An Efficient Algorithm to Compute the SRP Force and Torque on Spacecraft

By Juan Manuel GARCÍA,[1),2)]

[1)] *European Space Operations Center, ESA, Darmstadt, Germany*
[2)] *Flight Dynamics and Operations, GMV-Insyen AG, Darmstadt, Germany*

This paper presents an efficient algorithm to compute the force and torque exerted by the Solar Radiation Pressure on a generic spacecraft. Operational methods used so far either oversimplify the problem, thus computing only a rough approximation of the total effect, or disregard the geometrical structure of the spacecraft yielding slow performances. The current approach aims at taking advantage of all the geometrical information available from the beginning in order to reduce the number of computations to be performed. The problem is provided with geometrical structure by means of Binary Space Partitioning based on the mechanical units of the spacecraft. This structure is fully generic and allows obtaining a visibility ordering of the scene for any possible configuration at computational cost linear in the number of units. The shadows are then computed by means of a polygon Boolean operations algorithm in order to maximize performance. The obtained algorithm requires a much smaller processing time than the existing operational software used for previous missions like Venus Express or Rosetta.

Key Words: Solar Radiation Pressure, Binary Space Partitioning, Visibility Problem

Nomenclature

S/C	:	Spacecraft
BSP	:	Binary Space Partitioning
SRP	:	Solar Radiation Pressure

1. Introduction

The proper computation of the force and torque exerted by the SRP on a spacecraft is important for properly planning the spacecraft operations. On the one hand, the estimation of the acceleration produced by the SRP is important for the accurate navigation of the spacecraft. On the other hand, the torque exerted by the SRP is one of the main drivers for the angular momentum management strategy in missions with reaction wheels.

The computation of the SRP force and torque on the spacecraft depends on its surface optical properties and its geometrical configuration with respect to the Sun direction. An accurate estimation of the aforementioned quantities requires an evaluation of this relative geometry with a frequency driven by the spacecraft attitude rate and the rate of its articulations. Even for the case of idealized attitude profiles, this still requires the usage of a reasonably small time step. The situation is even worse in attitude simulators where the close-loop control of the attitude dynamics is simulated. Thus, the selection of a computation-time-efficient algorithm is of great importance to reduce the impact on the time required for the flight dynamics ground operations.

In the past, work on this sort of operational software has been performed following two different approaches. The first consists in using oversimplified S/C geometrical models allowing for a straightforward computation of the main SRP effect (e.g., flat plate model, cannonball model). This approach, when calibrated properly, can provide meaningful results in the force estimation, but is in general not enough to obtain a proper estimation of the SRP torque. The second approaches the problem in a numerical way by discretizing the scene, without taking into account any prior knowledge about the S/C geometry.[3)] The purpose of this paper is to present the possibility of introducing a technique from the field of computational geometry, Binary Space Partitioning trees, that allows providing global geometrical structure to the problem resulting in a more efficient algorithm.

The paper is structured as follows. In Section 2 a description of the geometrical modelling of a S/C is provided. Section 3 presents a high level formulation of the problem. Section 4 describes the two different methods that have been analysed and qualitatively compared. In particular, the newly developed method is described in detail. Section 5 performs a comparison of the presented methods. Finally, Section 6 brings conclusions and identifies future steps.

2. Geometrical representation of a spacecraft

The geometrical representation of the S/C is structured into different units which represent the different spacecraft mechanical elements (e.g. central body, solar arrays, ejectable elements, etc.). Each of these units is modeled by a polygonal mesh (or polymesh), [1)] i.e., a finite collection of vertices, edges and faces that fulfill the following three conditions (see Ref. 1)):

- No isolated vertices exist (every vertex belongs to, at least, one edge)
- No isolated edges exist (every edge belongs to, at least, one face)
- No interpenetration of faces is allowed (an edge of one face cannot live in the interior of another face)

This representation is advantageous because it can be easily stored and parsed from ASCII configuration files and it provides an easy intuition of the shape of the S/C at first glance. Its main limitation is that it only allows the existence of planar polygonal surfaces. The restriction to planar elements is however a very useful simplification when dealing with visibility problems, as will be explained below, so it is not considered an issue. Even though non-planar elements are common (e.g., antenna dishes), they can be sufficiently approximated by a set of planar shapes for the precision required here. The same applies to non-polygonal planar shapes (e.g. circles), where the original shape can be approximated using a sufficient number of edges.

Each face is represented by a simple polygon (i.e., one which is topologically equivalent to a circle) and is oriented counterclockwise as seen from the semispace defined by its positive normal, thus defining its interior. Given a scene and a Sun direction, a face is said to be active when it is facing the Sun, i.e., when

$$\vec{s} \cdot \vec{n} > 0 \qquad (1)$$

$\vec{s}$ being the Sun direction as seen from the spacecraft (i.e., the opposite to the SRP flow direction).

3. Problem statement

Assuming multiple scattering can be neglected, the core of the problem of computing the SRP effects on a S/C is determining which parts of the S/C are subject to the SRP flow. Only the parts facing the flow (Sun) and not shadowed by other S/C parts will contribute to the total SRP effect. This problem, which will be here referred to as the *Visibility Problem,* can be stated as follows:

Given a set of polygons in 3-D space, determine which parts of the set are visible from a given direction in space.

The problem is equivalent to the usually called *Hidden Surface Removal* problem of the field of computer graphics that deals with the determination of the visible parts of a scene for the sake of graphics rendering.[2)]

The previous is a narrowed down definition of the general visibility problem in two ways: only planar surfaces are allowed (polygons) and the viewpoint is considered to be represented by a single direction (unit vector), i.e., the viewer is supposed to be infinitely far from the scene and, thus, only the scene orientation and not its position is relevant to the problem. The internal faces of the spacecraft can be completely ignored.

The solution to the visibility problem can be applied to other than the SRP effects computation. In particular, any problem requiring shadow computation will be facing the same issue, and thus can be solved by the same means. This is the case, for example, in the determination of drag forces and torques in the free-molecular approximation. The same methods can also be applied for 3-D visualization of the spacecraft geometry from a given viewpoint.

The result of solving the visibility problem is a set of face fragments extracted from the original set of 3-D polygons. Once a certain fragment of a given active face (i.e. $cos\theta := \vec{s} \cdot \vec{n} > 0$) has been determined to be subject to the flow, its effect on the spacecraft can be computed as follows (see Ref. 6)):

$$\vec{F}_{el} = -p_{SRP} A_{el} cos\theta \left((1 - C_s)\vec{s} + 2\left(C_s cos\theta + {}^1/_3\, C_d\right)\vec{n} \right) \qquad (2)$$

where p_{SRP} is the value of the Solar Radiation Pressure, which is modelled as an inverse quadratic function of the distance to the Sun, A_{el} is the area of the face element, $\vec{s}$ is the S/C to Sun direction unit vector, $\vec{n}$ is the face element normal unit vector and C_s and C_d are coefficients that depend on the optical properties of the material. The force is applied in the centroid of the face element, which can be easily computed in-plane for any simple polygon as follows:

$$\vec{c}_{el} = \frac{\sum_{edges} (\vec{v}_{start} + \vec{v}_{end}) \left((\vec{v}_{start} \times \vec{v}_{end}) \cdot \vec{n} \right)}{6 A_{el}} \qquad (3)$$

The final force and torque exerted on the origin of the spacecraft mechanical frame can be computed by simply adding the contributions from all the visible elements:

$$\vec{F}_{total} = \sum_{el} \vec{F}_{el} \qquad (4)$$

$$\vec{T}_{total} = \sum_{el} \vec{c}_{el} \times \vec{F}_{el} \qquad (5)$$

The previous is applicable to the computation of the drag effects on the S/C by simply replacing the force model (see Ref. 7)):

$$\vec{F}_{el} = -p_{dyn} A_{el} C_D cos\theta \, (C_{acc}\vec{v} + 2cos\theta(1 - C_{acc})\vec{n}) \qquad (6)$$

where p_{dyn} is now the value of the drag dynamic pressure, $\vec{v}$ is spacecraft velocity direction unit vector, and C_D and C_{acc} are the drag and accommodation coefficients.

Thus the complexity of the problem lies in determining the solution to the *Visibility Problem,* i.e., the vertices of all the visible elements that contribute to the final solution. This is what will be addressed in the rest of the paper.

4. Review of methods

This section begins with a description of the operational method that has been used at ESOC for all interplanetary SC. For calculation of Rosetta drag forces and torques due to the comet coma, a numerically efficient Rosetta-tailored algorithm was developed, which was found to be a special case of the Binary Space Partitioning technique. Because of the operational advantages gained by that algorithm in the scope of Rosetta, a generic algorithm based on the technique of Binary Space Partitioning was developed. This technique is presented in the second part of this section and a discussion on its applicability to the current problem is performed together with a description of the newly developed algorithm.

4.1. Ray tracing: the current method

The idea behind ray tracing is to use the intersection points of rays shot from a projection plane perpendicular to the Sun direction and lying behind the scene to determine the visibility ordering of the active faces. A projection plane π perpendicular to the flow direction is introduced behind the S/C and an orthogonal equidistant adaptive grid is constructed in π, with a given initial cell size. All the active faces of the problem are then projected on π, leaving each grid cell in one of three possible states:

1. The cell is fully covered by one of the projected faces.
2. The cell doesn't contain any part of any projected face.
3. The cell partly overlaps with one or several projected faces.
   - 3.1. Two visible intersections from the same edge.
   - 3.2. Other configuration.

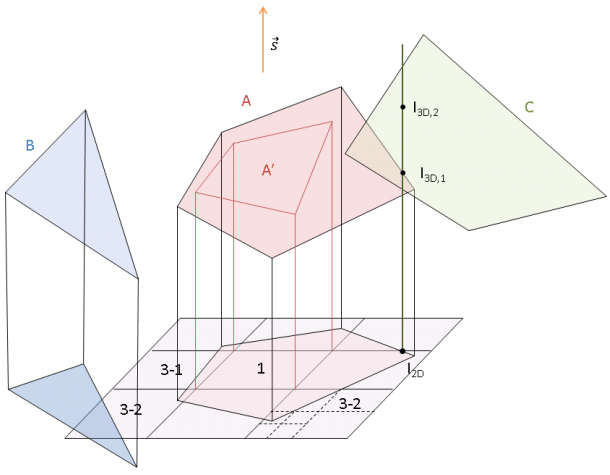

Fig. 1. Ray tracing algorithm.

Figure 1 shows a geometrical configuration that illustrates all the possible situations. When the cell is in state 1 or 2, the visibility ordering for that particular prismatic section of the problem can be solved directly as follows. A ray is shot from the centre of the cell towards the viewpoint. If this ray doesn't intersect any active face, then the cell is in state 2 and provides no contribution. If intersections do exist, then the intersection points are computed and sorted by distance to π. The point with highest distance determines the face which is visible in that particular section of the scene. The fragment of the face enclosed by the prismatic section determined by the cell is visible and its contribution can be added to the total result. Such is the case of the cell marked as *1* in Fig. 1, where the contribution of the 3D polygon *A'* can be added to the result straightaway.

If the cell is in state 3, the intersection points of the projected edges with the cell's borders are computed. From each intersection point, a ray is shot in the direction of the viewpoint and the intersections of the ray with all the active faces are computed and ordered by distance to the projection plane. If the farthest intersection point is not on the original edge, then the cell intersection point is being hidden by another face and can be discarded. This is the case of the point I_{2D} in Fig. 1, which belongs to an edge in face *A,* but is being hidden by face *C*. Once all the visible cell intersection points have been determined, two situations can arise: either the cell is being intersected twice by the same projected edge or the situation is more complex. In the first case (cell in state 3.1), the computation is equivalent to the one in state 1, but the ray is in this case shot from the barycentre of the covered part of the cell instead of the cell centre. The fragment of the face enclosed by the prismatic section determined by the cell is visible and its contribution can be added to the total result. This is the case for the element *3-1* in Fig. 1.

If, however, the number of intersections is bigger (cell in state 3.2), the cell is subdivided evenly into four smaller subcells for which the process is repeated. A maximum depth level is specified to prevent this refinement from going on endlessly. If the final depth level is reached and the cell is still in state 3.2, the cell is considered to be in state 1 and its contribution is added to the final result. An example of this subdivision process is shown in Fig. 1 by the cells marked as *3-2*. This illustrates one of the weaknesses of the algorithm: even for a simple layout in which the projection of a vertex is in the middle of a grid-cell, at least three extra subdivisions are needed in order to resolve the situation.

Another weakness of this algorithm is the great number of operations it requires. For each cell (or subcell), a ray needs to be shot and its intersection with all the existing planes containing active faces computed and checked against the interior of the polygons. In addition, a sorting algorithm needs to be invoked for every ray in order to sort the intersection points by their distance to the projection plane.

4.2. Introducing BSP trees

The previous method is conceptually very simple and, by dealing with each cell independently, it can cope with complex geometrical configurations since no assumptions are made about the geometrical arrangement of the 3D scene. However, the geometrical structure of the problem is totally ignored and therefore the method doesn't take advantage of a

big amount of information that can be used to simplify the process and improve the performance.

The following assumption is the key point that allows introducing some a priori geometrical knowledge to simplify the problem structure: *Given any two faces A and B, for any given scene configuration, either A shadows B or B shadows A (or both are fully visible)*. The previous follows from the fact that no interpenetrating faces are allowed, which implies that a plane π exists that separates both faces. With this assumption in mind, given two planar surfaces A and B (with faces on both sides), plane π can be used to divide the space in two different regions as shown in Fig. 2. When the viewer is in region I, A is said to be in front of B, and it is possible for A to shadow B but not the other way around. When the viewer is in region II, the opposite situation occurs. This split of the space allows to very simply check, for a given viewpoint direction, which of the faces is in front, by using a simple scalar product. The final information can be encoded into a binary tree, in which a simple computation at the root node based on the viewpoint direction provides the ordering of the faces.

The previous is the essential idea behind Binary Space Partitioning. When more surfaces are added to the problem, the partitioning can continue recursively in each of the subspaces defined by the previous level until each of the regions contains only one surface. All this information can be encoded in a binary tree, so called BSP tree, of which the height will depend on the number of divisions performed. As long as the configuration of the scene does not change, this tree is invariant and can be used to retrieve a face ordering with a simple traversal based on a sequence of scalar products.

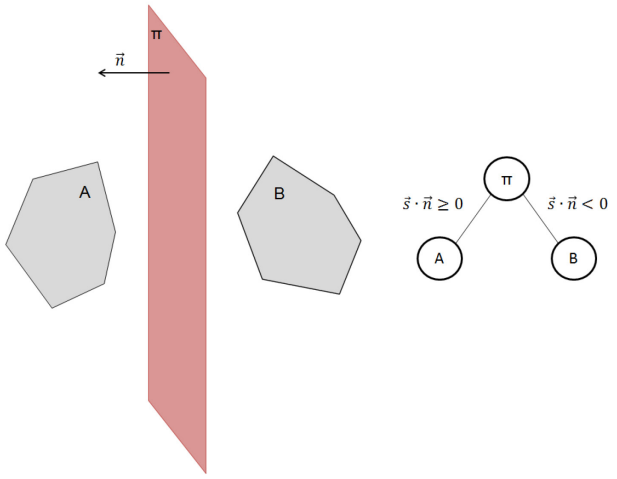

Fig. 2. Simple space partitioning with two faces.

4.3. Unit based BSP trees

The application of Binary Space Partitioning to the current problem can be approached in two different ways. The first is to use a face-based Binary Space Partitioning: the algorithm could recursively divide the scene using the planes defined by the different active polygons. This implies computing the intersection of each plane containing an active face with the rest of the active faces in order to obtain a consistent binary partition. The final result is a full ordering of all the obtained active face fragments. This is the approach usually followed in computer graphics for the rendering of static scenes by applying the *Painter's Algorithm*, since a one-time computation of the tree allows for a full representation of the scene as long as only the viewer's position changes.[2] However, in the context of S/C problems, the scenes are usually not static due to the movement of the articulated units.

The previous approach is however not taking advantage of the geometrical structure of the S/C. An extra fact can be considered in order to simplify the problem further: *each of the spacecraft units is either a convex polyhedron or can be easily split into convex polyhedrons*. The convexity of a unit ensures that faces belonging to the same unit cannot cast shadows on one another, since only the external faces of each unit need to be considered. Thus, a visibility ordering of the units is enough to obtain a full visibility ordering of the scene, since active faces belonging to the same unit can be considered to lie on the same visibility layer.

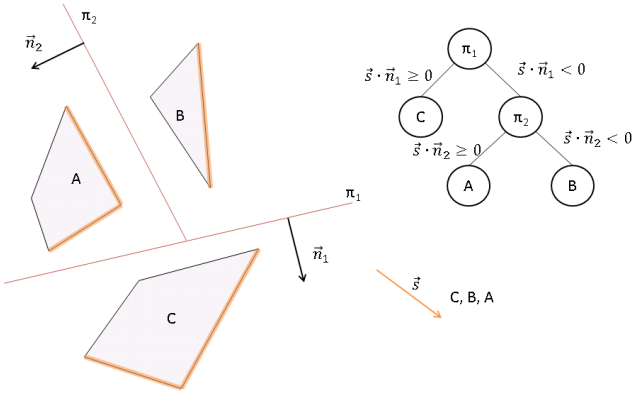

Fig. 3. Unit based BSP tree.

Figure 3 shows an example of unit based BSP tree in 2-D space (where units are represented by convex polygons and planes by lines). A visibility ordering can be generated by dividing the plane in different regions using specific lines. Lines π_1 and π_2 are used in this case, yielding the shown binary tree. For a Sun direction like the one shown, the obtained visibility ordering is C, B, A, which indicates that C is in front of the scene, not shadowed by anything, B can potentially be shadowed by C, and A can be shadowed by both C and B. Notice that, once the unit visibility ordering is determined, only the active faces (highlighted in orange) of each unit are considered in the following steps. The number of shadowing computations has been reduced to computing the shadows of the active faces of C on the active faces of A and B and the shadows of the active faces of B on the active faces of A.

Figure 3 also shows the representation of a unit based BSP tree: the internal nodes of the tree store their orientation (the positive normal of the division plane, where the criterion to define the positive semispace is arbitrary), whereas the leaves store pointers to the geometrical units lying in the corresponding subspace.

From the previous example, the conditions the scene must fulfill so that the desired unit based BSP tree exists can be inferred:

1. A plane must exist separating each pair of objects. This is ensured if all the bodies are convex. If a spacecraft unit is not convex it can always be split into convex parts.
2. The binary recursive division must be possible, i.e., a plane must exist dividing the whole scene in two regions, then a plane must exist dividing each subspace in two, and so on. These planes will necessarily be part of the set of planes that fulfill the first condition. However, the fulfillment of the condition 1 doesn't guarantee the fulfillment of condition 2. Figure 4 shows an example of such a case, where all possible two-body separation planes intersect at least one of the other bodies. This problem can be overcome by dividing C into two separate parts: C' and C''.
3. The bodies must be steady, i.e., if a body moves, the tree could potentially become invalid.

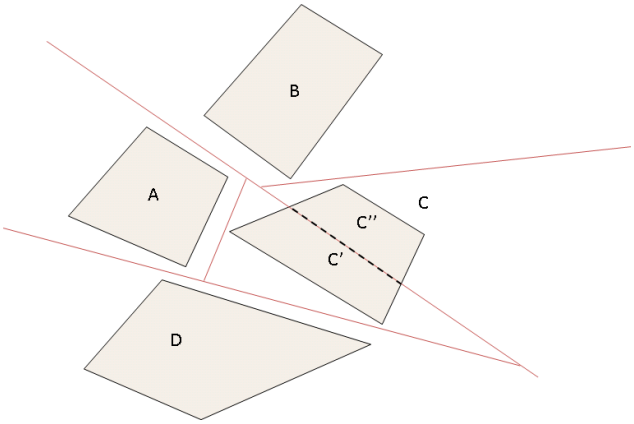

Fig. 4. Situation in which a unit split is required.

Conditions 1 and 2 are usually fulfilled by most spacecraft geometries, since privileged directions exist. In any case, a preprocessing of the unit partitioning is enough to ensure the compatibility with these two conditions if complex spacecraft are being analyzed (e.g. Bepi-Colombo).

The third condition however requires a special treatment. In many cases, the spacecraft have articulated units that modify the scene configuration while moving and can, in the worst case, invalidate a given BSP tree. This will happen only if the articulation can reach a position in which it intersects any of the current planes defining its subspace. To overcome this difficulty, it is enough to replace the initially unique BSP tree with a discrete set of BSP trees each of them valid for a specific range of articulation angles. When an articulated unit reaches a position where the current BSP tree is no longer valid, a different BSP tree consistent with the new scene is used. Usually, a valid BSP tree will exist for big intervals of the articulated unit positions. The problem can be minimized if the articulated units are pushed as down as possible in the tree, i.e., their separation from the rest of the scene is kept for the end of the process. Thus, the number of changing elements is usually limited to the close neighbors of the articulated unit.

The full geometrical structure of the problem can thus be provided by means of a set of BSP trees valid for different ranges of articulation angle combinations. These trees are static and can be encoded in a configuration file, thus requiring some configuration effort for each different spacecraft but reducing significantly the computational cost of the rest of the process. The following examples show the implementation of this technique for some ESA interplanetary missions. Figure 5 shows an example of unit based partitioning for Mars Express. In that case, even though the solar arrays are articulated, their motion can never lead them to intersect the separation planes, so a single BSP tree is enough to encode the geometrical structure of the spacecraft.

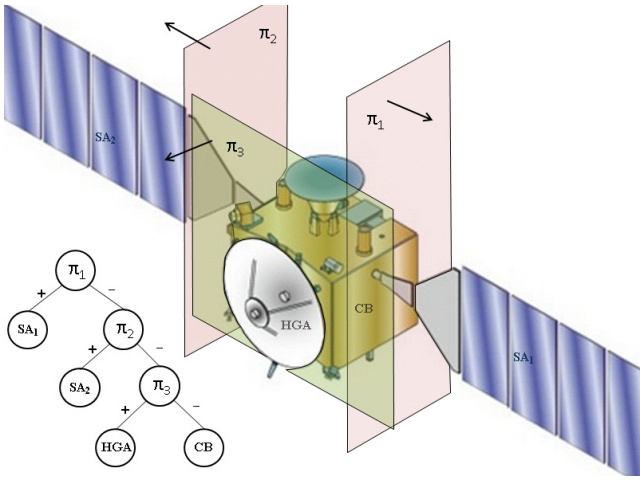

Fig. 5. Mars Express partitioning.

Figure 6 shows the same for the ExoMars TGO mission, with two different positions of the High Gain Antenna dish. In this case, the movement of the HGA requires the definition of two different BSP trees. The first one uses plane π_2 to separate the antenna dish from the central body when the elevation angle is between -24° and 185° degrees. Beyond 185° plane π_3 is used instead.

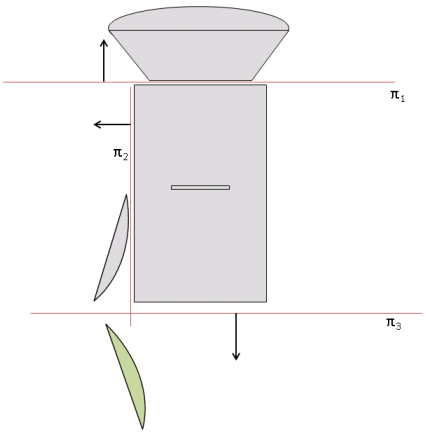

Fig. 6. ExoMars plane separation.

Finally, Fig. 7 shows the geometrical model of the Bepi-Colombo spacecraft, which violates conditions 1 and 2 and requires some unit splitting in order to define the corresponding BSP tree, which, however, can be unique in this case.

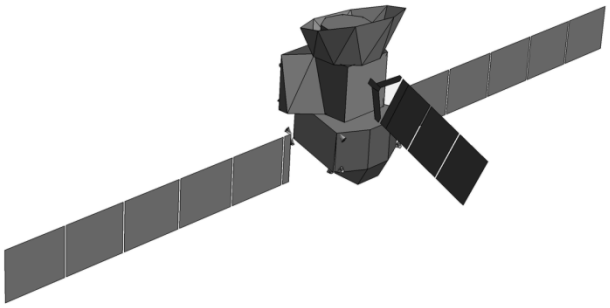

Fig. 7. Bepi-Colombo geometrical model.

4.4. Adding polygon Boolean operations

Once a visibility ordering is available as provided by the BSP tree, the scene structure turns into a set of layers of active faces, that can be represented by an array of sets of faces, the first element representing the layer that lies in front of the scene and the last element representing the layer that lies in the back. Figure 8 shows how such an array is obtained for a particular flight configuration of the ExoMars TGO.

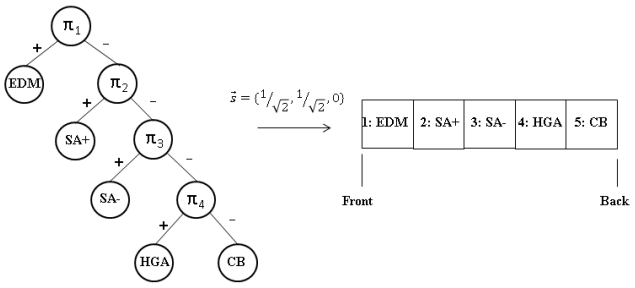

Fig. 8. Visibility layers.

The remaining part of the process requires determining which parts of the layers in the back are being shadowed by the layers in front. This can be computed very efficiently without discretizing the problem by performing polygon Boolean operations. The following is an outline of the algorithm:

1. Project all the faces in the layers onto a 2D plane perpendicular to the flow direction. All the following operations are performed on that projection plane.
2. For all the layers but the one in the back, compute the union of all the faces contained in the layer. This generates a shadowing layer for each of the visibility levels in the problem. Notice that, because of the way they are constructed, shadowing layers can consist in general of more than one polygon.
3. For each visibility layer but the one in the back, perform the union of the corresponding shadowing layer with all the shadowing layers in front of it. This modifies the shadowing layers computed in the previous step.
4. For each face in each visibility layer, starting from the back layer, compute the Boolean difference between the face and the shadowing layer in the level directly in front of it. The resulting polygons are visible.
5. Project the resulting visible polygons back to their original 3D planes.
6. Compute the SRP force and torque as described in 3.

Polygon Boolean operations are performed by means of a modified version of the Greiner-Hormann algorithm that can cope with any kind of polygon.[4,5] This is a well-known sweep-line algorithm for the computation of the overlay of planar subdivisions.[2] The previous process takes advantage of the capability of the algorithm to perform any type of Boolean operation (and not only polygon intersection), in order to minimize the number of operations to be performed by computing and merging the shadowing layers affecting each of the visibility levels of the scene. This is possible because the algorithm can deal with sets of simple polygons as operands (some of which can represent holes in a parent polygon), since they are all considered part of a single planar subdivision.

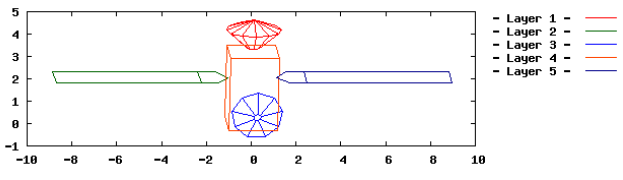

Fig. 9. Visibility layers before the processing.

Figure 9 shows the layered arrangement of the active faces of the ExoMars TGO for a given Sun direction. This arrangement is then processed as described above producing

the result shown in Fig. 10, where only the visible face elements are now present. The processed layers are then projected back to their original 3D planes and their contributions are added to the final result.

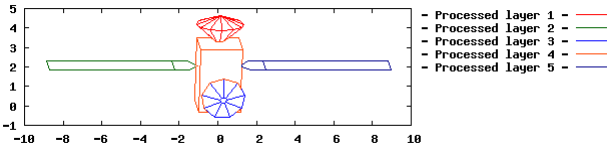

Fig. 10. Visibility layers after the processing.

Figure 11 shows an example of a merged shadowing layer for the previous scene, corresponding to the shadowing layer number 4, generated by the union of all faces in layers 1, 2, 3 and 4, which is then used to shadow the elements in layer 5.

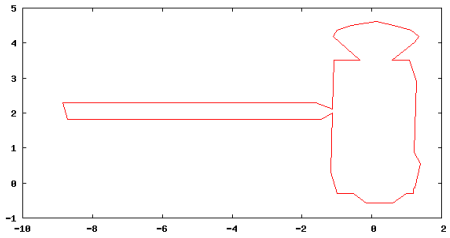

Fig. 11. Merged shadowing layer.

Figure 12 shows an example of a clipping computation that takes place during the scene processing. In this case, the shadow of shadowing layer 3 on the front panel of the central body is being computed. The final visible parts are shown in Fig. 13.

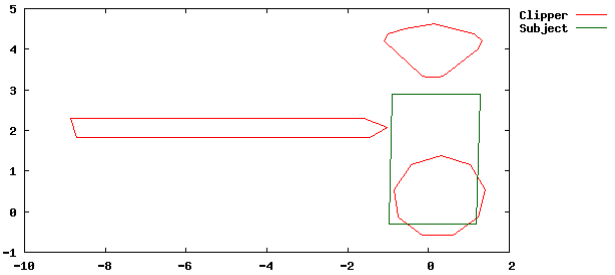

Fig. 12. Difference computation.

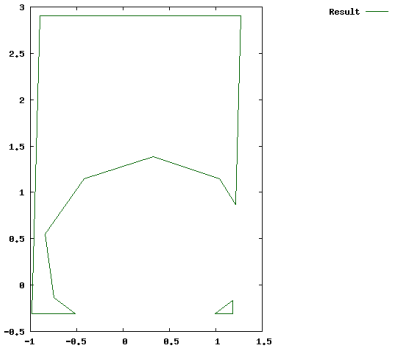

Fig. 13. Shadowing computation result.

5. Method comparison

In this section both methods described above are compared in terms of different relevant characteristics. For the sake of clarity, the algorithm based on ray tracing will be referred to as *algorithm 1* and the new algorithm based on BSP trees as *algorithm 2*.

The performance of both methods has been compared by performing a one month reaction wheel propagation for a slow varying attitude and articulation profile with a time step of 5 minutes, which corresponds to the computation of 8642 steps. The main driver of the execution time of such a propagation is the computation of the SRP torque at each propagation step. The test has been performed for both the ExoMars TGO and the Bepi-Colombo spacecraft and Table 1 shows the obtained results. It can be seen that the time improvement obtained from using algorithm 2 is in the order of a factor of 20, which results in a relevant operational advantage.

Table 1. Performance comparison.

Mission	Algorithm 1	Algorithm 2
EXM TGO with EDM	450 s	20 s
EXM TGO without EDM	200 s	10 s
Bepi-Colombo	883 s	53 s

Both methods require a significant amount of configuration effort in order to define the S/C geometrical model. The setup is usually based on models obtained from the industry manufacturer. The increase in performance of algorithm 2 is mostly accomplished thanks to the extra information available to the algorithm encoded in the BSP trees. Since this information needs to be provided by the user, the new algorithm requires some additional configuration effort compared to the existing one, in order to ensure the model fulfills the conditions described in 4.3. This effort is proportional to the geometrical complexity of the spacecraft and the existing number of articulations. Notice, however, that this is a one-time effort for each spacecraft that can be performed in the preparation phase and will provide a drastic performance improvement over the operational life of the mission.

In terms of accuracy and robustness, it has been shown that algorithm 1 can easily need a high amount of cell subdivisions even in simple geometrical configurations. The consequence of such situations is, not only a decrease in performance, but also the insertion of inaccuracies in the result that depend on the maximum depth of subdivisions allowed. The minimum cell size is typically configured to have a side length in the order of 1 cm, which means errors introduced by face fragments of $1cm^2$ can be systematically inserted by the algorithm. It can thus be concluded that algorithm 2 is in general more accurate, even though the differences will hardly be noticeable. A comparison of the results obtained with both methods for 700 different scene configurations of the ExoMars TGO (with EDM attached) is shown in Figs. 14 and 15 in terms of magnitude and angular errors of the obtained force and torque vectors. It can be seen that the difference between the results is below the typical precision of the model compared to reality, which is estimated to be around 10% due to the uncertainty on the knowledge of the S/C optical properties.

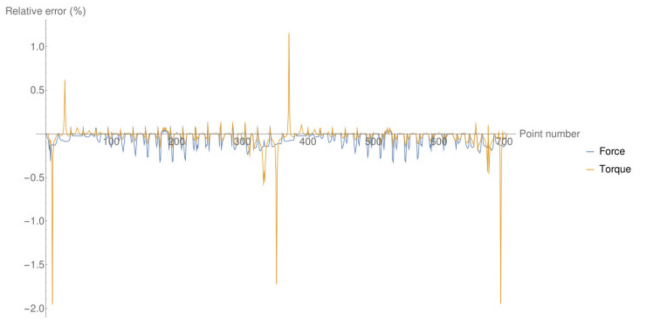

Fig. 14. Relative magnitude error.

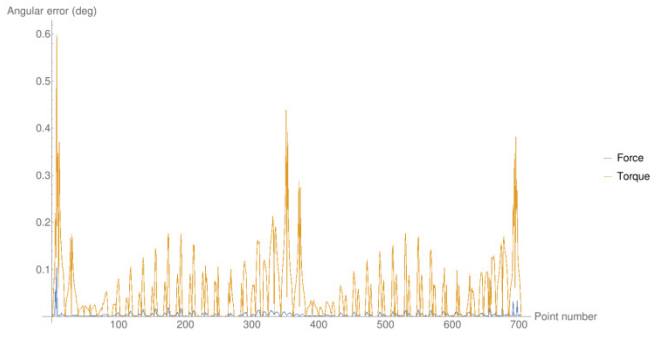

Fig. 15. Angular error.

In terms of complexity, algorithm 1 could be deemed to be simpler in the sense that it relies on a smaller number of assumptions. Thus, a general understanding of the method can be obtained in little time and the implementation of the method requires less effort. Algorithm 2 on the other hand is much more specialized and requires both higher analytical and implementation efforts. This again shows that, in a sense, algorithm 2 draws its power from shifting some of the effort from the operational phase of the mission to the development one.

6. Conclusion and future steps

It has been shown that the new algorithm based on Binary Space Partitioning trees and polygon Boolean operations increases greatly the performance of the SRP related computations, thus significantly reducing the time required for the flight dynamics ground operations. The algorithm is already operationally in place for the ExoMars TGO and is being used in the preparation of the Bepi-Colombo mission.

In the process, some future steps have been identified and will be explored in order to contribute further to a fully generic and multi-purpose method:

- Further exploration of the applicability of BSP trees to the problem, in particular analyzing the automation of the BSP tree generation process.
- Extension of the number of applications that make use of the algorithm, in particular making it easily available for analysis requiring visibility computations.
- Comparison of the performance of the shadowing computation approach (based on polygon Boolean operations) with a simpler one, to properly understand how much of the performance gain comes from each of the parts of the process.
- For performing numerical integrations robustly, the integration may need to be split at times when derivatives of the integrand are dis-continuous. Derivatives of SRP forces and torques can only become dis-continuous at times when the scene structure changes. The BSP method could be extended in order to identify these times.

References

1) Schneider, Philip J., Eberly, David H.: *Geometric Tools for Computer Graphics*, Morgan Kaufmann Publishers, San Francisco, 2003.
2) de Berg, Mark, Cheong, Otfried, van Kreveld, Marc and Overmars Mark: *Computational Geometry, Algorithms and Applications*, Springer, 2008.
3) Fainberg, J., Herfort, U.: *Rosetta Solar Radiation Pressure Force and Torque (RO-ESC-TN-5529),* ESOC, 2001.
4) Martinez, Francisco, Rueda, Antonio J., Feito, Francisco R.: *A new algorithm for computing Boolean operations on polygons,* Computer & Geosciences, Volume 35, Issue 6, June 2009, pages 1177-1185.
5) Martinez, Francisco, Ogayar Carlos, Jiménez, Juan R., Rueda, Antonio J.: *A simple algorithm for Boolean operations on polygons,* Advances in Engineering Software, Volume 64, October 2013, pages 11-19.
6) Wertz, James R.: *Spacecraft Attitude Determination and Control,* D. Reidel Publishing Company, 1986.
7) Budnik F., Damiani S., Flegel M., Lauer M., Mueller M.: *VEX Atmospheric Drag Measurements.*