Nano Satellite Optical Alignment for Laser Based Communication

Shai Aharon
Ariel University
ifryed@gmail.com

Abstract

From the beginning of time itself, mankind has sought out for communication. What started as camp-fire talks, evolved to smoke signals, for long range communications, to telephones, and finally the World Wide Internet. Today, long range connections are trivial, so trivial that one cannot imagine a life without it. The ability for long range communication is due to arrays of relay stations and satellites who use radio waves. Radio is a very efficient way to transmit and receive messages for short and long distances, but it's bandwidth is limited on low-energy devices. The limit is neglectable on ground stations or regular-sized satellites, but when dealing with Nano-Satellites, with limited power and size, it is noticeable. Current nano-satellites usually use the LoRA protocol, which can transmit packages up to 50 bytes each, which is enough for general information, and basic orders, but cannot send videos of even high-resolution images, or upload whole scripts for the satellite to execute.
Using Laser instead of LoRA, can expand the bandwidth significantly. The only problem is, while LoRA transmits in all directions, laser based communications needs the emitter and receiver to be pointed directly at each other. In order to align the satellite with the ground station, we came up with an optical base solution for following reasons a) low cost b) light weight c) a camera is already based on the satellite for other tasks d) high accuracy.

All of our code is available at https://github.com/ifryed/SatAlign.

1. Introduction

In recent years, the user of small satellites (AKA NanoSat) has been growing rapidly. Satellites, which were once a big and complicated systems, which required well established companies or government agencies to design and manufacture, can be now made by small talent full labs [2]. Today, a satellite the size of half a shoe box, can be manufactured using off-the-shelf consumer products and have a wide range of capabilities such as communications, self-tracking, imagery and much more.

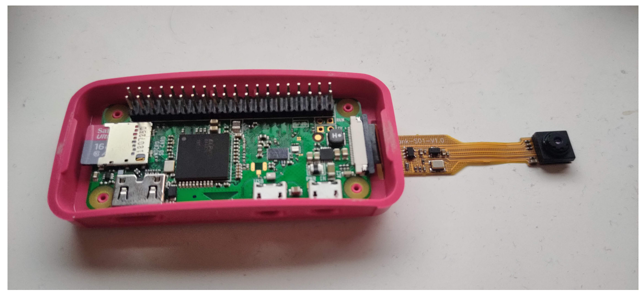

Figure 1. RaspberryPi Zero W module, with RaspberryPi Camera Module v1. This setup will be on the satellite as-is except the casing.

With all of the Nano-Satellites' advancements, the communication is still basic. Currently, due to their size, they need to use small and energy efficient means for communication. Most commonly used is LoRa [6], which stands for **Lo**ng **Ra**nge communication. LoRa has the advantage of transmitting and receiving using Radio Frequency (RF) signals over long distances, such that are suitable for space flight, while maintaining a low-energy consumption and a small size. The down side of LoRa is it's bandwidth, which can usually send up to 50 bytes per packet. This limitation is not noticeable for getting status updates or sending short commands to the satellite, but transmitting an image over LoRa, even a small as 16K will have to use 320 packets, which is not feasible, especially when you consider lost packets and narrow time window where the satellite is in transmitting range. Earlier this year, [5] tried to overcome the later issue by splitting the image frequencies into several images, compressing them, and sending them one by one. This allowed to receive the image in stages, where at each stage you would get the whole image, but range of frequencies starting with the lower ones to the highest. Another use for broad bandwidth will be to transmit code so the satellite could run updated missions and more.
Due too space and energy restrains, we cannot load a stronger RF transmitter on the satellite, instead, we switched the RF for laser based communications for high-bandwidth tasks. The idea is to transmit data using a laser

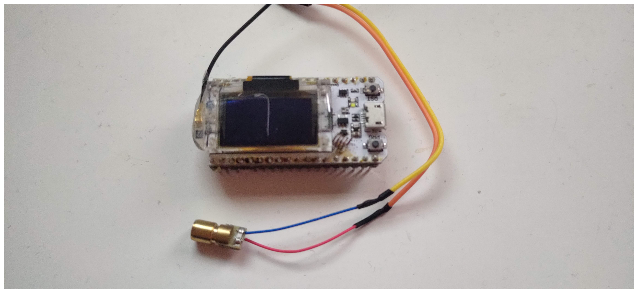

Figure 2. ESP32, with a laser module attached. The laser used in the experiment is weaker then the GS laser in order to simulate weak signal

that "blinks" in different frequencies. Laser can transmit more data then RF, but on the other hand, the emitter and the receiver, unlike RF, need to be aligned, due to the lasers directional quality. To do so, we propose the following protocol:

Algorithm 1 Satellite Laser communication procedure

1. Once the satellite is in range, the ground station (GS) will send a signal, using RF, telling the satellite to emit a blinking light
2. The GS, will lock on the satellite
3. The GS will send an RF signal to the satellite, notifying that it is tracking it
4. GS will point a laser to the satellite, and emit a blinking light at a agreed upon frequency
5. The satellite will then lock on the GS and send a signal notifying when ready
6. Once both are locked on each other, they can start communicating using laser frequency (LF)

In this paper, we focus on the alignment stage of the satellite (Stage 5). Specifically, the detection of the GS, we will treat the the satellite rotations as a given ability.

2. Our Approach

Aligning the satellite requires us to detect the GS light, in order to notice the light, the whole procedure would take place at night. Due to the fact that even at night, there is light-pollution from common house holds etc, we need a method of distinguishing them from the GS laser. In order to make the GS laser standout, except from using a strong laser, the laser will blink in an agreed-upon frequency thus enabling us to filter out other lights.

The idea of the algorithm, Alg. 2, is to sample (capture images) at twice the rate of the laser blinking frequency following the *Nyquist–Shannon sampling theorem* [7]. At line 6, we calculate the difference between to frames, and average them difference per pixel at line 7. Since each consecutive frames are taken at half the lasers *frames-per-seconds* (FPS) apart, we expect the difference to be noticeable. The laser is set to blink at 2 FPS, thus making the satellites capturing speed 4 FPS. Due to hardware limitations, the buffer is set to hold 3 images at a time as calculating step 6 would take too much time, and would not be able to maintain a 4 FPS loop. For that same reason, we did not take into account the lasers color, since capturing a color image would have slowed the process even more, which would have forced us to use a slower blink, and the refresh rate would have increased significantly.

Algorithm 2 Blinking Light Detector

1: $buffer \leftarrow [n\ empty\ frames]$
2: $cond \leftarrow inf$
3: **while** $conf > thres$ **do**
4: $\quad img \leftarrow CapureNewImage()$
5: $\quad buffer[i \mod n] = img$
6: $\quad diff = |buffer[1:] - buffer[:n-1]|$
7: $\quad \overline{diff} = \frac{1}{n}\sum_i^{n-1} diff[i]$
8: $\quad X, Y \leftarrow where(\overline{diff} > 0)$
9: $\quad \overline{X}, \overline{Y} \leftarrow \frac{1}{|X|}\Sigma X, \frac{1}{|Y|}\Sigma Y$
10: $\quad X_M, Y_M \leftarrow calcMoment(X, Y)$
11: $\quad conf \leftarrow calcConf(X, Y, X_M, Y_M)$
12: $\quad$ Wait for $\frac{1}{2FREQ}$ seconds
13: return $\overline{X}, \overline{Y}$

The $conf$ parameter, was used to indicate how confident the algorithm is that it found the correct location. It is calculated as seen in Eq. 1 with respect to the variance and the previous location in time.

$$conf(X, Y, X_M, Y_M) = var(X) + var(Y) + X_M^2 + Y_M^2 \tag{1}$$

To check the detection's consistence in time, we used a *ghost* location, which was updated each iteration using gradient momentum as seen in Eq. 2.

$$\begin{aligned} i &\in \{X, Y\} \\ i_M &= \alpha_M i_M - \alpha(\overline{i_t} - \overline{i}) \\ \overline{i_t} &= \overline{i_t} + i_M \\ \alpha_M &= 0.5 \\ \alpha &= 0.8 \end{aligned} \tag{2}$$

$\overline{i_t}$ is the *ghost* location, which should be more stable in time, and will not jitter to much. The momentum, can be viewed as the variance of the location in time, thus making it a good candidate to measure our confidence in our current location.

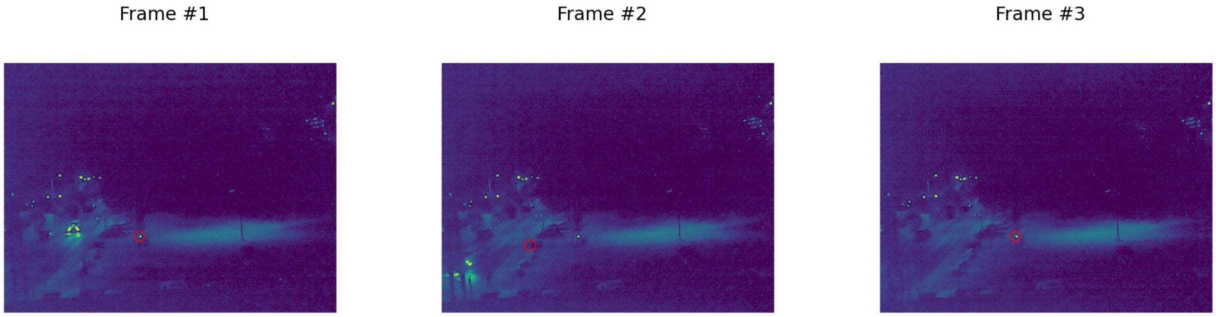

Figure 3. A few frames from the experiment. On frame #2, the detection was set off by a passing car, but at the next frame, it corrected itself. The frames are saved at intervals of 2.5 seconds.

3. Experiments

3.1. Hardware

Due to the small nature of *Nano-Satellites*, we had to consider a lean assembly of hardware that can still carryout the task.

On satellite Fig. 1:

1. RaspberryPi Zero W [4]
2. RasbperryPi Camera Module v1 [3]

GS hardware (simulated):

1. ESP32 Arduino board [1] Fig. 2
2. Arduino laser module
3. Generic battery pack

3.2. Setup

In order to test the performance, we set up an experiment to test the algorithm and hardware, in an environment as close as possible to space flight, without lunching the satellite into space. The setup consists of two parts, the mock GS (M-GS) and the mock satellite (M-S). The M-GS was stationed approximately 40 meters apart. The experiment took place at night, in medium lighting conditions (street lamps and some light emitting from residential houses). The M-GS was placed in a relative dark location, see Fig. 6,4.

3.3. Results

Our experiments were showed that our algorithm could handle a real life scenario quite well. Even though we our experiment included much more "noise" (passing cars etc.) then our satellite should encounter, it could find the M-GS in a matter of seconds, and was able to self-evaluate when it had quality readings. As seen in Fig. 3, passing cars, or the movement of trees, would occasionally set off the detection, but the algorithm responded with a high confidence error, and corrected the detection almost instantly.

Figure 4. The M-GS, the laser module is taped to the top of the wood.

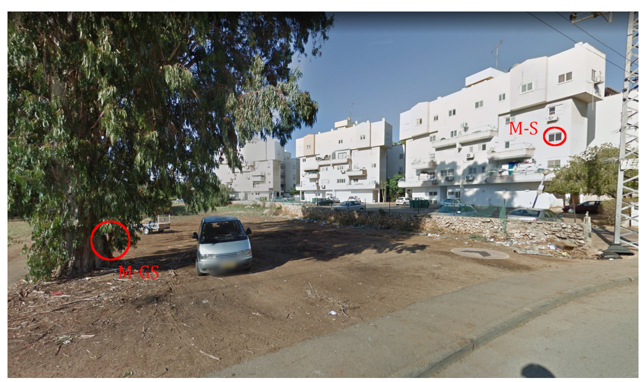

Figure 5. A wide shot of the setup scene. Screen shot using Google Earth.

4. Conclusion

Our method showed remarkable performance, despite the hardware limitations. We could add stabilization by in-

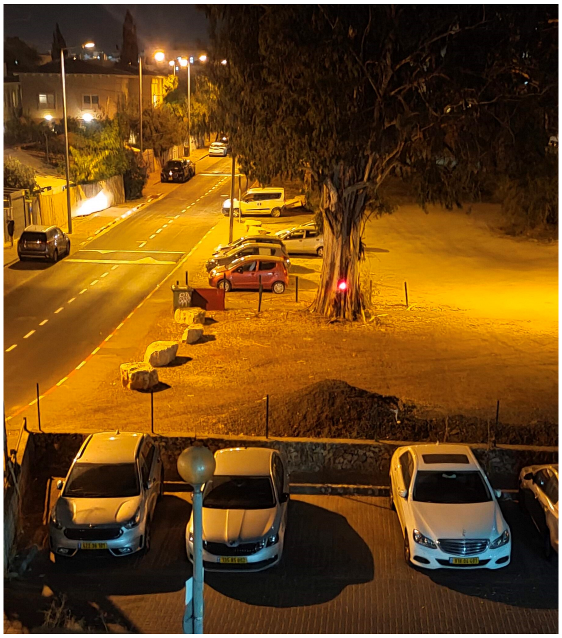

Figure 6. The point-of-view of that satellite, taken using an external camera for better quality

creasing the buffer size, but in order to maintain the FPS, we would have to down sample the image, thus costing us the detection accuracy. In the future, we expect that stronger embedded systems will be developed, thus allowing the buffer size to increase without it harming the accuracy.

References

[1] ESP esp32. https://www.espressif.com/en/products/socs/esp32. 3

[2] KCG kinematic and computational geometry lab. https://www.ariel-asc.com/. 1

[3] RSPICAM raspberry camera v1. https://www.geeetech.com/wiki/index.php/Raspberry_Pi_Camera_Module. 3

[4] RSPIZERO raspberrypi zero. https://www.raspberrypi.org/products/raspberry-pi-zero-w/. 3

[5] SIT satellite image transfer. https://github.com/ifryed/SatImageTransfer/. 1

[6] Dmitry Bankov, Evgeny Khorov, and Andrey Lyakhov. On the limits of lorawan channel access, 2016. 1

[7] C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, jan 1949. 2