Journal of Astronomical Telescopes, Instruments, and Systems

AstronomicalTelescopes.SPIEDigitalLibrary.org

Subpixel real-time jitter detection algorithm and implementation for polarimetric and helioseismic imager

David Roma
Manuel Carmona
Jose Bosch
Albert Casas
Atila Herms
Manel Lopez
Oscar Ruiz
Josep Sabater
Thomas Berkefeld
Thorsten Maue
Eiji Nakai
Wolfgang Schmidt
Dirk Soltau
Reiner Volkmer
Jose M. Gomez

David Roma, Manuel Carmona, Jose Bosch, Albert Casas, Atila Herms, Manel Lopez, Oscar Ruiz, Josep Sabater, Thomas Berkefeld, Thorsten Maue, Eiji Nakai, Wolfgang Schmidt, Dirk Soltau, Reiner Volkmer, Jose M. Gomez, "Subpixel real-time jitter detection algorithm and implementation for polarimetric and helioseismic imager," *J. Astron. Telesc. Instrum. Syst.* **5**(3), 039003 (2019), doi: 10.1117/1.JATIS.5.3.039003.

Subpixel real-time jitter detection algorithm and implementation for polarimetric and helioseismic imager

David Roma,[a,b] Manuel Carmona,[a,b,c] Jose Bosch,[a,b,c] Albert Casas,[a,b] Atila Herms,[a] Manel Lopez,[a] Oscar Ruiz,[a] Josep Sabater,[b] Thomas Berkefeld,[d] Thorsten Maue,[d] Eiji Nakai,[d] Wolfgang Schmidt,[d] Dirk Soltau,[d] Reiner Volkmer,[d] and Jose M. Gomez[a,b,c,*]

[a]Universitat de Barcelona, Department of Electronic and Biomedical Engineering, Barcelona, Spain
[b]Institute of Space Studies of Catalonia, Barcelona, Spain
[c]University of Barcelona, Institute of Cosmos Sciences, Barcelona, Spain
[d]Leibniz-Institut für Sonnenphysik, Freiburg, Germany

Abstract. The polarimetric and helioseismic imager instrument for the Solar Orbiter mission from the European Space Agency requires a high stability while capturing images, specially for the polarimetric ones. For this reason, an image stabilization system has been included in the instrument. It uses global motion estimation techniques to estimate the jitter in real time with subpixel resolution. Due to instrument requirements, the algorithm has to be implemented in a Xilinx Virtex-4QV field programmable gate array. The algorithm includes a 2-D paraboloid interpolation algorithm based on 2-D bisection. We describe the algorithm implementation and the tests that have been made to verify its performance. The jitter estimation has a mean error of $\frac{1}{25}$ pixel of the correlation tracking camera. The paraboloid interpolation algorithm provides also better results in terms of resources and time required for the calculation (at least a 20% improvement in both cases) than those based on direct calculation. [DOI: 10.1117/1.JATIS.5.3.039003]

Keywords: image stabilization system; electronics; active optics; control; Solar Orbiter; polarimetric and helioseismic imager.

Paper 19012 received Jan. 29, 2019; accepted for publication Jul. 16, 2019; published online Jul. 30, 2019.

1 Introduction

The polarimetric and helioseismic imager (SO/PHI) is an instrument of the Solar Orbiter (SO) mission from the European Space Agency (ESA).[1] As the name indicates, one of its goals is to generate polarimetric images (PIs). These images allow observation of the magnetic field of the solar photosphere as shown in Fig. 1.

The polarimetric images[3] are calculated using a set of images with different polarizations that have the same field of view (FoV). Hereinafter, these images are called scientific images (SIs), as they are the main input for the magnetic field computation. This process includes subtractions, implying the requirement of a high signal-to-noise ratio (SNR) in the SIs because the differences are small. As a result, it is necessary to stabilize the images, as any displacement between SIs will smear the resulting polarimetric one, reducing its quality. A simulation of the SNR resulting from the platform jitter is shown in Fig. 2.

In ground-based solar telescopes, stabilization is provided by the telescope itself,[4,5] while in space-borne ones it is usually done by spacecraft (S/C). In case of polarimeters, the requirements of the instruments are quite stringent (in the present case an order of magnitude higher than the SO platform capabilities). For this reason, an image stabilization system (ISS) is added[6–9] to damp this jitter. Hinode has been the first reported mission to use a correlator,[10] being SO/PHI the second mission using correlator base stabilization, although with technological differences.

The ISS[11] has to compensate the S/C jitter in real time (RT), while the SIs are being taken. No roll correction is needed, as the S/C's attitude control system meets the SO/PHI requirements. Based on the S/C specifications and the polarimetric requirements, the attenuation curve of the ISS has been defined:[12]

- The turn over frequency, i.e., the frequency for which the attenuation curve becomes smaller than unity shall be $f_{t-o} \geq 30$ Hz.
- The attenuation curve shall be at levels >10 for all frequencies $f < 10$ Hz.
- The attenuation curve shall be at levels >30 for all frequencies $f < 5$ Hz.

This is the reason why SO/PHI has two cameras. The first is the focal plane assembly (FPA), which provides the SIs. Although the exposure time for an SI is in the order of 20 ms, the capture and transfer of the whole set of images, over all wavelengths and polarimetric phases, required to calculate the final PI takes several seconds. During this period, the line of sight has to be stable.

The second camera, the correlation tracking camera (CTC),[13] is used as input for the ISS. For this purpose, a beam splitter (BS) has been introduced in the light path of the FPA, so both share the same telescope. This camera, based on a CMOS active pixel sensor, is able to reach up to 1250 fps with the smallest image size and the lowest integration time. The number of pixels of the CTC (16 kpx) is much smaller than the FPA (4 Mpx),

*Address all correspondence to Jose M. Gomez, E-mail: jm.gomez@ub.edu

2329-4124/2019/$28.00 © 2019 SPIE

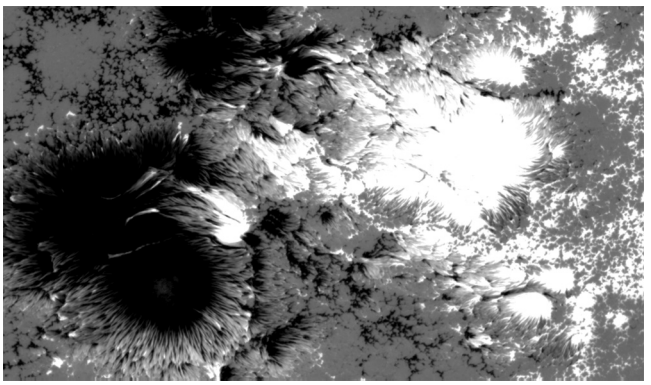

Fig. 1 Image of the Sun surface magnetic field distribution in the huge sunspot group in October 2014. [Observed by Solar Optical Telescope (SOT), wavelength: iron line (630 nm)].[2]

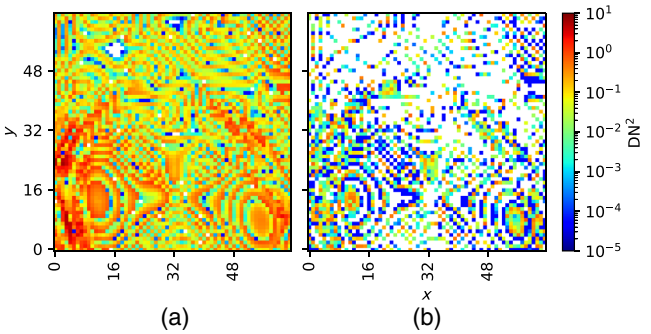

Fig. 2 (a) Simulation of the error introduced by the Solar Orbiter jitter and (b) the improvement if the jitter is reduced by a factor 10, both in DN^2. The white areas indicate an error value below 10^{-5}.

while the FoV of a pixel is similar (0.8 arc sec /pixel and 0.5 arc sec /pixel, respectively).

The jitter introduced by the platform is calculated by the ISS using a global motion estimation (GME) algorithm. The first image of the Sun surface taken by the CTC, similar to the one shown in Fig. 3, is used as a reference. This reference image (RI) is compared, using the GME, with the following CTC images, hereinafter live images (LIs). Both, the RI and the LI, are square and can have a size between 64×64 and 128×128 pixels.

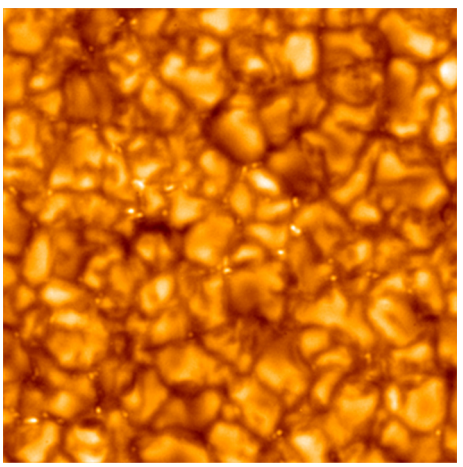

Fig. 3 Image of the Sun granules observed by SOT, wavelength: G band (430 nm).[14]

A subpixel paraboloid interpolation is applied to the GME results to further increase the displacement resolution. Thanks to the interpolation, the algorithm resolution is improved, which allows it to reach the required attenuation (factor of 10 at 10 Hz).

The resulting interpolation value is then fed to a control algorithm, which estimates the movement that has to make a tip-tilt mirror (TTM) to compensate the jitter. The new position is sent to the tip-tilt controller (TTC)[15] that moves the TTM, damping the jitter. The secondary mirror, which is part of the TTM assembly, is before the BS that feeds both, the CTC and the FPA.

The stabilized FoV reflected by the TTM is also received by the FPA. Only when the ISS loop is working and locked will the FPA capture SIs.

The ISS RI has to be updated after a number of LIs to take into account the evolution of the Sun surface. Based on the experience on ground-based Sun telescopes and the Sunrise balloon-borne telescope,[16] the period between RIs has been defined as 1 min. This value will be fine tuned during the commissioning phase of the mission. The flow diagram is shown in Fig. 4.

A similar solution has already been used in the Hinode mission[17] using a 50×50 pixel image. It has a delay of 3.2 ms including the sensor readout time of 2.6 ms. This implies a post-processing time of 0.6 ms. The upper delay limit for PHI to reach the desired damping is 3.5 ms,[11] while the number of pixels of the image can be 6 times bigger (128×128). The proposed algorithm minimizes the postprocessing time, using a pixel-wise strategy.

The ISS has been first implemented and tested using https://github.com/jmgc/myhdl-numeric, an open source fork of MyHDL,[18] which includes fixed-point support equivalent to the VHDL one.[19] The major advantage is that it allows use of the python unit testing facilities. It can also be used with a continuous integration server, simplifying the detection of possible inconsistencies in an early stage. Afterward, it has been automatically translated to VHDL and tested in a bread-board model of the ISS.

The remainder of this paper is organized as follows. Section 2 introduces the motion estimation systems, their constraints when used for jitter compensation, and finally focus on the spatial domain GMEs and subpixel interpolation. Section 3 presents the solution implemented in the ISS and explains the architecture used for the GME and the subpixel interpolation, optimizing the field programmable gate array (FPGA) resources. Section 4 describes the tests and results for both, the full chain and the paraboloid interpolation. Finally, the conclusions are presented in Sec. 5.

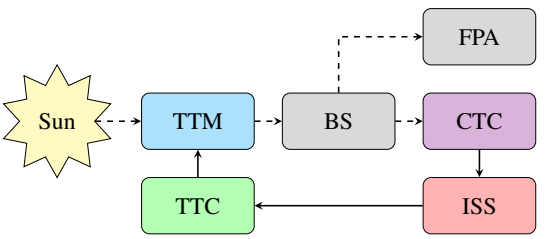

Fig. 4 ISS flow diagram. The OF is represented with dashed arrows, whereas the electrical flow is represented with a solid arrow. The acronyms are: TTM, tip-tilt mirror; BS, beam splitter; CTC, correlation tracking camera; FPA, focal plane assembly; TTC, tip-tilt controller; and ISS, image stabilization system

2 Motion Estimation Systems Overview

The core functionality for jitter compensation is based on motion estimation algorithms. The ISS structure is based on them and, specifically, on video based motion estimation ones, since the acquisition system is a camera.

Video-based subpixel motion estimation methods usually assume a sequential structure. First, the camera captures one frame, which is received and stored by the frame buffer (FB). Every image received by the FB follows an enhancement process to improve its quality. This enhancement can be at pixel level, e.g., offset and gain corrections, or at image level, e.g., gradient subtraction.

This first acquired image is considered the RI and all the displacements will be referred to it. Afterward, the subsequent LIs are stored also in the FB and enhanced. Then the motion estimation algorithm is applied between every LI and the RI. Finally, to improve the resolution, subpixel motion interpolation may be applied.

2.1 *Motion Estimation Algorithms*

As seen before, the core functionality for our system is the motion estimation algorithm. Taking into account our objective, we may distinguish between two kinds of algorithms:

- *Optical flow*. This kind of algorithm focused on the motion of patterns between the images. The most commonly used are Lucas–Kanade–Shi–Tomasi (LKST)[20,21] (based on the spatial intensity gradient), Horn–Schunk (HS)[22] (estimates the velocity field), and the most recent Farneback[23] (uses polynomial expansion). All these algorithms have also been implemented in FPGAs.[24–26]
- *GME algorithms*. In this case, the image is treated as a whole and a correlation (CT) between images is performed. They work in the spatial domain (e.g., HINODE), or pixel domain,[27–29] frequency or Fourier domain,[30,31] and wavelet domain.[32] In case of Sun surface motion estimation, the spatial and frequency domain alternatives are preferred[33] as, in general terms, they are less complex to implement, reducing the delay introduced by the algorithm calculations. For this reason, although there are implementations of wavelets in FPGAs or application-specific integrated circuits,[34–37] they have been discarded.

The major difference between them is that the optical flow (OF) methods focus on the displacement of specific patterns and, from there, try to extract the global displacement. On the contrary, GME algorithms process the image as a whole and extract directly the displacement between successive images. In this sense, OF may allow knowing the specific displacement of certain patterns on the image. But if we consider that the images are quasistatic and there is only a movement between scene and observer, the most natural solution seems to be the GME algorithms.

2.2 *Constraints for Jitter Compensation*

Jitter compensation and motion estimation systems are based on the same principles. However, jitter compensation sends the calculated displacements to an actuator which compensates the actual motion. As a result, the latency introduced by the calculation becomes critical. Furthermore, in some cases, it is possible to achieve a better jitter damping lowering the accuracy of the system if the decrease in response time is big enough.

As the ISS works in an RT closed loop, the stability of the loop depends on the time elapsed between the capture of an image by the CTC and the final displacement of the TTM (τ_{CL}). As a result, there is a maximum delay allowed to capture, receive, process a frame, and estimate the jitter. If this delay (τ_{CL}) is reduced, the loop stability improves. If an FB is used to store every LI, the maximum allowed processing time (τ_{jitter}) will be the difference between the τ_{CL} and the time required to receive and store an image (τ_{image}).

Conversely, if a pixel-wise strategy is used, the full τ_{image} can be used to calculate the jitter, maximizing the time for calculation (τ_{jitter}) while minimizing the latency (τ_{CL}). This implies to do most of the calculations while the stream of pixels is being received.

To ensure stability, the frame rate (f_{image}) shall be at least 10 times the f_{t-o}. As the f_{t-o} is 30 Hz, the $f_{image} \geq 300$ fps. Also the upper delay limit for τ_{CL} to reach the desired damping is 3.5 ms.[11]

The major contributions to the delay come from the photon budget, readout of the NOIS1SM100A[38] sensor and the interface bit rate (100 Mbps). The CTC has been optimized to allow the transfer of the pixel values at the same rate as they are sampled. As a result, the time required to transfer the full image can be considered the same as the readout time. This implies that for the CTC the f_{image} can be considered equivalent to τ_{image}. As the τ_{image} is 3.33 ms, the maximum allowed postprocessing time is 167 μs, nearly 4 times smaller than the HINODE case.

All frequency domain GMEs require the use of FBs. Furthermore, spatial covariance requires subtracting a fitted plane from the image to minimize the effect a linear trend in intensity has in the position of the peak.[39] Hence, all of them were discarded. The final solution makes use of a spatial domain GME with a new pixel-wise optimized implementation, which is described in the following sections.

2.3 *Spatial Domain Global Motion Estimation*

These methods are mainly based on the sum of squared differences (SSD) [Eq. (1)] or sum of absolute differences (SAD) [Eq. (2)]:

$$\mathrm{SSD}(x,y) = \sum_{k,l} [\mathrm{LI}(k,l) - \mathrm{RI}(k-x, l-y)]^2, \tag{1}$$

$$\mathrm{SAD}(x,y) = \sum_{k,l} |\mathrm{LI}(k,l) - \mathrm{RI}(k-x, l-y)|, \tag{2}$$

where k and l are the pixel indices of the CTC image, and x and y are the correlation space indices.

The values of k, l, x, and y are limited by the RI image size. Assuming that the image row and column counting starts with the value 0, the limits are defined by

$$\begin{cases} 0 <= x <= k < n_{\mathrm{cols}} \\ 0 <= y <= l < n_{\mathrm{rows}} \end{cases}, \tag{3}$$

where n_{rows} is the number of rows, and n_{cols} is the number of columns of the CTC image. In our case, both parameters will have the same value (n_{image}).

The point of maximum likelihood of the displacement between both images will be the position of the minimum value of the SSD or SAD.

2.4 *Subpixel Paraboloid Interpolation*

As the maximum allowed jitter is less than a pixel, it is necessary to reach subpixel resolution. Only using the GME, the maximum resolution reached is a pixel, and therefore, the uncertainty is still of ± 0.5 px. Therefore, a paraboloid interpolation algorithm around the point of maximum likelihood has been used. The paraboloid is calculated with the nine cells CT matrix obtained from the GME, i.e., Eq. (1) or Eq. (2).

The total displacement will be the addition of the GME displacement plus the subpixel displacement. Once found the pixel level displacement with the GME, we can assume that the remaining displacement between the RI and the LI is less than half a pixel, which is the uncertainty level of the GME. The subpixel algorithm will use as input data the result of calculating the SSD or SAD, which we define as $z_{i,j}$. So x and y are substituted by i and j, respectively, to avoid confusion with the previous section. The cell with the minimum value will be the (0,0) and its value $z_{0,0}$. The full set of values is represented as the matrix $z_{i,j}$:

$$z_{i,j} = \begin{pmatrix} z_{-1,-1} & z_{-1,0} & z_{-1,+1} \\ z_{0,-1} & z_{0,0} & z_{0,+1} \\ z_{+1,-1} & z_{+1,0} & z_{+1,+1} \end{pmatrix}. \tag{4}$$

The paraboloid used for the subpixel interpolation[40] can be defined as

$$\begin{cases} z(x,y) = a_{00} + a_{01}x + a_{10}y + a_{02}x^2 + a_{11}xy + a_{20}y^2 \\ x \in (-1,1) \\ y \in (-1,1) \end{cases}, \tag{5}$$

where the coefficients a_{mn} need to be determined using the values $z_{i,j}$.

Substituting x and y by the sample indices $(-1,0,1)$ in the previous paraboloid equation, we get the values of the matrix $z_{i,j}$:

$$\begin{cases} z_{-1,-1} = a_{20} + a_{02} + a_{11} - a_{10} - a_{01} + a_{00} \\ z_{-1,0} = a_{20} - a_{10} + a_{00} \\ z_{-1,+1} = a_{20} + a_{02} - a_{11} - a_{10} + a_{01} + a_{00} \\ z_{0,-1} = a_{02} - a_{01} + a_{00} \\ z_{0,0} = a_{00} \\ z_{0,+1} = a_{02} + a_{01} + a_{00} \\ z_{+1,-1} = a_{20} + a_{02} - a_{11} + a_{10} - a_{01} + a_{00} \\ z_{+1,0} = a_{20} + a_{10} + a_{00} \\ z_{+1,+1} = a_{20} + a_{02} + a_{11} + a_{10} + a_{01} + a_{00} \end{cases}, \tag{6}$$

where i represents a change of row, equivalent to a movement on the y direction in the image, while j is a change of column, x direction in the image. Also since the system is overdetermined, the coefficient a_{11} is calculated using the values of $z_{\pm 1,\pm 1}$. Notice that a_{11} represents the interactions between x and y that can be seen as the interpolation paraboloid rotating over the z axis.

The solution proposed by Ref. 41 is used to solve the overdetermination:

$$\begin{cases} a_{00} = z_{0,0} \\ a_{01} = \frac{z_{0,+1} - z_{0,-1}}{2} \\ a_{10} = \frac{z_{+1,0} - z_{-1,0}}{2} \\ a_{11} = \frac{z_{+1,+1} + z_{-1,-1} - z_{+1,-1} - z_{-1,+1}}{4} \\ a_{02} = \frac{z_{0,+1} - 2 \cdot z_{0,0} + z_{0,-1}}{2} \\ a_{20} = \frac{z_{+1,0} - 2 \cdot z_{0,0} + z_{-1,0}}{2} \end{cases}. \tag{7}$$

As it is a paraboloid, the coordinates of the minimum (x, y) can be determined calculating the partial derivatives and solving the resulting system of equations:

$$\begin{cases} \frac{\partial z}{\partial x} = 0 \\ \frac{\partial z}{\partial y} = 0 \end{cases}. \tag{8}$$

This yields the following equation for the subpixel displacement $(x_{\text{pi}}, y_{\text{pi}})$:

$$\begin{cases} x_{\text{pi}} = \frac{a_{10}a_{11} - 2a_{20}a_{01}}{4a_{02}a_{20} - a_{11}^2} \\ y_{\text{pi}} = \frac{a_{01}a_{11} - 2a_{02}a_{10}}{4a_{02}a_{20} - a_{11}^2} \end{cases}, \tag{9}$$

In these equations, the denominator can be zero in some pathological cases. This has to be taken into account in the division algorithm implementation. In the proposed algorithm, presented in the next section, this situation is avoided by design.

3 Optimized Jitter Estimation Algorithm

Jitter estimation algorithms in ground-based solutions can use up-to-date high-end computers, digital signal processor (DSP) platforms or parallel processing solutions, all of them including floating-point calculation, which simplifies the implementation of the algorithms. In space projects, the number of available components is small, as they have to be qualified for this application. When the decision was taken, the only high-end solution available for ESA missions was the Xilinx Virtex-4QV[42] FPGA.

To minimize resources, the FPGA needs to include all the elements of the chain, from the camera interface to the generation of control signals for the TTM. It also uses only square images, with size $n_{\text{image}} \times n_{\text{image}}$.

3.1 *System Blocks*

As explained before, to have the lowest possible latency, the processing of the displacement needs to be done while the image is being received. For this reason, the typical steps used in a motion estimation system have been modified avoiding the use of an FB.

- *CTC*. Captures the images and sends them coded in a bitstream.
- *Camera interface*. Decodes the bitstream received from the camera and extracts the pixels.
- *Pixel enhancement*. Corrections applied to the pixels, e.g., dark and flat image compensation.
- *Global motion estimation*. GME computation.
- *Subpixel paraboloid interpolation*. Improve the displacement resolution.

- *Control loop*. Control system to filter and command the jitter compensation to the actuators.

When the ISS starts, a number of images are discarded to ensure the CTC has reached its steady state. The first image that is not discarded is used as an RI, and it is stored for the calculation of the correlation matrix. The following images are considered LIs. As indicated previously, the RI is updated periodically. The steps are described in more detail in the following sections.

3.2 *Pixel Enhancement*

The jitter detection starts preprocessing the images obtained from the solar surface. The CTC includes the possibility to amplify the image pixel values digitally by powers of two and remove an offset. This does not modify the SNR of the signal but improves the use of the CTC communication interface.

The second step aims to remove the fixed-pattern noise[43] coming from the sensor read-out noise and internal biases. The dark noise has been considered negligible for the present application, as the integration time is very small (some milliseconds). These fixed patterns are minimized by applying an image enhancement algorithm with calibrated parameters. For every single pixel of the camera, the gain and the offset are determined using dark and flat frames, respectively. These calibration values are applied to every pixel while they are being received:

$$I(x,y) = a(x,y) \cdot [c(x,y) - o(x,y)], \tag{10}$$

where $c(x,y)$ is the incoming pixel, $a(x,y)$ is the gain parameter, $o(x,y)$ is the offset parameter, and $I(x,y)$ is the enhanced pixel used as input for the GME algorithm. The resulting image pixels have 8-bit resolution.

3.3 *Pixel Level Global MOTION Estimation*

The correlation matrix $\mathrm{CT}_{x,y}$ is calculated using a spatial domain method, as it allows to avoid the need of an FB for the LIs. The available spatial domain methods are mainly based on the SSD [Eq. (1)] or SAD [Eq. (2)].

The point with the minimum value is considered the point of maximum likelihood. As the system is going to work in close-loop, and it shall damp the jitter, the displacements are expected to be nearly 0. For this reason, only a small number of $\mathrm{CT}_{x,y}$ cells around the point of maximum likelihood (x and y equal to 0) are going to be computed and stored. This also allows to make these calculations in parallel.

Contrary to what we may expect, the SSD is preferred in front of the SAD for this FPGA implementation. The Virtex-4 has multiplication modules (DSP), but it has to synthesize the add modules using multiple logic cells. The absolute value calculation requires an add operation that has to be synthesized, while the square uses a multiplier module. As the cells are scarce and the multiplier is going to be idle if it is not used, the SSD is used. This allows to optimize the use of cells.

To minimize the calculations, the size of the correlation matrix has been optimized. The size value is the minimum needed to close the loop taking into account the platform stability. Since the SO platform is expected to have a stability (σ) of 1 pixel, using the $\pm 3 \cdot \sigma$ rule (or $6 \cdot \sigma$), a correlation matrix ($\mathrm{CT}_{x,y}$) of 7×7 elements is calculated by Eq. (11). Hence, to compute the SSD in parallel, 49 multipliers (DSP modules) are used at once:

$$\mathrm{CT}_{x,y} = \mathrm{SSD}(x,y) \quad \forall\, x,\, y \in (-3,3), \tag{11}$$

where x and y are the correlation matrix indices.

The algorithm has to take into account two aspects. The first is that frame of 3 pixels around the LI shall be discarded to avoid border effects. For this reason, only the pixels that are not part of this frame are considered valid to calculate the correlation. The second is the serialized reception of the pixels from the camera. When the first pixel that is considered valid is received, a 7×7 matrix with the RI pixel that is in the same position (k,l), and the 48 around are subtracted from the LI pixel, the result is self-multiplied and stored in an accumulator matrix ($\mathrm{CT}^{p}_{x,y}$). This process is described in Algorithm 1.

When the last LI pixel has been received and operated, the matrix $\mathrm{CT}_{x,y}$ will be available. The accumulator matrix is reset to zero for every new image. After a number of LIs, the RI will be refreshed with a new CTC image.

To allow this calculation to be done while a pixel is received, two strategies have been followed. The first was divide the image in the same number of subimages as the $\mathrm{CT}_{x,y}$ (7×7). Every subimage is stored in an independent memory inside the FPGA. The subimage size and pixel assignment inside the subimage is defined by the following equations:

$$\begin{cases} n_s = \lceil n_{\mathrm{image}}/7 \rceil \\ k_s = \mathrm{rem}(k,7) \\ l_s = \mathrm{rem}(l,7) \\ k_{\mathrm{px}} = k // 7 \\ l_{\mathrm{px}} = l // 7 \end{cases}, \tag{12}$$

where $\lceil x \rceil$ is the ceiling function, // is the integer division, rem is the remainder function, (k,l) are the incoming RI pixel column and row, n_s is the subimage size ($n_s \times n_s$), (k_s, l_s) determines

Algorithm 1 Correlation calculation

1: $p = 0$ ▷ The valid pixel counter is initialized

2: $\mathrm{CT}^{p}_{x,y} = 0 \quad \forall\, x,\, y \in (-3,3)$ ▷ The correlation matrix is initialized

3: **while** Receiving pixel $\mathrm{LI}_{k,l}$ **do** ▷ The pixels of the live image are received serialized

▷ If the LI pixel is not part of the outer frame, it is considered valid

4: **if** $(k >= 3)$ & $[k < (n_{\mathrm{cols}} - 3)]$ & $(l >= 3)$ & $[l < (n_{\mathrm{rows}} - 3)]$ **then**

5: $p = p + 1$ ▷ Increment the valid pixel counter

6: $\mathrm{CT}^{p}_{x,y} = \mathrm{CT}^{p-1}_{x,y} + (\mathrm{LI}_{k,l} - \mathrm{RI}_{k-x,l-y})^2 \quad \forall\, x,\, y \in (-3,3)$ ▷ Calculate the correlation

7: **end if**

8: **end while**

9: $\mathrm{CT}_{x,y} = \mathrm{CT}^{p}_{x,y}$ ▷ The correlation matrix $\mathrm{CT}_{x,y}$ has been calculated

10: **return** $\mathrm{CT}_{x,y}$ ▷ Return the correlation matrix and wait for the next LI to restart the process

the subimage where the pixel is stored, and $(k_{\mathrm{px}}, l_{\mathrm{px}})$ are the pixel position inside the subimage.

Using this method, it is possible to extract the 49 RI pixels needed to calculate the CT of an LI pixel in a single-clock period.

The second is the introduction of a small pipeline that fetches the 49 RI pixels from their corresponding subimages, calculates the difference with the present LI pixel, and calculates the 49 accumulations, as indicated in Eq. (1).

The process has been implemented in an isochronous way, ensuring that a correlation matrix is available after the last image pixel is received from the CTC. As a result, the time needed to calculate the CT is the same as the one needed to receive an image.

The cell with the minimum value is searched in the resulting $\mathrm{CT}_{x,y}$ matrix. The displacements at pixel level are the indices x_{ct} and y_{ct} for the minimum of $\mathrm{CT}_{x,y}$, with an error of $\pm 1/2$ pixel. This cell with the minimum value and its eight nearest neighbors are stored to be used for the interpolation:

$$z_{i,j} = \mathrm{CT}_{x_{\mathrm{ct}}+i, y_{\mathrm{ct}}+j} \quad \forall\, i,\, j \in (-1,1), \tag{13}$$

with $z_{0,0}$ having the value of $\mathrm{CT}_{x_{\mathrm{ct}}, y_{\mathrm{ct}}}$.

Using the present implementation, the SSD calculations are finished before the final image pixel is received.

3.4 *Subpixel Paraboloid Interpolation Algorithm*

The calculation of the paraboloid [Eq. (10)] can be done easily if floating point numbers[44] are available. However, floating point numbers will require a larger number of resources than the fixed point counterparts.[45] For this reason, an approach based on fixed-point numbers has been designed and implemented.

To minimize the resources, a successive approximation approach has been selected. The algorithm uses a 2-D bisection procedure. The process is done using shift and add operations with fixed-point numbers. The strategy is similar to the one used by coordinate rotation digital computer.[46]

As explained in Sec. 2.4, the method will start with a 3×3 matrix, as shown in Eq. (4), using the correlation matrix computed by the GME. As a first overview, before explaining it in depth, the algorithm goes as follows:

1. The paraboloid coefficients are calculated from the resulting 3×3 correlation matrix values.
2. Using these coefficients, a set of paraboloid values are interpolated around the current minimum which, by construction, is the correlation matrix center value [Eq. (13)].
3. A new minimum is searched between these new interpolated values, and its position is determined.
4. Knowing the position found at the previous step, the paraboloid coefficients are rescaled and translated to become the new center position.

The algorithm will iterate the two last steps. Since the minimum will always be closer to the center value than any other point, at each iteration the error is reduced by one half of the previous step resolution. Hence, the process just has to be iterated until the desired resolution is reached.

For the following part of this section, where the algorithm is explained in depth, the following nomenclature will be used. As the process is iterative, the superindex p will indicate the current iteration step for the paraboloid coefficients (a^p) and the interpolated correlation matrix values (z^p), while the subindex will be used in case of variables (step_p, x_p, and y_p). The initialization step is indicated by p equal to -1.

At the initialization, the first step calculates the paraboloid coefficients (a_{mn}^{-1}) using the correlation matrix values determined by the GME algorithm. This is done substituting the values of $z_{i,j}$ in Eq. (7).

The following steps will be repeated on every iteration. The first step is rescaling the paraboloid coefficients to the new subfield, centered on the current minimum of the correlation matrix. Since the new subfield has half the size of the previous one, this is done scaling the a_{mn} coefficients as shown in

$$\begin{cases} a_{00s}^{p} = a_{00}^{p-1} \\ a_{01\,s}^{p} = \frac{a_{01}^{p-1}}{2} \\ a_{10s}^{p} = \frac{a_{10}^{p-1}}{2} \\ a_{02s}^{p} = \frac{a_{02}^{p-1}}{4} \\ a_{11s}^{p} = \frac{a_{11}^{p-1}}{4} \\ a_{20s}^{p} = \frac{a_{20}^{p-1}}{4} \end{cases}, \tag{14}$$

where an s has been added to the subindices to indicate the scaling of the coefficients. Notice that the division by two and four is equivalent to a binary shift right by one or two places, respectively. This allows avoiding an explicit division module when programmed in an FPGA.

Now, using Eq. (6), the values of $z_{i,j}^{p}$ can be calculated. Although it may seem that with only four points, it may be enough, it is important to calculate the full eight points around the center position, as it is explained in detail later. Also the coefficient a_{00} does not affect the calculation of the minimum position, for this reason its value is always set to 0:

$$\begin{cases} z_{-1,-1}^{p} = a_{20s}^{p} + a_{02s}^{p} + a_{11s}^{p} - a_{10s}^{p} - a_{01s}^{p} \\ z_{-1,0}^{p} = a_{20s}^{p} - a_{10s}^{p} \\ z_{-1,1}^{p} = a_{20s}^{p} + a_{02s}^{p} - a_{11s}^{p} - a_{10s}^{p} + a_{01s}^{p} \\ z_{0,-1}^{p} = a_{02s}^{p} - a_{01s}^{p} \\ z_{0,0}^{p} = 0 \\ z_{0,1}^{p} = a_{02s}^{p} + a_{01s}^{p} \\ z_{1,-1}^{p} = a_{20s}^{p} + a_{02s}^{p} - a_{11s}^{p} + a_{10s}^{p} - a_{01s}^{p} \\ z_{1,0}^{p} = a_{20s}^{p} + a_{10s}^{p} \\ z_{1,1}^{p} = a_{20s}^{p} + a_{02s}^{p} + a_{11s}^{p} + a_{10s}^{p} + a_{01s}^{p} \end{cases}. \tag{15}$$

The minimum of the calculated z^p values is determined, and its indices i and j are stored in u_p and v_p, respectively. It is important to notice that we do not care about the values in 15, we are only interested in the u_p and v_p values (the position) of the minimum.

From a theoretical point-of-view, the objective is to use this minimum position as the new $z_{0,0}$ matrix value. In this sense, if this translation is applied to x and y to Eq. (5), yields:

$$z(x,y) = a_{00s} + a_{01s}(x+v_p) + a_{10s}(y+u_p) + a_{02s}(x+v_p)^2 + a_{11s}(x+v_p)(y+u_p) + a_{20s}(y+u_p)^2. \quad (16)$$

Rearranging the previous coefficients, the translated paraboloid coefficients in terms of x and y can be found:

$$\begin{cases} a_{00}^p = a_{00s}^p \\ a_{01}^p = a_{01s}^p + 2 \cdot a_{02s}^p \cdot v_p + a_{11s}^p \cdot u_p \\ a_{10}^p = a_{10s}^p + a_{11s}^p \cdot v_p + 2 \cdot a_{20s}^p \cdot u_p \\ a_{02}^p = a_{02s}^p \\ a_{11}^p = a_{11s}^p \\ a_{20}^p = a_{20s}^p \end{cases}. \quad (17)$$

Notice that the multiplication by u and v can be substituted by a simple comparison, as u_p and v_p can only have the values $(-1,0,1)$.

Also the step is scaled:

$$\text{step}_p = \frac{\text{step}_{p-1}}{2}. \quad (18)$$

Finally, the new x and y coordinates of the minimum position are calculated:

$$\begin{cases} x_p = x_{p-1} + v_p \cdot \text{step}_p \\ y_p = y_{p-1} + u_p \cdot \text{step}_p \end{cases}. \quad (19)$$

As in the case of Eq. (17), the multiplications can be substituted by comparisons. Most importantly, the last values of x_p and y_p will be the computed subpixel displacement. As indicated previously, the number of iterations will indicate the precision in bits of the paraboloid interpolation results $(x_{\text{pi}}, y_{\text{pi}})$.

It may be considered that for the subpixel interpolation, it is no need to interpolate at eight points around the center (minimum) value and that four should suffice. This may be true in some situations but fails in case the paraboloid representing the correlation matrix of the GME is rotated. In general, as we may see in Fig. 5, with four points, it is enough when moving to the new subfield. But when the paraboloid is rotated, Fig. 6, the minimum falls outside the new search area. Conversely, if eight positions are interpolated, Fig. 7 shows that this issue is solved, and the minimum is inside the new subfield search area. The full set of operations is summarized in Algorithm 2.

The final implementation makes eleven iterations. The first eight are needed to reach the desired subpixel resolution. The remaining three are added to calculate the rounding of the eighth bit. The total displacement (x_d, y_d) is the sum of the correlation displacement $(x_{\text{ct}}, y_{\text{ct}})$ and the paraboloid interpolation result $(x_{\text{pi}}, y_{\text{pi}})$.

4 Results

The results from four sets of tests are presented. The first validates the method to estimate the S/C jitter. The second compares the results between the proposed method (CT) and OF ones, starting with a simple paraboloid pattern, and afterward using CTC images. Finally, the paraboloid implementation is compared with other alternatives.

4.1 *Full Chain Results*

The objective of the tests is to determine the error of the jitter estimation using Algorithm 2. The displacements are in the subpixel region. As it has been stated before, the selected resolution for the test images is 1/128 of a pixel.

The optical characteristics of SO/PHI provides a pixel FoV of ~96 km/pixel on the Sun surface, when SO is in the perihelion. Nowadays, the maximum spatial resolution (2 pixels) of Sun images is ~50 km.[47] Hinode or GREGOR images could be used to generate the patterns. However, artificial images have determined features (noise, etc.), which have advantages for testing the algorithm. Only if the dynamic of the solar evolution of the granulation needs to be taken into account are time series from granulation needed, either simulated or observed ones.

4.1.1 *Test image generation*

There are different alternatives. The most common one is assuming that a pixel is a point. As a result, the displaced value can be modeled interpolating the values of the neighbor pixels.

A more realistic approach requires taking into account the area of the pixel. In this case, an image with a spatial resolution equivalent to the subpixel desired resolution is needed and afterward binning is applied on them to have the desired final pixel resolution. In our case, the RI and LI size is 128×128 pixels. As the required displacement resolution is 1/128, the image that will be used to simulate the image on the sensor needs to have a minimum size of 16384×16384 pixels. Hereinafter, we will use the term subpixel to refer to the resolution of this high-resolution image and pixel for the final 128×128 image. Since we will apply different displacements on the images, an additional frame around the image is required to avoid extrapolation.

To generate the high-resolution images, two alternatives have been considered. The first one is using the Sun equations to calculate the images.[48] This option has been discarded, as the equations are complex to solve and require computational power. The second one uses three-dimensional computer graphics (3DCG) to simulate the images.

For the present tests, the second option has been selected. In particular, a number of metaballs have been randomly generated in an FoV and rendered afterward. For this purpose, the Blender 3DCG application has been used. The final image size is $25{,}000 \times 25{,}000$ pixels with 16 bits resolution, and a contrast of a 13%. This means that we are using an outer frame of 8616 pixels for each side, providing a margin much higher than the maximum ± 3 pixels allowed by our algorithm.

From these high-resolution images, a 8-bit depth CTC image is generated tiling the image. Every tile corresponds to a pixel of the RI or LI. As a result, the value of every one of the 128×128 pixels of the image is the mean value of the 128×128 subpixels of its corresponding tile. An example is shown in Fig. 8. It can be seen that the images have some noise, whose source is the ambient occlusion[49] used for the images generation. The value change is around 1 DN, which is equivalent to the expected photon noise when the full linear region of the sensor is used, as should be the case for the present application, and the enhancement algorithm has been applied. The live (displaced) image is obtained the same way, but an integer offset is applied to the high-resolution images prior the tiling of the image.

The patterns produced by the granules are the ones used to detect the displacement. In practice, the difference between the RI and the LI comes from the displacement due to the

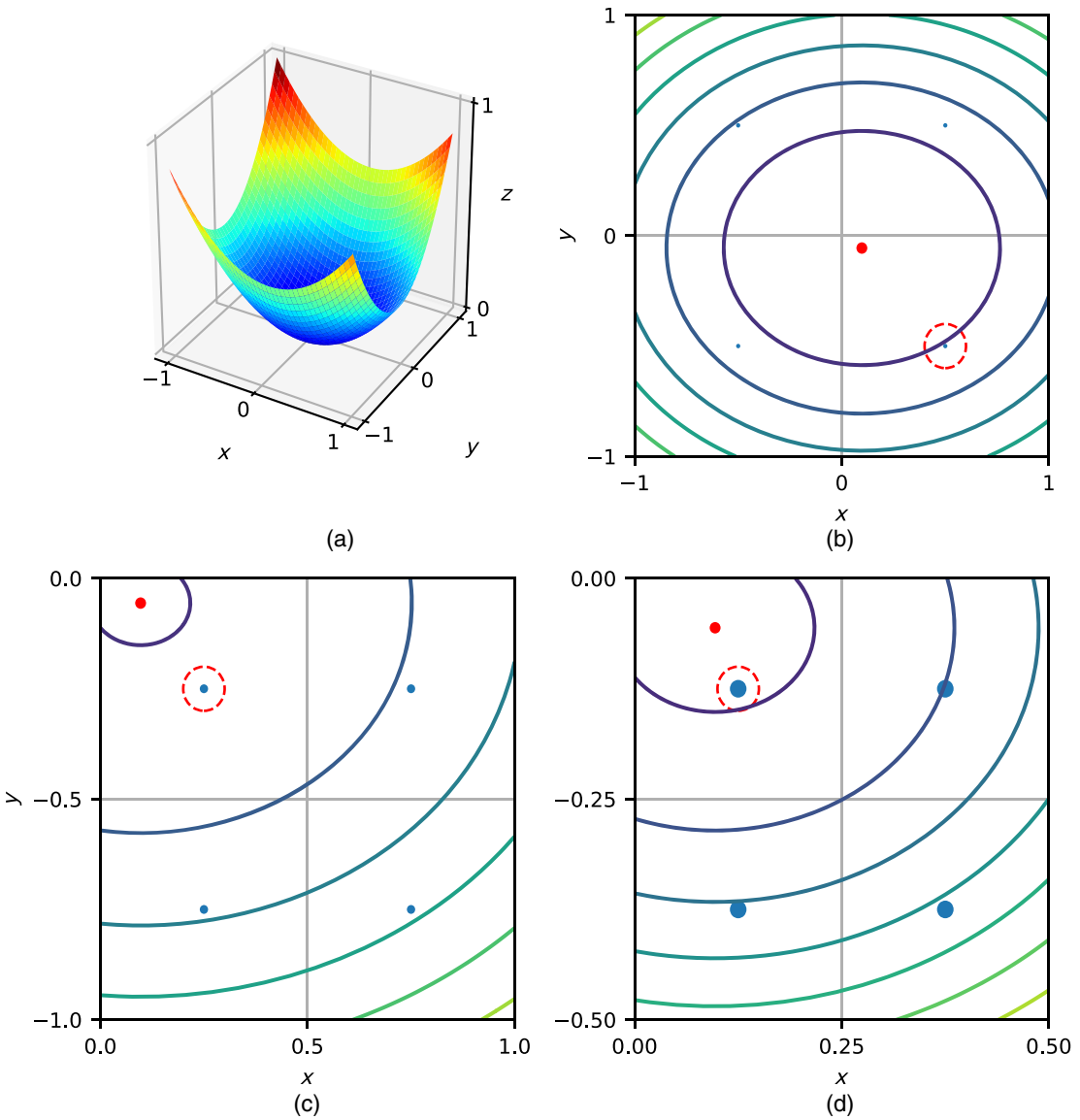

Fig. 5 (a) 3-D paraboloid plot and (b)–(d) three first steps of the algorithm calculation. The contour lines show the paraboloid values. The minimum of the paraboloid is the red dot near the coordinate (0, 0) in figure (b).

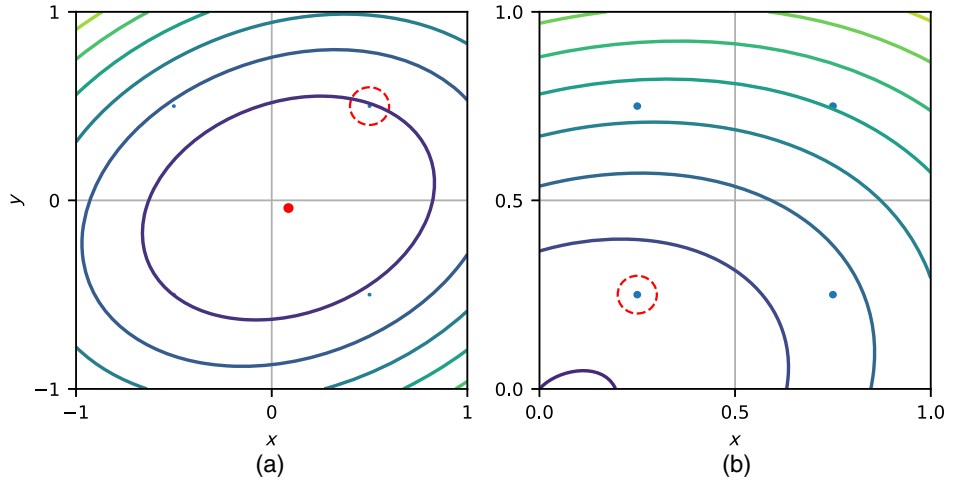

Fig. 6 (a) Anomaly when the paraboloid is rotated. The indicated point (red dashed line) has the minimum value of the four. (b) If the quadrant with the dashed circle is selected, it can be seen that the paraboloid minimum (red dot) is outside. As a result, the minimum cannot be reached.

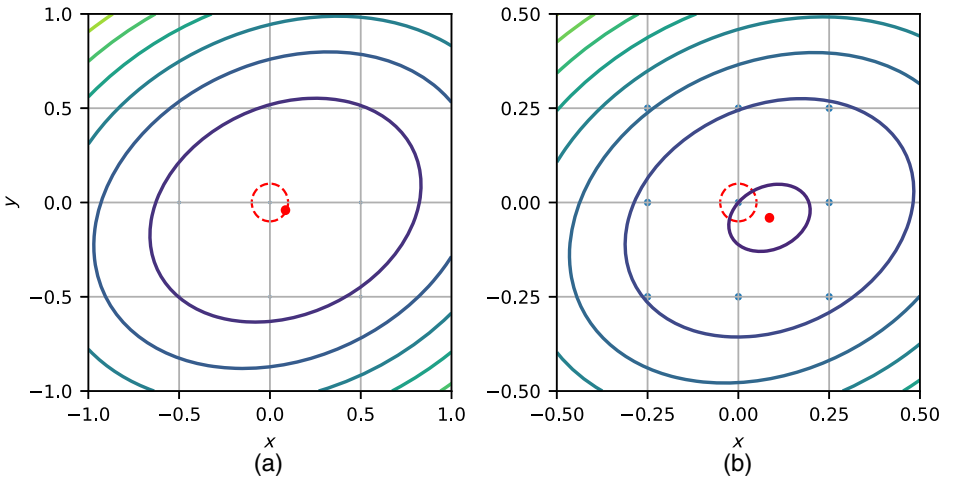

Fig. 7 (a) Final implementation of the paraboloid algorithm showing the overlapping subsquares with their centers as blue dots and (b) the paraboloid minimum (red dot) is now located inside the selected subsquare.

Algorithm 2 Paraboloid minimum finding algorithm using a 2-D bisection procedure.

1: **procedure** PARABOLOID($z_{i,j}$) ▷ Matrix cells
2: $z^{-1}_{i,j} \leftarrow z_{i,j}$ from SSD
3: Calculate a^{-1}_{mn}
4: step ← 1
5: $p \leftarrow 0$ ▷ initialize p to the bit 0
6: **for** $p <$ precision **do** ▷ loop until precision bit
7: Scale a^{p}_{mns} using Eq. (14)
8: Calculate $z^{p}_{i,j}$ using Eq. (15)
9: Find the minimum $z_{i,j} \Rightarrow u, v \leftarrow i, j$
10: Calculate a^{p}_{mn} using Eq. (17)
11: $\text{step}_p \leftarrow \frac{\text{step}_{p-1}}{2}$
12: Update x_p and y_p using Eq. (19)
13: **end for**
14: $x_{\text{pi}}, y_{\text{pi}} \leftarrow x_p, y_p$ ▷ The minimum location
15: **return** x, y
16: **end procedure**

movements of the platform and the dynamics of the solar surface. In our case, we will focus on the first. The second can be considered negligible as the frame rate is orders of magnitude higher than the time constant of this phenomenon.

An RI and an LI with a displacement of 0.2 pixels have been used to calculate the SSD between them. It can be seen that the differences are small, with a magnitude of 1 DN. Following the algorithm described in Sec. 3.3, the same calculation is done for the center position and the 48 neighboring ones ($\text{CT}_{x,y}$). Afterward, the 9 SSD points around the maximum likelihood

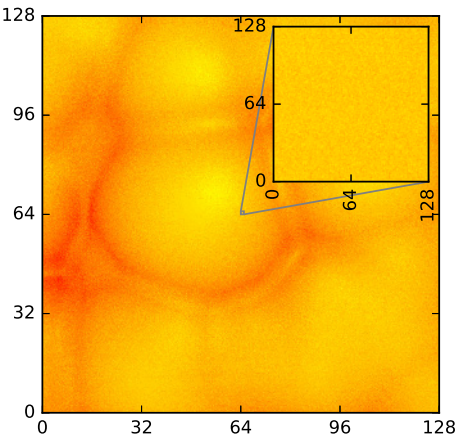

Fig. 8 The image represents a 128×128 image, which can be a RI or LI. Every pixel is determined calculating the mean value of the pixels shown in the inset.

point are selected ($z_{i,j}$), and the paraboloid interpolation is calculated to determine the minimum position. The full process is shown in Fig. 9.

SO/PHI is designed to be operative with nominal performance between minimum perihelion distance (0.28 AU) and ∼0.80 AU.[50] The granules sizes are very different depending on the S/C distance to the Sun. This aspect can increase the errors of the motion estimation. For this reason, different FoVs have been generated simulating different distances. These distances assume worse case scenarios. The first eight ones are simulations for 0.2 AU, whereas the last eight ones are for 0.9 AU. Figure 10 shows the simulated FoVs.

A test has also been done to detect possible privileged directions. For this purpose, the same tests have been done with the FoV flipped. This is equivalent to doubling the FoVs.

From every FoV, 1024 RIs and their corresponding LIs have been generated. The coordinates have been determined using two approaches. For the first one (we will name it XY), the distribution is uniform in the X and Y directions.

The second approach, which will be named Polar case, uses a uniform distribution in the radial direction. Although the radius

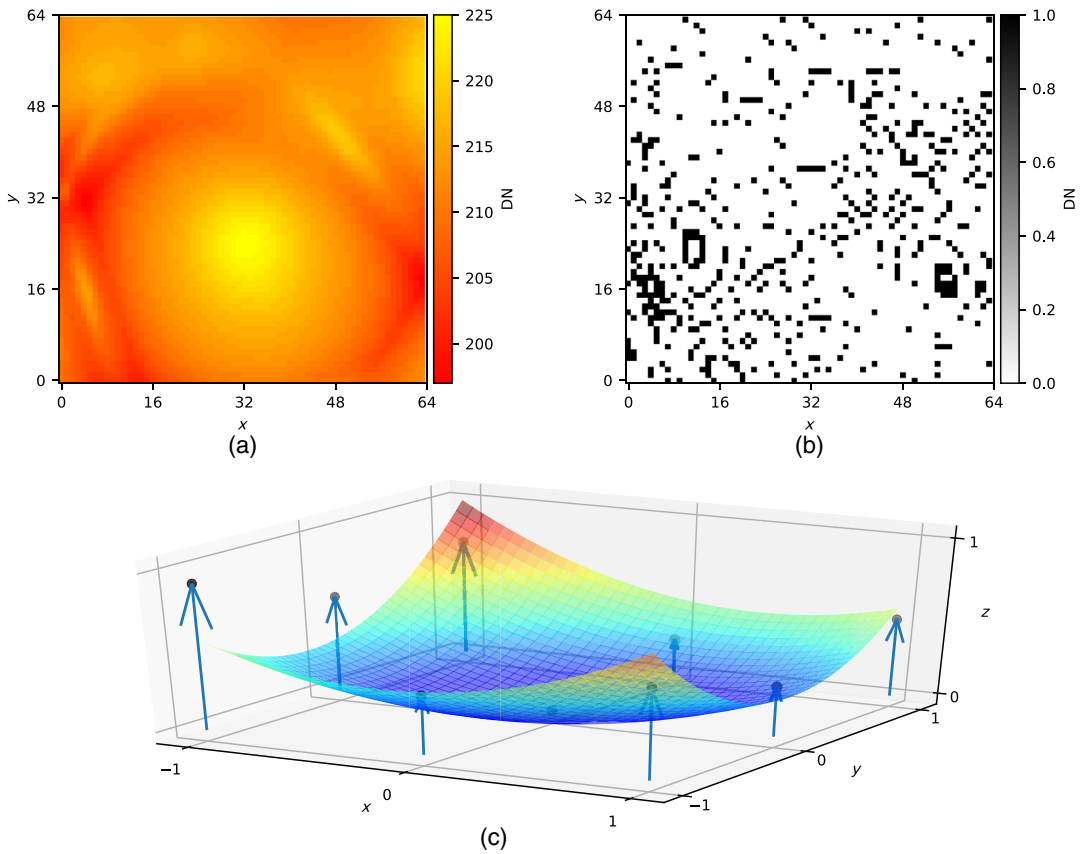

Fig. 9 Squared differences process: (a) RI, (b) squared differences with LI, and (c) resulting paraboloid.

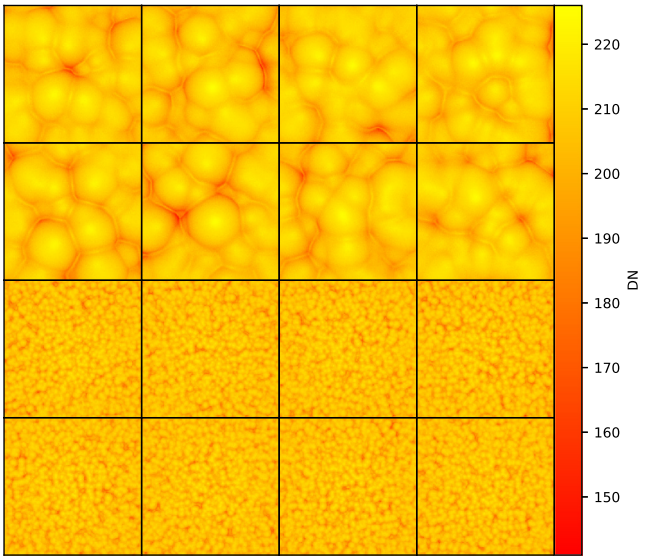

Fig. 10 Simulated FoV of the CTC with the Sun at 0.2 AU (upper two rows) and 0.9 AU (lower rows). The intensity distribution is in arbitrary units.

is smaller than half the length of the pixel, an uniform distribution over the angle Θ can be used. When the radius is over this distance, it is necessary to modulate the probability of Θ taking into account the shape of the corners of the pixel. This ensures that the histograms in the radial direction will be correct. Algorithm 3 describes the implementation.

Algorithm 3 Uniform radial distribution. To simplify the calculation, the random angle is calculated for one quadrant. Afterward a quadrant is assigned randomly, covering the four quadrants.

procedure UNIFORM_RADIUS(side) ▷ pixel side
  $R \leftarrow \frac{side}{2}$ ▷ The radius of the inscribed circle
  $\theta_{min} \leftarrow 0$ ▷ The minimum angle of a quadrant
  $\theta_{max} \leftarrow \frac{\pi}{2}$ ▷ The maximum angle of a quadrant
  $R_{max} \leftarrow R \cdot \sqrt{2}$ ▷ Radius of the circle that inscribes the pixel
  $\Delta_R \leftarrow \text{random}(0, R_{max})$
  $\Delta_{quadrant} \leftarrow \frac{\text{int}[\text{random}(-2,2)] \cdot \pi}{2}$
  if $\Delta_R > R$ **then** ▷ If the radius is bigger than the incircle radius, the angle shall be inside the quadrant
    $\theta_{min} \leftarrow \cos^{-1}\left(\frac{radius}{\Delta_R}\right)$
    $\theta_{max} \leftarrow \frac{\pi}{2} - \theta_{angle}$
  end if
  $\Delta_\theta \leftarrow \text{random}(\theta_{min}, \theta_{max})$
  $\Delta_\theta \leftarrow \Delta_\theta + \Delta_{quadrant}$
  return Δ_R, Δ_θ
end procedure

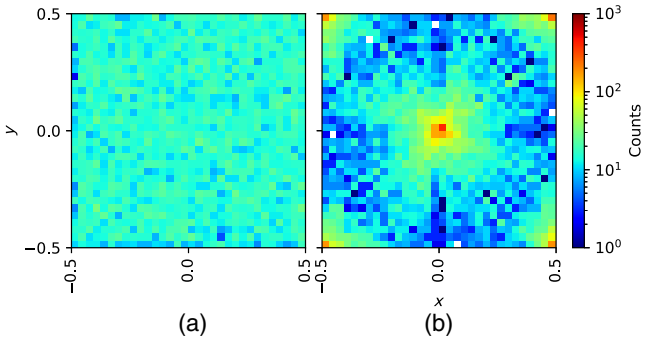

Fig. 11 Histogram of the distribution of the RI offsets: (a) uniform distribution in the X and Y directions and (b) the uniform radial distribution (Algorithm 3).

In Fig. 11, both distributions are shown. Notice that the polar one has a circular pattern in the middle and an increase of the density near the vertex. This ensures that the number of points at every radial distance is the same.

4.1.2 *Full chain test results*

First, the errors (ϵ) in the calculation of the displacement between different RIs and LIs have been determined. For this, a uniform X–Y random displacement (Δ_{sim}) has been applied. Both images (reference and live) have the same size.

Four different tests have been done for perihelion and aphelion with RI and LI sizes of 128×128 pixels and 64×64 pixels.

The displacements (Δ_{calc}) have been calculated. The ϵ between both displacements (Δ_{sim} and Δ_{calc}) has been derived. The histograms for the different cases are shown in Fig. 12.

From the plot, we can derive that the worst case is for 64 pixel images at the perihelion. The error is nearly distributed around the full pixel (maximum ϵ 0.4).

Using the uniform radial distribution, it is possible to analyze the error depending on the displacement modulus. The results

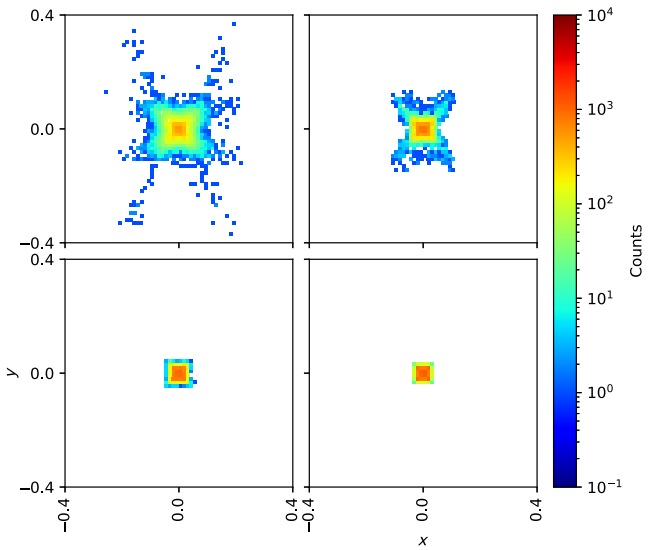

Fig. 12 ϵ_{cal} histogram calculated using the XY uniform distribution. The columns indicate the size of the RI (left 64 px and right 128 px), the rows indicate the distance (upper perihelion and lower aphelion).

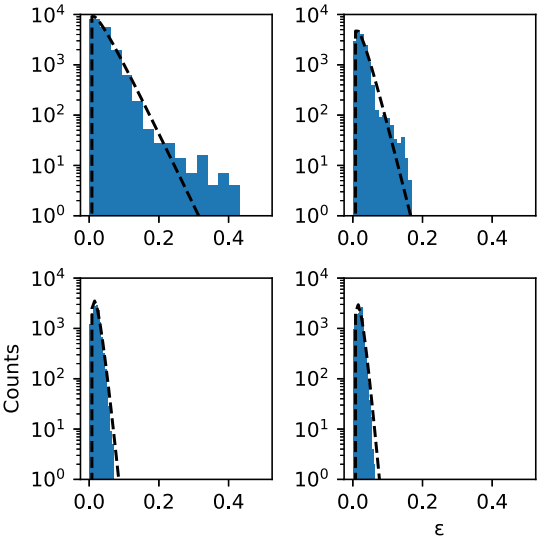

Fig. 13 ϵ_{cal} histogram calculated using the uniform radial distribution. The columns indicate the size of the RI (left 64 px and right 128 px), the rows indicate the distance (upper perihelion and lower aphelion).

Table 1 Calculated mean and std values for the error distribution resulting from the uniform radial displacements.

Size (px)	Distance	Mean	Std
64	Perihelion	0.040	0.034
128	Perihelion	0.026	0.019
64	Aphelion	0.020	0.010
128	Aphelion	0.019	0.010

are shown in Fig. 13. The behavior has been modeled using the Gamma probability distribution, which is presented with dashed lines. The calculated mean and std values are presented in Table 1.

From the histograms, it can be seen that the mean error is below $\frac{1}{25}$ pixel, being the worst case with 64 pixel images at the perihelion. Moreover, 80% of points are inside the circle with $\epsilon < \frac{1}{16}$ pixel. In case of aphelion, both sizes have equivalent results. For this reason, an smaller image can be selected, allowing to increase the control-loop working frequency.

These results will be used to determine the control loop parameters of the full ISS.[11] The selected resolution for the subpixel displacement is a 128'th of a pixel (7 bits).

4.2 *Algorithms Comparison*

The objective of the present test is to compare the results obtained from the proposed method using SSD and paraboloid interpolation (correlation), and the ones resulting from LKST, HS, and Farneback. The OpenCV[51] implementation is used.

A set of images has been generated using a paraboloid gradient. The RI and its 3-D representation are shown in Fig. 14.

The images have been displaced in the x direction following the Δ_x sequence $(0, 0, 0, \delta, 0, -\delta, 0, \delta, 2\delta, \delta, 0, -\delta, -2\delta, -\delta, 0)$, where δ has the value 0.766 to determine the subpixel accuracy. The results are shown in Fig. 15.

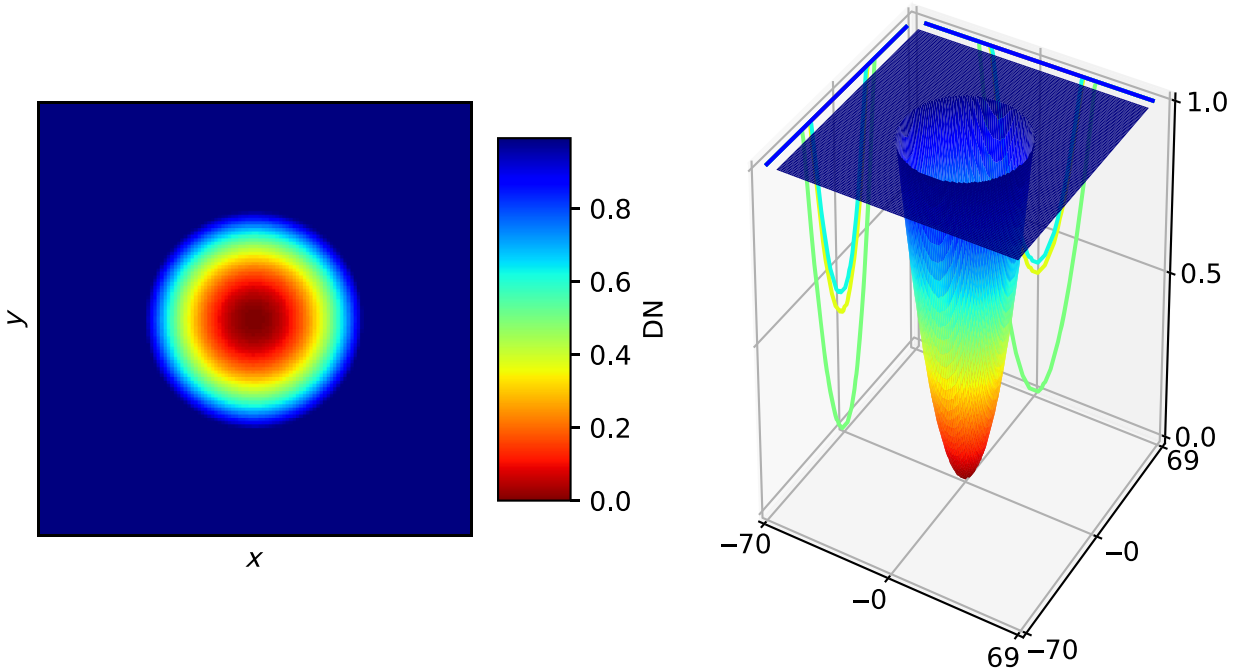

Fig. 14 Test image based on a paraboloid gradient.

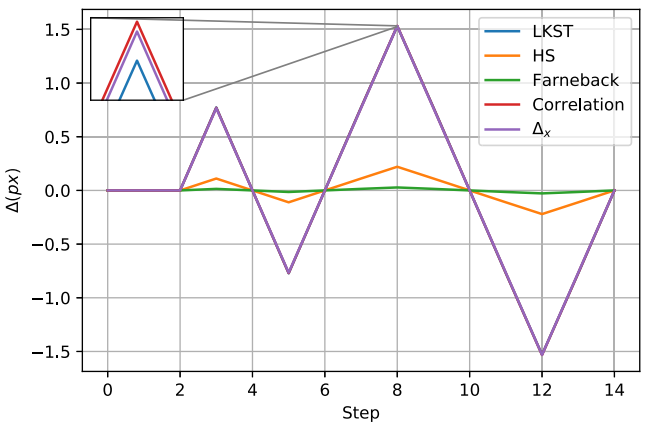

Fig. 15 Motion estimated by the different algorithms. It can be seen that the algorithm HS and Farneback detect a reduced displacement, whereas LKST and CT provide similar results, with a value nearly equal to Δ_x (see the inset).

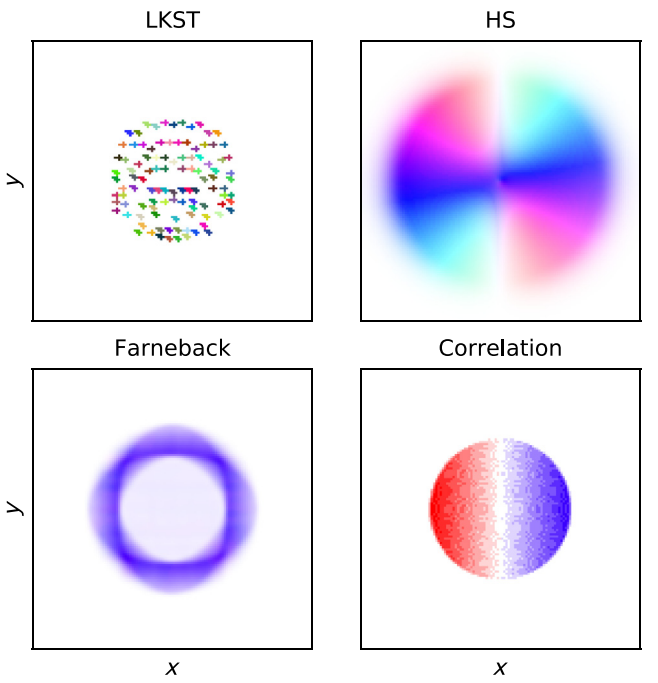

Fig. 16 Qualitative representation of the displacements detected by the different methods using Fig. 14: LKST is represented as arrows, for every one of the 128 pixels selected; HS and Farneback are represented colors, where the color represents the angle, and the intensity represents the magnitude (white is magnitude 0). Correlation is represented as the difference between the images.

It can be seen that LKST and correlation provide similar results and close to the simulated displacements, whereas HS and Farneback are much smaller, especially the last one.

These results can be analyzed using Fig. 16.

- The Shi–Tomasi algorithm selects features from the images, in this case a total of 128. It can be seen that the detected features are mostly on the shaded areas shown in the CT image. These features are afterward tracked using the Lucas–Kanade algorithm, which provides a vector with the displacement of every feature. The global motion can be estimated determining the mean value of these vectors.
- The Farneback and HS algorithms determine the displacement for all the points in the image. Most of the points are considered quasistatic (shown in white), as they have the same value in both images. This lowers the magnitude of the mean calculation used to estimate the displacement.

Table 2 Calculated maximum displacement error.

Method	Error (px)
LKST	1/128
HS	1/2.3
Farneback	1/2
Correlation	1/128

- The CT algorithm finds the displacement between two images minimizing the differences between them. From the image, it can be seen that there has been a displacement to the right. The blue shadows indicate that the motive has moved reducing the superposition, whereas the red ones indicate that the superposition has increased. If the full image is used, the algorithm can estimate the global motion. For these tests, a subpixel resolution of 8 bits has been used.

As presented in Table 2, the error using this image for LKST and correlation is around a 1/128'th of a pixel. In case of correlation, this is nearly the ideal case as the resolution is 8 bits. Hence, 1/128'th of a pixel is the resolution selected to generate the images in the following test cases.

Based on this result, the ISS subpixel interpolation resolution has been specified as 8 bits.

4.3 *CTC Image Comparison*

A test has been done using real images from the CTC engineering model. The pattern has been generated using a fiber bundle illuminated by a red LED. The output of the fibers has been focused using a lens on the CTC sensor. The pattern has been displaced on one direction using a motorized stage following the previous Δ_x sequence. The first image is shown in Fig. 17.

It is important to notice that the value of δ is unknown in this case, as the motors precision is below the required one. For this reason, as a first approach, the value calculated in every algorithm for the first unit displacement is used as the estimated value for δ. The estimated displacement is shown in Fig. 18. The four algorithms (correlation, LKST, HS, and Farneback) have a similar behavior as the one shown in Sec. 4.2.

The correlation and LKST algorithms have similar results, although they have differences of $\frac{1}{2}$ pixel. HS and Farneback also behave the same way, detecting a smaller displacement. To compare the results obtained by the different algorithms, a simulated image has been interpolated using the RI and displacing it the estimated value indicated by every algorithm. These images have been subtracted from the real one to determine the differences. The differences images for every algorithm are shown in Fig. 19.

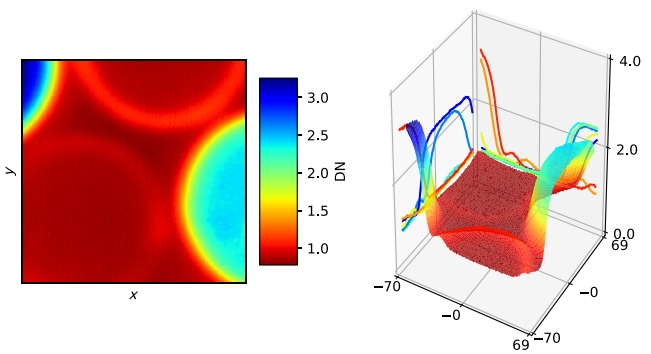

Fig. 17 Image captured with the CTC. It can be seen that the major part of the image has a nearly constant value.

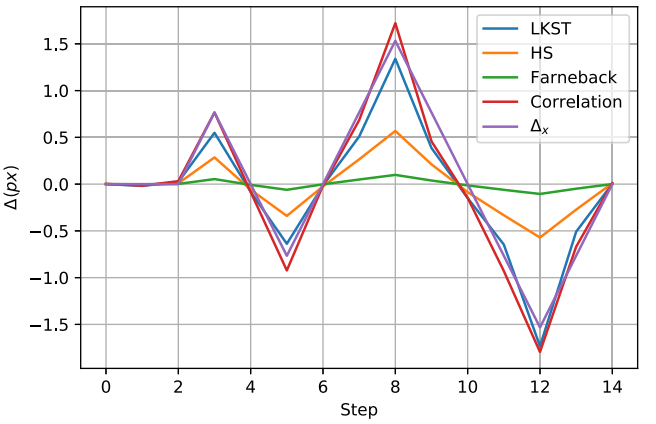

Fig. 18 Four algorithms' estimations and the motor sequence. It can be seen that the results are consistent with the test ones.

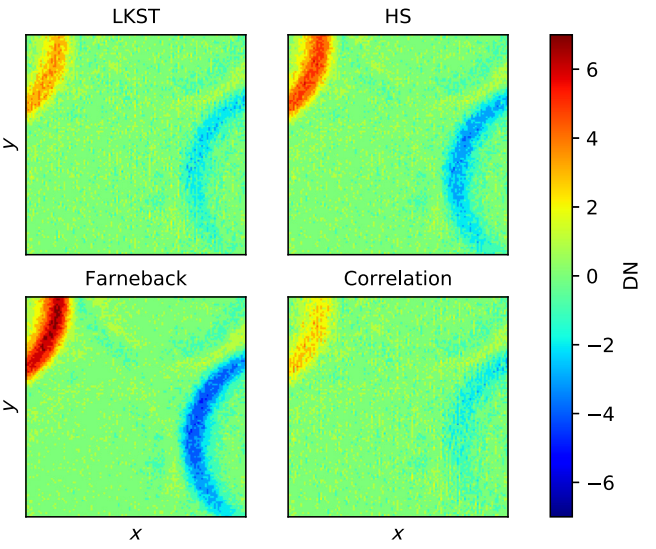

Fig. 19 Differences between an image displaced one motor step and the RI displaced the estimated value calculated by the four algorithms. The correlation is the one with smaller errors.

The absolute mean and squared error values are presented in Table 3. The correlation image seems to have less error than the other algorithms:

- correlation: 0.41
- LKST: 0.52

This ensures that the proposed algorithm minimizes the error.

4.4 *Paraboloid Implementation Comparison*

The presented paraboloid interpolation Algorithm 2 has been compared with four other implementations in terms of precision and resource usage.

Table 3 Calculated mean absolute error ($\langle|\epsilon|\rangle$) and mean squared error ($\langle\epsilon^2\rangle$).

Method	$\langle\|\epsilon\|\rangle$ (DN)	$\langle\epsilon^2\rangle$ (DN^2)
LKST	0.52	0.76
HS	0.70	1.79
Farneback	0.85	3.11
Correlation	0.41	0.45

The characteristics of these four implementations are:

- fixed-point implementation,
- fixed-point implementation using a successive approximation algorithm for the division,
- 32-bits floating-point implementation,
- 64-bits floating-point implementation.

All the implementations follow the same schema, with a final number of states ∼60. The first fixed-point implementation works with a big number of bits (∼150 bits in the present case) to avoid losing resolution. In the second case, the division requires a large number of periods for the same reason. The division is not optimized taking into account the results size. This would have implied the development of an algorithm with characteristics similar to a floating-point implementation, which is out of the scope of this study.

In all the cases, the size of the inputs $(z_{i,j})$ is 30 bits, whereas the outputs (x, y) are signed fixed-point with a fractional size of 8 bits. To minimize the rounding errors, 3 guard bits have been added to the calculations.

No precision is expected to be lost in case of fixed-point arithmetic (first two cases) as all the required bits are used for the calculations. In case of floating-point values, the precision will depend on the size of the mantissa. As the inputs have 30 bits size, the 32 bits floats could be in the limit with the 23 bits mantissa. However, the tests made show that the errors are smaller than the required output precision.

The total number of steps for every iteration of the proposed algorithm is nearly 50. This can be reduced if needed, introducing parallelization during the calculation of the a_{mn} coefficients or the $z_{i,j}$ values. The total number of steps needed for one calculation is the product of the number of steps in the loop by the required resolution, in this case 11 bits. Adding the overheads, this results in 583 steps or 583 clock cycles, to get the subpixel displacement. The other cases are indicated in Table 4.

This first test has been made with the five implementations (the one that uses the proposed Algorithm 2 and the four that use the direct Algorithm 4). A random test pattern has been generated for the input matrix cells. The random numbers were selected between 0 and the maximum value allowed by the cells. Afterward, the paraboloid coefficients a_{mn} were calculated using Eq. (7). To ensure that the coefficients describe a paraboloid and it had a minimum, the values of the coefficients a_{02} and a_{20} were checked to be positive. If not, a new set of random cell values were calculated. If the coefficients were correct, the values of x and y were calculated using Eq. (9). The resulting x and y were checked to be both in the interval $(-0.5, 0.5)$. If this was the case, the generated cells were used to test the different algorithms. If not, a new set of cells was generated. The pseudocode is presented in Algorithm 5.

The results using the described cells show that the precision is equivalent between the different algorithms with a maximum error of 1 less significant bit. This has been tested for outputs

Table 4 Synthesis comparison.

Implementation	Freq. (MHz)	Periods	Registers	Logic	DSP	Time (μs)
Proposed	152	583	1586	4449	0	3.8
Fixed	1.1	62	3634	101,765	16	56
Fixed with div.	90	660	5089	5865	16	7.3
Float32	13	62	1550	13,548	4	4.8
Float64	5.2	62	2711	54,605	33	12

Algorithm 4 Paraboloid minimum finding algorithm using direct equations

```
procedure PARABOLOID_MIN(z_{i,j})  ▷ Matrix cells
    Calculate a_mn substituting z_{i,j} in Eq. (7)
    Calculate x, y substituting a_mn in Eq. (9)
    return x, y
end procedure
```

Algorithm 5 Paraboloid cells generator

```
procedure PARABOLOID_CELLS(bits)  ▷ cells size
    max_value ← 2^bits − 1
    loop
        z_{i,j} ← random number ∈ (0, max_value)
        Calculate a_mn substituting z_{i,j} in Eq. (7)
    if a_02 < 0 ∪ a_20 < 0 then
        continue
    end if
    Calculate x, y substituting a_mn in Eq. (9)
    if x ∋ (−0.5, 0.5) ∪ y ∋ (−0.5, 0.5) then
        continue
    end if
    return z_{i,j}
end loop
end procedure
```

between 8 and 16 fractional bits. This is compliant with the application requirements (eight fractional bits).

A second test has been made synthesizing the five algorithms for the Virtex-4QV. First, the state machines have been translated to VHDL-2008. Then they have been synthesized using Synplify from Synopsys. Six parameters have been used to compare the results, which correspond to the columns of the Table 4:

1. maximum possible frequency,
2. number of periods needed by the state machine to reach the result,
3. registers occupation,
4. logic cells occupation,
5. DSP cells occupation,
6. total time required at the maximum expected frequency, where time $= \frac{\text{periods}}{\text{freq}}$.

It can be seen that the proposed solution is better in terms of maximum possible frequency, occupation (registers and logic), and total time. In case of a floating-point solution, the 32 bits one seems to be a good alternative for the present application. Regarding the fixed-point, an iterative division algorithm is needed to make it feasible. Finally, just mention that both, the float64 and fixed-point direct solution, do not fit inside the FPGA used in the project.

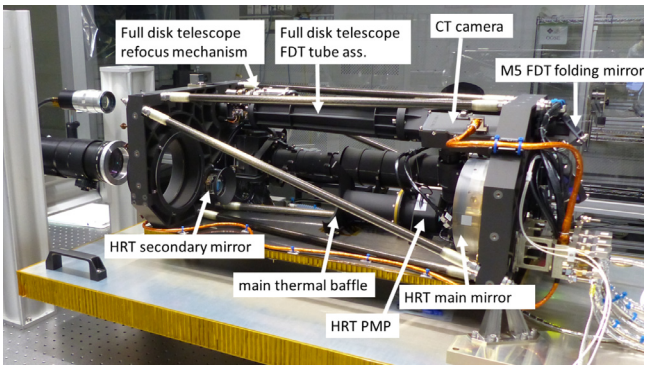

Fig. 20 Main subsystems as seen from the left side of SO/PHI. The CTC can be seen on the top right corner, while the TTM in the bottom left one. The TTM is labeled as HRT secondary mirror, the mechanism is behind it.

4.5 *Optical Test*

The system has been finally integrated inside SO/PHI, as shown in Fig. 20, and installed on top of an optical bench to determine its performance. This is the flight model (FM) that has already been delivered to ESA.

To measure the displacements, a pattern showing a crosshair target (Fig. 21) has been used. The pattern has a root mean square contrast around a 30%. The CTC takes 400 fps with a size of 64×64 px. The algorithm results are shown in Fig. 21.

It can be seen that the algorithm detects the optical bench noise, equivalent to 5 μm on the focal plane of the CTC.

5 Conclusions

An algorithm to estimate the jitter for the ISS of SO/PHI instrument in RT and its implementation has been presented. It provides subpixel estimation using paraboloid interpolation based on 2-D bisection. The proposed methodology has been tested with synthesized images, showing a mean error below $\frac{1}{25}$ pixel. Also the proposed paraboloid interpolation algorithm has been tested in terms of precision, resources, and time needed for the calculation. The algorithm surpasses the required 8-bits precision, being able to reach 16 bits of precision, and presents differences of ± 1 LSB with the other alternatives. The resources and time needed by the proposed algorithm are the least in both cases, with an improvement over 20%. Thanks to these optimizations, the total delay is the image read-out and a postprocessing of 50 μs, which includes the transmission of the new position to the TTC. This postprocessing delay is constant and independent of the image size. The system has been tested in a lab environment on subsystem level and within the SO/PHI FM. The full results of the calibration and performance test of the full image stabilization system will be published in a future paper.

Acknowledgments

This work has been funded by the Spanish MINECO through project ESP2015-66494-R, including a percentage from European FEDER funds. Disclosures: the authors have no relevant financial interests in the manuscript and no other potential conflicts of interest to disclose.

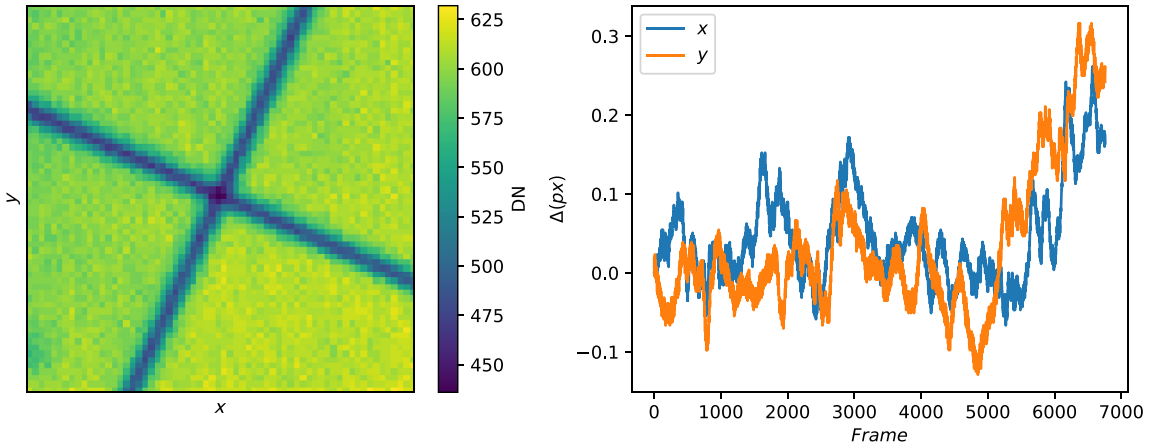

Fig. 21 Pattern used to calculate the displacements, seen by the CTC, and the displacements measured by the algorithm.

References

1. S. K. Solanki et al., "The polarimetric and helioseismic imager on Solar Orbiter," arXiv:1903.11061 [astro-ph.IM] (2019).
2. https://hinode.nao.ac.jp/uploads/2016/03/10/12_02.png, © NAOJ/JAXA (23 June 2019).
3. J. C. del Toro Iniesta, *Introduction to Spectropolarimetry*, Cambridge University Press, Cambridge (2003).
4. R. Sridharan et al., "An image stabilization system for solar observations," *Bull. Astron. Soc. India* **33**, 414–414 (2005).
5. B. Gelly et al., "Design and implementation of an image stabilization device at the THEMIS solar telescope," *Exp. Astron.* **22**(1), 67–85 (2008).
6. A. Title, "The SOUP and CIP instruments," *Adv. Space Res.* **4**(8), 67–74 (1984).
7. V. Domingo, B. Fleck, and A. I. Poland, "The SOHO mission: an overview," *Solar Phys.* **162**(1), 1–37 (1995).
8. J. Schou et al., "Design and ground calibration of the helioseismic and magnetic imager (HMI) instrument on the solar dynamics observatory (SDO)," *Solar Phys.* **275**, 229–259 (2012).
9. T. Berkefeld et al., "The adaptive optics system of the 1.5 m Gregor solar telescope: four years of operation," *Proc. SPIE* **9909**, 990924 (2016).
10. T. Shimizu et al., "Image stabilization system on SOLAR-B solar optical telescope," *Proc. SPIE* **5487**, 1199 (2004).
11. M. Carmona et al., "System model of an image stabilization system," *Proc. SPIE* **9150**, 91501U (2014).
12. J. Hirzberger and J. Woch, "PHI instrument requirements specification," Tech. Rep. SOL-PHI-MPS-DE2000-SP-1.1.2, Max–Planck–Institut für Sonnensystemforschung (2013).
13. D. Roma et al., "A space grade camera for image correlation," in *IEEE 13th Int. New Circuits and Syst. Conf. (NEWCAS)*, pp. 1–4 (2015).
14. https://hinode.nao.ac.jp/uploads/2016/03/10/17.png,© NAOJ/JAXA (23 June 2019).
15. A. Casas et al., "Design and test of a tip-tilt controller for an image stabilization system," *Proc. SPIE* **9911**, 991123 (2016).
16. W. Schmidt et al., "Auto alignment and image tracking system for the sunrise telescope," *Proc. SPIE* **6274**, 62740H (2006).
17. T. Shimizu et al., "Image stabilization system for Hinode (solar-b) solar optical telescope," *Solar Phys.* **249**(2), 221–232 (2008).
18. J. Decaluwe, "MyHDL: a python-based hardware description language," *Linux J.* **2004**, 5 (2004).
19. IEEE Std 1076-2008: IEEE Standard VHDL Language Reference Manual, IEEE (2008).
20. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Int. Joint Conf. on Artif. Intell.*, Vancouver, BC, Canada, pp. 674–679 (1981).
21. J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conf. Comput. Vision and Pattern Recognit (Proc. CVPR'94)*, IEEE, pp. 593–600 (1994).
22. B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.* **17**(1), 185–203 (1981).
23. G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds., pp. 363–370, Springer Berlin Heidelberg, Berlin, Heidelberg (2003).
24. J. Diaz et al., "FPGA-based real-time optical-flow system," *IEEE Trans. Circuits and Syst. Video Technol.* **16**, 274–279 (2006).
25. Z. Wei et al., "A fast and accurate tensor-based optical flow algorithm implemented in FPGA," in *IEEE Workshop on Appl. Comput. Vision (WACV '07)*, p. 18 (2007).
26. D. Bagni, P. Kannan, and S. Neuendorffer, "Demystifying the Lucas-Kanade optical flow algorithm with Vivado HLS," Tech. Rep. XAPP1300, Xilinx (2017).
27. F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Trans. Image Process.* **9**, 497–501 (2000).
28. Y. Keller and A. Averbuch, "Fast gradient methods based on global motion estimation for video compression," *IEEE Trans. Circuits and Syst. Video Technol.* **13**, 300–309 (2003).
29. Y. Su, M.-T. Sun, and V. Hsu, "Global motion estimation from coarsely sampled motion vector field and the applications," *IEEE Trans. Circuits and Syst. Video Technol* **15**, 232–242 (2005).
30. S. Erturk, "Digital image stabilization with sub-image phase correlation based global motion estimation," *IEEE Trans. Consum. Electron.* **49**, 1320–1325 (2003).
31. S. Kumar et al., "Real-time affine global motion estimation using phase correlation and its application for digital image stabilization," *IEEE Trans. Image Process.* **20**, 3406–3418 (2011).
32. H.-W. Park and H.-S. Kim, "Motion estimation using low-band-shift method for wavelet-based moving-picture coding," *IEEE Trans. Image Process.* **9**, 577–587 (2000).
33. M. G. Löfdahl, "Evaluation of image-shift measurement algorithms for solar Shack-Hartmann wavefront sensors," *A&A* **524**, A90 (2010).
34. A. M. Reza and R. D. Turney, "FPGA implementation of 2D wavelet transform," in *Conf. Rec. Thirty-Third Asilomar Conf. Signals, Syst. and Comput. (Cat. No.CH37020)*, Vol. 1, pp. 584–588 (1999).
35. X. Bing and C. Charoensak, "Rapid FPGA prototyping of Gabor-wavelet transform for applications in motion detection," in *7th Int. Conf. Control, Autom. Rob. and Vision (ICARCV 2002)*, Vol. 3, pp. 1653–1657 (2002).
36. K. Mei et al., "VLSI design of a high-speed and area-efficient JPEG2000 encoder," *IEEE Trans. Circuits Syst. Video Technol.* **17**, 1065–1078 (2007).
37. A. Pande et al., "Hardware architecture for video authentication using sensor pattern noise," *IEEE Trans. Circuits Syst. Video Technol.* **24**, 157–167 (2014).
38. ON Semiconductor, STAR1000 1M Pixel Radiation Hard CMOS Image Sensor (2015).
39. R. Smithson and T. Tarbell, "Correlation tracking study for meter-class solar telescope on space shuttle," Tech. Rep., Lockheed Missiles and Space Co. (1977).
40. S. S. Gleason, M. A. Hunt, and W. B. Jatko, "Subpixel measurement of image features based on paraboloid surface fit," *Proc. SPIE* **1386**, 135–144 (1990).
41. Z. Yi and R. L. Molowny-Horas, "Proc. LEST mini-workshop, software for solar image processing," LEST Tech. Rep. (56) (1992).
42. Xilinx, Space-Grade Virtex-4QV Family Overview (2014).
43. D. Roma et al., "APS fixed pattern noise modelling and compensation," in *Des. Circuits and Integr. Syst. Conf.* (2016).
44. D. Bishop, "Fixed- and floating-point packages for vhdl-2005," in *Desi. Verif. Conf. (DVCon)* (2005).
45. M. Parker, "High-performance floating-point implementation using FPGAs," in *IEEE Mil. Commun. Conf. (MILCOM 2009)*, pp. 1–5 (2009).
46. J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.* **EC-8**, 330–334 (1959).
47. W. Schmidt et al., "The 1.5 meter solar telescope GREGOR," *Astron. Nachr.* **333**(9), 796–809 (2012).
48. R. F. Stein and Å. Nordlund, "Simulations of solar granulation. I. general properties," *Astrophys. J.* **499**(2), 914–933 (1998).
49. G. Miller, "Efficient algorithms for local and global accessibility shading," in *Proc. 21st Ann. Conf. Comput. Graphics and Interact. Tech., SIGGRAPH '94*, ACM, New York, NY, USA, pp. 319–326 (1994).
50. PHI Instrument Team et al., "Experiment interface document Part B," Tech. Rep. SOL-PHI-MPS-MN1400-IF-2.3.0, MPS (2015).
51. G. Bradski, "The OpenCV library," *Dr. Dobb's J. Software Tools* (2000).

David Roma is an assistant professor of the Department of Electronic and Biomedical Engineering at the University of Barcelona (UB). He has worked as an electronics engineer for the SO/PHI instrument for SO mission, developing and testing the correlation camera hardware and firmware. He has been also deeply involved in the validation and software tools for the full image stabilization system. At the same time, he has also been doing, since 2015, his PhD with the UPC-IonSAT Group under the direction of Manuel Hernandez Pajares, with topics related to real-time ionospheric maps generation and validation. He has six published articles in peer-reviewed papers, six conference presentations and involved in three international projects related to space science and ionosphere. He is now working at the Institute of Space Sciences from the Spanish National Research Council.

Manuel Carmona is an associate professor of the Department of Electronic and Biomedical Engineering at UB. His research is focused on the modeling and simulation of systems in two areas: space and

microsystems. He is also involved in PA/QA space projects tasks. He is co-principal investigator of the UB contribution to the ISS for SO/PHI. He has worked in university and industry research projects.

Jose Bosch received his PhD in electronics and applied physics and his MSc degree in materials science from UB. He is a professor of the Department of Electronic and Biomedical Engineering at UB. From 1987 to 1993, his research field was the analysis and characterization of photoluminiscent semiconductors and quantum wells electronic-based devices. Since 1994, he has been working in the design of digital systems, mainly with microcontrollers and embedded systems, and smart instrumentation. He is a member of the team that developed the ISS for the SO/PHI instrument for SO mission.

Albert Casas is an assistant professor of the Department of Electronic and Biomedical Engineering at UB. He is also a member of the Institute of Space Studies of Catalonia (IEEC). In the last years, he has collaborated on designing the ISS for SO/PHI. Nowadays, he collaborates with Technical University of Catalonia, designing a new on-board computer platform for a nanosatellite, based on advanced FPGA technology.

Atila Herms received his PhD in physics from UB in 1984. He is a full professor of digital systems in the Department of Electronic and Biomedical Engineering, Faculty of Physics at UB. His field of research is digital systems design. He directs a research group dedicated to instrumentation and communication systems design specially for space applications. Now, he is the dean of the Faculty of Physics at UB.

Manel Lopez has been an associate professor of the Department of Electronic and Biomedical Engineering at UB since 2007. His research activity is devoted to the hardware system electronics designs, embedded software, and communications. He has mainly worked in the field of sensor networks and embedded systems. He had participated in the Spanish section of the ESA. His current research is focused in the characterization of sensors, design for instrumentation electronics, control electronics, and embedded systems.

Oscar Ruiz is an associate professor of the Electronics and Biomedical Engineering Department at UB and the current head of studies for the electronic engineering and telecommunication degree. His research is focused on the application of electronic instrumentation to biomedical tasks. He is also involved in dissemination activities in order to promote engineering careers, mainly STEM, among high school students.

Josep Sabater received his BS and MS degrees in computer science and computer engineering from the University Ramon Llull, Barcelona, Spain, in 2000 and 2001, respectively. Additionally, he received his MS degree in electronics from the UB, Spain, in 2010. In 2010, he joined the IEEC. Since then, he has been working on electronics and control software for satellites and ground telescopes. He was previously involved in the design of the ISS for SO/PHI. Furthermore, he is currently developing the control system for the cryogenic probe arms of MIRADAS instrument for Gran Telescopio Canarias (GTC). His current research interests include motion planning, robotics, control systems, and artificial intelligence.

Thorsten Maue is the SO/PHI ISS project engineer at the KIS. He completed his studies of aerospace engineering in 2006. With his experience in optical space instrumentation, he joined KIS in 2012 to help developing and qualifying the ISS for SO/PHI.

Reiner Volkmer received his PhD in physics and astronomy from the University of Göttingen, Germany. He was developed the data handling system for the IBIS instrument on the integral satellite at the University of Tübingen. He was a project manager for the development and construction of the 1.5-m solar telescope GREGOR at the KIS, Freiburg, Germany. Since 2012, he has been a project manager for the project of the ISS for the SO/PHI instrument at KIS.

Jose M. Gomez is an associate professor of the Department of Electronic and Biomedical Engineering at UB. He is also a member of the Institute of Cosmos Sciences at the University of Barcelona and the IEEC. His research is focused on the development of ground-based and space borne instrumentation for telescopes. He collaborates in the instrument MIRADAS for the GTC. He is a principal investigator of the UB contribution to the ISS for SO/PHI. He has also worked at IBM.

Biographies of the other authors are not available.