Hindawi
International Journal of Antennas and Propagation
Volume 2018, Article ID 3748623, 12 pages
https://doi.org/10.1155/2018/3748623

Research Article

Accurate 3D Mapping Algorithm for Flexible Antennas

Saed Asaly,[1] Boaz Ben-Moshe,[1] and Nir Shvalb[2]

[1]*Department of Computer Science, Aerospace and Nano-Satellite Research Center, K&CG Lab, Ariel University, Ariel, Israel*
[2]*Department of Industrial Engineering, Aerospace and Nano-Satellite Research Center, K&CG Lab, Ariel University, Ariel, Israel*

Correspondence should be addressed to Boaz Ben-Moshe; benmo@g.ariel.ac.il

Received 24 August 2017; Revised 21 December 2017; Accepted 9 January 2018; Published 20 March 2018

Academic Editor: Xuejun Lu

This work addresses the problem of performing an accurate 3D mapping of a flexible antenna surface. Consider a high-gain satellite flexible antenna; even a submillimeter change in the antenna surface may lead to a considerable loss in the antenna gain. Using a robotic subreflector, such changes can be compensated for. Yet, in order to perform such tuning, an accurate 3D mapping of the main antenna is required. This paper presents a general method for performing an accurate 3D mapping of marked surfaces such as satellite dish antennas. Motivated by the novel technology for nanosatellites with flexible high-gain antennas, we propose a new accurate mapping framework which requires a small-sized monocamera and known patterns on the antenna surface. The experimental result shows that the presented mapping method can detect changes up to 0.1-millimeter accuracy, while the camera is located 1 meter away from the dish, allowing an RF antenna optimization for Ka and Ku frequencies. Such optimization process can improve the gain of the flexible antennas and allow an adaptive beam shaping. The presented method is currently being implemented on a nanosatellite which is scheduled to be launched at the end of 2018.

1. Introduction

The vision of having a reliable and affordable global network which can be accessed from any point on the globe at any time is a huge scientific challenge which has attracted many researches during the last few decades. Most proposed solutions are based on a network of hundreds or thousands of LEO nanosatellites which will constitute a global network with the earth via RF communication. These *new-space* projects are of interest to major companies such as Google, Qualcomm, Facebook, and SpaceX. *OneWeb* is an example of such a project involving a large constellation of LEO satellites. Other projects such as Google's *Project Loon* [1] or Facebook's *Aquila Drone* are not directly focused on satellite constellations but generally assume that such global network already exists. The new-space industry includes many small- or medium-size companies which develop products for the new-space market (e.g., *Planet Labs* and *Spire* are focusing on global imaging [2] and global *IoT*). One of the most famous LEO satellite constellations is the *Iridium* network, developed in the '90s; this global network is still operational, and the second-generation network named *Iridium Next* is currently being deployed. Optimizing a global network in terms of coverage, deployment, and services involves extremely complicated problems from the computational point of view. In order to reduce the cost of deploying such network, many new-space companies are working on miniaturizing their satellites—as launching 100 LEO nanosatellites often costs less than launching a single large satellite into a geosynchronous orbit. In order to allow a long-range, wide-band RF communication between a satellite and a ground station, high-gain directional antennas are being used. Having such a dish antenna on-board of the satellite significantly increases its size and weight, and

therefore, almost all current nanosatellites have a limited bandwidth as they use small low-gain antennas allowing a bandwidth of sub-Mbps. *NSLComm* has developed a concept of nanosatellite with a relatively large expendable antenna, allowing a significantly better link budget from a nanosatellite [3]. Nevertheless, flexible antennas are sensitive to surface distortion especially in space, where significant temperature changes are common. In this paper, we present a generic method to accurately map the surface of a flexible antenna located on a satellite. The presented framework requires very limited space and computing power, allowing it to be implemented even for small nanosatellites.

2. Related Works

Mapping a 3D surface is an important problem which is of interest to many researches. Available literature suggests solutions for wide-range mapping techniques including time of flight [4], triangulation [5], structured light [6], RGBD [7], stereo vision [8], and image-based modeling [9].

In this work, we focus on the challenging task of mapping a satellite flexible antenna—which is not suitable for common 3D scanning techniques due to space limitations and the need to perform a 3D scan from a fixed and single angle (i.e., a single image). The ability to infer a 3D model of an object from a single image is necessary for human-level scene understanding. Tatarchenko et al. [10] have presented a convolution network capable of inferring a 3D representation of a previously unseen object given a single image of this object, while in the work of Williams et al. [11], a graph theory and dynamic programming techniques over the shape constraints were presented to compute the anterior and posterior surfaces in individual 2D images. Tanskanen et al. [12] have proposed a complete on-device 3D reconstruction pipeline for mobile monocular hand-held devices, which generates dense 3D models with an absolute scale on-site while simultaneously supplying the user with real-time interactive feedback. Medina et al. [13] suggest a resistor-based 2D shape sensor, and Shvalb et al. [3] show that, using a robotic flexible subreflector, even relatively significant changes in a dish surface can be fixed; naturally, having a 3D model of the current surface of the main dish antenna can improve the accuracy and the run time of such systems.

2.1. Our Contribution. In this work, we present a novel method which can robustly recover a surface shape from a single image with known markers (with known shape). The suggested method uses a set of visual markers in order to compute a pointcloud. To the best of our knowledge, this is the first work which presents a framework for performing 3D reconstruction of smooth surfaces with submillimeter accuracy that is applicable for on-board satellite flexible antenna.

3. Flexible Antenna for Nanosatellites

The general concept of the flexible antenna with an adjustable robotic subreflector was presented recently [3]. It is based on a flexible expandable main reflector and an adjustable robotic subreflector which can compensate for minor changes in the main reflector surface. Mechanical mechanisms for manipulating the robotic subreflector may be based on linear servo or piezoelectric motors [3] but can also be based on bioinspired manipulators (see [14]). In order to optimize high-frequency RF communication (e.g., Ka bands), the main antenna should be mapped with an accuracy level which is 25–50 times higher than the communication typical wavelength (about 1 cm in Ka), leading to a challenging mapping accuracy requirement of about 0.2–0.1 mm on average (see [15]). A nanosatellite with such flexible antenna should also be equipped with the following components: (i) a global position receiver (e.g., GPS), (ii) a star tracker in order to determine its orientation, and (iii) an altitude control mechanism (based on both reaction/momentum wheels and a magnetorquer). Using the above components, the satellite on-board computer can aim the antenna to a specific region on earth—in general, this process resembles the task of the imaging satellite that needs to aim its camera to a given region. Denote that for a LEO satellite, such process is a continuous (always on) process, unlike the case of geosynchronous satellites, which only need to maintain a fixed orientation.

The use of flexible antennas for space applications is a relatively new concept. Having an on-board accurate mapping system for the flexible antenna will allow two major benefits:

(1) A fast and accurate tuning of the robotic subreflector to compensate for the distortion of the main reflector and an adaptive beam-shaping capability of the transmitted pattern.

(2) A study of the changes in the flexible surface with respect to temperature and time.

Due to space and weight limitations, the on-board 3D mapping system should be as compact as possible. Moreover, the method should use limited computing power for on-board algorithms or limited bandwidth methods for ground-based algorithms. Following these requirements, we shall use a monocamera and known shape targets for the mapping task.

4. Monocamera Mapping Algorithm

In order to map the 3-dimensional pointcloud of the satellite antenna, we first embed a set of *targets* (or markers) with a known shape and size. A single camera is assumed to be located near the antenna focal point. We shall now present the general algorithm which analyzes the acquired image to compute the 3D surface of the dish. This process consists of the following stages: (i) camera calibration, (ii) initial pointcloud computation, and (iii) global adjustment.

4.1. Camera Calibration. We start by calibrating the camera using an algorithm proposed by Zhang in [16]. Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. Intrinsic parameters deal

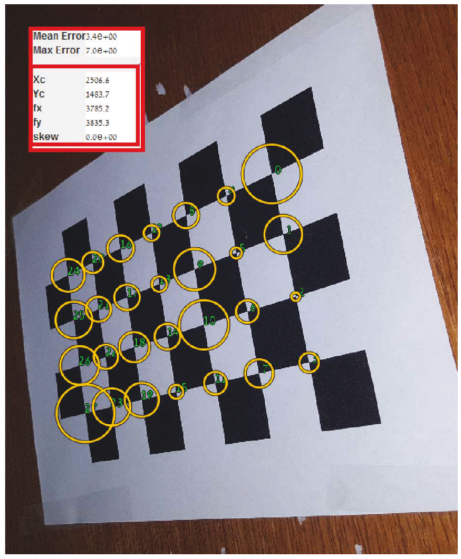

Figure 1: An example of an image (after the calibration process). The intrinsic parameters are marked in red, and the expected accuracy (error) is marked in yellow circles (a larger circle implies a larger expected error).

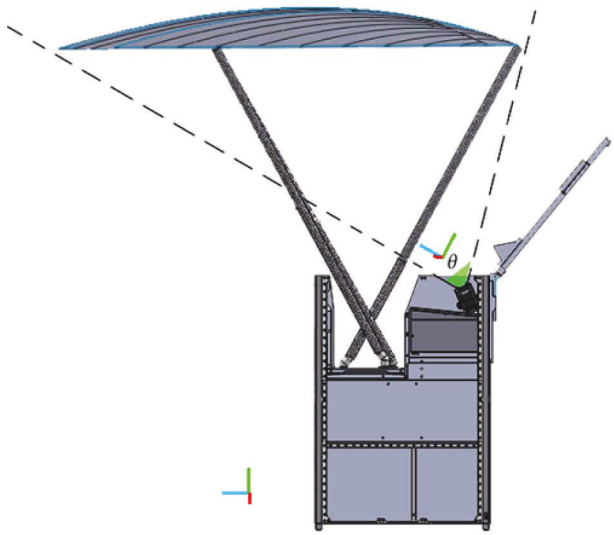

Figure 2: The camera location on-board; here, θ is the field of view which will be considered as $\in[60, 90]$. Note the difference between the coordinate systems of each satellite and camera.

with the camera's internal characteristics, such as its focal length, skewness, distortion, and image center. The camera calibration step is essential for a 3D computer vision, as it allows one to estimate the scene's structure in Euclidean space removing lens distortions, which degrades accuracy. Figure 1 depicts an image taken after a calibration process.

Figure 2 illustrates the position of the camera on the satellite which allows one to view the whole antenna span.

Input: Undistorted image (frame) F.
Output: 3*D* **pointcloud.**
1: Let $im_t = T(im)$.
2: $T \leftarrow Segment(F)$
3: **for each** triplet $t_i \in T$ **do** Compute a 3*D* point: $p_i \in P$ as follows:
(1) The x,y coordinates of p_i are the center values of t_i
(2) $\widehat{n}_i \leftarrow$ the normal of the target.
(3) $\Delta\alpha_i \leftarrow$ the angular difference between $\widehat{n}_i$ and the vector to t_i.
(4) Let $p_i.z = \sqrt{t_i.area} \cdot C_{camera}/\cos \Delta\alpha_i$
4: **end for**

Algorithm 1: Initial 3D mapping using circular targets.

Accordingly, the camera's FoV (field of view) should be chosen to be in the range of 60° and 90°. Such a relatively large FoV imposes a nonnegligible camera distortion. Thus, the calibration process is necessary to allow an accurate angular transformation between the camera coordinate system (i.e., pixel position) and the satellite global coordinate system.

Often, one would also like to express the position of points (x, y, and z) given in the camera coordinate system in a world (satellite) coordinate system. This may be done by simply rotating the set of points P about the angle of inclination of the camera (see Figure 2).

4.2. Initial Pointcloud Generator Algorithm. We start our discussion considering circular targets. Algorithm 1 produces the initial pointcloud which we further use in this paper. Here, the function $T(im)$ uses the information from the calibration step to remove lens distortion from the image. C_{camera} is the angular resolution (taken from the camera parameters). We mark by $Segment(F)$ the function that segments the acquired image and detect the targets T and compute the triplet (center, area, and geometry) for each target. In order to compute $\Delta\alpha$ for each target T, consider two of its vertices w_1 and w_2 and do the following:

(1) Calculate the normal of the surface that the target lies on—for each pattern, the manner in which the normal is calculated is different (we shall discuss this below).

(2) For each pair $w_1, w_2 \in T$, do the following:

(a) Consider the plane that passes through the camera point (the origin point) and points w_1 and w_2. Define line l_1 as the intersection between this plane and the plane that the target lies on.

(b) Let l_2 be the line connecting the camera and the midpoint between w_1 and w_2.

(3) Set α to be the angle between l_1 and l_2 (see Figure 3).

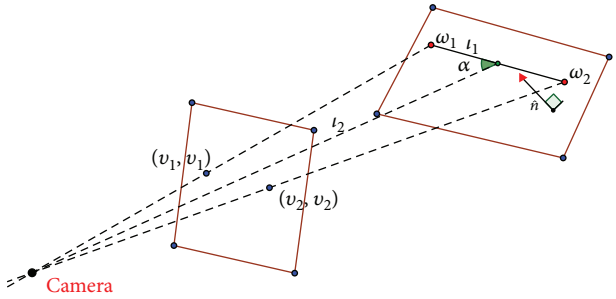

FIGURE 3: An illustration of the information needed to find $\Delta\alpha$, where α is the green angle.

(4) Define the angular difference as $90^\circ - \text{avg}(\alpha_i)$.

We implement the algorithm above for multicircular targets. The use of such targets is motivated by the ability of both preventing the pixel snapping problem—allowing a subpixel resolution accuracy of the center of the target and the accuracy of the normal of the plane the target is lying on. The advantages of using circles therefore contribute to the overall calculated accuracy of the Z dimension.

Figure 4 depicts the multicircular target cropped from an acquired image; note that each circle is distorted to be an ellipse rather than a circle as a result of the varying orientations and distances. The following explains how one computes the pointcloud using multicircular targets:

(1) We apply an ellipse detector algorithm which uses a nonlinear pattern (connected component) in the binarized image. Next, the estimation is refined by using a subpixel resolution algorithm in the grayscale image. We detect both outer and inner ellipses; then, for each pair of ellipses, we find the average center, which will be more robust to varying light intensity conditions which could cause a pixel-snapping problem (i.e., a pixel in one image can deviate by a single pixel in another image with the same conditions). In Figure 5, an example of the ellipse detector algorithm result is shown.

(2) For each target, calculate the center C_p by running the *K-means clustering* algorithm on its ellipse centers (in Figure 6 shown as an example of the K-means result).

(3) Let the x,y coordinates of the target in the pointcloud be the x, y coordinates of C_p.

(4) Find the normal of the ellipse as follows: Consider the largest ellipse in the target, find its $\max(a, b)$, where a is the major axis and b is the minor axis, and its intersections with the ellipse, and let them be p_1 and p_2, respectively.

Assume that the camera view is on the yz plane; then,

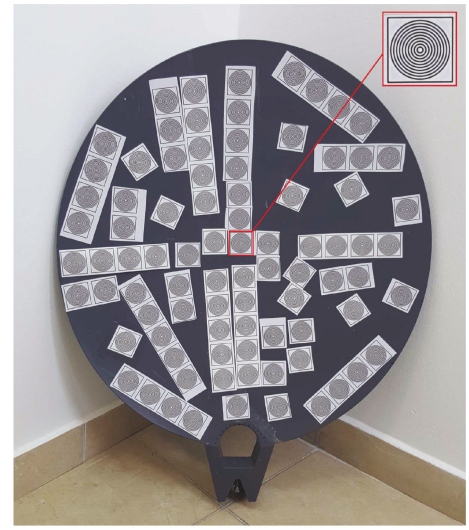

FIGURE 4: The multicircular targets on a solid main reflector as acquired by the monocamera.

(a) $p_1 = (x_1, u_1, v_1)$, where x_1 is unknown and u_1 and v_1 are the x, y coordinates for the intersection point; in the same way, let $p_2 = (x_2, u_2, v_2)$.

(b) Then, the center point $p_c = (0, u_c, v_c)$ (since we have no depth information, we arbitrarily place the yz plane at the center of the circle).

(c) Now, we must have $\|p_1 - p_c\|^2 = r^2 = x_1^2 + (u_1 - u_c)^2 + (v_1 - v_c)^2$ from which we can calculate the absolute value of x_1 and similarly $\|p_2 - p_c\|^2 = r^2 = x_2^2 + (u_2 - u_c)^2 + (v_2 - v_c)^2$ from which we can calculate the absolute value of x_2.

(d) Note that one needs to determine the signs of x_1 and x_2. If we choose p_1, p_2 so that they are opposite (in the sense that they have inverse coordinates about the center $u_1 - u_c = u_c - u_2, v_1 - v_c = v_c - v_2$), then x_1 and x_2 should have the same absolute value with opposite signs. Since we can correctly determine whether x_1 and x_2 are closer to the camera than the center or further away, this could be easily set.

(e) Let the vectors $\mathbf{v}_1' = (x_1, u_1 - u_c, v_1 - v_c)$ and $\mathbf{v}_2' = (x_2, u_2 - u_c, v_2 - v_c)$; then, $\hat{\mathbf{n}} = \mathbf{v}_1' \times \mathbf{v}_2'$, with the $\hat{\mathbf{n}}$ vector is normal to the ellipse.

(f) Then, compute the angular difference Δa_i and *Z-value* as we mentioned above.

As will be exemplified below, using circular targets enables an average accuracy level of below 0.1 mm. Yet, such a method requires high-quality printing of curved lines over a flexible antenna made of Kapton foil. Such printing is hard to perform with space-qualified ink. Therefore, we needed to adjust the algorithm to work with targets which are composed of just straight lines.

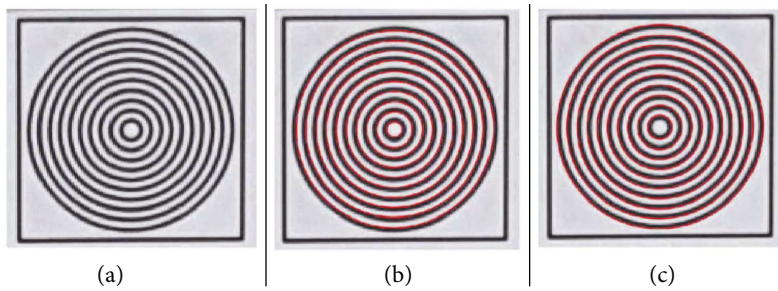

Figure 5: The result of the ellipse detector is the target itself (a), shows the inner ellipses detected (b), and shows the outer ellipses detected (c).

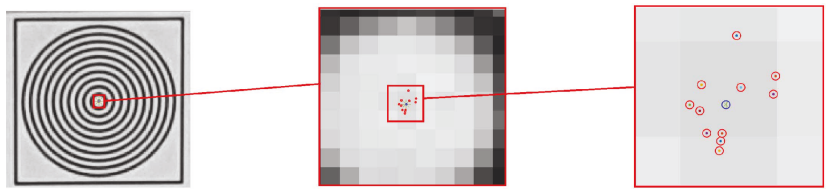

Figure 6: An example of the K-means clustering (KMC) result; the dotted points on the right are the centers of the ellipses, and the starred ones (blue) are the KMC result.

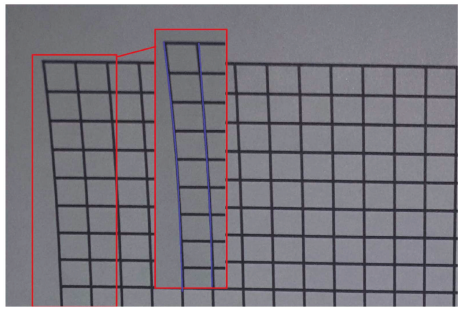

Figure 7: The grid pattern printed on a paper. Note that the blue lines are a bit curved.

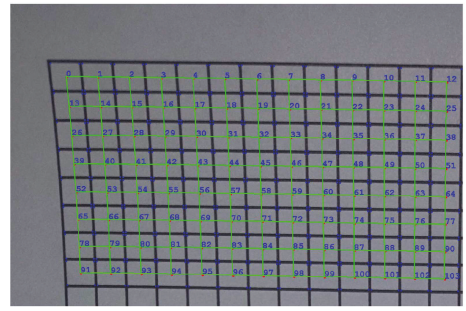

Figure 8: Computing Level_0 and Level_1 from a grid-based image: Level_0 is presented as the corners of the green grid. Level_1 is marked as blue dots in the center of each green square.

4.3. Mapping Using a Uniform Grid. After testing a wide range of possible straight-line patterns, we conclude that a simple uniform grid (see Figure 7) is the most suitable target available on the actual surface of the flexible antenna. At first, we have tested algorithms for detecting lines using edge detection methods. Such approach leads to relatively poor results as the edges on the antenna as captured by the camera (see Figure 7) are not straight lines but rather complicated curves. Performing regression to such curves introduced significant errors. Thus, we examined an alternative methodology where we first detect all the inner corners of each square (using regression to square). We then define Level_0 to be the set of all center points of each square and Level_1 to be the set of the centers of the unit squares implied by the points in Level_0 (see Figure 8). Algorithm 2 computes a 3D pointcloud from an image of a grid-based target using the notion of the Level_1 point set.

In order to implement the above algorithm, the following properties should be defined:

(i) Let S be the set of all small squares in L_1: including unit squares, two-unit squares, and up to some relatively small number of units—usually smaller than the 10-degree angle.

(ii) Given a square s_i, its area, and normal—one can approximate a distance of s_i from the camera, where the center of s_i is the angular coordinate of p_i. This is actually the same method which was used in the circular target of Algorithm 1.

(iii) In some implementations, p_i can be generalized to be a weighted point associated with a confidence of the distance approximation based on s_i, a_i, and n_i. That is, the expected distance accuracy to a two-unit

Input: Undistorted image (frame) F.
Output: 3D **pointcloud.**
1: Let $P^* \leftarrow \emptyset$
2: Let C_0 be a set of all points on corners of the grid.
3: Compute $Level_1$ (L_1) from F and C_0.
4: Let $S \leftarrow$ be all small squares in L_1
5: **for each** $s_i \in S$ **do**
(1) let $a_i \leftarrow$ be the area of s_i.
(2) let $n_i \leftarrow$ be the normal of s_i.
(3) let $p_i \leftarrow$ be a 3D point associated with s_i w.r.t. a_i, n_i.
(4) add p_i to P^*
6: **end for**
7: Return P^*

Algorithm 2: Initial 3D mapping using a grid.

Input: Undistorted reference image (P), and current image (I^*).
Output: 3D **pointcloud.**
1: Compute P^* (set of centers)from I^* (algo 4.3).
2: Perform a 2D registration P' between P and P^*.
3: **for each** pair $w_i, w_j \in P'$ **do**
(1) Compute it's ratio, $r = d(p^*)/d(p')/[1-\varepsilon, 1+\varepsilon]$
(2) Associate r with $RatioMap(RM) \leftarrow mid(P')$.
4: **end for**
5: Given RM, perform a *confidence*&*normal* analysis (fine tuning).
6: Call Minimum LAR algorithm with an input RM.

Algorithm 3: Computing the difference map between reference image and current image.

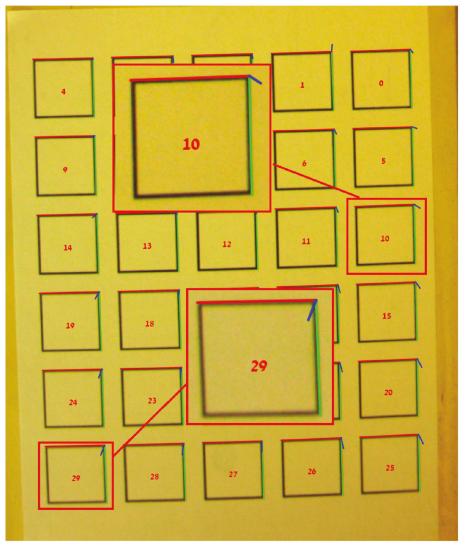

Figure 9: An example of normal detection results on some printed squares; note that target 10 and target 29 are oriented differently with different computed normal vectors.

square is usually better than the expected distance accuracy to a single-unit square.

(iv) Computing the normal of s_i can be performed using the *EPnP* algorithm [17] (see Figure 9).

The grid-based algorithm is relatively robust and simple to implement. Yet, in most cases, the average error level was too large, about 0.3–0.5 mm. Moreover, the manufacturing limitation of the flexible antenna requires us to design an "on-board" algorithm which is both accurate and feasible.

5. On-Board Satellite Implementation

In this section, we define the actual "space algorithm" for computing the 3D surface of a flexible antenna. Algorithm 3 compares two images: a reference image (P) and a current image (I). P is the optimal ("perfect") lab image of the flexible antenna from the satellite camera. This image is taken during an RF test of the complete satellite. I is the "space" image which is compared with P. Algorithm 3 computes the 3D difference map between P and I instead of the actual 3D pointcloud—as the 3D surface of P was mapped in high accuracy level during the final testing stage. As the satellite is about to be launched, its flexible antenna is unfolded and the surface of the main (flexible) antenna may suffer from global distortions due to the flexible nature of the antenna. In order to overcome such global distortions, we decided to use two different coordinate systems: satellite coordinate system and antenna coordinate system. For each target center (in the image 2D point), we consider it in the antenna coordinate system and then we consider its relative position in the satellite coordinate system. In order to determine the place of the target in the antenna coordinate system, we detect the contour of the antenna; then, for each point, we consider its relative position to the contour (edge). Figure 10 depicts the contour detection step flowchart. Having the 2D points in the antenna coordinate systems, we use Algorithm 1 to map the antenna surface.

5.1. 3D Optimized Surface Generation Algorithm. We define two levels of noise filtering: $level_0$, which uses direct position measurements for estimation (e.g., estimation of a target's center) and $level_1$, which averages out $level_0$ estimations (e.g., the center of neighboring target's centers). We define $level_i$ in the same iterative manner.

5.1.1. Minimum LAR Algorithm. After we have found the 3D pointcloud by using the algorithm above, we now find the minimum RMS function which returns a best fitting surface for the 3D points.

Fitting requires a parametric model that relates the response data to the predictor data with one or more coefficients. The result of the fitting process is an estimate of the model coefficients.

The following algorithm returns the best fitting plane for a given 3D pointcloud with a least absolute residual (LAR)

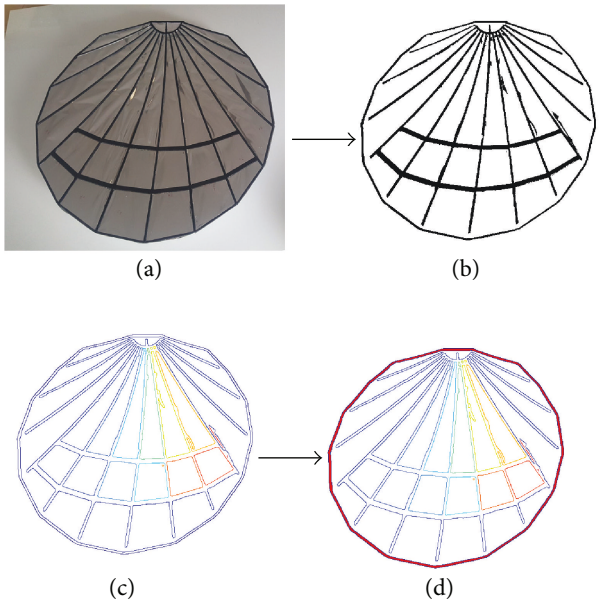

Figure 10: An example of the flowchart of the dish edge detection process. (a) The acquired image is binarized. (b) The intersections between the ribs of the dish with the dish are "cleaned." (c) Detecting the connected components in the binarized image is exemplified. (d) The edge of the dish is then located by finding the resultant convex hull.

Figure 11: The camera calibration test setup. Using a calibrated 3D printer with a robotic moving panel, we were able to calibrate the Z approximation up to the 0.4 mm accuracy level.

Input: a 3*D* **pointcloud** (*P*).
Output: A best fitting 3*D* **surface** *LAR* **optimized.**
1: Fit the model by weighted least squares.
2: **for each** $p_i \in P$ **do**
(1) Let $r_{adj} \leftarrow ri/\sqrt{1-h_i}$.
(2) Let $w_i = \begin{cases} (1-(u_i)^2)^2, & \text{if } |u_i| < 1, \\ 0, & \text{if } |u_i| \geq 1. \end{cases}$
3: If the fit converges, exit. Otherwise, repeat.
4: **end for**

Algorithm 4: Minimum LAR 3D surface algorithm.

surface optimization in order to increase the expected z-accuracy to below 0.1 mm.

Here, r_i is the usual least-squares residuals and h_i is the leverage that adjusts the residuals by reducing the weight of high-leverage data points, which have a large effect on the least-squares fit. The standardized adjusted residuals are given by $u = r_{adj}/K_s$, where K is a tuning constant and s is the robust variance given by MAD/c, in which c is the constant and MAD is the median absolute deviation of the residuals.

6. Experimental Results

In this section, we show the experimental results of each step of the proposed algorithm.

6.1. Camera Calibration Step. For the setup step, we positioned the calibration targets (chessboard) on a 3D printer plate that has movement accuracy of sub-0.1 mm (up/down). A camera fixed at an 80 cm distance from the plate as shown in Figure 11 was located. Then, the plate was translated up and down in 0.1 mm steps. By comparing the translation with a naive distance calculation for the movement, detection calibration was ratified.

6.2. Normal Detection Accuracy Test. In this step, we gave the normal detector Algorithm 4 points with known real normal and ran it. For Figure 12, Table 1 lists some numeric results of the algorithm.

The result above shows an average angular error of ~0.4°. Assuming that the monocamera is located close to the subreflector, such angular "noise" induces only minor errors which are commonly smaller than the 10^{-4} ratio.

We have built a practical setting that provides an accurate movement of a rigid body (plate) that can obtain a page with printed shapes (targets). In Figure 13, we show the setting with an explanation of its components. Two types of cameras were used: (1) an embedded 14-megapixel sensor with a FoV of about 75° and (2) an Android phone with 16 megapixels (Galaxy S6) with a FoV

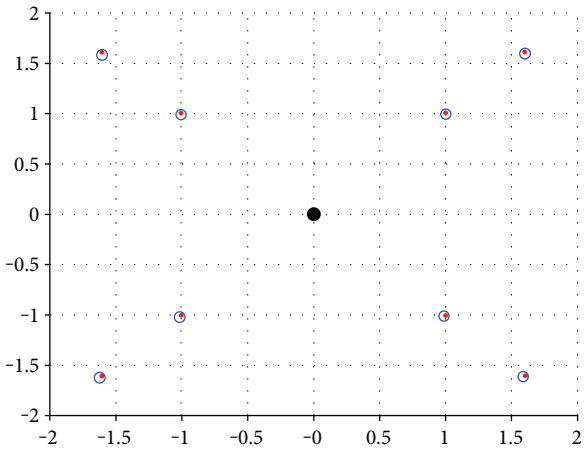

Figure 12: The normal detection algorithm tester: shown in blue are the real points of the orthogonal segment corners to the normal and shown in red is the algorithm approximation.

Table 1

Real normal			Approximated normal		
X	Y	Z	X	Y	Z
0.1808	0.6601	0.7291	0.1716	0.6743	0.7182
0.0417	0.2682	0.9625	0.0386	0.272	0.9615
0.0682	0.666	0.7428	0.0695	0.6716	0.7377
0.0594	0.0738	0.9955	0.0515	0.0694	0.9963

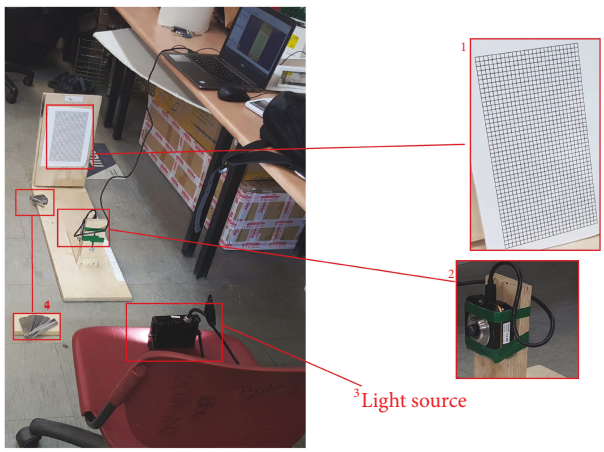

Figure 13: The practical setting of the next experiments that proves the accuracy level. The setting is divided into 4 main components: 1 is a dynamic plate that has the surface (plate) to be mapped which can move toward front or back. 2 is a stabilized stand that holds the camera. 3 is a light source to overcome the problem of the pixel snapping. 4 is a submillimeter spacer that we can measure the movement of the plate (1), so we have a reference for the actual movement.

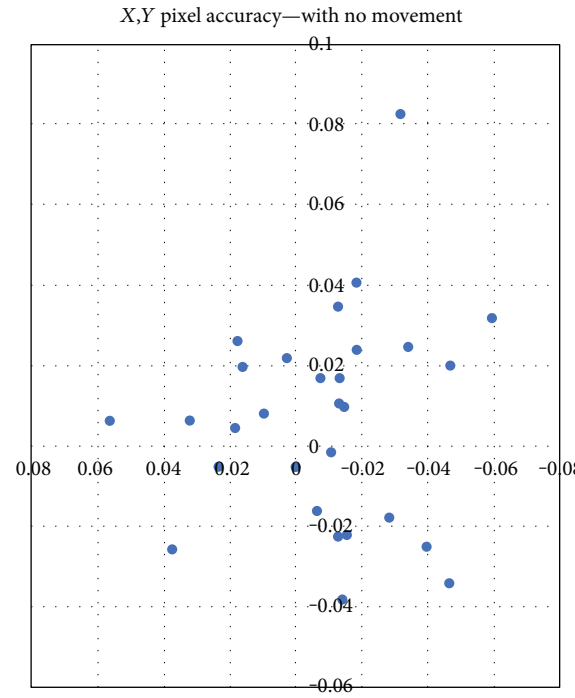

Figure 14: The typical noise of $level_0$ targets. Comparing two images of the same targets, they were shot with the same lighting and camera parameters. The typical noise level is about 0.03–0.04 pixel. In almost all tested cases, the $level_0$ noise is below 0.1 pixel.

of about 68°. In general, both cameras have reached the same accuracy level.

We have attached various types of paper and Kapton targets on the panel and tested the algorithm's ability to detect fine changes. Figure 14 shows the expected noise level from comparing two images of the same targets (without moving the panel), which is usually lower than 0.04 pixel. Figure 15 presents the proposed algorithm "in action"; the panel with 30 targets was moved 1 mm on average ((a) almost no movement, (b) 0.2 mm, (c) about 1.8 mm, and (d) about 2 mm) (Figure 16), and the presented graph shows the linearity as well as the accuracy of the $level_0$ dataset, presenting an accuracy level which is better than 0.2 mm on average.

In Figure 17, we show a test on the Kapton foil which introduced a reflection and lighting problems.

6.3. Computing the Difference Map and Minimum LAR Surface Algorithm. In this subsection, we present two typical examples of computing the difference map between two pairs of images. In the first example, we compared two images with no movement—this example is needed in order to test the expected noise level of the suggested method. Figures 18 and 19 show that in general such noise is significantly below the required accuracy. The second pair of images includes a linear movement of 0.3–1.2 mm of the image plate. Figures 20 and 21 show how such

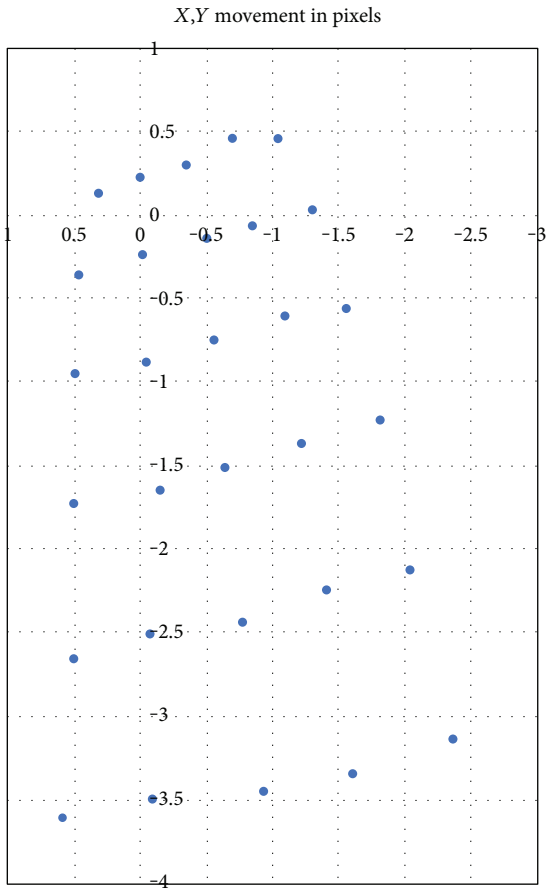

Figure 15: An example of a surface movement observed by 30 (5×6) targets (see Figure 16). Comparing two images, with an average tilt change of about 1 mm. A change of 0.1 pixel at targets is detectable (strongly above the noise level), allowing a z-accuracy of about 0.2 mm on level$_1$.

movement is detected—with an overall accuracy better than 0.1 mm.

7. Discussion and Future Work

We introduced a novel methodology for mapping flexible adaptive aerospace antennas. The proposed method can detect submillimeter distortions even on relatively large reflectors. The presented methodology allows autonomous (i.e., on-board) computation of surface, which can be used in order to continuously investigate the unknown nature of Kapton foil flexible antennas in the extreme temperatures of space. Using surface 3D mapping, the robotic subreflector can overcome minor distortions in the main reflector, allowing a typical gain improvement of 3–7 dB [3] and a new capability of dynamic beam shaping. The presented method was implemented and tested on a laboratory prototype of a nanosatellite with a two-foot flexible main reflector. The presented method reached an accuracy level of 0.1 millimeter on circular targets and a 0.3–0.5-millimeter accuracy on grid-based targets (with low-quality printed grid-based targets).

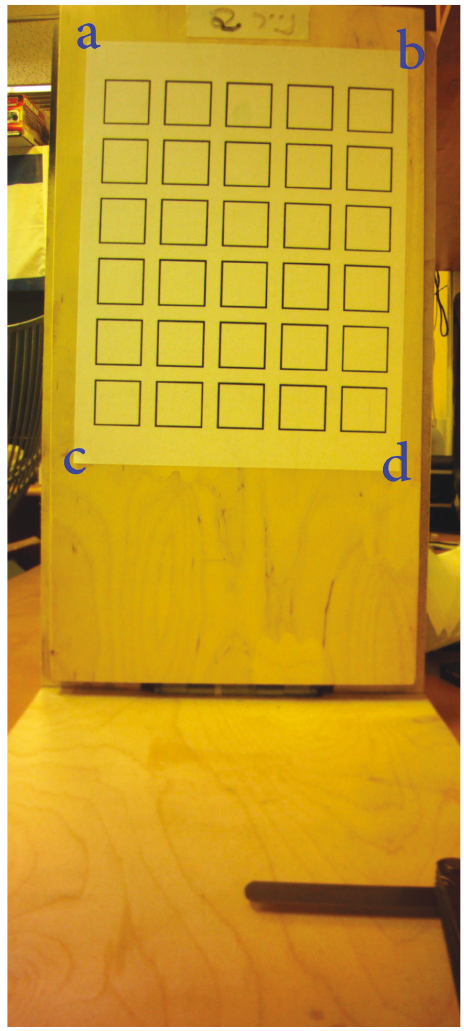

Figure 16: A testing setup: a paper with 30 (5×6) targets was attached to a flat panel. This panel can be moved using a set of accurate spacers (at a, b, c, and d), allowing us to test global movements with a high level of accuracy.

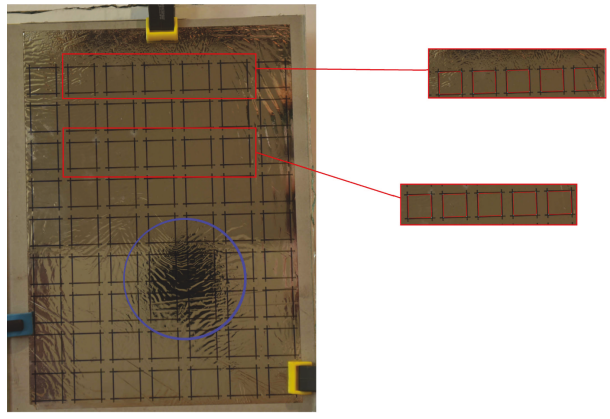

Figure 17: Kapton foil test. Denote the black reflection (marked by a blue circle) at the lower part—such reflection issues can be solved with a controlled lighting. Yet, any real implementation should be able to overcome reflection problems by detecting them and addressing them as outliers.

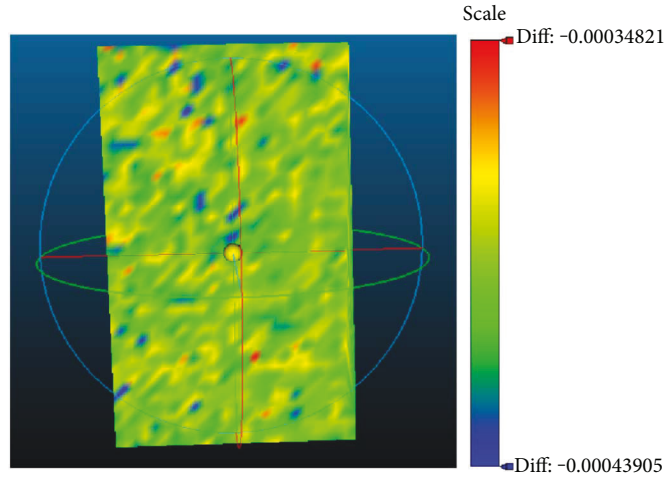

Figure 18: A difference map between two images taken continuously with no movement. In the scale of the difference, we can see that the maximum difference between the two images is 0.00035 pixel, which is significantly below 0.1 millimeter; this result demonstrates the algorithm's accuracy.

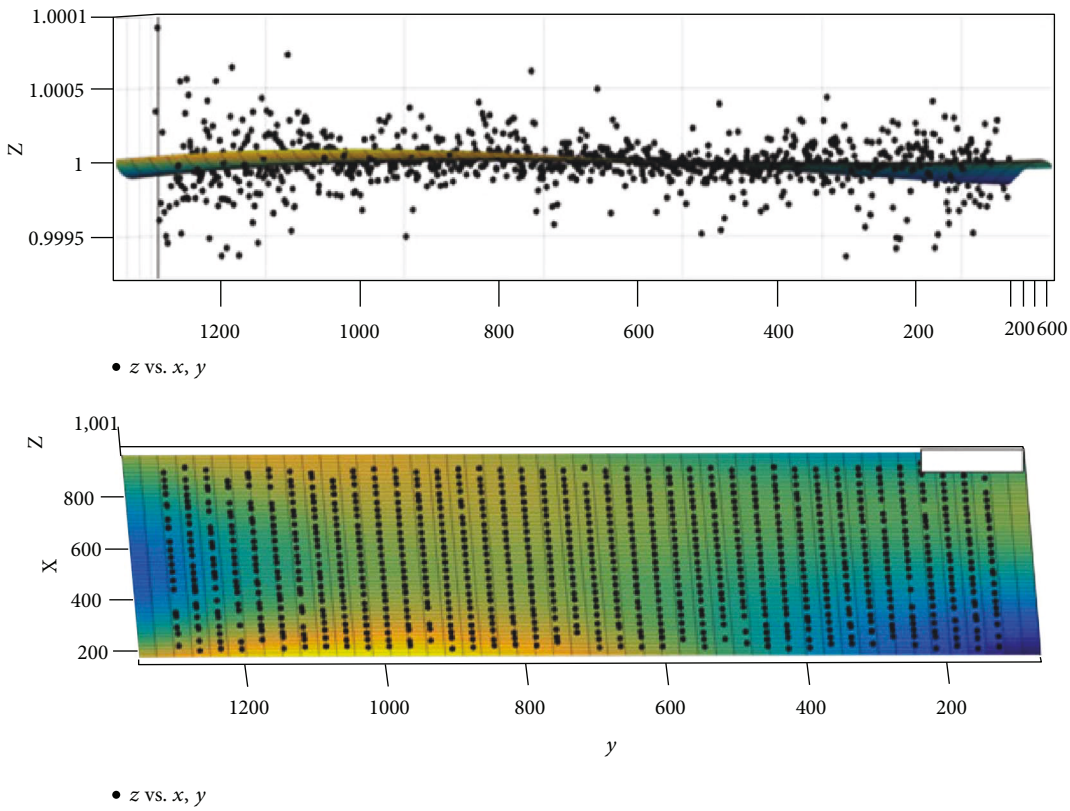

Figure 19: The minimum LAR surface optimization algorithm on the setting, running on two images taken continuously with no movement. The upper part of the figure shows the surface from the *ZY* plane viewpoint, and the lower part shows the *XY* plane viewpoint.

Using the "on-board" algorithm which uses a reference image, the expected accuracy reached the required level of 0.1 millimeter.

We plan to implement the current method on a real nanosatellite with a flexible antenna, which is scheduled to be launched at the end of 2018. We hope that using the presented framework, the vision of having a large-scale LEO satellite constellation which is both affordable and globally accessible can get one step closer to reality.

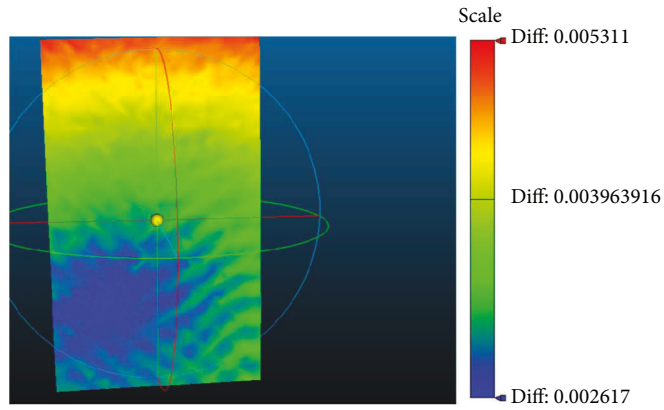

Figure 20: The result of the difference map with a 1.2 mm movement of the top of the plate (and a movement of about 0.3 mm of the lower part of the plate). This result demonstrates the ability of the algorithm to compute a smooth and continuance difference map.

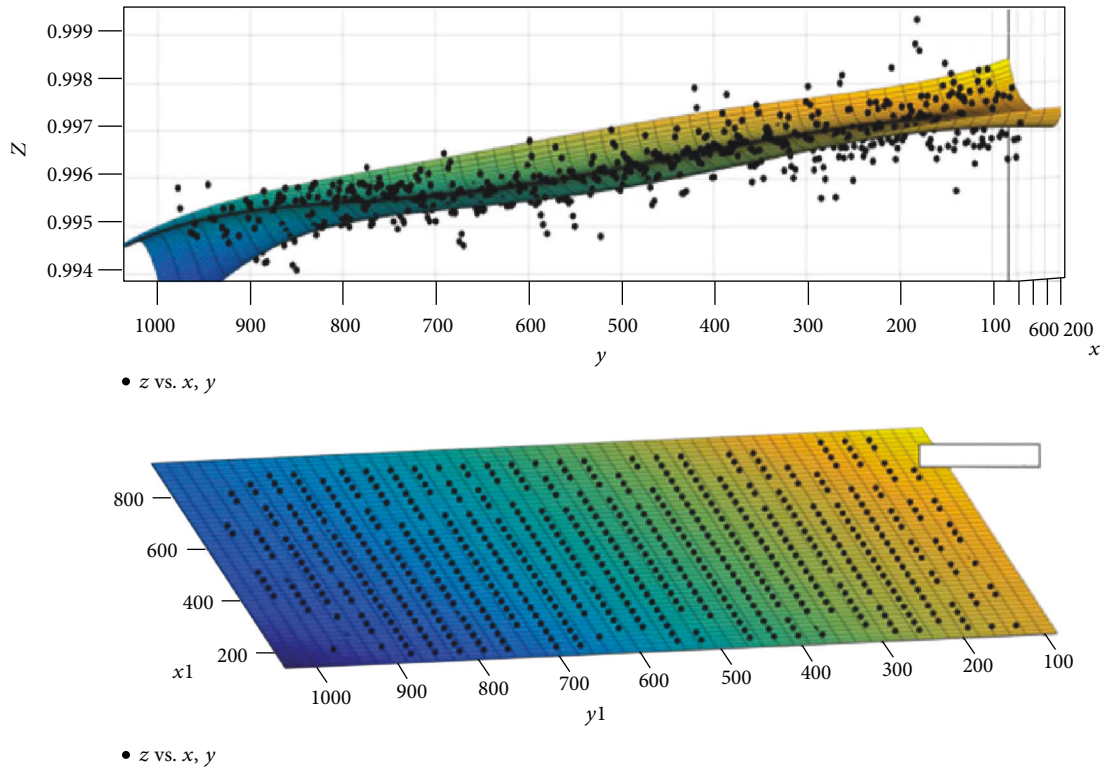

Figure 21: In this figure, we show the result of the minimum LAR surface optimization algorithm with a 1.2 mm movement of the top of the plate (and a movement of about 0.3 mm of the lower part of the plate). The upper part of the figure shows the surface from the *ZY* plane viewpoint, and the lower part shows the *XY* plane viewpoint.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by NSLComm. NSLComm (http://nslcomm.com) provides satellite manufacturers with unprecedented expandable antennas for satellite telecommunications. The authors would like to thank Peter Abeles for his amazing open source named *BoofCV* [18]—by far the most advanced geometric computer vision software.

References

[1] K. Kamnani and C. Suratkar, "A review paper on Google Loon technique," *International Journal of Research In Science & Engineering*, vol. 1, no. 1, pp. 167–171, 2015.

[2] R. Houborg and M. McCabe, "High-resolution NDVI from planets constellation of earth observing nanosatellites: a new data source for precision agriculture," *Remote Sensing*, vol. 8, no. 9, p. 768, 2016.

[3] N. Shvalb, B. Ben-Moshe, O. Medina, and R. I. Tamir, "Flexible-robotic reflector for aerospace applications," *International Journal of Antennas and Propagation*, vol. 2015, Article ID 252371, 8 pages, 2015.

[4] S. Schuon, C. Theobalt, J. Davis, and S. Thrun, "High-quality scanning using time-of-flight depth superresolution," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–7, Anchorage, AK, USA, June 2008.

[5] D. Acosta, O. Garcia, and J. Aponte, "Laser triangulation for shape acquisition in a 3D scanner plus scan," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA'06)*, vol. 2, pp. 14–19, Cuernavaca, Mexico, September 2006.

[6] R. A. Newcombe, S. Izadi, O. Hilliges et al., "KinectFusion: real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, Basel, Switzerland, 2011.

[7] E. E. Hitomi, J. V. L. Silva, and G. C. S. Ruppert, "3D scanning using RGBD imaging devices: a survey," in *Developments in Medical Image Processing and Computational Vision*, Lecture Notes in Computational Vision and Biomechanics, pp. 379–395, Springer, Cham, 2015.

[8] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 519–528, New York, NY, USA, 2006.

[9] F. Remondino and S. El-Hakim, "Image-based 3D modelling: a review," *The Photogrammetric Record*, vol. 21, no. 115, pp. 269–291, 2006.

[10] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3D models from single images with a convolutional network," in *Computer Vision – ECCV 2016. ECCV 2016*, Lecture Notes in Computer Science, pp. 322–337, Springer, Cham, 2016.

[11] D. Williams, Y. Zheng, P. G. Davey, F. Bao, M. Shen, and A. Elsheikh, "Reconstruction of 3D surface maps from anterior segment optical coherence tomography images using graph theory and genetic algorithms," *Biomedical Signal Processing and Control*, vol. 25, pp. 91–98, 2016.

[12] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys, "Live metric 3D reconstruction on mobile phones," in *2013 IEEE International Conference on Computer Vision*, pp. 65–72, Sydney, NSW, Australia, 2013.

[13] O. Medina, A. Shapiro, and N. Shvalb, "Resistor-based shape sensor for a spatial flexible manifold," *IEEE Sensors Journal*, vol. 17, no. 1, pp. 46–50, 2017.

[14] O. Medina, A. Shapiro, and N. Shvalb, "Kinematics for an actuated flexible n-manifold," *Journal of Mechanisms and Robotics*, vol. 8, no. 2, article 021009, 2016.

[15] Y. Demers, A. Liang, E. Amyotte, E. Keay, and G. Marks, "Very large reflectors for multibeam antenna missions," in *Fifteenth KA and Broadband Communications, Navigation and Earth Observation Conference*, pp. 23–25, Cagliari, Italy, 2009.

[16] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[17] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: an accurate $o(n)$ solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, 2009.

[18] P. Abeles, "Boofcv v0.27," 2017, http://boofcv.org/.