УДК 004.519.712+004.92

В.М. ТЕРЕЩЕНКО

РОЗВ'ЯЗАННЯ ЗАДАЧІ ПЕРЕТИНУ m ОПУКЛИХ МНОГОГРАННИКІВ

Анотація. *У статті запропоновано модифікацію алгоритму GJK для знаходження спільної точки двох опуклих многогранників. Знаючи цю точку та використовуючи теорему двоїстості, можна знаходити область перетину двох політопів. Розглядаючи політопи попарно, можна знайти область перетину m політопів. Розроблена паралельна програмна реалізація розв'язку проблеми, яка ефективно використовує ресурси сучасних багатопроцесорних систем.*
Ключові слова: *перетин, політоп, многогранник, різниця Мінковського, область перетину, геометрична двоїстість.*

Аннотация. *В статье предложена модификация алгоритма GJK для нахождения общей точки двух выпуклых многогранников. Зная эту точку и используя теорему двойственности, можно определить область пересечения двух политопов. Рассматривая политопы попарно, можно найти область пересечения m политопов. Разработана параллельная программная реализация решения проблемы, которая эффективно использует ресурсы современных многопроцессорных систем.*
Ключевые слова: *пересечение, политоп, многогранник, разность Минковского, область пересечения, геометрическая двойственность.*

Abstract. *In this paper we consider the modified version of GJK algorithm for finding a common point of two convex polyhedrons. Knowing this point and using the duality theorem, we can find the intersection of two polytopes. Considering the polytopes in pairs, we can find an intersection of m polytopes. A parallel software implementation of problem solution which effectively utilizes resources of modern multiprocessor systems was developed.*
Keywords: *intersection, polytope, polyhedron, Minkowski difference, domain intersection, geometric duality.*

1. Вступ

Постановка проблеми. В роботі розглядається один із підходів до розв'язання задачі знаходження області перетину m опуклих політопів. Ця задача пов'язана з відомою і дуже популярною проблемою "Collision detection" [1], яка широко використовується у моделюванні фізичних процесів та в комп'ютерних іграх [1–4]. Процедура визначення перетину є одна з основних геометричних операцій, яка виникає у багатьох застосуваннях комп'ютерної графіки та робототехніки [5, 6]. Вона стала важливою з появою нових застосувань у віртуальній реальності, моделюванні та фізично-орієнтованій анімації як один із основних кроків алгоритмів. Окрім того, ця процедура має своє застосування в задачах лінійного програмування для багатовимірних просторів або ж у знаходженні спільних ознак об'єктів, які можуть бути подані у вигляді многогранників. Більшість доступних сьогодні бібліотек вимагають, щоб об'єкти були представлені у вигляді опуклих многогранників через наявність ефективних алгоритмів визначення їх перетину. Також вона є базовою під час розв'язання більш складних проблем, наприклад, знаходження області перетину m неопуклих многогранників. У цьому випадку задачу можна звести до перетину опуклих многогранників, використовуючи декомпозицію або спеціальні структури даних, наприклад, BSP-дерева [1] . Першим недоліком цього підходу є те, що декомпозиція вводить багато фіктивних ребер і граней, які повинні бути перевірені на наявність перешкод. Але другим, більш фундаментальним, недоліком є те, що алгоритми, засновані на таких розкладах, не завжди працюють належним чином на об'єктах, отриманих у результаті проективного перетворення. Так, зокрема, в результаті такого перетворення ми можемо одержати многогранник, який не є опуклим.

Альтернативне представлення об'єкта з використанням інших бібліотек полягає в описі границі об'єкта як сукупності граней, які специфікуються координатами своїх вершин, упорядкованих за (або проти) годинниковою стрілкою, якщо дивитися з зовнішнього боку. Перевірка на перетин повинна проходити зазвичай в реальному часі і працювати з дуже детальними об'єктами, а тому надто важливою є складність алгоритмів та об'єм пам'яті, яка при цьому використовується.

Аналіз останніх досліджень. На сьогоднішній день існують три основних підходи до розв'язання цієї задачі: відсікання частин одного політопа гранями іншого [1], метод plane-sweep [7] та використання теореми двоїстості [8]. Перший метод є інтуїтивним, простим, але не дуже ефективним. Більш ефективними є другий та третій методи, порівняно з першим. Другий метод є узагальненням методу плоского замітання для тривимірних випадків і детально описаний у [7]. В той самий час перший та останній методи є більш прості у реалізації. Зокрема, третій підхід детально описаний у роботах [8, 9]. Так, у роботі [8] представлено алгоритм з часом $O(n\log n)$. А в роботі Б. Чазеле [9] запропоновано алгоритм з лінійним часом. Б. Чазеле показав, як можна застосувати до розв'язання задачі побудови перетину двох многогранників ієрархічне представлення многогранників Добкіна-Кіркпатрика [10]. Особливо цікавим в алгоритмі є використання простої ієрархії для представлення внутрішньої області кожного многогранника та подвійної ієрархії для представлення зовнішньої частини кожного многогранника. Таким чином, останній підхід є найоптимальнішим з точки зору швидкості та простоти реалізації. Цей підхід детально описаний у роботах [1, 8, 9].

У ряді робіт, присвячених цій тематиці, пропонується підхід до вирішення задачі шляхом розділення її на підзадачі. Тут варто відзначити роботи [5, 10, 11]. Однією із основних підзадач є задача визначення перетину відрізків, а найефективнішим алгоритмом її вирішення на сьогодні є алгоритм Балабана [12].

Для задач пошуку перетинів в умовах наявності когерентності дуже часто використовується GJK-алгоритм [2, 6], який базується на тому факті, що два опуклих об'єкти (многогранники) перетинаються тільки у тому випадку, якщо початок координат належить їх різниці Мінковського (РМ). Насправді, GJK-алгоритм також дає можливість визначити мінімальну відстань між РМ та початком координат. Так як пошук РМ на кожній ітерації алгоритму потребує дуже багато часу, в GJK використовується функція support mapping. Це функція, яка за заданим напрямом повертає екстремальну точку для опуклого об'єкта. Екстремальні вершини можуть бути знайдені за $O(n)$ часу. Якщо зберігати інформацію про сусідні вершини, то екстремальну вершину можна знайти звичайним Hill Climbing алгоритмом. Для великих многогранників можна пришвидшити цей алгоритм, зберігаючи окрім сусідніх вершин ще одну або декілька віддалених вершин. Перевагою цього методу є можливість його використання для будь-якого опуклого об'єкта [3, 4]. Також для розв'язку такого роду задач використовуються діаграми Вороного (V-Clip algorithm) [13], BSP (Binary Space Partitioning)-дерева. Ідея, запропонована у даній роботі, базується на алгоритмі GJK [6, 8].

Метою статті є спроба розробити узагальнений і в той же час ефективний метод побудови області перетину *m* опуклих політопів, удосконаливши існуючі методи і підходи.

Новизна та ідея. В роботі запропоновано модифікований алгоритм GJK, який дозволяє за лінійний час визначити спільну точку двох політопів, що робить можливим знаходження їх області перетину за час $O(n\log n)$. Алгоритм узагальнений для m політопів і дуже добре розпаралелюється. Окрім того, у роботі представлено новий підхід, який узагальнює існуючі алгоритми, покращуючи їх ефективність. Підхід передбачає випадок використання передобробки, що полягає у створенні для многогранника його представлення у вигляді ієрархії Добкіна-Кіркпатрика.

2. Постановка та побудова розв'язку задачі

Сформулюємо геометричну постановку задачі побудови області перетину m полі топів.

Постановка задачі побудови області перетину політопів. Нехай задано m опуклих поліедрів з N вершинами кожний. Необхідно побудувати їх перетин.

Область перетину m політопів можна знайти, використовуючи відому техніку «Розділяй та володарюй». Для цього ми будемо знаходити попарно перетин політопів до тих пір, доки не залишиться один політоп (результат перетину) або ж не буде показано, що перетин – порожня множина. Далі буде розглядатися алгоритм перетину двох політопів.

2.1. Побудова розв'язку задачі перетину політопів

Для розв'язання задачі перетину двох політопів скористаємось методом, запропонованим у роботі [8]. Центральна ідея методу полягає у визначенні першої точки перетину p, що далі дозволяє побудувати перетин за допомогою техніки двоїстості. Згідно з цією технікою задається перетворення просторів, яке переводить точку p в заданому просторі у площину $\delta(p)$ двоїстого простору і навпаки, площину $\delta(\pi)$ двоїстого простору у двоїсту до π точку заданого простору. Згідно з [8], мають місце такі твердження.

Твердження 1. Нехай P – опуклий політоп, що містить початок координат, а S – множина точок, двоїстих до площин, що несуть грані P. Тоді будь-яка точка p, що лежить в середині P, відображується у таку площину $\delta(p)$, яка не перетинається з поліедром $conv(S)$.

Твердження 2. Якщо P – опуклий поліедр, що містить початок координат, то таким самим є і двоїстий до нього поліедр $P^{(\delta)}$.

Наведені твердження демонструють той факт, що якщо P містить початок координат, то будь-яка зовнішня до P площина відображується в точку в середині $P^{(\delta)}$ і навпаки.

Отже, для того щоб знайти перетин двох опуклих політопів, скористаємось таким алгоритмом.

1) Знаходимо точку, що лежить в середині обох політопів, за допомогою модифікованого алгоритму GJK.

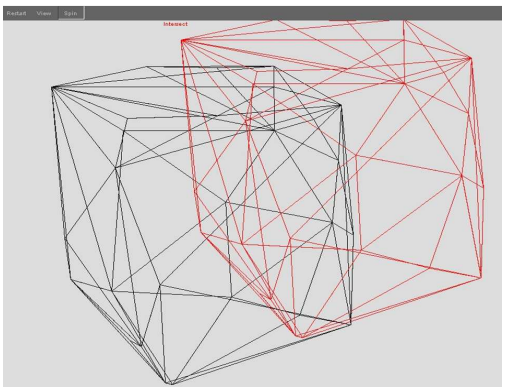

Рис. 1. Приклад роботи алгоритму визначення перетину двох політопів

2) Переміщуємо початок координат у знайдену точку. Нормуємо всі рівняння площин, до яких належать грані політопів, і знаходимо множину точок S, двоїстих до граней політопів.

3) Будуємо опуклу оболонку $CH(S)$ і знаходимо множину точок $\overline{S}$, двоїстих до граней $CH(S)$.

4) Будуємо опуклу оболонку $CH(\overline{S})$.

Отриманий політоп є шуканим перетином.

На рис. 1 показано реалізацію алгоритму для знаходження перетину двох політопів.

2.2. Побудова розв'язку задачі знаходження точки в середині обох політопів

Означення 1. Різниця Мінковського двох множин точок визначається за формулою

$$A\Theta B = \{a - b : a \in A,\ b \in B\}.$$

Важливою властивістю різниці Мінковського є те, що дві множини точок перетинаються (мають одну або більше спільних точок) тоді і тільки тоді, коли різниця Мінковського $C\left(C = A\Theta B\right)$ містить у собі початок координат.

Лема 1 [6]. Визначення мінімальної відстані між множинами точок A і B еквівалентне визначенню мінімальної відстані між їх різницею Мінковського С та початком координат.

$$distance(A,B) = \min\{\|a - b\| : a \in A,\ b \in B\} = \min\{\|c\| : c \in A\Theta B\}.$$

Цією лемою ми будемо користуватися в подальшому для знаходження спільної точки двох політопів.

Алгоритм GJK [6]

1. Ініціюємо симплексну множину Q однією або більшою кількістю точок (не більше ніж $d+1$, де d – розмірність простору) з різниці Мінковського множини вершин політопів A і B.
2. Визначаємо точку P мінімальної норми в опуклій оболонці $P \in CH(Q')$.
3. Якщо P – початок координат, то початок координат міститься в середині $A\Theta B$. Отже A і B перетинаються.
4. Зменшуємо Q до найменшої підмножини Q' з Q такої, що $P \in CH(Q')$. Робимо це видаленням усіх точок з Q, які не визначають підсимплекс Q, до якого P належить.
5. Нехай $V = S_{A\Theta B}(-P) = S_A(-P) - s_B(P)$ є опорною точкою у напрямку $-P$.
6. Якщо V не більш екстремальна у напрямку $-P$, ніж сама P, то зупиняємось і повертаємо, що A та B не перетинаються.
7. Додаємо V до Q та повертаємось до кроку 2.

Опишемо процедуру знаходження точки P мінімальної норми в опуклій оболонці $CH(Q)$ для симплексної множини $Q = \{Q_1, Q_2, ..., Q_k\}$, $1 \le k \le 4$. Для цього переглядаємо регіони Вороного для вершин, ребер, граней і зупиняємось, коли початок координат лежить в якомусь регіоні для ознаки (вершини, ребра, грані). Коли ми знайшли ознаку, точка мінімальної норми на цій ознаці може бути знайдена шляхом ортогональної проекції початку координат на цю ознаку.

Розглянемо випадок, коли $Q = \{Q_1, Q_2, Q_3, Q_4\}$. Довільна точка P (в тому числі початок координат) лежить в середині регіону Вороного до, наприклад, вершини Q_1 тоді і тільки тоді, коли задовольняються такі нерівності:

$$(P - Q_1) * (Q_2 - Q_1) \le 0,$$

$$(P - Q_1) * (Q_3 - Q_1) \le 0,$$

$$(P - Q_1) * (Q_4 - Q_1) \le 0.$$

P лежить в середині регіону Вороного, асоційованого з ребром Q_1Q_2 тоді і тільки тоді, коли виконуються такі нерівності:

$$(P - Q_1) * (Q_2 - Q_1) \ge 0,$$

$$(P - Q_2) * (Q_1 - Q_2) \ge 0,$$

$$(P-Q_1)*\big((Q_2-Q_1)\times\big((Q_2-Q_1)\times(Q_3-Q_1)\big)\big)\ge 0,$$

$$(P-Q_1)*\big(\big((Q_4-Q_1)\times(Q_2-Q_1)\big)\times(Q_2-Q_1)\big)\ge 0.$$

Якщо P не лежить в середині регіонів Вороного для вершин або ребер, перевіряються регіони для граней шляхом перевірки того, чи лежать P та точка з Q, що залишилась, на протилежних сторонах від площини, сформованої трьома іншими точками з Q. Наприклад, P лежить в регіоні Вороного для $Q_1Q_2Q_3$ тоді і тільки тоді, коли виконується нерівність:

$$\big((P-Q_1)*\big((Q_2-Q_1)\times(Q_3-Q_1)\big)\big)\big((Q_4-Q_1)\times\big((Q_2-Q_1)\times(Q_3-Q_1)\big)\big)<0$$

Аналогічно перевіряються й інші вершини, ребра та грані.

Модифікація алгоритму GJK

Таким чином, алгоритм GJK вказує на наявність перетину фігур. Проте він не дає нам точку перетину або точку в середині обох політопів. Цей недолік можна легко виправити. Для цього на кожному кроці нам необхідно зберігати не лише симплекс, відносно якого ми робимо перевірку, а й точки з A та B, які формують цей симплекс. Тоді точку перетину (або точку в середині обох політопів) можна знайти таким чином:

$$V=\lambda_0A_0+\lambda_1A_1+\lambda_2A_2+\lambda_3A_3=\lambda_0B_0+\lambda_1B_1+\lambda_2B_2+\lambda_3B_3,$$

де A_0, A_1, A_2, A_3 – вершини симплексу A;

B_0, B_1, B_2, B_3 – вершини симплексу B;

$\lambda_0, \lambda_1, \lambda_2, \lambda_3$ – невідомі коефіцієнти >0, такі, що $\lambda_0+\lambda_1+\lambda_2+\lambda_3=1$.

Віднімемо від одного рівняння інше і зазначимо, що $A_i-B_i=T_i$, де T_i – вершини результуючого симплексу, отриманого на виході алгоритму GJK. Після цього розкладемо отримане рівняння за координатами. Отримаємо таку систему рівнянь:

$$\begin{cases}\lambda_0T_{0x}+\lambda_1T_{1x}+\lambda_2T_{2x}+\lambda_3T_{3x}=0\\ \lambda_0T_{0y}+\lambda_1T_{1y}+\lambda_2T_{2y}+\lambda_3T_{3y}=0\\ \lambda_0T_{0z}+\lambda_1T_{1z}+\lambda_2T_{2z}+\lambda_3T_{3z}=0\\ \lambda_0+\lambda_1+\lambda_2+\lambda_3=1\end{cases}.$$

Розв’язуючи подану систему рівнянь, отримаємо коефіцієнти λ_i, за допомогою яких можемо знайти точку V. Ця точка буде належати обом симплексам A та B, а отже буде належати перетину політопів. Можливий випадок, коли в результаті роботи даного алгоритму отримана точка буде точкою перетину політопів. У такому випадку для того, щоб знайти точку, що лежить строго в середині обох політопів, нам потрібно змістити знайдену точку на мінімальне Δ в одному з 26 напрямків у просторі. Таким чином, ми гарантовано знайдемо точку, що належить обом політопам одночасно.

2.3. Побудова розв’язку задачі знаходження опуклої оболонки

Нехай у нас на вході множина точок S.

1) Будуємо тетраедр шляхом з’єднання перших чотирьох точок з S, що не лежать в одній площині.

2) Для кожної точки p, що залишились.

• Уявімо, що p – це точка видимості. Визначимо множину граней, видимих з p. Назвемо цю множину граней гранями, що видимі з p.

• Визначимо множину ребер горизонту для точки p. Ребро горизонту – ребро, що належить до видимої і невидимої граней одночасно. Ребра горизонту формують замкнений цикл навколо видимих граней.

• Для кожного ребра горизонту створюємо нову трикутну грань, з'єднуючи ребро з точкою p. Точка *p* тепер є частиною опуклої оболонки.

• Нарешті, видаляємо усі грані, що були видимі з p.

Цей алгоритм має складність $O(n^2)$. Проте його можна пришвидшити до $O(n\log(n))$, зберігаючи для кожної точки множину граней, які з неї видимі, та навпаки. Маємо дводольний граф. Назвемо його графом конфліктів.

Ініціюємо граф конфліктів після першого кроку, тестуючи кожну точку відносно чотирьох граней початкового тетраедра (якщо точка розташована перед гранню, то грань видима з точки). Для цього використовуємо вектори нормалі граней. Після додання кожної нової точки до опуклої оболонки на кроці 2c нові грані, що були створені, мають бути додані до графу конфліктів. У цьому випадку розглянемо ребро горизонту. До того, як ми додали грань f, були дві грані, які ділили це ребро. Назвемо їх g та h. Якщо з точки було видиме ребро горизонту, то з неї також були видимі або g, або h. Таким чином, щоб знайти множину точок, видимих з f, треба лише розглянути множину точок, видимих з g або h.

3. Обґрунтування складності

Твердження 1. Задачу перетину двох опуклих n- вершинних політопів можна розв'язати за час $O(n\log(n))$.

Доведення. Для знаходження перетину двох політопів потрібно знати точку, що буде лежати в середині обох політопів. Це можна зробити за $O(n)$, використовуючи алгоритм GJK. Після цього переміщення центра координат в цю точку та знаходження двоїстих точок до граней потребують $O(n)$ операцій. Найбільше часу витрачається операцією побудови опуклої оболонки $(O(n\log(n)))$. Побудова області перетину, яка є двоїстою фігурою до опуклої оболонки, потребує $O(n)$ часу. Таким чином, побудова області перетину двох опуклих політопів потребує $O(n\log(n))$ часу. ■

Твердження 2. Задачу перетину m опуклих n-вершинних політопів можна розв'язати за час $O(mn\log(n))$ та $O(n\log(n)\log(m))$ з використанням розпаралелення.

Доведення. Складність без використання розпаралелення очевидна. Якщо ми використовуємо паралельні обчислення, то можемо розбити усю множину політопів на пари і робити їх попарний перетин. Після цього в нас буде $m/2$ перетинів. На наступному кроці – $m/4$ і т.д. Усього потрібно буде зробити $\log(m)$ таких кроків, на кожному з яких кожен процесор буде виконувати $O(n\log(n))$ операцій. ■

4. Практична частина

Робота виконана на мові програмування C# з використанням технології WPF для наочного представлення результатів роботи алгоритму. Задання багатокутників для алгоритму мож-

ливе двома шляхами: читання вершин політопів з бази даних (використовується LINQ to SQL) та генерація політопів випадковим чином (кожен політоп генерується на випадково заданій множині точок). Користувачеві також доступні деякі маніпуляції зі зміною відображення отриманих політопів та їхнього перетину, а саме: вибір того, як має відображатися політоп (тільки вершини, тільки ребра, тільки грані); вибір прозорості та кольору передніх та бокових граней. Перетин політопів відбувається попарно, з використанням паралельних обчислень та пулу потоків, що дозволяє істотно підвищити швидкодію алгоритму за рахунок повного використання ресурсів комп'ютера.

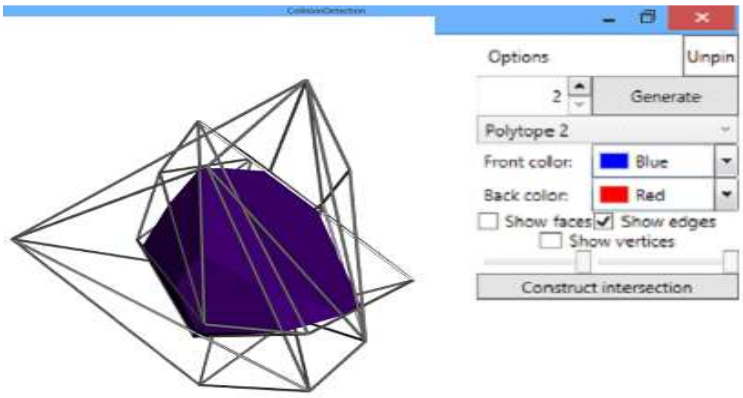

Рис. 2. Результат виконання програми та вікно налаштувань

Результат роботи програми та вікно налаштувань представлені на рис. 2.

5. Висновки

У роботі запропоновано відносно простий для реалізації метод для побудови перетину опуклих політопів, що базується на модифікованому алгоритмі GJK та властивостях двоїстості фігур. Цей алгоритм дозволяє отримати перетин двох опуклих політопів за час $O(nlongn)$, що є досить непоганим результатом порівняно з прямим методом відтинання одного політопа гранями іншого, який має складність $O(n^3)$.

Якщо ж політопів більше ніж 2, то їх перетин повинен виконуватись попарно. Взагалі, уся складність алгоритму полягає у знаходженні точки, що лежить в середині перетину усіх політопів. Якщо ця точка відома, то, скориставшись властивістю двоїстості, ми зможемо знайти перетин m політопів лише за допомогою побудови опуклої оболонки. Але приросту швидкості це не дасть, бо нам потрібно буде будувати опуклу оболонку для $m \cdot n$ точок, що дасть ту саму оцінку складності, але алгоритм буде в той же час погано розпаралелюватись.

СПИСОК ЛІТЕРАТУРИ

1. Mount D.M. Geometric Intersection / D.M. Mount // Handbook of Discrete and Computational Geometry / eds. J.E. Goodman, J. O'Rourke. – [2nd ed.]. – Chapman & Hall: CRC, Boca Raton, 2004. – P. 857 – 876.
2. Ericson C. Real time collision detection / Ericson C. – New York: Taylor & Francis Group, 2005. – 593 p.
3. Berden Van den G. Collision Detection in Interactive 3D Environments / Van den G. Berden. – Amsterdam; Boston: Morgan Kaufmann Publishers, 2004. – 277 p.
4. Berden Van den G. A Fast and Robust GJK Implementation for Collision Detection of Convex Objects / Van den G. Berden // Journal of Graphics Tools. – 1999. – Vol. 4 (2). – P. 7 – 25.
5. A new linear algorithm for intersecting convex polygons / J. O'Rourke, C.-B. Chien, T. Olson [et al.] // Comput. Graph. Image Process. – 1982. – N 19. – P. 384 – 391.
6. Gilbert E.G. A fast procedure for computing the distance between complex objects in three-dimensional space / E.G. Gilbert, D.W. Johnson, S.S. Keerthi // Robotics and Automation, IEEE Journal. – 1988. – Vol. 4 (2). – P. 193 – 203.
7. Space sweep solves intersection of two convex polyhedra elegantly / S. Hertel, K. Mehlhorn, M. Mäntylä [et al.] // Acta Informatica. – 1984. – N 21. – P. 501 – 519.

8. Препарата Ф. Вычислительная геометрия: Введение / Ф. Препарата, Г. Шеймос. – М.: Мир, 1989. – 478 с.
9. Chazelle B. An optimal algorithm for intersecting three-dimensional convex polyhedral / B. Chazelle // SIAM Journal on Computing. – 1992. – N 4 (21). – P. 671 – 696.
10. Dobkin D.P. Fast detection of polyhedral intersection / D.P. Dobkin, D.G. Kirkpatrick // Theoret. Comput. Sci. – 1983. – N 27. – P. 241 – 253.
11. O'Rourke J. Computation Geometry in C / O'Rourke J. – Cambridge University Press, Cambridge, 1994. – 1497 p.
12. An Optimal Algorithm for Finding Segment Intersections / I.J. Balaban // Proc. of the 11th ACM Symposium on Computational Geometry, 1995. – P. 211 – 219.
13. V-Clip: Fast and robust polyhedral collision detection / B. Mirtich // ACM Transactions on Graphics (TOG). – 1998. – N 17 (3). – P. 177 – 208.

Стаття надійшла до редакції 01.02.2013