# A NEW SATELLITE IMAGERY STEREO PIPELINE DESIGNED FOR SCALABILITY, ROBUSTNESS AND PERFORMANCE

J. Michel[1],*E. Sarrazin[1], D. Youssefi[1], M. Cournet[1], F. Buffe[1], J.M. Delvit[1], A. Emilien[2], J. Bosman[2], O. Melet[1], C. L'Helguen[1]

[1] Centre National d'Etudes Spatiales (CNES), 18 avenue E. Belin, Toulouse cedex 9, France
[2] CS, 5 rue Brindejonc des Moulinais, Toulouse Cedex 5, France

**Commission II, WG II/2**

**KEY WORDS:** Image processing, Photogrammetry, Stereovision, Digital Surface Model, High Performance Computing

**ABSTRACT:**

This paper presents a new Multiview Stereo Pipeline (MVS), called CARS, dedicated to satellite imagery. This pipeline is intended for massive Digital Surface Model (DSM) production and has therefore been designed to maximize scalability robustness and performance. Those two properties have driven the design of the workflow as well as the choice of algorithms and parameter trends, making our pipeline unique with respect to existing solutions in literature. This paper intends to serve as a reference paper for the pipeline implementation, and therefore provides a detailed description of algorithms and workflow. It also demonstrates the pipeline robustness and stability in several use cases, and compares its accuracy with the state-of-the-art pipelines on a reference dataset.

## 1. INTRODUCTION

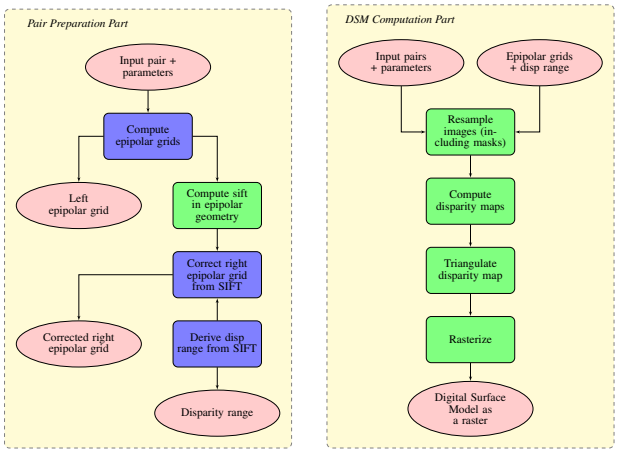
Using Very High Resolution (VHR) imagery to derive precise 3D models of the earth is of primary interest for the remote sensing community as shown by the multiple research challenges proposed during the last few years (Bosch et al., 2016, Bosch et al., 2019). In preparation for the CO3D mission (Lebegue et al., 2020), CNES is looking for a pipeline able to massively process stereoscopic acquisitions from existing Very High Resolution optical images such as Worldview3 or Pleiades, as well as upcoming ones. Such a generic pipeline should provide robustness in order to cope with different landscapes (urban environment, mountainous areas ...), images defects (clouds or poor signal to noise ratio ...), and sensors. It should be able to run on many nodes at once on a High Performance Computing or Cloud infrastructure, with a linear relationship between processing time and available resources. It should also provide user parameters with physical meaning to ease parameters setting during massive production. After a review of existing pipelines in the literature, which is briefly presented section 1.1, we found that none of them were fully meeting those criteria, and decided to develop a new pipeline, soon to be released as open-source software. The aim of this paper is to serve as a reference paper for this new pipeline, introduced in section 1.2.

### 1.1 Review of existing pipelines

In literature, several automatic stereo pipeline for processing DSM from satellite imagery have been published: ASP (Beyer et al., 2018), CATENA (Krauß et al., 2013), , MicMac (Rupnik et al., 2017), RSP (Qin, 2016), S2P (de Franchis et al., 2014b) and SETSM (Noh, Howat, 2017). All those pipelines share the same steps, which are pre-processing (bundle adjustment, disparity interval determination ...), image resampling in geometry compatible with image matching, dense image matching, triangulation and rasterisation. Most of them propose several processing options, such as multiview or multiview stereo (pairwise processing) mode, and 1D or 2D dense image matching based on variations of the SGM algorithm (Hirschmuller, 2008). Their main features are listed in Tab.1 (except from SETSM which implements a different method). They also offer additional post-processing steps like DSM fusion or color image generation. In (Bosch et al., 2017), the performance of some of them have been compared in the frame of the IARPA challenge.

None of these solutions fully fulfill our requirements in terms of license compatibility or capacity to make the best use of CNES computing infrastructure for operational use in the frame of the CO3D mission for instance.

### 1.2 Proposed pipeline




(a) Prepare part (b) Compute DSM part

Figure 1. Workflow for the two steps of CARS. Green blocks are distributed and blue blocks are sequential

CARS performs MultiView Stereo (MVS), meaning that multiple views are grouped and processed by pairs by means of epipolar resampling and 1D dense matching (also called dis-

*Corresponding author

| | Features | ASP | CATENA | MicMac | RSP | S2P | Proposed pipeline |
|---|---|---|---|---|---|---|---|
| *Mode* | Multiview Stereo mode | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Multiview mode | | | ✓ | | | |
| *Step: Pre-processing* | Bundle adjustment | ✓ | ✓ | ✓ | ✓ | | |
| *Step: Image resampling* | Epipolar geometry | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Related to reference image | ✓ | ✓ | ✓ | | | |
| *Step: Image matching* | 1D dense matching | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | 2D dense matching | ✓ | ✓ | ✓ | | | |
| *Step: Triangulation* | Pairwise triangulation | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Multiview triangulation | ✓ | | ✓ | | | |
| *Step: Rasterisation* | Point cloud fusion | ✓ | | ✓ | | ✓ | ✓ |
| | DSM fusion | ✓ | ✓ | ✓ | ✓ | ✓ | |

Table 1. Features comparison for existing stereo pipelines, established by reading reference papers and software manuals.

parity map). It should be noted that bundle adjustment is excluded from the scope of our pipeline, as it is performed beforehand by an in-house tool. However, our pipeline is tolerant to lack of or imprecise bundle adjustment. Pair-selection is considered application-dependent and therefore also excluded. Elements about the strategy that will be used in CO3D can be found in (Lebegue et al., 2020).

CARS is composed of well known steps in MVS pipelines, and its originality resides in the arrangement of those steps in the workflow as well as on some original algorithms. The pipeline overview is presented Fig. 1. The philosophy of the worflow is to regroup all steps that either require a global view on the data or that may require data-induced failure and degraded cases in a preliminary step called the Pair Preparation Step. By doing so, we ensure that the DSM Computation step, which is the most intensive, does not have to handle branches or corner cases and is therefore very reliable. The policy at hand is that if data-induced failures (corrupted data, lack of sparse matches, ...) happen, they must occur very early in the processing.

The Pair Preparation Step is run pair by pair and mainly produces refined epipolar resampling grids and the estimation of the disparity range. The DSM Computation Step processes the output of the Pair Preparation Step for several pairs and computes a unique DSM from them.

The remaining of the paper is organized as follows. Section 2 describes algorithms used at each step. Original algorithms are extensively described, while more standard ones are only referenced, the emphasis being on how they are used. Section 3 gives the macroscopic view of the pipeline, e.g. how the algorithms are implemented and how they can be distributed on many computing nodes. Section 4 presents several use cases to demonstrate the stability and efficiency of the pipeline, as well as an accuracy assessment based on the well-known IARPA dataset (Bosch et al., 2016).

## 2. ALGORITHMIC GROUNDS

This section will present the mathematical formulation of all algorithms, as well as illustrations based on cases. In this section, $(x, y)$ denotes image coordinates, $h$ an elevation and $(\lambda, \phi)$ a pair of latitude and longitude.

### 2.1 Localisation and colocalisation functions

Let $f : (x, y, h) \mapsto (\lambda, \phi)$ and $f^{-1} : (\lambda, \phi, h) \mapsto (x, y)$ respectively denote the forward localisation function, that maps image coordinates and given elevation above ellipsoid to latitude and longitude coordinates and the inverse localisation function. Those functions may be implemented using a physical sensor model based on orbit, attitude and focal plane, or using the Rational Polynomial Coefficients (Baltsavias, Stallmann, 1992) that comes with most Very High Resolution data. They may come from raw sensor metadata or a preliminary bundle adjustment step. If the elevation of the $(x, y)$ image point is not known (which is usually the case), and if a low resolution Digital Terrain Model $DTM(\lambda, \phi)$ of the scene is available, $h$ can be retrieved by an iterative procedure, forming the $f^{\star}(x, y) \mapsto (\lambda, \phi, h)$ function.

Given two images, it is therefore possible to define a colocalisation function $f_{1\mapsto 2}(x_1, y_1, h) \mapsto (x_2, y_2)$ that maps image coordinates from image 1 to image 2 as shown in Eq. 1. $f_{2\mapsto 1}$ can be defined in a similar way.

$$f_{1\mapsto 2}(x, y, h) = f_2^{-1}(f_1(x, y, h)) \tag{1}$$

We also assume that a low resolution Digital Terrain Model $DTM_{lr}(\lambda, \phi) \mapsto h$ is available and defined everywhere, giving a coarse estimate of the height above the ellipsoid.

$f_{1\mapsto 2}(x_1, y_1, h), h \in [h_{min}, h_{max}]$ describes the epipolar curves yielded by point $(x_1, y_1)$ from image 1 in image 2 within altitude range $[h_{min}, h_{max}]$.

### 2.2 Stereo-rectification of satellite images

Stereo-rectification transforms the images so that displacement corresponding to variation of $h$ in the colocalisation function only occur in the horizontal direction, forming the epipolar geometry. Epipolar geometry is perfectly defined for pinhole cameras, and modeled by an affine transform which can be estimated from camera parameters or tie points (Hartley, Zisserman, 2004). For push-broom images, which is the most used acquisition mode for Very High Resolution Spatial imagery, an epipolar geometry does not strictly exist (Habib et al., 2005). However, it is possible to approximate such a geometry, for instance by considering that the camera is locally affine (de Franchis et al., 2014a). Yet, the local affine approximation is only valid for a few hundreds of pixels and one can not define a global epipolar geometry for full scene spanning several tenths of thousands pixels. Other methods consist in applying a piece-wise approach to resample image in epipolar geometry (Oh et al., 2010)(Koh, Yang, 2016).

In our pipeline, we use a novel method based on a non rigid iterative approximation of a geometry fitting the epipolar constraint, by means of the colocalisation function introduced in

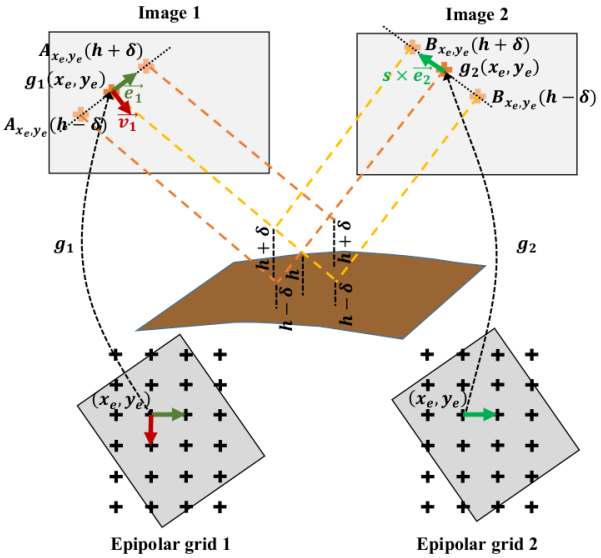



Figure 2. Epipolar grids are constructed iteratively by moving along local epipolar curves

Eq. 1. To that end, we will jointly recursively estimate two resampling grids $g_1(x_e, y_e) \mapsto (x_1, y_1)$ (resp. $g_2$) mapping from the estimated epipolar geometry to the input images. Those grids can be used along with an interpolation function to resample images to the epipolar geometry, or to convert coordinates between epipolar and sensor geometries.

**2.2.1 Moving along lines** Let assume that we have computed $g_1$ and $g_2$ for a given epipolar grid position $(x_e, y_e)$. Equations 2 and 3 show how to compute values for the next position in line $(x_e + 1, y_e)$, i.e. in the epipolar direction.

$$g_1(x_e + 1, y_e) = g_1(x_e, y_e) + \overrightarrow{e_1}(g_2(x_e, y_e)) \quad (2)$$
$$g_2(x_e + 1, y_e) = g_2(x_e, y_e) + s * \overrightarrow{e_2}(g_1(x_e, y_e)) \quad (3)$$

$\overrightarrow{e_1}$ (resp. $\overrightarrow{e_2}$) is a unit vector tangent to the epipolar curve in image 1 (resp. 2) at location derived from $g_2(x_e, y_e)$ (resp. $g_1(x_e, y_e)$), as shown in Fig. 2. Using notations introduced in equations 4 and 5, their formulations are given by equations 6 and 7, where $h$ is the altitude and $\delta$ is a small elevation difference used to estimate the local tangent.

$$A_{x_e,y_e}(h) = f_{2\mapsto 1}(g_2(x_e, y_e), h) \quad (4)$$
$$B_{x_e,y_e}(h) = f_{1\mapsto 2}(g_1(x_e, y_e), h) \quad (5)$$
$$\overrightarrow{e_1}(g_2(x_e, y_e)) = \frac{\overrightarrow{A_{x_e,y_e}(h-\delta)A_{x_e,y_e}(h+\delta)}}{\|\overrightarrow{A_{x_e,y_e}(h-\delta)A_{x_e,y_e}(h+\delta)}\|} \quad (6)$$
$$\overrightarrow{e_2}(g_1(x_e, y_e)) = \frac{\overrightarrow{B_{x_e,y_e}(h-\delta)B_{x_e,y_e}(h+\delta)}}{\|\overrightarrow{B_{x_e,y_e}(h-\delta)B_{x_e,y_e}(h+\delta)}\|} \quad (7)$$

In Eq. 3, $s$ is a scale factor ensuring that displacement in image 1 and 2 are equivalent, as defined in Eq. 8.

$$s = \|\overrightarrow{B_{x_e,y_e}(h)B_{x_e+1,y_e}(h)}\| \quad (8)$$

$h$ is the altitude for which disparity (e.g. horizontal displacement between left and right image) will be null. $h$ can be a fixed value for the whole grids, or sampled from the low resolution DTM using Eq. 9. This yields an epipolar geometry where disparity only accounts for elevation differences with the low resolution DTM, as highlighted by Fig. 3. In such a geometry, the disparity range (and thus processing time needed) to retrieve elevation of the scene is drastically reduced. In our pipeline, we systematically generate epipolar grids with a low resolution DTM.

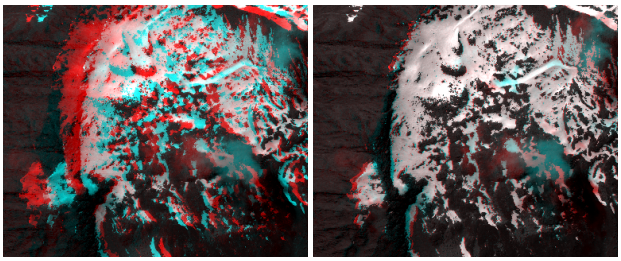


Figure 3. Anaglyphs generated from epipolar grids from PHR Mont-Blanc case (see section 4.1), with $h$ as an average elevation of 2319 meters (left) and with $h$ sampled from SRTM at 30 meters resolution (right). One can clearly see the reduced disparity between both cases.

$$h(x_1, y_1) = DTM_{lr}(f_1^\star(x_1, y_1)) \quad (9)$$

**2.2.2 Moving to next line** We also need to find the start of next line in epipolar geometry, which is given by Eq. 10, where $\overrightarrow{v_1}$ is the unit vector orthogonal to epipolar direction and has the following properties: $(\overrightarrow{v_1}, \overrightarrow{e_1}(g_2(0, y_e))) = \frac{\pi}{2}$ and $\|\overrightarrow{v_1}\| = 1$, as illustrated in Fig. 2. To find corresponding starting point in $g_2$, we can observe that disparity in epipolar geometry should be null at $h$, as shown in Eq. 11.

$$g_1(0, y_e + 1) = g_1(0, y_e) + \overrightarrow{v_1} \quad (10)$$
$$g_2(0, y_e + 1) = f_{1\mapsto 2}\big(g_1(0, y_e + 1), h(g_1(0, y_e + 1))\big) \quad (11)$$

**2.2.3 Starting point and grid sizes** To be able to fully compute $g_1$ and $g_2$, we need to determine starting point $g_1(0, 0)$ as well as the number of lines and columns of the epipolar geometry. This is done by estimating an average affine transform at full image scale to make epipolar lines horizontal while preserving images resolution. We then use it to transform corners of image 1 and use the bounding box of the result to derive starting point and size.

**2.2.4 Scale and baseline ratio** Note that in equations 2 to 11, the grids are computed with a step of 1 pixel, but they can be computed with a coarser step. Note that it is also possible to compute a grid to map epipolar geometry to terrain positions at a coarse altitude using $f_1^\star$.

One interesting thing that can be computed is the ratio $\alpha$ between variations of altitude and disparity in epipolar geometry, in meters per pixel, as expressed in Eq. 12, where $r$ is the image resolution and $B/H$ is the classical baseline ratio. It is not expected to vary much in the grid, since it relates to imaging conditions and not to image content. In our pipeline, we use the average value over the grid when we need to convert between disparities and elevation differences.

$$\alpha(x_e, y_e) = \frac{2 * \delta}{\|\overrightarrow{A_{x_e,y_e}(h-\delta)A_{x_e,y_e}(h+\delta)}\|} = \frac{r}{B/H} \quad (12)$$

### 2.3 Sparse matching in epipolar geometry

For the next steps in our pipeline, we need fast and accurate matches between both images, but we do not require matches to be dense. We naturally came to the SIFT algorithm introduced in (Lowe, 2004). We use the standard matching procedure based on ratio between first and second closest descriptors according to Euclidean distance, and also ensure that the same match is detected in both matching direction (from image 1 to image 2 and reversely).

This matching procedure has a quadratic cost of $O(N_1 * N_2)$ where $N_1$ and $N_2$ is the number of points detected in image 1 and image 2 respectively. In our case, those numbers can be huge (up to several millions of points) if we match both full images. In order to perform matching at full image scale in a reasonable time, we will use the epipolar geometry presented in section 2.2, which has the following advantages:

- In generated epipolar geometry, valid matches are located on the same images rows, up to the geometric sensor model precision,
- In generated epipolar geometry, points located at the altitude of the low resolution DTM are located on the same images columns (null disparity). Points whose altitude is below $\delta_h$ of the low resolution DTM are located with $\frac{\delta_h}{\alpha}$ columns in right image with respect to left image.

For a given epipolar region $R_1 = [x_m, x_M] \times [y_m, y_M]$ of image 1, we can therefore define the corresponding region in image 2 according to three user defined parameters:

- $\epsilon$, the ortho-epipolar error upper bound, in pixels (this is related to a priori knowledge on sensor, and reflects how well lines of sight intersect each other),
- $\delta_m^h$ and $\delta_M^h$, which are the expected signed minimum and maximum elevation differences with respect to the low resolution DTM, in meters.

With those parameters, corresponding region $R_2$ in image 2 is given by $R_2 = \left[x_m + \delta_m^h/\alpha, x_M + \delta_M^h/\alpha\right] \times \left[y_m - \epsilon, y_M + \epsilon\right]$.

In order to compute the complete set of matches, we split image 1 domain in non-overlapping tiles, derive the corresponding (overlapping) tiles using equation above, perform matching on all pairs of tiles separately, and gather all matches into a single matches set. By doing so, we ensure that we only match sets of a few thousands key-points for each tile. Matching for each tile can also be performed completely independently, making the whole process completely scalable. It is noteworthy that $\delta_m^h$ and $\delta_M^h$ can be greatly relaxed by using multiple right tiles for a single left image tile so as to span the full range.

Let $S = \left\{\left((x_{e1}, y_{e1}), (x_{e2}, y_{e2})\right)\right\}$ denote the complete set of $N_S$ matches obtained from this procedure. In order to eliminate obvious outliers, we filter those matches according to the user defined parameters presented earlier. First, we discard points that exhibit a strong epipolar error $|y_{e2} - y_{e1}| > \epsilon$. Then we discard matches that exhibit a strong disparity $(x_{e2} - x_{e1}) \notin \left[\delta_m^h/\alpha, \delta_M^h/\alpha\right]$.

Figure 4. Detail of left epipolar image (left), right epipolar image before correction (center) and right epipolar image after correction (right), from the PHR Mont-Blanc case (see section 4.1). The horizontal red line helps noticing the 1.39 pixel epipolar error that we are able to perfectly re-align.

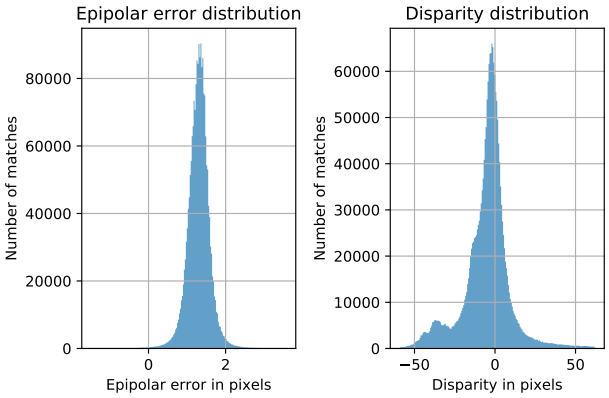



Figure 5. Distributions of epipolar error before geometry refinement (left) and disparity after geometry refinement (right) from sparse matches, computed on full epipolar images of the PHR Mont-Blanc case (see section 4.1).

### 2.4 Refinement of epipolar geometry

As stated in previous section, our epipolar geometry is subject to the imprecision of sensor modeling, that may result in a misalignment of epipolar lines, as shown in Fig. 4. This epipolar error may impact the performance of the disparity map computation algorithm used downstream, as this algorithm will only look for corresponding pixels on the same epipolar image lines. This effect and the result of the proposed correction in this section are illustrated Fig. 4. Fig. 5 shows an example of statistical distribution of the epipolar error estimated from matches.

Following the approach proposed in (de Franchis et al., 2014b), we are going to use our matches to estimate a correction of the epipolar geometry so as to get the best alignment we can. However, in this related work the authors use the affine approximation of the push-broom geometry, and the epipolar geometry is different for each tile. They therefore need to estimate the alignment correction locally, whereas in our case we can estimate a global correction model for the full scene. This correction will be directly incorporated in $g_2$, the epipolar rectification grid for image 2.

We start by forming the epipolar error vector in image 2 geometry, as shown in Eq. 13, by noting that matches with non epipolar error would have $y_{e1} = y_{e2}$.

$$E = \{g_2(x_2, y_1) - g_2(x_2, y_2), (x_1, y_1, x_2, y_2) \in S\} \quad (13)$$

We then perform a least-square fitting on $E$ of the two components correction model $c_2(x_{e2}, y_{e2})$. In our work, $c_2$ is a bi-linear model, but more complex models could be used if needed, in

order to account for sensor vibrations for instance. Finally, we update the stereo-rectification grid by means of Eq. 14.Note that we also apply the correction to $S$, forming a set of filtered and corrected matches $S^\star$.

$$g_2^\star(x_{e2}, y_{e2}) = g_2(x_{e2}, y_{e2}) + c_2(x_{e2}, y_{e2}) \quad (14)$$

### 2.5 Disparity range estimation

Since our epipolar geometry integrates a low resolution DTM, the disparity only accounts for the elevation difference with respect to this DTM. In our pipeline, we therefore estimate a fixed disparity range for the whole scene to be explored by the disparity map computation algorithm. A good estimation is important to optimize time and performance of the disparity map computation step.

To estimate this range in a reliable way, we further filter $S^\star$ corrected matches to eliminate matches that still exhibit an epipolar error of 3 times the standard deviation of the epipolar error on corrected matches. The rationale for this is that those points will likely not be on the same line in epipolar geometry, and are therefore unreachable for the disparity map computation algorithm. The distribution of disparity is derived from filtered corrected matches $S^\star_{filtered}$ by Eq. 15 and is shown in Fig. 5.

$$D = \left\{(x_{e2} - x_{e1}), (x_{e1}, y_{e1}, x_{e2}, y_{e2}) \in S^\star_{filtered}\right\} \quad (15)$$

To eliminate any remaining outliers, $d_m$ is set to 0.01% lower percentile of $D$, and $d_M$ is set to 0.01% upper percentile of $D$. An additional margin depending on the disparity range width is added, so that the final disparity range is $[d_m - 0.25 * (d_M - d_m), d_M + 0.25 * (d_M - d_m)]$.

### 2.6 Disparity map computation

We are using state-of-the art procedure, implemented in a separate tool called Pandora, which calculates a cost volume from the two images using census (Zabih, Woodfill, 1994), optimizes the cost volume with Semi-Global Matching (Hirschmuller, 2008) and select at each pixel the disparity associated with the minimum cost. Disparities are then refined to sub-pixel precision using vfit (Haller et al., 2010) and post-processed with median filter and cross-checking (Fua, 1993). More details on Pandora can be found in (Cournet et al., 2020). Let $d_{1\mapsto2}(x_e, y_e)$ denote the measured disparity.

### 2.7 From disparity to 3D points

From disparity map $d_{1\mapsto2}$ we can use $g_1$ and $g_2$ to obtain a list of homologous points in the sensor image geometries, as shown in Eq. 16.

$$H(x, y) = \Big(g_1(x_e, y_e), g_2\big(x_e + d_{1\mapsto2}(x_e, y_e), y_e\big)\Big) \quad (16)$$

Note that since epipolar alignment correction presented in section 2.4 is integrated in $g_2$, this mapping undoes the correction and so the homologous points are not affected. Note that a preliminary bundle-adjustment step is required to get the best performance from this procedure with several pairs.

One can then use $f_1$ and $f_2$ to generate 3D lines $(P^1_{h_m} P^1_{h_M})$ and $(P^2_{h_m} P^2_{h_M})$, by using two different altitude values $h_m$ and $h_M$, as shown in equations 17 to 18.

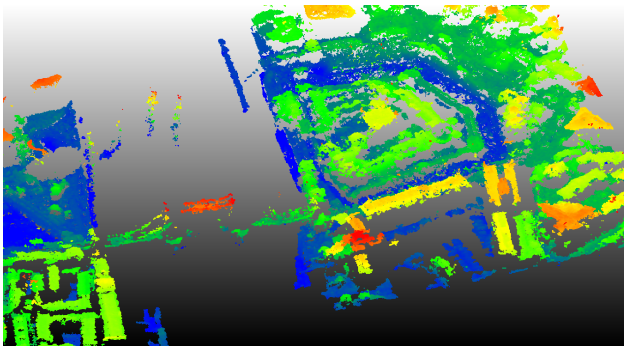


Figure 6. 3D point cloud generated with our pipeline on the PHR London dataset (see section 4.1). On can clearly see the Tower Bridge and Tower of London castle. Note that a water mask has been used over the Tames and nearby basins.

$$P^1_h(x_e, y_e) = \Big(f_1\big(g_1(x_e, y_e)\big), h\Big) \quad (17)$$

$$P^2_h(x_e, y_e) = \Big(f_2\big(g_2(x_e, y_e)\big), h\Big) \quad (18)$$

Those lines are then intersected by looking for the 3D point closest to both lines, which is the final 3D measurement. Note that this computation is done in ECEF coordinates for the sake of numerical precision, and converted back to WGS84. By applying this intersection to all computed disparities, we form a 3D point cloud $P = \left\{(\lambda, \phi, h)_k\right\}$, as shown in Fig. 6.

### 2.8 From 3D points to raster elevation maps

The purpose of this step, called rasterisation, is to convert computed 3D point clouds into a geolocated raster. To do so, we employ the same Gaussian weighting algorithm as in (de Franchis et al., 2014b). The 3D point cloud is converted to the target coordinate system (UTM for instance). Then we define a regular terrain grid using the user-defined resolution $res$. For a given cell center $(c_x, c_y)$, we look for the set of contributing points $C$ with a distance $d_c(x, y) = \|(x, y) - (c_x, c_y)\|$ lower than a user defined multiplier $k$ of the resolution, as defined in Eq. 19.

$$C = \left\{(x, y, h) \in P, d_c(x, y) < k * res\right\} \quad (19)$$

Estimated value of elevation at cell center $h^\star(c_x, c_y)$ is derived by Gaussian weighting with respect to distance of points to cell center, as shown in Eq. 20. Obtained raster map is shown in Fig. 7. It is noteworthy that during this step we can also derive cell-based quality indices such as the number of contributing points or the standard deviation of $h$ for contributing points.

$$h^\star(c_x, c_y) = \frac{1}{\sum\limits_C e^{-\frac{d_c(x,y)^2}{2\sigma^2}}} \sum_C h \times e^{-\frac{d_c(x,y)^2}{2\sigma^2}} \quad (20)$$

## 3. IMPLEMENTATION AND WORFLOW

### 3.1 Implementation

CARS provides a command-line tool based on a Python API which provides functions to perform each step. This API makes use of XArray (Hoyer, Hamman, 2017) to model and exchange

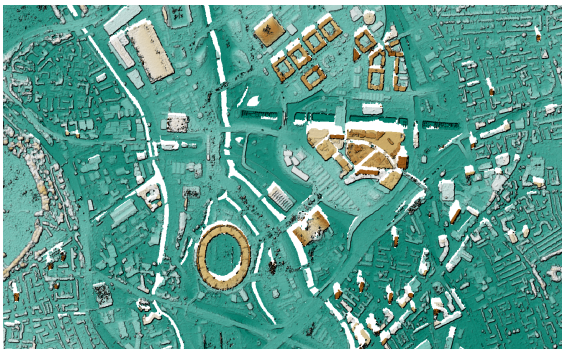


Figure 7. Detail of the final DSM for the PHR London case (see sec. 4.1). Elevation ranges from 42 m. (green) to 92 m. (brown).

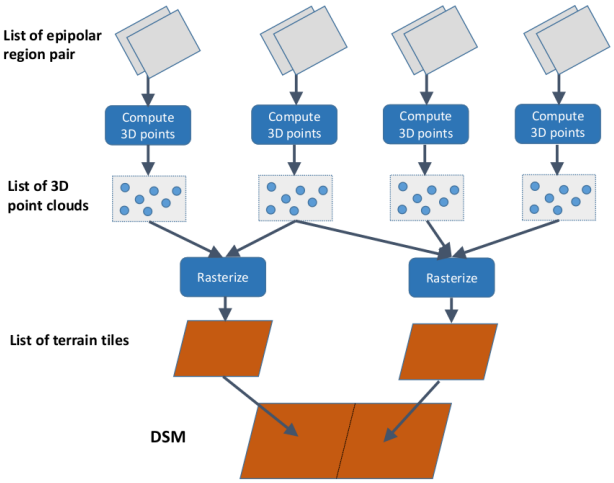



Figure 8. Dependency graph between terrain and epipolar tiles

data. It also makes use of the full Python stack, along with rasterio (Gillies et al., 2013-) to handle reading and writing of raster files, Fiona and Shapely to read and write vector files, Orfeo ToolBox for image processing (Grizonnet et al., 2017), and VLFeat for SIFT computation (Vedaldi, Fulkerson, 2008). It is mainly written in Python, with a few parts written in C++. Dask (Dask Development Team, 2016) is used to distribute the worflow on our High Performance Computing infrastructure. It allows to model dependencies as a task graph which is lazily evaluated and dynamically distributed among workers spawned on a HPC or cloud infrastructure.

### 3.2 Distributed workflow

**3.2.1 Pair Preparation Step** The worflow for the Pair Preparation Step is organized as presented in Fig. 1. Steps are run sequentially, except for the computation of sparse matches. This step is performed tile-wise and distributed among Dask workers, as explained in section 2.3.

**3.2.2 DSM Computation Step** To implement the workflow of the DSM Computation part, shown in Fig. 1, we first split each input stereo pair into regular epipolar tiles. We also derive the terrain area that should be produced. The bounding box for terrain area is either derived from the total area covered by input pairs or by user defined parameters. This terrain area is then split into regular terrain tiles.

For each terrain tile, we then look for all contributing epipolar tiles from all input pairs, and derive a task dependency graph presented in Fig. 8. Next, this task graph is submitted to Dask for distributed and lazy evaluation. Completed terrain tiles are written on the flow to final DSM file as they become available on Dask cluster.

## 4. RESULTS AND PERFORMANCE ANALYSIS

### 4.1 Cases study

In order to demonstrate that our pipeline is robust and generic, we applied it to a representative set of data, including 3 different VHR sensors, different landscapes, and pairs as well as triplets:

- Pair of Pleiades images over London, United Kingdom of size 38 881 × 21 197 pixels acquired on 2012.09.03,
- Pair of Pleiades images over Mont Blanc, France of size 39 545 × 51 979 pixels acquired on 2017.06.10,
- Triplet of Pleiades images over Napier, New Zealand of size 40 000 × 84 188 pixels acquired on 2013.02.08,
- Triplet of SPOT7 images over Alps, France of size 38 609 × 28 553 pixels acquired on 2016.09.30.
- Pair of Worldview3 images over Buenos Aires, Argentina, of size 43 008 × 36 864 acquired on 2015.12.18 (from the IARPA dataset (Bosch et al., 2016)).

The resolution of output DSM has been set to 0.5 meters for Pleiades cases, 0.3 meters for the Worldview3 case and 2 meters for the SPOT7. The spatial reference system is the local UTM zone. Cases involving triplet of images have been processed by forming two pairs sharing the middle view. Those pairs were independently processed by the pair preparation step, and then merged during the DSM computation step. For all experiments we used SRTM at 30 meters resolution as the initial low resolution DTM, except from the SPOT7 Alps cases, were we used SRTM at 90 meters because of the poor quality of the 30 meters version over the area. $\epsilon$ is set to 10 pixels for all experiments, while $\delta_m^h$ and $\delta_M^h$ are set to respectively $\pm 50$ meters, except for the PHR Mont Blanc case, where it is set to $\pm 100$ meters, and the SPOT7 Alps case, where it is set to $\pm 150$ meters. The wider ranges for those last cases is due to the lesser accuracy of SRTM 30 meters and SRTM 90 meters over mountainous areas with snow cover and glacier. Note that those values were chosen arbitrarily for the sake of the demonstration. Thanks to the multiple tiles strategy described in 2.3, we can actually use a much larger range if needed. The rasterisation of point cloud is done with $k = 1$ and $\sigma = 0.3$ for all experiments.

All experiments have been conducted on our in-house High Performance Computing center, using a master node of 24 CPUs and 120 Go of RAM, and a dask cluster of at most 200 workers, each with 4 CPUs and 20 Go of RAM. Resources allocation with Dask is dynamic: processing begins as soon as one worker is available, and new workers are spawned when they become available. It is therefore not possible to directly compare total processing time between runs, and following timings are only indicative. For the pair preparation step, total processing time ranges from 9 minutes for the PHR London case to 39 minutes for the PHR Napier pair 2 case. For the DSM computation step, total processing time ranges from 16 minutes for the PHR London case to 2 hours and 30 minutes for the SPOT7 Alps case. In addition to dynamic resources allocation, processing

| Case | Epipolar size | $\alpha$ (m/p) | Nb. matches | disp. range (p) | raw epi. err. (p) | DSM Size |
|---|---|---|---|---|---|---|
| WV3 Buenos Aires | 45 746 × 50 317 | 2.50 | 2 038 795 | [-21.2, 13.9] | -0.802 | 48 888 × 40 198 |
| PHR London | 31 147 × 43 974 | 1.29 | 458 246 | [-52.8, 37.9] | 0.437 | 41 119 × 22 782 |
| PHR Mont Blanc | 59 574 × 50 997 | 1.54 | 2 087 706 | [-49.7, 49.9] | 1.295 | 41 576 × 52 533 |
| PHR Napier Pair 1 | 93 064 × 59 754 | 2.42 | 2 688 452 | [-24.5, 24.3] | 0.469 | 45 070 × 92 047 |
| PHR Napier Pair 2 | 92 921 × 59 172 | 2.92 | 5 103 475 | [-19.5, 21.0] | 0.031 | |
| SPOT7 Alps Pair 1 | 37 938 × 44 526 | 4.78 | 1 037 189 | [-152.9, 154.5] | -0.017 | 31 131× 24 541 |
| SPOT7 Alps Pair 2 | 37 896 × 44 499 | 8.4 | 2 406 123 | [-88.6, 87.3] | 0.078 | |

Table 2. Main parameters estimated during pipeline execution. Collection of 3 images are processed by forming to pairs and merging them during DSM computation step. Number of matches is the raw number of matches found before filtering. Epipolar error is the mean epipolar error before correction (value after correction is not given, since it is always almost zero)

time changes according to input images size, number of pairs to process and estimated disparity range.

Tab.2 gives an overview of the main parameters of the pair preparation and DSM computation steps. One can note that all cases involve full size images, with more than one billions of pixels. $\alpha$ ranges from 1.29 to 2.92 meters per pixel, except for the SPOT7 Alps case, which is expected since input images resolution is 4 times larger. Number of matches found for the full epipolar images ranges from around 500 000 to more than 5 million, which gives a very strong population to estimate global disparity range and epipolar error. Mean epipolar error absolute value before correction is always below 1.5 pixels, which is excepted thanks to the good accuracy of attitude estimation for those sensors. This also validates our value for $\epsilon$. It is interesting to note that the global disparity range estimated for the full epipolar images is quite narrow, with only a few tenth of pixels in both side, except for the SPOT7 Alps pair 1, which is explained both by its smaller $\alpha$ value and larger values of $\delta_m^h$ and $\delta_M^h$.

### 4.2 Accuracy assessment on IARPA challenge data subset

In order to assess the performance of the proposed pipeline, we selected a subset of the well-known IARPA dataset. We selected the six Worldview3 images acquired on 18th December 2015 on the same track. It is noteworthy that since the proposed pipeline does not perform any bundle adjustment, we used an in-house tool to bundle adjust the images beforehand, without ground control points. We then paired one of the center view (4th image in the sequence) with the five remaining ones, forming five pairs of images, and ran the pair preparation step at full image scale for each pair. Note that pair $(4, 3)$ has been discarded since it exhibits strong oscillations of line of sights, along with a narrow viewing angle ($\alpha = 2.49m/p$).

Finally, we ran the compute DSM step with the four pairs simultaneously processed and raterised. In order to increase the coherency between the four pairs, we used a mode that does not invert the epipolar correction during triangulation step. This is easily done by using $g_2$ instead of $g_2^\star$ when converting disparity to right sensor image geometry. Other parameters are the ones used for the *WV3 Buenos Aires case* presented in section 4.1.

We used the algorithm presented in (Nuth, Kääb, 2011) to register the produced DSM with the lidar reference raster, yielding a shift of 1.1 meters in x and 0.26 meters in y direction. Difference between registered DSM and reference lidar is presented in Fig. 9. One can see that the main differences are caused by vegetation changes and inaccurate building edges. It is noteworthy that the produced DSM has 0.29 % of missing values.

We then measured and compensated a mean altimetric shift of $-0.98$ meters. With those corrections, we measured the accuracy with the metrics proposed in (Bosch et al., 2017), and compared them to the best results obtained in the IARPA challenge. As shown in Tab. 3, performances of the proposed pipeline are in the same range as those obtained during the challenge, using only five of the fifty images, which might explain the slightly better RMS and median errors. Nevertheless, this demonstrates the accuracy of the proposed method.

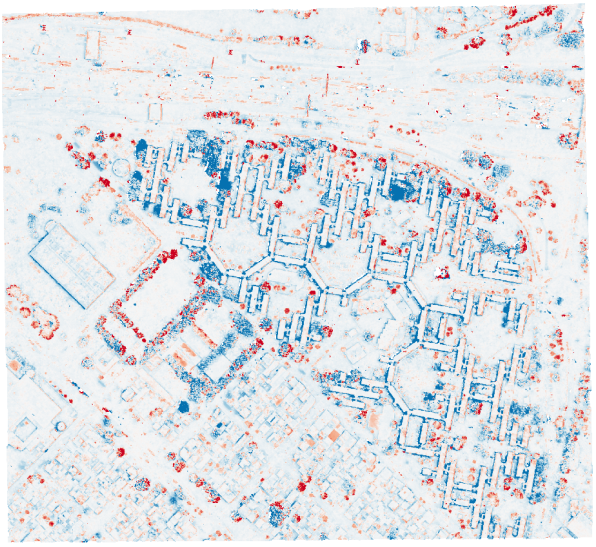


Figure 9. Difference between computed DSM and reference lidar image, ranging from -4 meters (blue) to 10 meters (red)

| | Completeness | Median (m) | RMSE (m) |
|---|---|---|---|
| Proposed | 68.39 % | 0.24 | 2.19 |
| IARPA | 64.1 % - 73.2 % | 0.35 - 0.47 | 2.2 - 2.59 |

Table 3. Performances comparison between our pipeline and metrics ranges from the IARPA challenge (Bosch et al., 2017)

## 5. CONCLUSION

While still in an early and active development state at CNES, CARS already achieves most of the goals it has been designed for: distributed computing on many nodes, stability and robustness enforced by the anticipation of error-prone steps in the pair preparation phase, and ability to produce full scale collections of images from the latest VHR sensors. In this paper, we also detailed our original algorithm for epipolar geometry estimation and refinement, which includes an initial low resolution DTM and allows to greatly reduce the disparity range to explore. Finally, we demonstrated that accuracy is comparable to other state-of-the-art pipelines on using images from the IARPA dataset. Future work include missing features, such

as outliers rejection before rasterisation, smarter rasterisation strategies and point cloud regularisation. The deployment of the pipeline in a cloud based, production oriented environment is also in progress in preparation for the CO3D mission (Melet et al., 2020). CARS will soon be made available as open-source (CNES, 2020).

## REFERENCES

Baltsavias, E., Stallmann, D., 1992. Metric Information Extraction from SPOT Images and the Role of Polynomial Mapping Functions. 358-364.

Beyer, R. A., Alexandrov, O., McMichael, S., 2018. The Ames Stereo Pipeline: NASA's open source software for deriving and processing terrain data. *Earth and Space Science*, 5.

Bosch, M., Foster, K., Christie, G., Wang, S., Hager, G. D., Brown, M., 2019. Semantic stereo for incidental satellite images. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1524–1532.

Bosch, M., Kurtz, Z., Hagstrom, S., Brown, M., 2016. A multiple view stereo benchmark for satellite imagery. *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 1–9.

Bosch, M., Leichtman, A., Chilcott, D., Goldberg, H., Brown, M., 2017. Metric Evaluation Pipeline for 3D Modeling of Urban Scenes. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1/W1, 239–246.

CNES, 2020. Cars. https://github.com/CNES/cars.

Cournet, M., Sarrazin, E., Dumas, L., Michel, J., Guinet, J., Youssefi, D., Defonte, V., Fardet, Q., 2020. Ground-truth generation and disparity estimation for optical satellite imagery. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Dask Development Team, 2016. Dask: Library for dynamic task scheduling.

de Franchis, C., Meinhardt-Llopis, E., Michel, J., Morel, J. ., Facciolo, G., 2014a. On stereo-rectification of pushbroom images. *2014 IEEE International Conference on Image Processing (ICIP)*, 5447–5451.

de Franchis, C., Meinhardt-Llopis, E., Michel, J., Morel, J.-M., Facciolo, G., 2014b. An automatic and modular stereo pipeline for pushbroom images. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3, 49–56.

Fua, P., 1993. A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. *Machine Vision and Applications*, 6(1), 35-49.

Gillies, S. et al., 2013-. Rasterio: geospatial raster I/O for Python programmers. Mapbox.

Grizonnet, M., Michel, J., Poughon, V., Inglada, J., Savinaud, M., Cresson, R., 2017. Orfeo ToolBox: Open source processing of remote sensing images. *Open Geospatial Data, Software and Standards*, 2(1), 15.

Habib, A., Morgan, M., Jeong, S., Kim, K.-O., 2005. Analysis of Epipolar Geometry in Linear Array Scanner Scenes. *The Photogrammetric Record*, 20, 27 - 47.

Haller, I., Pantilie, C., Oniga, F., Nedevschi, S., 2010. Real-time semi-global dense stereo solution with improved sub-pixel accuracy. *2010 IEEE Intelligent Vehicles Symposium*, 369–376.

Hartley, R., Zisserman, A., 2004. *Multiple View Geometry in Computer Vision*. second edition edn, CambridgeUniversity Press.

Hirschmuller, H., 2008. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2), 328-341.

Hoyer, S., Hamman, J., 2017. xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, 5.

Koh, J.-W., Yang, H.-S., 2016. Unified piecewise epipolar resampling method for pushbroom satellite images. *EURASIP Journal on Image and Video Processing*, 2016.

Krauß, T., d'Angelo, P., Schneider, M., Gstaiger, V., 2013. The Fully Automatic Optical Processing System CATENA at DLR. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-1/W1, 177–183.

Lebegue, L., Cazala-Hourcade, E., Languille, F., Artigues, S., Melet, O., 2020. Co3d, a worldwide one-meter accuracy dem for 2025. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Lowe, D., 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60, 91-.

Melet, O., Youssefi, D., L'Helguen, C., Michel, J., Sarrazin, E., Languille, F., Lebegue, L., 2020. Co3d mission digital surface model production pipeline. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Noh, M.-J., Howat, I. M., 2017. The Surface Extraction from TIN based Search-space Minimization (SETSM) algorithm. *ISPRS Journal of Photogrammetry and Remote Sensing*, 129, 55 - 76.

Nuth, C., Kääb, A., 2011. Co-registration and bias corrections of satellite elevation data sets for quantifying glacier thickness change. *The Cryosphere*, 5(1), 271–290.

Oh, J., Lee, W., Toth, C., Grejner-Brzezinska, D., Lee, C., 2010. A Piecewise Approach to Epipolar Resampling of Pushbroom Satellite Images Based on RPC. *Photogrammetric Engineering and Remote Sensing*, 76, 1353-1363.

Qin, R., 2016. RPC Stereo Processor (RSP) – A Software Package for Digital Surface Model and Orthophoto Generation from Satellite Stereo Imagery. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-1, 77–82.

Rupnik, E., Daakir, M., Pierrot Deseilligny, M., 2017. MicMac – a free, open-source solution for photogrammetry. *Open Geospatial Data, Software and Standards*, 2.

Vedaldi, A., Fulkerson, B., 2008. VLFeat: An Open and Portable Library of Computer Vision Algorithms.

Zabih, R., Woodfill, J., 1994. Non-parametric local transforms for computing visual correspondence. J.-O. Eklundh (ed.), *Computer Vision — ECCV '94*, Springer Berlin Heidelberg, Berlin, Heidelberg, 151–158.