INSTITUTE OF CONTROL AND COMPUTATION ENGINEERING
FACULTY OF ELECTRONICS AND INFORMATION TECHNOLOGY
WARSAW UNIVERSITY OF TECHNOLOGY

MASTER OF SCIENCE THESIS

STAR-TRACKER PROGRAM FOR SMALL SATELLITES

Szymon MICHALSKI

Supervisor:
prof. dr hab. inż. Ryszard Romaniuk

Warszawa 2019

Abstract

Last years directed space industry towards small satellites. Many countries, which did not have possibility to enter this branch of industry before, now create their own solutions. The goal of this work is to create fully functioning star-tracker software eligible to be used in future satellites as Polish solution of determination of satellite attitude as well as research of possible usage of off-the-shelf embedded computer with GPU - NVIDIA Jetson TX2, which will be used on HyperSat satellite platforms for data processing acquired from cameras observing Earth. The use of Jetson module also for star-tracking would eliminate the need for separate star-tracker device. Work contains also description of individual parts and variants of solutions connected with star-tracker.

Streszczenie

Ostatnie lata ukierunkowały przemysł kosmiczny na małe satelity. Wiele krajów, które wcześniej nie miały możliwości wejścia w tę gałąź przemysłu, teraz tworzy własne rozwiązania. Celem niniejszej pracy dyplomowej jest stworzenie w pełni działającego oprogramowania star-trackera nadającego się do wykorzystania w przyszłych satelitach jako polskie rozwiązanie problemu określania orientacji satelity, a także zbadanie możliwości użycia gotowego komputera wbudowanego z GPU - NVIDIA Jetson TX2, które będą używane na platformach satelitarnych HyperSat do przetwarzania danych z kamer obserwacji Ziemi. Użycie modułu Jetson pozwoliłoby wyelminować zapotrzebowanie na oddzielne urządzenie star-trackera. Praca zawiera też opis poszczególnych części i wariantów rozwiązania problemów związanych ze star-trackerem.

Contents

Acronyms

ADCS Attitude Determination and Control System.

CCD Charge-Coupled Device.

CMOS Complementary Metal–Oxide–Semiconductor.

DCM Direction Cosine Matrix.

ECEF Earth-Centered, Earth-Fixed.

ECI Earth-Centered Inertial.

ESA European Space Agency.

FOV Field Of View.

GEO Geostationary Orbit.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GSO Geosynchronous Orbit.

IMU Inertial Measurement Unit.

LEO Low Earth Orbit.

LIS Lost-In-Space.

MEO Medium Earth Orbit.

QUEST QUaternion ESTimator.

SVD Singular Value Decomposition.

TRIAD Three Axis Attitude Determination.

List of Symbols

ϕ Euler angle, roll.

θ Euler angle, pitch.

ψ Euler angle, yaw.

$\boldsymbol{R}(\cdot)$ Rotation matrix using Euler angles.

$\boldsymbol{q}$ Unit quaternion.

q_0 Scalar part of unit quaternion.

$\boldsymbol{q}_{vec}$ Vector part of unit quaternion.

v General Euler angle.

$\boldsymbol{u}$ Unit vector.

$\boldsymbol{I}$ Identity matrix.

$\boldsymbol{R}_n^b$ Rotation matrix representing a rotation from n to b.

$\boldsymbol{r}$ Known directional unit vector in the inertial (ECI) frame.

$\boldsymbol{b}$ Known directional unit vector in the BODY frame.

α right ascension.

δ declination.

J Polar moment.

A Area.

μ Camera size in pixels.

f Camera focal length.

$\boldsymbol{A}$ Proper orthogonal matrix that minimizes the non-negative loss function.

$L(\boldsymbol{A})$ Loss function.

a_{ROI} the size of the Region of Interest (ROI) in pixels.

a_{MAG} the size of star in pixels (for star magnitude limiting).

$I(xy)$ pixel light intensity at coordinates (x,y).

I_{thresh} limiting pixel light intensity for considering only bright enough pixels.

$I_{thresh_{mag}}$ limiting pixel light intensity for star magnitude limiting.

i_{max} limiting star brightness in star magnitude unit (the higher i, the dimmer the star).

1 Introduction

Stars were used for navigation already ages ago. Together with technological development, spread of sailing and seafaring people started to use more advanced tools helping to more precisely estimate position of ships on sea. In XVIII century new tool was created, named sextant. This device helped to quite precisely calculate angle between the line ship-horizon and ship-star, what let to estimate position.

Nowadays, hundreds of years later, thanks to other technologies like for example Global Positioning System (GPS) stars are not necessary any more for traveling. As it is commonly known, GPS technology is based on satellites, and today those satellites need stars for determining their attitude towards Earth.

With the miniaturization of electronics and batteries new types of smaller satellites were invented: CubeSat and HyperSat. They have now greater sensory and processing power possibilities, previously found only in larger satellites, however they still need cheaper means of accurate attitude determination.

1.1 Motivation

The goal of this work is to make fully operational star-tracker software, that could be used on small satellites. Such program could be used on space missions and could start Polish state-of-the-art technology in growing space technology sector. The main focus is on the star identification part, as it is the most crucial part of the whole star-tracker, without which there would be no attitude determination possible. Another part of the thesis is to research possibility of using off-the-shelf small computer NVIDIA Jetson TX2, as it would greatly lower the price of the hardware design and production, and it could be used together with software created during this work as complete star-tracker.

1.2 Outline of thesis

This thesis consists of several chapters. Here they are shortly summarized:

Chapter 1 serves as introduction to this thesis and describes the motivation and goal of this work. It also describes the background of the topic.

Chapter 2 describes all the important foundations for the fully understanding given work.

Chapter 3 describes how the star-tracker program works and goes through detailed comparison of different approaches.

Chapter 4 describes the created prototype of star-tracker in Python language and performance on Jetson TX2

Chapter 5 contains conclusions about this work and created star-tracker program, and about possible future work.

Chapter 6 summarizes the work and describes possible future development of created star-tracker.

1.3 Related work

There exists a number of works connected to topic of star-tracker. Many works are cited later in this thesis as they describe important parts of algorithms. There are just a few works dealing with whole start-tracker program, not just parts of it, and nearly no works about whole program together with implementation on the real equipment. The related work can be divided into few sections:

- Works describing small satellites in general: Heidt et al. [1] and Swartwout [2].
- Algorithms for calculating star centroids: Liebe [3], Samaan et al. [4], Knutson [5], Azizabadi et al. [6], Lindh [7] and Zhang et al. [8].
- Star identification (which star in image corresponds to which star in on-board catalog) divided by design:

  - Grid Algorithm - Padgett and Kreutz-Delgado [9],
  - SP-Search - Mortari [10],
  - Spherical Triangle - Cole and Crassidus [11],
  - Pyramid - Mortari et al. [12],
  - Planar Triangle - Cole and Crassidis [13],
  - Brightness Independent - Dong et al. [14],
  - Geometric Voting - Kolomenkin et al. [15],
  - Super k-ID and Multi Poles Algorithm created for European Space Agency (ESA) competition[16],
  - use of neural networks:
    * Lindblad et al. [17],
    * Li et al. [18],
    * Miri and Shiri [19],
  - separately discussed k-vector algorithm for faster search:
    * Mortari [20],
    * Mortari and Neta [21],
    * Mortari and Rogers [22].
- Attitude Determination divided by design:
  - Attitude estimation using Image Matching - Delabie [23],
  - QUEST - Shuster [24],
  - Extended Quest - Psiaki [25],
  - Singular Value Decomposition (SVD) - Juang et al. [26],
  - EQUEST - Rinnan [27],
  - Optimal Image Matching - Delabie [23]
  - QUEST improvement - Cheng and Shuster [28],
  - summarized description of attitude estimation algorithms:
    * Markley and Mortari [29],
    * Hall [30],
    * Tappe [31].
- Works on hardware for star-tracker: Felikson et al. [32], Azizabadi et al. [6], Gąska [33].
- Works on full star tracker but without implementing on real device/ simulations only: Rose [34], Huffman et al. [35], Diaz [36] and Kandiyil [37].
- Full star-tracker with device/on real device: Mortari and Romoli [38], Cannata et al. [39], Lizy-Destrez and Mimoun [40] and Jalabert et al. [41].

Many of those works will be mentioned and cited later in this thesis as

they are crucial for this work and describe the ways how star-tracker should be designed. All those main algorithm types, like finding star centroids or star identification, are also described here in full detail in next sections.

It is also interesting to mention one work, which was done at Warsaw University of Technology together with Space Research Centre of Polish Academy of Sciences - Telegwiazda (eng. Telestar, rus. Telezvezda) - star-tracker designed for Gamma satellite, gamma ray telescope, in years 1978-90[42]. That satellite was joint French-Soviet project with star-tracker made in Poland. Due to superior role of Soviet Union in that project comparing to Poland, the name of the star-tracker was changed to Russian (Telezvezda) and as such exists in international documents. Polish version of name exists only in Polish documents. There were two teams of scientists from Warsaw University of Technology working on this star-tracker: one from Faculty of Precision Mechanics under professor Romuald Jóźwicki designing optomechanical part of the system, and one from Faculty of Electronics under Grzegorz Czajkowski designing electronic part with on-board computer[43]. The star-tracker was analyzing image of sky with Field Of View (FOV) 9°. Four brightest stars in FOV were followed and their coordinates were sent to Earth, where they were compared with real star map. The software for checking stars' coordinates with real star map was created at Space Research Centre of Polish Academy of Sciences. Star-tracker could register stars with magnitude equal or lower than 7 and with accuracy below 2 arcmin[44]. Although the satellite failed shortly after its launch in 1990 due to power failure, star-tracker functioned properly and was among the few parts of the satellite which were functioning till the end of satellite's life in 1992. Comparing to Telegwiazda, modern star-trackers do not need to send stars' coordinates to Earth for comparing with sky map, but they can to this by themselves.

1.4 Small satellites

In recent years there has been the development of micro-satellites called CubeSats. CubeSat is a standard created in 1999 at the California Polytechnic State University[1], which is used for low-cost micro-satellites. CubeSats are measured in units. Most are 1U, 2U and 3U. CubeSat 1U has a size of 10 cm on each edge and the maximum weight of 1.333 kg, while the CubeSat 2U is 20x10x10 cm and can weigh up to 2.666 kg. CubeSat can have very low power supply for small units, but for units 6U and higher it can have from 17 Watts[45] to even 126 Watts[46].

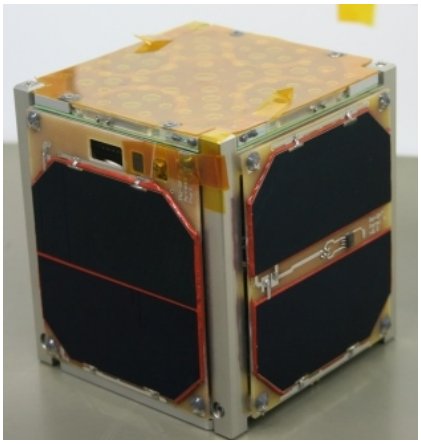
The advantage of the standard is primarily reduction of the price of elevation of such satellites into orbit. Their popularity is evidenced by the fact that the percentage of satellites weighing less than 10 kg has increased to about 60% of all satellites, and only CubeSats make up about half of the small satellites that were launched in last years[2][47]. Till now there were around 1100 CubeSats launched[48].

There exist wide range of CubeSat examples. Mostly they are scientific and students' satellites, for learning, experimental and science purposes. There are also few Polish examples: PW-SAT (1U) and PW-SAT 2 (2U) designed by students of Warsaw University of Technology, and Lem (8U) and Heweliusz (8U), both made at Polish Academy of Sciences. PW-SAT and PW-SAT 2 were used for learning and experimenting with new technologies, like deorbitation sail, solar sensor, etc.[49]. Scientific satellites Lem and Heweliusz were built as part of BRITE programme - joint programme of Austria, Canada and Poland. The goal of this programme is to research mechanisms of energy transportation and angular momentum which happen inside hottest stars[50].

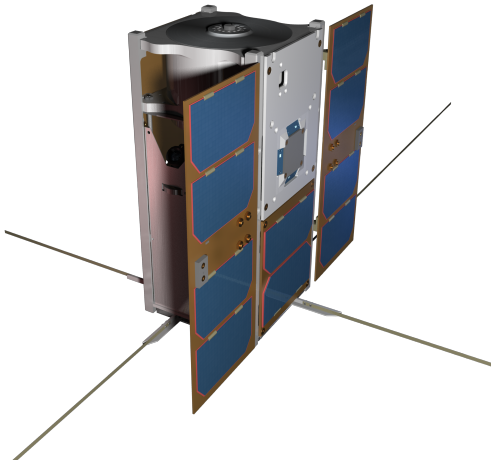
Typical CubeSat consist of many different components. Quite often it has antennas and radios for communication with mission centre on Earth, Electric Power System (subsystem responsible for managing electric power in the satellite), Attitude Determination and Control System (ADCS) (subsystem responsible for analysing data from sensors and taking decision about trajectory, etc.), magnetometer, star-tracker, sun sensors, payload processor with mission instructions, etc. CubeSats are quite small, hence they usually do not have any propulsion system[51]. Figure 2 shows the components of CubeSat on example of NASA's Interplanetary NanoSpacecraft Pathfinder In a Relevant Environment (INSPIRE).

Costs of such CubeSat satellite on example of PW-SAT 2: 120,000-200,000 EUR for putting the satellite on the orbit and around 70,000 EUR for designing and building the satellite itself[55].

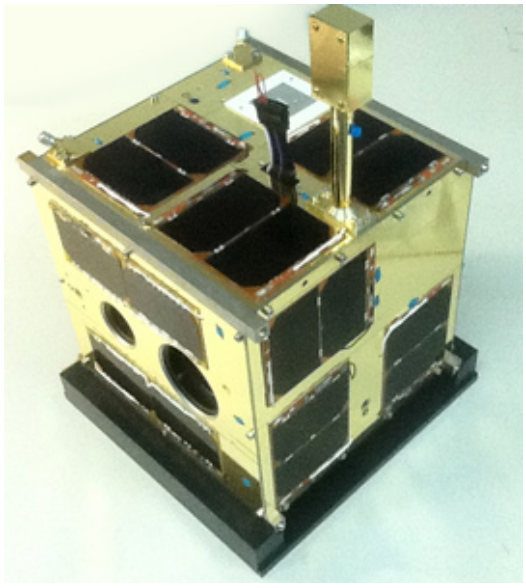
Another platform for slightly larger than CubeSat satellites was designed by Polish company Creotech - HyperSat[56]. It is based on experiences of CubeSat: it is also modular and universal. In the smallest configuration satellite can have dimensions of 30x30x10 cm and weight od 10 kg, while at largest it can have 30x30x60 cm and weight 60 kg. Because of the larger size and weight HyperSat can have more applications than CubeSat, however also lowering the price of satellite design and construction due to the



(a) PW-SAT[52]



(b) PW-SAT 2[49]



(c) BRITE-PL (Lem, Heweliusz)[53]

Figure 1: Example of Polish CubeSats

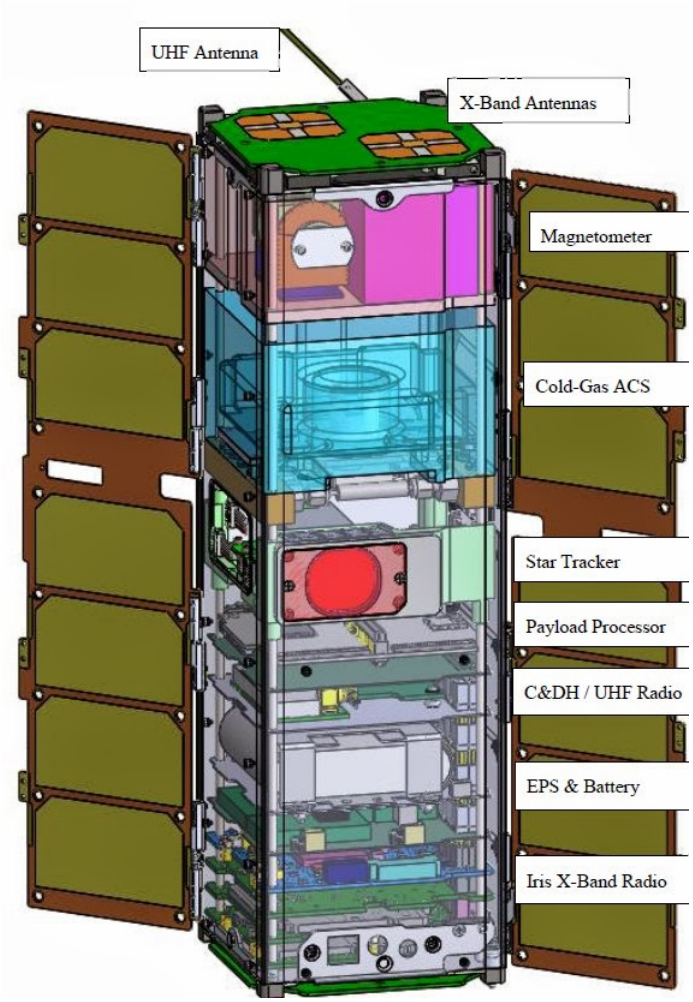
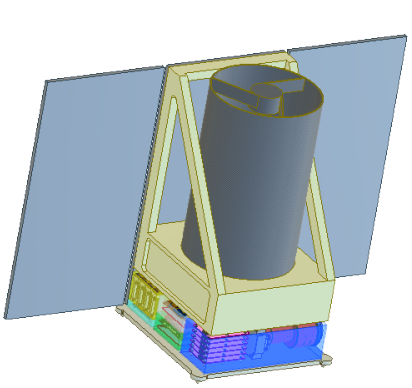
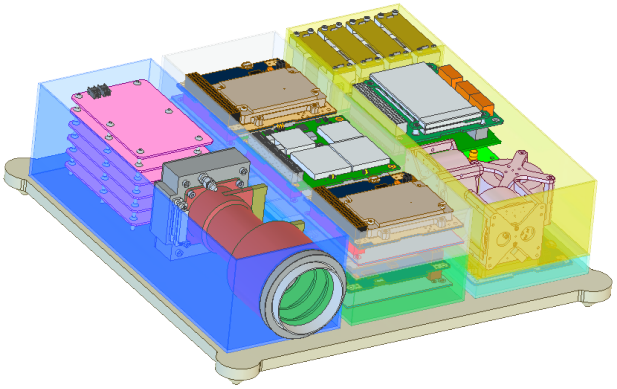


Figure 2: CubeSat build - INSPIRE, Image[54]

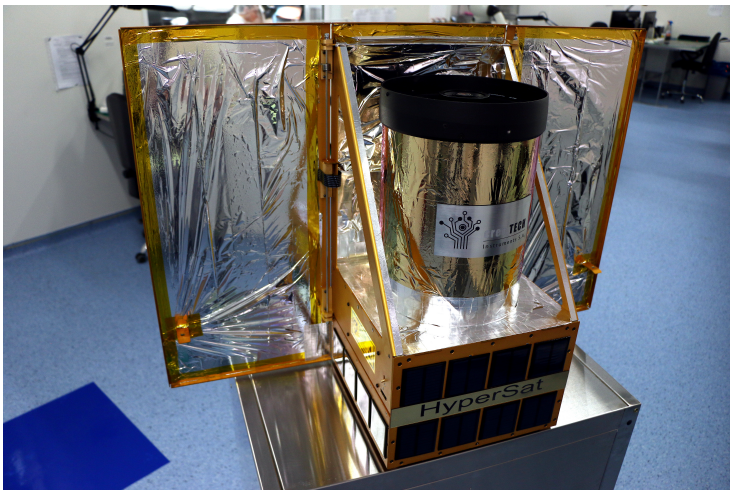
standardization. HyperSats will be equipped with NVIDIA Jetson modules used for data processing acquired from cameras observing Earth[57]. The use of Jetson module also for star-tracking would eliminate the need for separate star-tracker device. The first HyperSats can enter the orbit even as soon as in 2020[58]. The HyperSat plan considers to start serial manufacture where platform would be already prepared and configured in 6 months after ordering it[59]. Figure 3 shows example and prototype of HyperSat.



(a) HyperSat example[56]



(b) HyperSat example[56]



(c) HyperSat prototype[56]

Figure 3: HyperSat

1.5 Means of attitude determination

Attitude of spaceships must be usually stabilized and controlled for various reasons. It is necessary for satellite antenna to be pointing towards Earth for the proper communication, to intelligently control the heat by using the effects of cooling and heating of the shadows and sunlight, as well as to navigate: maneuvers must be performed in the right direction.

Attitude is determined between two coordinate systems (where one is reference system) and defines by what angles the coordinate system connected with the researched object has to be shifted in order to cover the reference system. Devices such as planes and satellites have so called ADCS, which controls attitude of object relative to an inertial reference frame or another entity (the celestial sphere, certain areas, the nearby objects, etc.).

Currently, attitude determination of CubeSats is limited mainly to the sun sensors, magnetometers and measurements of inertia. The following table describes the accuracy of different sensors. Unquestionable winner here is the star-tracker, which, due to its quality and uniqueness of the constellations is ideal for navigation.

1.5.1 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) detects linear acceleration using one or more accelerometers and rotational rate using gyroscopes, sometimes they also include magnetometers. They are commonly used in planes, spacecrafts and many consumer devices like smart-phones. The biggest disadvantage of IMU is accumulated error. Even small error that is accumulated with time brings device to think it is in different location than it actually is. Usually the errors are corrected via use of Kalman filter, which corrects IMU errors using other sources (i.e. GPS).

1.5.2 Sun Sensors

Sun sensors typically consist of few sensors oriented perpendicularly, which can tell the angle of the sun. They can give the information how the satellite is oriented towards the sun.

1.5.3 Star-Tracker

Star-tracker is quite sophisticated device, which consists of camera, processor and memory for database. In the database is catalog of stars, built basing on data from Earth's observatories. Simply put star-tracker, when in orbit, takes picture of space - stars. Then processes the image to recognize stars in it and compares with data in catalog or with previous image. When stars are identified star-tracker estimates the attitude of satellite towards Earth. The main advantage over other means: great accuracy and star-tracker can find its location again after satellite was restarted or lost in space.

1.5.4 Horizon Sensors

Earth horizon sensors use infra-red radiation from the Earth's surface via use of bolometer. The device detects when infra-red signal was first detected and then lost. The time between is used to determine the Earth's width, what later can be used to determine roll angle. In other words device detects the contrast between the cold of deep space and the heat of Earth's atmosphere. Problems of this sensors are the fact that Earth is not perfectly circular and sensors detect infra-red in the atmosphere, which varies depending on the latitude.

1.5.5 Magnetometer

The device actually consists of three magnetometers - one for measuring Earth's magnetic field in each direction. Measurements can be compared with known magnetic field given by International Geomagnetic Reference Field model. The problem with this sensors is that they are susceptible to noise and outside of Earth's orbit it is completely unpredictable.

1.5.6 Global Navigation Satellite System (GNSS)

Nowadays nearly everyone has personal device with Global Navigation Satellite System (GNSS). They are more commonly known by their imple-

mentation by various countries: GPS by USA, GLONASS by Russia, Galileo by European Union, and in close future BeiDou by China. It is possible to use GNSS to determine spacecraft's attitude. This is possible when at least four GNSS satellites are visible by spacecraft, what provides three-dimensional position. Even though it gives quite good accuracy, but it decreases with the altitude. GPS satellites are located on Medium Earth Orbit (MEO), around 20,200 km above sea[60]. It is possible to navigate spacecraft between Low Earth Orbit (LEO) (160 to 2,000 km) and Geostationary Orbit (GEO) (35,786 km)[61]. For missions beyond Earth's orbit GNSS navigation is however not useful. Example of GNSS usage for navigation in space is International Space Station (ISS), which uses GPS.[62] There are however works that claim it could make using GPS systems possible for attitude determination up to Moon attitude in the future[63].

1.5.7 Conclusions

The above means of attitude determination are usually used together with the use of Kalman filter, to compensate errors of one sensor with the other. However none of the means is more accurate than star-trackers. Sun sensors can provide very accurate measurements, but can only operate in sunlight. For low-Earth orbit (LEO), even 30% of the orbit can be done in the darkness. Magnetometers are small and can give accurate measurements if properly calibrated. Their drawback is the limited knowledge of the magnetic field and electromagnetic interference due to highly integrated construction of CubeSats. Microelectromechanical gyroscopes are small enough to fit into a CubeSats. However, they suffer from sudden movements, and could not maintain the correct measurement of the 15-minute period of the eclipse orbit LEO. GPS navigation is accurate, however only on Earth's orbit. To be truly competitive and reliable platform, CubeSats must provide the correct determination of attitude. The best way to meet this goal is via using star-trackers. The typical accuracies of different sensors are available in Table 1.

1.6 NVIDIA Jetson TX2

As mentioned before, part of the thesis is to run star-tracker program on off-the-shelf device: NVIDIA Jetson TX2[65]. Price of this device is around

Sensor	**Accuracy** (less is better)
Inertial Measurement Unit	0.001°/hr to 1°/hr
Sun Sensors	0.005° to 3°
Star Sensors	0.0003° to 0.01° (1 arc sec to 1 arc min)
Horizon Sensors	0.05° (GEO) 0.1° (LEO)
Magnetometer	1.0° (5000km alt) 5.0° (200 km alt)

Table 1: Sensor Accuracy Ranges. Adapted from Hall [30] and Larson and Wertz [64]

479 USD[66] (around 1800 PLN), what in case of successful experiment would significantly lower the price of star-trackers due to usage of already available components instead of designing new hardware.

Jetson TX2 runs Linux and provides greater than 1TFLOPS of FP16 compute performance with 7.5 watts of typical energy usage. Module has dimensions of 50 mm x 87 mm, weights 85 grams, and features the following processing components: dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57, 256-core Pascal GPU and 8GB LPDDR4. High RAM, fast multi-core CPUs and GPU allow quite complex, computing-demanding calculations to be done quickly, while some parts of star-tracker program could be done parallelly. Possibly Jetson TX2 could also be used for machine learning or deep learning star-tracker solutions, as there already exist works exploring that options. Figure 4 shows NVIDIA Jetson TX2 and Developer kit, while Figure 5 shows block diagram.

Low power consumption, low weight, small dimension, low price, already available and powerful computing components make NVIDIA Jetson TX2 good candidate for possible usage on satellites.

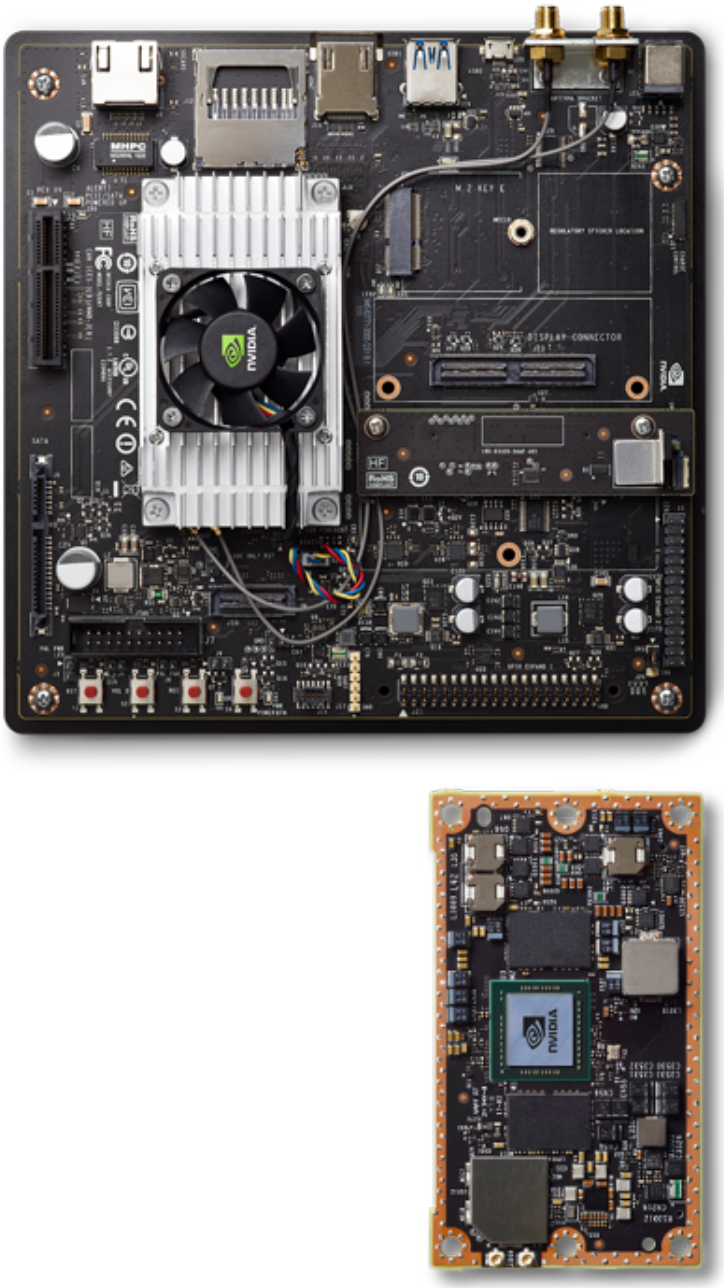


Figure 4: Top: Developer kit, bottom: NVIDIA Jetson TX2 module[65]

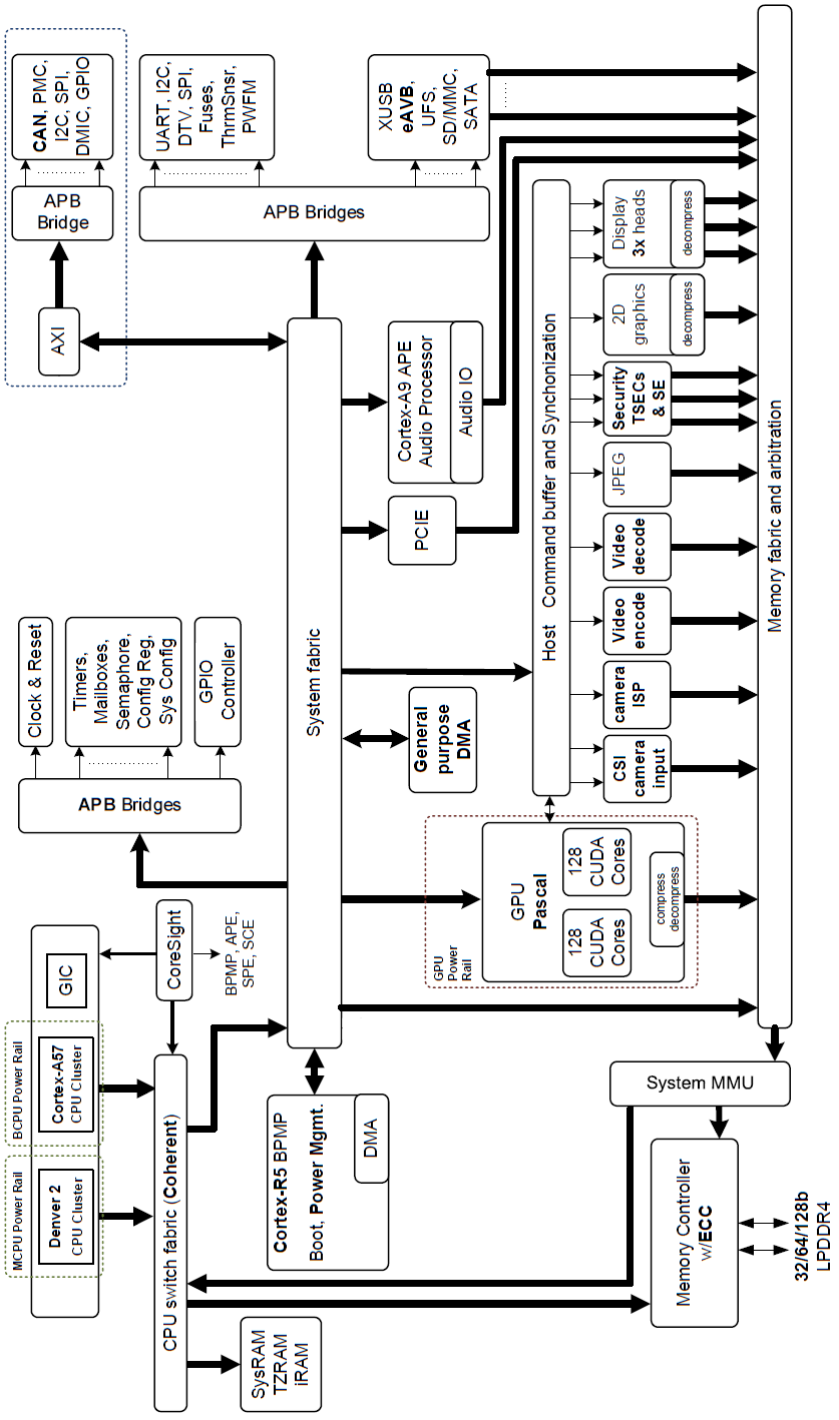


Figure 5: NVIDIA Jetson TX2 block diagram[65]

2 Preliminaries

2.1 Earth's orbits

Most of spacecrafts' missions take place on Earth's orbits. There exist a variety of different classifications of orbits. Here will be presented only a few necessary to understand the topic.

Centric classification:

- geocentric - orbit around Earth
- heliocentric - orbit around the Sun
- Lunar orbit - orbit around the Earth's moon

Orbital period classification:

- Geosynchronous Orbit (GSO) - period of rotation is equal to one sidereal day (23 hours, 56 minutes, and 4 seconds) in the same direction as Earth. This results that satellite stays in the same meridian, but can move in the North-South axis. The altitude of such orbits is 35,786 km above sea level.
- GEO - Special case of GSO. A GEO stays exactly above the equator. For the observer on the surface the satellite stays always at the same point in the sky. Most commercial communication, broadcast and Satellite-Based Augmentation System (SBAS - system complementing GNSS) satellites use GEOs.
- semi-synchronous - orbit has an orbital period of ½ sidereal day. The example here are orbits of GPS satellites.

Altitude classification:

- LEO: altitudes from 160 to 2,000 km. Used among others for ISS, Earth observation and spy satellites, some communication satellites like Iridium phones network, Hubble Space Telescope. Every object on altitude below 160 km will quickly loose it's altitude and it's orbit will decay.
- MEO: altitudes from 2,000 km to 35,786 km. Used for navigation, communication, space environment science. Most commonly used is the altitude approximately 20,00 km, because it has orbital period of ½ sidereal day. It is used for example by GPS. Other GNSS satellites' orbits are 19,000 km for GLONASS and 23,222 km for Galileo.
- GSO and GEO orbits - altitude approximately 35,786 km.
- High Earth orbit: altitude above 35,786 km. Orbital periods are longer than 1 sidereal day.

The mentioned orbits are visible to some extend in the Figure 6. Of course this is just a small variety of possible orbits, however describing more is not necessary for the purpose of this work.

2.2 Coordinate frames

Describing attitude is not that simple outside Earth's surface. Earth rotates and moves around the Sun, therefore it is necessary to understand means needed for correct attitude description. Below are described a few important frames important for understanding this work.

2.2.1 Earth-Centered, Earth-Fixed (ECEF) frame

Earth-Centered, Earth-Fixed (ECEF) frame is visible in Figure 7. Its x axis points towards intersection between Greenwich Meridian and equator, while its z-axis points along the rotation axis of the Earth. The y-axis completes a right handed orthogonal coordinate system. The origin of the

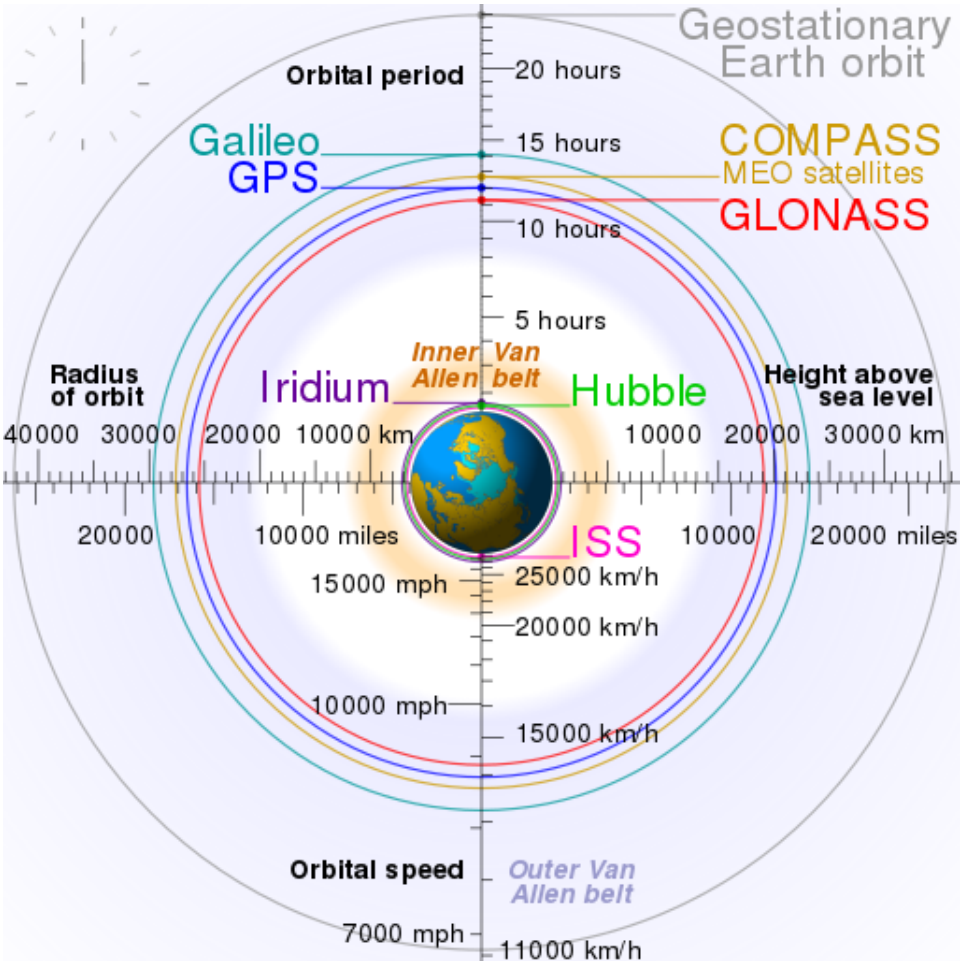


Figure 6: Earth's satellites navigation orbits[67]

frame is at the center of the Earth. This all means that the coordinates move together with Earth's rotation. It is good frame for representing objects on Earth's surface, but not for objects in space.

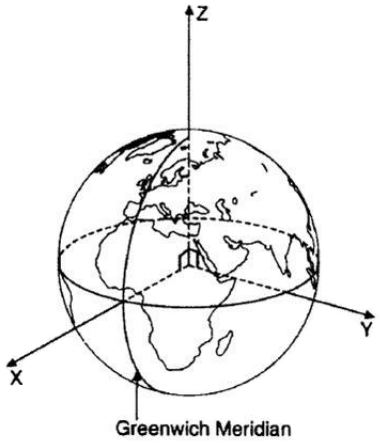


Figure 7: ECEF frame, Image from Larson and Wertz [64]

2.2.2 Earth-centered inertial (ECI) frame

The Earth Centered Inertial frame is similar to ECEF, with the difference that it is inertial - it does not follow Earth's rotation, but stays the same despite the planet's rotation. The origin of the frame is at the center of the Earth. It is much better frame for representing objects in space[64]. This is the frame used in multiple catalogs, also in Hipparcos catalog used in this work, more precisely J2000 ECI frame defined with the Earth's Mean Equator and Equinox at 12:00 Terrestrial Time on 1 January 2000.

2.2.3 North-East Down frame

The North East Down frame is represented in Figure 8. Its z-axis points downwards, perpendicular to the tangent plane of Earth. The x-axis points towards true north and the y-axis points East. The NED frame is an inertial frame - not dependent on Earth's rotation.

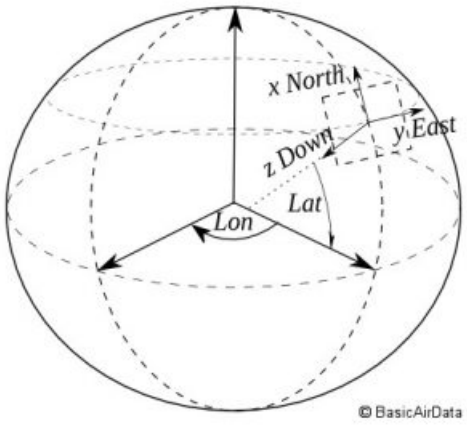


Figure 8: NED frame[68]

2.2.4 BODY frame

In this frame the origin is the center of the spacecraft. The moves and rotates with the spacecraft. The x-axis points forward from spacecraft, the y-axis points to the right side and the z-axis points downwards. The frame is represented in Figure 9. The result of the star identification is in this frame, later converted into Earth-Centered Inertial (ECI) frame.

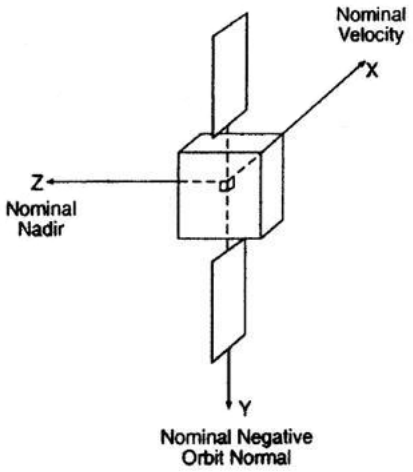


Figure 9: BODY frame, Image from Larson and Wertz [64]

2.3 Attitude representations

Attitude is not only were one object is located relative to other, but also how it is rotated. It is possible to represent attitude in few ways. Here are described two most commonly used for such case: Euler angels and quaternions. Quaternions are used in all presented estimation methods.

2.3.1 Euler angles

Euler angles were described by Leonard Euler in 1776[69]. They are used for representing an orientation of an object. To fully understand the orientation between two frames it is necessary to have three parameters: one angle for the rotation around each axis. Those angles are called roll, pitch, yaw typically written as ϕ, θ and ψ respectively. The Euler angles are often used for the definition of rotation matrices about the x, y and z-axis. The equations 1, 2 and 3 describe those rotation matrices.

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{1}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2}$$

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

2.3.2 Quaternions

In 1843 Sir William Rowan Hamilton described quaterions[70] and in 1845 multiplication of quaternions to describe rotations were published by

Arthur Cayley[71]. Quaternions consist of four elements: three give the coordinates and fourth describing angle of rotation Courant and Hilbert [72]. A quaternion can be described as a four-dimensional vector:

$$\boldsymbol{q} \coloneqq \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{4}$$

where real part can be described using a rotation angle v, as follows:

$$q_0 = \cos(v/2) \tag{5}$$

The imaginary part can be written as a vector:

$$\boldsymbol{q}_{vec} \coloneqq \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = [\boldsymbol{u} \sin(v/2)] \tag{6}$$

where $\boldsymbol{u} = \frac{\boldsymbol{u}}{||\boldsymbol{u}||}$.

In code the so called x, y, z, w representation is used, where the real part is the last element of the quaternion:

$$\boldsymbol{q} \coloneqq \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_0 \end{bmatrix} \tag{7}$$

2.3.3 Advantages of quaternions

Quaternions are easier to use in calculations, have more compact notation and errors easier to handle than in matrix representation. Also normalization of quaternions is easier and cheaper computationally than normalization of matrices. The most important advantage of quaternions over Euler

angles is being safe from gimbal lock. Gimbal lock is the loss of one degree of freedom that occurs when axes of two of three gimbals are driven into a parallel configuration. This could lead to real disaster Shoemake [73].

Quaternions show many advantages, and due to possible spins of satellites, quaternions are the only logical choice to be used for attitude estimation methods. However the visualization of attitude it is mostly done with Euler angles due to its intuitiveness.

2.3.4 Wahba's problem

In 1965 Grace Wahba posed the problem of finding the proper orthogonal matrix $\boldsymbol{A}$ that minimizes the non-negative loss function[74]

$$L(\boldsymbol{A}) = \frac{1}{2}\sum_{i}^{n} |\boldsymbol{b}_i - \boldsymbol{A}\boldsymbol{r}_i|^2 \tag{8}$$

where $\boldsymbol{b}_i$ are unit vectors in body frame, $\boldsymbol{r}_i$ are the corresponding unit vectors in a reference (ECI) frame and n is the number of observations.

If the vectors are error-free and the true attitude matrix $\boldsymbol{A_{true}}$ is assumed to be the same for all the measurements, then $\boldsymbol{b}_i = \boldsymbol{A_{true}}\boldsymbol{r}_i$ for each i and the loss function $L(\boldsymbol{A}) = 0$ for $\boldsymbol{A} = \boldsymbol{A_{true}}$.

3 Star-tracker program

Star-tracker program is designed to determine the attitude of the satellite with the image of the stars made by the satellite camera. Before the start of the mission star catalog is generated, based on the star catalogs obtained from the observatories on Earth and uploaded to the computer on-board the satellite.

After the start, the satellite is released from rocket's tank in space and has no idea where it is. Then the star-tracker on the satellite enters into Lost-In-Space (LIS) and analyses the current image of the stars of the camera, and then searches the corresponding result of stars in the on-board star catalog database. If it finds an entry in the database, the satellite goes into tracking mode. This means each next calculation of the orientation of the satellite attitude is going to be based on a comparison of the current pictures of stars with the preceding ones. If it cannot find the result in the database, this action is repeated from time to time, until a match is found in the database and the program will go into tracking mode.

Of course LIS does not happen only at the beginning of the satellite's flight, but can also result from many causes, e.g. a satellite which is for a long time in the darkness may discharge its battery, and during the next entry into sunlit zone will turn on and look again its attitude. Another case is when satellite becomes lost, although once found the orientation and follow her. This happens, because the successive results are based on preceding ones and even the smallest mistake will grow until the satellite will not be able to correctly calculate their attitude. It may also happen that the satellite will rotate around its axis so fast that the program would not keep up with the processing the image and calculations. In this case, the corresponding other satellite's systems should take an action reducing his spin, but this is not part of this work. It is showed in simple conceptual diagram - Figure 10.

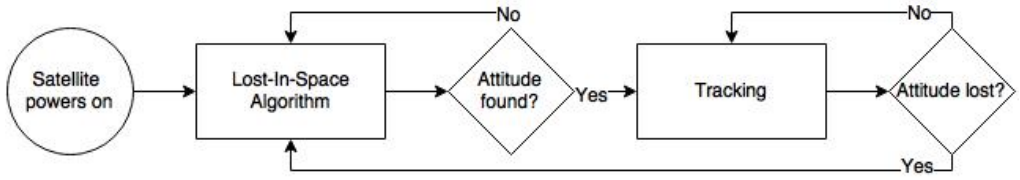


Figure 10: Star-tracker conceptual algorithm diagram

Each part - LIS and tracking - consists of smaller algorithms. Lost-In-

Space at the beginning of reading an image from the camera, thresholds the image. Thresholding is basically throwing away parts of the image which do not exceed the threshold brightness, making just bright enough stars taken into account. Next it calculates centroid (center of mass) of selected stars, identifies these stars using one of the possible methods (here Planar Triangle, described later) and searches the directory on-board technology (k-vector). If the database does not contain corresponding stars, the algorithm returns to the input state and starts analyzing the next image. If it manages to find the corresponding stars, the program determines the attitude of the satellite (by what angle is satellite shifted comparing to the referenced object - Earth or previous image) and enters tracking mode. Tracking mode is in large part similar to the LIS. The algorithm also first analyses the photo, selects and calculates centroids stars, identifies them, but does not search for stars in the directory board. In this algorithm are compared two images, one following after the other, and on this basis the current attitude is calculated.

Generally star-tracker is divided into three main parts[75]:

- Star recognition - recognizing stars on the image and converting the data into list of star vectors by calculating star centroids;
- Star identification - identifying which star vector represents which real star in catalog. This is done by comparing star vectors from the image with data in star catalog, which is generated before space mission;
- Attitude Estimation - estimating the attitude by calculating the displacement between two frames.

3.1 Star recognition

The whole star-tracker program is based on very precise calculations. For this reason, the calculation based on the position of the pixels only can give incorrect results. It is necessary to calculate the position of stars with an accuracy exceeding pixels. This is why the calculation of the star centroids is necessary.

The first step is the determination of stars' position in the plane of the image. If focused star pictures are recorded, the image of each star will fall only on one or two pixels, and most likely the saturate these pixels, resulting

in a pixel level accuracy. Many star-trackers are doing deliberately blurred images, in order to spread the photons on a larger number of pixels, which allows the algorithm for calculating the centroid in sub-pixel accuracy.

After the registration of such image, the centroid of the star is found similarly to the centroid weight points array, with a few differences. Firstly, instead of weight light intensity is used. Secondly, the intensity of light is usually normalized by the pixels around the star in order to filter out glare or noise. The resulting outcome is a series of two-dimensional coordinates on the photo plane with the starting point at the center of the image. This allows the system to the coordinates of the stars can be easily converted into unit vectors in the next step.

The algorithm requires specification of the light intensity threshold (to select the brightest stars) I_{thresh} and the size of the Region of Interest (ROI) a_{ROI} in pixels. These values are adjusted to manipulate the performance of the algorithm. For example, the higher the value of I_{thresh} is more resistant to noise, but can miss some real stars in the picture. Similarly, a large value a_{ROI} means a more exact value of centroid, but the algorithm can see one star there, where are actually two within a short distance of each other. The centroiding part is about trade-off between noise resistance and star recognition, and also between recognizing few close stars as one and recognizing one big star as a few. Please note that a_{ROI} must be a positive odd number for the proper functioning of the algorithm and while higher value of pixel means higher light intensity, for star magnitude the lower value, the higher star brightness.

The idea of how to calculate such centroids is adapted from McBryde and Lightsey [75] and described below:

1. For each pixel at image coordinates (x, y) it is checked if:
   (a) light intensity is high enough: $I(x, y) > I_{thresh}$, hence only bright points of image are taken into account;
   (b) the sum of light intensity of given amount of pixels around the pixel is high enough (Equation 9), hence only low magnitude stars are taken into account. Describing the equation: it is the pixel intensity sum of vertical, horizontal and both diagonals of pixels around the given pixel, where a_{MAG} is limit of pixels for magnitude ROI and $I_{thresh_{mag}}$ is limit threshold of star magnitude pixels

intensity.

$$\begin{aligned} &\sum_{i=x-a_{MAG}}^{x+a_{MAG}} I(x_i, y) + \\ &\sum_{j=y-a_{MAG}}^{y+a_{MAG}} I(x, y_j) + \\ &\sum_{i=x-a_{MAG}}^{x+a_{MAG}} \sum_{j=y-a_{MAG}}^{y+a_{MAG}} I(x_i, y_j) + \\ &\sum_{i=x-a_{MAG}}^{x+a_{MAG}} \sum_{j=y+a_{MAG}}^{y-a_{MAG}} I(x_i, y_j) \\ &> I_{thresh_{mag}} \end{aligned} \tag{9}$$

If both above conditions are met, the algorithm moves forward. Otherwise the pixel is discarded and algorithm moves to another pixel. In case of success, the ROI for given pixel is defined as the square of pixels with side length a_{ROI} and bottom-left corner at (x_{start}, y_{start})

$$x_{start} = x - \frac{a_{ROI} - 1}{2} \tag{10}$$

$$y_{start} = y - \frac{a_{ROI} - 1}{2} \tag{11}$$

$$x_{end} = x_{start} + a_{ROI} \tag{12}$$

$$y_{end} = y_{start} + a_{ROI} \tag{13}$$

2. If $x_{start} < 0$ or $y_{start} < 0$, discard the pixel and return to previous step with the next pixel.

3. Find the average intensity value of the border pixels I_{border}

$$I_{bottom} = \sum_{i=1}^{x_{end}-1} I(i, y_{start}) \tag{14a}$$

$$I_{top} = \sum_{i=2}^{x_{end}} I(i, y_{end}) \tag{14b}$$

$$I_{left} = \sum_{j=1}^{y_{end}-1} I(x_{start}, j) \tag{14c}$$

$$I_{right} = \sum_{j=2}^{y_{end}} I(x_{start}, j) \tag{14d}$$

$$I_{border} = \frac{I_{top} + I_{bottom} + I_{left} + I_{right}}{4(a_{ROI} - 1)} \tag{14e}$$

4. Normalize all the non-boarder pixels, yielding normalized light intensity matrix $\tilde{I}$

$$\tilde{I}(x,y) = I(x,y) - I_{border} \tag{15}$$

5. Calculate centroid locations (x_{CM}, y_{CM}) and brightness B

$$B = \sum_{i=x_{start}+1}^{x_{end}-1} \sum_{j=y_{start}+1}^{y_{end}-1} \tilde{I}(i,j) \tag{16}$$

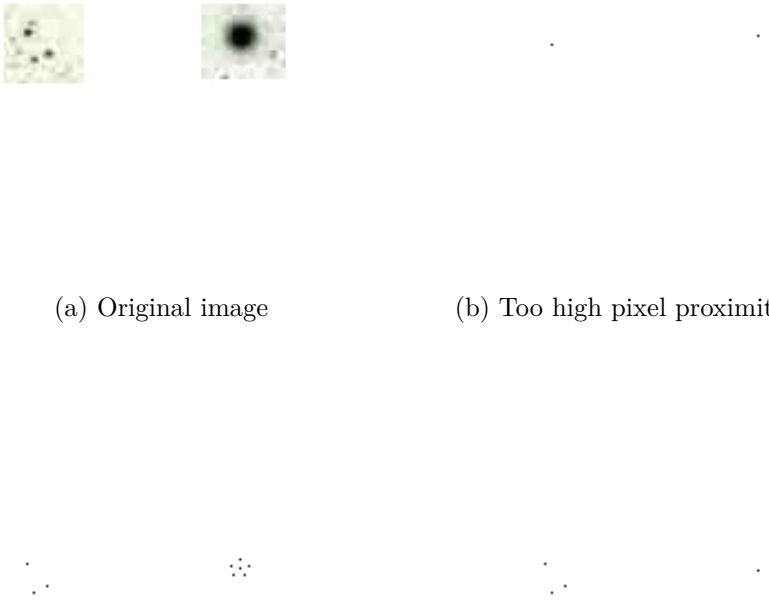
$$x_{CM} = \sum_{i=x_{start}+1}^{x_{end}-1} \sum_{j=y_{start}+1}^{y_{end}-1} \frac{i \times \tilde{I}(i,j)}{B} \tag{17}$$

$$x_{CM} = \sum_{i=x_{start}+1}^{x_{end}-1} \sum_{j=y_{start}+1}^{y_{end}-1} \frac{j \times \tilde{I}(i,j)}{B} \tag{18}$$

6. When the centroid location has been calculated for each pixel above the intensity threshold, it is still necessary to make clustering of them. This is done via averaging together all values that are clustered together. Before clustering it is sure that some stars will have more centroids than one, and clustering helps merge them into one. It is configurable by the number of proximity pixels of the centroids. Of course it is not possible to do it perfectly. The main problem is that sometimes there are few stars in close proximity to each other on the photo and it is possible to count those few stars as one, while doing this correctly for all other stars in the photo. On the other hand, if threshold is lowered too much, it is possible to count few stars correctly, but also possible to count one star as a few. The Figure 11 shows example of such trade-off. The clustering can be accomplished by checking each new centroid against a list of already processed centroids. If the new location is within the given range (for example 5 pixels) it is assumed they represent the same star and therefore they values are averaged. The output of this step is the list of star centroids, where each centroid should represent separate star.

7. Now convert each found centroid location into unit vector $\boldsymbol{u}$ using the camera pixel size μ and camera focal length f

$$\boldsymbol{u} = \frac{\begin{bmatrix} \mu x_{CM} & \mu y_{CM} & f \end{bmatrix}^T}{|| \begin{bmatrix} \mu x_{CM} & \mu y_{CM} & f \end{bmatrix} ||} \tag{19}$$



(a) Original image

(b) Too high pixel proximity threshold

(c) Too low pixel proximity threshold

(d) Ideal result of clustering

Figure 11: Example of centroids' clustering trade-off (in negative colors)

3.2 Star identification

Star identification is the part of star-tracker program, which actually finds out which recognized stars are which. Prior this part it is only known that some stars were read from the image, and now it is the time to identify them by comparing to the on-board star catalog. There is a number of techniques to do that[76]. In this section most of them will be described.

3.2.1 Angle Matching

The Angle Matching is probably the oldest way for identifying stars. It needs at least two stars as an input[77]. The Figure 12 visualizes the way the method works.

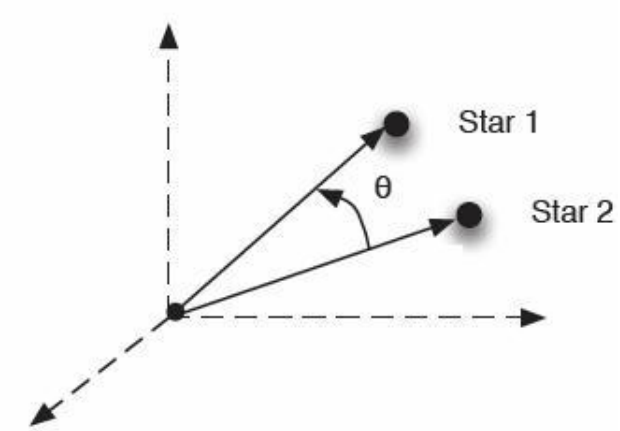


Figure 12: Vector angle method, Image from Gottlieb [77]

The angle between two vectors pointing to the stars is given by:

$$\rho = \cos^{-1}(\boldsymbol{r}_1 \cdot \boldsymbol{r}_2) \tag{20}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are unit vectors pointing to each star. The vectors are given in inertial space. The angle ρ is the same when inertial vectors or body vectors are used. The problem is that the angle measured between stars in the image contain significant measurement error, that cannot be ignored. What is needed by this technique, is the standard deviation of the angle between two stars when each star measurement possesses the error

described. It can be used to provide a bound on the expected errors. Firstly standard coordinate transformation is necessary:

$$\boldsymbol{b}_i = A\boldsymbol{r}_i \tag{21}$$

where $\boldsymbol{r}_i$ is direction the direction of star in the inertial coordinate system, $\boldsymbol{D}$ is the Direction Cosine Matrix (DCM), $\boldsymbol{b}_i$ is direction of star in the star-tracker body coordinate system.

Nearly all the probability of errors is focused on a very small area about the direction of $\boldsymbol{D}\boldsymbol{r}_i$, that the sphere containing that point can be approximated by a tangent plane, characterized by:

$$\tilde{\boldsymbol{b}}_i = \boldsymbol{D}\boldsymbol{r}_i + \boldsymbol{v}_i, \quad \boldsymbol{v}_i^T \boldsymbol{D}\boldsymbol{r}_i = 0 \tag{22}$$

where $\tilde{\boldsymbol{b}}_i$ is the ith measurement and sensor error $\boldsymbol{v}_i$ is approximately Gaussian, which satisfies

$$E\left\{\boldsymbol{v}_i\right\} = 0 \tag{23a}$$

$$E\left\{\boldsymbol{v}_i\boldsymbol{v}_i^T\right\} = \sigma_i^2[\boldsymbol{I} - (A\boldsymbol{r}_i)(A\boldsymbol{r}_i)^T] \tag{23b}$$

where σ_i^2 is the variance and E is expectation. The dot product of two body observations gives

$$\boldsymbol{b}_1^T\boldsymbol{b}_2 = \boldsymbol{r}_1^T A^T A\boldsymbol{r}_2 = \boldsymbol{r}_1^T\boldsymbol{r}_2 \tag{24}$$

what show that the dot product is attitude-invariant measurement. In case of two measurements $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ with noise, where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are uncorrelated:

$$\tilde{\boldsymbol{b}}_1 = A\boldsymbol{r}_1 + \boldsymbol{v}_1$$
$$\tilde{\boldsymbol{b}}_2 = A\boldsymbol{r}_2 + \boldsymbol{v}_2$$

define the effective measurement:

$$z \equiv \tilde{\boldsymbol{b}}_1^T\tilde{\boldsymbol{b}}_2 = \boldsymbol{r}_1^T\boldsymbol{r}_2 + \boldsymbol{r}_1^T A^T\boldsymbol{v}_J + \boldsymbol{r}_2^T A^T\boldsymbol{v}_1 + \boldsymbol{v}_1^T\boldsymbol{v}_2 \tag{26}$$

As $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are uncorrelated, then:

$$E\left\{z\right\} = \boldsymbol{r}_1^T\boldsymbol{r}_2 \tag{27}$$

Define the following variable:

$$p \equiv z - E\left\{z\right\} = \boldsymbol{r}_1^T A^T\boldsymbol{v}_J + \boldsymbol{r}_2^T A^T\boldsymbol{v}_1 + \boldsymbol{v}_1^T\boldsymbol{v}_1 \tag{28}$$

Taking $E\{p^2\}$ gives the variance for the dot product measurement error:

$$\begin{aligned} \sigma_p^2 \equiv E\left\{p^2\right\} = \\ \boldsymbol{r}_1^T A^T R_2 A \boldsymbol{r}_1 + \boldsymbol{r}_2^T A^T R_a A \boldsymbol{r}_2 + Trace(R_1 R_2) = \\ Trace(A\boldsymbol{r}_1 \boldsymbol{r}_1^T R_2) + Trace(A\boldsymbol{r}_2 \boldsymbol{r}_2^T R_1) + Trace(R_1 R_2) \end{aligned} \tag{29}$$

where $E\left\{\boldsymbol{v}_1 \boldsymbol{v}_1^T\right\} \equiv R_1$ and $E\left\{\boldsymbol{v}_2 \boldsymbol{v}_2^T\right\} \equiv R_2$.

The effective measurement noise in the dot product consist of both Gaussian and chi-squared distribution, but since the noise is small in comparison to the unit vector observation, the effective noise can be treated as purely Gaussian. As the attitude is not known, the $\boldsymbol{A}\boldsymbol{r}_1$ and $\boldsymbol{A}\boldsymbol{r}_2$ are replaced by $\tilde{\boldsymbol{b}}_1$ and $\tilde{\boldsymbol{b}}_2$. Errors introduced in this substitution are second-order in nature, hence the variance is completely independent of the attribute matrix $\boldsymbol{A}$.

In case there exist more than one possible solution, so called pivoting is made. This means that if for first angle more than one solution was found, the second angle is selected, that shares one common star with the first angle. When all solutions for second angle are obtained, the solutions not common for those two angles are discarded. If there is still more than one solution, then another pivot is made. Now it compares the second angle with third angle, with a common star. The rest goes analogically. This happens till finally there is only one solution for some two angles, or there are no more angles left in the input, and system enters into LIS mode.

3.2.2 Spherical Triangle Matching

Instead of measuring the inter-star angle, this method creates spherical triangles out of three stars. Due to that more information can be obtained from such triangle than an angle, what enables to identify stars faster and using fewer stars overall than in previous method. In this technique the area and polar moment are used instead of angles. The main disadvantage comparing to angle matching method is that this one requires not two but three stars in order to work properly, however under typical conditions in angle matching it is necessary to use more stars due to measurement error. Also the spherical triangle catalog is about 10 times larger than angle catalog[11].

Given three unit vectors pointing toward three stars, the area of a

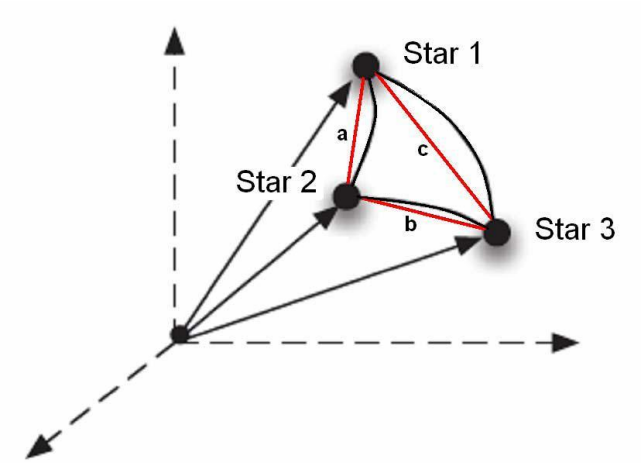


Figure 13: Spherical Triangle Method, Image from Cole and Crassidus [11]

spherical triangle is as follows:

$$A = 4\tan^{-1}\sqrt{\tan\frac{s}{2}\tan\frac{s-a}{2}\tan\frac{s-b}{2}\tan\frac{s-c}{2}} \tag{30}$$

where

$$s = \frac{1}{2}(a+b+c)$$
$$a = \cos^{-1}\left(\frac{\boldsymbol{b}_1 \cdot \boldsymbol{b}_2}{|\boldsymbol{b}_1||\boldsymbol{b}_2|}\right)$$
$$b = \cos^{-1}\left(\frac{\boldsymbol{b}_2 \cdot \boldsymbol{b}_3}{|\boldsymbol{b}_2||\boldsymbol{b}_3|}\right)$$
$$c = \cos^{-1}\left(\frac{\boldsymbol{b}_3 \cdot \boldsymbol{b}_1}{|\boldsymbol{b}_3||\boldsymbol{b}_1|}\right)$$

The equation is given for the body frame, but can be used in the ECI frame too. To get a bound for the measurement error, the standard deviation must be calculated.

The polar moment is good supplement to area, because it is possible for two spherical triangles to have the same area but different polar moment, and all the way around. This helps to quickly reduce the number of possible solutions.

The polar moment can be obtained as follows

$$I_p = \sum \beta^2 dA \tag{32}$$

The polar moment of each spherical triangle is calculated by recursive algorithm, which divides the triangle into smaller triangles till the recursion's depth is met.

The standard deviations for area and polar moment are given by Cole and Crassidus [11].

In this technique pivoting works analogically to the angle matching method. Note that while calculation of the standard deviation of area is quite straightforward, the standard deviation of the second moment calculation is difficult to compute analytically.

3.2.3 Planar Triangle

The third technique, quite similar to previous one is Planar Triangle. Instead of spheres it uses planes, what makes it much less computationally demanding. It also uses the same two properties: triangle area and polar moment. In this case also three stars are necessary[13].

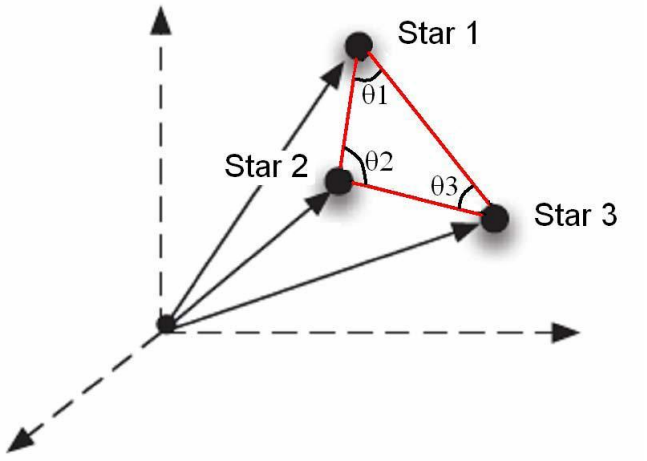


Figure 14: Planar Triangle Method, Image from Cole and Crassidis [13]

Given thee unit vectors pointing toward three stars $\boldsymbol{u_p}$, $\boldsymbol{u_q}$, $\boldsymbol{u_r}$, the area for planar triangle is as follows:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{33}$$

where

$$s = \frac{1}{2}(a + b + c) \tag{34a}$$

$$a = ||\boldsymbol{u_p} - \boldsymbol{u_q}|| \tag{34b}$$

$$b = ||\boldsymbol{u_q} - \boldsymbol{u_r}|| \tag{34c}$$

$$c = ||\boldsymbol{u_p} - \boldsymbol{u_r}|| \tag{34d}$$

The equation is given for the body frame, similarly as for the spherical triangle. It can be also used for ECI frame. Also here the standard deviation has to be calculated in order to obtain a bound of the measurement error.

Similarly as in the previous method, here also polar moment is calculated:

$$J = A\frac{(a^2 + b^2 + c^2)}{36} \tag{35}$$

Since the planar area is a nonlinear function, a linearization must be used to determine its variance. To compute this, the following matrix must be evaluated:

Derivatives

$$H = \begin{bmatrix} \boldsymbol{h}_1^T & \boldsymbol{h}_2^T & \boldsymbol{h}_3^T \end{bmatrix} \tag{36}$$

where

$$\boldsymbol{h}_1^T \equiv \frac{\delta A}{\delta a}\frac{\delta a}{\delta \boldsymbol{b}_1} + \frac{\delta A}{\delta c}\frac{\delta c}{\delta \boldsymbol{b}_1} \tag{37a}$$

$$\boldsymbol{h}_2^T \equiv \frac{\delta A}{\delta a}\frac{\delta a}{\delta \boldsymbol{b}_2} + \frac{\delta A}{\delta b}\frac{\delta b}{\delta \boldsymbol{b}_2} \tag{37b}$$

$$\boldsymbol{h}_3^T \equiv \frac{\delta A}{\delta b}\frac{\delta b}{\delta \boldsymbol{b}_3} + \frac{\delta A}{\delta c}\frac{\delta c}{\delta \boldsymbol{b}_3} \tag{37c}$$

To calculate the partials with respect to a, b and c the following equations are used:

$$\frac{\delta A}{\delta a} = \frac{u_1 - u_2 + u_3 + u_4}{4A} \tag{38a}$$

$$\frac{\delta A}{\delta b} = \frac{u_1 + u_2 - u_3 + u_4}{4A} \tag{38b}$$

$$\frac{\delta A}{\delta c} = \frac{u_1 + u_2 + u_3 - u_4}{4A} \tag{38c}$$

where

$$u_1 = (s-a)(s-b)(s-c) \tag{39a}$$

$$u_2 = s(s-b)(s-c) \tag{39b}$$

$$u_3 = s(s-a)(s-c) \tag{39c}$$

$$u_4 = s(s-a)(s-b) \tag{39d}$$

The partials with respect to $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$ are calculated with formulas:

$$\frac{\delta a}{\delta \boldsymbol{b}_1} = (\boldsymbol{b}_1 - \boldsymbol{b}_2)^T/a, \quad \frac{\delta a}{\delta \boldsymbol{b}_2} = -\frac{\delta a}{\delta \boldsymbol{b}_1} \tag{40a}$$

$$\frac{\delta b}{\delta \boldsymbol{b}_2} = (\boldsymbol{b}_2 - \boldsymbol{b}_3)^T/b, \quad \frac{\delta b}{\delta \boldsymbol{b}_3} = -\frac{\delta b}{\delta \boldsymbol{b}_2} \tag{40b}$$

$$\frac{\delta c}{\delta \boldsymbol{b}_1} = (\boldsymbol{b}_1 - \boldsymbol{b}_3)^T/c, \quad \frac{\delta c}{\delta \boldsymbol{b}_3} = -\frac{\delta c}{\delta \boldsymbol{b}_1} \tag{40c}$$

The variance of the area is as follows:

$$\sigma_A^2 = HRH^T \tag{41}$$

where

$$R \equiv \begin{bmatrix} R_1 & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & R_2 & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & R_3 \end{bmatrix} \tag{42}$$

As with the area, it is necessary to obtain the variance of the polar moment. The following partial matrix has to be evaluated first:

$$\bar{H} = \begin{bmatrix} \bar{\boldsymbol{h}}_1^T & \bar{\boldsymbol{h}}_2^T & \bar{\boldsymbol{h}}_3^T \end{bmatrix} \tag{43}$$

where

$$\bar{\boldsymbol{h}}_1^T \equiv \frac{\delta J}{\delta a}\frac{\delta a}{\delta \boldsymbol{b}_1} + \frac{\delta J}{\delta c}\frac{\delta c}{\delta \boldsymbol{b}_1} + \frac{\delta J}{\delta A}\boldsymbol{h}_1^T \tag{44a}$$

$$\bar{\boldsymbol{h}}_2^T \equiv \frac{\delta J}{\delta a}\frac{\delta a}{\delta \boldsymbol{b}_2} + \frac{\delta J}{\delta b}\frac{\delta b}{\delta \boldsymbol{b}_2} + \frac{\delta J}{\delta A}\boldsymbol{h}_2^T \tag{44b}$$

$$\bar{\boldsymbol{h}}_3^T \equiv \frac{\delta J}{\delta b}\frac{\delta b}{\delta \boldsymbol{b}_3} + \frac{\delta J}{\delta c}\frac{\delta c}{\delta \boldsymbol{b}_3} + \frac{\delta J}{\delta A}\boldsymbol{h}_3^T \tag{44c}$$

with

$$\frac{\delta J}{\delta a} = \frac{-b^3 * A * 2 * a * b^2 * A}{18 * b^2} \tag{45a}$$

$$\frac{\delta J}{\delta b} = \frac{\frac{4 * b^3 * A - 3 * a * b^2 * A + 2 * a^2 * b * A}{18 * b^2} - }{\frac{b^* 4 * A - a * b^3 * A + a^2 * b^2 * A + 4 * A^3}{9 * b^3}} \tag{45b}$$

$$\frac{\delta J}{\delta c} = \frac{b^4 - a * b^3 + a^2 * b^2 + 12 * A^2}{18 * b^2} \tag{45c}$$

$$\frac{\delta J}{\delta A} = (a^2 + b^2 + c^2)/36 \tag{45d}$$

Other quantities are given from the area variance calculations. The variance of the polar moment is calculated with the following formula:

$$\sigma_J^2 = \bar{H} R \bar{H}^T \tag{46}$$

Pivoting works analogically as in the previous methods.

This method gives similar results as spherical triangle method, but planar triangle method does not have to perform recursive calculations of polar moment like the previous method, hence it has lower computational demand for star-identification[78].

3.2.4 Pyramid

Pyramid method is described by Mortari et al. [12]. This algorithm is designed to be used efficiently with 4-star input, however can also function for 3-star input if no more are available. It uses set of star pairs in elementary measured star polygons (3 for a triangle, six for a 4-star pyramid). The vertices between adjacent measured star pairs share a common cataloged star.

The Figure 15 shows how the algorithm works. If there are less than 3 stars on input, the algorithm returns error, what actually brings back to start of the algorithm with new input data. In case there are three stars only, algorithm gets rid of redundant solutions, and if the solution is unique, the stars are identified and solution is outputted. In case of not finding unique solution, algorithm returns an error, and starts again with new data. If there are more than three stars, algorithm gets three stars for a triangle, and then one reference star. Later it checks pairs of reference star and triangle's vertices. If unique solution is found, it identifies the remaining stars and outputs

all the identified stars. If no unique solution has been found, algorithm tries to get new triangle and repeat the steps. There are two possible exits of this loop: the already mentioned finding unique solution, or running out of new triangles. In the last case algorithm removes redundant solutions, then check again if it is unique solution. Finally either unique solution is found and algorithm identifies all other stars from the input and outputs them, or returns an error and stars again with the new input data.

According to authors the algorithm can solve LIS problem and is highly robust. The key to algorithm's efficiency lies in the usage of k-vector method and the usage of expected random frequencies connected to matching inter-star angles from measured star polyhedron. The algorithm has been tested on Draper's "Inertial Stellar Compass" star tracker[79] and on MIT's satellites HETE and HETE-2[80] successfully. This algorithm is currently under exclusive contract to StarVision Technologies[81][82].

3.2.5 Voting

Algorithm was presented by Kolomenkin et al. [15]. Pair of stars in the catalog votes for a pair of stars in the image if angular distance between the stars of both pairs is similar. Since angular distance is a symmetric relationship, each catalog star in the pair vote for each in the image pair. Then the image star is identified as the catalog star which gave the most votes.

Authors' testing has shown that the algorithm is as accurate, as fast and more robust than other methods. The authors tested the algorithm in simulation only, and it does not appear to be tested in space yet. Furthermore, all simulations and testing were done on assumption of slow rotation, hence it is not known how it would function in the real environment. The more details about this algorithm can be found in Kolomenkin et al. [15].

3.2.6 Grid

The Grid Algorithm was designed by Padgett and Kreutz-Delgado [9]. The main ideas of the algorithm are to generate a specified pattern database and to construct a grid pattern from the measured star field in the Charge-

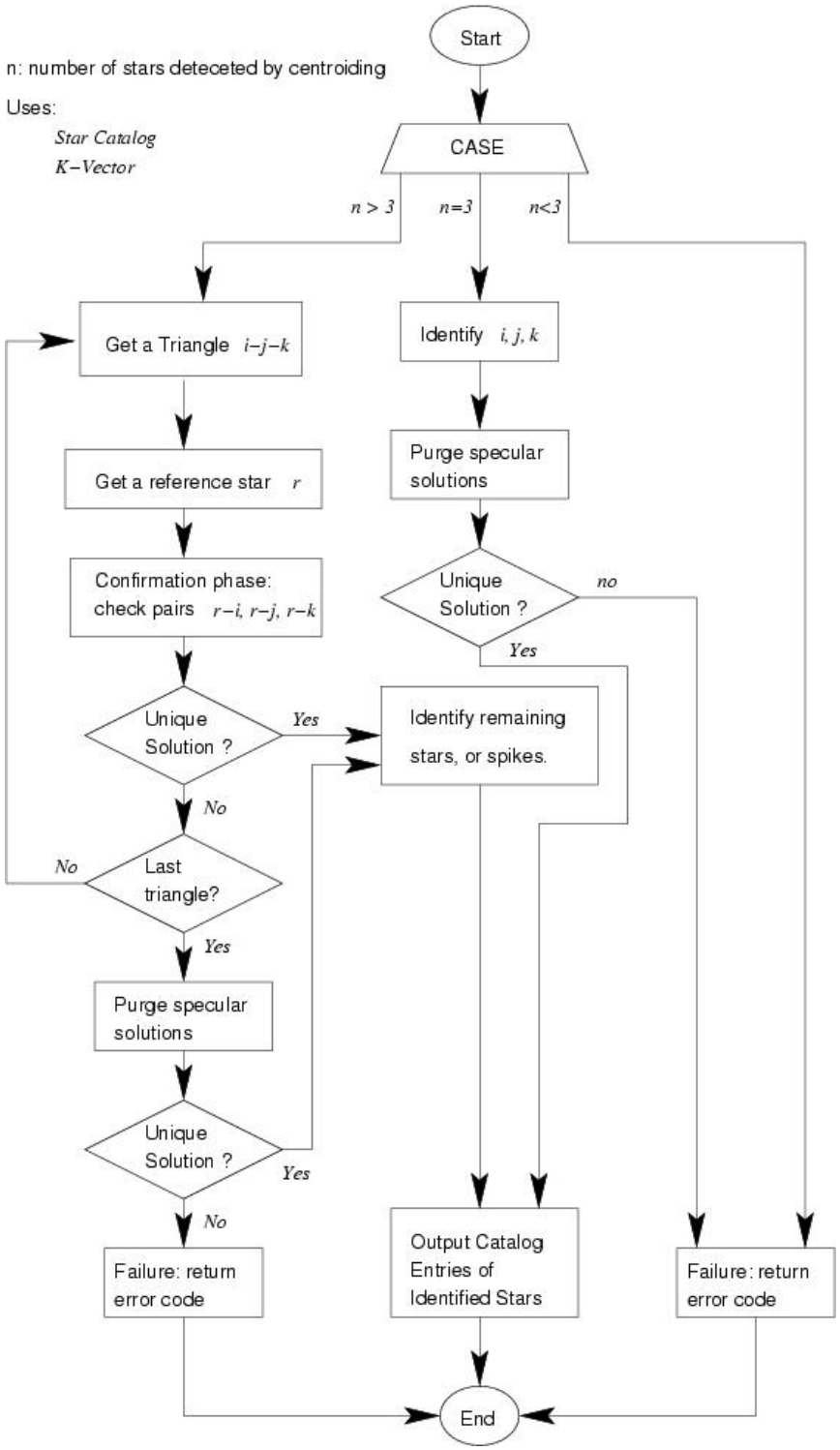


Figure 15: Pyramid Method Flowchart, Image from Mortari et al. [12]

Coupled Device (CCD) plane. The specified pattern is designed by using grids. Then the measured grid pattern is identified by best matching pattern in the database.According to authors the algorithm is robust and comparable with other methods in terms of accuracy and performance.

This algorithm will not be described in more detail, since it is designed for CCD camera, and this thesis focuses on star-tracker for OBC mentioned before, which uses Complementary Metal–Oxide–Semiconductor (CMOS) camera. More information can be found in Padgett and Kreutz-Delgado [9].

3.2.7 Other techniques

Other techniques involve:

- Spherical-Polygon (SP) Search by Mortari [10]:

  The idea of this method is that any star direction can be expressed as a linear combination of two non-parallel star directions (star pair basis) together with their vector cross-product. The star identification problem is transformed into the simpler problem of finding which stars, within a large star catalog, are admissible with a given direction. This is solved by approximating a cone with a spherical polygon and search for all of the three components of the given direction.

- Star Neighbourhood Approach by Samaan et al. [83]:

  This method does not solve LIS problem, but can be used only in star-tracker's tracking mode. The algorithm makes use of the knowledge of the previous frame data to initiate the current frame star.

- Brightness Independent 4-Star Matching Algorithm by Dong et al. [14]:

  This technique needs at least 4 stars as an input, than calculates angular separations between each pair, and tries to find corresponding star pairs with similar angular separations in the database. Identified stars are not used again, and the remaining stars are used with additional stars to make a new 4-star group. Then this step is repeated. In case no star was identified, at least one star will be removed to build a new 4-star group. If there would be no more than 4 stars left, then already

identified stars have to be used. No information about real-life usage has been found.

- Use of neural networks and machine learning algorithms:
  - 'Star Identification using Neural Networks' by Lindblad et al. [17] describes approach to star identification in two different ways: Radial Basis Funcion (RBF) Neural Network and k Nearest Neighbours,
  - Parallel Backpropagation (BP) multi-subnets are designed by Li et al. [18],
  - Method based on Delunay Triangulation algorithm and neural networks is designed by Miri and Shiri [19].

3.2.8 Conclusions

From the described techniques the Planar Triangle method was chosen. Comparing to Angle Method it gives better results and comparing to Spherical Triangle it less computationally demanding[31][78]. It requires more memory for the database, but still in reasonable amount. Pyramid technique cannot be used as it is patented. Algorithms based on neural networks need significantly more computational power and memory. What is more, no information about the usage of such algorithms on real spacecraft was found. As for other algorithms, Star Neighbourhood does not support LIS mode, for Brightness Independent, SP-Search, Voting and Grid methods no information about real-life use was found. Moreover Planar Triangle was tested in at least two other works (but only against Angle and Spherical Triangle Methods), and was chosen as giving the best overall results, and in one work it was chosen for implementation along one other algorithm[35] due to its speed.

3.3 Star-catalog and database search method

3.3.1 Star Catalog Generation

For each of previously mentioned algorithms it is necessary to match patters, angles, areas, polar moments, etc. with already known data. Such

data saved in the database prior to launching the spacecraft. The database is usually made out of star database collected by Earth's observatories. Also for each different star identification method and different FOV the catalog must be recreated.

For nearly every start pattern matching it is necessary to convert firstly the star positions to unit vectors. Most star catalogs use right ascension-declination coordinates while for star identification unit vectors must be used[75]. The following formula transforms the coordinates to unit vectors:

$$\boldsymbol{u} = \begin{bmatrix} \cos\alpha\cos\delta \\ \sin\alpha\cos\delta \\ \sin\delta \end{bmatrix} \tag{47}$$

where $\boldsymbol{u}$ is unit vector, α is right ascension and δ is declination.

Since Planar Triangle method was chosen, the rest of catalog generation will be described for this method. For each three catalog stars that are above certain light intensity threshold (in star magnitude)

$$i_a \geq i_{max} \tag{48a}$$

$$i_b \geq i_{max} \tag{48b}$$

$$i_c \geq i_{max} \tag{48c}$$

and in camera's FOV

$$\boldsymbol{u_a^T u_b} \geq \cos\gamma_{FOV} \tag{49a}$$

$$\boldsymbol{u_b^T u_c} \geq \cos\gamma_{FOV} \tag{49b}$$

$$\boldsymbol{u_c^T u_a} \geq \cos\gamma_{FOV} \tag{49c}$$

area A and polar moment J should be calculated, exactly the same as in Planar Triangle method, however without variances. If all above conditions are met, star identification numbers, area and polar moment are saved in the database for later.

$\boldsymbol{u_x^T u_y}$ is equal to cosine of angle between the two unit vectors, therefore checking if condition in Equation 49 is satisfied is the same as stating the

angular distance is less than FOV angle, what ensures that all three stars can be seen by camera at the same time.

3.3.2 Search Less Algorithm and k-vector

Search Less Algorithm (SLA) by Mortari [20] is special algorithm used to speed up looking up the corresponding stars in the on-board computer. Its heart is k-vector technique. Comparing to binary search technique, k-vector demonstrates high speed gain rate, from 10 to more than 50 times[21].

The k-vector database is built before spacecraft's start. The k-vector table is a structural database of all cataloged star pairs that could possibly fit into the camera FOV over the whole sky. The star pairs are ordered with increasing inter-star angle.

The k-vector technique works in the following way: $\boldsymbol{y}$ is the data vector of n elements ($n \gg 1$) and $\boldsymbol{s}$ is the same vector but sorted in ascending mode: $\boldsymbol{s}(i) \leq \boldsymbol{s}(i+1)$, and $\boldsymbol{s}(i) = \boldsymbol{y}(\boldsymbol{J}(i))$, where $\boldsymbol{J}$ is the integer vector of the sorting with length n[22].

In the Figure 16 it is shown how such k-vector construction looks like. In this example the database has 10 elements. X-axis describes k-vector elements $\boldsymbol{k}(i)$ and y-axis describes the sorted values $\boldsymbol{s}(i)$ (inter-star angle in this case). The value of $\boldsymbol{k}(i)$ represents the number of elements of the database vector $\boldsymbol{y}$ below the value $z(i)$. The resulting k-vector is $\boldsymbol{k} = \{0, 2, 2, 3, 3, 5, 6, 8, 9, 10\}^T$.

The following equations describe how the k-vector database is created. The line connecting two extreme points, $[1, y_{min}]$ and $[n, y_{max}]$, has to be a little stepper and connect points $[1, y_{min} - \delta\epsilon]$ and $[n, y_{max} + \delta\epsilon]$. This assures $\boldsymbol{k}(1) = 0$ and $\boldsymbol{k}(n) = n$ and simplifies the code by avoiding many index checks.

$$z(x) = mx + q \begin{cases} m = \frac{y_{max} - y_{min} + \delta\epsilon}{n-1} \\ q = y_{min} - m - \delta\epsilon \end{cases} \tag{50}$$

$$\delta\epsilon = (n-1)\epsilon \tag{51}$$

ϵ is relative machine precision for double precision arithmetic

$$\epsilon \approx 22.2 \times 10^{-16} \tag{52}$$

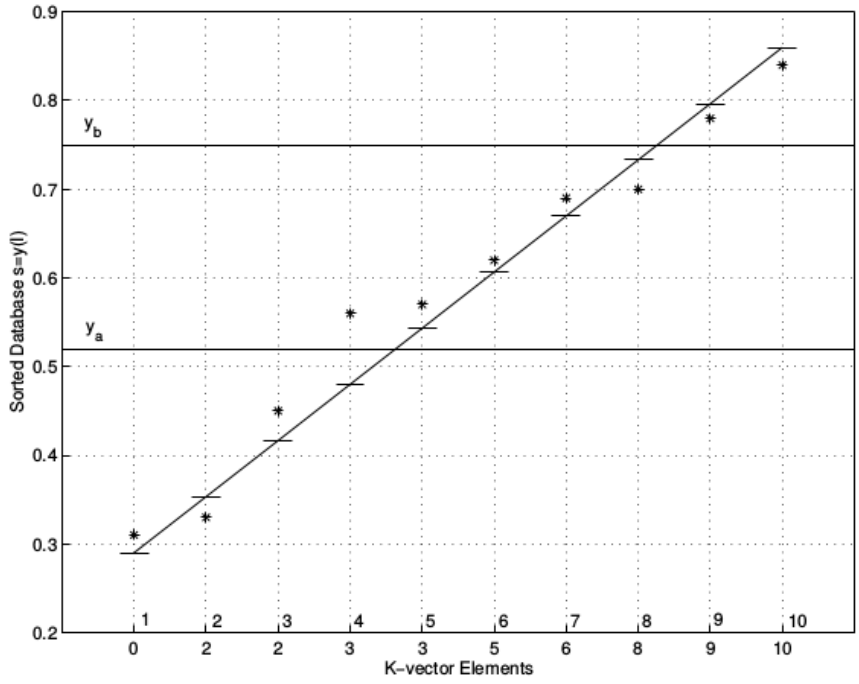


Figure 16: K-vector technique example, Image from [21]

Each other element can be found using the following equation:

$$\boldsymbol{k}(i) = j \quad where \quad s(j) \leq z(i) < s(j+1) \tag{53}$$

or

$$\boldsymbol{k}(i) = j \quad \text{where j is the greatest index such} \quad s(j) \leq \boldsymbol{y}(\boldsymbol{J}(i)) \quad \text{is satisfied.} \tag{54}$$

At this point k-vector has been built, and further use is quite simple. The equation to find indices identifying in $\boldsymbol{s}$ vector all possible data within the range $[y_a, y_b]$ is following:

$$j_b = \left\lfloor \frac{y_a - q}{m} \right\rfloor \quad and \quad j_t = \left\lceil \frac{y_b - q}{m} \right\rceil \tag{55}$$

In the example of Figure 16 it is $j_b = 4$ and $j_t = 9$. After indices are evaluated it is possible to calculate

$$k_{start} = \boldsymbol{k}(j_b) + 1 \quad and \quad k_{end} = \boldsymbol{k}(j_t) \tag{56}$$

Finally, having found k_{start} and k_{end} it is immediately known, that searched elements are $\boldsymbol{y}(i) \in [y_a, y_b]$, which are all the $\boldsymbol{y}(\boldsymbol{J}(i))$ where k ranges from k_{start} and k_{end}.

Note that in example the searched elements should be $k_{start} = 4$ and $k_{end} = 8$ while the presented technique returns $k_{start} = 4$ and $k_{end} = 9$. This happens due to fact that k_{start} and k_{end} have 50% possibility to not belong to the $[y_a, z(j_a + 1)]$ and $[z(j_b), y_b)]$ ranges respectively.

3.4 Attitude Determination

The last main part of the star-tracker program is attitude determination. After star pattern is found, there come two lists of unit vectors. One list is in the body frame while the other is in the inertial frame. In order to find rotation between them and later CubeSat's attitude, an attitude determination method must be applied. A number of techniques were designed to solve this problem. Typically the output can come in the form of a quaternion, Euler angles or a rotation matrix (DCM).

The attitude determination methods should be fast and robust. This section describes few most common algorithms designed to solve this problem.

3.4.1 Three Axis Attitude Determination (TRIAD)

Three Axis Attitude Determination (TRIAD) is based on constructing two triads of orthonormal unit vectors using the vector information from input. Both triads are components of the same reference frame expressed in the body and inertial frames[30].

$\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are two vectors in body frame and $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are in reference frame. First step of the algorithm is following:

$$\boldsymbol{bb}_1 = \boldsymbol{b}_1/|\boldsymbol{b}_1| \tag{57a}$$

$$\boldsymbol{P}_2 = (\boldsymbol{P}_1 x \boldsymbol{b}_2)/|\boldsymbol{P}_1 x \boldsymbol{b}_2| \tag{57b}$$

$$\boldsymbol{P}_3 = \boldsymbol{P}_1 x \boldsymbol{P}_2 \tag{57c}$$

and the following matrices in reference frame:

$$\boldsymbol{T}_1 = \boldsymbol{r}_1 / |\boldsymbol{r}_1| \tag{58a}$$
$$\boldsymbol{T}_2 = (\boldsymbol{T}_1 x \boldsymbol{r}_2) / |\boldsymbol{T}_1 x \boldsymbol{r}_2| \tag{58b}$$
$$\boldsymbol{T}_3 = \boldsymbol{T}_1 x \boldsymbol{T}_2 \tag{58c}$$

where $\boldsymbol{P}_i, i \in \{1, 2, 3\}$ are matrices in body frame and $\boldsymbol{T}_i, i \in \{1, 2, 3\}$ are matrices in reference frame.

Finally the attitude matrix $\boldsymbol{A}$ is computed:

$$\boldsymbol{A} = \boldsymbol{P}_1 \cdot \boldsymbol{T}_1^T + \boldsymbol{P}_2 \cdot \boldsymbol{T}_2^T + \boldsymbol{P}_3 \cdot \boldsymbol{T}_3^T \tag{59}$$

The disadvantage of this algorithm is discarding some of the information, therefore losing the accuracy. The main advantage is that TRIAD is quite fast.

3.4.2 q-method

In 1968 Paul Davenport applied Wahba's problem to spacecraft attitude determination, via method called q-method[84].

Let loss function be represented in quaternion form:

$$L(A) = \frac{1}{2} \sum |\boldsymbol{b}_i - D(\boldsymbol{q}) \boldsymbol{r}_i|^2 \tag{60}$$

where quaternion q is searched, that minimizes cost function L. Here, q is the eigenvector corresponding to the largest eigenvalue of K matrix, which is generated as follows:

$$\sigma = \sum \boldsymbol{b}_i^T \boldsymbol{r}_i \tag{61a}$$

$$B = \sum \boldsymbol{b}_i \boldsymbol{r}_i^T \tag{61b}$$

$$S = B + B^T \tag{61c}$$

$$\boldsymbol{Z} = \sum \boldsymbol{b}_i x \boldsymbol{r}_i \tag{61d}$$

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{S} - \sigma \boldsymbol{I}_3 & \boldsymbol{Z} \\ \boldsymbol{Z}^T & \sigma \end{bmatrix} \tag{62}$$

$\boldsymbol{I}_3$ is 3x3 identity matrix.

3.4.3 QUaternion ESTimator (QUEST)

QUaternion ESTimator (QUEST) method was proposed by Shuster [85]. The method builds on the q-method. The main goal for this method was to develop the quaternion estimator was to obtain finding the optimal eigenvalue for the loss function. The eigenvalue problem is reformulated to problem of solving a characteristic equation. The minimization of loss function is following:

$$\begin{aligned} J(\boldsymbol{q}) = \frac{1}{2} \sum_{j=1}^{n} \frac{1}{\sigma_j^2} (\boldsymbol{b}_j - \boldsymbol{R}_b^i(\boldsymbol{q}) \boldsymbol{r}_j)^T (\boldsymbol{b}_j - \boldsymbol{R}_b^i(\boldsymbol{q}) \boldsymbol{r}_j) = \\ \frac{1}{2} \sum_{j=1}^{n} \frac{1}{\sigma_j^2} (\boldsymbol{b}_j^T \boldsymbol{b}_j - 2 \boldsymbol{b}_j^T \boldsymbol{R}_b^i(\boldsymbol{q}) \boldsymbol{r}_j + \boldsymbol{r}_j^T \boldsymbol{r}_j) \end{aligned} \tag{63}$$

where $\boldsymbol{r}_j$ is unit vector given in the ECI frame, while $\boldsymbol{b}_j$ is unit vector given in the fixed BODY frame. The parameter σ_j is the standard deviation of the measurement error. Due to fact that both unit vectors have to be of equal length, the cost function can be reduced to:

$$J(\boldsymbol{q}) = \sum_{j=1}^{n} \frac{1}{\sigma_j^2} (1 - \boldsymbol{b}_j^T \boldsymbol{R}_b^i(\boldsymbol{q}) \boldsymbol{r}_j) \tag{64}$$

The main advantage is that QUEST method can find exact solution after one step. The method is designed to preserve quaternion normalization. The problem is that it is mostly estimation, hence some accuracy is not perfect.

3.4.4 Singular Value Decomposition (SVD)

This method was proposed by Markley [86]. It is not usually used while it output type is DCM, not quaternions. It is used however for simulations, like for example by McBryde and Lightsey [75]. Since the needed output from attitude determination must be quaternion, not DCM, this method will not be used.

The algorithm works as follows:

Firstly the matrix $\boldsymbol{B}$ is calculated with n observed stars, where b_i is image star and r_i catalog star for real star i:

$$\boldsymbol{B} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{b_i r_i^T} \tag{65}$$

SVD of matrix $\boldsymbol{B}$ has to be found

$$\boldsymbol{B} = \boldsymbol{U S V^T} \tag{66}$$

where matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal and diagonal matrix of singular values $\boldsymbol{S}$.

Now the proper orthogonal matrices $\boldsymbol{U}_+$ and $\boldsymbol{V}_+$ using:

$$\boldsymbol{U}_+ = \boldsymbol{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det\boldsymbol{U} \end{bmatrix} \tag{67}$$

$$\boldsymbol{V}_+ = \boldsymbol{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det\boldsymbol{V} \end{bmatrix} \tag{68}$$

The DCM $\boldsymbol{D}$ can be found by:

$$\boldsymbol{D} = \boldsymbol{U}_+ \boldsymbol{V}_+^{\boldsymbol{T}} \tag{69}$$

3.4.5 Other techniques

There exist also other methods:

- The Fast Optimal Attitude Matrix[87],
- Predictive Attitude Determination[88].

However they are not very popular and not often used, therefore they will not be described here.

3.4.6 Conclusions

From all the presented methods, QUEST was selected due to low computational requirements and good accuracy in attitude estimation.

QUEST outperforms q-method in terms of speed[29], and while it is less accurate it still qualifies for satellite. Comparing to TRIAD, QUEST is more robust to noise. TRIAD is designed only for two vectors, hence in case of inaccurate measurements, it will yield erroneous result, even if there are more vector that could be used for estimation. According to Çelik and Hajiyev [89] the computational time and accuracy in q-method and TRIAD are comparable, hence QUEST is faster from both of them.

QUEST was also chosen by Huffman et al. [35] and Tappe [31] to be used on satellites' star-trackers.

4 Designed star-tracker program

4.1 Choice of tools and solutions

Star-tracker program was written in Python 3.6 under Anaconda distribution run on ARM processor (on Jetson TX2) with usage of numba library[1] for easy GPU computation. Python was chosen due to simplicity of programming - it is easier and faster to build a program. Additionally due to usage of libraries like numba and numpy[2] it is only few times slower than if it would be written in C++[92][93][94]. Of course it is possible to rewrite parts of program to C++ for better performance, however it will be around 2-10 times faster with a lot of additional work.

Using CUDA on GPU was chosen because it made program run significantly faster. The exact difference is visible in Figure... TODO ADD CUDA vs NO CUDA!!!!!

4.2 Comparison of designed star-tracker with existing solutions

4.3 Star-tracker design

The program has main file which is responsible for running whole program, and the rest is divided in three parts: star recognition - calculating centroids of stars from image and discarding the ones with too high magnitude; star identification - finding stars' catalog IDs and discarding false stars, using isolated Planar Triangle algorithm; attitude estimation using isolated QUEST algorithm. Additionally there is separate part for catalog generation for Planar Triangle algorithm.

[1] "Numba translates Python functions to optimized machine code at runtime using the industry-standard LLVM compiler library. Numba-compiled numerical algorithms in Python can approach the speeds of C or FORTRAN."[90]

[2] "NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities."[91]

During star-tracker development it was necessary to firstly create testing environment in order to ensure that program will work correctly. Each part of algorithm was tested separately on wide range of examples, then they were together as fully functioning star-tracker program.

During development k-vector algorithm was discarded because results of star identification with k-vector were inconsistent. Moreover for just 1500 stars it didn't give any visible speed advantage. It is possible it would be worth using in case of catalogs with more stars.

The Tracking Mode does not function properly and will have to be rewritten in the future version of star-tracker. Although correctly working Tracking Mode would make star-tracker faster, it is possible for it to work all the time in LIS mode.

Figure 17 shows general schema of designed star-tracker program.

For clarification the following entities will be used in this chapter:

- image star - star in the image;
- image star ID - ID of star recognized in the image (serves as ordinal number);
- catalog star - star in the Hipparcos catalog;
- Hipparcos star ID - ID of the star in Hipparcos catalog;
- false star - object initially recognized as star that is later identified as false star (marked with -1), it can be planet, satellite, star with magnitude higher than in catalog, etc.

4.3.1 Star Triangle Catalog

The star triangle catalog was created with the following parameters: - stars with max magnitude 5; - FOV 10°what gave 1625 stars which gave 44922 triangles.

In other work[12] around 6000 stars were used to create catalog, hence another catalog was created with stars' magnitude 6.2, what gave 6279 stars and 1160853 triangles. This however proved to be both not accurate and a lot slower in testing than in case of smaller catalog, hence the smaller catalog was chosen to be used in this star-tracker program.

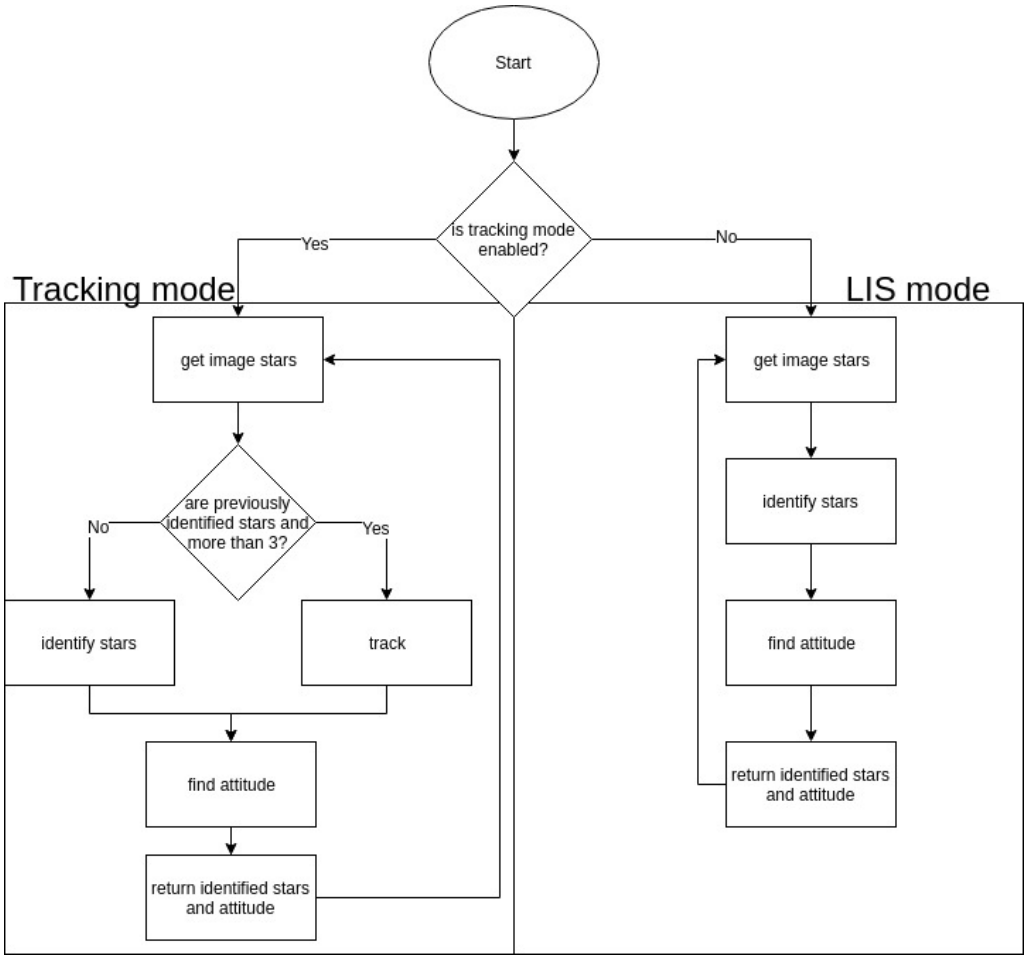


Figure 17: Designed star-tracker full program diagram

The small catalog is good enough, because it covers most of the sky with at least 3 stars. Tests prove that in random part of the sky with few magnitude 5 stars program is able to identify stars in around 90% of cases. It is best if no star with magnitude greater than the given in the catalog would be passed from star centroid algorithm into star identifier, although star identifier can deal with some false stars with quite high accuracy.

The catalog has following columns:

1. star 0 Hipparcos ID,
2. star 1 Hipparcos ID,
3. star 2 Hipparcos ID,
4. area,
5. polar moment,
6. k-vector ID.

4.3.2 Star Catalog

There was additional catalog created for stars only. It is used by Star Identifier for eliminating incorrect stars and by Orientation Finder as internal frame star vectors for LIS mode. The catalog was created with stars of max magnitude 6.2.

The catalog has following columns:

1. star Hipparcos ID,
2. unit vector 1st element,
3. unit vector 2nd element,
4. unit vector 3rd element.

4.3.3 Star recognition

First camera reads its input and gets image, which is later converted into matrix of light intensity values per pixel. After that program enters into centroiding algorithm. On GPU, each pixel is checked firstly for its light intensity value, average light intensity of neighboring pixels and if the pixel

is not on the border. If all those conditions are passed, then pixel centroid is calculated for this pixel, according to the instructions in the previous chapter. After that pixel centroids are being clustered in order to merge neighboring centroids into one star centroid. Finally all found star centroids are converted to unit vectors.

Centroid algorithm was tested using images created with Stellarium, free planetarium software, which made possible to mock input to star-tracker camera. Images are with following parameters: FOV 10°and 900x900 pixels, and imitate parts of that should be visible by star-tracker on the actual orbit.

This is the most computation-consuming part of the program, however the usage of GPU made it quite fast.

On Jetson star recognition lasts around 200 ms in most cases, but it can go up higher with more stars in the input image. It will also go much higher in images with higher resolution.

Computational complexity is $O(x * y * (\frac{a_{MAG}}{2} + \frac{a_{ROI}}{2}))$, where:

- x is number of horizontal pixels,
- y is number of vertical pixels,
- a_{MAG} is star size limit in pixels,
- a_{ROI} is size of ROI in pixels.

4.3.4 Star identification

After having list of image star centroids in form of unit vectors program enters Star Identification phase. Firstly 3-star combinations are created, and for each combination all eligible stars are being found. It is done by calculating Planar Triangle for each 3-star combination and looking up all triangles in Star Triangle Catalog that are withing area and polar moment ranges. Each image star has a list of eligible stars, which is being extended after each lookup, i.e. each image star in the 3-star combination has added the same eligible stars to their lists. After doing this for all 3-star combinations, each image star has list with all eligible stars and one of the most common Hipparcos star IDs in the list should be Hipparcos star ID. Next step is done for each image star: take 10 most common eligible Hipparcos star IDs. If

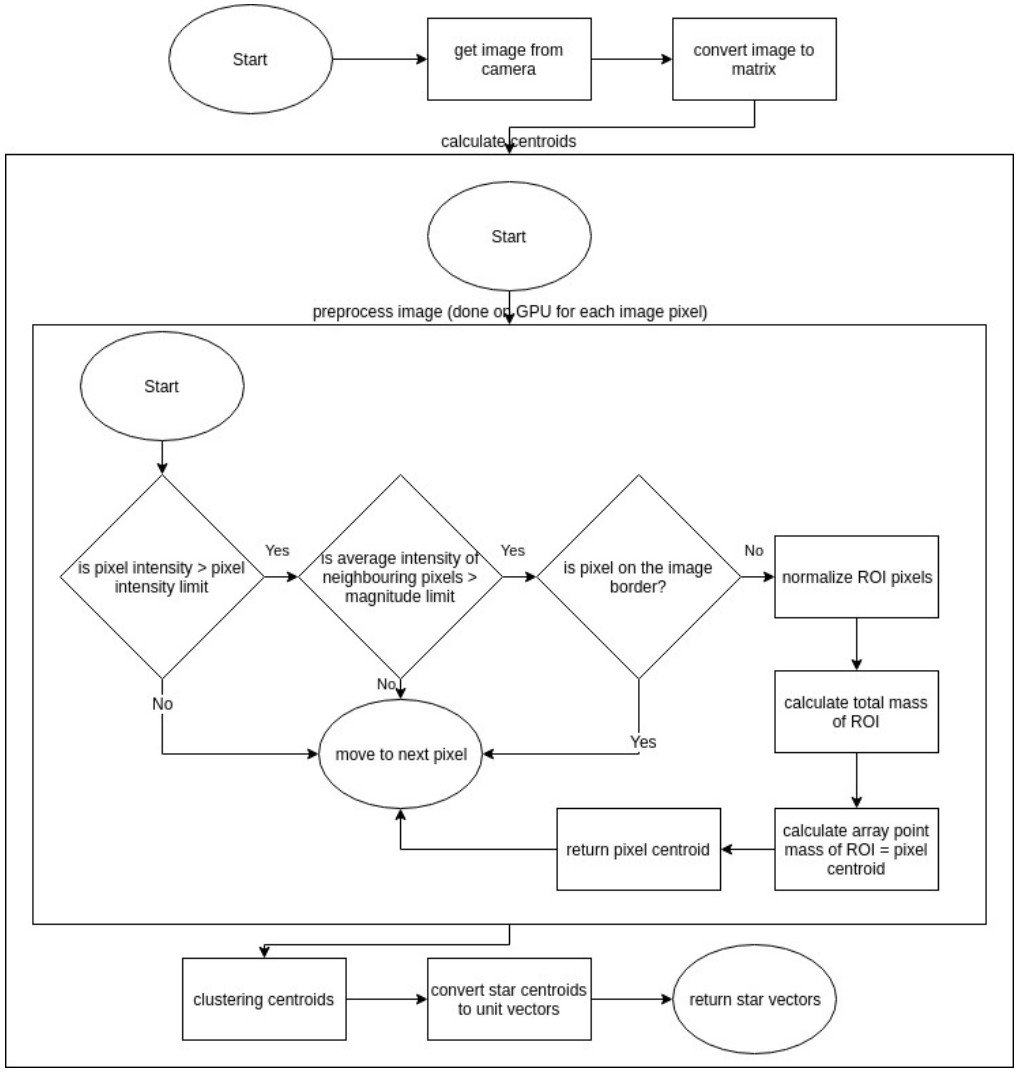


Figure 18: Getting centroids of image stars diagram

there are none, then this image star is identified as false star. If there are any eligible stars, then for each eligible Hipparcos star ID it is checked, whether this ID was already assigned to any other image star in this image. If it was not, then this eligible Hipparcos star ID is assigned to this image star as its Hipparcos star ID. If it was, then move to next most common eligible Hipparcos star ID. If this is the last most common eligible star, then this image star is identified as false star. After above steps are done for all image stars, the last step of stars identification takes place: removal of incorrect stars. For each image star with found Hipparcos star ID, those stars are looked up in Star Catalog together with their unit vectors. Next each pair of stars is checked if they actually even could be in the same image scene. This is done the same way as during Triangle Catalog creation, by checking if the dot product of unit vectors is higher or equal to the cosine of FOV. The stars which have been marked as incorrect most times are marked as false, the rest stays the same. After this step is finished, the program already has finished star identification and as result it returns list of image stars with information what is its Hipparcos star ID or if it is false star.

On Jetson Star Identification lasts up to 100-200 ms in most cases, but it can go much higher with more stars in the input image, even to 5 seconds. Possibly rewriting parts of this algorithm to use GPU would make it execute faster.

Computational complexity is $O(n^3 + n^2)$, where n is number of image stars.

Star Identifier algorithm was tested basing on ESA's star-tracker simulator created for competition needs[16]. This simulator creates data two sets of data: one that would be the actual output of star recognition algorithm, i.e. list of stars' centroids and magnitudes (x position, y position, magnitude); and the second set of data - list of stars' ids in Hipparcos catalog. All tests were done with FOV 10°. On average algorithm using star triangle catalog of magnitude $\leqslant 5$ has accuracy above 90%. Accuracy in this case means the percent of scenes in tests where no star was recognized incorrectly. It means that algorithm could also not recognize any star from the input data in given scene. Such particular accuracy definition comes from the fact that for given scene no star can be recognized incorrectly in order to get correct result in the next part of program, which is QUEST algorithm. The percent of all identified stars in those test cases is around 95% for stars within the catalog magnitude.

Test cases using star triangle catalog with magnitude $\leqslant 6$ proven to be also as accurate ($\tilde{9}5\%$) for input data with stars with magnitude $\leqslant 5.56$, however the time of execution per scene rose significantly (from 50ms to 2.2s, 44 times). Test cases with input data with stars with magnitude $\leqslant 6$ has worsen the accuracy - less scenes without incorrectly identified stars comparing to the same test with catalog magnitude $\leqslant 5$, while time rose significantly (from 200ms to 4.9s, 24.5 times). In order to make this algorithm more accurate on higher magnitude stars it is necessary to explore options of rewriting parts of this algorithm to GPU, using k-vector and finding better constant values for Planar Triangle algorithm.

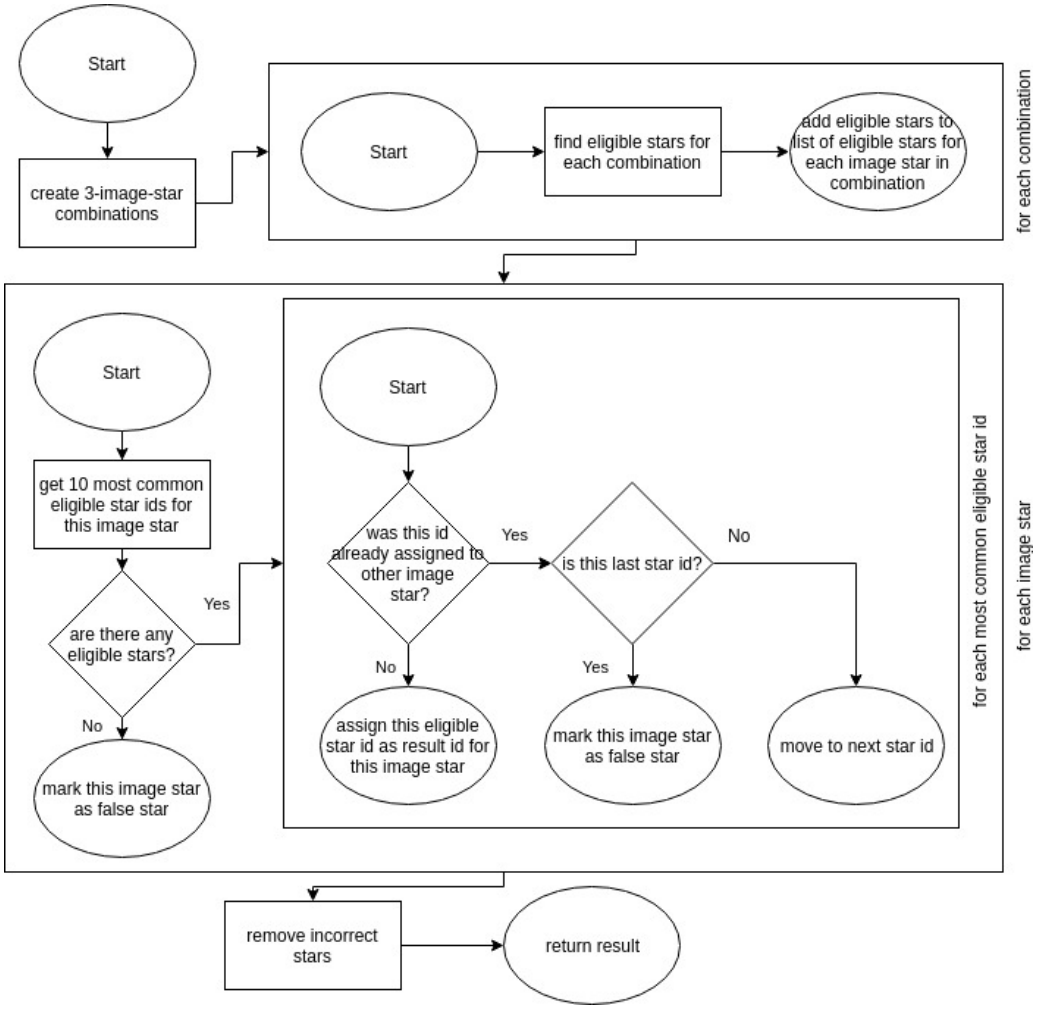


Figure 19: Star identification diagram

4.3.5 Finding attitude

After having list of image stars with Hipparcos star IDs it is possible to determine attitude. For that there are two ways: in Tracking mode current star vectors are compared with previous star vectors; in LIS mode the current star vectors are compared with corresponding star vectors from Star Catalog. Attitude is calculated using QUEST algorithm. The QUEST algorithm was tested with example given in[30] and gives identical result as in the book.

On Jetson Attitude Determination lasts below 10 ms in most cases, but it can go even up to 200 ms. This can happen if high percentage of stars in scene is misidentified and it takes more iterations to converge.

Computational complexity is $O(n)$, where n is number of star vectors.

No block diagram is needed to be presented for this part of the program, because it is just sequential calculating using equations in previous chapter.

4.4 Example of working star-tracker

This example is the first scene from "test full star-tracker" in the code. On the input Figure 20 is given - the image imitating what star-tracker should get from camera. Image was created using Stellarium program.

After recognizing stars in image and calculating their centroids, the data in Table 21 are obtained. Each star has assigned its image ID, vector values and image coordinates. The data are visualized in Figure 21, where red numbers indicate image ID of stars in the image.

Next, for each combination of 3 image stars, eligible star are calculated and added list of eligible stars for each image star. Table 3 shows lists of eligible stars after doing it for all combinations. Only 10 most common occurrences of Hipparcos star ID are shown in the table due to long list. It is clearly visible, that stars 1 and 2 have significantly smaller number of highest occurring Hipparcos star IDs.

After that each image star is being assigned its Hipparcos star ID according to algorithm described in previous chapter. The result is visible in

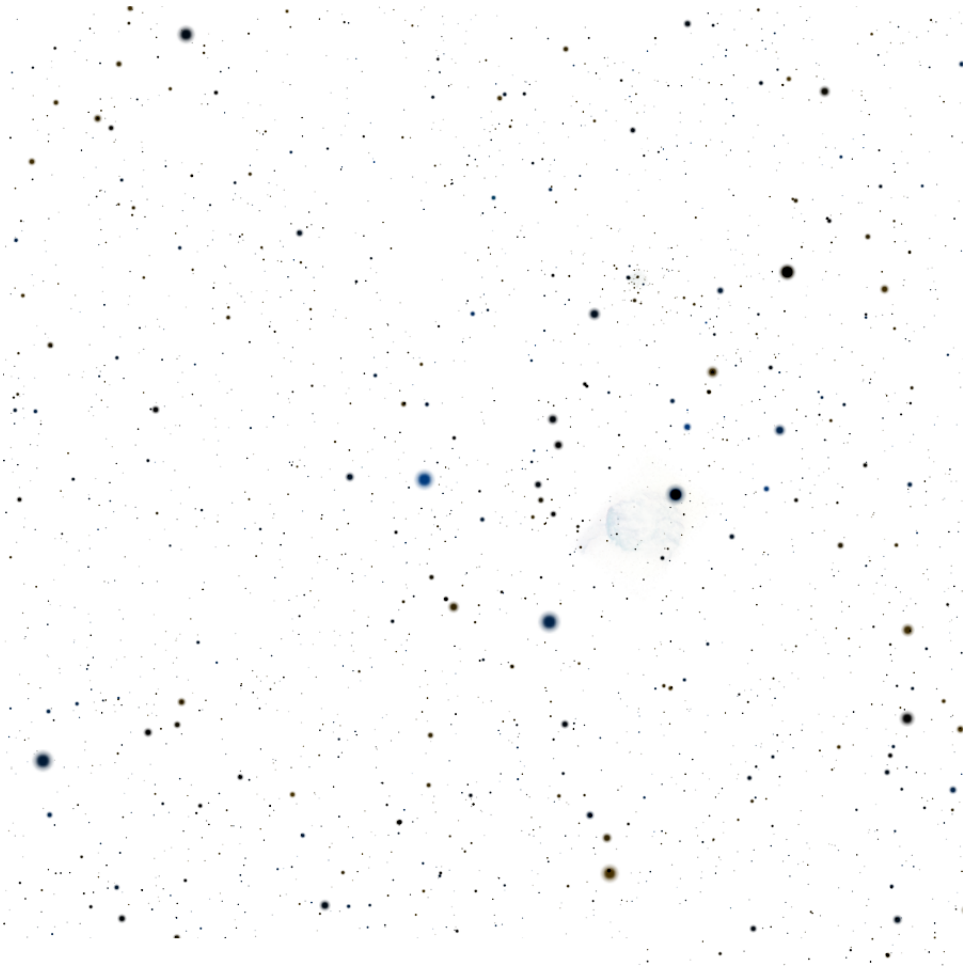


Figure 20: Example image input (colors were inverted for better visibility on paper)

image ID	uv1	uv2	uv3	pos x	pos y
0	-0.0703	0.0227	0.997	567.0	86.9
1	-0.0423	0.0767	0.996	846.18	231.46
2	0.0009	-0.0108	0.999	394.28	455.09
3	-0.0499	-0.0799	0.995	37.00	191.95
4	-0.0248	0.0119	0.999	511.42	322.07
5	0.0817	-0.0539	0.995	171.12	872.48
6	-0.0016	0.0348	0.999	629.57	441.50
7	0.0386	0.0550	0.997	734.03	649.48

Table 2: Result of star recognition in the image - stars with unit vectors and image coordinates

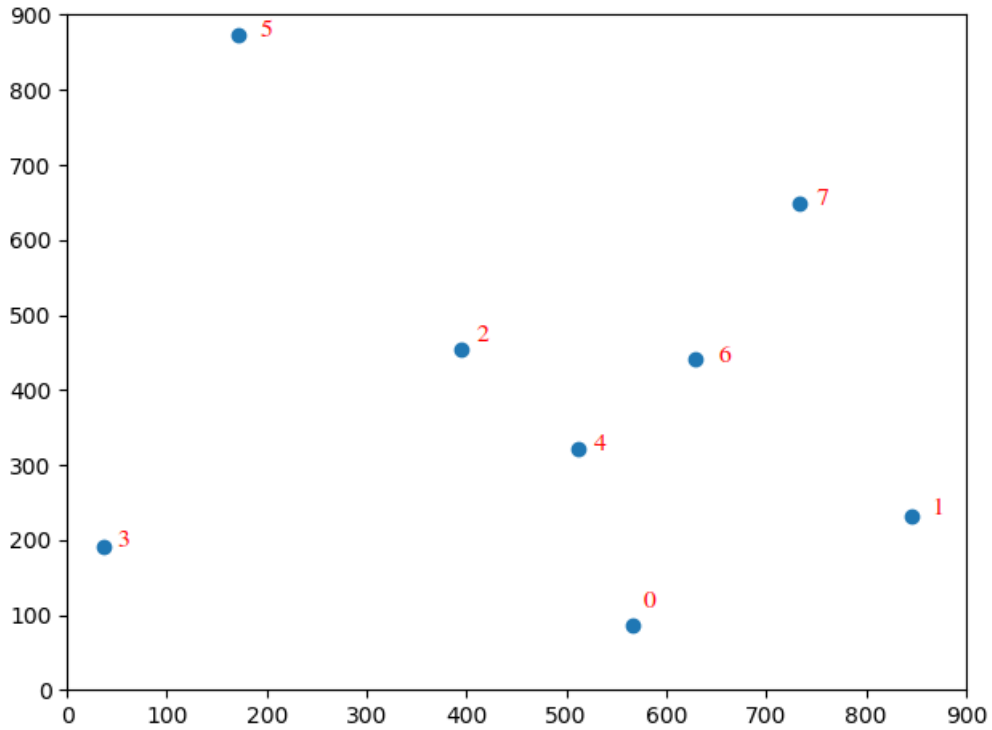


Figure 21: Result of star recognition (red indicates image star ID assigned by algorithm)

image ID	number of all stars	10 most common stars (ID: amount), decreasing
0	42	30883: 5, 29655: 3, 29696: 3, 30343: 2, 25737: 1, 26594: 1, 26727: 1, 34059: 1, 38500: 1, 38957: 1, ...
1	24	26727: 2, 52468: 2, 56561: 2, 57363: 2, 25737: 1, 26594: 1, 86974: 1, 47391: 1, 87998: 1, 51849: 1, ...
2	33	52468: 2, 34059: 1, 38500: 1, 38957: 1, 86974: 1, 47391: 1, 87998: 1, 51849: 1, 88267: 1, 47854: 1, ...
3	75	32246: 7, 29655: 4, 52468: 3, 29696: 3, 30343: 3, 28734: 3, 57363: 2, 91919: 1, 92791: 1, 94713: 1, ...
4	48	30343: 8, 29696: 4, 29655: 4, 32246: 3, 28734: 3, 30883: 2, 33856: 2, 91919: 1, 92791: 1, 94713: 1, ...
5	57	29696: 9, 30343: 4, 29655: 4, 28734: 4, 30883: 3, 32246: 3, 34059: 1, 38500: 1, 38957: 1, 17304: 1, ...
6	54	29655: 9, 32246: 4, 30343: 4, 29696: 4, 30883: 3, 28734: 3, 57363: 2, 57851: 2, 2912: 1, 4436: 1, ...
7	63	28734: 7, 29696: 4, 30343: 3, 32246: 3, 29655: 3, 25737: 2, 57363: 2, 26594: 1, 26727: 1, 25302: 1, ...

Table 3: Eligible stars after combinations planar triangle

Table 4.

image ID	Hipparcos ID	uv1	uv2	uv3	pos x	pos y
0	30883	-0.0703	0.0227	0.997	567.0	86.9
1	26727	-0.0423	0.0767	0.996	846.18	231.46
2	52468	0.0009	-0.0108	0.999	394.28	455.09
3	32246	-0.0499	-0.0799	0.995	37.00	191.95
4	30343	-0.0248	0.0119	0.999	511.42	322.07
5	29696	0.0817	-0.0539	0.995	171.12	872.48
6	29655	-0.0016	0.0348	0.999	629.57	441.50
7	28734	0.0386	0.0550	0.997	734.03	649.48

Table 4: Result of star identification before removing incorrect stars

To ensure the correctness of the data, the last part of star identification is executed: removing incorrect stars. Table 5 shows the most commonly appearing Hipparcos star IDs that could not be in the same image. Two IDs: 26727 and 52468 appear significantly more often than the rest of IDs, therefore they are going to be removed from the result and marked with ID "-1", because it represents false star. 26727 and 52468 happen to be found Hipparcos star ID for image stars 1 and 2, respectively.

The image star 1 was not found, because it is star with Hipparcos ID 29650 and magnitude higher than limiting 5. Therefore program marked this star as false star. Similarly with image star 2 - it is Mars planet, not a star, hence program marked it also as false star.

The final result of star identification is visible in Table 6 and is visualized in Figure 22. The red labels in the image are Hipparcos stars IDs and name of the planet Mars.

26727: 7, 52468: 7, 28734: 2, 29655: 2, 29696: 2, 30343: 2, 30883: 2, 32246: 2

Table 5: Hipparcos IDs of incorrect stars

image ID	Hipparcos ID	uv1	uv2	uv3	pos x	pos y
0	30883	-0.0703	0.0227	0.997	567.0	86.9
1	-1	-0.0423	0.0767	0.996	846.18	231.46
2	-1	0.0009	-0.0108	0.999	394.28	455.09
3	32246	-0.0499	-0.0799	0.995	37.00	191.95
4	30343	-0.0248	0.0119	0.999	511.42	322.07
5	29696	0.0817	-0.0539	0.995	171.12	872.48
6	29655	-0.0016	0.0348	0.999	629.57	441.50
7	28734	0.0386	0.0550	0.997	734.03	649.48

Table 6: Final result of star identification

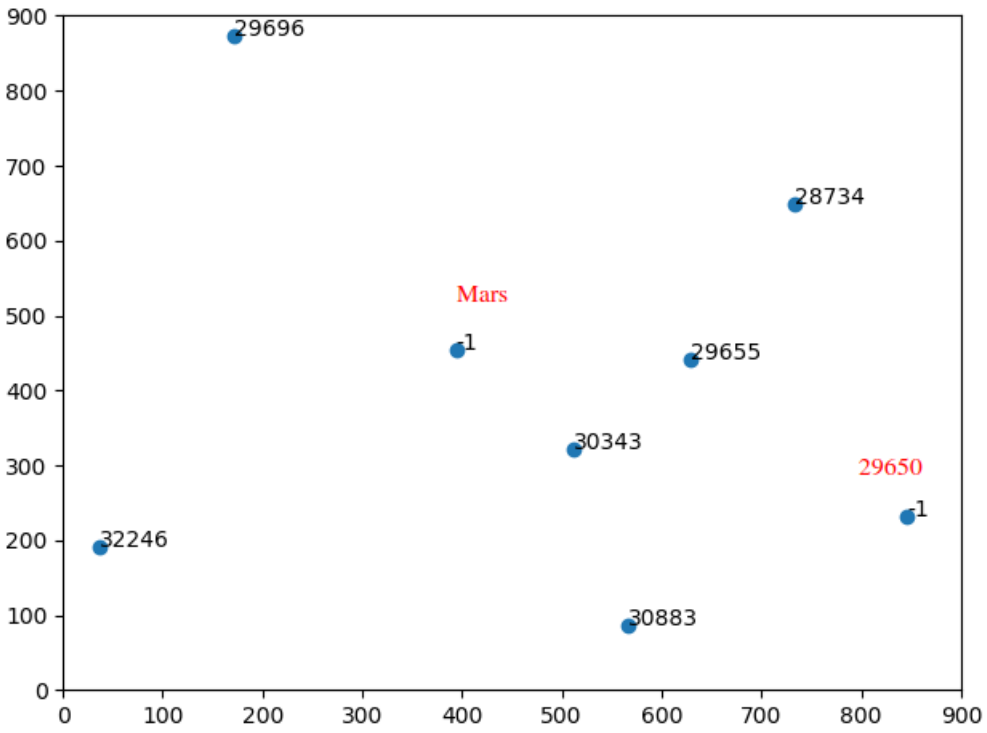


Figure 22: Final result of star identification (black indicates the Hipparcos star ID found by algorithm, red indicates the real ID, "-1" indicates "false star")

Finally the identified stars are used for calculating quaternion. The resulting quaternion is visible in Equation 70, where the form of quaternion is described in Equation 7.

$$\boldsymbol{q} \coloneqq \begin{bmatrix} -0.52101746 \\ -0.15332225 \\ -0.16303070 \\ 0.82368324 \end{bmatrix} \tag{70}$$

The resulting quaternion can be translated to Euler angles as Right Ascension, Declination and Roll of the satellite in J2000 frame (Hipparcos catalog). The Equation 71 shows the result quaternion in Euler angles. Right Ascension is given in angle hours for easier applying in Stellarium program, where the result could be visualized. Figure 23 shows the attitude of satellite according to star-tracker, while Figure 24 shows the real attitude. The small difference could come from the imperfect transformation from quaternion to Euler angles.

$$\begin{aligned} RA &= 6h27m34.6927s \\ DEC &= 24.990097882696112 \\ Roll &= 63.100846657835916 \end{aligned} \tag{71}$$

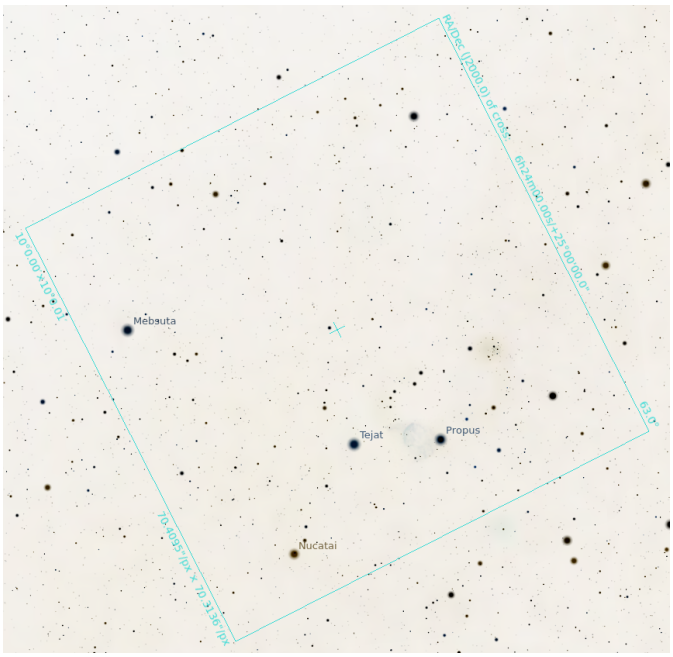


Figure 23: Attitude according to star-tracker. The square frame is the FOV of star-tracker's camera

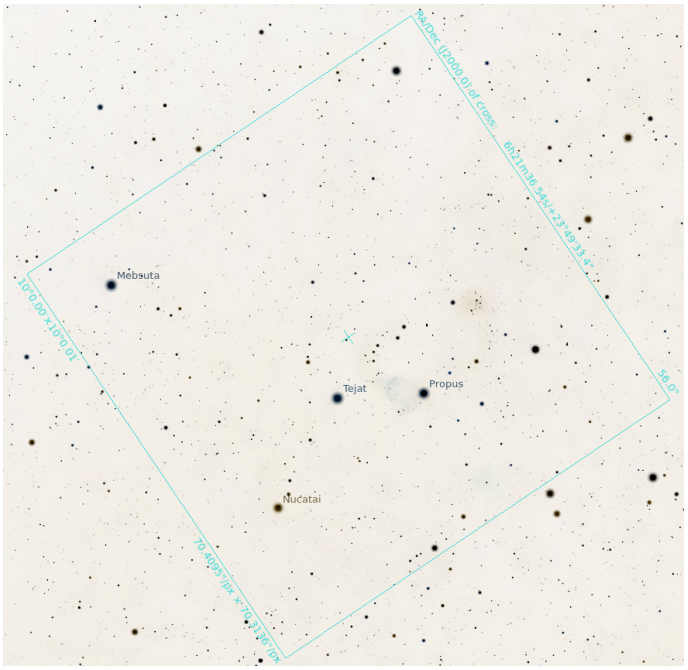


Figure 24: Real attitude. The square frame is the FOV of star-tracker's camera

5 Testing on NVIDIA Jetson TX2

Testing of full star-tracker program was done in two ways:

First way of testing was to give image of sky on input and then to go through all parts of the program. The images were created using Stellarium program, which imitates part of sky with given FOV and coordinates. The parameters of images used during testing are FOV 10°, and resolution of 900x900 pixels.

Second way of testing was showing image of the sky on the separate screen and try to read it with camera on JETSON device. Although program clearly was able to find stars in the camera input, it needs precise configuration in order to find centroids accurately. Example of precise testing was done by Tappe et al. [95]. Surely such testing will have to be done to test using this star-tracker program with dedicated camera.

The following results come from first way of testing: simulating input of star-tracker with already made images. The series of 14 images was run one by one. Figure 25 shows the power consumption of algorithm on Jetson TX2. The values were collected by reading metrics collected by Ubuntu system while running the star-tracker test. The first peak is the loading of catalogs into the memory. At around 3rd second the first scene is read. Even with the highest load power consumption does not exceed 7 Watts. It is clearly visible that CPU consumption is significantly higher than GPU and is due to the fact that still most of star-tracker computation is done on CPU. High power consumption peaks in seconds 9-14 and 23-28 are due to the higher amount of stars in input scenes at that time, what leads to longer time of star identification process, which is the CPU consuming process.

Figure 26 shows how much time was necessary to calculate attitude for each of 14 scenes. Scenes 2 and 12 have higher time of Attitude calculation because in those scenes quite big percentage of stars were misidentified and it took more iterations to finalize calculations. Enormously higher Star Identification times for scenes 10, 11 and 12 come from the much higher amount of stars in the scene image.

Figure 27 shows power consumption with marked times of each section calculation. Scene 10 takes time between 6th and 8th second and it is visible that power consumption raises a little. However in the next scene, scene

11 (seconds 8-13) the power consumption peaks from 5 to nearly 7 Watts. Scene 12 (seconds 13-18) also notes higher consumption, however smaller than previous scene. The next two scenes take very little time and around second 19 the next sequence of 14 images is started (red vertical line).

It is visible that the higher amount of stars on the input make program work significantly slower. While Cenotriding part needs rather the similar amount of time and Attitude Determination is not computationally expensive (takes longer only in case of misidentified stars, and still it was at most 200 ms), Star Identification can take even 40 times more than usual. The possible solution would be to rewrite Star Identification algorithm to be computed on GPU, especially because GPU power consumption stays on much lower level than CPU's.

NVIDIA Jetson TX2 seems to be good candidate to be used in space as the star-tracker hardware, however not for small CubeSats due to their low power budget. It could be used in little larger satellites, like in at least 6U CubeSats or HyperSats. Although existing star-trackers with dedicated hardware have much lower power consumption, even around 0.65-2.5 W, it is possible that optimization of star-tracker software would lower the power consumption. Jetson's low power consumption and GPU give possibility to create new star-tracker without spending time and resources on hardware part, but on software only. Possible Jetson could either be used to test machine learning solutions, or be used as the main on-board computer, what would lower the need for other devices on the satellite therefore lower the power consumption.

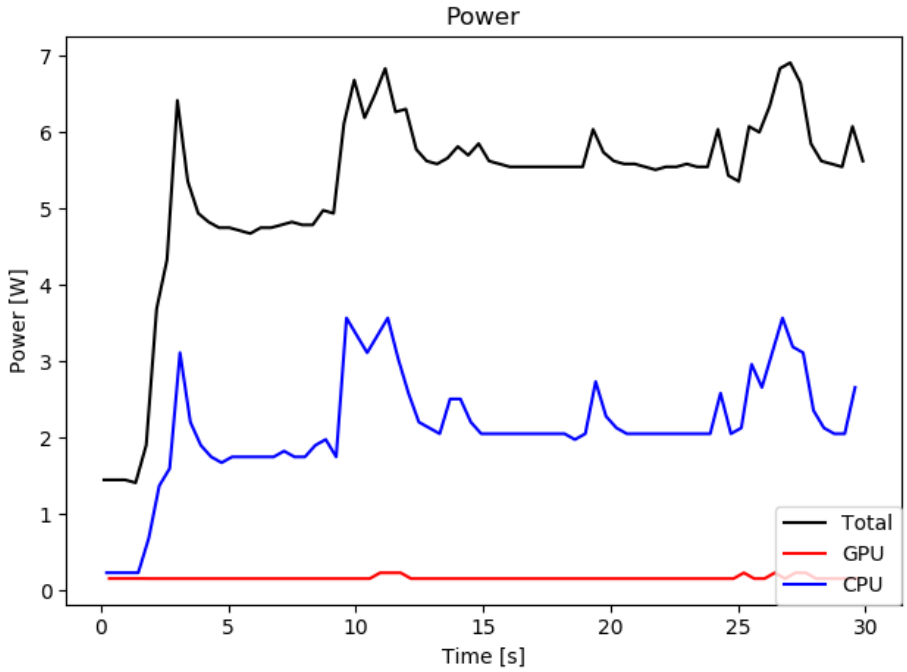


Figure 25: Power consumption of star-tracker on Jetson TX2

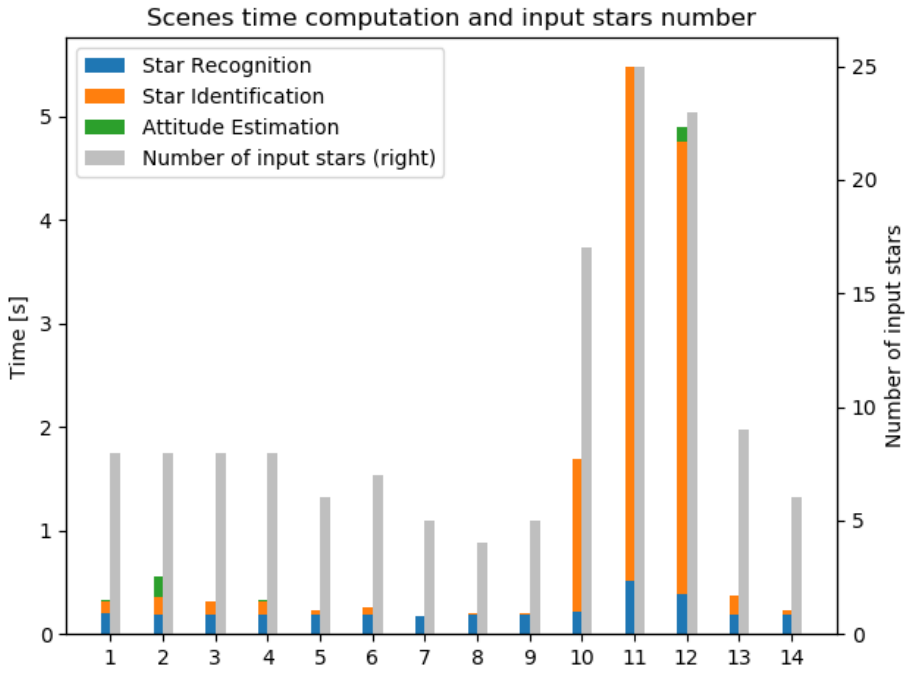


Figure 26: Star-tracker execution time of each scene

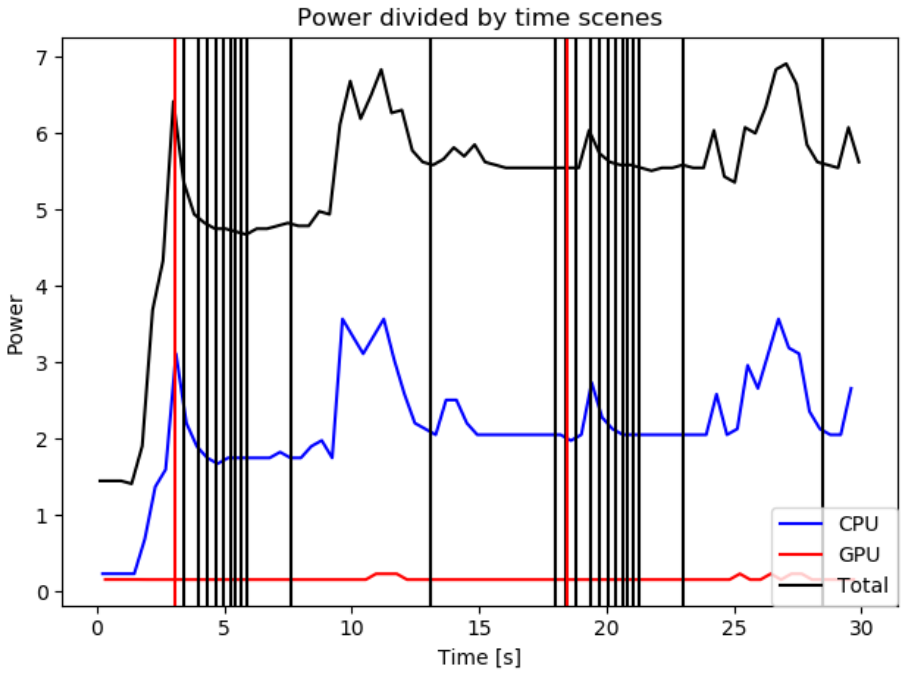


Figure 27: Power consumption with marked scenes' computation times. Red vertical lines mark start of next sequence of 14 input scenes, black vertical lines describe the start of next scene

6 Summary and future steps

The program was tested using two sources of truth: images from Stellarium program and tests built basing on ESA's star-tracker simulator created for competition needs, what ensures high quality of tests. Testing indicated that all parts of star-tracker: star recognition in the images (centroiding), star identification with star catalog (planar triangle) and attitude determination (QUEST) work with high accuracy separately. The main goal of the thesis - star identification part - was thoroughly tested with few thousands mocked scenes and few generated taken from Stellaris program. The speed of execution of the program shows that program could be used on the real satellite with refresh frequency at around $1Hz$ on average for LIS mode, and at least few times more with correctly working Tracking mode. The star-tracker is able to find attitude from the image of stars and calculate quaternion from it.

In order to use created star-tracker program on satellite the following things are necessary to be done:

- test with the dedicated camera,
- run program in CPython implementation or rewrite program (or at least GPU parts) to C/C++ because the Python Anaconda distribution for ARM architecture is experimental and it could simply break on the satellite,
- rerun catalog generation and test for the dedicated set of variables (FOV, etc.).

There exist a number of optional things to explore for bettering existing star-tracker:

- rewrite star identifier on GPU,
- rewrite tracking mode to work more accurately,
- optimize Planar Triangle algorithm,
- catalog with more stars,

- use k-vector algorithm for more stars,
- explore other algorithms, especially for star identification and attitude estimation, because this is area which is being improved mostly.

NVIDIA Jetson TX2 seems to be good candidate to be used in space as the star-tracker hardware for slightly bigger satellites. With software optimization it could lower its consumption below 7 Watts, possibly even to around 5 Watts (power consumption under low CPU load). Other option is to use Jetson also instead of other devices what would lower the need for other components and would lower overall power consumption of satellite. Jetson can also be used for testing machine and deep learning solutions for star-trackers. All goals set for this thesis have been met. Although there are few parts which could be improved, the existing star-tracker program works and could be with only few changes used on the real satellite. Additionally all the work done around the star-tracker program, like tests, etc., can be used for future development of even more accurate programs.

References

[1] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. Twiggs, "CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation," *14th Annual/USU Conference on Small Satellites*, 2000.

[2] M. A. Swartwout, "A brief history of rideshares (and attack of the CubeSats)," in *Aerospace Conference, 2011 IEEE*. IEEE, 2011, pp. 1–15.

[3] C. C. Liebe, "Accuracy performance of star trackers-a tutorial," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 587–599, 2002.

[4] M. A. Samaan, D. Mortari, T. Pollock, and J. L. Junkins, "Predictive centroiding for single and multiple FOVs star trackers," *Advances in the Astronautical Sciences*, vol. 112, pp. 59–71, 2002.

[5] M. W. Knutson, "Fast star tracker centroid algorithm for high performance CubeSat with air bearing validation," Master's thesis, Massachusetts Institute of Technology, 2012.

[6] M. Azizabadi, A. Behrad, and M. Ghaznavi-Ghoushchi, "VLSI implementation of star detection and centroid calculation algorithms for star tracking applications," *Journal of real-time image processing*, vol. 9, no. 1, pp. 127–140, 2014.

[7] M. Lindh, "Development and implementation of star tracker electronics," Master's thesis, KTH ROYAL INSTITUTE OF TECHNOLOGY, 2014.

[8] P. Zhang, Q. Zhao, J. Liu, and N. Liu, "A brightness-referenced star identification algorithm for aps star trackers," *Sensors*, vol. 14, no. 10, pp. 18 498–18 514, 2014.

[9] C. Padgett and K. Kreutz-Delgado, "A grid algorithm for autonomous star identification," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 1, pp. 202–213, 1997.

[10] D. Mortari, "SP-search: A new algorithm for star pattern recognition," *Advances in the Astronautical Sciences*, vol. 102, no. Pt II, pp. 1165–1174, 1999.

[11] C. L. Cole and J. Crassidus, "Fast star pattern recognition using spherical triangles," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit. Providence, Rhode Island: AIAA*, 2004.

[12] D. Mortari, M. A. Samaan, C. Bruccoleri, and J. L. Junkins, "The pyramid star identification technique," *Navigation*, vol. 51, no. 3, pp. 171–183, 2004.

[13] C. L. Cole and J. L. Crassidis, "Fast star-pattern recognition using planar triangles," *Journal of guidance, control, and dynamics*, vol. 29, no. 1, pp. 64–71, 2006.

[14] Y. Dong, F. Xing, and Z. You, "Brightness independent 4-star matching algorithm for lost-in-space 3-axis attitude acquisition," *Tsinghua Science & Technology*, vol. 11, no. 5, pp. 543–548, 2006.

[15] M. Kolomenkin, S. Pollak, I. Shimshoni, and M. Lindenbaum, "Geometric voting algorithm for star trackers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 2, pp. 441–456, 2008.

[16] ""Star Trackers: First Contact" competition, European Space Agency," https://kelvins.esa.int/star-trackers-first-contact/, accessed: 2019-06-20.

[17] T. Lindblad, C. S. Lindsey, Å. Eide, Ö. Solberg, and A. Bolseth, "Star Identification using Neural Networks," *Proceedings of SPIE - The International Society for Optical Engineering*, 1997.

[18] C. Li, K. Li, L. Zhang, S. Jin, and J. Zu, "Star pattern recognition method based on neural network," *Chinese Science Bulletin*, vol. 48, no. 18, pp. 1927–1930, 2003.

[19] S. S. Miri and M. E. Shiri, "Star identification using Delaunay triangulation and distributed neural networks," *International Journal of Modeling and Optimization*, vol. 2, no. 3, p. 234, 2012.

[20] D. Mortari, "A fast on-board autonomous attitude determination system based on a new star-ID technique for a wide FOV star tracker," *Advances in the Astronautical Sciences*, vol. 93, pp. 893–904, 1996.

[21] D. Mortari and B. Neta, "K-vector range searching techniques," *Adv. Astronaut. Sci*, vol. 105, pp. 449–464, 2000.

[22] D. Mortari and J. Rogers, "A k-vector Approach to Sampling, Interpolation, and Approximation," *The Journal of the Astronautical Sciences*, vol. 60, no. 3-4, pp. 686–706, 2013.

[23] T. Delabie, "A highly efficient attitude estimation algorithm for star trackers based on optimal image matching," in *AIAA Guidance, Navigation and Control Conference, Minneapolis, Minnesota*, 2012.

[24] M. Shuster, "Kalman filtering of spacecraft attitude and the QUEST model," *Journal of the Astronautical Sciences*, vol. 38, pp. 377–393, 1990.

[25] M. Psiaki, "Extended quest attitude determination filtering," in *NASA CONFERENCE PUBLICATION*. NASA, 1999, pp. 1–16.

[26] J.-N. Juang, H.-Y. Kim, and J. L. Junkins, "An efficient and robust singular value method for star pattern recognition and attitude determination," *NASA, Tech. Rep. TM-2003-212142*, 2003.

[27] T. B. Rinnan, "Development and comparison of estimation methods for attitude determination," Master's thesis, Institutt for teknisk kybernetikk, 2012.

[28] Y. Cheng and M. D. Shuster, "Improvement to the Implementation of the QUEST Algorithm," *Journal of Guidance, Control, and Dynamics*, 2014.

[29] F. L. Markley and D. Mortari, "How to estimate attitude from vector observations," *AAS/AIAA Astrodynamics Specialist Conference*, 1999.

[30] C. D. Hall, "Spacecraft attitude dynamics and control," *Lecture Notes posted on Handouts page [online]*, vol. 12, no. 2003, 2003. [Online]. Available: http://www.dept.aoe.vt.edu/~cdhall/courses/aoe4140/attde.pdf

[31] J. A. Tappe, "Development of star tracker system for accurate estimation of spacecraft attitude," Master's thesis, Monterey, California. Naval Postgraduate School, 2009.

[32] D. Felikson, J. Hahmall, M. F. Vess, and M. Ekinci, "On-Orbit Solar Dynamics Observatory (SDO) Star Tracker Warm Pixel Analysis," in *AIAA Guidance, Navigation and Control Conference, Portland, Oregon*, vol. 6728, 2011.

[33] M. Gąska, "Moduł komputera pokładowego do satelity CubeSat z funkcją Star Tracker," Master's thesis, Warsaw University of Technology, 2016.

[34] A. Rose, "STAR integrated tracker," *arXiv preprint nucl-ex/0307015*, 2003.

[35] K. M. Huffman *et al.*, "Designing star trackers to meet micro-satellite requirements," Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2006.

[36] K. D. Diaz, "Performance Analysis Of A Fixed Point Star Tracker Algorithm," Master's thesis, California Polytechnic State University, 2006.

[37] R. Kandiyil, "Attitude determination software for a star sensor," Master's thesis, Luleå University of Technology, 2010.

[38] D. Mortari and A. Romoli, "StarNav III: a three fields of view star tracker," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 1. IEEE, 2002, pp. 1–57.

[39] M. Cannata, M. Greene, S. Mulligan, and V. Popovici, "Autonomous star-imaging attitude sensor (ASIAS)," *Faculty of Science and Engineering. York University*, pp. 2006–07, 2007.

[40] S. Lizy-Destrez and D. Mimoun, "Str: a student developed star tracker for the esa-led esmo moon mission," *Global Lunar Conference*, 2010.

[41] E. Jalabert, E. Fabacher, N. Guy, S. Lizy-Destrez, W. Rappin, and G. Rivier, "Optimization of star research algorithm for ESMO star tracker," *8th International ESA Conference on Guidance, Navigation and Control Systems*, 2011.

[42] "The Gamma Satellite, NASA," https://heasarc.gsfc.nasa.gov/docs/heasarc/missions/gamma.html, accessed: 2019-06-24.

[43] "Space research at Warsaw University of Technology," http://panorama.varsovia.pl/varsovia/warstwy/cosmic/badania.php.htm, accessed: 2019-06-24.

[44] "Optical devices, lecture materials, Space Research Centre of Polish Academy of Sciences," http://www.cbk.waw.pl/teledetekcja/studia/wiosna2009/, accessed: 2019-06-24.

[45] "CubeSat solar panels, ISIS – Innovative Solutions In Space," https://www.isispace.nl/product/isis-cubesat-solar-panels/, accessed: 2019-07-21.

[46] "CubeSat Tables," http://www.nanosats.eu/tables, accessed: 2019-07-21.

[47] M. Swartwout, "The first one hundred cubesats: A statistical look," *Journal of Small Satellites*, vol. 2, no. 2, pp. 213–233, 2013.

[48] "Nanosatsatellite and CubeSat Database," http://www.nanosats.eu/, accessed: 2019-06-21.

[49] "PW-Sat2: oficjalna strona projektu PW-SAT 2 (oraz PW-SAT)," https://pw-sat.pl/, accessed: 2017-08-15.

[50] "BRITE-PL Pierwszy polski satelita naukowy," http://www.brite-pl.pl/pliki/nauka.html, accessed: 2017-08-15.

[51] K. Lemmer, "Propulsion for cubesats," *Acta Astronautica*, vol. 134, pp. 231–243, 2017.

[52] "Polskie Radio: Pierwszy polski satelita poleciał w kosmos," http://www.polskieradio.pl/7/129/Artykul/536883,Pierwszy-polski-satelita-polecial-w-kosmos, accessed: 2017-08-15.

[53] "BRITE-PL, PL2 (CanX 3C, 3D / Lem, Heweliusz) - Gunter's Space Page," http://space.skyrocket.de/doc˙sdat/brite-pl.htm, accessed: 2017-08-15.

[54] "Future Planetary Expeditions: CubeSats to the Planets," http://futureplanets.blogspot.com/2013/10/, accessed: 2017-08-15.

[55] "Warsaw University of Technology - News - PW-SAT 2 gets funding," https://www.pw.edu.pl/Studenci/Aktualnosci/Satelita-studencki-PW-Sat2-otrzyma-dofinansowanie, accessed: 2017-08-15.

[56] "HyperSat - open micro satellite platform," http://hyper-sat.com/en/index.html, accessed: 2019-06-21.

[57] P. Czapski, G. K. *, T. Zawistowski, M. Stolarski, S. Hanasz, M. Kuklewski, and M. S. Bieda, "State of The Art of Earth Observation Instruments for Small Satellites," in *69th International Astronautical Congress (IAC), Bremen, Germany*, Oct. 2018.

[58] "HyperSat - platforma dla mikrosatelitów z Piaseczna, Space24," https://www.space24.pl/hypersat-platforma-dla-mikrosatelitow-z-piaseczna, accessed: 2019-06-21.

[59] "Platforma satelitarna, Space24," https://www.space24.pl/platforma-satelitarna-szkielet-uklad-nerwowy-i-krwioobieg-satelity-analiza, accessed: 2019-06-21.

[60] "GPS: The Global Positioning System," http://www.gps.gov/systems/gps/space/, accessed: 2017-08-18.

[61] "NASA's Utilization of Global Positioning System (GPS)," https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS˙Utilization.html, accessed: 2017-08-18.

[62] "Space Station Using GPS in Attitude Control," https://www.nasa.gov/centers/johnson/news/releases/2002/j02-61.html, accessed: 2017-08-18.

[63] V. Capuano, C. Botteron, Y. Wang, J. Tian, J. Leclère, and P.-A. Farine, "GNSS/INS/Star tracker integrated navigation system for Earth-Moon transfer orbit," in *ION GNSS+ 2014*, no. EPFL-CONF-202129, 2014.

[64] W. J. Larson and J. R. Wertz, "Space mission analysis and design," Microcosm, Inc., Torrance, CA (US), Tech. Rep., 1992.

[65] "NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge, NVIDIA," https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/, accessed: 2019-06-24.

[66] "Jetson TX2 price, NVIDIA," https://www.nvidia.com/en-us/autonomous-machines/jetson-store/, accessed: 2019-06-24.

[67] "Earth's navigation orbits image," https://upload.wikimedia.org/wikipedia/commons/thumb/b/b4/Comparison˙satellite˙navigation˙orbits.svg/512px-Comparison˙satellite˙navigation˙orbits.svg.png, accessed: 2017-08-18.

[68] "NED frame image," http://www.basicairdata.eu/knowledge-center/background-topics/coordinate-system/, accessed: 2017-09-02.

[69] L. Euler, "Formulae generales pro translatione quacunque corporum rigidorum," *Novi Acad. Sci. Petrop*, vol. 20, pp. 189–207, 1775.

[70] W. R. Hamilton, "LXXVIII. On quaternions; or on a new system of imaginaries in Algebra: To the editors of the Philosophical Magazine and Journal," *Philosophical Magazine Series 3*, vol. 25, no. 169, pp. 489–495, 1844.

[71] A. Cayley, "XIII. On certain results relating to quaternions: To the editors of the Philosophical Magazine and Journal," *Philosophical Magazine Series 3*, vol. 26, no. 171, pp. 141–145, 1845.

[72] R. Courant and D. Hilbert, "Methods of mathematical physics, Volume I," 1953.

[73] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH computer graphics*, vol. 19, no. 3. ACM, 1985, pp. 245–254.

[74] G. Wahba, "A least squares estimate of satellite attitude," *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[75] C. R. McBryde and E. G. Lightsey, "A star tracker design for CubeSats," in *Aerospace Conference, 2012 IEEE*. IEEE, 2012, pp. 1–14.

[76] B. B. Spratling and D. Mortari, "A survey on star identification algorithms," *Algorithms*, vol. 2, no. 1, pp. 93–107, 2009.

[77] D. Gottlieb, "Star pattern recognition techniques," *Spacecraft Attitude Determination and Control, The Netherlands*, pp. 257–266, 1978.

[78] F. Alidoost, F. Dadras Javan, and AWT_TAG, "A Review of Pattern Matching Methods in Star Trackers," *Geospatial Engineering Journal*, vol. 4, 2013. [Online]. Available: http://gej.issge.ir/article-1-126-en.html

[79] T. Brady, C. Tillier, R. Brown, A. Jimenez, and A. Kourepenis, "The inertial stellar compass: A new direction in spacecraft attitude determination," *16th Annual USU Conference on Small Satellites*, 2002.

[80] G. Crew, R. Vanderspek, and J. Doty, "Hete experience with the pyramid algorithm," *MIT Center for Space Research, Cambridge, MA*, vol. 2139, 2002.

[81] M. Jacox, J. Ochoa, J. Zbranek, B. Wood, A. Katake, and C. Bruccoleri, "Method and Apparatus for Automatic Identification of Celestial Bodies," Apr. 13 2006, uS Patent App. 11/279,668.

[82] "StarVision Technologies Secures Exclusive License to Star Identification Software," http://www.prweb.com/releases/2005/05/prweb245145.htm, accessed: 2017-09-06.

[83] M. A. Samaan, D. Mortari, and J. L. Junkins, "Recursive mode star identification algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1246–1254, 2005.

[84] J. Keat, "Analysis of least-squares attitude determination routine DOAOP," Technical Report CSC/TM-77/6034, Comp. Sc. Corp, Tech. Rep., 1977.

[85] M. D. Shuster, "Approximate algorithms for fast optimal attitude computation," in *Guidance and Control Conference*, 1978, pp. 88–95.

[86] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[87] ——, "Attitude determination using vector observations: A fast optimal matrix algorithm," *Journal of the Astronautical Sciences*, no. 41/2, Jul. 1993.

[88] J. Crassidis and L. Markley, "A Predictive Attitude Determination Algorithm," in *NASA Conference Publication*. NASA, 1997, pp. 249–264.

[89] O. Çelik and C. Hajiyev, "A comparison of attitude determination methods for small satellites," in *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*. IEEE, 2013, pp. 261–264.

[90] "Numba: A High Performance Python Compiler," https://numba.pydata.org/, accessed: 2019-08-29.

[91] "NumPy," https://numpy.org/, accessed: 2019-08-29.

[92] "A Speed Comparison Of C, Julia, Python, Numba, and Cython on LU Factorization, IBM," https://www.ibm.com/developerworks/community/blogs/jfp/entry/A˙Comparison˙Of˙C˙Julia˙Python˙Numba˙Cython˙Scipy˙and˙BLAS˙on˙LU˙Factorization?lang=en, accessed: 2019-08-29.

[93] "NUMBA VERSUS C++," https://murillogroupmsu.com/numba-versus-c/, accessed: 2019-08-29.

[94] "Benchmarks of speed (Numpy vs all)," https://arogozhnikov.github.io/2015/01/06/benchmarks-of-speed-numpy-vs-all.html, accessed: 2019-08-29.

[95] J. Tappe, J. J. Kim, A. Jordan, and B. Agrawal, "Star tracker attitude estimation for an indoor ground-based spacecraft simulator," in *AIAA Guidance, Navigation, and Control Conference*, vol. 1, 2011, pp. 1116–1122.

List of Tables

List of Figures