
Sujet : Détection de collisions

Il est rappelé que le jury attend un exposé de 35 minutes, pédagogique et structuré, fondé sur ce sujet. Ce texte se conclut par une liste de pistes de réflexion pour guider la conception de l'exposé, mais toute initiative personnelle pertinente est appréciée. L'exposé doit contenir une ou plusieurs illustrations informatiques ainsi qu'une discussion autour d'une dimension éthique, sociétale, environnementale, économique ou juridique. Une piste particulière est proposée à cet effet.

1 Introduction

Ce sujet porte sur les algorithmes de détection de collisions entre des objets bidimensionnels, modélisés comme des parties du plan $\mathbb{R}^2$. On considère que deux objets A et B sont en *collision* si leur intersection est non vide ($A \cap B \neq \emptyset$).

Les algorithmes de détection des collisions sont utilisés dans de nombreux domaines : simulations physiques (billards, systèmes de particules, etc.), planification du mouvement en robotique, rendu d'environnements 3D en 2D, systèmes d'information géographique (SIG) ou encore jeux vidéo. Concevoir des algorithmes efficaces est un enjeu majeur, tout particulièrement pour les systèmes devant fonctionner en temps réel.

Dans la suite, nous étudions différents types de problèmes :

- le test de collision entre deux objets polygonaux (section 2) ;
- le test de collision entre un ensemble d'objets – vu comme des obstacles fixes – et un objet se déplaçant entre ces obstacles (section 3) ;
- le problème de la détection de toutes les collisions entre toutes les paires d'objets dans le cadre d'un système en cours d'évolution (section 4).

2 Collision entre deux objets polygonaux

Dans la suite de ce sujet, nous étudions des objets polygonaux d'au moins trois sommets – le bord étant toujours inclus. Chaque objet est décrit par la liste des coordonnées de ses sommets, donnés dans l'ordre du contour. Un exemple de tel objet est donné sur la figure 1.

2.1 Un algorithme pour la détection de collision d'objets polygonaux convexes

L'algorithme le plus simple permettant de tester la collision d'objets polygonaux convexes repose sur le théorème suivant.

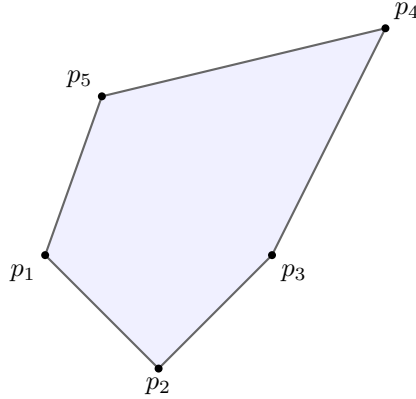


FIGURE 1 – Un objet polygonal convexe à cinq sommets.

Théorème 2.1 *On considère deux polygones convexes P_1 et P_2 . Alors $P_1 \cap P_2 = \emptyset$ si et seulement s'il existe un côté C de l'un des polygones P tel que*

- *tous les sommets de P sont inclus dans un demi-plan délimité par la droite prolongeant C (au sens large); (1)*
- *et tous les sommets de l'autre polygone sont inclus dans l'autre demi-plan délimité par cette même droite (au sens strict). (2)*

Algorithme 1 : Test de collision entre deux objets convexes polygonaux

Entrées : deux polygones convexes P_1 et P_2

- 1 pour** chaque côté C de chacun des polygones **faire**
 - 2** **|** Tester si C satisfait les conditions (1) et (2) du théorème.
-

L'implémentation de cet algorithme nécessite de déterminer de quel côté d'une droite (D) est situé un point $p = (x, y)$ du plan. Si une telle droite passe par les deux points distincts $p_1 = (x_1, y_1)$ et $p_2 = (x_2, y_2)$, on peut calculer la quantité $d = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$. Alors, $d > 0$ si et seulement si p est à droite de (D) (en orientant p de p_1 vers p_2 , $d < 0$ si et seulement si p est à gauche de (D), et $d = 0$ si et seulement si $p \in (D)$).

2.2 Calcul de boîtes englobantes

Dans un contexte où l'on devra effectuer de nombreux tests de collision entre des objets polygonaux ayant de nombreux sommets, l'algorithme 1 peut s'avérer trop coûteux.

Il peut alors devenir pertinent de calculer des *boîtes englobantes* de chacun des polygones. Ces boîtes doivent être des objets géométriques simples, rapides à calculer, et incluant l'objet de départ.

Les rectangles avec côtés parallèles aux axes forment le type le plus simple de boîtes englobantes. Un exemple en est donné sur la figure 2.

Lorsque l'on souhaite tester la collision de deux objets, on peut commencer par tester la collision de leurs boîtes englobantes : si celles-ci ne s'intersectent pas, les objets ne pourront pas non plus s'intersecter.

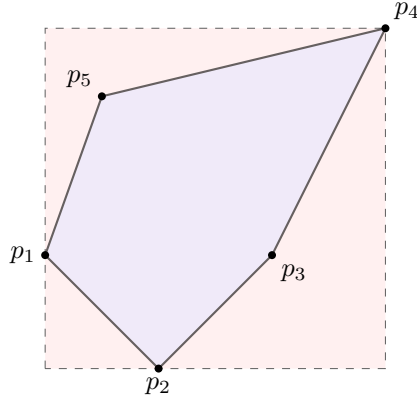


FIGURE 2 – Boîte rectangulaire englobant un polygone convexe à 5 sommets.

3 Objet dans un environnement fixe

On se place désormais dans un *environnement* $\mathcal{E}$ fixe décrit par un ensemble de polygones appelés *obstacles*. Étant donné un objet polygonal O , on souhaite déterminer s'il est en collision avec $\mathcal{E}$ et/ou déterminer l'ensemble des obstacles $\mathcal{E}$ avec lesquels il est en collision (figure 3). De tels environnements vous sont fournis dans les fichiers `polygones1.txt` et `polygones2.txt`. Il est possible d'afficher les environnements en utilisant `trace_polygones.py`.

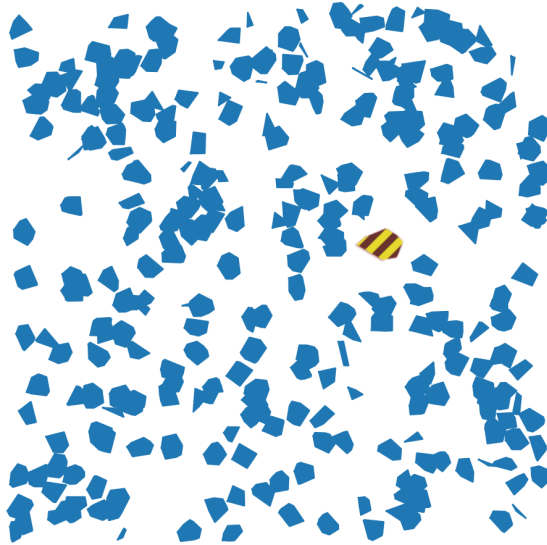


FIGURE 3 – Un environnement de 300 obstacles convexes et l'objet O (hachuré)

On cherche à limiter le nombre de tests de collisions entre paires d'objets à effectuer pour résoudre ce problème. L'algorithme naïf consiste simplement à tester la collision de O avec chaque obstacle de $\mathcal{E}$. Les améliorations qui suivent sont basées sur un prétraitement des environnements permettant par la suite de réduire le nombre de collisions à tester.

3.1 Dichotomie sur les abscisses puis balayage

Pour chaque obstacle $P_i \in \mathcal{E}$, on définit les quantités suivantes :

- x_i^{min} est l'abscisse minimale d'un sommet de P_i ;
- x_i^{max} est l'abscisse maximale d'un sommet de P_i ;
- $r_i = (x_i^{max} - x_i^{min})/2$ est le *rayon horizontal* de P_i ;
- $m_i = (x_i^{min} + x_i^{max})/2$ est son *abscisse moyenne*.

On note par ailleurs $R = \max_{P_i \in \mathcal{E}} r_i$ le rayon horizontal maximal des obstacles de $\mathcal{E}$.

On définit également x_O^{min} et x_O^{max} comme les abscisses minimales et maximales d'un sommet de l'objet O , ainsi que $m_O = (x_O^{min} + x_O^{max})/2$.

On remarque qu'il ne peut pas y avoir de collision entre O et P_i dès lors que $x_O^{min} > m_i + R$ ou que $x_O^{max} < m_i - R$ (conditions (3)).

Si l'on prétraite $\mathcal{E}$ en triant ses éléments par m_i croissants, on peut alors utiliser l'algorithme 2 afin de déterminer les collisions entre O et les P_i .

Algorithme 2 : Algorithme par balayage

Entrées : Un tableau de polygones T ordonné par abscisses moyennes croissantes, un polygone O

- 1 Rechercher l'emplacement i où m^O devrait être inséré dans T .
 - 2 Effectuer à partir de i un balayage du tableau vers la droite jusqu'à ce que la condition (3) soit vérifiée, et tester la collision de O avec chaque polygone de T rencontré lors du balayage.
 - 3 Effectuer le balayage symétrique vers la gauche.
-

3.2 Arbres AABB

Les *arbres AABB* (pour Axis Aligned Bounding Box Trees¹) sont une variante hiérarchique des boîtes englobantes rectangulaires (figure 4). Étant donné un ensemble d'obstacles $\mathcal{E}$, un arbre AABB est un arbre binaire

- dont chaque feuille correspond au rectangle englobant d'un obstacle de $\mathcal{E}$;
- et dont chaque nœud interne A de fils A_1 et A_2 correspond au rectangle englobant les deux rectangles englobants de A_1 et A_2 .

Un arbre AABB pour un ensemble d'obstacles donné peut être construit par un algorithme de regroupement hiérarchique ascendant : une feuille est construite à partir de chacun des obstacles – formant ainsi un ensemble d'arbres $\mathcal{A}$. Tant que $\mathcal{A}$ contient au moins deux arbres, on en choisit deux, et on les regroupe pour former un nœud.

Une approche descendante est également possible : pour construire un arbre sur un ensemble d'obstacles $\mathcal{E}$, on partitionne $\mathcal{E}$ en deux parties P_1 et P_2 , pour lesquelles on construit récursivement des arbres A_1 et A_2 , puis on renvoie l'arbre A dont les fils sont A_1 et A_2 .

La recherche de l'ensemble des obstacles de $\mathcal{E}$ en collision avec O peut alors être effectuée par l'algorithme 3.

1. Arbres de boîtes englobantes alignées avec les axes

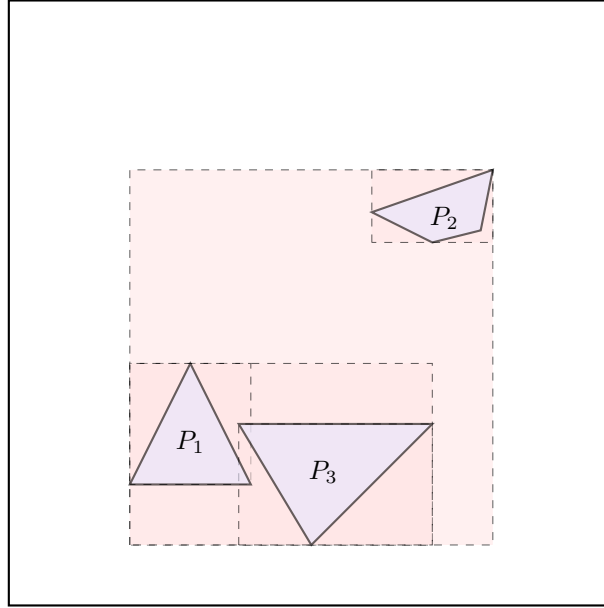


FIGURE 4 – Un arbre ABB à trois feuilles et deux nœuds internes pour un ensemble d’obstacles $\mathcal{E} = \{P_1, P_2, P_3\}$

Algorithme 3 : Recherche de collision dans un ABB

Entrées : un arbre ABB A , un objet O

- 1 **si** A est une feuille **alors**
 - 2 └ Tester si l’obstacle contenu dans A est en collision avec O .
 - 3 **sinon**
 - 4 └ $A_1, A_2 \leftarrow$ les fils de A .
 - 5 **si** O est en collision avec le rectangle englobant de A_1 **alors**
 - 6 └ Rappeler récursivement l’algorithme sur A_1
 - 7 **si** O est en collision avec le rectangle englobant de A_2 **alors**
 - 8 └ Rappeler récursivement l’algorithme sur A_2
-

3.3 Quadrees

Les *Quadrees* (arbres quaternaires) sont une méthode de partitionnement récursif d’un plan.

Partant d’un espace carré à partitionner, la racine de l’arbre représente le carré en entier. On découpe ce carré initial en quatre carrés égaux, qui seront représentés par les fils de la racine, et ainsi de suite récursivement. On peut décider de s’arrêter systématiquement à une profondeur donnée, auquel cas on obtient un arbre complet (dont tous les niveaux sont remplis), ou bien de s’arrêter de subdiviser un carré dès lors qu’il n’intersecte aucun obstacle de $\mathcal{E}$ (figure 5). Chaque nœud interne a donc quatre fils et chaque feuille contient la liste des obstacles de $\mathcal{E}$ que son carré intersecte.

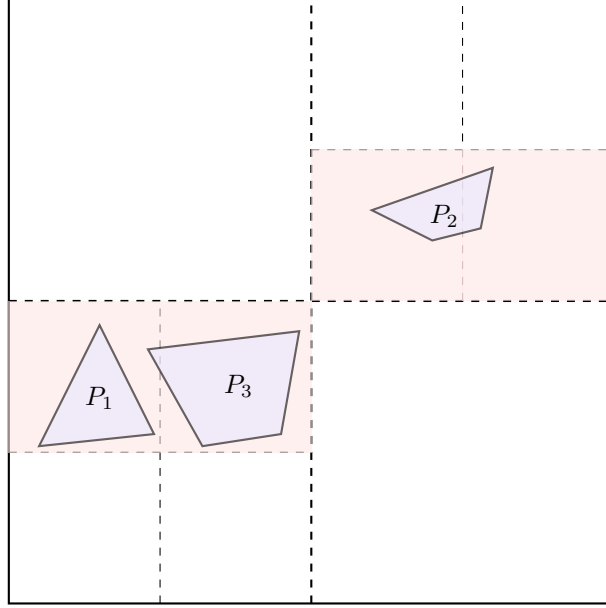


FIGURE 5 – Un quadtree Q incomplet de profondeur 2 pour un ensemble d'obstacles $\mathcal{E} = \{P_1, P_2, P_3\}$. Q a 10 feuilles, dont les 4 qui contiennent au moins un obstacle sont coloriées. L'obstacle P_1 est contenu dans une unique feuille (qui contient également P_3), tandis que P_2 et P_3 sont chacun contenus dans deux feuilles.

Le test d'intersection avec un objet O consiste alors comme pour les arbres AABB à partir de la racine et à n'explorer que les fils représentant des carrés qui intersectent O .

4 Détection de collisions deux à deux dans un environnement dynamique

On considère maintenant un ensemble d'objets $\mathcal{E}$ en mouvement. On peut par exemple penser à une simulation physique d'un billard. En l'absence de collision, chaque boule B_i , en position (x_i, y_i) se déplace en ligne droite à vitesse constante $\vec{v}_i$. Dans une simulation à temps discret, on avance par pas de temps dt . Le passage de l'instant t à $t + dt$ implique la mise à jour des positions en modifiant la position de chaque objet ainsi : $(x_i, y_i) \leftarrow (x_i, y_i) + dt \cdot \vec{v}_i$.

Une fois ce pas de simulation effectué, il importe alors de détecter les collisions deux à deux des objets de $\mathcal{E}$ pour pouvoir modifier leur vitesse en utilisant des lois physiques décrivant l'effet de la collision sur celle-ci. On parle de *détection de collision a posteriori* pour ce type d'approche – car les collisions sont détectées après qu'un pas de simulation a été effectué.

Il y a donc deux enjeux nouveaux par rapport aux sections précédentes :

- Comment détecter les collisions deux à deux d'un ensemble d'objets ?
- Comment tirer parti de la cohérence temporelle (i.e. du fait que la position des objets varie «peu» lors d'une itération temporelle de la simulation) pour accélérer les calculs de collisions ?

On propose pour ce faire une méthode dédiée, basée sur les boîtes englobantes des objets.

On considère un ensemble d'objets $\mathcal{E} = \{P_1 \cdots P_n\}$ dans une position initiale, chaque objet P_i ayant un rectangle englobant $[a_i, b_i] \times [c_i, d_i]$.

Au début de la simulation, on place tous les a_i et b_i dans un unique tableau $\mathbf{tx}$ trié par ordre croissant. Un parcours linéaire de $\mathbf{tx}$ permet alors de déterminer l'ensemble S_x des couples d'objets

(P_i, P_j) tels que les intervalles d'abscisses $[a_i, b_i]$ et $[a_j, b_j]$ se chevauchent. On procède de la même façon pour les intervalles $[c_i, d_i]$ pour calculer l'ensemble S_y correspondant pour les ordonnées. On obtient alors l'ensemble $S = S_x \cap S_y$ des couples d'objets dont les boîtes englobantes s'intersectent, à partir duquel on effectue les tests de collisions effectifs.

Lors d'une itération temporelle de la simulation, chaque objet P_i est légèrement déplacé ainsi que sa boîte englobante $[a_i, b_i] \times [c_i, d_i]$. On considère itérativement chaque objet P_i et on note $[a'_i, b'_i] \times [c'_i, d'_i]$ sa nouvelle boîte englobante après déplacement. Dans **tx**, on remplace a_i et b_i par a'_i et b'_i , puis on déplace ces éléments en les permutant itérativement avec leurs voisins pour rétablir la monotonie de **tx** tout en mettant à jour S_x . On procède de même pour S_y . On obtient finalement un nouvel ensemble S de couples d'objets dont les boîtes englobantes s'intersectent à partir duquel on effectue à nouveau les tests de collisions effectifs.

Pistes de réflexion pour l'exposé

1. Implémenter l'algorithme pour la détection de collision d'objets polygonaux convexes, avec ou sans boîte englobante, et discuter sa complexité.
2. Discuter et implémenter des solutions pour générer des jeux de données pour les différents problèmes posés.
3. Implémenter l'algorithme 2, et le comparer à l'algorithme naïf. Discuter ses limites, en commentant notamment le rôle de R .
4. Implémenter un algorithme de construction d'un arbre AABB ascendant ou descendant – en discutant le choix des arbres à regrouper ou de la partition de l'ensemble $\mathcal{E}$ à choisir à chaque étape, puis implémenter la recherche de l'ensemble des collisions avec un objet O . Comparer avec l'approche naïve.
5. Implémenter un algorithme de construction d'un quadtree Q , puis implémenter la recherche de l'ensemble des collisions avec un objet O . Comparer avec l'approche naïve, en commentant le choix de la profondeur de Q .
6. Les quadrees découpent le plan de façon régulière, ce qui semble sous-optimal si les obstacles ne sont pas répartis de façon homogène. Discuter et implémenter une approche de type *arbre k-dimensionnels* pour obtenir un découpage du plan plus adapté à la répartition des obstacles. Comparer avec les quadrees.
7. Discuter les structures de données nécessaires à l'implémentation de la méthode présentée dans la section 4 pour la détection de collisions deux à deux dans un environnement dynamique, et préciser les détails de son implémentation.
8. Discuter de l'impact de la représentation des nombres réels sur les méthodes présentées. On pourra notamment discuter du test de collision de la section 2 ou du choix d'un pas de temps pour la méthode présentée dans la section 4.
9. Discuter des possibilités de parallélisation des différents algorithmes présentés dans ce document.
10. Les algorithmes de détection de collision sont au cœur des systèmes de freinage d'urgence des véhicules autonomes. Discuter des enjeux juridiques et éthiques liés à la fiabilité de ces algorithmes en cas d'accident.