SERIE B — INFORMATIK

An Application of Point Pattern Matching in Astronautics[◇]

Helmut Alt*
Lars Knipping*
Gerald Weber*

B 93–16
November 1993

Abstract

We consider a point pattern matching problem occuring in astronautics: Given are pictures from a camera with unknown orientation, showing a section of the starry sky. We want to determine the orientation of the camera by matching the constellations in the pictures with a star catalogue. This problem occurs for satellite attitude determination with star field sensors. We present a new solution, which uses incremental Delaunay Triangulation.

[◇]This research was supported by the Deutsche Forschungsgemeinschaft under Grant Al 253/1-4, SPP "Datenstrukturen und effiziente Algorithmen" and by ESPRIT Basic Research Action No. 7141, Project ALCOM II.
*Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, e-mail: `alt@tcs.fu-berlin.de`, `knipping@tcs.fu-berlin.de`, `weber@tcs.fu-berlin.de`.

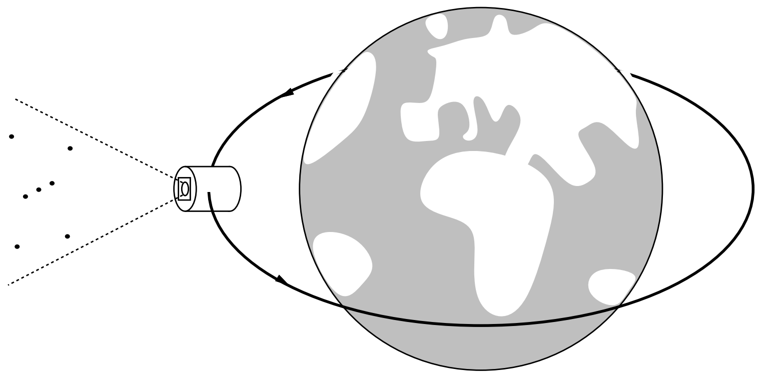


Figure 1: A satellite with a star field sensor, observing the starry sky.

1 Introduction

In this article we consider a pattern matching problem, which occurs in a special kind of satellite attitude determination. This method is used for example by the two experimental satellites TUBSAT A (which is already in orbit) and TUBSAT B (starting soon) from the "Institut für Luft- und Raumfahrttechnik" (Institute for Aero- and Astronautics) of the Technical University of Berlin (see [11]) and can be described as follows:

To determine the orientation of the satellite with respect to the fixstar background, the satellite possesses a so-called star field sensor — this is a video chip camera with fixed magnification, which is immovably mounted on the satellite and shows a section of the starry sky. The output of the camera can be considered as a list of the brightest m ($m \leq 10$) light sources detected by the camera, containing for each source the pixel position and a brightness information, which has no high reliability, however. If the position of the constellation seen by the camera is localized in a star catalogue, the orientation of the satellite can be determined.

We want to consider the problem in an abstract form. The star catalogue can be considered as a point set on the unit sphere. This is also true for the points in the camera picture if we apply to them the inverse of the camera projection.

The resulting point set we call in the following the *pattern*. We can now state the *exact matching problem* as follows:

> Given a set C of n points (the *catalogue*) and a set P of m points (the *pattern*) on the unit sphere, does there exist a rigid motion that maps P into C? (report all such motions, they are called *matchings*.)

Here, in case of the unit sphere, a rigid motion is a rotation around some axis through the origin. Previous articles considered point pattern matching in the plane, which is closely related, but then a rigid motion is a combination of some translation and some rotation.

In the given application, the set C is fixed. Therefore it is important to solve the *query problem*, that means to preprocess C and produce a reasonably small static query data structure D which can be used to find a quick answer to the matching problem for any query pattern P.

Since P represents data from a camera with bounded resolution, the data contain errors and the match will only be approximate. Therefore, following [5] we define the *approximate matching problem*:

> Given point sets C and P on the unit sphere, determine the smallest ε such that there is a rigid motion mapping each point in P within distance ε of some point in C.

We can consider the query problem for this version, as well. Moreover many solutions use additional information about the pattern, for example that the angle of aperture of the camera (and therefore the diameter of the pattern) is known, m is known, or (as in our solution) brightness information is accessible.

In Section 2 we will report previous solutions to the problem, in particular ones that were developed in application areas like the algorithm used now by the TUBSAT crew. In addition we will present theoretical results concerning the problem. In contrast to the TUBSAT-solution, those results are highly complicated and do not take advantage from considering the query problem.

In Section 3 we present a new data structure, which solves the query problem, and is time- and space-efficient. We implemented this solution, and the properties of our implementation are discussed in Section 4. The Institute for Aero- and Astronautics of the Technical University of Berlin was so kind to provide us with test data from TUBSAT A, taken in the orbit.

2 Previous Solutions

The following straightforward idea is common to many previous solutions and will be used by us as well:

A pair q of points from P is chosen, and a search for all pairs r of the same distance in C is performed. Each of the two possible mappings from q onto any such r can be uniquely extended to a rigid motion of P (see Figure 2). Therefore each pair r defines exactly two candidates for matchings. What remains to be done is to check for each matching candidate whether it really matches each point in P onto some point of C.

The programs used in practice are mostly very specially adapted to the special star field sensor and star catalogue, which is used. For example, the solution, which is already in use for the TUBSAT satellites [11] uses the fact, that the diameter of the pattern is bounded by the angle of aperture of the camera. The query structure simply contains all distances of two stars in the star catalogue, which are close enough that they can occur in the pattern. For any given pattern P, all pairs q of points in P are considered. For each q pairs of points in C having (approximately) the same distance are searched for in the query structure. In this way for each q a list of matching candidates is obtained. The actual matchings are found by intersecting all these lists.

In [3] a solution for another type of star field sensor with very small angle of aperture ($4^\circ \times 3^\circ$) and quite good brightness information (error ± 0.25 magnitude) is presented. The algorithm determines for each detected light source the set of stars with similar brightness, and searches for representatives, one from each set, which have the correct distances between each other.

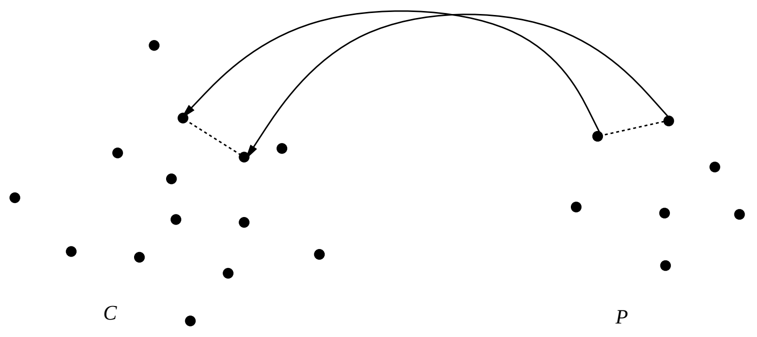


Figure 2: The mapping between the two pairs defines the whole mapping

Within the Computational Geometry community point pattern matching usually was investigated for sets of points in the plane (see [1, 2, 5, 7, 8, 13]), but it seems that the bounds and algorithms described in the following can be transferred to sets of points on the unit sphere, as well.

In case of the exact matching problem with no additional information about the pattern P matching candidates are found by determining all occurrences of one given distance in a set C of n points.

The best upper bound known for the number of such occurrences is $\mathcal{O}(n^{4/3})$, which was first stated by Spencer, Szemeredi and Trotter in [12]. A simpler proof of the same bound, which also leads to a better constant, was presented by Clarkson, Edelsbrunner, Guibas, Sharir and Welzl in [4]. The best algorithm for finding all these occurrences can be derived from a result by Matoušek [9] and takes time $\mathcal{O}(n^{4/3}\, 2^{\mathcal{O}(\log^* n)})$.

After having generated these candidates for matchings, for each one it has to be determined whether it really is a match. This can be done in time $m \log n$ per candidate by checking for all other points in P whether they are matched to points in C as well. So the resulting upper bound for the running time of the algorithm is $\mathcal{O}(n^{4/3} m \log n)$. This algorithm, however, is highly complicated and therefore seems not to be appropriate for our application. In addition, it is not particularly designed for solving the query problem, in fact it seems that this variant of the problem has not been considered theoretically before.

An algorithm solving the *approximate* matching problem in time $\mathcal{O}(m^3 n^2 \log^2 mn)$ is presented in a recent paper [5]. Again, no particular solutions for the corresponding query problem are known.

3 Our Approach

We present a solution, which solves the query versions of the exact and the approximate matching problem, but uses additional information about the pattern.

First some preliminary deliberations: The pattern from the camera consists of course of the brightest stars in the observed area. Therefore it is natural to look for "bright patterns" in the star catalogue. This term is made precise by the following definitions.

We call a triangle of stars *bright*, if and only if its enclosing disc contains only stars, that are fainter than the faintest of the three stars. We call a pair of stars a *bright pair*, if and only if there exists a third star, so that the three stars form a bright triangle (see Figure 3).

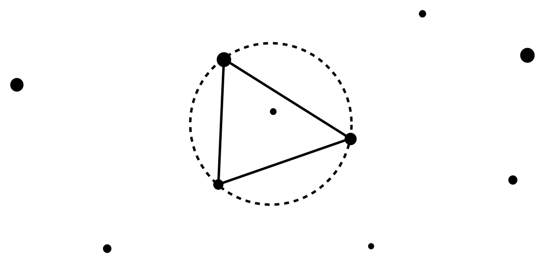


Figure 3: A bright triangle.

The crucial point of our solution is, that the set of bright pairs in C can be efficiently computed and stored, because it has a strong relation to a concept from computational geometry, namely the *Delaunay triangulation.* We will shortly explain this concept.

Let P be a finite set of points in the plane. A set of three points from P is called a *Delaunay triangle*, if the circumcircle of the three points contains no other point from P. The set of Delaunay triangles of P forms a triangulation of P, the Delaunay triangulation.

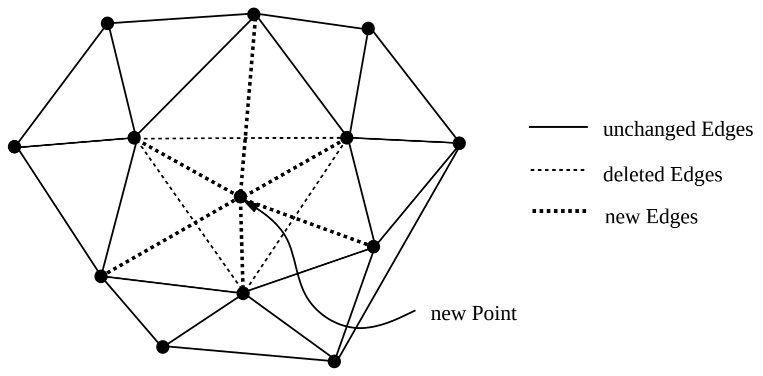


Figure 4: Insertion of a point in a Delaunay triangulation.

There are several algorithms known to compute the Delaunay Triangulation of a finite set of points. Especially there are some *incremental* Algorithms. They insert the points one after the other, and maintain the Delaunay triangulation of the points already inserted (see Figure 4). We now state the connection between the set of bright pairs in C and the edges occurring during this incremental Delaunay triangulation, in the form of three claims:

1. If we incrementally compute the Delaunay triangulation of the star catalogue (for points on the unit sphere this is the convex hull, see Figure 5), inserting the stars in order of decreasing brightness, the set of Delaunay triangles (resp. Delaunay edges)

appearing during the process is exactly the set of all bright triangles (resp. bright pairs) in the star catalogue.
2. The number of bright pairs is roughly $6n$.
3. The set of all bright edges can be computed by an algorithm which in general has runtime $\mathcal{O}(n \log n)$ with a reasonably small constant for random patterns.

The first claim can be seen if we take one bright triangle and look only at the stars that are not fainter than the faintest of the triangle stars. In this set the enclosing disk of the triangle is empty – the triangle is Delaunay. If we compute the Delaunay triangulation of the star catalogue, inserting the stars in order of decreasing brightness, then this triangle will appear as a Delaunay triangle at the time of the insertion of the faintest of the triangle stars. The second and third claim can be made sure by using the algorithm presented by Guibas, Knuth, and Sharir in [6] to compute the incremental Delaunay triangulation of the star catalogue. Better bounds can be obtained using the technique of backwards analysis by Seidel [10]. In fact, it can be shown that, if the order of insertion is randomized, the expected total number of Delaunay edges produced by the algorithm is $6n$, and the expected runtime is $\mathcal{O}(n \log n)$. Since the star data have the property to be a "good random set", these bounds roughly hold for the star catalogue, as it is stated in the second and third claim.

The conclusion from these deliberations is that we get a solution for the exact query problem, but with the following condition for P: P must contain a pair of points q, that is matched to a bright edge in the star catalogue. In practice this condition usually is satisfied, especially since the camera picture contains the brightest stars from the observed area. The matching candidates are the possible matches of q in the star catalogue. For each of them we search for all other points of the image of P in C. This matching criterion of course requires a searching structure in the star catalogue.

This solution described can easily be extended to a solution of the approximate query problem.

Our solution consists of two algorithms: A preprocessing algorithm for the star catalogue C and a query algorithm for given patterns P. Let us shortly summarize both algorithms:

1. **Preprocessing**
   - Compute the Delaunay triangulation of the star catalogue C incrementally, inserting the stars in the order of decreasing brightness.
   - Store all Delaunay edges occuring during the process in a data structure D.
   - Sort the edges in D by length (which corresponds to the angle spanned by an edge on the surface of the sphere).
2. **Query for Pattern P**
   - Take a pair in P that surely corresponds to a bright pair q in C (Details will be given in Section 4.2).
   - Search for all edges of (approximately) this length in D; these are the matching candidates generated by q.
   - Check for each matching candidate, whether it is a matching by searching for the other points of the image of P in C.

As was explained before the preprocessing algorithm has an expected runtime of

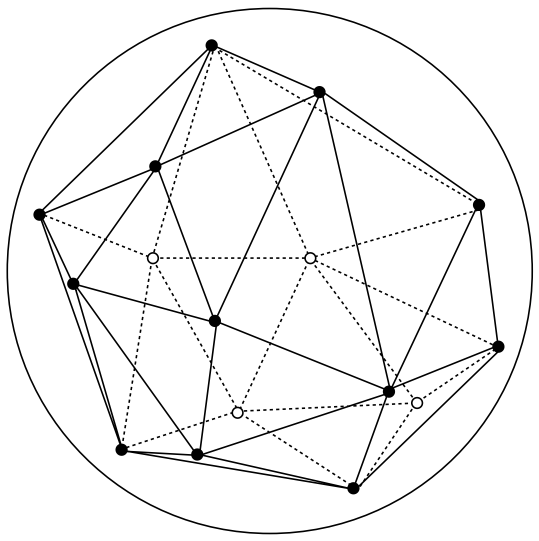


Figure 5: The Delaunay triangulation of a point set on the sphere is the convex hull of the points.

$\mathcal{O}(n \log n)$ for the construction of the Delaunay triangulation and, since the expected total number of edges ever produced is $\mathcal{O}(n)$, the same bound holds for the sorting step.

The runtime for a query is $\mathcal{O}(m \log n)$ per matching candidate, so it depends on the number of matching candidates which in turn depends on the tolerance used in the second step of the query algorithm. Also, in the implementation, to be on the safe side we do not only use one but several pairs of points in P. Therefore, precise bounds on the asymptotic runtime of the query cannot be given.

4 The Implementation

We implemented the algorithm and a window-oriented user interface for the special variant of the problem in connection with the TUBSAT satellites. We have tested this implementation with real-life data of TUBSAT A from the orbit. We will discuss here experiences and problems during the implementation and testing of the algorithm. Our program is implemented in C++, compiled with GNU C++ 2.3.3 and runs under SUN OS 4.1.3 and OpenWindows 3.0. The preprocessing algorithm is performed for a catalogue of stars up to the 5th order of magnitude (about 3000 stars) which takes about three seconds on a SUN SPARC station 10-20. The running time for a query is about 1/4 second per considered edge of the pattern.

4.1 Extracting the Pattern from TUBSAT Data

The TUBSAT data have a special format. Each picture of the camera is a sequence of sixteen single snapshots, which are slightly rotated because of the movement of the satellite (see Figure 7 a)). Each snapshot is a list of the ten brightest light sources detected by the camera. This format was chosen by the developers for two reasons. First the TUBSAT satellites send the pictures back to earth where the matching is done. Reasons are the low budget, that does not allow enough on-board hardware, as well as the fact that the matching algorithm still needs human interaction. So the transmitted data must be of small size, but still redundant (for error detection). Secondly the camera pictures themselves have a considerable amount of noise; in each snapshot besides the star images there is a number of so called *ghost stars*, i.e. bright spots that occur because of transmission errors or because of hardware errors due to thermal fluctuations. This also makes redundant information necessary.

The pattern points are automatically extracted by the program. The points which occurred only once in the picture sequence are supposed to be ghost stars, those which leave "tracks" in the sequence should be pattern points (see Figure 7 a)). As there is only a small constant number of snapshots and points in each snapshot we could afford to use a "brute force" approach. So we identified pattern points by looking for occurrences of equal distances in successive snapshots. For each point in a snapshot we compute the list of all distances to other points. For each point p in one snapshot we compare its list to the ones of all points in the next snapshot. Then we assign that point q to p whose list has the highest number of coincidences with the one of p, i.e. we assume that p and q represent the same star. This is only done, however, if there are at least two coinciding distances in the lists of p and q [1] and only those q are considered which have a small but nonzero distance to p [2], because a star moves not too far between successive pictures. p is supposed to be a ghost star if there is no such q. This method is repeated a second time but this time distances to points supposed to be ghost stars are not considered, so again some points are dropped. Then all double assignments are removed; if q is a correspondence candidate for p1 and p2, the one with more equal distances is taken. Now all points having a corresponding point are candidates for being pattern points. Finally the candidate points of that snapshot with the most candidates are taken as pattern points (see Figure 7 b)).

For our test data this worked very well in most cases but we also provided a pattern editing tool as our automatic extraction not always recognizes all pattern points. This tool allows to select and to discard points of the snapshots, so it allows for selecting a pattern by hand. It also proved useful for "correcting" extracted patterns since in some cases a light source in the snapshot sequence represents a *planet* (see Figure 8), which of course cannot be found in the star catalogue.

4.2 Implementation of the Incremental Delaunay Triangulation

The implementation of the incremental Delaunay triangulation is the major part of the preprocessing step. Our implementation follows in its combinatorial structure the algorithm presented in [6], adapted to geometric data that are three-dimensional (points on the unit sphere). Fortunately, the star catalogue represents a point set that is in general position, therefore we do not have to handle degenerate cases.

Also the first step of the query algorithm, which we only vaguely described above as "find an edge that surely corresponds to a bright pair" is realized with incremental

[1] We only allow patterns containing at least three points as one can almost always find a good match in the star catalogue for two points.

[2] We quite often observed points which clearly represented no stars occuring at exactly the same coordinates in different snapshots of a sequence.

Delaunay triangulation: The pattern is triangulated, as well, and it is attempted to find an edge in this triangulation which surely corresponds to a bright pair in the star catalogue. Experiences show, that at least one of the edges normally does so. (In fact, any edge within a triangle whose circumcircle completely lies within the area of the pattern will do.) For each such edge q a search in the data structure D of Delaunay edges of C is performed to find all pairs in C, whose length (angle) differs from q's only by a small multiple of the camera pixel size. So each such edge q generates a list of matching candidates. Since a pair r in D representing a candidate is not exactly of the same length as q, the rotation matrix mapping q onto r is not fixed exactly. The algorithm chooses the matrix M obtained from mapping the first point of q onto the first point of r and then rotating q around this point till it has the same direction as r.

Because the pattern is derived from a set of gridpoints (the pixel grid of the camera), degeneracies in the triangulation of the pattern are very likely. A simple deterministic perturbation algorithm is applied. This is justified, because the degeneracies are artificial: The matching pattern will be nondegenerate.

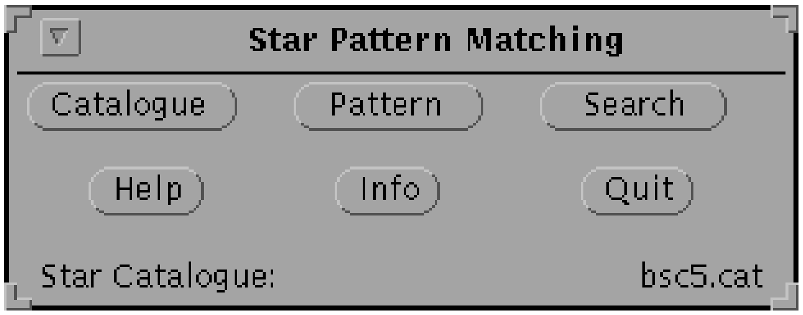
For the third step of the query algorithm we have chosen the matching criterion which is described above, in an approximate form: We apply the rotation M to the pattern, search for each point of $M(P)$ its nearest neighbor in the star catalogue, and compute the distance (angle) to this point. We define the *deviation* of the matching candidate as the sum of all these distances.

The search for the nearest neighbour of a point p in the star catalogue normally requires a *Voronoi diagram* of the star catalogue. For reasons of simplicity and space efficiency we decided to use another method which, however, may not give the correct result in any case. We determine within the (final) Delaunay triangulation of the star catalogue that triangle in which p is located. The triangle vertex nearest to p is returned as the "nearest neighbour" of p. This may be incorrect but it is correct for example if the neighboring triangles have no obtuse angle. Experiences have shown, that this usually is the case in the Delaunay triangulation of the star catalogue so our simplification causes no problem.

In this way the algorithm determines for each edge q of the pattern's triangulation the best matching among the matching candidates for q, Then it sorts these matches in the order of increasing deviation. This list of matches is returned (presented to the user).

4.3 Numerical Problems

The set of rotations around an axis through the origin is represented by the group of unitary matrices. This group has well known favorable properties, for example inversion is done by transposition. Nevertheless the computation of the matrices, which represent the matching candidates, must be done carefully to avoid numerical problems. For example, the following subproblem has turned out to be critical: Given two unit vectors u and v, determine the unit vector w, that lies in the plane of u and v and suffices $uw = 0$, $vw > 0$. A straightforward solution would be to compute $v - u(uv)$ and normalize. However, the angle between u and v is usually small, and therefore this computation is numerically instable and leads to insufficient results. The correct result can be obtained by the following solution: We compute the vector product $z := u \times v$. Then we normalize z obtaining z'. The vector w is now $z' \times u$.



Star Catalogue: bsc5.cat

Figure 6: The control window

4.4 The User Interface

When the program is started it first displays the control window shown in Figure 6.

By clicking the button "Catalogue" the user can load the star catalogue from a file which is then preprocessed as described before.

After clicking "Pattern" the program asks for a file containing TUBSAT data, i.e. a sequence of snapshots of a section of the sky. From these a pattern will be extracted as was described in Section 4.1. The program then displays two windows showing pictures of the complete data and the pattern selected (see Figures 7 a) and b)).

The button "Edit" in the first window can be used to edit the pattern, in particular to remove stars from it or even to select a pattern out of the snapshots by hand.

Pressing "Search" in the control window starts the search for the pattern within the data structure of the catalogue. Then a window will be displayed showing the section of the sky where the best matching was found (see Figure 7 c)) .

Also the next best matchings are available, they can be browsed through using the buttons "Next" and "Prev" in this window.

4.5 Reliability of the Output

Among the TUBSAT data are pictures from the starry sky as well as occasional pictures from the earth, which look considerably different. For all pictures, which are presumably constellations, our program finds convincing matches.

In the list of matches produced by the query algorithm, the best matchings represent the same and presumably correct constellation, but the matrices for each of these matchings differ slightly. This is because each matrix is generated for another edge q in the triangulation of P; it follows from Section 4.2, that this leads to different matrices. Also the deviation differs for these matches. The best deviation among these is usually about 0.5 pixel diameters per pattern point, the worst 2 to 3 times higher.

One key value for the reliability of the matching algorithm is the difference between the (best) deviation for the presumably correct answer and the smallest deviation of a wrong answer. This difference depends strongly on the number of pattern points. For three points it is very small, for four points it is observable. But to obtain really significant gaps, the pattern should contain at least five points. The smallest deviation of a wrong answer is then about 5 to 10 pixel diameters per pattern point.

In comparison we have tested the average deviation per point of the "matches" that the algorithm finds for random point patterns instead of real photographs of the starry sky. The results are listed in the following table.

points in pattern	3	4	5	6	7	8	9	10
random patterns	0.6	3.5	4.8	6.6	7.5	9.25	9.5	11.6
TUBSAT photographs	0.29	0.50	0.72	0.67	0.59	0.54	0.51	0.70

5 Acknowledgements

Our special thanks go to the people of the TUBSAT team at the Technical University, who generously supported us with their data. Also we have to thank Bernd Gärtner, who adapted an efficient map labeling algorithm developed by F. Wagner and M. Forman for the use in our screen output of the star catalogue.

References

[1] H. Alt, K. Mehlhorn, H. Wagener and E. Welzl, "Congruence, Similarity, and Symmetries of Geometric Objects", *J. on Discr. Comp. Geom.*, 1988, pp. 237-256.

[2] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak and M. Werman, "Matching Points into Noise Regions: Combinatorial Bounds and Algorithms.", Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms, 1991, pp. 42-51.

[3] D. Baldini, M. Barni, G. Benelli, A. Foggi and A. Mecocci, "A New Star-Constellation *(sic!)* Matching Algorithm for Satellite Attitude Determination", *ESA Journal* 1993, Vol. 17, pp. 185-198.

[4] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir and E. Welzl, "Combinatorial Complexity Bounds for Arrangements of Curves and Spheres", *J. on Discr. Comp. Geom.* 5, 1990, pp. 99-160.

[5] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg and D. Kravets, "Geometric Pattern Matching under Euclidean Motion", Proceedings Can. Conf. on Comp. Geom., 1993, pp. 151-155.

[6] L. J. Guibas, D. E. Knuth and M. Sharir, "Randomized Incremental Construction of Delaunay and Voronoi Diagrams", *Algorithmica*, 1992, pp. 381-413.

[7] K. Imai, S. Sumino and H. Imai, "Minimax Geometric Fitting of Two Corresponding Sets of Points.", Proc. 5th ACM Symp. on Comp. Geom., 1989, pp. 266-275.

[8] P. J. Heffernan and S. Schirra, "Approximate Decision Algorithms for Point Set Congruence.", Proceedings, 8th Annual ACM Symp. on Computational Geometry, 1992, pp. 93-101.

[9] J. Matoušek, "Range searching with efficient hierarchical cuttings", Proc. 8th ACM Symp. on Computational Geometry, 1992, pp. 276-285.

[10] R. Seidel, “Backwards Analysis of Randomized Geometric Algorithms”, Technical Report ICSI Berkley, TR-92-014, 1992.

[11] U. Renner, B. Lübke-Ossenbeck and P. Butz, “TUBSAT, Low Cost Access to Space Technology”, Proceedings, Symp. (international) Small Satellites Systems and Services, 1992, Arcachon.

[12] J. Spencer, E. Szemeredi and W. T. Trotter, Jr., “Unit Distances in the Euclidean Plane”, *Graph Theory and Combinatorics*, 1984, pp. 293-303.

[13] G. Weber, “Point Pattern Matching”, Diplomarbeit, Institut für Informatik, FU Berlin, 1993, to appear.

Examples for Patterns and their Matchings

Figure 7

a) A data record from the star field camera, consisting of 16 single snapshots. Five star traces are visible. The extracted stars are marked with circles.
By clicking the "Edit"-button, the extracted pattern can be changed. The "Grow"-button zooms the panel.

b) The five stars extracted from the data record.

c) The best matching found by the algorithm. The pattern is matched against the stars Dubhe, Merak, Megrez and Phekda in the Big Dipper and λ Draconis. In the lower right corner the deviation of this match is shown. Because the choice-item labeled "edge" was clicked, the edge in the pattern, that was used for the candidate generation, is visible.
The text in the lower left corner says, that 10 matches are returned by the algorithm, and this is the first of these matches. The user can get the other matches with the "Next"- and "Prev"-buttons in the upper left corner.
With the two checkboxes the user can switch the labels for stars and constellations on and off. A slider allows the user to choose the maximal order of magnitude of the stars displayed.

d) The "best" matching different from the previous. That means the first four matches are nearly the same. The deviation, that is about ten times higher than for the previous matching shows clearly this is a wrong matching. Such wrong matchings are produced, if an bright edge in the pattern does not correspond to a bright edge in the star catalogue.

Figure 8

a) Another data record from the camera.

b) The extracted pattern. The big circle represents a very bright light source. In fact this source is too bright for a star.

c) The best matching found by the algorithm. The bright "star" is matched against a rather faint star. The fact, that the matching found lies within the zodiac, gives strong evidence, that the bright source is a planet, in fact, in this case it is Jupiter.

d) The bright "star" was removed from the pattern with the editing facility. The matching stays the same, but the deviation decreases considerably.

Figures 9 and 10

Both figures consist of the following parts:

a) A data record from the camera.

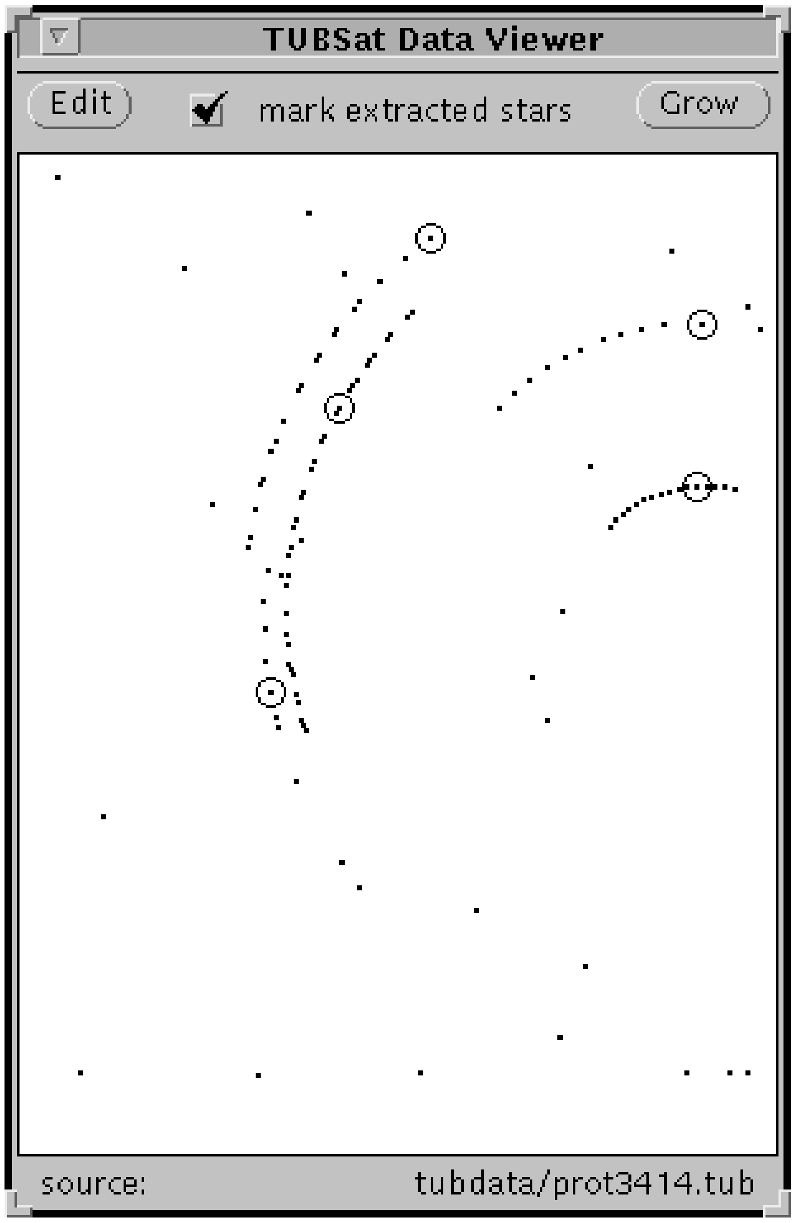
b) The extracted pattern.

c) The best matching found by the algorithm.

d) The "best" wrong matching.

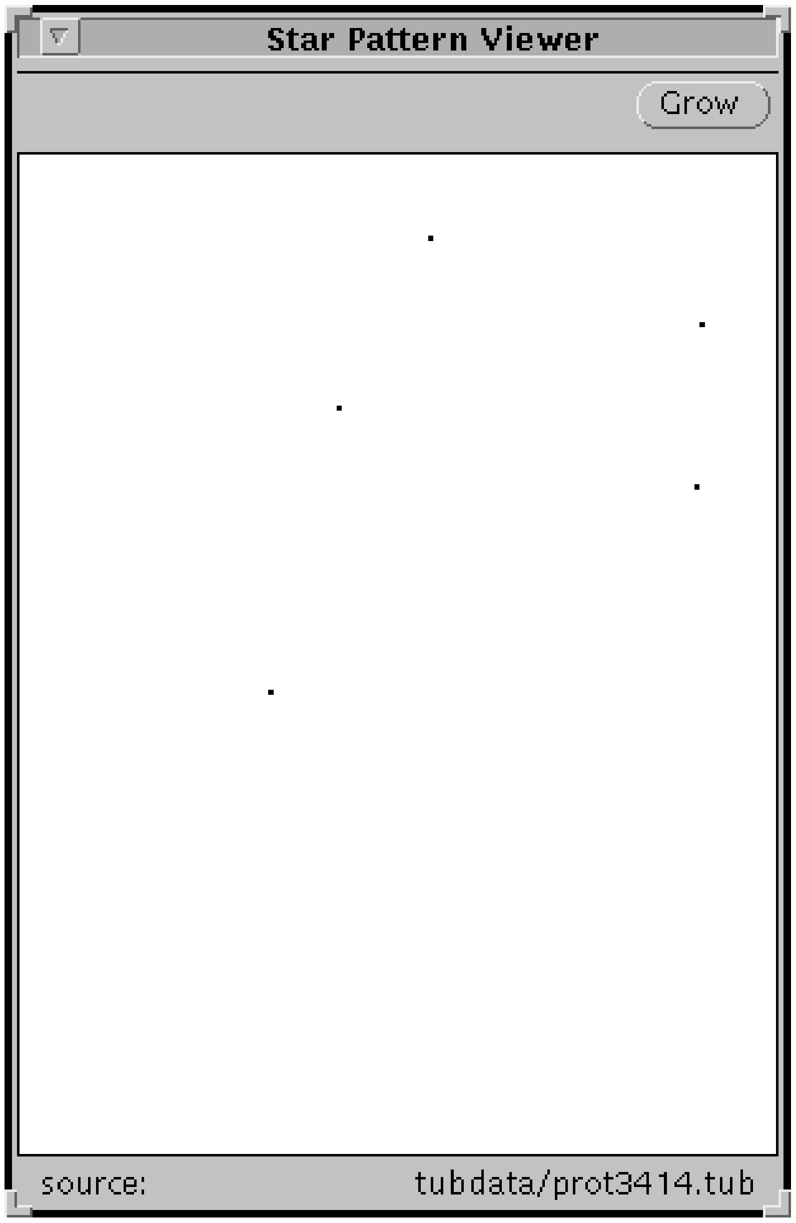
Figure 7. The Big Dipper.

a)



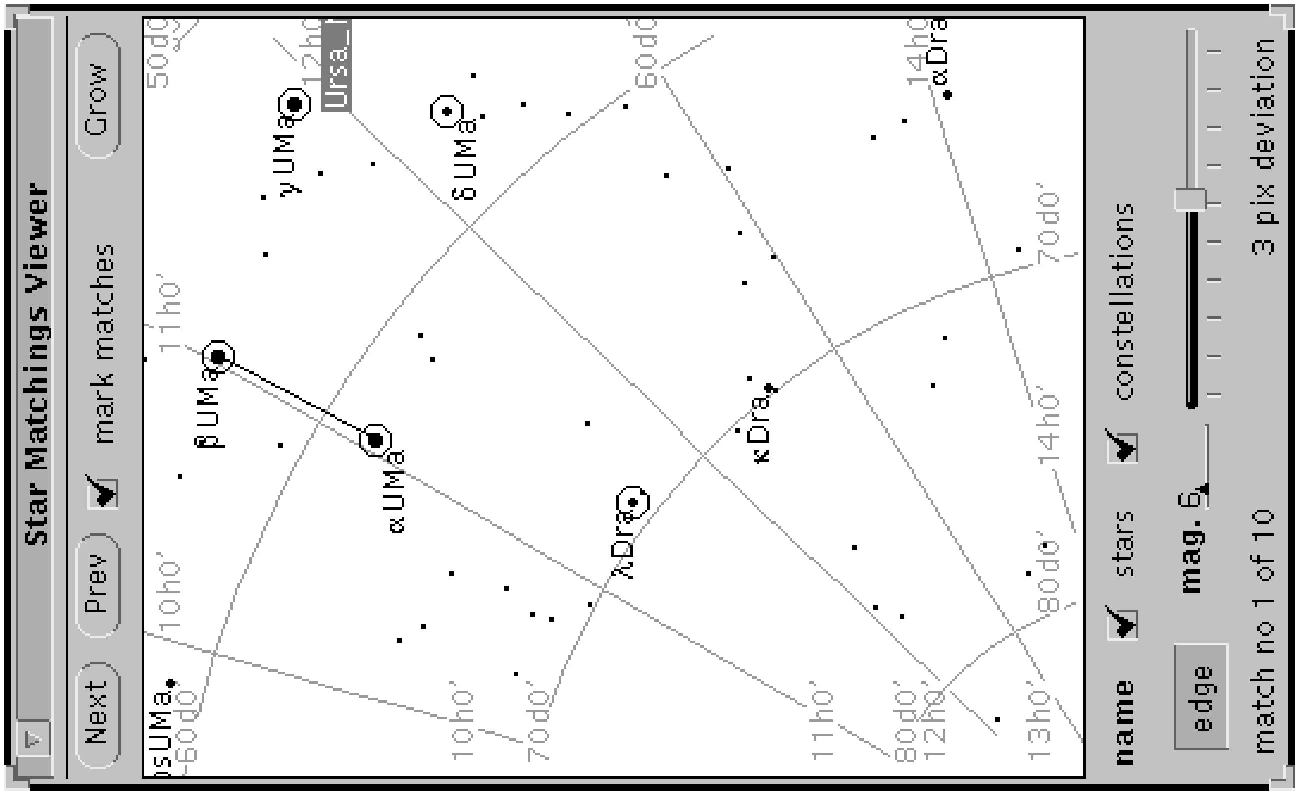
source: tubdata/prot3414.tub

b)



source: tubdata/prot3414.tub

c)


Star Matchings Viewer
Next
Prev
mark matches
Grow
βUMa
αUMa
γUMa
δUMa
Ursa_
λDra
κDra
αDra
10h0'
11h0'
12h0'
13h0'
14h0'
50d0'
60d0'
70d0'
80d0'
name
stars
constellations
edge
mag. 6
match no 1 of 10
3 pix deviation

d)

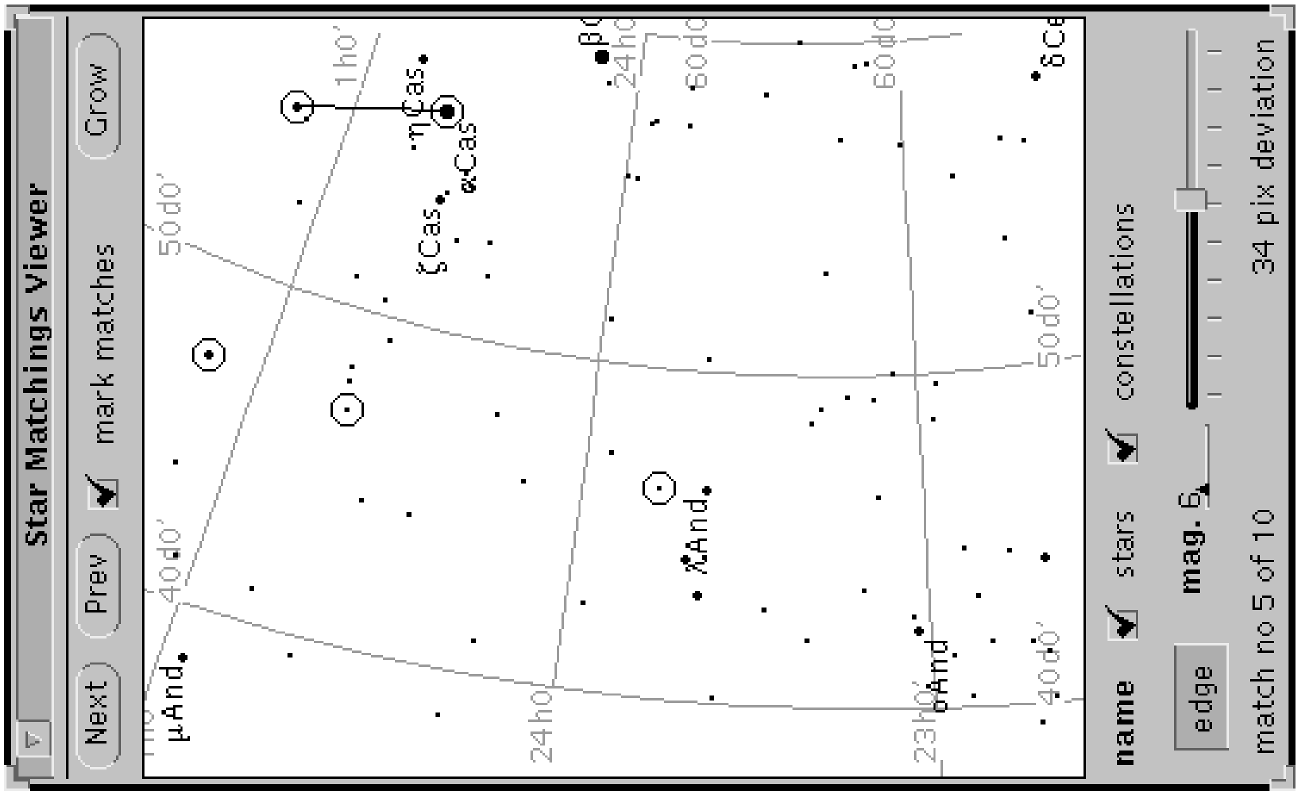
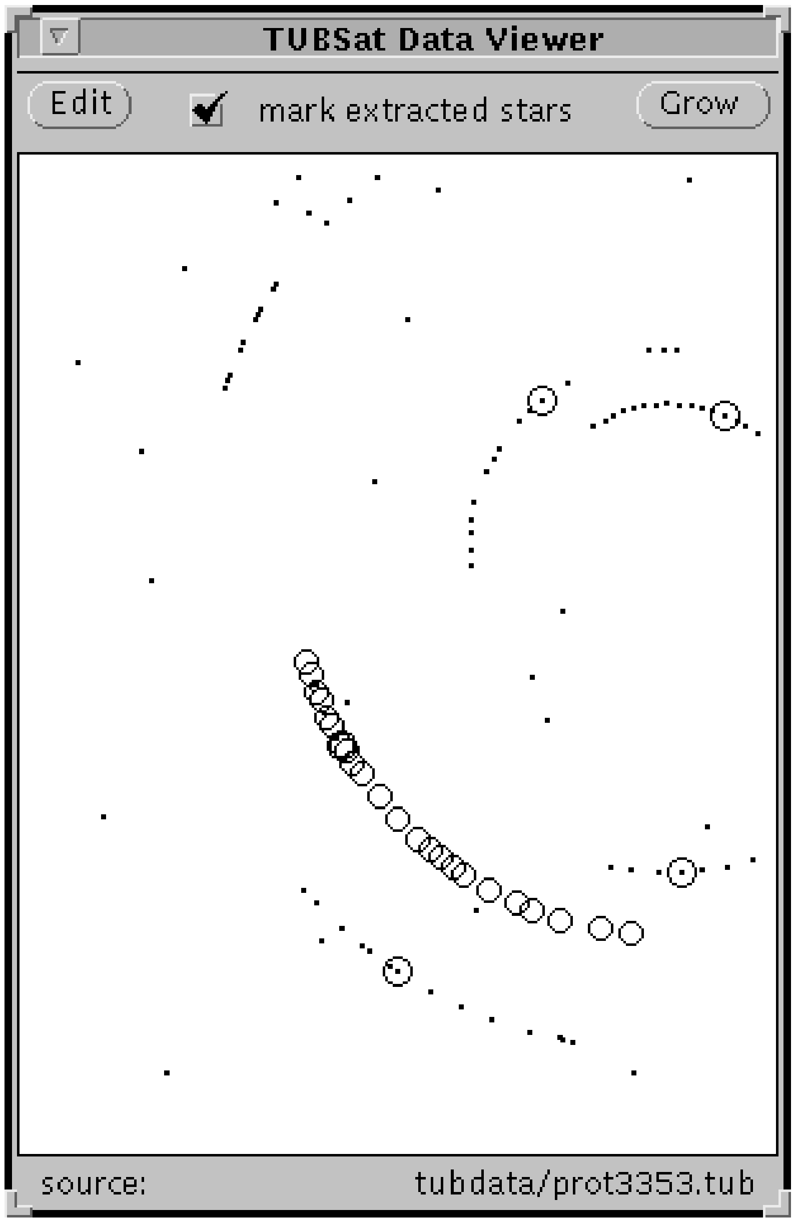
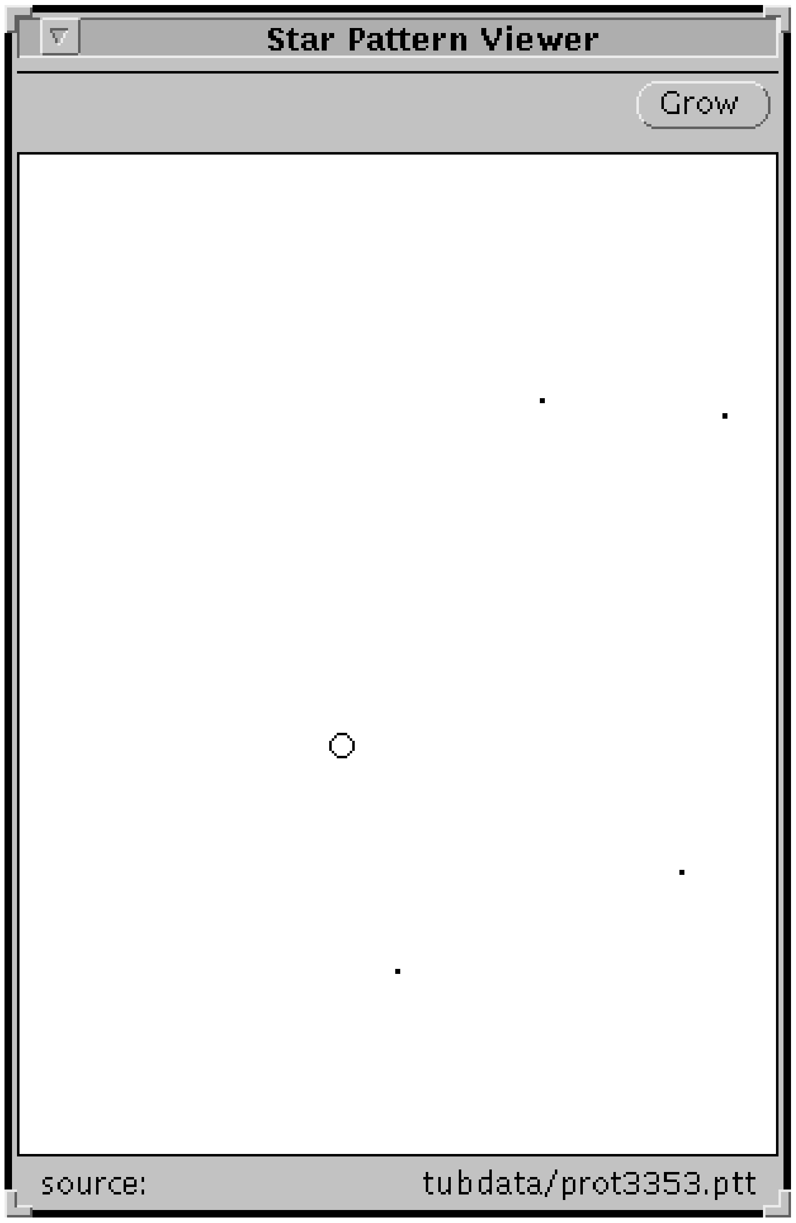

Star Matchings Viewer
Next
Prev
mark matches
Grow
μAnd
λAnd
οAnd
ζCas
αCas
ηCas
βC
δCe
23h0'
24h0'
1h0'
40d0'
50d0'
60d0'
name
stars
constellations
edge
mag. 6
match no 5 of 10
34 pix deviation

Figure 8. The Lion.

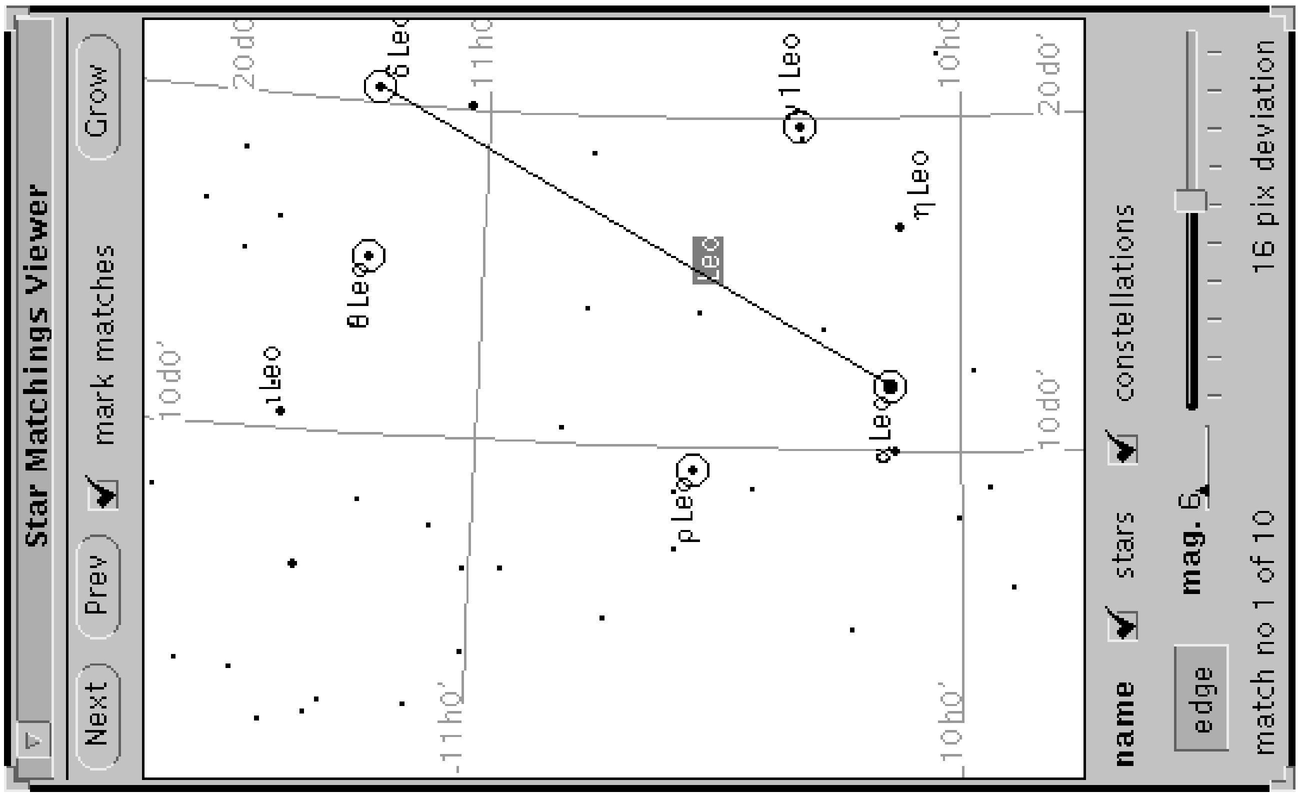
a)



b)



c)


Star Matchings Viewer
Next
Prev
Grow
mark matches
-11h0'
11h0
10h0'
10h0
20d0
10d0'
20d0'
ι Leo
θ Leo
δ Leo
ρ Leo
α Leo
γ 1 Leo
η Leo
Leo
name
stars
constellations
edge
mag. 6
match no 1 of 10
16 pix deviation

d)

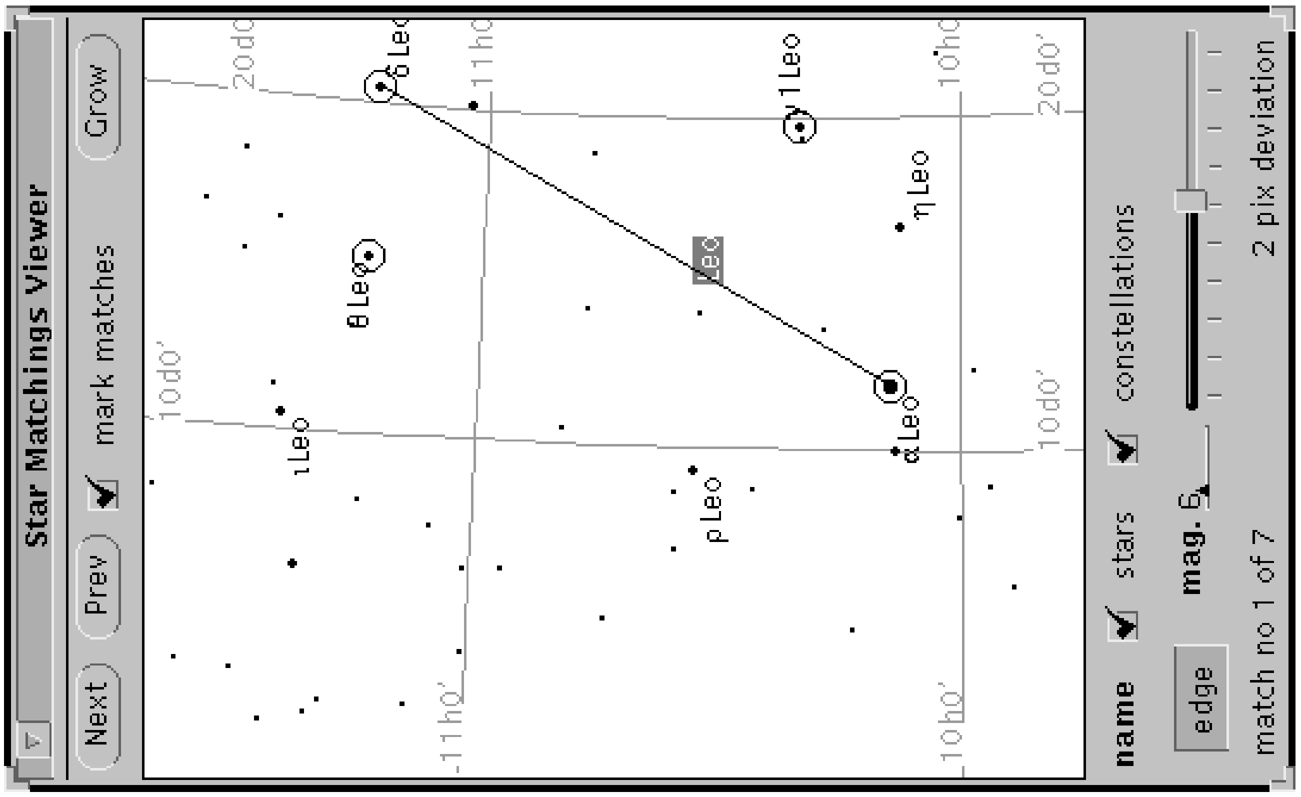
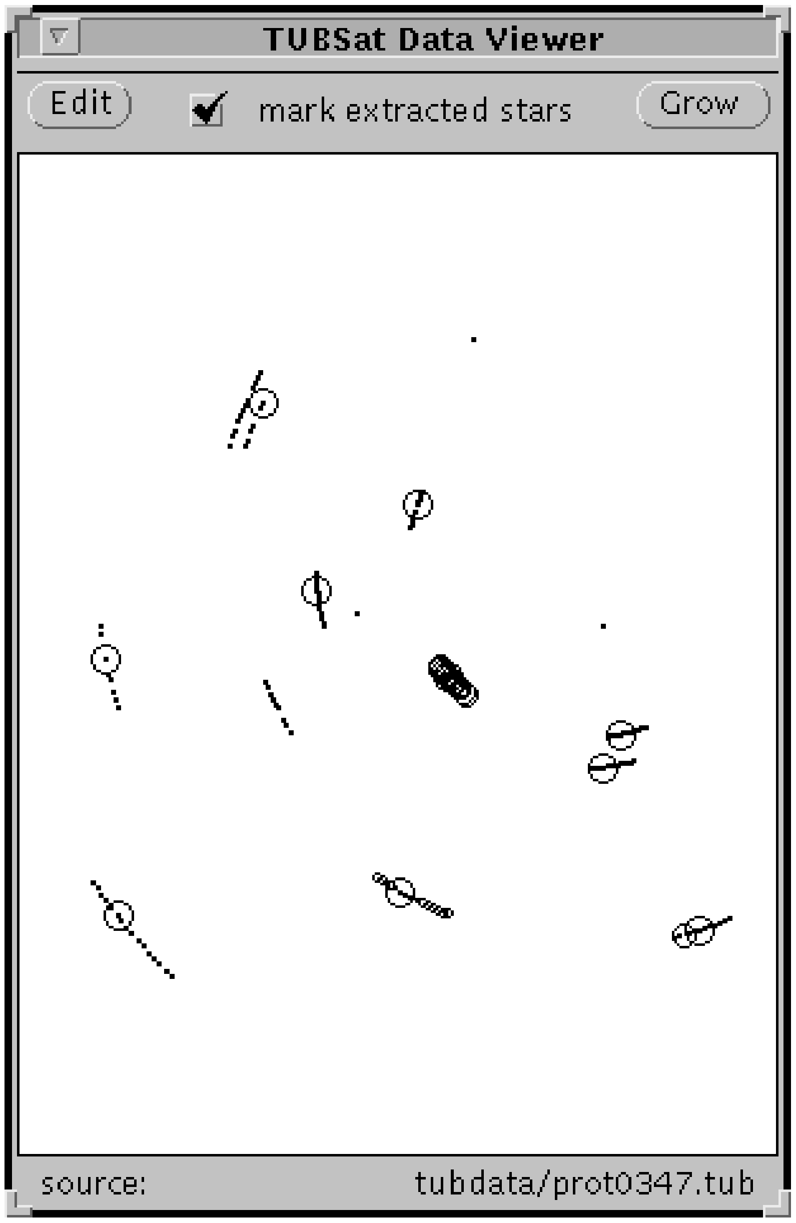
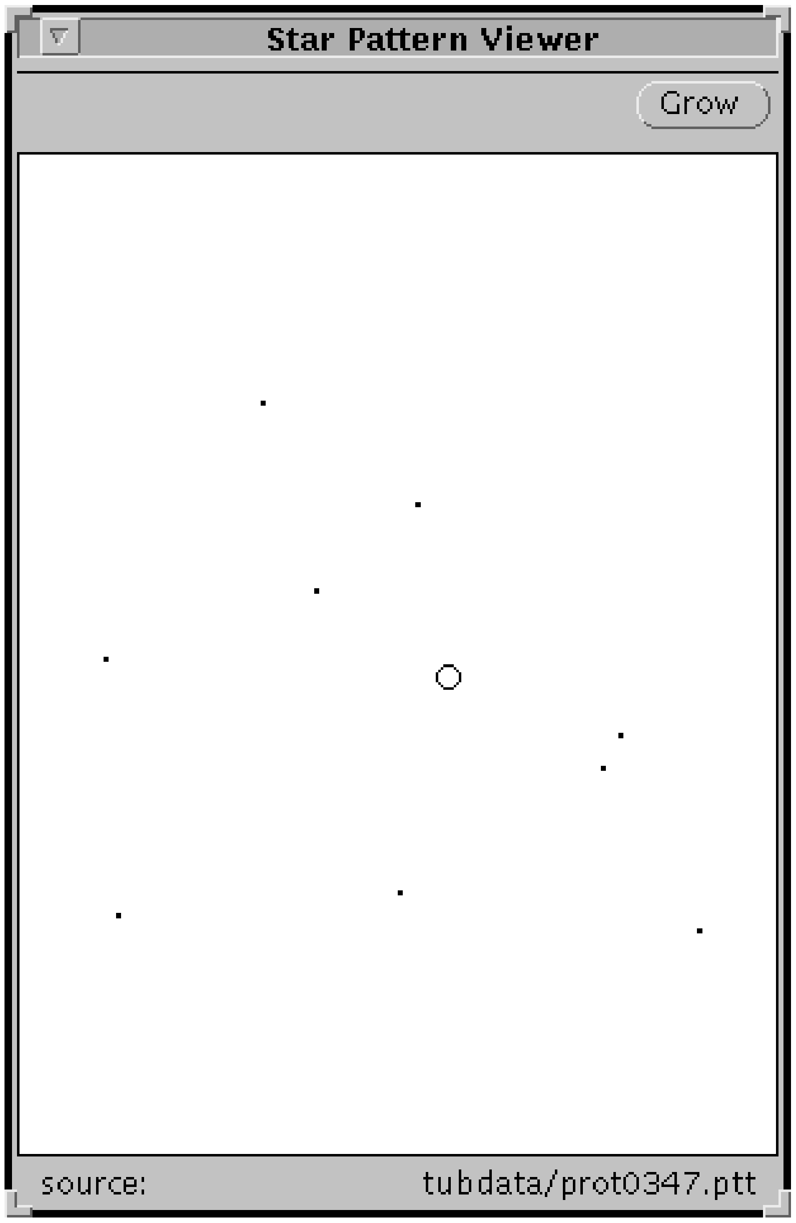

Star Matchings Viewer
Next
Prev
Grow
mark matches
-11h0'
11h0
10h0'
10h0
20d0
10d0'
20d0'
ι Leo
θ Leo
δ Leo
ρ Leo
α Leo
γ 1 Leo
η Leo
Leo
name
stars
constellations
edge
mag. 6
match no 1 of 7
2 pix deviation

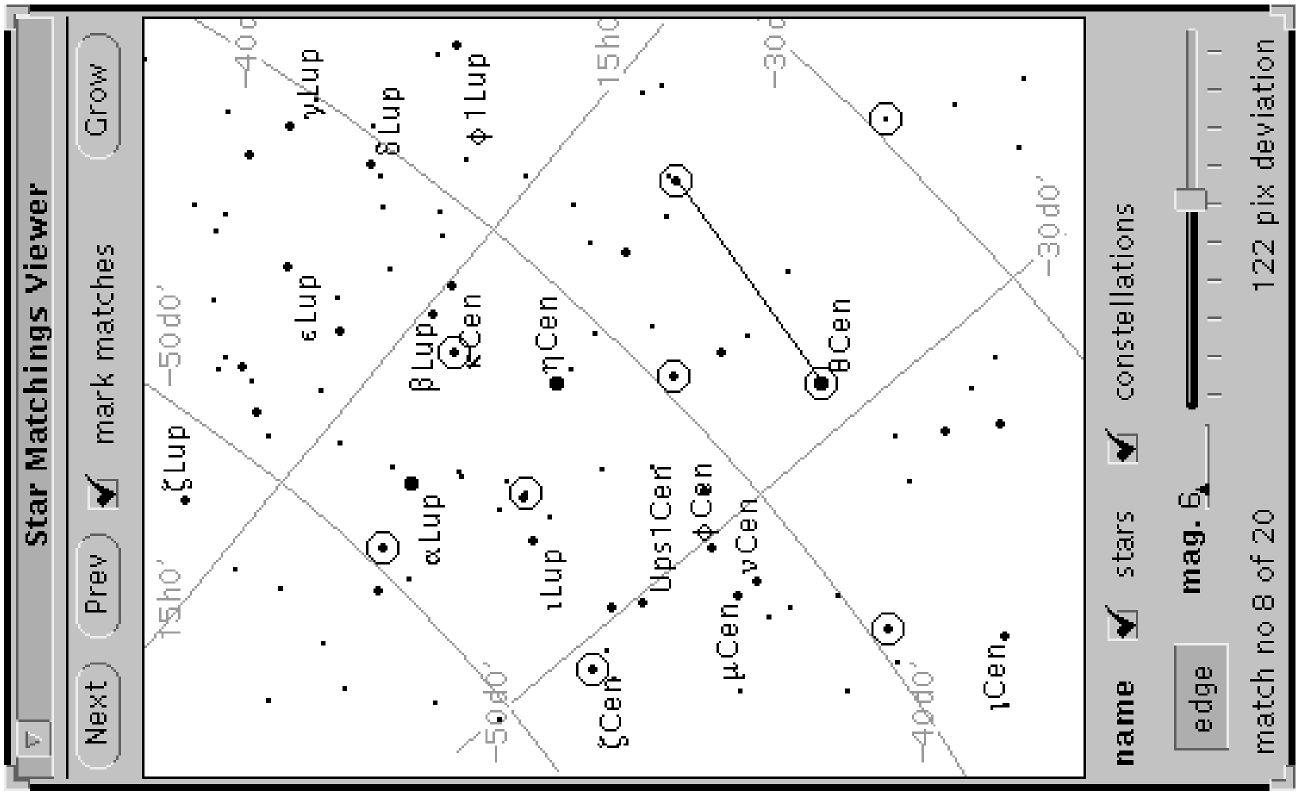
Figure 9. The Swan.

a)



b)

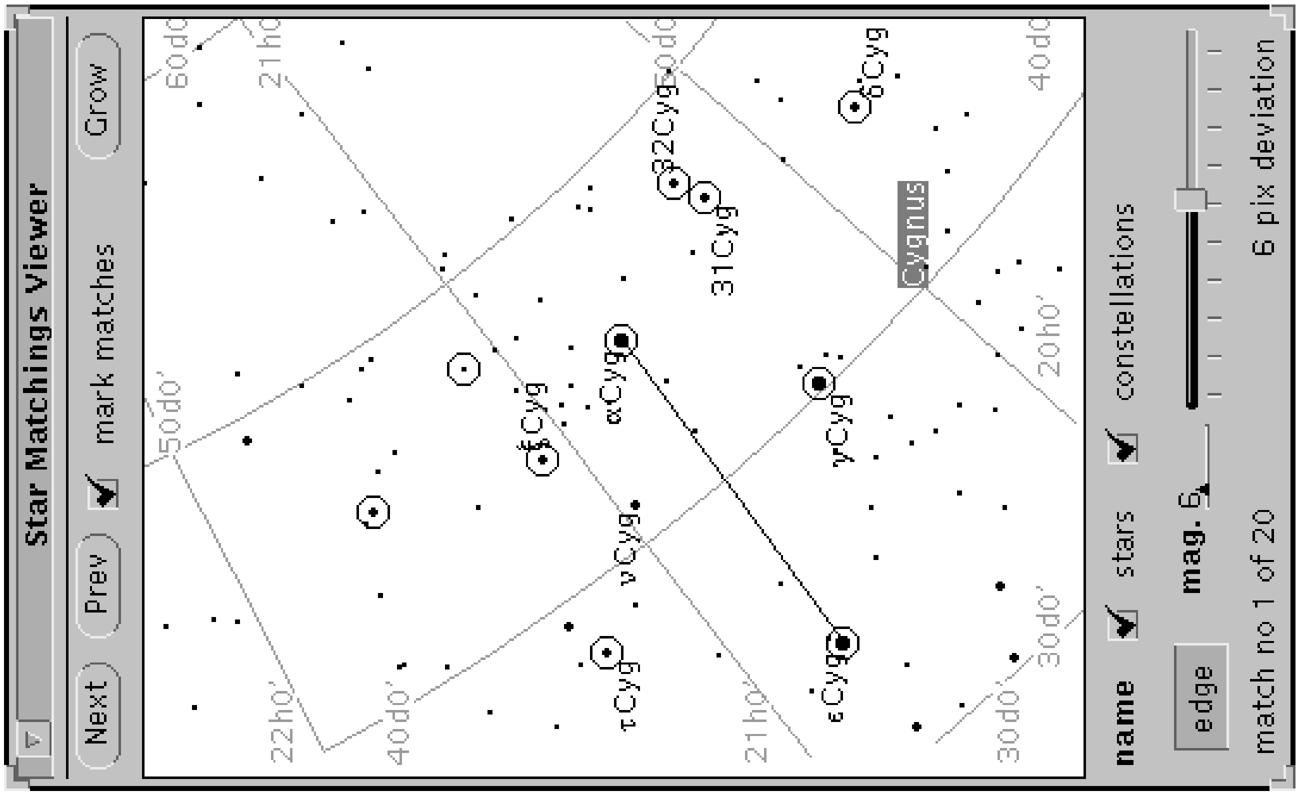




match no 8 of 20

122 pix deviation

d)



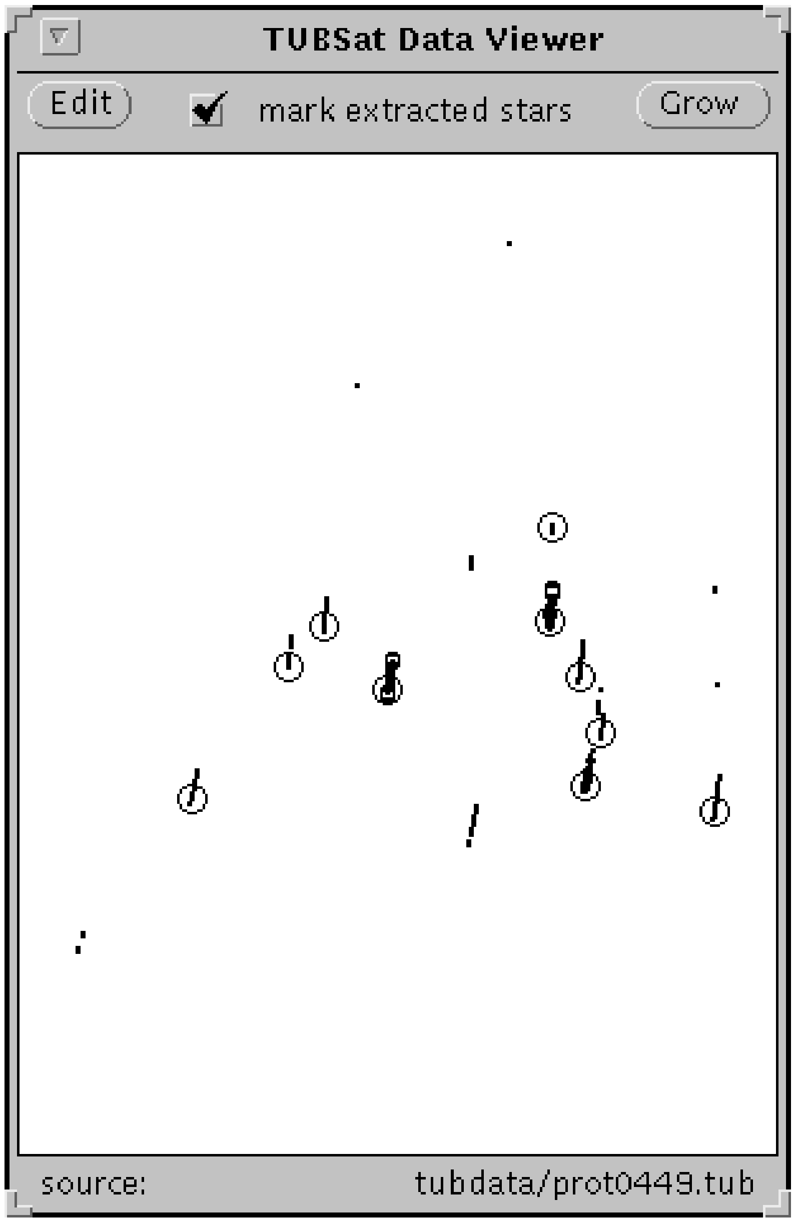
match no 1 of 20

6 pix deviation

c)

Figure 10. Cassiopeia.

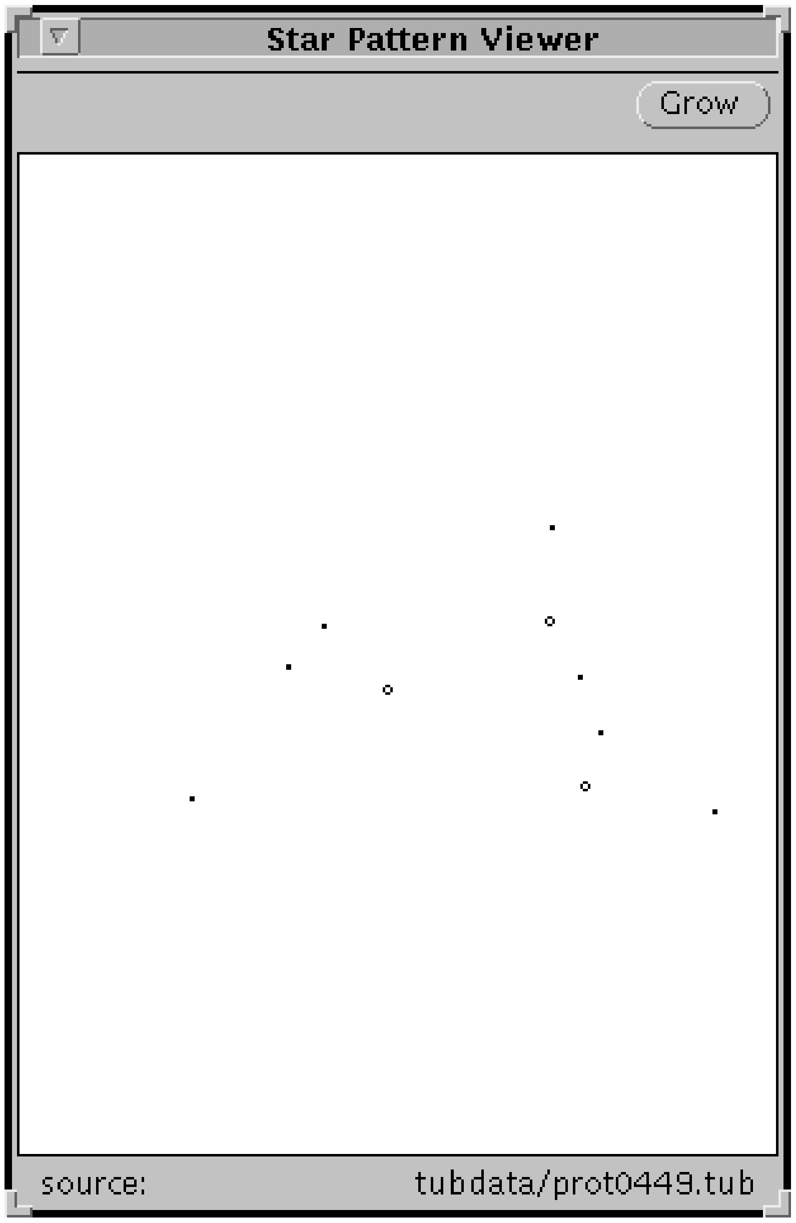
a)



[✓ mark extracted stars]

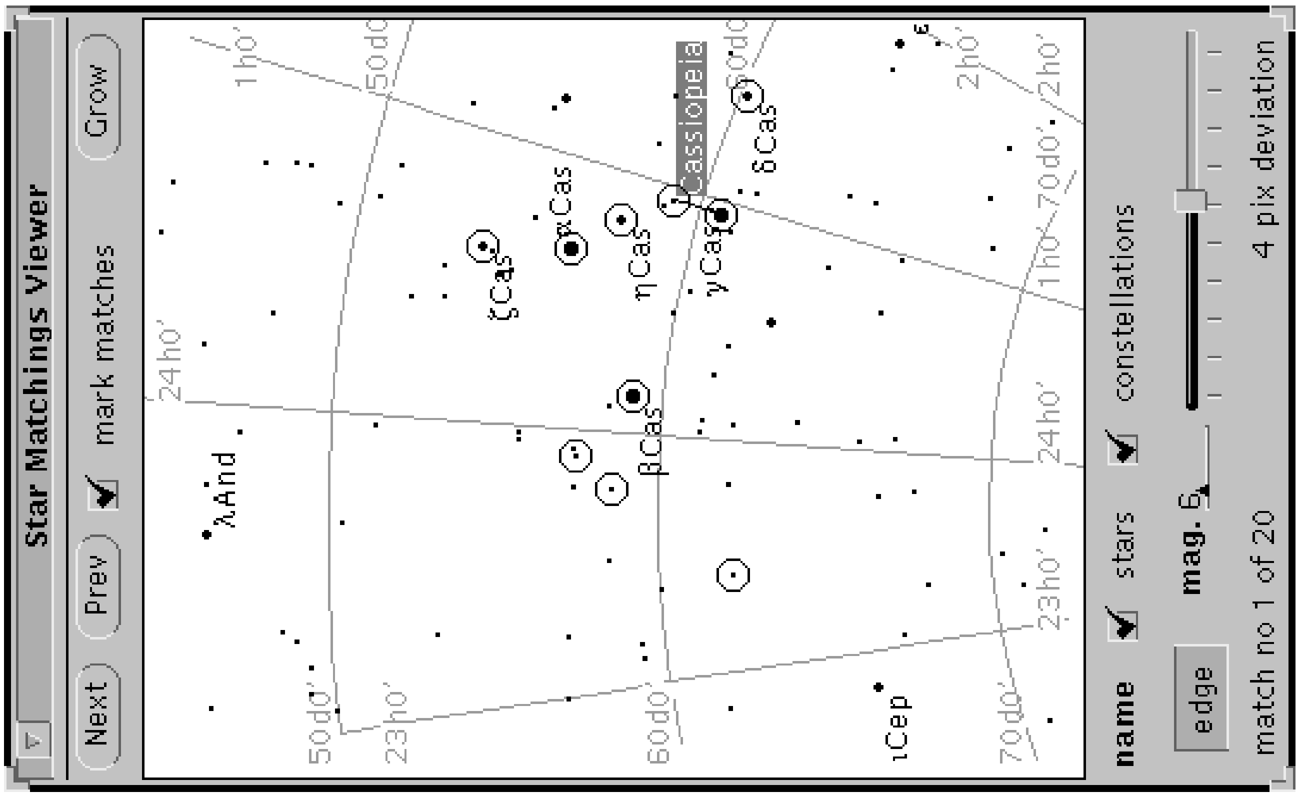
source: tubdata/prot0449.tub

b)



source: tubdata/prot0449.tub

c)



Star Matchings Viewer

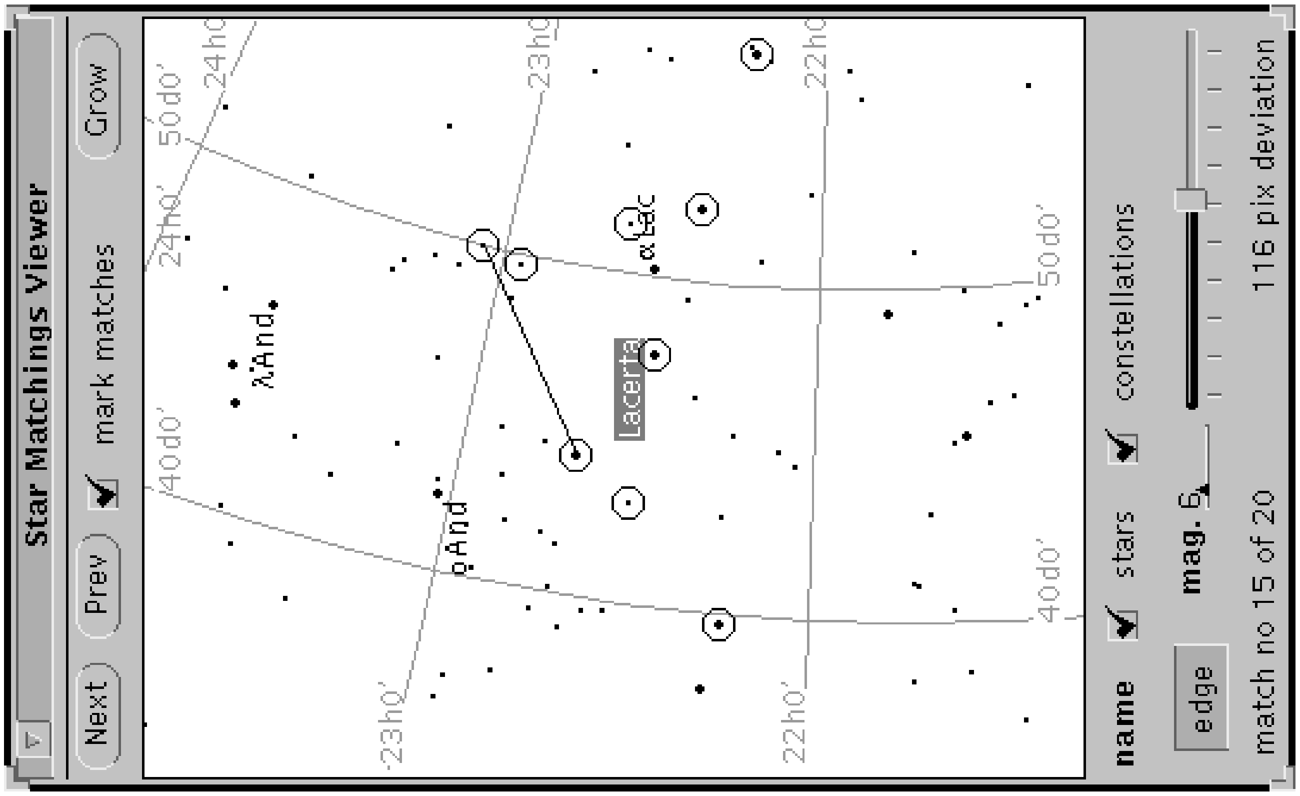
Next Prev Grow ✓ mark matches

name ✓ stars ✓ constellations

edge mag. 6

match no 1 of 20 4 pix deviation

d)



Star Matchings Viewer

Next Prev Grow ✓ mark matches

name ✓ stars ✓ constellations

edge mag. 6

match no 15 of 20 116 pix deviation