Performance of a Smartphone based Star Tracker

Andrey Khorev, Lionel Torres
University of Montpellier, LIRMM UMR CNRS
France

Eric Nativel
University of Montpellier, IES UMR CNRS
France

***Abstract*—Nowadays, CubeSat missions grow more and more complicated. Such tasks as telecommunication, Earth observation and astronomy attract attention of nanosatellite developers. One of the main requirements for the success of a complex mission is the precision and reliability of satellite's attitude determination and control system. Better pointing accuracy and better stabilization may be achieved by using a star tracker (ST) as a main attitude sensor. Since it's method of operation is based on capturing images of stars, star tracker can provide a pointing accuracy better than 1 angular minute.**

In the last couple of years several laboratories and companies performed huge work on star tracker miniaturization, designing and delivering first prototypes that comply with size, mass and power restrictions of 3U CubeSats. Newly developed miniature star trackers while preserving core functionality are noticeably different compared to existing large-sized star trackers. The differences might be found in their optics, image sensors, algorithms and processing hardware.

Newly developed miniature star trackers have a set of hardware similar to a modern smartphone. At the same time fast improving application program interfaces (API) of smartphone operating systems give developers today a better control over the smartphone internals. It becomes possible to implement a complete star tracker algorithm in a form of a smartphone application. During the tests, even with an expected overhead of OS stack, lost-in-space task was solved in less than one second. That defined the choice of a smartphone as a hardware platform for star tracker performance study.

In this work we analyze the performance of a polestar algorithm for autonomous attitude determination. By changing capture parameters, such as sensitivity, resolution etc. the flaws and bottlenecks of the algorithm are exposed. Subsequently, algorithmic and hardware solutions are proposed to mitigate discovered performance losses.

Keywords—CubeSat; Star Tracker; lost-in-space; smartphone

This study was performed as a part of the project "Development and evaluation of miniature star tracker system for nanosatellite application" financed by the Van Allen Foundation

I. INTRODUCTION

Star trackers have been around for more than 30 years. With the evolution of image sensors, microprocessors and algorithms their characteristics were changing, and their role in the attitude control was changing too.

Since the very first implementations star tracker systems were considered to be a source of the most accurate and precise attitude data for a satellite's ADCS [1]. But they were used almost exclusively alongside with other sensors, such as sun sensors, an infrared Earth sensor and gyroscopes. Not every ST could be used autonomously, some required a priori information about satellite's attitude. Nowadays, it is expected from a ST not only to be precise, but also fast, compact and lightweight. Autonomous attitude determination in lost-in-space mode and angular rate sensing mode have entered the list of desired capabilities [8]. This is why such performance aspects as initial attitude acquisition time and output rate are extremely important for newly developed STs.

II. PLATFORM

A. Hardware

In the latest models of camera phones the imaging system is represented by a camera module, which contains an image sensor, optics with microactuators, and an image processor (ISP) [5]. Tiny size of camera optics often brings various geometrical and color distortions, so image processor plays an important role of fixing those distortions as well as bringing the noise level down to acceptable values [6]. In addition to that by placing the frame scaling, re-coding and transformation routines into the ISP, smartphone manufacturers lower the power consumption and leave more resources of system microcontroller available for user applications (see Fig. 1).

B. Software

For the software part a simple autonomous star identification algorithm for lost-in-space (LIS) mode was implemented, which is in many ways similar to a polestar algorithm described in [2, 3]. Algorithm flow, as it was implemented in the Android application, used for this study, may be described by a sequence of four steps with corresponding subroutines:

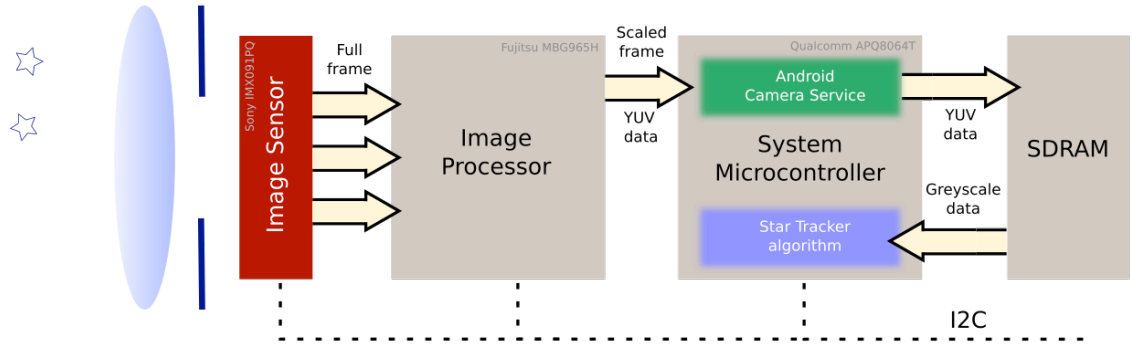

Fig. 1. Hardware structure of and paths of data transfer

- Capture
  - Capture request
  - Integration
  - Readout into memory
- Extract
  - Threshold
  - Clustering
  - Centroid calculation
  - Pattern extraction
- Match
  - Database search
  - Candidate verification
- Calculate
  - Attitude estimation
  - Propagation, residual calculation

III. Experiments

In order to locate the bottleneck within the algorithm a number of tests were carried out using a commercial off-the-shelf smartphone Samsung GT-I9505. First test gave a rough picture of a time distribution between algorithm steps (see Table 1).

TABLE I. TIME DISTRIBUTION BETWEEN STEPS OF ST ALGORITHM

Measurements	Capture	Extract	Match	Calculate
Execution time, %	5%	50%	44%	1%
Spread, %	± 0.1%	± 8%	± 12%	± 0.1%

Capturing images of a real night sky using a smartphone camera, although feasible, requires integration time close to one second [7], so during all tests celestial sphere were simulated on a PC and random regions were displayed on an LCD screen. This ensured that the integration time fixed at $1/14^{th}$ of a second was sufficient to detect 8÷30 spots, required for stable algorithm operation. Delivery of pixel data into the algorithm is performed using smartphone's out-of-the box hardware and software solutions. Android API (since version 21) allows allocation of buffer in system memory, and gives applications access to this buffer for the purpose of further processing [4]. OS camera service introduced in Lollipop and dedicated ISP ensures that image data will be available without any noticeable delay for any of supported image resolutions. All that makes the influence of *capture* step on algorithm execution time very small and predictable. The same may be said about *calculate* step, its duration is negligible and since it uses strictly defined sequence of mathematical operations, it will always complete in the same finite amount of time. With that in mind, subsequent experiments were carried out with the main attention given to *extract* and *match* steps.

Subroutines that take the most time to complete are thresholding and database search (see Fig.2). One way to reduce computation time is to reduce number of pixels to which thresholding is applied. This approach can be easily tested by simply changing ISP's output resolution settings (see Fig.3). Another way to increase the performance is to limit number of spots (possible stars) handled by the algorithm (see Fig.4). But of course spot count will directly affect the success rate of star identification and star tracker sky coverage so this method should be used with caution.

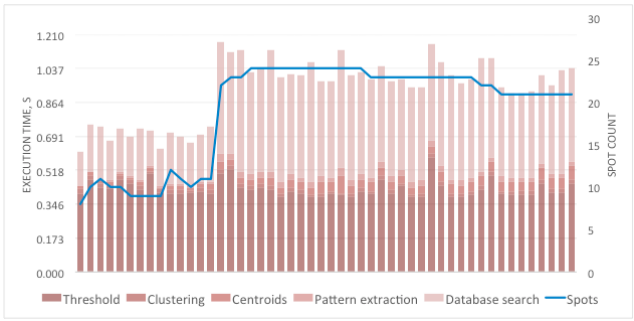

Fig. 2. Algorithm execution time and number of detected spots

A noticeable difference between implemented algorithm and a polestar algorithm described in [2] lies in the match candidate verification subroutine. Our approach was to simply

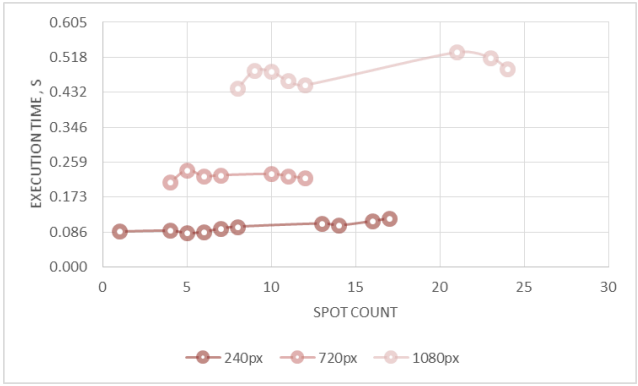

Fig. 3. Pattern extraction time for different resolutions

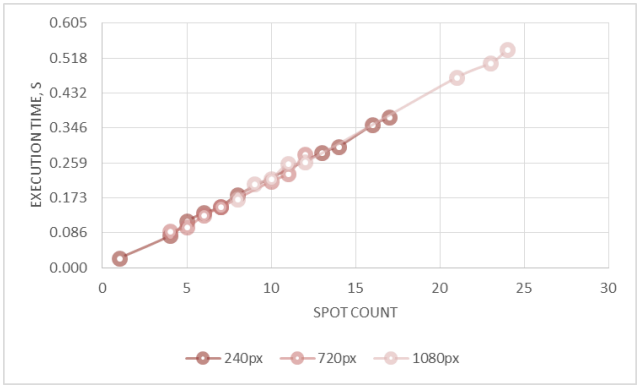

Fig. 4. Execution time of match step for different number of spots

filter the results based on a complex confidence value, calculated with respect to match statistics and database irregularities. In this case verification is reduced to simple sorting routine of 10÷100 match candidate records. In addition to that, as validation subroutine only handles one spot object at a time as opposed to verification based on cross-check, voting or sub-graph search, so verification may be considered as an additional operation in database search subroutine and may be executed for each spot independently. Bearing this in mind, we can further improve the performance if we take advantage of microcontroller's multicore architecture and process the spots in parallel threads, distributed over available microprocessor cores (4 cores in our case). Ideally, since during match step every spot object can be treated separately, algorithm should be executed in N+1 parallel threads, where N is amount of centroids. But even smaller amount of threads will give a good boost in performance to the match step (see Fig.5) cutting up to 62% of execution time (up to 28% for the complete algorithm, running on 1Mpx image with 14 spots).

IV. Discussion

A. Performance improvements and related drawbacks

Tests show that both limiting amount of pixels and limiting amount of spots may help improve execution time of a lost-in-space algorithm. Both approaches may be combined, as they affect different parts of the algorithm. Optimized for multi-threaded execution this implementation can deliver up to 8-10 successive runs per second.

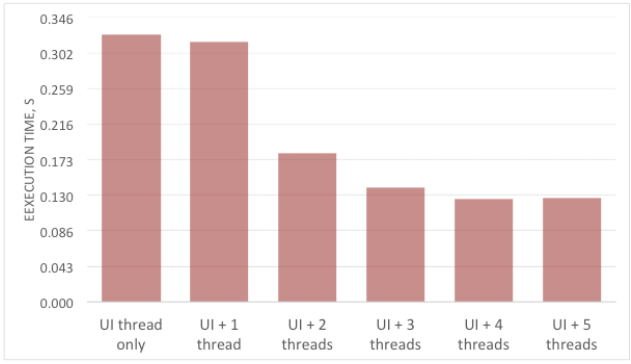

Fig. 5. Execution time of match step for different number of threads

Needless to say, limiting resolution of image or decreasing sensitivity will have side-effects, like worse sky coverage, lower success rate of star identification and increased number of false positives. So in order to balance this either the algorithm or the star database (or both) might have to be adapted. In the end it all depends on ADCS and its ability to cope with noisy measurements.

B. Other influencing factors

One of the aspects that influence matching performance that was intentionally left out of the conversation is the database size and search algorithm. In fact, database size (in records) will depend on magnitude cut-off value, star tracker field of view (FOV), exposure and desired success rate. For this study database was intentionally kept small, just enough to produce a valid 3D Quaternion solution for the most parts of the celestial sphere. Only stars with apparent magnitude below 5.5 were used to create the database. That in most cases is enough for a star tracker with effective FOV around 20˚ so it is for a smartphone, which has a significantly larger field of view. But if we are forced to use smaller FOV and/or we need to identify less bright stars, database size will grow significantly and both search and validation will require significant optimization, which may in turn make processing in parallel no longer possible.

C. Real star sky operation

For this study a sky simulator was used and a LCD screen that produced star images that were much brighter than they would appear on the night sky. That was made to avoid the camera's light sensitivity limitation.

Main reasons for poor light sensitivity of a smartphone camera are small aperture size, presence of color filters and digital noise on higher pixel gain values. Even for a ST based on a smartphone with the most recent sensor, advanced ISPs and the widest aperture on the market, exposure time will be the limiting factor of performance. Still even for integration time around $1/10^{th}$ of a second and relatively safe ISO 1000 there's good probability of capturing two-three bright stars thanks to a wide 52˚ FOV of a smartphone camera. And although that number of spots is clearly not enough to solve lost-in-space task, it may be sufficient for operation in tracking or angular rate sensor modes.

V. Conclusions

In the searching for ready off-the-shelf solutions that could be used as the star tracker development platform a study of a typical modern smartphone (camera phone) has been performed. Both its hardware and software capabilities allowed successful porting of a complete star tracker algorithm to a regular Android smartphone in a form of a user application. After several rounds of optimization, its performance measured in a hardware-in-the-loop laboratory setup proved to be comparable to existing specialized star tracker solutions. And despite of the limitations imposed by miniature imaging system, it can be used as a reference platform for future studies, focused both on software and hardware aspects of star tracker design.

Acknowledgment

I would like to thank Laurent Dusseau and Gilles Sassatelli for the attention they paid to my work, all the ideas and constructive critics expressed during numerous meetings.

References

[1] Liebe, C. C.
Pattern Recognition of Star Constellations for Spacecraft Applications
IEEE AES Magazine, 7, 6 (Jan. 1993), 31

[2] Silani E., Rovera M.
Star Identification Algorithms: Novel Approach & Comparison Study
*IEEE Transactions on Aerospace and Electronic systems, **42**, 4 (Oct. 2006), 1279-1282*

[3] Jonas, M.
Performance Analysis of Mars Express Star Trackers
Luleå University of Technology Department of Computer Science, Electrical and Space Engineering, Master Thesis, Jul. 2011, 68-70

[4] Android's camera Hardware Abstraction Layer (HAL)
Internet: https://source.android.com/devices/camera/index.html, *[May 20, 2015]*

[5] Wu J.J.
Qualcomm Snapdragon 600-based SmartPhone
Internet: http://www.slideshare.net/jjwu6266/qualcomm-snapdragon-600-smartphone, *May 16, 2013 [May 20, 2015]*

[6] Milbeaut image signal processor (ISP)
Internet:
http://www.fujitsu.com/cn/en/products/devices/semiconductor/fsp/imaging/milbeaut/ *[May 20, 2015]*

[7] Nexus 5 astrophotography
Internet: http://imgur.com/a/BXMGu#0 *Dec. 17, 2014 [May 20, 2015]*

[8] J.Enright et al.
Towards star tracker only attitude estimation
24th Annual AIAA/USU Conference on Small Satellites, 2010, 3