Proceedings of the 9th International Conference of Control Systems, and Robotics (CDSR'22)
Niagara Falls, Canada – June 02-04, 2022
Paper No. 123
DOI: 10.11159/cdsr22.123

Continuous Appropriately-Oriented Collision Detection Algorithm for Physics-Based Simulations

Alexander Schock, Robert Langlois[1],
[1]Carleton University
1125 Colonel by Drive, Ottawa, Canada
AlexRSchock@cmail.carleton.ca; Robert.Langlois@carleton.ca

Abstract - The presented algorithm enables the continuous detection of the appropriate collision surface for a point penetrating into a convex bounding area in two-dimensional space. The appropriate collision surface is resolved through the transient evaluation of the separating axis theorem which tracks the last bounding area surface to lose separation from the point, thus collision. This ensures faithful application of collision restitution forces. The algorithm is presented in two forms: looping conditional statements, and binary matrix operations. The initial implementation of the algorithm is also presented and discussed.

***Keywords*:** Continuous collision detection, dynamic collision detection, separating axis theorem, physics-based simulations.

1. Introduction

Collision detection algorithms which detect the interference between two or more objects in both static and dynamic environments are integral components of physics-based simulations. Further, these algorithms subsequently enable the application of collision responses. As simulations are increasingly becoming the main choice for modelling and design evaluation, emphasis is placed on the real-world fidelity that these simulations can provide. The pursuit of real-world fidelity consequently requires increasingly-accurate models, and collision detection is no exception.

Broadly speaking, collision detection algorithms are separated into two phases of detection: 'broad-phase' detection to detect potential collision pairs at the object level, and 'narrow-phase' detection which directly evaluates collision at the feature level [1]. Narrow-phase collision detection can be further classified into two types: spatial partitioning representations (SPR) and bounding volume hierarchies (BVH) [2]. For physics-based forward dynamics simulations, BVHs are advantageous as the bounding volumes can be used to represent the bodies in the simulated environment.

As shown in Figure 1, bounding volume hierarchies have been the subject of typical progressive generalization. Early research proposed coordinate system axis-aligned bounding boxes (AABBs). Axis-alignment was relaxed with oriented bounding boxes (OBBs) [3]. Further refinement followed in the form of discrete orientation polytopes (k-DOPs) [4], or fixed-direction hulls (FDH) [5] through the intersection of intervals associated with a set of object-fixed axes. The furthest generalization of the bounding volume is the convex hull. With convex hulls, the interval characteristic is relaxed such that a tight bounding volume is achieved through a set of arbitrary surfaces forming a convex hull.

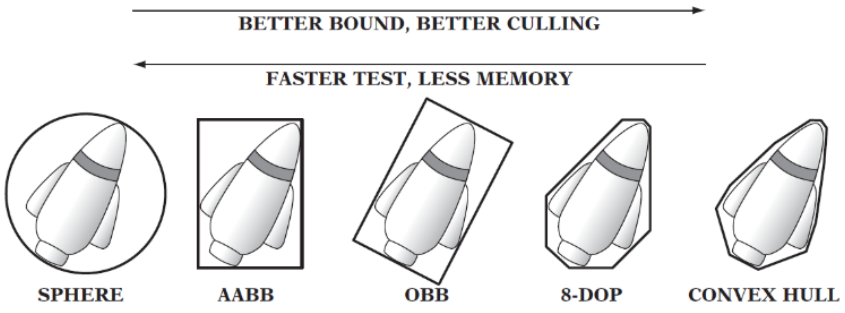

Figure 1: Types of bounding volume [4].

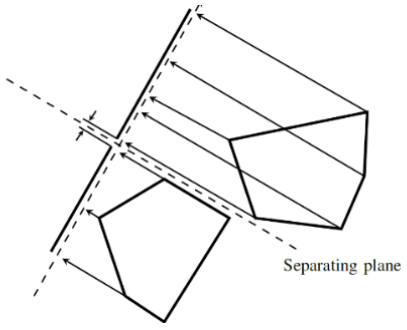

Figure 2: Separating axis for two disjoint polygons.

At the narrow phase, the collision pair must be evaluated for overlap. Typically, this can be done using the separating axis theorem (SAT) [3], [6]. The SAT is effectively the rejection of overlap (collision) if a separating axis (plane) between the collision pair can be obtained as shown in **Error! Reference source not found.**.

Where overlap is detected, collision models then apply the necessary forces to simulate a collision response. Typically, the minimum translation vector (MTV) strategy is used to direct the collision restitution forces in the direction of the smallest distance to remove collision. This is taken as the shortest distance between a reference point on one colliding body relative to the surface of the other body. However, the MTV is indiscriminate in the direction of collision force application. The direction can change unexpectedly near corners or thin objects. This can significantly hinder the fidelity of the simulation.

In this paper, we present an alternate scheme to the MTV for directing the restitutive collision forces for physics-based simulations. We present a narrow-phase collision detection algorithm which continuously evaluates separation, thereby enabling the appropriate orientation of collision forces. Consequently, real-world fidelity of collision is better achieved.

To establish context and present the development of the algorithm, the rest of this paper is organized into the following sections. Section 2 establishes the simulation environment. Section 3 develops the algorithm in two-dimensional space. Then, the initial application of the algorithm is discussed in Section 4. Section 5 offers concluding remarks.

2. Collision Environment

In this paper, the simulation environment is a planar two-dimensional space having a Cartesian coordinate system. In the collision pair, we prescribe 'target' bodies represented by convex bounding volumes, and 'initiator' bodies represented by points in space.

Collision is unilaterally evaluated and applied on the initiators. This collision scheme stems from typical simulations which produce the response of a studied object as it interacts with its environment. The target bodies form the simulation environment and normally have motions known for the duration of the simulation. The initiators are the bodies for which motions are propagated by the system dynamics, including their interaction with the target bodies.

The presented algorithm evaluates the collision of initiators onto a single target to produce a response which affects the initiators. Subsequently, the algorithm can be independently applied to all targets in the system.

First, let the target be the planar convex bounding area with vertices v defined in a clockwise order as illustrated in Figure 3. The position of the *i*th vertex v_i is

$$\vec{\boldsymbol{r}}_{v_i}(t) = \begin{Bmatrix} r_x(t) \\ r_y(t) \end{Bmatrix}, \text{for } i = 1, \dots, N_v \tag{1}$$

where the target bounding area is defined by N_v number of vertices.

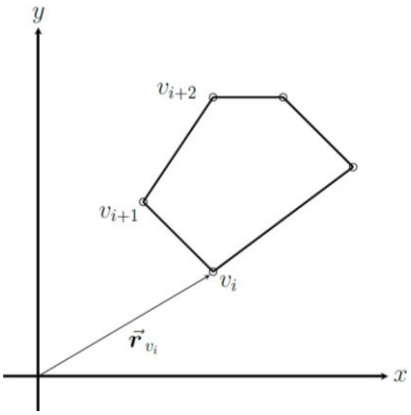

Figure 3: Propagated body position vector.

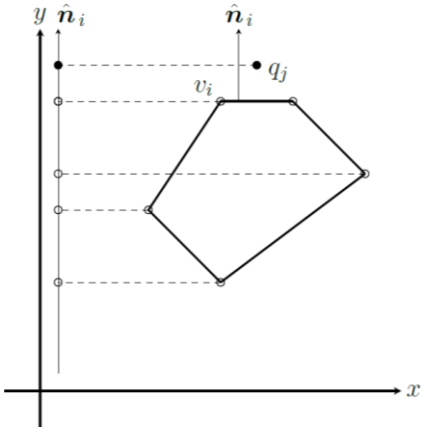

Figure 4: Target-initiator separation.

Next, let the initiator be a point in space q. The position of the *j*th initiator is

$$\vec{\boldsymbol{r}}_{q_j}(t) = \begin{Bmatrix} r_x(t) \\ r_y(t) \end{Bmatrix}, \qquad \text{for } i = 1, \dots, N_q \tag{2}$$

where the number of initiators in the simulated system is N_q.

In defining the initiators as points, we can represent any type of convex, concave, rigid, and even deformable body as a combination of initiators. Further, it pinpoints exactly where restitutive forces will be applied to these bodies, and greatly simplifies the evaluation of separation through the SAT, as will be discussed in Section 3.

3. Oriented Collision Configuration Algorithm

The continuous appropriately-oriented collision configuration algorithm is underpinned by three key notions. First, there is a distinction between *penetration* and *collision*. Penetration is the overlap between the target and initiator as determined by the SAT. The penetration configuration describes where separation exists between the target and initiator. On the other hand, collision implies the feature of the target physically interacting with the initiator. The collision configuration describes which surface is in contact with the initiator, thereby orienting the restitutive forces. Second, the SAT evaluates separation along axes normal to the surfaces to relate separation (otherwise penetration) to the surface features of the target. Third, the transient evolution of the penetration configuration enables the determination of the correct collision surface. At the time where the initiator penetrates the target, separation is lost along a surface which corresponds to the collision surface.

3.1 Separating Axis Theorem

Fundamentally, the algorithm is the continuous evaluation of the SAT. In theory, infinite separating axes exist where two objects are not exactly separated. A classic approach to circumvent this issue is to establish a finite set of potential separating axes which lie parallel to the surfaces of the convex bounding area polygon.

To evaluate the existence of a potential separating axis along a surface of the target, we must project the vertices of the target and the initiator to an axis orthogonal to the surface. Separation will exist where the projected initiator is disjoint from the projected target vertices. The axes used to evaluate separation are the target surface normal unit vectors $\hat{\boldsymbol{n}}$. Since the vertices of the target are defined in a clockwise order, we can first develop the set of unit vectors parallel to the target's surfaces as:

$$\hat{\boldsymbol{p}}_i = \frac{\left(\vec{\boldsymbol{r}}_{v_{i+1}} - \vec{\boldsymbol{r}}_{v_i}\right)}{\left\|\vec{\boldsymbol{r}}_{v_{i+1}} - \vec{\boldsymbol{r}}_{v_i}\right\|}, \qquad \text{for } i = 1, \dots, N_v \tag{3}$$

where $\hat{\boldsymbol{p}}_i$ denotes the *i*th unit vector parallel to the *i*th surface of the target, and *i* wraps around to one.

Rotating $\hat{\boldsymbol{p}}$ counter-clockwise by 90-degrees yields unit vectors orthogonal to the surfaces of the target:

$$\hat{\boldsymbol{n}}_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \hat{\boldsymbol{p}}_i, \qquad \text{for } i = 1, \dots, N_v \tag{4}$$

where the matrix is the counterclockwise rotation of $\hat{\boldsymbol{p}}_i$ to obtain the *i*th $\hat{\boldsymbol{n}}_i$.

To evaluate separation between the *j*th initiator and target along the *i*th surface normal, we project all vertices along with the *j*th initiator to the *i*th surface normal. Separation exists when the following equation is satisfied:

$$\max(\hat{\boldsymbol{n}}_i \cdot \vec{\boldsymbol{r}}_v) < \left(\hat{\boldsymbol{n}}_i \cdot \vec{\boldsymbol{r}}_{q_j}\right) \tag{5}$$

where '·' is the dot product. If Equation (5 is satisfied, the *j*th initiator's projection lies outside of the range formed by the target's vertices. Thus, separation exists as illustrated in Figure 4. Equation (5 can be greatly simplified to

$$0 < \hat{\boldsymbol{n}}_i \cdot \left(\vec{\boldsymbol{r}}_{q_j} - \vec{\boldsymbol{r}}_{v_i}\right) \tag{6}$$

because v_i is a maximal point on the projection of the target's vertices to $\hat{\boldsymbol{n}}_i$.Therefore, only the *i*th vertex must be projected when evaluating separation against the *i*th surface. In conjunction with the 'outwards' facing orientation of $\hat{\boldsymbol{n}}_i$, Equation (6 yields a value greater than zero where separation exists. It should be noted that separation may also exist when the initiator's projection is less than the minimal value of the target's projected vertices. However, we disregard this case because separation will be determined with the evaluation of Equation (6 along the other surface normals.

We can conclude that the initiator has penetrated the target if Equation (6 produces negative values for all $\hat{\boldsymbol{n}}_i$.

3.2 Penetration Configuration

Equation (6 establishes the separation between the initiator and target. Where no separation exists, the initiator has penetrated the target. It is useful to express the evaluation of Equation (6 as a matrix:

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & \cdots & p_{1j} \\ \vdots & & \vdots \\ p_{i1} & \cdots & p_{ij} \end{bmatrix}, \qquad \text{for } i = 1, \dots, N_v, \qquad j = 1, \dots, N_q \tag{7}$$

with

$$p_{ij} = \begin{cases} 0, & \hat{\boldsymbol{n}}_i \cdot \left(\vec{\boldsymbol{r}}_{q_j} - \vec{\boldsymbol{r}}_{v_i}\right) > 0 \\ 1, & \hat{\boldsymbol{n}}_i \cdot \left(\vec{\boldsymbol{r}}_{q_j} - \vec{\boldsymbol{r}}_{v_i}\right) \leq 0 \end{cases} \tag{8}$$

where $\boldsymbol{P}$ is the binary penetration configuration matrix. Each entry p_{ij} indicates the separation status of the *j*th initiator from the target along the *i*th surface normal of the target. By Equation (8, a p_{ij} value of one indicates penetration along the *i*th surface feature of the bounding area. In contrast, a value of zero indicates separation.

Naturally, if the only objective is to detect penetration of the *j*th initiator, then it is easily calculated as

$$\sum_{i=1}^{N_v} p_{ij} = N_v \tag{9}$$

If Equation (9 holds, then the initiator has penetrated the target since the sum of surface penetrations is equal to the number of surfaces. In other words, no separation exists.

However, this is the limit of information provided by the separating axis theorem. It does not provide the necessary information to appropriately orient the restitutive collision forces required to emulate the real-world impenetrability of the simulated bodies. Here is where the presented algorithm extends the application of the SAT to determine the continuous collision configuration of the system.

3.3 Collision Configuration

As previously stated, there is a distinction between the penetration and collision configuration of a collision pair. The penetration configuration provides the necessary information to determine if the initiator overlaps the target. In contrast, the collision configuration provides the necessary information to apply restitutive collision forces in the appropriate direction.

Typically, the minimum translation vector (MTV) strategy is used to determine the collision configuration. In fact, it can be easily determined from the negative values in Equation (6 as

$$\min\left(\hat{\boldsymbol{n}}_i \cdot \left(\vec{\boldsymbol{r}}_{q_j} - \vec{\boldsymbol{r}}_{v_i}\right)\right) \tag{10}$$

where the index i corresponding to the minimum value of Equation (10 is the target's closest surface to the penetrated initiator. Consequently, the restitutive force can be applied in the direction of $\hat{\boldsymbol{n}}_i$. However, with the MTV, there are common edge-cases which negatively affect the fidelity of the collision model. As shown in Figure 5, near the vertices of the target, and for thin targets, there is an undesirable effect where the restitutive force is applied through the wrong surface.

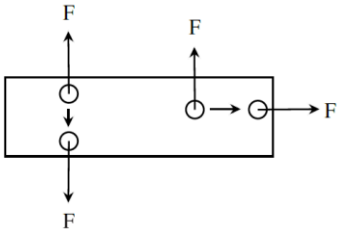

Figure 5: Minimum translation vector edge cases.

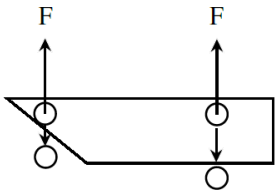

Figure 6: 'Collision-without-penetration' edge-cases for bounding area.

The main objective of the proposed algorithm is to remedy this problem in order to achieve higher fidelity in simulations. To this end, the algorithm must determine and retain the correct target collision surface for the duration of initiator penetration into the target.

At the onset of initiator penetration into the target, the last possible axis of separation is lost. By comparing the penetration configurations before and after the onset of penetration, we can determine with which surface the initiator has collided. We let $\boldsymbol{P}$ take on time dependent values $\boldsymbol{P}(t_0)$ and $\boldsymbol{P}(t)$ at reference time t_0 and current time t. Thus, we can characterize the time-dependent change in penetration configuration as

$$\Delta = \begin{bmatrix} \delta_{11} & \cdots & \delta_{1j} \\ \vdots & & \vdots \\ \delta_{i1} & \cdots & \delta_{ij} \end{bmatrix} = \boldsymbol{P}(t) - \boldsymbol{P}(t_0) \tag{11}$$

where Δ is the matrix which represents the transient change in the penetration configuration. This matrix represents the change in penetration at the current time t with respect to the reference time t_0. Each element δ_{ij} can take on one of three values with specific implications:

$$\delta_{ij} = \begin{cases} 1, & \text{then loss of separation on } \hat{\boldsymbol{n}}_i \\ 0, & \text{then no change on } \hat{\boldsymbol{n}}_i \\ -1, & \text{then separation on } \hat{\boldsymbol{n}}_i \end{cases} \tag{12}$$

Detection of the collision surface is possible when considering Equations (9, (11 and (12 together. Where Equation (9 is satisfied for the jth column of and $\boldsymbol{P}(t)$, then there is overlap between the jth initiator and target. At the time of collision Equations (11 and (12 show which surface is the last feature to lose separation as represented by $\delta_{ij} = 1$. Alternatively stated, the collision surface corresponds to the last surface normal to undergo `loss of separation'. Therefore, we introduce the binary collision configuration matrix $\boldsymbol{C}$:

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & \cdots & c_{1j} \\ \vdots & & \vdots \\ c_{i1} & \cdots & c_{ij} \end{bmatrix}, \qquad \text{for } i = 1, \ldots, N_v, \qquad j = 1, \ldots, N_q \tag{13}$$

with

$$c_{ij} = \begin{cases} 1, & \sum_{i=1}^{N_v} p_{ij} = N_v \ \wedge \ \delta_{ij} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $c_{ij} = 1$ indicates the collision surface. Thus far, $\boldsymbol{P}$, Δ and $\boldsymbol{C}$ provide the necessary information to detect the appropriate collision surface as presented in Equations (13 and (14. Yet, they do not account for the continuous retention of the collision configuration. For example, once a collision surface is determined, if the initiator is still in penetration on the next time step, then the change in penetration $\delta_{ij} = 0$,, which does not satisfy the first case in Equation (14 for collision. Therefore, it is necessary to treat $\boldsymbol{C}$ as a time-dependent value, similar to $\boldsymbol{P}(t)$, which takes on a reference configuration at time t_0 and a calculated configuration at the current time t. Given the reference configuration $\boldsymbol{C}(t_0)$, the current collision configuration $\boldsymbol{C}(t)$ adopts the reference configuration if penetration is retained in the current time step. This can be expressed through the addition of a case to Equation (14:

$$c_{ij}(t) = \begin{cases} 1, & \sum_{i=1}^{N_v} p_{ij} = N_v \ \wedge \ \delta_{ij} = 1 \\ c_{ij}(t_0), & \sum_{i=1}^{N_v} p_{ij} = N_v \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

As a result, the appropriate collision configuration is retained while the initiator is in penetration.

3.4 Collision Fidelity and Edge-case Management

Since the algorithm operates on the requirement that the initiator must be in penetration for collision to exist, a significant edge case presents itself: the restitutive collision force may not be strong enough to maintain impenetrability of the initiator onto the target. Consequently, the initiator may pass through the target. This edge-case is of particular concern for target bounding areas that are thin or contain acute angles, as shown in **Error! Reference source not found.**. To maintain collision fidelity, it is important to address this edge-case. However, it implies that collision can exist without of penetration.

The presented algorithm has constructed conditions for the determination of the collision configuration. As it relates to the presented edge case, conditions must also be constructed for the release from collision. To this end, we can establish a set of conditions where separation from the target does not incur a release from collision. As such, we introduce the binary surface range matrix $\boldsymbol{S}(t)$ as

$$\boldsymbol{S}(t) = \begin{bmatrix} s_{11} & \cdots & s_{1j} \\ \vdots & & \vdots \\ s_{i1} & \cdots & s_{ij} \end{bmatrix}, \qquad \text{for } i = 1, \dots, N_v, \qquad j = 1, \dots, N_q \tag{16}$$

with

$$s_{ij} = \begin{cases} 0, & \hat{\boldsymbol{p}}_i \cdot \vec{\boldsymbol{r}}_{v_i} \leq \hat{\boldsymbol{p}}_i \cdot \vec{\boldsymbol{r}}_{q_j} \leq \hat{\boldsymbol{p}}_i \cdot \vec{\boldsymbol{r}}_{v_{i+1}} \\ 1, & \text{Otherwise} \end{cases} \tag{17}$$

The surface range matrix $\boldsymbol{S}(t)$ is constructed from the projection of the incident vertices v_i and v_{i+1} and the *j*th initiator to the axis $\hat{\boldsymbol{p}}_i$, which lies parallel to the *i*th surface of the target. If the initiator lies within the range of the projected incident vertices, then the initiator is ‘on range'. For the *i*th surface, s_{ij} is the first condition to be met for collision retention without penetration. Collision between the *j*th initiator and *i*th surface can only continue to occur on its span; the range formed by its incident vertices.

The second condition considers the axis of separation. By Equation (12, separation, therefore an axis of separation, occurs where $\delta_{ij} = -1$. Normal release from collision occurs if separation exists along the same surface as the collision surface in the reference collision configuration. Otherwise, in order to retain collision after separation on the *i*th surface, the separation must not occur along the same surface as the collision surface. Therefore, we set the second condition as

$$\delta_{ij} \equiv 0 \tag{18}$$

which further implies the third condition

$$c_{ij}(t_0) \equiv 1 \tag{19}$$

where the initiator must be in collision at the reference configuration such that a collision configuration can be retained.

These three conditions resolve the case of collision without penetration for time steps where there is a reference collision configuration. It is important to note that this improvement is not a catch-all solution. While it mitigates some shortcomings of using penalty methods in conjunction with collision detection, it cannot address scenarios where the initiator passes from one side of the target to the other in a time step; or for surfaces with obtuse corners as the $s_{ij} \equiv 1$ condition may be violated under sliding conditions.

3.5 Implementation

There are two ways to program the algorithm. The first method is through the looping of conditional statements, while the second method applies matrix operations to express logic statements. Nevertheless, both algorithms follow the same scheme for the evaluation of the penetration configuration. The pseudocode for the penetration configuration is:

Penetration Configuration Algorithm

```
    Current penetration matrix
1:  for (j = 1 to N_q) do
2:      for (i = 1 to N_v) do
3:          if (n̂_i · (r⃗_{q_j} − r⃗_{v_i}) ≤ 0) do
4:              p_ij(t) = 1
5:          else
6:              p(t) = 0
7:          end if
8:      end for
9:  end for
    Current range matrix
10: for (j = 1 to N_q) do
11:     for (i = 1 to N_v) do
12:         if (p̂_i · r⃗_{v_i} ≤ p̂_i · r⃗_{q_j} ≤ p̂_i · r⃗_{v_{i+1}}) do
13:             s_ij(t) = 1
14:         else
15:             s_ij(t) = 0
16:         end if
17:     end for
18: end for
    Current penetration change matrix
19: Δ= P(t) − P(t_0)
```

Then, with the information provided by the penetration configuration algorithm, the collision configuration can be constructed. As previously mentioned, the two schemes are:

Collision Configuration Algorithm 1

Collision configuration matrix
for ($j = 1$ to N_q)
  for ($i = 1$ to N_v)
    if (sum($\boldsymbol{P}_j(t)$) $= N_v$) & ($\delta_{ij} = 1$)
      $c_{ij}(t) = 1$
    else if (sum($\boldsymbol{P}_j(t)$) $= N_v$)
      $c_{ij}(t) = 1$
    else if ($c_{ij}(t_0) = 1$) & ($s_{ij} = 1$) & ($\delta_{ij} = 0$)
      $c_{ij}(t) = 1$
    else
      $c_{ij}(t) = 0$
    end if
  end for
end for
Contact mechanics
Act on current collision configuration
Update reference configurations
$\boldsymbol{P}(t_0) = \boldsymbol{P}(t)$
$\boldsymbol{C}(t_0) = \boldsymbol{C}(t)$

Collision Configuration Algorithm 2

Collision guard matrices
$\boldsymbol{N}_0 = \boldsymbol{J}_{N_v}\boldsymbol{C}(t_0)$
for ($j = 1$ to N_q)
  if (sum($\boldsymbol{P}_j(t)$) $= N_v$)
    $\boldsymbol{N}_{1,j} = 1$
  else if ($\boldsymbol{P}_j(t)^T(\boldsymbol{C}_j(t_0) \circ \boldsymbol{S}_j(t) \circ (1 - \Delta_j)) = 1$)
    $\boldsymbol{N}_{1,j} = 1$
  else
    $\boldsymbol{N}_{1,j} = 0$
  end if
end for
$\boldsymbol{N}_2 = \boldsymbol{N}_0 \circ \boldsymbol{N}_1$
Retained collision
$\boldsymbol{C}_0 = \boldsymbol{C}(t_0) \circ \boldsymbol{N}_2$
New collision
$\boldsymbol{C}_1 = \Delta \circ (\boldsymbol{N}_1 - \boldsymbol{N}_2)$
Collision configuration matrix
$\boldsymbol{C}(t) = \boldsymbol{C}_0 + \boldsymbol{C}_1$
Contact mechanics
Act on current collision configuration
Update reference configurations
$\boldsymbol{P}(t_0) = \boldsymbol{P}(t)$
$\boldsymbol{C}(t_0) = \boldsymbol{C}(t)$

where Algorithm 1 is a nested loop of conditional statements. In contrast, Algorithm 2 uses matrix operations on binary matrices to obtain the collision configuration matrix. In a worst-case scenario, Algorithm 1 must exhaustively evaluate conditional statements. However, Algorithm 2 requires the introduction of the unitary matrix $\boldsymbol{J}_{N_v}$ of size $[N_v \times N_v]$ with ones in all entries, collision guard matrices $\boldsymbol{N}_0, \boldsymbol{N}_1$ and $\boldsymbol{N}_2$ of size $[N_v \times N_q]$ with columns which take on values of one or zero, and the Hadamard element-wise matrix multiplication '$\circ$'.

For Algorithm 2, the guard matrices ensure that the retained collisions and new collisions are applied to the appropriate initiators. Guard $\boldsymbol{N}_2$ ensures that retained collisions are applied only to initiators which are in collision at both t_0 ($\boldsymbol{N}_0$) and at t ($\boldsymbol{N}_1$). The Hadamard multiplication effectively acts as an element-wise logical 'AND' statement in this case. Likewise, new collisions are applied only to initiators which come into penetration in the current time step. This is achieved through the subtraction of $\boldsymbol{N}_2$ from $\boldsymbol{N}_1$.

When considering N_q initiators interacting with a target with N_v vertices, the processing of the penetration, range, and penetration change matrices are the same for both implementations. However, the key difference lies within the processing of the current collision configuration. While the second algorithm requires the construction of guard matrices, it requires fewer conditional statement evaluations and enables the algebraic construction of the current collision configuration. Since the Hadamard products, additions, and subtractions are all element-wise operations, the second algorithm may lend itself well to parallelization optimizations for larger systems.

Nevertheless, once the collision configuration has been constructed by either algorithm, it can be passed to the model which generates the collision restitution forces. Subsequently, the current collision configuration is passed to the reference collision configuration for the next time step.

4. Sample Application

The algorithm was originally developed as part of a contact mechanics model for larger scope of research in dynamic interface analysis (DIA) – the characterization and analysis of the dynamic interaction between ship decks and vertical take-off aircraft under varying hydrodynamic, aerodynamic, and mechanical conditions. The research yielded the software package SRAMSS-2D (Skid-equipped Rotary-wing Aircraft Manoeuvring and Securing Simulation), which simulates the landing, securing, and manoeuvring of ship-embarked rotary-wing aircraft with skid tube landing gear [7].

As shown in Figure 7, finite element (FE) techniques were chosen to represent the skid landing gear since the tubing configuration of skid landing gear resembled FE meshing when modelled using beam-type elements. Considering that external forces were applied at the FE nodes and stiff skid landing gear present complex and often intermittent contact between with ship deck, a suitable collision detection scheme was necessary.

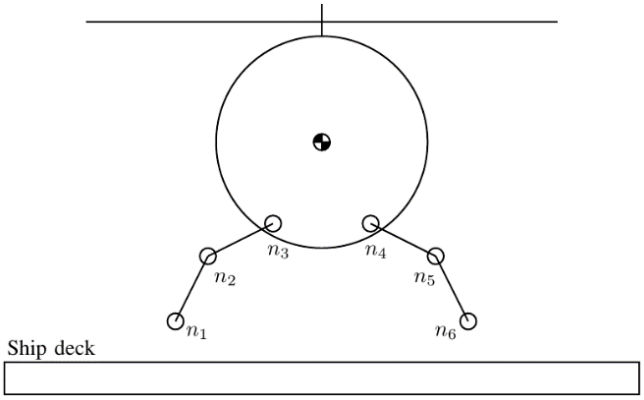

Figure 7: Generic aircraft with 6-node finite element landing gear. Nodes 1 and 6 are designated as initiators.

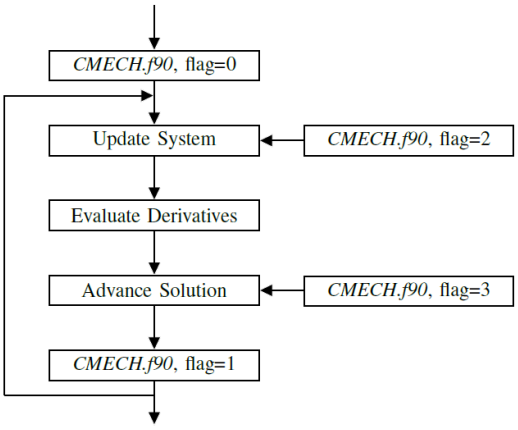

Figure 8: Contact mechanics subroutine calls in SRAMSS-2D.

The presented algorithm was developed in order to satisfy: the penalty method for generating restitutive forces which act proportional to the penetration depth; point initiators representing the FE nodes such that the collision forces could be applied to the FE node forcing vector; and retain collision fidelity to achieve more accurate simulations. Further, the unilateral nature of the algorithm arose from the fact that the motion of the bounding area representing the ship deck was known for the duration of the simulation. Therefore, only the collision forces acting on the FE landing gear were required, which greatly improved computational performance. Additionally, by culling potential initiators to only include FE nodes (initiators) expected to make contact, performance was further increased.

For blind forward time propagation, the two implementations of the collision configuration algorithm are satisfactory. However, a more nuanced implementation in SRAMSS-2D was required. The numerical integration scheme used by SRAMSS-2D, the *DLSODAR.f* subroutine written in Intel® FORTRAN [8], [9], had the ability to reject propagated time steps. This posed potential problems when passing the current penetration and collision configurations to the reference configurations. The issue was solved by passing different flags to the contact mechanics subroutine *CMECH.f90* when called, as shown in Figure 8, to perform only the necessary calculations.

Flag 0 was used for the initial call of the subroutine prior to the solution advancing loop to determine the initial reference configurations. After successful numerical integration through the current time step, the subroutine was called with Flag 1 to evaluate the current penetration and collision configurations, then assigned them to the reference configurations for subsequent time step integrations. During numerical integrations of the time step, Flag 2 was used. The current collision configuration was calculated to generate the collision forces within the system. Since the subroutine was repeatedly called during integration, reference configurations were not updated due to possible intermediate time-step rejections. The subroutine was called with Flag 3 during integrator root function evaluations. Root functions, expressed in terms of the actual separation values of Equation (6, were leveraged to terminate time-steps exactly at collision in order to separate the in-collision and out-of-collision dynamical subsystems. This bypassed numerical integration through the discontinuities presented by collision.

Remarkably, the algorithm's implementation in SRAMSS-2D realizes significant potential in the broader scope of physics-based simulations. Multiple initiators were used to represent the FE nodes of the modelled skid-type landing gear. This suggests that the initiators, as presented in this work, can be combined to represent complex bodies with no restrictions on convexity or deformability so long as the desired models are applied to capture the internal forces at play between initiators. Some examples of potential applications include the modelling of very flexible tires over terrain with prominent features and simulating mechanically-complex robotic grippers. In any case, it extends unilateral collision detection using the SAT to collision pairs where the initiator may have convex features.

5. Conclusion

Collision detection is a crucial component in a wide variety of computer and engineering applications. The continual advancements in computational power have enabled the development of higher-fidelity simulation packages. As a result, for physics-based simulations, the collision detection and response schemes must also be refined.

This work has presented a scheme for the continuous detection and retention of the appropriate collision surface between a convex bounding area and point. This is achieved through the continuous evaluation of the SAT. The governing notion that the last feature of the target to lose separation from the initiator constitutes the collision feature, is what enables the appropriate collision surface determination.

Ultimately, the presented algorithm provides realistic collision detection for contact mechanics models in physics-based simulations. Moreover, limiting the initiator in the collision pair to a single point enables the construction of complex bodies with known locations and orientations for the application of collision restitution forces. With the growing trend in application of finite element methods, the presented algorithm realizes many useful applications.

6. References

[1] S. Kockara, T. Halic, K. Iqbal, C. Bayrak and R. Rowe, "Collision detection: A survey," in *IEEE International Conference on Systems, Man and Cybernetics*, Montreal, 2007.

[2] M. Figueiredo, L. Marcelino and T. Fernando, "A Survey on Collision Detection Techniques for Virtual Environments," Centre for Virtual Environments, University of Salford, Salford, UK, 2002.

[3] S. Gottschalk, "Separating Axis Theorem," Department of Computer Science, UNC Chapel Hill, Chapel Hill, US, 1996.

[4] C. Ericson, Real-Time Collision Detection, CRC Press, Inc., 2004.

[5] M. Fi, P. Konecn, K. Zikan and P. Konen, "Lower Bound Distance in 3D," 2001.

[6] S. Gottschalk, M. Lin and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, 1996.

[7] A. Schock and R. Langlois, "Spacial Simulation of Shipboard Operations for Skid-equipped Rotary-wing Aircraft," in *Canadian Society for Mechanical Engineering International Congress*, Charlottetown, P.E.I., 2020.

[8] A. C. Hindmarsh, "ODEPACK.f," Center for Applied Scientific Computing, L-561, Livermore, 2008.

[9] A. C. Hindmarsh, "A Systematized Collection of ODE Solvers," *IMACS Transactions on Scientific Computation*, vol. 1, 1983.