BLIND ASTROMETRIC CALIBRATION FOR ALL-SKY CAMERAS: AN AUTOMATIC ALGORITHM BASED ON A BAYESIAN APPROACH

Annalisa Ghiberti

Dipartimento di Matematica Giuseppe Peano
Scuola di Scienze della Natura
M.Sc. in Stochastics and Data Science

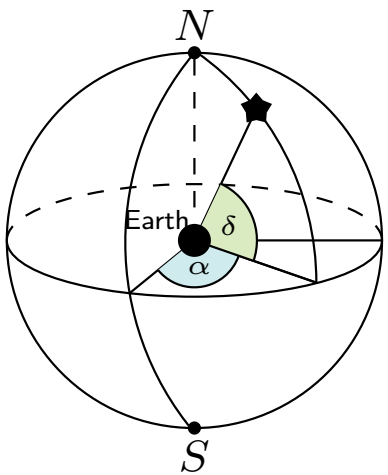
UNIVERSITÀ
DI TORINO

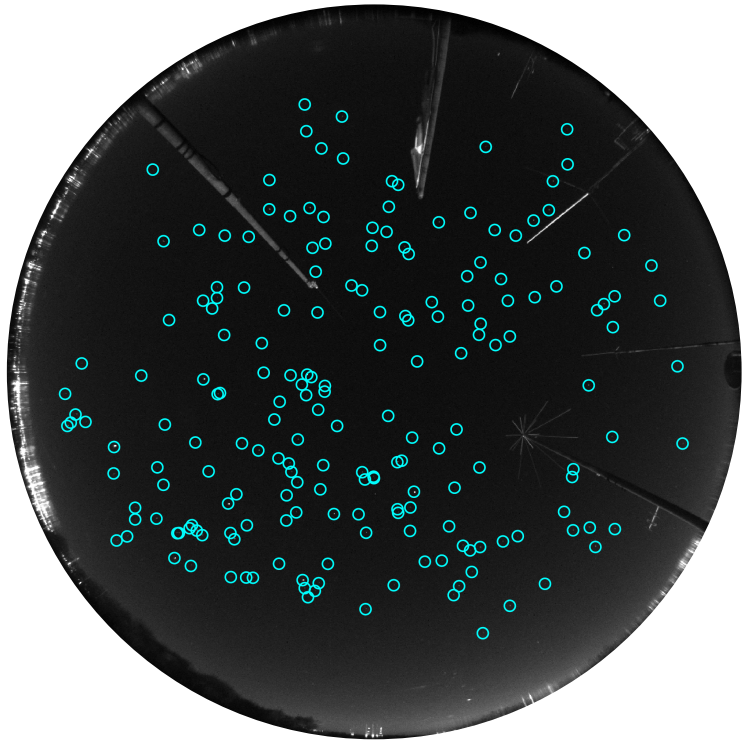
Academic Year 2024/2025

Introduction: astrometric calibration

Astrometric calibration

Astrometric calibration is the process that aligns astronomical images with their true celestial coordinates, determining the geometric transformation that maps image pixel coordinates (x, y) to their correct equatorial coordinates (α, δ), i.e. Right Ascension and Declination.





It allows us to place every detected source at its correct position on the sky, and it is essential for measuring the position, distance, and motion of celestial objects.

PRISMA: currently used astrometric calibration algorithm

PRISMA algorithm for each subset from one Julian Day:

1) Application of a rotation to every image.
2) Computation of Sun and Moon ephemerides.
3) Generation of a median flat frame.
4) Detection of bright sources and association to the respective catalog stars positions. Computation of the expected positions of the stars, using inverse relations of the astrometric projection:

$$\begin{cases} x_{\text{cat}} = x_C + r_{\text{cat}} \cos\left(a_{\text{cat}} - a_0\right) \\ y_{\text{cat}} = y_C + r_{\text{cat}} \sin\left(a_{\text{cat}} - a_0\right) \end{cases}$$

$$(x, y) \rightarrow (a, z) \rightarrow (a_{\text{cat}}, z_{\text{cat}}) \rightarrow (x_{\text{cat}}, y_{\text{cat}})$$

The algorithm is implemented using iteration: it starts from very bright stars and gradually increases the magnitude limit.

Summary

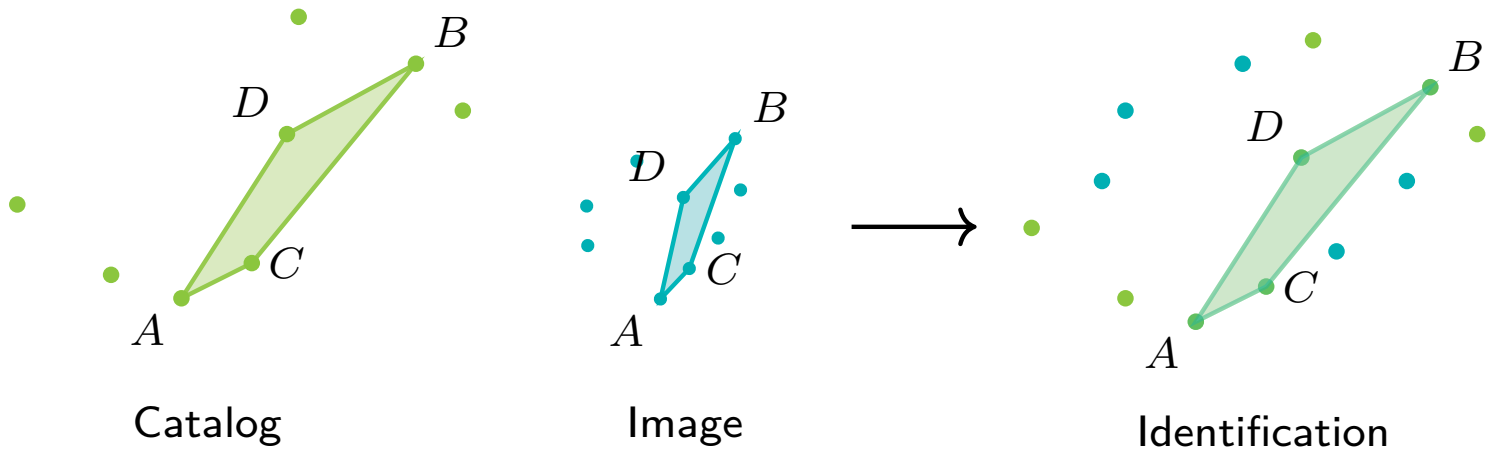
New method for the astrometric calibration of all-sky camera images

The method, introduced by *Astrometry.net*, needs to be adapted to handle the severe geometric distortions present in all-sky images.

(1) Identify patterns of four stars — ***quads*** — in the image and match them with quads from a reference star catalog.

(2) Every match is tested by a verification criterion.

(3) Once a good match is found, the positions of the four stars are used to compute the transformation between image coordinates and celestial coordinates.



Astrometry.net: preparation of the reference star catalog

Application of the bright source detection process to the image.

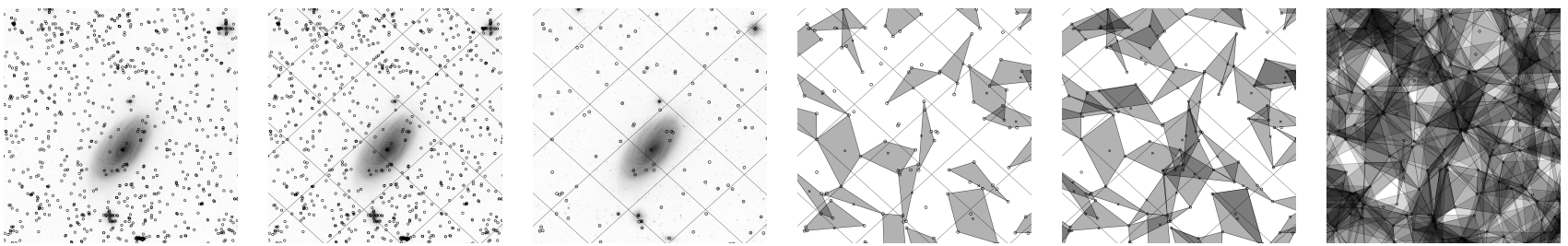
↓

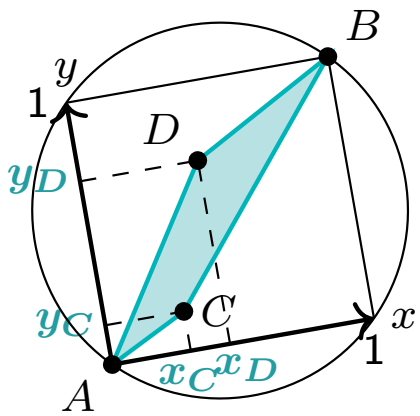
Loading of the reference star catalog, generation of *quads* and computation of their *geometric hash codes*.

↓

Placing of a grid of equal-area sections over the sky and selection of a fixed number of stars. Each cell is visited to find a valid quad, but the algorithm skips the stars that have been used too many times.

The process is repeated a certain number of times, and in the end the restriction on the number of times a star can be used is removed for cells without acceptable quads.



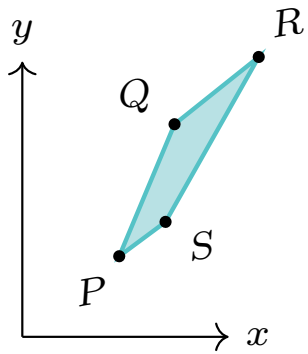


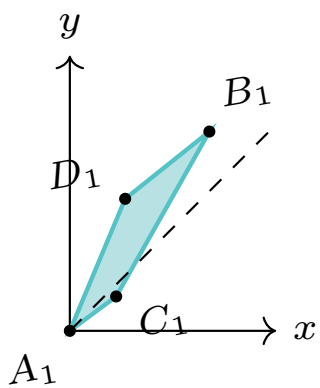
Hash code

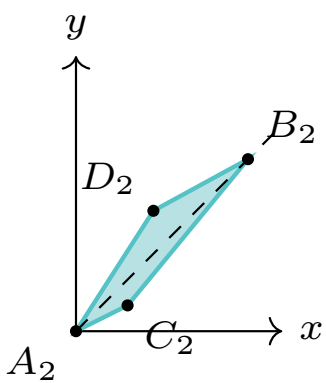
The most widely separated pair of stars is identified to define a local coordinate system, and the ***hash code*** is the four-dimensional vector given by the positions of the other pair of stars.

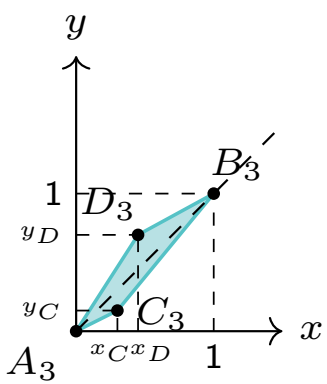
Figure: Quad hash code: (x_C, y_C, x_D, y_D)

We require the following conditions: $\boxed{x_C \leq x_D} \wedge \boxed{x_C + x_D \leq 1}$. This unique code vector is invariant to translation, rotation and scaling of stars positions.









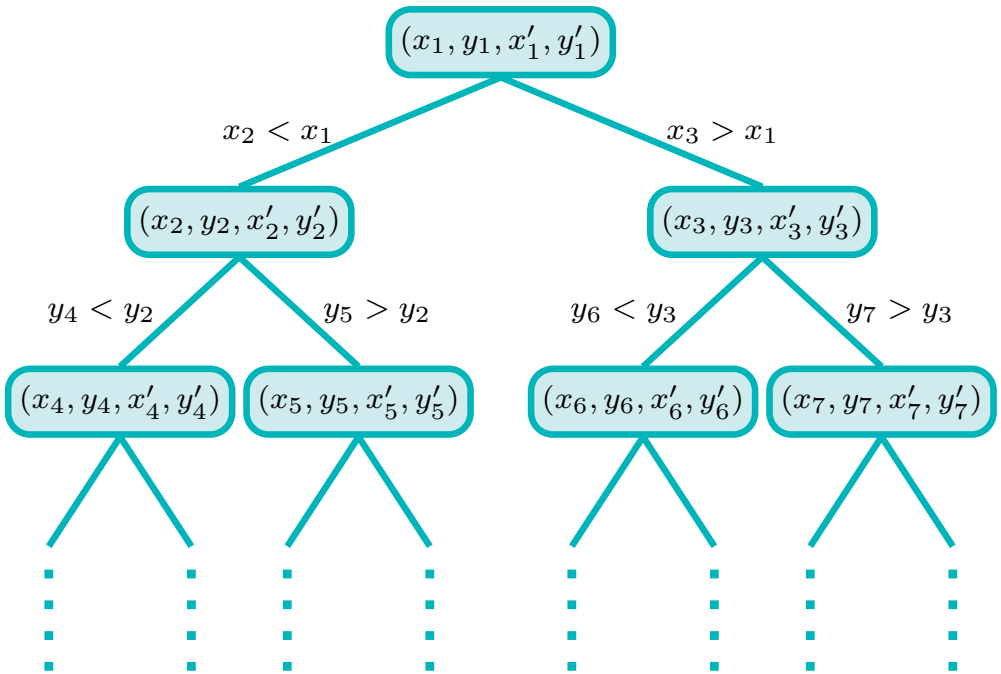


Figure: KD-tree of geometric hash codes.

K-dimensional tree

A ***KD-tree*** is a binary tree in which every node represents a point in a K-dimensional space. In this case, points are geometric hash codes and the space has 4 dimensions.

It has the following structure:

- every node contains a point and the references to its left child and right child nodes;
- every level is divided along a different dimension.

Nearest Neighbor Search algorithm

Once the tree is created, almost identical hash codes are found using the ***NNS*** method:

1) Starting from the root, the algorithm computes the distance between it and the hash code in question.
2) It goes down following the side closer to the query.
3) When a leaf is reached, it saves the best point found.
4) It comes back going up, checking if there are better points in the other subtree. If the distance between the query and the best point is greater than the distance from the "*division plane*", it has to search in the other branch.
5) Repeat until the root node is reached and the closest hash code is returned.

A verification criterion that exploits a Bayesian decision problem is applied.
The hypothesized alignment is computed and other stars that are within the bounds of the image are retrieved.

Hypothesis verification

1) Computation of the ***Bayes factor*** K, where
   - D is our data
   - F is the ***foreground*** model
   - B is the ***background*** model

$$K = \frac{p(D|F)}{p(D|B)} = \prod_{i=1}^{N_t} \frac{p(t_i|F)}{p(t_i|B)} = \frac{\prod_{i=1}^{N_t} \left(\frac{d}{A} + \frac{1-d}{N_r} \sum_{j=1}^{N_r} \mathcal{N}(t_i|r_j, \sigma_{i,j}^2) \right)}{\prod_{i=1}^{N_t} \frac{1}{A}}$$

2) If $K > 10^9$, the astrometric solution is accepted.
3) Else it is refused and the algorithm tries with another match.

PRISMA: preparation of the reference star catalog

Change of coordinates
$(\alpha, \delta) \longrightarrow (a, z) \in [0°, 360°] \times [0°, 90°]$

↓

Division of the catalog into spatial sections of $10° \times 10°$ areas.

↓

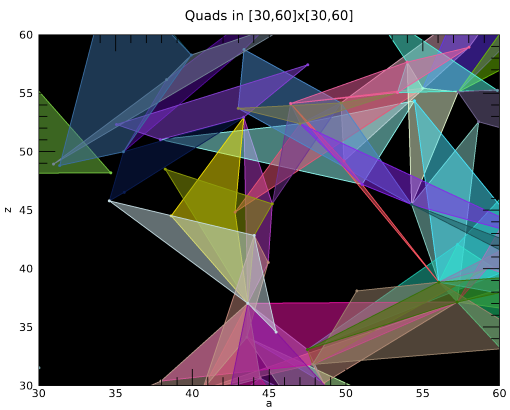
For each section, selection of a random star (a_1, z_1)
and compilation of a list of neighboring stars:
$a_1 - 10° \leq a \leq a_1 + 10° \ \wedge \ z_1 - 10° \leq z \leq z_1 + 10°$

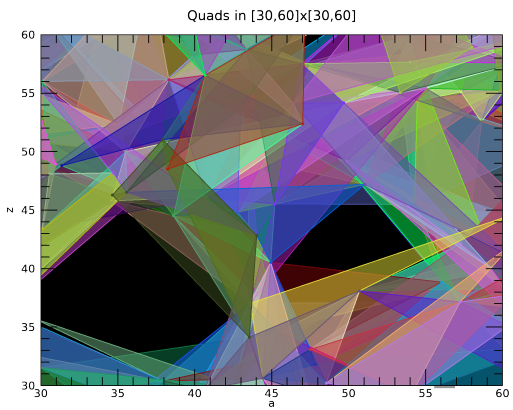
↓ If the list contains at least three elements

Attempt to form a quad.

↓ If its center lies within the section under consideration

Addition of the valid quad to the list and computation of its hash code.





PRISMA: preparation of the input image and search for a candidate match

From the image, PRISMA star detection algorithm extracts a list of pixel coordinates of bright sources. To account for image distortion, we apply:

$$(x, y) \longrightarrow (\theta, r) \longrightarrow (\theta, \varphi)$$

Using these angular coordinates – directly comparable to the ones adopted for the catalog – we repeat the procedure:

Division of detected bright sources into the same spatial sections.

↓

Addition of valid quads to a new list and computation of their hash codes.

The selection of a random hash code and the application of the NNS procedure to the KD-tree produce a **candidate match**:

- $A : (x_1, y_1) \longleftrightarrow star_1 = (x_1^{cat}, y_1^{cat})$
- $B : (x_2, y_2) \longleftrightarrow star_2 = (x_2^{cat}, y_2^{cat})$
- $C : (x_3, y_3) \longleftrightarrow star_3 = (x_3^{cat}, y_3^{cat})$
- $D : (x_4, y_4) \longleftrightarrow star_4 = (x_4^{cat}, y_4^{cat})$

Computation of the Bayes Factor K

- The ***test*** list consists of the bright sources detected in the image.
- The ***reference*** list contains catalog stars after being rotated according to the candidate match.

$$\boldsymbol{K} := \log_{10} K = \sum_{i=1}^{N_t} \left[\log_{10} \left(\frac{d}{A} + \frac{1-d}{N_r} \sum_{j=1}^{N_r} \mathcal{N}(t_i | r_j, \sigma_{i,j}^2) \right) + \log_{10}(A) \right]$$

- $A = 400 \times 400$ px^2 is the area of the selected square region centered at $\boldsymbol{C}$
- N_t and N_r are the numbers of test and reference stars in the selected region
- $d = \max \left\{ 1 - \frac{N_r}{N_t}; 0.1 \right\}$ is the estimate of the fraction of distractors
- $\sigma_{i,j}^2 = 5$ is an estimate for the positional variance

The hypothesis is accepted if $\boldsymbol{K}$ is larger than a certain threshold.
Otherwise, the algorithm selects another random hash code and repeats the procedure.

PRISMA: analysis aimed at finding a suitable threshold for K

To establish a suitable threshold for K, we analyzed its behavior with 1200 catalog stars while introducing different numbers of *distractors* – bright sources different from stars:

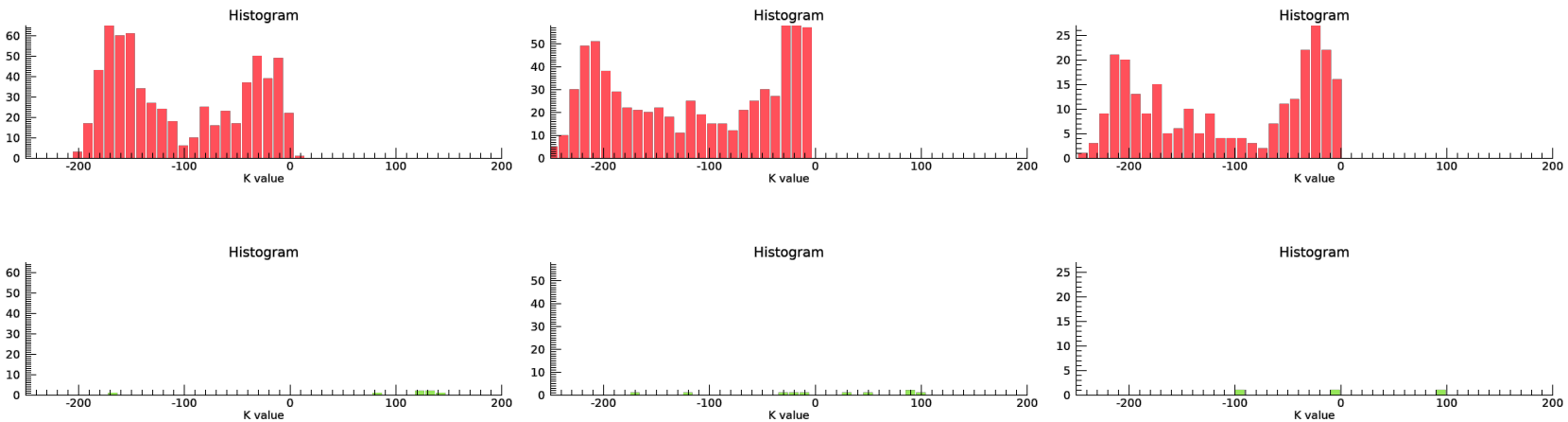


Figure: K frequencies for simulations with 100, 300 and 500 distractors. Red bars correspond to incorrect solutions, while green bars correspond to good estimates for the astrometric calibration.

The estimated threshold remains approximately $K = 10$ in every simulation and can therefore be adopted for real PRISMA data.

PRISMA: results of a simulation

V is : 0.1264 °/px
a_0 is : 5.000°
$\boldsymbol{K} = 87.626 \Longrightarrow$ Attempt n. 195 : the hypothesis is accepted.
The Nearest Neighbor of our random code

(-0.053, 0.228, 0.003, 0.511)

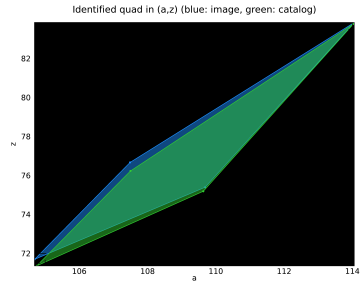
is:

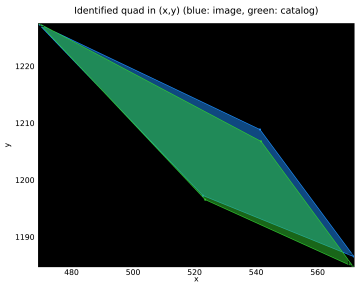
(-0.064, 0.238, -0.016, 0.529)

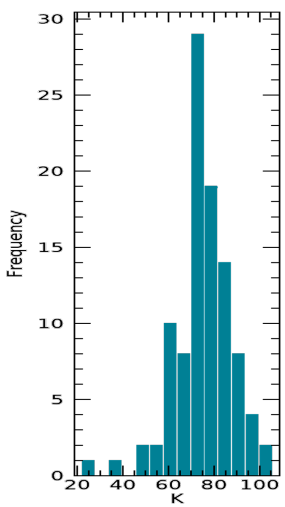
The stars we identified thanks to this match are

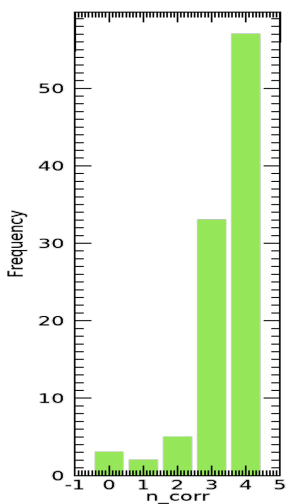
- A: $(180.245^\circ, 78.286^\circ) \longleftrightarrow$ *-f-Eri* $= (177.563^\circ, 80.312^\circ)$
- B: $(172.503^\circ, 74.910^\circ) \longleftrightarrow$ *-43-Eri* $= (169.917^\circ, 77.467^\circ)$
- C: $(180.040^\circ, 76.906^\circ) \longleftrightarrow$ *-g-Eri* $= (177.327^\circ, 78.911^\circ)$
- D: $(179.111^\circ, 75.454^\circ) \longleftrightarrow$ *-i-Eri* $= (176.378^\circ, 77.498^\circ)$

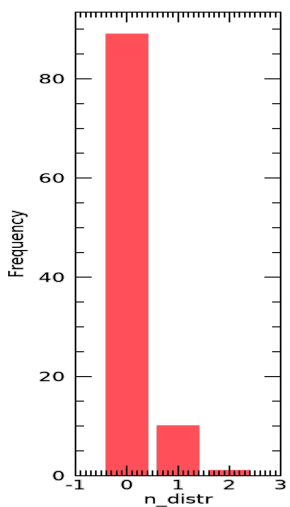
Number of correctly identified stars = 4
Number of distractors = 0











PRISMA: results for a real image taken by ITTO09 camera (2024-11-03, 00:01:40)

V is : 0.1276 $^\circ$/px
a_0 is : 355.105 $^\circ$
$\boldsymbol{K}$= 44.202 $\Longrightarrow$ Attempt n. 176 : the hypothesis is accepted.
The Nearest Neighbor of our random code

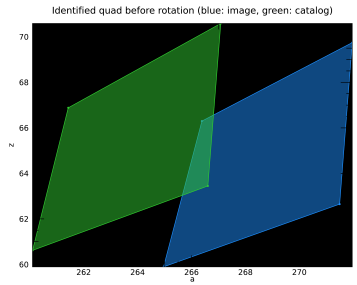
(0.193, 0.810, 0.711, 0.289)

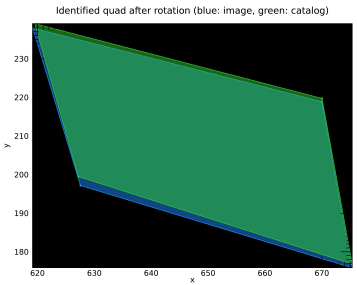
is:

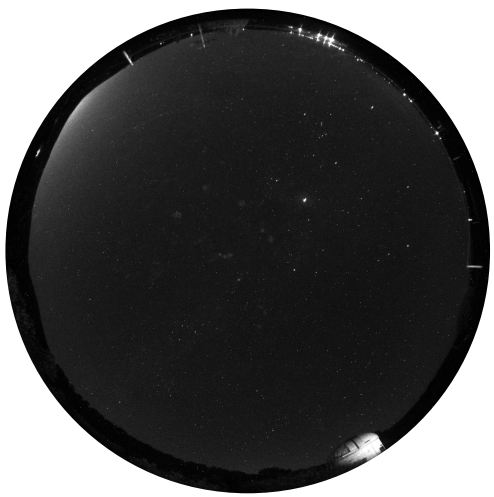
(0.199, 0.807, 0.722, 0.310)

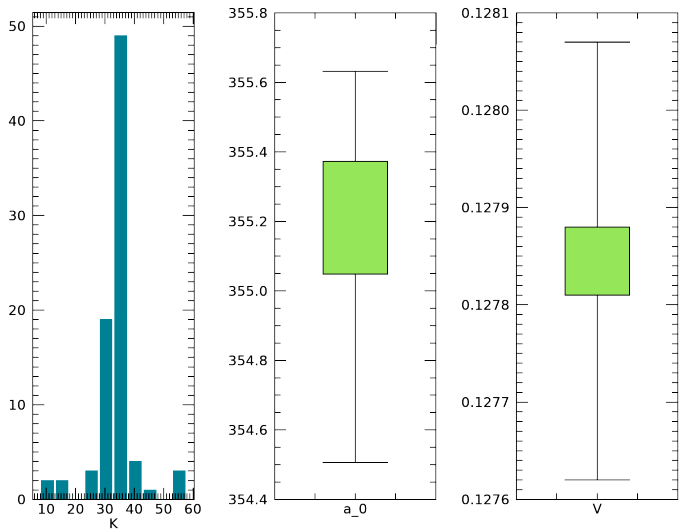
The stars we identified thanks to this match are

- A: (675.237 px, 175.876 px) $\longleftrightarrow$ **-42-Peg* = (675.351 px, 176.805 px)
- B: (619.131 px, 237.994 px) $\longleftrightarrow$ **-70-Peg* = (619.509 px, 239.115 px)
- C: (669.992 px, 218.910 px) $\longleftrightarrow$ **-alf-Peg* = (670.028 px, 219.737 px)
- D: (627.550 px, 197.155 px) $\longleftrightarrow$ **-55-Peg* = (627.193 px, 199.482 px)









PRISMA: results for a real image taken by ITER10 camera (2024-06-26, 21:04:22)

V is : 0.1272°/px
a_0 is : 8.849°
$\boldsymbol{K}$= 120.262 $\Longrightarrow$ Attempt n. 21 : the hypothesis is accepted.
The Nearest Neighbor of our random code

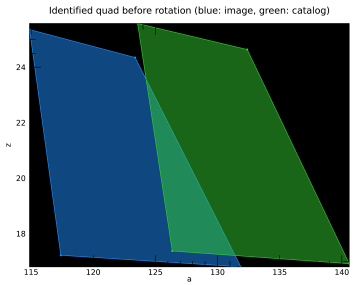
(0.269, 0.576, 0.633, -0.012)

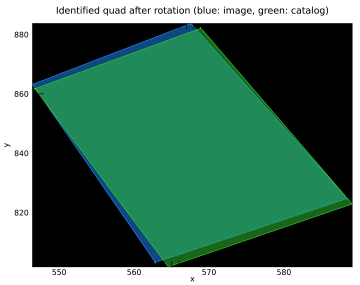
is:

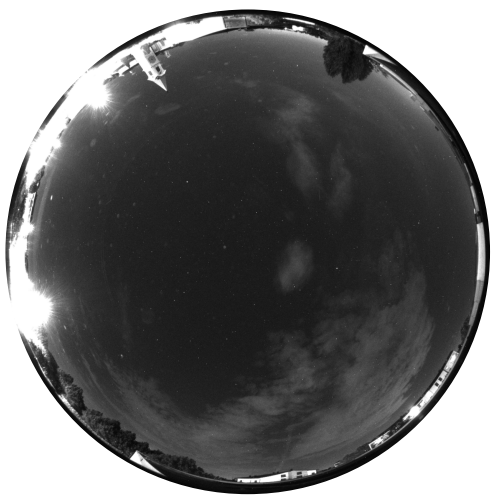
(0.268, 0.599, 0.641, 0.008)

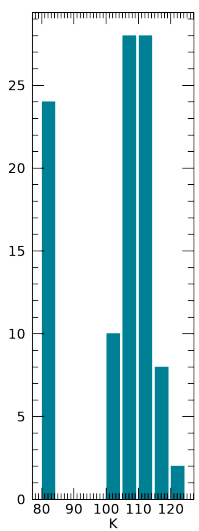
The stars we identified thanks to this match are

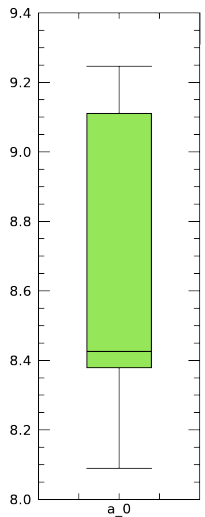
- A: (567.591 px, 883.730 px) $\longleftrightarrow$ **-mu.01-Her* = (568.922 px, 882.135 px)
- B: (562.840 px, 803.230 px) $\longleftrightarrow$ **-eps-Her* = (564.567 px, 801.724 px)
- C: (546.446 px, 863.218 px) $\longleftrightarrow$ **-lam-Her* = (546.770 px, 861.738 px)
- D: (588.454 px, 824.787 px) $\longleftrightarrow$ **-u-Her* = (589.127 px, 822.904 px)

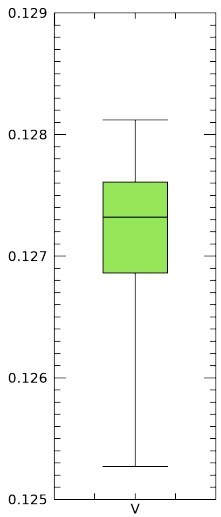












PRISMA: results for a real image taken by ITER10 camera (2024-06-26, 21:44:22)

V is : $0.1278°/\text{px}$
a_0 is : $8.794°$
$\boldsymbol{K}= 10.335 \Longrightarrow$ Attempt n. 12 : the hypothesis is accepted.
The Nearest Neighbor of our random code

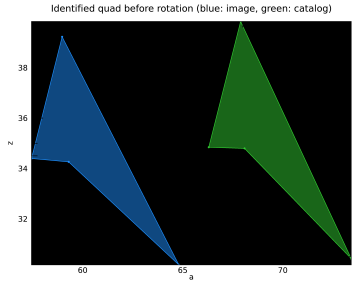
(0.338, 1.062, 0.372, 0.820)

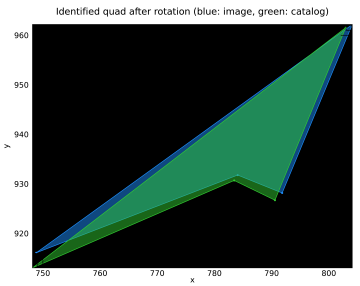
is:

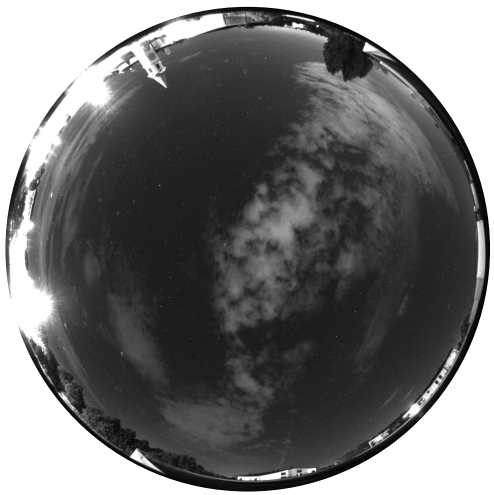
(0.323, 1.037, 0.375, 0.811)

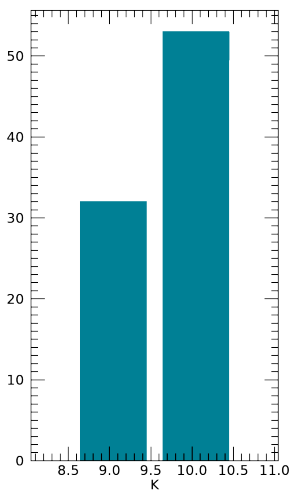
The stars we identified thanks to this match are

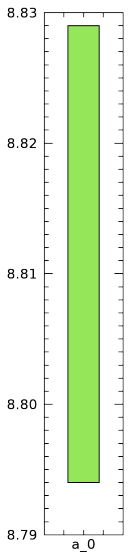
- A: (748.842 px, 916.098 px) $\longleftrightarrow$ **-del-Cyg* = (748.200 px, 913.058 px)
- B: (804.095 px, 962.181 px) $\longleftrightarrow$ **-alf-Cyg* = (802.955 px, 961.535 px)
- C: (791.835 px, 928.167 px) $\longleftrightarrow$ **-omi02-Cyg* = (790.574 px, 926.712 px)
- D: (784.058 px, 931.782 px) $\longleftrightarrow$ **-31-Cyg* = (783.449 px, 930.696 px)

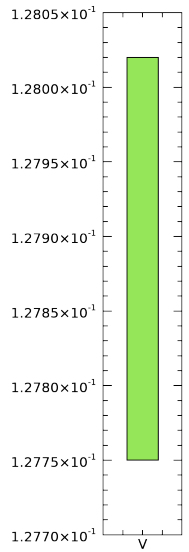












PRISMA: results for a real image taken by ITLO06 camera (2023-04-26, 20:04:23)

V is : 0.1665°/px
a_0 is : 336.474°
$\boldsymbol{K}$= 38.886 $\Longrightarrow$ Attempt n. 10 : the hypothesis is accepted.
The Nearest Neighbor of our random code

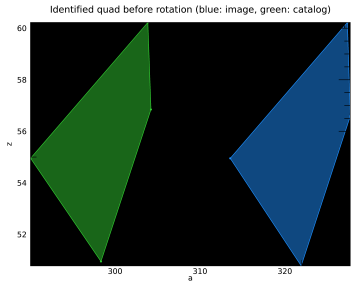
(-0.172, 0.304, 0.106, 1.037)

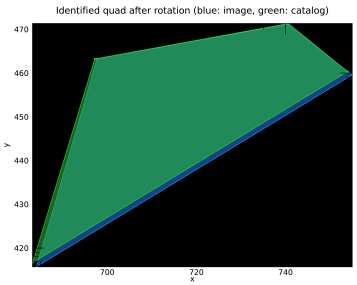
is:

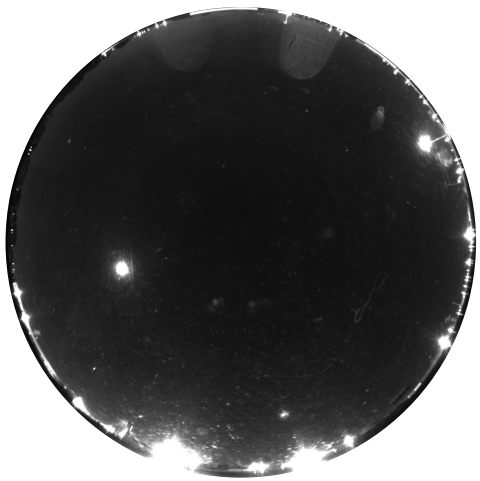
(-0.164, 0.280, 0.119, 1.025)

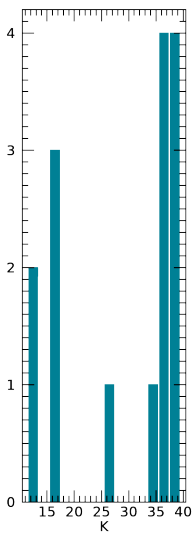
The stars we identified thanks to this match are

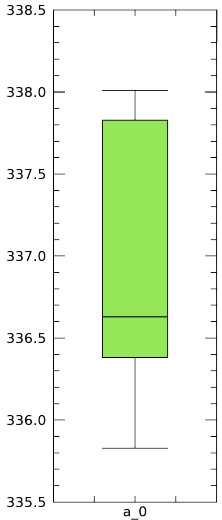
- A: (754.963 px, 459.597 px) $\longleftrightarrow$ **-eps-Aur* = (753.933 px, 460.348 px)
- B: (684.100 px, 415.706 px) $\longleftrightarrow$ **-tet-Aur* = (683.240 px, 416.641 px)
- C: (740.465 px, 471.148 px) $\longleftrightarrow$ **-alf-Aur* = (740.639 px, 471.347 px)
- D: (697.685 px, 463.268 px) $\longleftrightarrow$ **-bet-Aur* = (697.195 px, 463.244 px)

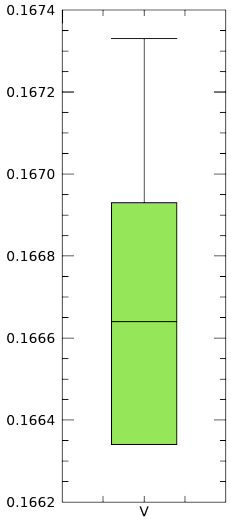












Conclusions

- Overall, the algorithm provides a solid starting point for a new reliable astrometric calibration of all-sky cameras.
- Future work could focus on improving the model, handling varying star visibility more effectively, and increasing robustness under challenging observing conditions.

- It is possible to adapt blind astrometric calibration techniques to all-sky images, as long as the geometric distortions are properly accounted for.
- Tests on PRISMA data confirm that the method performs reliably under favorable conditions.

THANK YOU
for your
ATTENTION

Bibliography

[1] D. Barghini, D. Gardiol, A. Carbognani, and S. Mancuso, "*Astrometric Calibration for All-sky Cameras revisited*," *Astronomy & Astrophysics*, vol. 626, A105, 2019.

[2] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis, "*Astrometry.net: Blind Astrometric Calibration of Arbitrary Astronomical Images*," *The Astronomical Journal*, vol. 139, pp. 1782–1800, 2010.

[3] D. Lang, "*Astrometry.net: Automatic recognition and calibration of astronomical images*," Ph.D. dissertation, University of Toronto, 2009.

[4] M. Skrodzki, "*The k-d tree data structure and a proof for neighborhood computation in expected logarithmic time*,", 2019. arXiv: 1903.04936 [cs.DS].

[5] H. J. Wolfson, "*Geometric Hashing: An Overview*," *IEEE Computational Science & Engineering*, vol. 4, pp. 10–21, 1997.

[6] J. Gerhard, "*A geometric hashing technique for star pattern recognition*," Ph.D. dissertation, WVU, 2016.

Hash Code: pseudocode (1/2)

Hash Code for Quads

1: **function** BUILDHASHCODE($quad$)
2: $P \leftarrow quad.point1$
3: $Q \leftarrow quad.point2$
4: $R \leftarrow quad.point3$
5: $S \leftarrow quad.point4$
6: $distances \leftarrow \left[\overline{PQ}, \overline{PR}, \overline{PS}, \overline{QR}, \overline{QS}, \overline{RS}\right]$
7: $m \leftarrow \max(distances)$
8: $A \leftarrow$ one endpoint of the segment
9: $B \leftarrow$ other endpoint of the segment
10: $C \leftarrow$ one of the remaining two points
11: $D \leftarrow$ last remaining point

▷ First step: **translation**

12: $A_1 \leftarrow A - A$
13: $B_1 \leftarrow B - A$
14: $C_1 \leftarrow C - A$
15: $D_1 \leftarrow D - A$

▷ Second step: **rotation**

16: $\theta \leftarrow \frac{\pi}{4} - \arctan\left(\frac{y_{B_1}}{x_{B_1}}\right)$
17: $M_{rot} \leftarrow$ rotation matrix
18: $A_2 \leftarrow M_{rot} \cdot A_1$
19: $B_2 \leftarrow M_{rot} \cdot B_1$
20: $C_2 \leftarrow M_{rot} \cdot C_1$
21: $D_2 \leftarrow M_{rot} \cdot D_1$

Hash Code: pseudocode (2/2)

Hash Code for Quads

21: **function** BUILDHASHCODE($quad$)

▷ Third step: **scaling**

22: $s \leftarrow \frac{1}{y_{B_2}}$
23: $A_3 \leftarrow s \cdot A_2$
24: $B_3 \leftarrow s \cdot B_2$
25: $C_3 \leftarrow s \cdot C_2$
26: $D_3 \leftarrow s \cdot D_2$

▷ Conditions

27: **if** $x_{C_3} \leq x_{D_3} \ \wedge \ x_{C_3} + x_{D_3} \leq 1$ **then**
28: **return** $Code(x_{C_3}, y_{C_3}, x_{D_3}, y_{D_3})$
29: **else if** $x_{C_3} > x_{D_3} \ \wedge \ x_{C_3} + x_{D_3} \leq 1$ **then**
30: **return** $Code(x_{D_3}, y_{D_3}, x_{C_3}, y_{C_3})$
31: **end if**
32: **if** $x_{C_3} \geq x_{D_3} \ \wedge \ x_{C_3} + x_{D_3} > 1$ **then**
33: **return** $Code(1 - x_{C_3}, 1 - y_{C_3}, 1 - x_{D_3}, 1 - y_{D_3})$
34: **else if** $x_{C_3} < x_{D_3} \ \wedge \ x_{C_3} + x_{D_3} > 1$ **then**
35: **return** $Code(1 - x_{D_3}, 1 - y_{D_3}, 1 - x_{C_3}, 1 - y_{C_3})$
36: **end if**
37: **end function**

KD-tree: pseudocode

Algorithm: KD-tree

1: **function** KD-TREE(P)
2: **if** $|P| = 0$ **then**
3: **return** empty tree
4: **else if** $|P| = 1$ **then**
5: **return** tree with only one node
6: **else**
7: $d' \leftarrow depth \ \% \ d$ (or $d' \leftarrow$ most spread dimension)
8: $q \leftarrow$ median according to d'
9: $P_1 \leftarrow \{p_i \in P \mid p_i^{d'} \leq q^{d'}, p_i \neq q\}$
10: $P_2 \leftarrow \{p_i \in P \mid p_i^{d'} \geq q^{d'}, p_i \neq q\}$
11: $H \leftarrow \{x \in \mathbb{R}^d \mid x^{d'} = q^{d'}\}$
12: $T_l \leftarrow$ KD-TREE(P_1)
13: $T_r \leftarrow$ KD-TREE(P_2)
14: **return** node containing q, H with T_l as left child and T_r as right child
15: **end if**
16: **end function**

KD-tree: toy example

$$P = \Big\{(1,7);(9,3);(5,4);(2,5);(8,10);(6,7);(11,2)\Big\} \neq \varnothing$$

1) First step: KD-tree(P)
$d' = 1$
$q = (6,7)$
$P_1 = \{(1,7);(2,5);(5,4)\}$
$P_2 = \{(8,10);(9,3);(11,2)\}$

2) Second step: KD-tree(P_1)
$d' = 2$
$q = (2,5)$
$P_1' = \{(5,4)\}$
$P_2' = \{(1,7)\}$

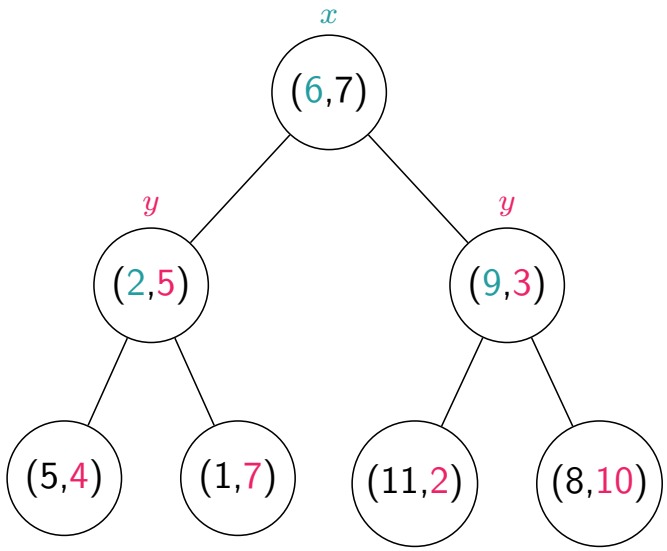
3) Third step: KD-tree(P_1')
$|P_1'| = 1 \rightarrow (5,4)$ is a leaf (left child of $(2,5)$)

3) Third step: KD-tree(P_2')
$|P_2'| = 1 \rightarrow (1,7)$ is a leaf (right child of $(2,5)$)

2) Second step: KD-tree(P_2)
$d' = 2$
$q = (9,3)$
$P_1'' = \{(11,2)\}$
$P_2'' = \{(8,10)\}$

3) Third step: KD-tree(P_1'')
$|P_1''| = 1 \rightarrow (11,2)$ is a leaf (left child of $(9,3)$)

3) Third step: KD-tree(P_2'')
$|P_2''| = 1 \rightarrow (8,10)$ is a leaf (right child of $(9,3)$)



NNS: pseudocode

Algorithm: Nearest Neighbor Search

```
1:  function NNS(current_node, p, current_best)
2:      if current_node is null then
3:          return current_best
4:      end if
5:      d ← distance(p, current_node)
6:      if d < current_best.distance then
7:          current_best ← current_node.point
8:      end if
9:      dim ← current_node.dim
10:     delta ← | p[dim] − current_node[dim] |
11:     if p[dim] < current_node[dim] then
12:         current_best = NNS(current_node.left, p, current_best)
13:         if delta < current_best.distance then
14:             current_best = NNS(current_node.right, p, current_best)
15:         end if
16:     else
17:         current_best = NNS(current_node.right, p, current_best)
18:         if delta < current_best.distance then
19:             current_best = NNS(current_node.left, p, current_best)
20:         end if
21:     end if
22:     return current_best
23: end function
```

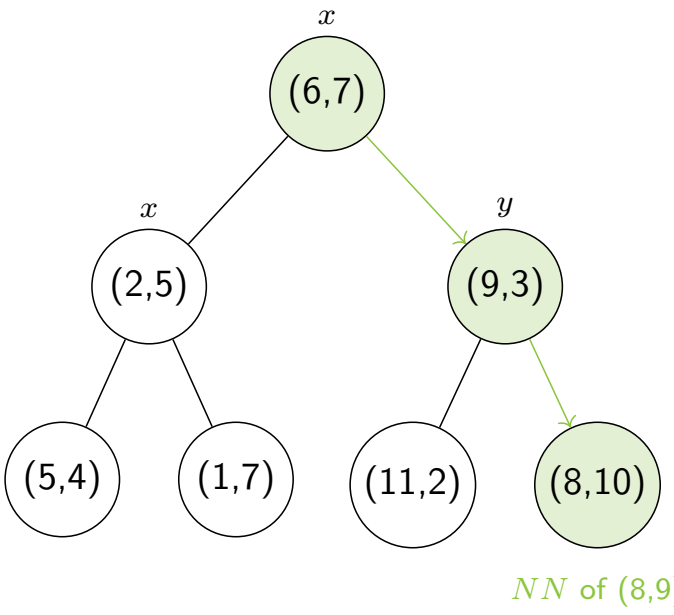
NNS: toy example

Find the Nearest Neighbor of $(8, 9)$.

1) We call $NNS((6,7),(8,9),(6,7))$:
$current\_node := (6,7)$
$p := (8,9)$
$current\_best := (6,7)$
$d := \sqrt{(8-6)^2 + (9-7)^2} = 2\sqrt{2}$
$dim := 1$
$\delta := |8-6| = 2$

2) Since $8 > 6$ we call $NNS((9,3),(8,9),(6,7))$:
$current\_node = (9,3)$
$p = (8,9)$
$current\_best = (6,7)$
$d = \sqrt{(8-9)^2 + (9-3)^2} = \sqrt{37} > 2\sqrt{2}$
$dim = 2$
$\delta = |9-3| = 6$

3) Since $9 > 3$ we call $NNS((8,10),(8,9),(6,7))$:
$current\_node = (8,10)$
$p = (8,9)$
$current\_best = (6,7)$
$d = \sqrt{(8-8)^2 + (9-10)^2} = 1$
Since $1 < 2\sqrt{2}$, we get $current\_best = (8,10)$ which is a leaf. This is our first candidate.



2) $\delta = 6 > 1 = current\_best.dist$, so we do not explore the left subtree from the second level.

1) $\delta = 2 > 1 = current\_best.dist$, so we do not explore the left subtree from the first level.

Verification criterion: a Bayesian decision problem

Bayes Factor

Given a *reference* list and a *test* list of stars, which are our data D, the Bayes Factor is

$$K = \frac{p(D|F)}{p(D|B)}$$

We have to consider the ***prior*** $p(F)/p(B)$, which is typically set to a very small number (10^{-6}), and the ***utility*** table:

		Reality	
		True alignment	False alignment
Decision	Accept	$u(TP) = +1$	$u(FP) = -1999$
	Reject	$u(FN) = -1$	$u(TN) = +1$

The decision to accept or reject the alignment is taken by computing the expected utility $\mathbb{E}[u]$ for each decision.

Verification criterion: a Bayesian decision problem

- $\mathbb{E}[\, u \mid \text{Accept}\,, D\,] = u(TP) \cdot p(TP|D) + u(FP) \cdot p(FP|D)$
  $= u(TP) \cdot p(F|D) + u(FP) \cdot p(B|D)$

- $\mathbb{E}[\, u \mid \text{Reject}\,, D\,] = u(FN) \cdot p(FN|D) + u(TN) \cdot p(TN|D)$
  $= u(FN) \cdot p(F|D) + u(TN) \cdot p(B|D)$

The alignment is accepted if

$$
\begin{aligned}
& \mathbb{E}[\, u \mid \text{Accept}\,, D\,] > \mathbb{E}[\, u \mid \text{Reject}\,, D\,] \\
\iff \quad & u(TP) \cdot p(F|D) + u(FP) \cdot p(B|D) > u(FN) \cdot p(F|D) + u(TN) \cdot p(B|D) \\
\iff \quad & p(F|D) \cdot [u(TP) - u(FN)] > p(B|D) \cdot [u(TN) - u(FP)] \\
\iff \quad & \frac{p(F|D)}{p(B|D)} > \frac{u(TN) - u(FP)}{u(TP) - u(FN)} \\
\iff \quad & \underbrace{\frac{p(D|F)}{p(D|B)}}_{=:K} \cdot \frac{p(F)}{p(B)} > \frac{u(TN) - u(FP)}{u(TP) - u(FN)} \quad \text{by Bayes' Theorem} \\
\iff \quad & K > \frac{p(B)}{p(F)} \cdot \frac{u(TN) - u(FP)}{u(TP) - u(FN)} \\
\iff \quad & K > \frac{1}{10^{-6}} \cdot \frac{1 - (-1999)}{1 - (-1)} = 10^9
\end{aligned}
$$

PRISMA: candidate match hypothesis

$$\begin{aligned} A: \quad (x_1, y_1) &\longleftrightarrow (a_1, z_1) \\ B: \quad (x_2, y_2) &\longleftrightarrow (a_2, z_2) \\ C: \quad (x_3, y_3) &\longleftrightarrow (a_3, z_3) \\ D: \quad (x_4, y_4) &\longleftrightarrow (a_4, z_4) \end{aligned}$$

- Estimate of the linear plate scale V: the mean of

$$V_i := \frac{z_i - S(e^{Dr_i} - 1)}{r_i} \quad \forall\, i \in \{1, 2, 3, 4\}$$

only if $\boxed{V - \frac{1}{60}^\circ/\text{px} < V_i < V + \frac{1}{60}^\circ/\text{px} \;\; \forall\, i = 1, 2, 3, 4}$

- Estimate of a_0: the mean of

$$a_{0,i} = z_i - \arctan\left(\frac{y_i - y_C}{x_i - x_C}\right) \quad \forall\, i \in \{1, 2, 3, 4\}$$

only if $\boxed{a_0 - 5^\circ < a_{0,i} < a_0 + 5^\circ \;\; \forall\, i = 1, 2, 3, 4}$

The estimate of a_0 is then used to convert catalog coordinates in pixels for the computation of $\boldsymbol{K}$.

PRISMA: analysis aimed at finding a suitable threshold for K

number of distractors	notable doubtful solutions (K values)	estimated threshold for K
0	-79.2	10
100	-167.7	10
200	-203.7,-139.7,-51.1,-19.5,68.0	10
300	-170.4,-21.5,-113.7,-10.5	10
400	-25.0,-24.5	10
500	-95.3,-2.9	10

Table: Summary of the results for the estimation of a threshold for K, in the setting with a fixed number of 1200 catalog stars and different fractions of distractors.

Results for images acquired by ITTO09 and ITLO06 cameras

image	conditions	number of detected sources	number of solutions [/100]	mean number of attempts	mean $\boldsymbol{K}$	mean a_0 $[355^\circ]$	mean V $[0.126^\circ/\text{px}]$	mean time $[' \; '']$
ITTO09 2024-11-03 00:01:40	cloudless sky, Moon below the horizon	1238	83	130	35.89	355.19	0.128	$1'$ $16''$

image	conditions	number of detected sources	number of solutions [/100]	mean number of attempts	mean $\boldsymbol{K}$	mean a_0 $[336.3^\circ]$	mean V $[0.166^\circ/\text{px}]$	mean time $[' \; '']$
ITLO06 2023-04-26 20:04:23	cloudless sky, Moon 50° above the horizon at 40% phase	352	15	14	30.11	336.98	0.167	$6''$
ITLO06 2023-04-27 00:09:20	cloudless sky, Moon below the horizon	380	100	11	48.29	336.38	0.167	$9''$

Results for images acquired by ITER10 camera

image	conditions	number of detected sources	number of solutions [/100]	mean number of attempts	mean $\boldsymbol{K}$	mean a_0 [8.5°]	mean V [0.126°/px]	mean time [′ ″]
ITER10 2024-06-26 21:04:22	a few clouds, Moon below the horizon	459	100	20	106.02	8.61	0.127	19″
ITER10 2024-06-26 21:24:22	majority of the sky covered by clouds, Moon below the horizon	137	-	-	-	-	-	-
ITER10 2024-06-26 21:44:22	some clouds covering nearly 50% of the field of view, Moon below the horizon	275	85	17	9.85	8.81	0.128	6″
ITER10 2024-06-26 22:44:22	cloudless sky, Moon below the horizon	1285	100	48	90.22	8.70	0.127	1′ 5″
ITER10 2024-06-27 01:54:23	cloudless sky, Moon 30° above the horizon at 70% phase	951	99	49	114.55	8.70	0.127	1′

PRISMA: possible improvements

- The current algorithm is not suitable for images with **high cloud coverage**.
- Enhancements to the data structure of the KD-tree and to the performance of the NNS algorithm could **reduce computational costs** and **increase robustness**.
- **Estimation of parameters**: our implementation of the algorithm is not *blind*, because we used estimates for some parameters due to the distortion of all-sky cameras. PRISMA can already estimate the distortion parameters (V, S, D), but an automatic procedure to determine the image center C without prior knowledge still has to be developed.

PRISMA: issues in the simplified formula used for the Bayes Factor

- Reference stars and test stars are treated **asymmetrically**.
- The choice of a **uniform background distribution** is not optimal.
- The **number of reference and test stars** affects the behavior and performance of the model:
  - If the reference list contains only a few stars, each correctly matched test star produces a large increase in the Bayes Factor.
  - When the reference list contains many stars, correctly matched test stars lead to a slower increase in the Bayes Factor.
  - If the test list contains stars that are absent from the reference list, as more such stars are included, the Bayes Factor decreases steadily.
- "**Lucky donut**" effect: it is possible for several test stars to be associated to the same reference star, or the other way around.