IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2021–11–13.
Submitted on 2021–04–23, accepted on 2021–09–27.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.352

A Generic Bundle Adjustment Methodology for Indirect RPC Model Refinement of Satellite Imagery

Roger Marí[1], Carlo de Franchis[1,2], Enric Meinhardt-Llopis[1], Jérémy Anger[1,2], Gabriele Facciolo[1]

[1] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, France
[2] Kayrros SAS
({roger.mari, carlo.de-franchis, enric.meinhardt, jeremy.anger, gabriele.facciolo}@ens-paris-saclay.fr)

Abstract

This article describes a generic bundle adjustment methodology for multi-view stereo pipelines for 3D reconstruction from high-resolution optical satellite imagery. The Rational Polynomial Camera (RPC) model of each input view is refined by composing it with a rotation that compensates errors related to the attitude angles encoding the satellite orientation. A set of tie points, derived from feature tracks detected across the input images, is used to find the optimal corrective rotations by minimization of the reprojection error. We evaluate the performance of our method using time series of SkySat acquisitions over two different areas of interest. As an internal evaluation metric we compute the standard deviation between corresponding height estimations derived from different stereo pairs, before and after the bundle adjustment correction. Lastly, the results are also compared to an alternative solution for multi-view stereo pipelines that eludes any explicit correction of the camera models by directly registering independent dense surface models.

Source Code

The reviewed source code and documentation for this method are available at the web page of this article[1]. Usage instructions are included in the `README` file.

Keywords: satellite images; bundle adjustment; RPC model; SkySat; 3D reconstruction

[1] https://doi.org/10.5201/ipol.2021.352

1 Introduction

The Rational Polynomial Camera (RPC) model is a generic sensor model, which is widely used to describe the acquisition process of optical satellite sensors independently from the specific physical properties of the sensor. The RPC model associated with each satellite image relates 3D object space coordinates to 2D image coordinates. Both the 3D to 2D mapping (the *projection* function) and its inverse (the *localization* function) are expressed mathematically as a ratio of cubic polynomials.

The projection function $\mathcal{P}$ of an RPC model can be expressed as

$$(r, c) = \mathcal{P}(X, Y, Z) = \left(\frac{a(X, Y, Z)}{b(X, Y, Z)}, \frac{e(X, Y, Z)}{f(X, Y, Z)} \right), \tag{1}$$

where X, Y, Z represent the longitude, latitude and altitude of a 3D point; and r, c are the row and column of its projection on the image plane. In practice (X, Y, Z) and (r, c) are expressed in normalized coordinates, within a range of $[-1, 1]$, for reasons of numerical stability [2].

Equation (1) is characterized by a, b, e, f, which are cubic polynomials of 20 coefficients. Each of these polynomials is defined as

$$\begin{aligned} p\,(X, Y, Z) = {} & p_0 + p_1 Z + p_2 Y + p_3 X + p_4 ZY + p_5 ZX + p_6 YX + p_7 Z^2 + p_8 Y^2 \\ & + p_9 X^2 + p_{10} ZYX + p_{11} Z^2 Y + p_{12} Z^2 X + p_{13} Y^2 Z + p_{14} Y^2 X \\ & + p_{15} ZX^2 + p_{16} YX^2 + p_{17} Z^3 + p_{18} Y^3 + p_{19} X^3, \end{aligned} \tag{2}$$

where p_i is the i-th coefficient of p, and p can be a, b, e, or f.

Given a 2D point correspondence across two or more satellite images, the associated RPC models can be used to triangulate and retrieve the 3D point that projects onto them. Although RPCs provided by image vendors are expected to be precise enough, the complex system they encode is subject to measurement errors, which are mainly related to the attitude angles that define the sensor orientation. Such inaccuracies can easily cause a 3D point to project tens of pixels away from its real location on the image plane, according to the RPC projection function $\mathcal{P}$. This behavior implies that different views of a scene are typically not consistent in a common frame of reference. The RPC models may therefore introduce systematic errors in the triangulation of corresponding image points, substantially degrading the accuracy of 3D models reconstructed from multiple views.

Nowadays, small push-frame satellites such as SkySat from Planet are capable of covering large areas of the Earth by acquiring long strips of partially overlapping frames with a small geographic footprint [33]. In stereo acquisition mode, a second (and even a third) strip of images is captured during the same passage with a different inclination after reorienting the satellite. RPC inconsistencies are especially problematic when it comes to using such fragmented acquisitions for 3D reconstruction. The reason is that the relief of the entire area of interest can only be obtained by fusing large collections of local and partially overlapping 3D models derived from the small footprint images. RPC inaccuracies produce different coordinates for corresponding local surfaces, which enormously increases the difficulty of the fusion process. In this context, designing efficient RPC correction strategies has become essential to effectively harness push-frame satellite imagery for large-scale multi-view 3D reconstruction, as illustrated in Figure 1.

This work describes a bundle adjustment methodology, and its implementation, to refine the RPC models of a group of satellite images observing a certain area of interest (AOI) from different viewpoints. Our contributions are:

1. A minimalist solution, which seeks to estimate a light and non-redundant amount of parameters to address the root of the problem from a physical point of view. The only inputs that are strictly necessary are the initial RPC models and the corresponding georeferenced images.

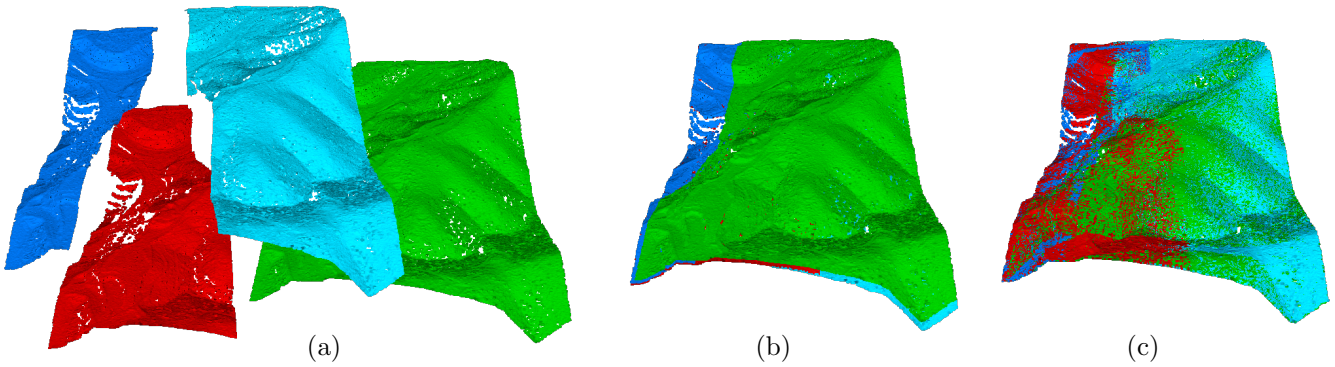


Figure 1: Toy example illustrating the benefits of RPC refinement on multi-view 3D reconstruction. (a) Independent point clouds computed from four different pairs of satellite images. (b) Relative location of the models in the object space, using unrefined RPC models. (c) Relative location of the models in the object space, using RPC models corrected with the proposed methodology. RPC inaccuracies result in non-consistent reconstructions, which accumulate in different layers, as shown in (b). The refined RPCs solve this problem and provide accurately aligned models, as shown in (c).

2. A stand-alone design. The refined cameras are output using the same standard as the input cameras, the RPC model, which makes them directly pluggable to any satellite stereo pipeline. Current open-source state of the art alternatives employ different formats to represent the corrected camera models, which is useful for internal use but not for reproducibility, e.g. the NASA Ames Stereo Pipeline [7].

3. A performance validation by measuring the practical impact in a multi-view 3D reconstruction context. We compute the deviation between corresponding height values of digital surface models (DSMs) derived from different pairs of SkySat cameras, before and after the RPC correction. The results are also compared to the equivalent deviation values obtained using a geometry registration algorithm for dense 3D models.

Bundle adjustment in a nutshell. Bundle adjustment is the problem of jointly optimizing the viewing parameters (external and/or internal) of multiple cameras and the 3D locations of the objects they observe [43]. Given an initial set of K 3D points $\{\boldsymbol{X}_k\}_{k=1,\dots,K}$, and their 2D observations $\boldsymbol{x}_{mk}$ across M cameras with projection functions $\{\mathcal{P}_m\}_{m=1,\dots,M} : \mathbf{R}^3 \to \mathbf{R}^2$, bundle adjustment finds the optimal solution by minimizing the reprojection error of the setting

$$E(\mathcal{P}_m, \boldsymbol{X}_k) = \sum_{k=1}^{K} \sum_{m=1}^{M} \|\boldsymbol{x}_{mk} - \mathcal{P}_m(\boldsymbol{X}_k)\|^2, \tag{3}$$

where $\boldsymbol{X}_k$ and $\mathcal{P}_m$ contain the variables to be adjusted. Following (3), the reprojection error is typically defined as the sum of squared Euclidean distances between the estimated reprojected 3D points, i.e. $\mathcal{P}_m(\boldsymbol{X}_k)$, and their actual 2D observations in the images, i.e. $\boldsymbol{x}_{mk}$.

The alternative approach to the bundle adjustment problem proposed in [38] is worth mentioning. Bundle adjustment creates a significant number of variables to be estimated when the number of observed points is large. To avoid this growth in search variables, [38] introduce an alternative cost function based on the cross product of the projection rays. In particular, the classic reprojection error is replaced by a sum of squares of the minimum distance between the projection rays for each point. Since in this formulation the 3D point locations are implicit, the search space is reduced to the camera parameters only; however, the authors conclude that the standard bundle adjustment formulation provides similar convergence properties for most applications.

2 Related Work

Over the past two decades, researchers have thoroughly investigated the problem of RPC refinement. The literature can be broadly divided into *indirect* or *direct* approaches. Indirect refining methods propose complementary functions, defined in image or object space, which are composed with the original RPC functions to correct the camera models. Oppositely, direct methods seek to explicitly modify the RPC coefficients. The variables involved in the RPC correction process are usually tuned based on a set of Ground Control Points (GCPs), i.e. tie points seen in the images whose absolute location in the 3D object space is known [23, 42, 46]. In absence of GCPs, keypoint feature detectors can be used to infer a substitute set of tie points between two or more images and perform a relative (not absolute) correction between the input cameras [21, 22, 45]. Due to the complexity of the task, direct methods require a larger number of tie points and are prone to poorer stability and accuracy with respect to indirect methods, which are the most common practice [46].

Nowadays, bundle adjustment strategies based on the minimization of the reprojection error of a set of tie points represent the standard solution for RPC correction in most state of the art pipelines for 3D reconstruction from satellite images [7, 26]. The projection of each tie point across the different cameras after bundle adjustment should ideally result in exactly corresponding 2D locations in the images, implying that the corrected RPCs are consistent in a common frame of reference.

In the case of satellite images, since cameras are far from the Earth's surface (typically above 500 km), the main component of the reprojection error comes from the inaccurate knowledge of the satellite orientation [23]. Thus, the energy in (3) could ideally be minimized by composing each projection map $\mathcal{P}_m$ with a 3D transformation $\mathcal{R}_m$, whose objective is to reorient the camera with respect to the object coordinates. This way the problem can be reformulated as

$$E(\mathcal{R}_m, \boldsymbol{X}_k) = \sum_{k=1}^{K} \sum_{m=1}^{M} \|\boldsymbol{x}_{mk} - \mathcal{P}_m(\mathcal{R}_m(\boldsymbol{X}_k))\|^2. \tag{4}$$

In [23] it was shown that the net effect of RPC inaccuracies related to the attitude angles reduces to a 2D translation (also termed as *correction offset*) in images covering lengths up to 50 km. Based on this observation, *bias compensation* approaches for such scenarios have in common the optimization of a 2D correction offset for each RPC model [17, 20, 34]. Following the notation from (3), this amounts to finding the 2D translations $\mathcal{T}_m$ in the image domain that minimize

$$E(\mathcal{T}_m, \boldsymbol{X}_k) = \sum_{k=1}^{K} \sum_{m=1}^{M} \|\boldsymbol{x}_{mk} - \mathcal{T}_m(\mathcal{P}_m(\boldsymbol{X}_k))\|^2. \tag{5}$$

Note that (4) inserts the correction transformation before the original RPC projection, while (5) does it after. More generally, RPCs can also be corrected by composing them with polynomial functions depending on the image or object coordinates of the tie points. This allows modeling other error sources beyond attitude inaccuracies, such as time-dependent drift or lens distortion. Although they are normally orders of magnitude smaller, these errors may not be negligible for certain scenarios, e.g. large images covering hundreds of kilometers [23, 42].

To bundle adjust or not? For small areas of interest, covering lengths up to some hundreds of meters or a few kilometers, RPCs can be locally modeled as affine cameras, using the first order Taylor approximation [16, 18, 23, 41]. Under this approximation, a translation in the image domain is equivalent to a translation in the object world. This implies that triangulation errors due to RPC inaccuracies can ideally be corrected by a 3D translation, which can be estimated after the 3D reconstruction stage, e.g. with a point cloud registration algorithm. Previous works show that

geometry based alignments can be used at a local scale to elude any prior RPC correction [18, 24, 29], but they offer poor scalability as the amount of models to register increases. Running a bundle adjustment to refine the RPC models seems a more natural and less-constrained solution to the problem, as long as enough tie points are available. To further explore this issue, we compare our methodology to an alternative algorithm that can be used to explicitly register independent dense surface models (Section 4.4).

3 Methodology

We present a bundle adjustment methodology to refine the RPC models of a group of satellite images observing a certain AOI from different viewpoints. Our objective is to produce a set of corrected RPC models, consistent with each other, which will result in the automatic alignment of corresponding relief estimates derived from different stereo pairs processed by a stereo reconstruction pipeline.

A block diagram of our method is shown in Figure 2. The first part of the pipeline is dedicated to feature tracking, which serves to initialize the image and object coordinates of a set of tie points observed across the input images. The tie points and their image observations, together with the input camera models, are input to the second part of the pipeline, where the bundle adjustment problem is solved. Based on the solution, a corrected RPC model is finally fitted for each input image. The different blocks of the chain are explained in detail in the following subsections.

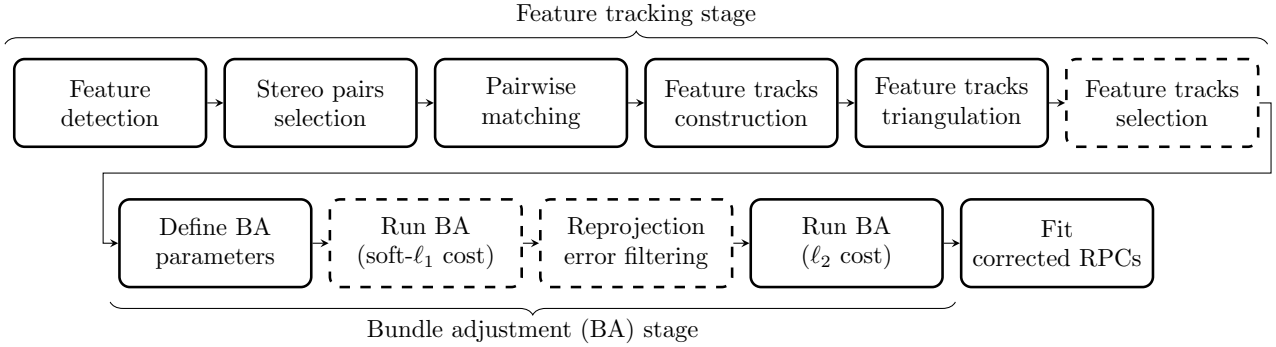


Figure 2: Block diagram of the proposed bundle adjustment methodology for RPC correction. Dashed blocks indicate optional but recommended processing, intended to improve accuracy and/or efficiency.

3.1 Feature Tracking

As stated in Section 1, bundle adjustment problems are solved based on the reprojection of a set of tie points across the input cameras. The list of 2D coordinates containing the observation of a 3D point across multiple images is known as a *feature track* (Figure 3). In particular, the feature tracks employed in our methodology result from pairwise correspondences of distinctive keypoints.

This section describes in detail the blocks involved in the feature tracking stage, which are listed in the diagram in Figure 2.

3.1.1 Feature Detection

We employ the SIFT method to extract distinctive keypoints from the input images [28, 37]. Each SIFT keypoint is identified by a descriptor of 128 values, invariant to image translation, rotation, and scaling. SIFT descriptors are also robust to a wide range of image transformations, such as slight changes of viewpoint, noise, blur, contrast changes or scene deformation.

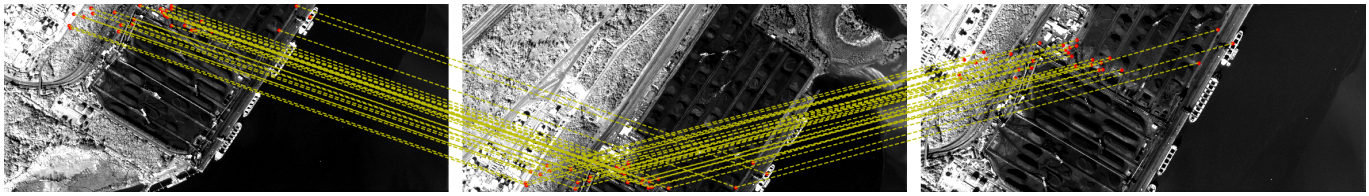


Figure 3: Example of 30 feature tracks detected across 3 overlapping views.

For large images, limiting the maximum number of keypoints per view can be useful to regulate the cost of the matching step, which usually represents the bottleneck of feature tracking algorithms. To this end, we order the SIFT features of each image from the coarsest to finest scale and select the first N_{kp} keypoints to proceed to the matching step. The rationale is that coarse keypoints are less affected by image noise, thus more likely to be seen in the other images. In this work we consider a large value for N_{kp}, 60000 SIFT keypoints are allowed per image. Lower values of N_{kp}, e.g. only a few thousand points, can be used to speed up the feature tracking stage, at the risk of a possible loss of accuracy. This is because keypoints from the coarser scales are located with less precision. In the same way, fewer keypoint correspondences can be expected to be found as N_{kp} decreases.

3.1.2 Stereo Pairs Selection

A key aspect of feature tracking is to define which pairs of images are suitable to be matched to construct feature tracks and which pairs can be omitted. Avoiding unnecessary pairs is the most effective way to save computational time and prevent undesired mismatches, which would result in erroneous tie points for the bundle adjustment problem.

Algorithm 1 is employed to select suitable image pairs to match from all possible stereo pairs based on the projected area overlap, in UTM coordinates, between the two images. The list of pairs to match is denoted by *pairs_to_match*. Out of all the pairs in this list, the subset of pairs offering a well-posed baseline-to-altitude ratio is stored in *pairs_to_triangulate*. The pairs in *pairs_to_triangulate* are used to define the 3D tie point associated with each feature track (Section 3.1.5). Matches from pairs in *pairs_to_match* that are not part of *pairs_to_triangulate* are kept to contribute to the reprojection error of the bundle adjustment stage if they are part of a feature track containing observations in at least one triangulation pair.

Algorithm 1: Stereo pairs selection

input : list of initial stereo pairs, *input_pairs* [default value = all possible pairs]
minimum image footprint overlap ratio, t_{overlap} [default value = 0.1]
minimum baseline-to-altitude ratio, $t_{\text{baseline-alt}}$ [default value = 1/4]
orbit altitude, in meters [default value = 500000] *Orbit altitude ≈ 500 km for SkySat*

output : list of stereo pairs suitable to match, *pairs_to_match*
list of stereo pairs suitable to triangulate, *pairs_to_triangulate*

Initialize *pairs_to_match* and *pairs_to_triangulate* as empty lists
for each pair **in** *input_pairs* **do**
  Get the two images of the current pair, i.e. I_{ref}, I_{aux}
  Intersect footprints of I_{ref} and I_{aux} in UTM coordinates
  if $\frac{\text{UTM intersection area}}{\text{UTM area } I_{ref}} \geq t_{\text{overlap}}$ **then**
    Add pair to *pairs_to_match*
    Compute baseline between I_{ref} and I_{aux}
    if $\frac{\text{baseline}}{\text{orbit altitude}} \geq t_{\text{baseline-alt}}$ **then**
      Add pair to *pairs_to_triangulate*

3.1.3 Pairwise Matching

For each stereo pair in the *pairs_to_match* list, SIFT keypoints are matched using an automatic Fast Approximate Nearest Neighbors algorithm [32]. The search for matches is restricted to points located within the regions of the images corresponding to the intersection of their geographic footprints.

Mismatches in the feature track observations can strongly undermine the bundle adjustment performance, leading to failure or strong biases [3, 29]. We remove erroneous matches by means of a distance ratio test with a relative threshold of 0.6 [28]. Optionally, if the images cover small geographic footprints and the local affine camera approximation holds, as explained in Section 2, this can be followed by a RANSAC geometric filtering using the fundamental matrix[2] [25].

Lastly, the distance d_{geo} between the approximate geographic coordinates of matched keypoints is also used to further remove pairwise mismatches. d_{geo} is computed, in meters, as

$$d_{\text{geo}} = \|\text{UTM}(\mathcal{L}_{\text{ref}}(x_{\text{ref}}, y_{\text{ref}}, z_{\text{ref}})) - \text{UTM}(\mathcal{L}_{\text{aux}}(x_{\text{aux}}, y_{\text{aux}}, z_{\text{aux}}))\|, \tag{6}$$

where $x_{\text{ref}}, y_{\text{ref}}$ and $x_{\text{aux}}, y_{\text{aux}}$ are the image coordinates of a keypoint and its match, respectively. The approximate geographic coordinates of each point are obtained using the localization functions of the two cameras, denoted by $\mathcal{L}$, evaluated at a certain reference altitude, i.e. z_{ref}. The reference altitude is arbitrarily set as the SRTM altitude at the center of the image footprints. The geographic coordinates are converted to the UTM system to compute d_{geo}.

The values of d_{geo} should concentrate around an offset induced by the RPC errors. In presence of outliers, such distances, if sorted, typically describe a well-defined elbow shaped function. Based on this observation, an outlier rejection threshold τ is automatically set using the elbow point (i.e. the point with largest distance with respect to the line defined by the minimum and maximum d_{geo} values), as shown in Figure 4. If τ rejects more than 20% of the matches it is assumed that the distribution does not follow an elbow shape and no thresholding takes place, i.e.

$$\tau = \begin{cases} \tau, & \text{if } \tau \text{ is larger than the 80th percentile of } d_{\text{geo}} \\ \infty, & \text{otherwise.} \end{cases} \tag{7}$$

All pairwise matches where $d_{\text{geo}} > \tau$ are discarded.

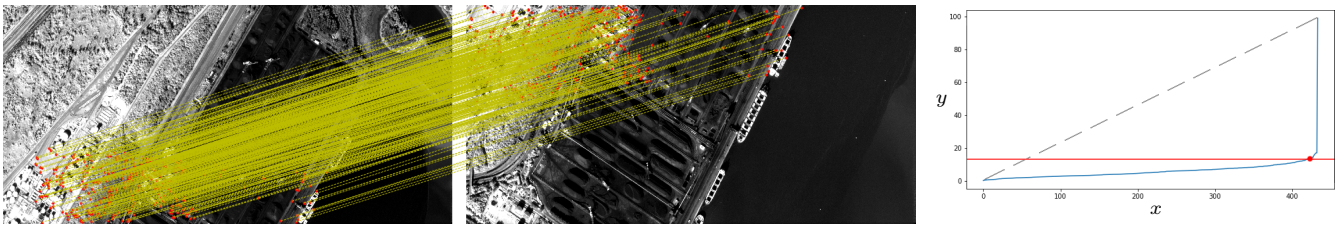


Figure 4: Example of pairwise matching result. The plot on the right shows, in blue, the sorted approximated geographic distances d_{geo} (y-axis) for each match of a stereo pair (x-axis), computed using (6). An outlier rejection threshold τ, in red, is automatically set using the elbow point. The elbow point is defined as the point with largest distance with respect to the dashed line, i.e. the line defined by those points of the distribution with the minimum and maximum y values.

3.1.4 Feature Tracks Construction

The recursive union-find algorithm from [31] is used to efficiently extend pairwise matches to feature tracks of arbitrary length (Algorithm 2). Keypoints that belong to the same track are assigned a common *parent* value. This strategy is independent of the order by which stereo pairs were matched.

[2] The code uses a generic fundamental matrix, derived using the 7-point algorithm. In many respects the epipolar geometry of two affine cameras is identical to that of two perspective cameras, with some simplifications due to the fact that the epipoles are at infinity [25]. The RANSAC rejection threshold is set to 0.3 pixels.

Algorithm 2: Feature tracks construction

input : list of pairwise matches connecting N keypoints, *pairwise_matches*
output : list of feature tracks, $\{T_j\}_{j=1,\dots,N_{\text{tracks}}}$
Create a unique id for each matched keypoint from 1 to N
Initialize an empty vector *parents* with length N, where each position corresponds to an id
Define the union-find functions:
  Function `Find`(*parents, id*)*:*
    $parent\_id = parents[id]$, i.e. the id-th value of *parents*
    return id **if** $parent\_id$ is empty; **else** `Find`(*parents, parent_id*)
  Function `Union`(*parents,* id_{left}*,* id_{right}*)*:
    $parent\_id_{\text{left}}$ = `Find`(*parents,* id_{left}), $parent\_id_{\text{right}}$ = `Find`(*parents,* id_{right})
    if $parent\_id_{\text{left}} \neq parent\_id_{\text{right}}$ **then** $parents[parent\_id_{\text{left}}] = parent\_id_{\text{right}}$
for each pairwise match **in** *pairwise_matches* **do**
  `Union` (*parents*, left keypoint id, right keypoint id)
N_{tracks} = number of unique values in *parents*
Each track T_j corresponds to the set of keypoints whose ids have the same value in *parents*

Algorithm 3: Feature tracks triangulation

input : list of stereo pairs suitable to triangulate, *pairs_to_triangulate*
  list of feature tracks, $\{T_j\}_{j=1,\dots,N_{\text{tracks}}}$
output : list of 3D points corresponding to the input feature tracks, $\{X_j\}_{j=1,\dots,N_{\text{tracks}}}$
for each track T_j **do**
  Initialize $\boldsymbol{X}_j = (0, 0, 0)$ — *$\boldsymbol{X}_j$ is the 3D point projecting on the j-th feature track*
  Set $w_j = 0$ — *w_j counts how many triangulation pairs have been used to compute $\boldsymbol{X}_j$*
for each pair **in** *pairs_to_triangulate* **do**
  Get the two images of the current pair, i.e. I_{ref}, I_{aux}
  for each track T_j **do**
    if T_j is seen in I_{ref} and I_{aux} **then**
      Get the 2D point observations of T_j in I_{ref} and I_{aux}, i.e. $\boldsymbol{x}_{\text{ref}j}$, $\boldsymbol{x}_{\text{aux}j}$
      $\boldsymbol{X}$ = `Triangulate` $(\boldsymbol{x}_{\text{ref}j}, \boldsymbol{x}_{\text{aux}j})$ — *Triangulate using the RPC models of I_{ref} and I_{aux}*
      $\boldsymbol{X}_j = (w_j \boldsymbol{X}_j + \boldsymbol{X}) \,/\, (w_j + 1)$ — *Update the coordinates of the j-th 3D point*
      $w_j = w_j + 1$ — *Update the counter of pairs employed to compute the j-th 3D point*

3.1.5 Feature Tracks Triangulation

Similarly to [29], the object coordinates of the tie point that projects onto each feature track are initialized by triangulating all the pairwise matches of the track and taking the mean of the resulting 3D locations (Algorithm 3). The RPC triangulation algorithm from [15], denoted by `Triangulate` in Algorithm 3, is employed to triangulate keypoint correspondences with the RPC camera models.

3.1.6 Feature Tracks Selection

In large-scale scenes, using an optimal subset of tracks may benefit bundle adjustment in two ways:

1. Speed up the process and reduce memory usage by removing redundant constraints.
2. Increase the calibration accuracy due to the rejection of tracks with higher localization error.

Our pipeline employs the feature tracks selection method from [12], summarized in Algorithm 4, to select an optimal subset of feature tracks (or equivalently tie points) for the bundle adjustment.

Algorithm 4: Feature tracks selection

input : epipolar graph, $EG = \{V, E\}$ *V and E denote the nodes/cameras and edges of EG*
list of tracks, $\{T_j\}_{j=1,\dots,N_{\text{tracks}}}$
number of spanning trees, K_{EG} [default value = 60]
output : subset of tracks, S
T_{ranked} = set of ranked tracks in T in decreasing priority (*length-scale-cost* criterion)
Initialize $k = 0$ *k is the counter of spanning trees*
Initialize $S = \{\}$ *S is the output subset of tracks*
while $k < K_{\text{EG}}$ **do**
  Compute camera weights *Equation (8)*
  C_{root} = camera with largest weight,
  Create *inverted_list* using T_{ranked} *a list that sorts the tracks seen in each camera using* T_{ranked}
  Set $l = 1$ *l counts the layers in the current tree*
  Initialize $H_k(l) = \{C_{\text{root}}\}$ *$H_k(l)$ is the set of cameras in the l-th layer of the current tree*
  Set $h = 1$ *h counts the number of cameras in $H_k(l)$*
  Initialize $S_k = \{\}$ *S_k is the set of tracks selected by the current tree*
  Initialize $I_k = \{C_{\text{root}}\}$ *I_k is the set of cameras connected by the current tree*
  while $V - I_k \neq \emptyset$ **and** $h > 0$ **do**
    Initialize $H_k(l+1) = \{\}$ *$H_k(l+1)$ is the set of cameras to visit in the next layer of the tree*
    for each node C_q in $H_k(l)$ **do**
      for each track T_q in *inverted_list* visible in node N_q **do**
        if $T_q \notin S_k$ **then**
          W_q = cameras where T_q is seen
          R_q = neighbor cameras of C_q according to the EG
          $Z_q = W_q \cap R_q$ *Z_q = neighbor cameras of camera C_q where track T_q is seen*
          if $Z_q \not\subset I_k$ **and** $Z_q \neq \emptyset$ **then**
            Add $\{Z_q - Z_q \cap I_k\}$ to $H_k(l+1)$ *neighbor cameras part of the track and not yet visited will be visited in the next layer*
            $S_k = S_k \cup T_q$ *add track to the set of tracks selected by the current tree*
            $I_k = I_k \cup Z_q$ *update the set of cameras visited by the current tree*
    $l = l + 1$
    h = number of cameras in the l-th layer of tree H_k
    Sort cameras in the l-th layer in descending order according to their camera weight
  $S = S \cup S_k$ *add all tracks selected by the current tree to the output subset*
  $T_{\text{ranked}} = T_{\text{ranked}} - S_k$ *remove tracks selected by the current tree from the set of ranked tracks*
  $k = k + 1$ *update the counter of spanning trees, a new tree will be explored next*

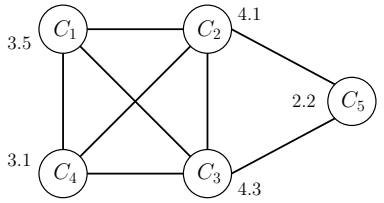
Algorithm 4 represents the input cameras and the feature tracks connecting them with an epipolar graph (EG). In the EG each camera is a node and edges represent the epipolar relationship between pairs of nodes. The weight of each edge can be seen as the number of pairwise matches between two cameras, while the weight associated with the i-th camera, denoted by C_i, is defined as

$$w(C_i) = n(C_i) + e^{-cost(C_i)}, \tag{8}$$

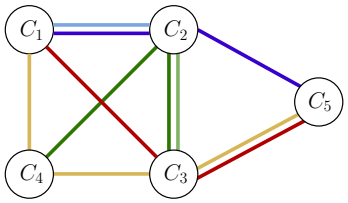
where $n(C_i)$ is the number of neighbor cameras of C_i and $cost(C_i)$ is computed as

$$cost(C_i) = mean(C_i) + \mu \cdot std(C_i), \tag{9}$$

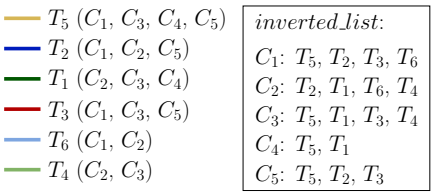
where $mean(C_i)$ and $std(C_i)$ correspond to the mean and the standard deviation of the average reprojection errors of visible tracks in C_i. The scalar μ is a balancing factor set to 3 [12].



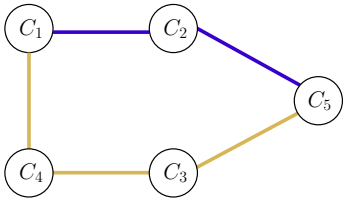
(a) Epipolar graph (EG), with the weight $w(C_i)$ next to each camera C_i



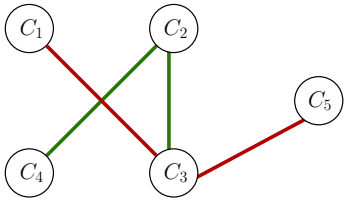
(b) Input feature tracks, where each track T_j is assigned a different color



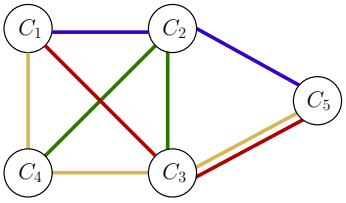
(c) Ranked feature tracks from highest to lowest priority and inverted list



(d) First spanning tree selection, i.e. S_k with $k = 0$



(e) Second spanning tree selection, i.e. S_k with $k = 1$



(f) Output set of feature tracks S for $K_{EG} = 2$, T_6 and T_4 are discarded

Figure 5: Toy example of Algorithm 4 applied to five cameras connected by six feature tracks.

The selection process is a sequential procedure that constructs a number of *spanning trees*, denoted by K_{EG}. Each spanning tree selects a group of feature tracks, from those not chosen before, that can be used to connect all nodes in the EG. An example of EG, with the corresponding camera weights, is shown in Figure 5(a). Given the set of feature tracks in Figure 5(b), two different spanning tree selections are shown in Figure 5(d) and 5(e). The final result of the algorithm is obtained from the union of the feature tracks selected by all spanning trees, as illustrated in Figure 5(f).

Each spanning tree is constructed following a logic that establishes which cameras of the EG have to be visited first and which of the tracks visible in each camera have to be considered first. In general terms, cameras connected to a larger number of nodes are visited first, and longer tracks are selected first. In case of a tie (e.g. if two cameras are connected to the same number of nodes or if two tracks have the same length), other secondary characteristics are used to decide the order.

More specifically, the aforementioned logic is obtained by sorting the cameras in descending order according to their weight (8) and by ranking all input feature tracks from highest to lowest priority. Tracks are ranked using the *length-scale-cost* criterion: first, comparing the number of images where they are visible (*track length*), i.e. longer tracks are given higher priority; then, for those tracks with the same length, the average scale of the keypoints (*track scale*) is considered, i.e. lower scale tracks, susceptible of being located with higher precision, have higher priority; finally, for those tracks equal in length and scale, the ones with lower cost have a higher priority (*track cost*). Similarly to (9), the cost associated with the j-th track, denoted by T_j, is defined as

$$cost(T_j) = mean(T_j) + \mu \cdot std(T_j), \tag{10}$$

where $mean(T_j)$ and $std(T_j)$ correspond to the mean and the standard deviation of the reprojection errors of all feature observations in track T_j. Based on the tracks ranking, an inverted list, which sorts the visible tracks per camera in decreasing priority, can be generated. An example of an inverted list derived from a set of ranked tracks is shown in Figure 5(c).

The only hyper-parameter of Algorithm 4 that needs to be set is K_{EG}. Ultimately, K_{EG} indirectly controls the number of selected tracks, since higher values will result in the union of subsets found by a wider variety of spanning trees. Further details and other toy examples similar to Figure 5 describing Algorithm 4 can be found in [12].

3.2 Bundle Adjustment for RPC Model Refinement

Bundle adjustment is essentially a nonlinear weighted least squares problem, which can be written in generic form as

$$\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{N} w_i r_i^2(\mathbf{x}), \tag{11}$$

where w_i are the weights and r_i is the nonlinear function employed to compute the reprojection residual of the i-th feature track observation, i.e. $r_i = \|\boldsymbol{x}_{mk} - \mathcal{P}_m(\boldsymbol{X}_k)\|$ in (3).

The vector $\mathbf{x}$ in (11) contains all the parameters to estimate, which consist of

1. The parameters that are necessary to correct each camera model.

2. The coordinates of the tie points in object space.

The initial values of 1 define a set of rotations compensating for RPC inaccuracies (Section 3.2.1), and the initial values of 2 are obtained after the feature track triangulation step (Section 3.1.5).

Given the solution of (11), the corrected tie points, projected with the corrected camera models, should ideally match the corresponding feature track observations. Since SIFT keypoints are located with a certain degree of noise, a minimum subpixel error should be expected.

The next section describes the camera correction parameters and the details of the optimization process related to the bundle adjustment stage in Figure 2.

3.2.1 Camera Correction Parameters

The presented pipeline aims to refine the input RPC models by means of a corrective rotation around the camera center, which compensates errors in the attitude angles defining the satellite orientation. Given a 3D point $\boldsymbol{X}$, its bundle adjusted projection is obtained as

$$\boldsymbol{x}_{\mathrm{BA}} = \mathcal{P}(R(\boldsymbol{X} - \boldsymbol{C}) + \boldsymbol{C}), \tag{12}$$

where $\boldsymbol{x}_{\mathrm{BA}} = (r_{\mathrm{BA}}, c_{\mathrm{BA}})$, i.e. the bundle adjusted image coordinates, $\mathcal{P}$ is the unrefined projection function of the input RPC model and R is the corrective rotation estimated by bundle adjustment. The camera center $\boldsymbol{C}$ is derived by regressing a projective model from each RPC model, and it remains fixed during the optimization[3]. The corrective rotation R is initialized as the identity matrix and it is represented using Euler angles, which entails three values to optimize per camera:

$$R_m = R_x(\phi_m)R_y(\theta_m)R_z(\alpha_m) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_m & -\sin\phi_m \\ 0 & \sin\phi_m & \cos\phi_m \end{pmatrix} \begin{pmatrix} \cos\theta_m & 0 & \sin\theta_m \\ 0 & 1 & 0 \\ -\sin\theta_m & 0 & \cos\theta_m \end{pmatrix} \begin{pmatrix} \cos\alpha_m & -\sin\alpha_m & 0 \\ \sin\alpha_m & \cos\alpha_m & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{13}$$

where ϕ_m, θ_m and α_m are the three Euler angles to optimize for the m-th camera. This formulation is similar to the one introduced in [29], the key distinction is that in this work the estimated rotation corrects the original RPC model itself instead of a local affine camera that approximates it.

[3]Each projective model is obtained by creating a 3D grid in the scene space and finding its projection with the RPC. A DLT resectioning algorithm is used to estimate a projective matrix using the 3D to 2D correspondences [25]. The center $\boldsymbol{C}$ of the projective model is used as center of rotation in (12). This is a coarse approximation, similar to [7], but from our experience the exact value of $\boldsymbol{C}$ has little impact on the bundle adjustment solution.

3.2.2 Numerical Optimization

The bundle adjustment problem is iteratively solved using a Trust Region Reflective method for large-scale bound-constrained nonlinear minimization problems [8]. Our code uses an implementation from the open-source Python library SciPy[4], which employs the LSMR solver[5] [19]. Instead of explicitly computing the derivatives of the problem, the algorithm relies on a finite difference estimation using the sparsity structure of the Jacobian to speed up the optimization process [13, 4].

Trust region methods can be seen as an evolution of Levenberg-Marquardt algorithms (both being *damped* Newton step methods) which are able to follow the negative curvature of the objective function for faster convergence [5, 27, 43]. Further details on damped Newton methods and the construction of large sparse Jacobian matrices for bundle adjustment problems are given in Appendix A.

The numerical optimization stops iterating when the change of the cost function falls below a certain threshold controlled by a f_{tol} tolerance value. Similarly, other tolerances for termination checking the change of the independent variables (x_{tol}) and the norm of the gradient (g_{tol}) are used as well. The default values of f_{tol}, x_{tol} and g_{tol} are set to 10^{-4}, 10^{-10} and 10^{-8} respectively.

3.2.3 Cost Functions and Reprojection Error Based Filtering

In addition to the outlier rejection tests conducted during the pairwise matching step (Section 3.1.3), a wide variety of compatible strategies have been studied in the literature to further minimize the impact of erroneous observations in bundle adjustment problems. The rejection of points whose reprojection error exceeds a certain threshold after a series of initial iterations is a popular strategy [39, 44, 29]. Other approaches use cost functions that enforce robustness to occasional large residuals, or employ iterative reweighting according to the reprojection error of each observation [43, 3, 47].

In line with the previous ideas, the proposed bundle adjustment optimization starts with some initial iterations, employing a soft-ℓ_1 cost function (a variant of the Huber loss). Equation (11) thus takes the form

$$\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{N} 2 \left(\sqrt{1 + r_i^2(\mathbf{x})} - 1 \right). \tag{14}$$

After this, a second series of iterations is applied using a classic sum of squared ℓ_2 differences, which yields the optimal estimator for Gaussian distributed perturbations. If we assume uniform weights:

$$\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{N} r_i^2(\mathbf{x}). \tag{15}$$

At the end of each series of iterations, the average reprojection distance ρ per feature track observation is computed as the average value of r_i, i.e. $\rho = \frac{1}{N} \sum_{i=1}^{N} r_i$.

Since (14) offers higher robustness to outliers than (15), after the first iterations we expect the gap in terms of reprojection distance between inlier and outlier observations to increase. In presence of outliers, at the end of the soft-ℓ_1 iterations, the sorted observation-wise reprojection distances for each image generally describe an elbow function. The method illustrated in Figure 4 can be used to automatically set a rejection threshold τ_ρ, following the same logic as (7). If $r_i > \tau_\rho$, the i-th feature track observation is discarded. The remaining tracks that contain at least two observations are fed to the second series of iterations, that quickly converges to the final solution.

We set the maximum number of iterations $N_{\text{iter soft-}\ell_1}$ and $N_{\text{iter }\ell_2}$ to 50 and 300, for (14) and (15) respectively. A summary of all the parameters listed so far, their meaning and their default values, can be found in Table 1.

[4]Generic interface for least-squares minimization, https://github.com/scipy/scipy/blob/master/scipy/optimize/_lsq/least_squares.py

[5]LSMR software, David Fong, Michael Saunders, https://web.stanford.edu/group/SOL/software/lsmr, 2011.

Parameter	Meaning [default value]	Section
N_{kp}	max number of SIFT keypoints per image (larger scale points have priority) [60000]	3.1.1
t_{overlap}	min geographic area overlap ratio between two images to match, from 0 to 1 [0.1]	3.1.2
$t_{\text{baseline-alt}}$	min baseline-to-altitude factor between two images to triangulate [1/4]	3.1.2
$t_{\text{sift-ratio}}$	SIFT matching relative threshold between nearest and second nearest descriptors [0.6]	3.1.3
K_{EG}	number of spanning trees of the epipolar graph covered after feature track selection [60]	3.1.6
K_{ranking}	ranking criteria for feature track selection, in decreasing priority [*length-scale-cost*]	3.1.6
f_{tol}	tolerance for BA termination by the change of the cost function [10^{-4}]	3.2.2
x_{tol}	tolerance for BA termination by the change of the independent variables [10^{-10}]	3.2.2
g_{tol}	tolerance for BA termination by the norm of the gradient [10^{-8}]	3.2.2
$N_{\text{iter soft-}\ell_1}$	max BA iterations with a soft-ℓ_1 cost function [50]	3.2.3
$N_{\text{iter }\ell_2}$	max BA iterations with a squared ℓ_2 cost function [300]	3.2.3

Table 1: Main hyper-parameters of the feature tracking and bundle adjustment blocks.

3.3 RPC Model Fitting

After the bundle adjustment optimization, a new RPC model is fitted for each input image. The new model encodes the corrected projection function $\mathcal{P}_{\text{BA}}$, which results from the composition of the estimated rotation matrix R and the input RPC projection function $\mathcal{P}$, as defined in (12).

The RPC fitting error derived from the i-th tie point with known 3D coordinates (X, Y, Z) and image coordinates (r, c), can be expressed as a vector of two components $(\delta r_i, \delta c_i)$,

$$\delta r_i = \frac{a\,(X,Y,Z)}{b\,(X,Y,Z)} - r = \frac{\boldsymbol{a}^T \boldsymbol{u}_i}{\boldsymbol{b}^T \boldsymbol{u}_i} - r, \tag{16}$$

$$\delta c_i = \frac{e\,(X,Y,Z)}{f\,(X,Y,Z)} - c = \frac{\boldsymbol{e}^T \boldsymbol{u}_i}{\boldsymbol{f}^T \boldsymbol{u}_i} - c, \tag{17}$$

where δr_i and δc_i are the differences in pixel units between the projected and the real image coordinates. The vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{e}$, $\boldsymbol{f}$ contain 20 coefficients each, defining the cubic polynomials a, b, e, f that characterize the RPC projection function (1), i.e.

$$\begin{aligned} \boldsymbol{a} &= [a_0 \;\; a_1 \;\; a_2 \;\; \cdots \;\; a_{19}]^T, \quad \boldsymbol{b} = [b_0 \;\; b_1 \;\; b_2 \;\; \cdots \;\; b_{19}]^T, \\ \boldsymbol{e} &= [e_0 \;\; e_1 \;\; e_2 \;\; \cdots \;\; e_{19}]^T, \quad \boldsymbol{f} = [f_0 \;\; f_1 \;\; f_2 \;\; \cdots \;\; f_{19}]^T. \end{aligned}$$

The vector $\boldsymbol{u}_i$, in turn, contains the variables associated with each polynomial coefficient in (2),

$$\begin{aligned} \boldsymbol{u}_i = [&1 \;\; Z \;\; Y \;\; X \;\; ZY \;\; ZX \;\; YX \;\; Z^2 \;\; Y^2 \;\; X^2 \;\; ZYX \\ &Z^2Y \;\; Z^2X \;\; Y^2Z \;\; Y^2X \;\; ZX^2 \;\; YX^2 \;\; Z^3 \;\; Y^3 \;\; X^3]. \end{aligned}$$

Following (16) and (17), the new RPC coefficients are estimated by solving $\min_{\boldsymbol{a},\boldsymbol{b}} \sum_i |\delta r_i|^2$ and $\min_{\boldsymbol{e},\boldsymbol{f}} \sum_i |\delta c_i|^2$ with a sufficient amount of 3D to 2D point correspondences, which are known as Control Points (CNPs). This is done in the presented pipeline by means of the open-source regularized weighted least squares algorithm described in [2]. The projection function $\mathcal{P}_{\text{BA}}$ obtained from the resulting RPC coefficients encodes the corrected mapping given by the bundle adjustment solution with high precision. The average absolute δr and δc errors are of the order of 10^{-4} pixels or less.

The set of CNPs employed to fit each output RPC model results from a virtual regular grid of $n \times n \times n$ samples in the object space. This grid is projected to the image space using the corrected mapping, to establish the 3D to 2D correspondences that are necessary to fit $\mathcal{P}_{\text{BA}}$ (Algorithm 5).

Algorithm 5: RPC model fitting using bundle adjustment solution (for one camera)

input : input RPC model, with unrefined projection function $\mathcal{P}$ and localization function $\mathcal{L}$
bundle adjusted corrective rotation, R
camera center, $\boldsymbol{C}$
altitude range, $[Z_{\min}, Z_{\max}]$ *this range can be defined based on the tie points 3D location*
number of samples per dimension, n [default value = 10] *defines a grid of $n \times n \times n$ points*
output : corrected RPC model, with a new projection function $\mathcal{P}_{\text{BA}}$
Set a certain margin (in pixel units), $m = 10$
Define a regular grid of $n \times n$ 2D points, G_{2D}, covering the input image + m in each dimension
Define n elevation layers $\{l_i\}_{i=1,\dots,10}$, uniformly distributed within $[Z_{\min}, Z_{\max}]$
Define a regular grid of $n \times n \times n$ 3D points, G_{3D}, by localizing G_{2D} at each elevation layer l using $\mathcal{L}$
Obtain the corrected image coordinates $(r_{\text{BA}}, c_{\text{BA}})$ of each point (X, Y, Z) in G_{3D} *Equation (18)*
if the convex hull of the set of $(r_{\text{BA}}, c_{\text{BA}})$ coordinates covers the entire input image **then**
  Fit $\mathcal{P}_{\text{BA}}$ by running [2] with the n^3 correspondences between (X, Y, Z) and $(r_{\text{BA}}, c_{\text{BA}})$
else
  Dilate the grid of fitting points by repeating all previous steps using $m = 2m$

3.3.1 Drift Correction in Object Space

The proposed methodology does not set any constraint to fix the bounding box that contains the 3D coordinates of the tie points employed by the bundle adjustment. The optimization process is therefore free to move the scene through the object space to find a better fit between the input cameras. As a result, a certain drift may exist between the initial and output location of the scene. To minimize any possible drift, we compose the corrected projection mapping of each camera with a translation in object space, $\boldsymbol{T}_{\text{drift}}$, which is common for all cameras. The corrected projection function that maps a 3D point $\boldsymbol{X}$ to its location $\boldsymbol{x}_{\text{BA}}$ in the m-th image evolves from (12) to

$$\boldsymbol{x}_{\text{BA}} = \mathcal{P}_m(R_m(\boldsymbol{X} + \boldsymbol{T}_{\text{drift}} - \boldsymbol{C}_m) + \boldsymbol{C}_m), \tag{18}$$

where $\boldsymbol{T}_{\text{drift}}$ is given by the solution of

$$\min_{\boldsymbol{T}_{\text{drift}}} \sum_{k=1}^{K} \|\boldsymbol{X}_k - (\hat{\boldsymbol{X}}_k + \boldsymbol{T}_{\text{drift}})\|^2. \tag{19}$$

Following equation (18), $\boldsymbol{T}_{\text{drift}}$ is applied previously to the corrected projection mapping found by bundle adjustment. In (19), $\hat{\boldsymbol{X}}_k$ represents the k-th tie point initial 3D location, while $\boldsymbol{X}_k$ is its location after the bundle adjustment. Therefore $\boldsymbol{T}_{\text{drift}}$ seeks to bring the tie points from the initial scene location to that location in the object space where the bundle adjustment found the optimal consistence between the input cameras. Note that the absolute location of the corrected camera models remains subject to the absolute localization error of the input RPC functions, but our objective is to ensure consistency between cameras so that they can be exploited for multi-view 3D reconstruction, not their exact absolute location.

4 Experiments

This section aims to assess the performance of the proposed bundle adjustment pipeline for RPC correction. First, we introduce the data, the evaluation procedure and its metrics. Secondly, we briefly describe a geometry based method for DSM registration, which aligns DSMs from different stereo pairs without any preceding RPC correction, and we compare our approach to it. Finally, the results of the conducted experiments are analyzed from a quantitative and qualitative point of view.

4.1 Data

We test our method on two time series of SkySat panchromatic L1B frames, each covering a specific AOI. Each time series consists of groups of partially overlapping images acquired at different dates (Figure 6). SkySat L1B frames have a nadir resolution of ∼0.72 m per pixel and a total size of 1349×3199 pixels. The main characteristics of SkySat acquisitions and each series are detailed next.

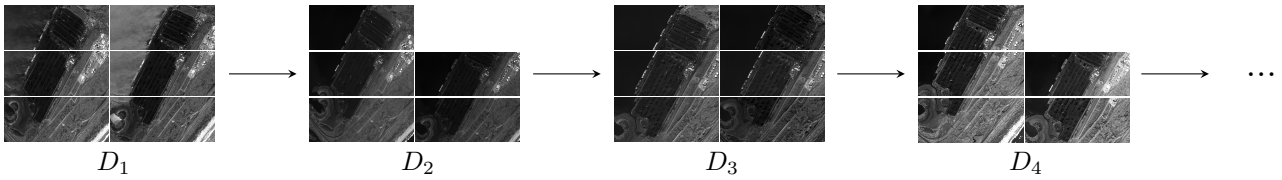


Figure 6: Illustration of a time series of SkySat imagery. A collection of partially overlapping views covering a certain area of interest is available at each acquisition date D_i of the time series.

4.1.1 SkySat Acquisition Geometry

SkySats have 3 staggered sensors in their acquisition platform (Figure 7(a)), and each sensor acquires a continuous strip of single frame images at a time (Figure 7(b)). The minimum swath width across the strips amounts to ∼6.6 km on the ground. There is a small overlap between images of different strips and between images of the same strip [14, 1].

SkySats can provide tri-stereo acquisitions, which can be used for 3D reconstruction [36]. The tri-stereo acquisition mode produces image triplets, consisting of a forward, nadir and backward view of each area, taken from the same orbit, with a time difference of a few seconds. Thus, the content of each strip is seen at least three times in the tri-stereo data (Figure 7(c)).

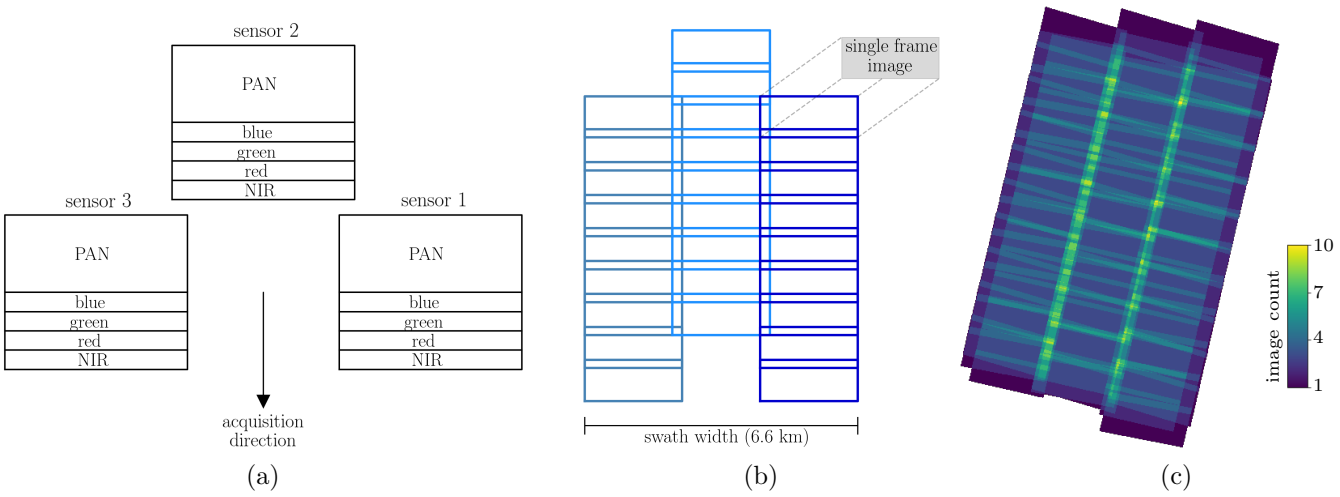
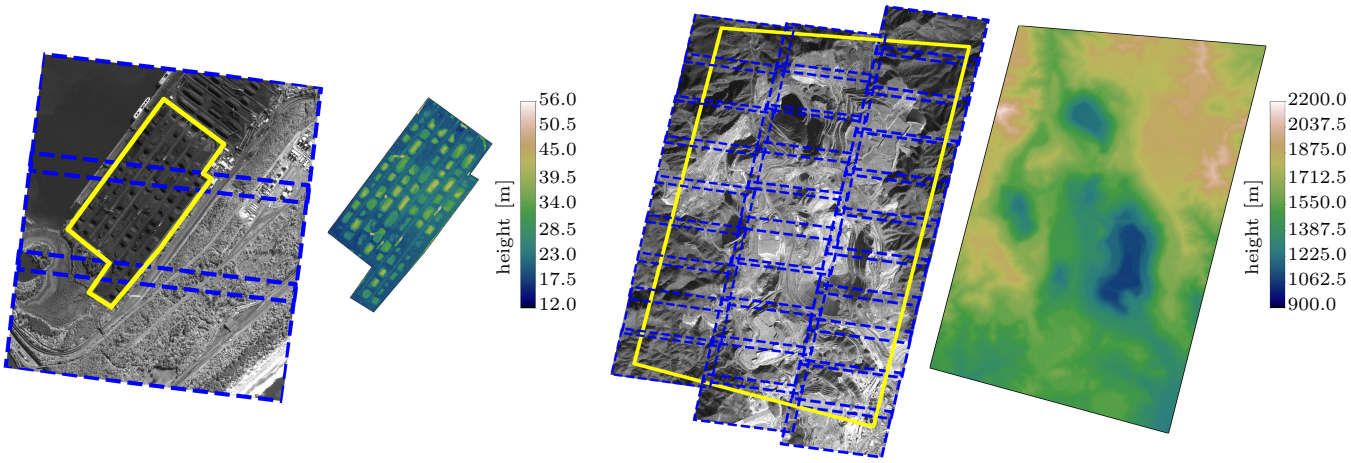


Figure 7: (a) Scheme of a SkySat focal plane. The upper half of the sensors is used for panchromatic capture (PAN), the lower half is divided into 4 bands with the blue, green, red and near infra-red (NIR) filters. (b) Scheme of a SkySat acquisition, where the strips of frames acquired by each sensor are outlined in different colors. (c) Number of overlapping views of a SkySat tri-stereo acquisition.

4.1.2 Time Series Description

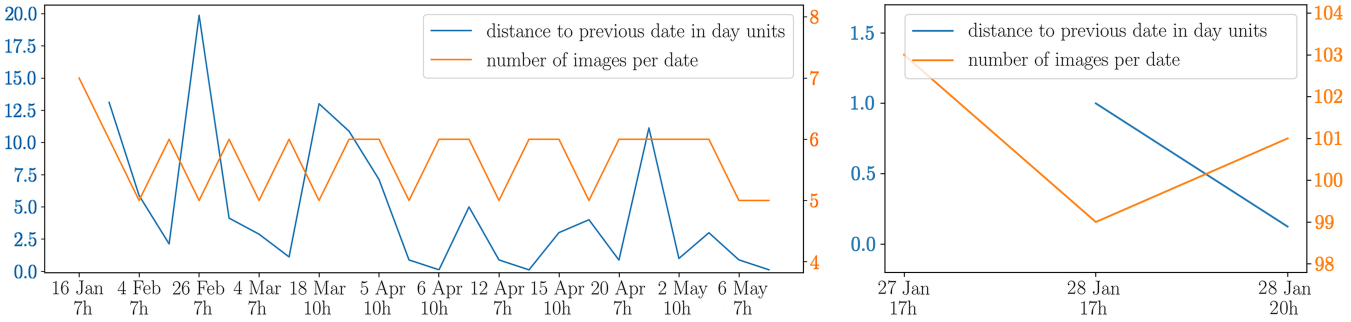
- **Richards Bay:** 136 images, distributed in groups of 5-6, over 24 acquisition dates from January to May 2020. The AOI measures 1 km^2 and covers part of the Richards Bay coal terminal (Figure 8(a)). The maximum time interval between consecutive dates is 20 days (Figure 9(a)).

- **Morenci Mine:** 3 entire tri-stereo acquisitions, from 3 different dates in January 2019. Each tri-stereo acquisition amounts to ∼100 images, resulting in a total of 303 frames. The AOI (Figure 8(b)) measures 62 km^2 and covers the Morenci copper mine. As shown in Figure 9(b), the time interval between consecutive dates is of a few hours.



(a) Richards Bay site (South Africa) (b) Morenci Mine site (United States)

Figure 8: On the left, the selected area of interest (AOI) of each time series, marked in yellow, covered by a collection of SkySat L1B frames outlined in dashed blue lines. On the right, a digital surface model (DSM) of the AOI, obtained by multi-view stereo reconstruction after RPC correction.



(a) Richards Bay time series. (b) Morenci Mine time series.

Figure 9: Number of images per acquisition date and time distance between consecutive dates.

Note that all images in the Richards Bay series were acquired by the same sensor (sensor 2 in Figure 7(a)), while the Morenci Mine acquisitions contain the 3 strips per date, i.e. images from the 3 SkySat sensors. Also, note that all the images are free from clouds in the concerned AOIs.

4.2 Evaluation Procedure

The RPC correction is done in a date-wise manner, as in [30], solving an independent bundle adjustment problem for the group of cameras of each date in the time series. This ensures that the scene geometry is coherent for each group, which is not guaranteed if multiple dates are treated at once.

We evaluate the performance of our method by measuring its practical impact on the registration of 3D reconstructions from different stereo pairs of the same acquisition date. To this end, for each stereo pair, a DSM is generated with the satellite stereo pipeline SP2 [15, 11] using the corrected RPC models resulting from the bundle adjustment. The resulting pairwise DSMs are restricted to

the part of the AOI that is visible in each stereo pair, as shown in Figure 10. If the RPC correction was successful, the pairwise DSMs of each date should be automatically registered in the object space. This allows us to easily merge them into a denser and highly accurate model of the entire AOI by taking the average height value in each cell of the DSM. Figure 10 illustrates the complete multi-view stereo reconstruction procedure that is carried out for each date of the time series.

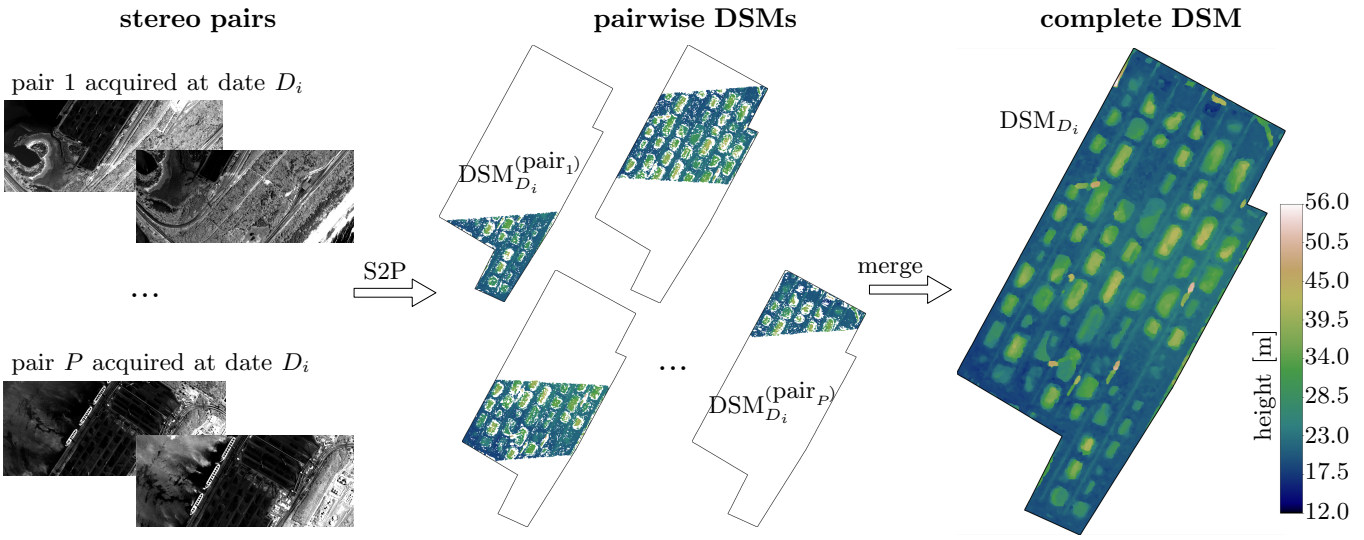


Figure 10: Multi-view stereo reconstruction process run at each acquisition date of the time series. The DSMs from different stereo pairs obtained with S2P [11], using the corrected RPC models, are merged into a complete DSM of the entire AOI by taking the average height value at each cell.

We compute the average evaluation metrics across all dates of each time series. Note that the proposed date-wise evaluation procedure does not seek to register DSMs from different acquisition dates. By splitting the time series into independent bundle adjustment problems, we aim to measure the degree of consistency between cameras that can be attained in a single date scenario, without the risk of reaching a solution that is affected by changes in the geometry of the AOI over time.

4.3 Evaluation Metrics

The following metrics are employed to assess the functioning of bundle adjustment:

- **Number of feature tracks:** denoted by N_{tracks}, it corresponds to the number of feature tracks employed to perform the bundle adjustment.
- **Number of iterations:** denoted by N_{iters}, it corresponds to the number of iterations completed by the numerical optimization. This amount results from the addition of the number of iterations required to minimize the cost functions (14) and (15).
- **Reprojection error:** denoted by ρ, it amounts to the average reprojection error, in terms of Euclidean distance and measured in pixels, that is obtained from all feature track observations.
- **DSM height deviation:** denoted by σ_{height}, this measure quantifies the consistency of corresponding height values retrieved from different stereo pairs of the same acquisition date. It results from computing the average point-wise standard deviation, in height and measured in meters, of the 2D points that are seen in more than one of the pairwise DSMs (i.e. middle image in Figure 10). Ideally, these DSMs should coincide exactly, with a standard deviation equal to zero. In practice, a small amount of noise derived from the computation of depth

from stereo can be expected, but this indicator should reach values as close to zero as possible. Visual examples of σ_{height} are shown in Figure 11. Small holes or unfilled parts in Figure 11 correspond to points of the AOI that are not seen by at least two pairwise DSMs, where σ_{height} cannot be computed.

4.4 Comparison with a Geometry Based DSM Registration

We compare the proposed bundle adjustment methodology to an alternative geometry based strategy that explicitly aligns dense surface models, described in Algorithm 6. As mentioned in Section 2, working with images with small geographic footprint allows the local approximation of RPCs as affine cameras. Geolocation errors related to RPC inaccuracies can then be corrected by applying 3D translations in the object space, to register the 3D models derived from different stereo pairs [18, 41].

Algorithm 6 aligns independent DSMs based on their 3D points and, additionally, it also uses the panchromatic frames originally employed to generate the DSMs. Using image based features to compute the transformations for point cloud registration is a well established methodology in 3D vision [40, 9, 35]. Under reasonable viewing angles, strategies of this kind can offer certain advantages (e.g. fast solution, robustness to different cloud formats or a coarse initialization) compared to classic exhaustive point cloud registration methods that exclusively rely on the 3D coordinates, such as the Iterative Closest Point (ICP) algorithm and its variants [6].

Algorithm 6: Geometry based DSM registration (N-inputs)

input : N pairwise DSMs, i.e. $\{DSM^{(n)}\}_{n=1,\dots,N}$
the pairs of images used to compute each DSM, i.e. $I_{\text{ref}}^{(n)}$, $I_{\text{aux}}^{(n)}$ for $DSM^{(n)}$
output : N co-registered DSMs, i.e. $\{DSM_{\text{r}}^{(n)}\}_{n=1,\dots,N}$
Compute all pairwise DSM intersections using the georeferenced bounds of the models
for each DSM intersection **do**
  Retrieve the indices of the intersecting DSMs, denoted by i and j
  Retrieve the reference image of each DSM, denoted by $I_{\text{ref}}^{(i)}$ and $I_{\text{ref}}^{(j)}$
  Find a set of matches, i.e. $\{\boldsymbol{x}_m^{(i)}, \boldsymbol{x}_m^{(j)}\}_{m=1,\dots,M}$, between SIFT keypoints in $I_{\text{ref}}^{(i)}$ and $I_{\text{ref}}^{(j)}$
  Retrieve $\{\boldsymbol{X}_m^{(i)}, \boldsymbol{X}_m^{(j)}\}_{m=1,\dots,M}$, where $\boldsymbol{X}_m^{(i)}$ are the 3D coordinates of $\boldsymbol{x}_m^{(i)}$ as given by $DSM^{(i)}$
  Define the equations derived from the M 3D matches resulting from the previous step:
    The m-th correspondence amounts to $\boldsymbol{A}_m y = \boldsymbol{b}_m$, where $\boldsymbol{b}_m = \boldsymbol{X}_m^{(j)} - \boldsymbol{X}_m^{(i)}$,
    $\boldsymbol{A}_m$ is a sparse vector of length N with 1 and -1 at i-th and j-th positions,
    and y is the solution matrix with shape $N \times 3$
Build the linear system $Ay = b$, where A and b result from stacking all rows $\boldsymbol{A}_m$ and $\boldsymbol{b}_m$
Solve $\min_y ||Ay - b||$ to find the optimal solution y
for each input DSM **do**
  Obtain $DSM_{\text{r}}^{(n)}$ by applying the 3D translation $\boldsymbol{t}^{(n)}$ to all the 3D points in $DSM^{(n)}$,
  where $\boldsymbol{t}^{(n)}$ is the n-th row of y

As noted in [41], DSM registration strategies offer little scalability for very large-scale scenarios, with possibly hundreds of surface models to align. In line with the evaluation procedure introduced in Section 4.2, we run Algorithm 6 in a date-wise manner, to register the DSMs obtained from stereo pairs of the same acquisition date. The resulting height deviation, σ_{height}, is compared to the value obtained by the analogous experiments relying on the proposed RPC correction pipeline.

4.5 Results

We conducted experiments using three different configurations for the proposed bundle adjustment methodology. BA-v1 corresponds to the presented bundle adjustment pipeline using only its non-optional blocks as shown in Figure 2. BA-v2 incorporates the two optional blocks from the bundle adjustment stage, dedicated to filtering outlier tie point observations. BA-v3 additionally activates the feature tracks selection, thus employing all blocks in Figure 2. The resulting evaluation metrics are shown in Table 2. The metrics obtained with the original unrefined RPC models and Algorithm 6, which eludes any RPC correction, are also reported.

Similarly to [1], for the Morenci Mine data, the images of each SkySat sensor are treated independently, i.e. the AOI is divided into three smaller AOIs corresponding to each sensor strip and we provide the average evaluation metrics across the three strips. This excludes from the evaluation all possible inconsistencies between the intrinsic parameters of the three sensors (e.g. lens distortion).

The most important ideas that come out of Table 2 are discussed below.

		N_{tracks}	N_{iter}	ρ before BA [px]	ρ after BA [px]	σ_{height} [m]
Richards Bay	Unrefined RPC models	-	-	-	-	1.472
	Geometry based DSM registration	-	-	-	-	0.304
	Basic BA RPC correction (BA-v1)	4310	17	1.327	0.133	0.291
	BA-v1 + filter after soft-ℓ_1 iter. (BA-v2)	4309	50	1.326	0.132	0.290
	BA-v2 + feature tracks select. (BA-v3)	197	35	1.697	0.167	0.295
Morenci Mine	Unrefined RPC models	-	-	-	-	2.675
	Geometry based DSM registration	-	-	-	-	0.588
	Basic BA RPC correction (BA-v1)	76943	8	1.352	0.178	0.662
	BA-v1 + filter after soft-ℓ_1 iter. (BA-v2)	76594	94	1.291	0.129	0.434
	BA-v2 + feature tracks select. (BA-v3)	1094	56	1.829	0.206	0.370

Table 2: Quantitative results obtained for the two time series of SkySat imagery. Geometry based DSM registration refers to Algorithm 6. BA-v1 corresponds to the presented bundle adjustment pipeline using all its non-optional blocks (Figure 2). BA-v2 incorporates the two optional blocks from the bundle adjustment stage, dedicated to filtering outlier tie point observations. Finally, BA-v3 additionally activates the feature tracks selection, thus employing all blocks in Figure 2.

BA RPC correction results in accurate DSM registration. The bundle adjusted RPC models succeed to accurately align in the object space the DSMs reconstructed from different stereo pairs of the same acquisition date. This is reflected by the low deviation of height values, σ_{height}, which reaches average values of a few tens of centimeters for both series, normally below half a meter. We obtain similar σ_{height} values for Richards Bay, regardless of the bundle adjustment configuration (BA-v1, BA-v2 or BA-v3). This seems reasonable as Richards Bay represents a not particularly challenging scenario, where the AOI is small and mostly flat. Things are different for the Morenci Mine, which covers a vast and mountainous AOI, with a more irregular relief. Thus, a wider range of altitudes and a larger amount of feature tracks and cameras needs to be handled. The σ_{height} associated with the Morenci Mine strongly improves from BA-v1 to BA-v3, showing the benefits of the reprojection error filtering after some initial iterations (BA-v2), as well as the feature track selection (BA-v3). Using the complete pipeline (BA-v3) reduces σ_{height} almost to the half of what is obtained with the basic blocks (BA-v1).

BA RPC correction offers higher robustness than geometry based DSM registration. Using the RPC models corrected with the complete presented pipeline (BA-v3) provided lower σ_{height}

than the equivalent experiments using geometry based DSM registration (Algorithm 6). Again, the results were close for the Richards Bay data, but stronger differences are spotted for the Morenci Mine, whose size requires the alignment of a larger set of independent pairwise DSMs (Figure 11). Visual inspection reveals that this is because the registration methods exclusively based on the geometry of dense models are subject to poor results derived from the stereo reconstruction process. Algorithm 6 compensates geolocation errors derived from the camera models by assuming that the input DSM geometries are coherent, which may not always be true, e.g. due to outlier points or incomplete parts. Oppositely, our bundle adjustment methodology for RPC correction is applied before the stereo reconstruction process, in a totally independent manner, and optimizes the camera models jointly with the geometry of the scene. The fact that bundle adjustment can natively handle initial inaccurate geometry estimates is a key advantage.

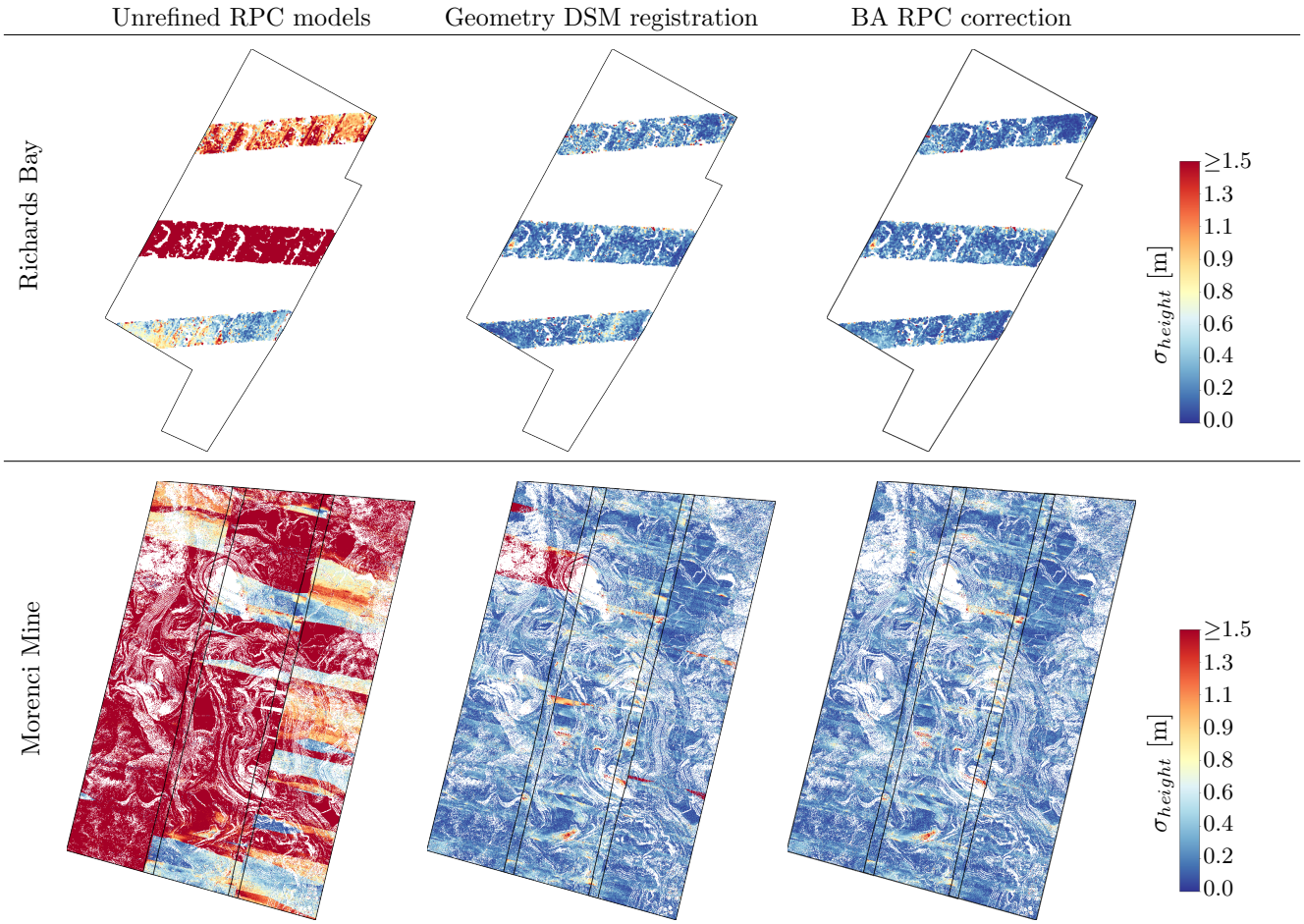


Figure 11: Deviation of corresponding DSM height values, σ_{height}, for an acquisition date of each time series. 144 DSMs from different stereo pairs are used to cover the Morenci Mine AOI (48 DSMs for each sensor strip), while 4 pairwise DSMs are enough for the smaller AOI in Richards Bay. σ_{height} is computed where at least two pairwise DSMs overlap, which only occurs in some bands for Richards Bay. Geometry registration after depth from stereo fails to align some DSMs near the top-left corner of the Morenci Mine, which is corrected when bundle adjusted RPCs are used to compute the models.

The proposed pipeline may struggle in challenging scenarios for stereo matching, which is necessary to guarantee that all cameras are connected in terms of tie point observations. However, dense stereo reconstruction is also prone to failure in front of difficulties to find stereo matches. By extension, this weakness is shared indirectly (to a certain degree) with any geometry registration method for dense reconstructions from stereo-photogrammetry, as is the case of Algorithm 6.

Feature track selection improves bundle adjustment convergence and accuracy. The average number of bundle adjustment iterations is reduced significantly when a suitable subset of feature tracks is selected using Algorithm 4. N_{iter} drops by at least 30% from BA-v2 to BA-v3 in Table 2. For Richards Bay, BA-v3 attains very similar DSM registration (σ_{height}) to the equivalent run where all tracks are used (BA-v2). For the Morenci Mine, σ_{height} improves significantly using BA-v3. The more cameras to be registered, the more important it seems to prioritize long tracks and the use of a balanced set across all images. In Algorithm 4 we set $K_{\text{EG}} = 60$, to ensure at least 60 track observations are used per camera, if available. The reprojection error filtering step of our pipeline (Section 3.2.3) may further reduce the tie point observations per camera after feature track selection, so we chose a more generous value for K_{EG} than the one suggested in [12], where $K_{\text{EG}} = 30$.

5 Conclusion

A generic pipeline to refine the RPC camera models of satellite imagery was presented and tested using collections of SkySat images acquired across multiple dates over two different areas of interest. Errors due to inaccuracies in sensor orientation are corrected by composing the original RPC projection function with a rotation around the camera center. The method relies on the identification of distinctive point correspondences between images, which are employed to automatically infer a set of tie points observed across the input cameras. The corrective rotation associated with each RPC model is found by running a bundle adjustment that minimizes the reprojection error of the tie points. A strategy for selecting an optimal subset of tie points was also detailed to remove redundant constraints from the bundle adjustment problem and improve its performance.

For each input camera, the presented pipeline outputs a new and corrected RPC model based on the bundle adjustment solution. The output cameras are highly consistent, in a common frame of reference, and can therefore be used for 3D multi-view reconstruction in a straightforward manner. This is validated by measuring the standard deviation between corresponding height values of surface models computed from different pairs of images of the same acquisition date. Using SkySat L1B data we observe that the average value of such deviation decreases from the order of meters to the order of a few tens of centimeters after employing the refined camera models.

Finally, the presented method was compared to a different state of the art solution, which eludes any RPC correction step by directly registering the geometry of independent local models. When a sufficient amount of tie points connecting all cameras is available, we obtain that the bundle adjustment for RPC correction exhibits better performance. This is mainly because bundle adjustment offers robustness to initial inaccurate geometry estimates, as a consequence of simultaneously optimizing not only the camera models but also the scene geometry. Oppositely, 3D model registration methods purely based on geometry are subject to errors derived from dense stereo reconstructions.

A Fundamentals of Bundle Adjustment Optimization

This appendix overviews the fundamental concepts of numerical analysis behind the optimization algorithms employed to solve bundle adjustment problems [10, 43].

Bundle Adjustment as Nonlinear Weighted Least Squares

Given a set of K 3D points $\boldsymbol{X}_k$ seen across M cameras $\mathcal{P}_m$, the 2D observation of a point $\boldsymbol{X}_k$ in the m-th image is denoted by $\boldsymbol{x}_{mk}$. The bundle adjustment problem aims at minimizing the reprojection distance $r_i = \|\boldsymbol{x}_{mk} - \mathcal{P}_m(\boldsymbol{X}_k)\|$, of all points in all cameras, by jointly optimizing the parameters of

the camera projection functions $\mathcal{P}_m$ and the positions of the 3D points $\boldsymbol{X}_k$

$$\sum_{k=1}^{K}\sum_{m=1}^{M}\|\boldsymbol{x}_{mk}-\mathcal{P}_m(\boldsymbol{X}_k)\|^2. \tag{20}$$

If each 3D point is seen by all cameras, the total number of 2D observations $\boldsymbol{x}_{mk}$ is $N = K \cdot M$.

Equation (20) defines a nonlinear weighted least squares problem. For the following discussion we rewrite it as a generic problem of the form

$$\min_{\mathbf{x}} \frac{1}{2}\sum_{i=1}^{N} w_i r_i^2(\mathbf{x}), \tag{21}$$

where r_i is a set of N nonlinear functions, $\mathbf{x} = (x_1, \ldots, x_L)$ is a vector of length L with all the parameters to estimate, and w_i are the weights. By defining W as a diagonal matrix with the weights w_i, and $\mathbf{r} = (r_1, \ldots, r_N)$ as the vector of nonlinear functions, omitting the dependency on $\mathbf{x}$, the above expression rewrites as

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{r}^T W\,\mathbf{r}. \tag{22}$$

We will detail the fundamentals behind the Gauss–Newton algorithm, which iteratively finds the value of the variables $\mathbf{x}$ starting from an initial guess $\mathbf{x}^0$ by iterating

$$\mathbf{x}^{t+1} = \mathbf{x}^t - (J^T W J)^{-1} J^T W \mathbf{r}(\mathbf{x}^t), \tag{23}$$

where $J = (J)_{ij} = \frac{\partial r_i(\mathbf{x}^t)}{\partial x_j}$ is the Jacobian matrix of $\mathbf{r}(\mathbf{x}^t)$.

Newton's Method

Given a twice differentiable function $f : \mathbb{R}^N \to \mathbb{R}$, we seek to solve the optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}). \tag{24}$$

This can be solved by gradient descent, which constructs a sequence $\{\mathbf{x}^t\}$ from an initial guess $\mathbf{x}^0$ by taking small steps $\delta\mathbf{x} = -\mu\mathbf{g}$ in the direction of the gradient $\mathbf{g}$,

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \delta\mathbf{x}. \tag{25}$$

The step $\delta\mathbf{x}$ must be set small enough (by controlling μ) to assure that $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$ at each iteration until convergence.

Newton's method improves the gradient descent strategy by determining the step at each iteration $\mathbf{x}^t$ by solving a second-order Taylor approximation of f

$$f(\mathbf{x}+\delta\mathbf{x}) \simeq f(\mathbf{x}) + \mathbf{g}^T\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T H\delta\mathbf{x}, \tag{26}$$

where $\mathbf{g}$ and H represent the gradient and Hessian of f, respectively. The local minimum of (26) is obtained by setting its differential to 0. This yields the optimal step

$$\delta\mathbf{x} = -H^{-1}\mathbf{g}, \tag{27}$$

which is known as the Newton step. Based on this definition, Newton's method iterates according to (25) and estimating $\delta\mathbf{x}$ at each iteration as defined in (27).

Newton's method is faster than a classic gradient descent algorithm. Unfortunately it can fail in many ways: it may converge to a saddle point rather than a minimum or it may be unstable if the Hessian is close to a singular matrix. To enforce stability, a *damped* version of (27) can be used, i.e.

$$\delta\mathbf{x} = -(H+\lambda I)^{-1}\mathbf{g}, \tag{28}$$

where λ is some weighting factor and I is the identity.

Gauss-Newton Method

When applying Newton's method to a nonlinear weighted least squares problem as (22), the gradient and Hessian of f can be expressed in terms of the Jacobian matrix of $\mathbf{r}(\mathbf{x})$. Given

$$J = (J)_{ij} = \frac{\partial r_i(\mathbf{x})}{\partial x_j}, \tag{29}$$

the gradient writes as

$$\mathbf{g} = J^T W \mathbf{r}, \tag{30}$$

and the Hessian as

$$H = J^T W J + S, \tag{31}$$

where $(S)_{jk} = \sum_{i=1}^{N} w_i r_i(\mathbf{x}) \frac{\partial^2 r_i(\mathbf{x})}{\partial x_j \partial x_k}$. The S term is usually very small compared to $J^T W J$ if the current $\mathbf{x}$ is not too far from a minimum, so it can be omitted leading to the Gauss-Newton approximation of the problem, which consequently defines $\delta\mathbf{x}$ as

$$\delta\mathbf{x} = -(J^T W J)^{-1} J^T W \mathbf{r}. \tag{32}$$

In the same way as (28), a damped Gauss-Newton step can also be defined

$$\delta\mathbf{x} = -(J^T W J + \lambda I)^{-1} J^T W \mathbf{r}. \tag{33}$$

Equation (33) is used by the Levenberg-Marquardt algorithm, which is a classic and well-known iterative algorithm to solve bundle adjustment problems.

Damped Newton methods use λ to regulate a trade-off between the Newton and gradient descent directions, i.e. if $\lambda \to \infty$ they behave as a gradient descent ($\delta\mathbf{x} \propto -\mathbf{g}$). λ can be chosen to limit the step to a dynamically chosen maximum size (Trust Region methods), or manipulated more heuristically, to shorten the step if the prediction is poor (Levenberg-Marquardt methods).

Construction of the Jacobian Matrix

The Jacobian J of a bundle adjustment problem corresponds to a large sparse matrix [10]. This can be easily illustrated with a toy example. Suppose we have $M = 3$ cameras and $K = 4$ 3D points. For simplicity we assume that each point is seen across all cameras. J can be expressed as

$$J = \begin{array}{c} \\ \text{number of 2D observations} \\ \downarrow \end{array} \begin{array}{c} \overrightarrow{\text{number of parameters to estimate}} \\ \begin{array}{ccccccc} c_1 & c_2 & c_3 & p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{pmatrix} A_1 & 0 & 0 & B_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & B_2 & 0 & 0 & 0 \\ 0 & 0 & A_3 & B_3 & 0 & 0 & 0 \\ A_4 & 0 & 0 & 0 & B_4 & 0 & 0 \\ 0 & A_5 & 0 & 0 & B_5 & 0 & 0 \\ 0 & 0 & A_6 & 0 & B_6 & 0 & 0 \\ A_7 & 0 & 0 & 0 & 0 & B_7 & 0 \\ 0 & A_8 & 0 & 0 & 0 & B_8 & 0 \\ 0 & 0 & A_9 & 0 & 0 & B_9 & 0 \\ A_{10} & 0 & 0 & 0 & 0 & 0 & B_{10} \\ 0 & A_{11} & 0 & 0 & 0 & 0 & B_{11} \\ 0 & 0 & A_{12} & 0 & 0 & 0 & B_{12} \end{pmatrix} \end{array} \begin{array}{l} \\ (p_1, c_1) \\ (p_1, c_2) \\ (p_1, c_3) \\ (p_2, c_1) \\ (p_2, c_2) \\ (p_2, c_3) \\ (p_3, c_1) \\ (p_3, c_2) \\ (p_3, c_3) \\ (p_4, c_1) \\ (p_4, c_2) \\ (p_4, c_3) \end{array}, \tag{34}$$

where the i-th row contains the partial derivatives of the reprojection error r_i related to the i-th feature observation, which corresponds to the k-th point seen in the m-th camera, i.e. (p_k, c_m).

The vector of parameters to optimize $\mathbf{x}$ usually follows a camera-structure order, $\mathbf{x} = [c, p]$, where c and p are the sets of camera parameters and 3D point coordinates to estimate, respectively. Since r_i is a norm of a residual we can write $r_i^2 = rx_i^2 + ry_i^2$, where rx and ry represent the components of the residual. Thus the first block of columns in J contains the partial derivatives with respect to the camera parameters c_m for each dimension of the image plane, i.e. $A_i = \frac{\partial r_i}{\partial c_m} = [\frac{\partial rx_i}{\partial c_m}, \frac{\partial ry_i}{\partial c_m}]^T$, while the second block of columns contains the partial derivatives with respect to the coordinates of the 3D points p_k, i.e. $B_i = \frac{\partial r_i}{\partial p_k} = [\frac{\partial rx_i}{\partial p_k}, \frac{\partial ry_i}{\partial p_k}]^T$. Therefore J has a total size equal to

$$(2 \cdot N) \times (q_c \cdot M + q_p \cdot K), \tag{35}$$

where

- K is the number of 3D points, 4 in this example.
- M is the number of cameras, 3 in this example.
- N is the number of 2D observations. $N = K \cdot M = 12$ in this example, since we assume that each point is observed in all cameras.
- q_c is the number of parameters to optimize per camera and defines the number of columns of each block A_i in (34). E.g. $q_c = 3$ if the camera parameters to estimate consist of 3 Euler angles related to the camera orientation, as presented in this work.
- q_p is the number of parameters to optimize per 3D point and defines the number of columns of each block B_i in (34). E.g. $q_p = 3$ if all 3D coordinates of each point are optimized.

Finite Differences Approximation and Sparsity Structure of the Jacobian

Computing the Jacobian matrix J of a cost function $f(\mathbf{x})$ can be difficult for large sparse problems as in bundle adjustment. Finite differences are a popular approach to approximate J in a simpler and faster manner. Instead of explicitly computing the partial derivatives, these are approximated as a difference between the values of $f(\mathbf{x})$, e.g. $f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x})$ for a simple forward difference with step $\mathbf{h}$. Under this approximation, finite differences offer the advantage that the cost function can be treated as a black box in the optimization process.

The task can be further simplified by providing a *sparsity structure* that indicates where to place each approximated partial derivative in J [13, 4]. The sparsity structure of J, denoted by J_S, corresponds to the binarization of J into zeros and non-zeros. J_S can be used to minimize the number of function evaluations that are necessary to compute J. Instead of constructing each column of J individually, J_S allows to group the columns beforehand into structurally orthogonal groups, where the columns in each group do not have non-zeros in the same row position. This is useful to speed up the process, since it allows the calculation of the partial derivatives of each group at once.

B Code Structure

This appendix summarizes the structure of the Python code associated with this paper. The `main` script and the `BundleAdjustmentPipeline` class are described in Table 3. The rest of scripts are briefly commented in Table 4. Usage instructions are included in the `README` file.

File	Description
`main.py`	Reads the input data and uses it to initialize an instance of the `BundleAdjustmentPipeline` class. Afterwards, it calls `BundleAdjustmentPipeline.run()`, which runs the RPC correction methodology described in this paper.

Table 3: Summary of the `main` script and the `BundleAdustmentPipeline` class. Continued on next page.

Table 3 – Continued from previous page

File	Description
`ba_pipeline.py`	Implements the `BundleAdjustmentPipeline` class, illustrated by the block diagram in Figure 2. The chain is executed by `run()`, which sequentially launches the list of methods below. For each method, the blocks in Figure 2 that are covered and the related scripts of code are indicated. - `compute_feature_tracks()`: Computes a set of feature tracks. This can be done from zero using an instance of `FeatureTracksPipeline` or by recycling a series of predefined pairwise matches. Blocks Figure 2: *feature detection*, *stereo pairs selection*, *pairwise matching* and *feature tracks construction*. Related scripts: `ft_pipeline.py`, `ft_opencv.py`, `ft_s2p.py`, `ft_match.py`, `ft_utils.py`. - `initialize_pts3d()`: Computes the 3D tie point associated with each feature track. Blocks Figure 2: *feature tracks triangulation*. Related scripts: `ft_triangulation.py`. - `select_best_tracks()`: Selects a subset of optimal feature tracks (or equivalently tie points). Blocks Figure 2: *feature tracks selection*. Related scripts: `ft_ranking.py`. - `define_ba_parameters()`: Takes the input cameras, feature tracks and tie points and creates an instance of `BundleAdjustmentParameters`, which is the data format employed by the bundle adjustment numerical optimization. Blocks Figure 2: *define BA parameters*. Related scripts: `ba_params.py`. - `run_ba_softL1()`: Runs some initial bundle adjustment iterations using the cost function (14). Blocks Figure 2: *Run BA (soft-ℓ_1 cost)*. Related scripts: `ba_core.py`. - `clean_outlier_observations()`: Removes possibly erroneous feature track observations based on their reprojection error. Blocks Figure 2: *Reprojection error filtering*. Related scripts: `ba_outliers.py`. - `run_ba_L2()`: Runs the bundle adjustment using the cost function (15). Blocks Figure 2: *Run BA (ℓ_2 cost)*. Related scripts: `ba_core.py`. - `save_corrected_cameras()`: Fits a corrected RPC model for each input camera. Blocks Figure 2: *Fit corrected RPCs*. Related scripts: `ba_rpcfit.py`.

	File	Description
Feature tracking stage	`ft_opencv.py`	Functions to detect and match SIFT keypoints using the OpenCV library.
	`ft_s2p.py`	Functions to detect and match SIFT keypoints using the S2P library.
	`ft_match.py`	Functions to compute the stereo pairs to match (Algorithm 1) and restrict the pairwise matching of keypoints to the geographic area where two satellite images intersect. Calls `ft_opencv.py` or `ft_s2p.py` depending on which implementation of SIFT is chosen.
	`ft_utils.py`	Functions for feature tracks construction from pairwise matches (Algorithm 2) and other secondary tasks such as verifying that all cameras are properly connected.
	`ft_triangulate.py`	Functions for feature tracks triangulation (Algorithm 3).
	`ft_ranking.py`	Functions for feature tracks selection (Algorithm 4).
	`ft_pipeline.py`	Implements the `FeatureTracksPipeline` class, which coordinates the execution of the functions from all scripts in `feature_tracks` to produce a set of feature tracks connecting an input collection of satellite images.

Table 4: Summary of all the scripts used by an instance of the `BundleAdjustmentPipeline` class. Files are classified according to the stage where they intervene in the pipeline (Figure 2), except for the third row, which groups generic tools that are not specific to a certain task. Continued on next page.

Table 4 – Continued from previous page

	File	Description
Bundle adjustment stage	`ba_params.py`	Implements the `BundleAdjustmentParameters` class, which sets all variables of the problem in the data format used to feed the numerical optimization process.
	`ba_core.py`	Functions directly used during numerical optimization: definition of Jacobian sparsity structure, computation of reprojection error and launching of Trust Region algorithm.
	`ba_outliers.py`	Functions employed to remove outlier feature track observations from an instance of `BundleAdjustmentParameters` based on the reprojection error.
	`ba_rpcfit.py`	Functions to run the regularized weighted least squares algorithm employed to fit the corrected RPC models based on the bundle adjustment output (Algorithm 5).
	`cam_utils.py`	Functions dedicated to handle camera models: composition and decomposition, local approximation of a projection matrix from an RPC model and other secondary tasks.
	`geo_utils.py`	Functions dedicated to deal with different geographic coordinate systems and the GeoJSON format, which is used to represent the boundaries of geographical areas.
	`loader.py`	Functions dedicated to load and store data on the disk.

Third-party Code

The files commented in Table 3 and Table 4 were specifically coded for this paper. In addition, the implementation also uses third-party scripts from the S2P pipeline [11] and the implementation of the SIFT method from [37]. These are found in the `c` and `3rdparty/sift` directories of the code, respectively. The third-party code, in C, is employed for the detection of SIFT keypoints and the triangulation of pairwise correspondences using RPC models. It is handled by a series of Python bindings, which are found in the directory `bundle_adjust/s2p`.

Acknowledgment

This work was supported by a grant from Région Île-de-France. It was also partly financed by IDEX Paris-Saclay IDI 2016, ANR-11-IDEX-0003-02, Office of Naval research grant N00014-17-1-2552, DGA Astrid project « filmer la Terre » n° ANR-17-ASTR-0013-01, MENRT.

Image Credits

The Planet SkySat images used in this work were provided by Kayrros SAS.

References

[1] S. Aati and J-P. Avouac, *Optimization of optical image geometric modeling, application to topography extraction and topographic change measurements using PlanetScope and SkySat imagery*, Remote Sensing, 12 (2020), p. 3418. https://doi.org/10.3390/rs12203418.

[2] R. Akiki, R. Marí, C. de Franchis, J-M. Morel, and G. Facciolo, *Robust rational polynomial camera modelling for SAR and pushbroom imaging*, 2021. https://arxiv.org/abs/2102.13423.

[3] A. Aravkin, M. Styer, Z. Moratto, A. Nefian, and M. Broxton, *Student's T robust bundle adjustment algorithm*, in IEEE International Conference on Image Processing (ICIP), 2012, pp. 1757–1760. https://doi.org/10.1109/ICIP.2012.6467220.

[4] B.M. Averick, J.J. Moré, C.H. Bischof, A. Carle, and A. Griewank, *Computing large sparse Jacobian matrices using automatic differentiation*, SIAM Journal on Scientific Computing, 15 (1994), pp. 285–294. https://doi.org/10.1137/0915020.

[5] F.V. Berghen, *Levenberg-Marquardt algorithms vs Trust Region algorithms*, 2004. http://www.applied-mathematics.net/LMvsTR/LMvsTR.pdf.

[6] P.J. Besl and N.D. McKay, *A method for registration of 3-D shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14 (1992), pp. 239–256. https://doi.org/10.1109/34.121791.

[7] R.A. Beyer, O. Alexandrov, and S. McMichael, *The Ames Stereo Pipeline: NASA's open source software for deriving and processing terrain data*, Earth and Space Science, 5 (2018), pp. 537–548. https://doi.org/10.1029/2018EA000409.

[8] M.A. Branch, T.F. Coleman, and Y. Li, *A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems*, SIAM Journal on Scientific Computing, 21 (1999), pp. 1–23. https://doi.org/10.1137/S1064827595289108.

[9] S. Byun, K. Jung, S. Im, and M. Chang, *Registration of 3D scan data using image reprojection*, International Journal of Precision Engineering and Manufacturing, 18 (2017), pp. 1221–1229. https://doi.org/10.1007/s12541-017-0143-z.

[10] Y. Chen, Y. Chen, and G. Wang, *Bundle adjustment revisited*, 2019. https://arxiv.org/abs/1912.03858.

[11] CMLA and CNES, *S2P - Satellite Stereo Pipeline*, 2020. https://github.com/cmla/s2p.

[12] H. Cui, S. Shen, and Z. Hu, *Tracks selection for robust, efficient and scalable large-scale structure from motion*, Pattern Recognition, 72 (2017), pp. 341–354. https://doi.org/10.1016/j.patcog.2017.08.002.

[13] A.R. Curtis, M.J.D. Powell, and J.K. Reid, *On the estimation of sparse Jacobian matrices*, IMA Journal of Applied Mathematics, 13 (1974), pp. 117–119. https://doi.org/10.1093/imamat/13.1.117.

[14] P. d'Angelo, G. Kuschk, and P. Reinartz, *Evaluation of Skybox video and still image products*, ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 40-1 (2014), pp. 95–99. https://doi.org/10.5194/isprsarchives-XL-1-95-2014.

[15] C. De Franchis, E. Meinhardt-Llopis, J. Michel, J-M. Morel, and G. Facciolo, *An automatic and modular stereo pipeline for pushbroom images*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2-3 (2014), pp. 49–56. https://doi.org/10.5194/isprsannals-II-3-49-2014.

[16] C. de Franchis, E. Meinhardt-Llopis, J. Michel, J-M. Morel, and G. Facciolo, *On stereo-rectification of pushbroom images*, in International Conference on Image Processing (ICIP), 2014. https://doi.org/10.1109/ICIP.2014.7026102.

[17] P. d'Angelo and P. Reinartz, *DSM based orientation of large stereo satellite image blocks*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 39-B1 (2012), pp. 209–214. https://doi.org/10.5194/isprsarchives-XXXIX-B1-209-2012.

[18] G. Facciolo, C. De Franchis, and E. Meinhardt-Llopis, *Automatic 3D reconstruction from multi-date satellite images*, in IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017. https://doi.org/10.1109/CVPRW.2017.198.

[19] D. C-L. Fong and M. Saunders, *LSMR: An iterative algorithm for sparse least-squares problems*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2950–2971. https://doi.org/10.1137/10079687X.

[20] C.S. Fraser and H.B. Hanley, *Bias-compensated RPCs for sensor orientation of high-resolution satellite imagery*, Photogrammetric Engineering & Remote Sensing, 71 (2005), pp. 909–915. https://doi.org/10.14358/PERS.71.8.909.

[21] K. Gong and D. Fritsch, *Relative orientation and modified piecewise epipolar resampling for high resolution satellite images*, ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42-1/W1 (2017), pp. 579–586. https://doi.org/10.5194/isprs-archives-XLII-1-W1-579-2017.

[22] ——, *Point cloud and digital surface model generation from high resolution multiple view stereo satellite imagery*, ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42-2 (2018), pp. 363–370. https://doi.org/10.5194/isprs-archives-XLII-2-363-2018.

[23] J. Grodecki and G. Dial, *Block adjustment of high-resolution satellite images described by rational polynomials*, Photogrammetric Engineering & Remote Sensing, 69 (2003), pp. 59–68. https://doi.org/10.14358/PERS.69.1.59.

[24] Y. Han, S. Wang, D. Gong, Y. Wang, and X. Ma, *State of the art in digital surface modelling from multi-view high-resolution satellite images*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 5-2-2020 (2020), pp. 351–356. https://doi.org/10.5194/isprs-annals-V-2-2020-351-2020.

[25] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, second ed., 2004. https://doi.org/10.1017/CBO9780511811685.

[26] M.J. Leotta, C. Long, B. Jacquet, M. Zins, D. Lipsa, J. Shan, B. Xu, Z. Li, X. Zhang, and S-F. Chang, *Urban semantic 3D reconstruction from multiview satellite imagery*, in IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019. https://doi.org/10.1109/CVPRW.2019.00186.

[27] M.L.A. Lourakis and A.A. Argyros, *Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?*, in IEEE International Conference on Computer Vision, vol. 2, 2005, pp. 1526–1531. https://doi.org/10.1109/ICCV.2005.128.

[28] D.G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60 (2004), pp. 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[29] R. Marí, C. de Franchis, E. Meinhardt-Llopis, and G. Facciolo, *To bundle adjust or not: A comparison of relative geolocation correction strategies for satellite multi-view stereo*, in IEEE International Conference on Computer Vision Workshops, 2019. https://doi.org/10.1109/ICCVW.2019.00274.

[30] ——, *Automatic stockpile volume monitoring using multi-view stereo from SkySat imagery*, 2021. https://arxiv.org/abs/2103.00945.

[31] P. Moulon and P. Monasse, *Unordered feature tracking made fast and easy*, in European Conference on Visual Media Production (CVMP), 2012.

[32] M. Muja and D.G. Lowe, *Scalable nearest neighbor algorithms for high dimensional data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (2014), pp. 2227–2240. https://doi.org/10.1109/TPAMI.2014.2321376.

[33] K. Murthy, M. Shearn, B.D. Smiley, A.H. Chau, J. Levine, and M.D. Robinson, *SkySat-1: Very high-resolution imagery from a small satellite*, in Sensors, Systems, and Next-Generation Satellites XVIII, vol. 9241, International Society for Optics and Photonics, 2014, p. 92411E. https://doi.org/10.1117/12.2074163.

[34] O.C. Ozcanli, Y. Dong, J.L. Mundy, H. Webb, R. Hammoud, and T. Victor, *Automatic geo-location correction of satellite imagery*, in IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 307–314. https://doi.org/10.1109/CVPRW.2014.54.

[35] R.A. Persad and C. Armenakis, *Automatic co-registration of 3D multi-sensor point clouds*, ISPRS Journal of Photogrammetry and Remote Sensing, 130 (2017), pp. 162–186. https://doi.org/10.1016/j.isprsjprs.2017.05.014.

[36] H. Raggam, *Surface mapping using image triplets*, Photogrammetric Engineering & Remote Sensing, 72 (2006), pp. 551–563. https://doi.org/10.14358/PERS.72.5.551.

[37] I. Rey Otero and M. Delbracio, *Anatomy of the SIFT method*, Image Processing On Line, 4 (2014), pp. 370–396. https://doi.org/10.5201/ipol.2014.82.

[38] D.E. Schinstock, C. Lewis, and C. Buckley, *An alternative cost function to bundle adjustment used for aerial photography from UAVs*, in ASPRS Annual Conference, 2009, pp. 9–13.

[39] J.L. Schonberger and J-M. Frahm, *Structure-from-motion revisited*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104–4113. https://doi.org/10.1109/CVPR.2016.445.

[40] A. Sehgal, D. Cernea, and M. Makaveeva, *Real-time scale invariant 3D range point cloud registration*, in International Conference Image Analysis and Recognition, Springer, 2010, pp. 220–229. https://doi.org/10.1007/978-3-642-13772-3_23.

[41] D.E. Shean, O. Alexandrov, Z.M. Moratto, B.E. Smith, I.R. Joughin, C. Porter, and P. Morin, *An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very-high-resolution commercial stereo satellite imagery*, ISPRS Journal of Photogrammetry and Remote Sensing, 116 (2016), pp. 101–117. https://doi.org/10.1016/j.isprsjprs.2016.03.012.

[42] X. Tong, S. Liu, and Q. Weng, *Bias-corrected rational polynomial coefficients for high accuracy geo-positioning of QuickBird stereo imagery*, ISPRS Journal of Photogrammetry and Remote Sensing, 65 (2010), pp. 218–226. https://doi.org/10.1016/j.isprsjprs.2009.12.004.

[43] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon, *Bundle adjustment — A modern synthesis*, in International Workshop on Vision Algorithms, Springer, 1999, pp. 298–372. https://doi.org/10.1007/3-540-44480-7_21.

[44] S. Verykokou and C. Ioannidis, *A photogrammetry-based structure from motion algorithm using robust iterative bundle adjustment techniques*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 4-4/W6 (2018), pp. 73–80. https://doi.org/10.5194/isprs-annals-IV-4-W6-73-2018.

[45] J. Wohlfeil, H. Hirschmüller, B. Piltz, A. Börner, and M. Suppa, *Fully automated generation of accurate digital surface models with sub-meter resolution from satellite imagery*, ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 39-B3 (2012), pp. 75–80. https://doi.org/10.5194/isprsarchives-XXXIX-B3-75-2012.

[46] Z. Xiong and Y. Zhang, *A generic method for RPC refinement using ground control information*, Photogrammetric Engineering & Remote Sensing, 75 (2009), pp. 1083–1092. https://doi.org/10.14358/PERS.75.9.1083.

[47] C. Zach, *Robust bundle adjustment revisited*, in European Conference on Computer Vision (ECCV), Springer, 2014, pp. 772–787. https://doi.org/10.1007/978-3-319-10602-1_50.