

5.6. Probability of the Orbital Hypothesis

As linkages become longer leading to more refined orbital projections, and we re-process older data in this context, many outlier detections assigned to an otherwise good linkage will be claimed by other linkages or the noise hypothesis. However, we still need to estimate the quality of the linkages, both for queries, and guidance to when we should disband the linkage altogether.

To estimate our belief, $p(O_i|L_i)$, that the detections in L_i are due to a single object, we consider the relative strength of this hypothesis compared with the possibility that the detections are from sources other than O_i (denoted by $\tilde{O}_i$). We assume that the refinement process has made removed all cases where L_i is a mixture of detections due to a single object together with some outliers. Then we have:

$$p(O_i|L_i) = \frac{p(O_i, L_i)}{p(O_i, L_i) + p(\tilde{O}_i, L_i)}.$$

We first consider $p(O_i, L_i)$. Since we want to include the effect of missed detections, we have:

$$p(O_i, L_i) = \int \prod_{d_s \in L_i} p(d_s|o) \prod_{\tilde{d} \in \tilde{L}_i} p(\tilde{d}|o) p(o) do$$

where $p(d_s|o)$ is the probability density for the detection given the orbit, $p(\tilde{d}|o)$ is the probability of a missed detection, and $p(o)$ is the prior density over orbital parameters.

The probability, $p(\tilde{d}|o)$, can be expressed as:

$$p(\tilde{d}|o) = 1 - \int p(d|s) p(s|o) ds$$

We anticipate that rough approximations for these integrals will suffice, and that we will be able to compute those approximations reasonably efficiently. For example, the integral for $p(O_i, L_i)$ has limited support, and $p(o)$ is nearly constant in the region of non-negligible support.

To estimate $p(\tilde{O}_i, L_i)$ we consider how detections can arise without O_i. On the surface it appears that we should consider that they might come from other orbits, but this does not make sense in our framework because those orbits have already claimed detections for many images. Thus we simply assume that the only counter explanation for L_i is noise. We further assume that the probability density of noise is small enough that we can ignore the interaction between noise and $\tilde{L}_i$. This means we simply have:

$$p(\tilde{O}, L_i) \approx p(N, L_i) \approx \prod_{d_s \in L_i} p(d_s|N)$$

Together, these estimates can provide an estimate for the likelihood that the linkage is correct.

6. CONCLUSION

Our approach emphasizes careful estimation of the relevant probabilities in the context of a very high throughput system. We need to have a good estimates of how likely it is that linkages correspond to a real object, as well as corresponding error estimates of orbital parameters. These characteristics are very desirable for the scientific goals of identifying potentially hazardous asteroids, as well as having an accurate moving object catalog with many new entries that supports both standard and probabilistic queries.

ACKNOWLEDGMENTS

The LSST design and development activity is supported by the National Science Foundation under Scientific Program Order No. 9 (AST-0551161) through Cooperative Agreement AST-0132798. Additional funding comes from private donations, in-kind support at Department of Energy laboratories and other LSSTC Institutional Members. We also acknowledge the collaboration of the Pan-STARRS project for the development of linkage algorithms and providing realistic simulated data. Jeremy Kubica was supported by a grant from the Fannie and John Hertz Foundation.

REFERENCES

1. J. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM* **18**(9), pp. 509–517, 1975.
2. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, Boston, 1999.
3. R. Jonker and A. Volgenant, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," *Computing* **38**, pp. 325–340, 1987.
4. J. Kubica, *Efficient Discovery of Spatial Associations and Structure with Application to Asteroid Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2005.
5. J. Kubica, J. Masiero, A. Moore, R. Jedicke, and A. Connolly, "Variable KD-Tree Algorithms for Spatial Pattern Search and Discovery," in *to appear in Advances in Neural Information Processing Systems (NIPS)*, 2005.
6. The MOPS Team, "Pan-STARRS PS-1 Moving Object Processing System (MOPS) Algorithm Design Description (ADD)," PSDC-530-002-00, 2005.
7. L. Lovasz and M. D. Plummer, *Matching Theory*, North-Holland, Amsterdam, 1986.
8. A. Milani, M. Sansaturio, G. Tommei, O. Arratia, and S. Chesley, "Multiple solutions for asteroid orbits: computational procedure and applications," *Astronomy and Astrophysics*, 2004.
9. M. Granvik, K. Muinonen, J. Virtanen, M. Delbo, L. Saba, G. D. Sanctis, R. Morbidelli, A. Cellino, and E. Tedesco, "Linking VLT asteroid observations," in *Dynamics of Populations of Planetary Systems, Proceedings of IAU Colloquium 197*, Z. Knezevic and A. Milani eds., 2004.
10. S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for general multiple target tracking problems.," in *Decision and Control (CDC-04)*, (Paradise Island, Bahamas), 2004.
11. "OrbFit.", OrbFit consortium, http://newton.dm.unipi.it/orbfit/.
12. J. M. Petit, M. Holman, H. Scholl, J. Kavelaars, and B. Gladman, "A Highly Automated Moving Object Detection Package," *Monthly Notices of the Royal Astronomical Society* **347**, pp. 471–480, September 2003.
13. D. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control* **24**, pp. 843–854, December 1979.
14. J. K. Uhlmann, "Algorithms for Multiple-Target Tracking," *American Scientist* **80**(2), pp. 128–141, 1992.
15. J. Virtanen, K. Muinonen, and E. Bowell, "Statisical Ranging of Asteriod Orbits," *Icarus* **154**, p. 412-431, 2001.