DOI: 10.15587/2312-8372.2017.105505

ПРОГРАМУВАННЯ АЛГОРИТМУ ПОШУКУ ПРИНАЛЕЖНОСТІ ТОЧКИ БАГАТОКУТНИКУ І ВЗАЄМНОГО НЕПЕРЕТИНУ ФІГУР

Гаврилов Т. М.

1. Вступ

В наш час потреба в високоякісному та комфортному взутті стала в один ряд з іншими фізіологічними потребами людини. Та на ряду з якістю та комфортністю суттєво зросла потреба в кількості цього взуття. За даними статистики та досліджень потреба в кількості взуття за останні роки зросла до понад 13 млрд пар в рік.

Для задоволення такої зростаючої потреби взуттєва галузь легкої промисловості має вдосконалити методи раціоналізації схем розкрою. Проектуючи раціональну схему розкрою розкрійник має дотримуватись правила найщільнішого розміщення деталей, забезпечуючи при цьому їх взаємний неперетин. Для вирішення даного питання необхідно побудувати навколо деталей годографи вектор-функцій щільного розподілу фігур цих деталей.

2. Об'єкт дослідження та його технологічний аудит

Об'єктом дослідження є взуттєві деталі різної конфігурації, які виготовляються на підприємствах взуттєвої галузі легкої промисловості. Досліджувались різні варіанти розкрою, основною рисою яких являється те, що навколо деталей будуються годографи.

По суті, годограф – траєкторія руху полюса однієї фігури навколо іншої, зберігаючи умову їх взаємного дотикання [1–3]. Проілюструємо виконання даної умови на прикладі двох багатокутників (рис. 1).

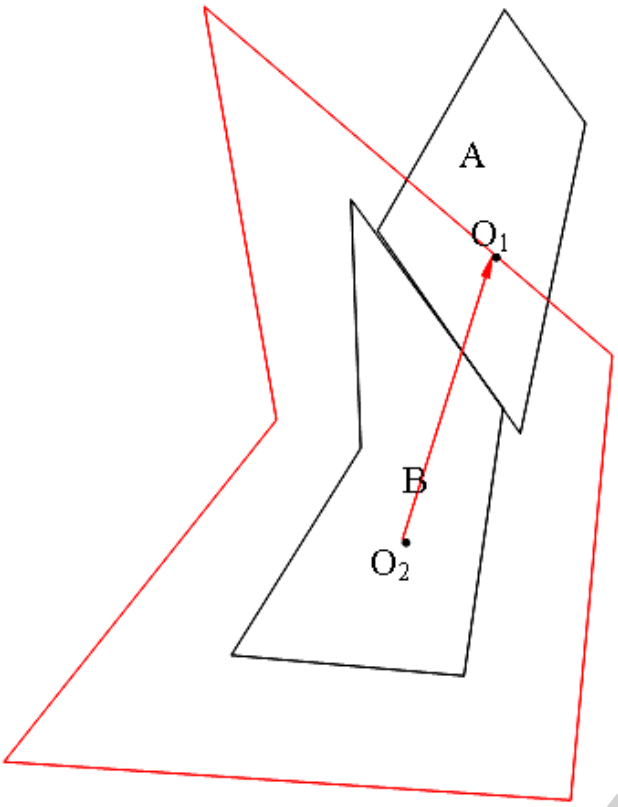


Рис. 1. Побудова годографа навколо фігури: A, B – фігури; O_1,O_2 – полюси фігур

Для інтерактивного вилучення деталей з розкрійної схеми необхідно вирішити задачу про належність точки багатокутнику.

Нехай маємо деякий абстрактний багатокутник A, заданий масивами координат точок d[i].x, d[i].y, де i=0..n–1, де n – кількість точок, та деяку точку C з координатами (x0,y0) (рис. 2). Необхідно довести належність точки фігурі.

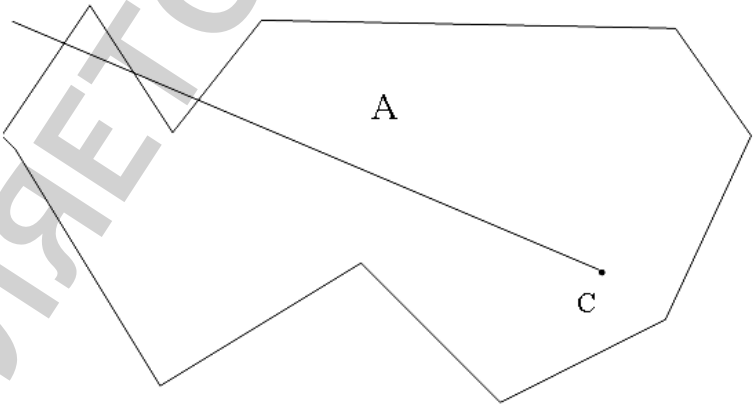


Рис 2. Задача про належність точки багатокутнику: A – багатокутник; C – точка

На рис. 1, 2 представлений опукло-ввігнутий багатокутник, проте суттєвим недоліком годографів являється саме те, що вони коректно будуються лише для опукло-ввігнутих фігур.

3. Мета та задачі дослідження

Метою дослідження є розробка програмного модуля вилучення деталей із схеми розкрою, які виготовляються на підприємствах взуттєвої галузі легкої промисловості. Даний програмний модуль, після розробки, стане частиною розроблюваного програмного забезпечення.

Для досягнення поставленої мети необхідно виконати такі задачі:

1. Проаналізувати алгоритми пошуку приналежності точки багатокутнику.
2. Створити алгоритм вилучення деталей.

4. Дослідження існуючих рішень проблеми

Задача локалізації точки має багато підвидів і розв'язуються вони різними методами. Серед існуючих рішень поставленої проблеми слід відзначити:

– метод трасування променя [4, 5];

– метод підсумовування кутів [6, 7];

– метод обходу Грєхема [8, 9].

Зокрема, робота [4] повністю присвячена проблемі методу трасування променя, його перевагам і недолікам, аналізу його складності та часу виконання.

Як відмічає автор в роботі 6], незважаючи на складність методу підсумовування кутів його спрощена форма дає непогані перспективи для його ширшого застосування.

Автор роботи [10] стверджує, що при побудові випуклого простого багатокутника час виконання алгоритмів можна скоротити до O(cn), де c – константа.

Проте, в даній статті досліджується проектування схем розкрою, а значить фокус проблеми дещо змінюється.

На думку авторів [1–3], для побудови раціональних схем розкрою необхідне щільне розміщення деталей, тобто треба створювати годограф. В той же час автори [1] зазначають, що суттєвим недоліком годографів являється те, що вони будуються лише для однотипних фігур.

5. Методи дослідження

Для розв’язання поставленої задачі існує багато алгоритмів. Проаналізуємо основні:

Сутність алгоритму пошуку приналежності точки багатокутнику по *методу обходу Грєхема* [4] – доволі проста. З однієї вершини до іншої проводять вектор $\vec{a}$, а до третьої – вектор $\vec{b}$. Напрям обходу від вектора $\vec{a}$ до вектора $\vec{b}$ і покаже вилучати деталь чи ні. Якщо обходити будемо за годинниковою стрілкою, то точка належить багатокутнику, а якщо – проти, то не належить. Проте, цей алгоритм не передбачає ситуації, коли точка лежить проти годинникової стрілки від вектора $\vec{a}$, але знаходиться в межах багатокутника.

Сутність методу кутів полягає в пошуку знаку суми кутів [7]. З точки, приналежність якої треба довести, проводяться промені до інших вершин багатокутника. Кути між променями додаються. По знаку цієї суми визначається

приналежність точки багатокутнику. Недоліки алгоритму полягають в складності підрахування кутів і невизначеному положенні точки відліку.

Алгоритм методу трасування променя шукає приналежність точки багатокутнику по кількості перетинів променем сторін багатокутника [6]. З деякої віддаленої точки до точки, приналежність якої треба довести, проводиться промінь, і кількість його перетинів границь багатокутника визначить приналежність точки багатокутнику. На рис. 3 показано три переваги методу променя. По-перше, для вирішення задачі підходить будь-який промінь, який проходить через задану точку. По-друге, не важливий порядок перетину сторін багатокутника, важлива лише парність їх загальної кількості. І, по-третє, що найголовніше, його буде значно легше реалізувати програмно.

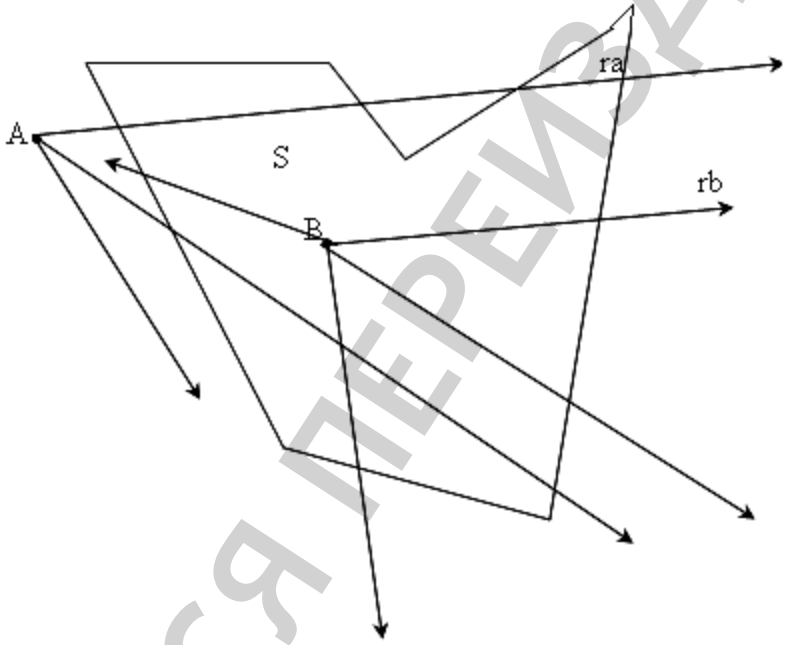


Рис. 3. Особливості методу променя: S – фігура; A, B – точки; ra, rb – промені

5.1. Побудова променя

Необхідно знайти кількість перетинів променя з ребрами багатокутника. Фізично, для того, щоб побудувати промінь (фактично, промінь – це відрізок прямої), треба дві точки. Першою буде точка вилучення. Другу точку виберемо в межах деталі, а для простоти – полюс деталі. Тоді рівняння променя запишеться у вигляді:

$$\frac{x-x_0}{xp-x_0}=\frac{y-y_0}{yp-y_0}=0\Rightarrow(yp-y_0)x+(x_0-xp)y+y_0xp-ypx_0=0, \qquad (1)$$

тобто $A_1x+B_1y+C_1$.

А рівняння ребра багатокутника – у вигляді:

$$\frac{x-d[i].x}{d[i+1].x-d[i].x}=\frac{y-d[i].y}{d[i+1].y-d[i].y}=0\Rightarrow$$
$$\Rightarrow\left(d[i+1].y-d[i].y\right)x+\left(d[i].x-d[i+1].x\right)y+ \quad (2)$$
$$+d[i].yd[i+1].x-d[i+1].yd[i].x=0,$$

тобто $A_2x+B_2y+C_2$.

Таким чином, координати точки перетину D будуть розв'язком системи рівнянь (1) і (2):

$$x=\frac{B_1C_2-C_1B_2}{A_1B_2-B_2A_1}$$
$$y=\frac{A_2C_1-C_2A_1}{A_1B_2-B_2A_1} \quad (3)$$

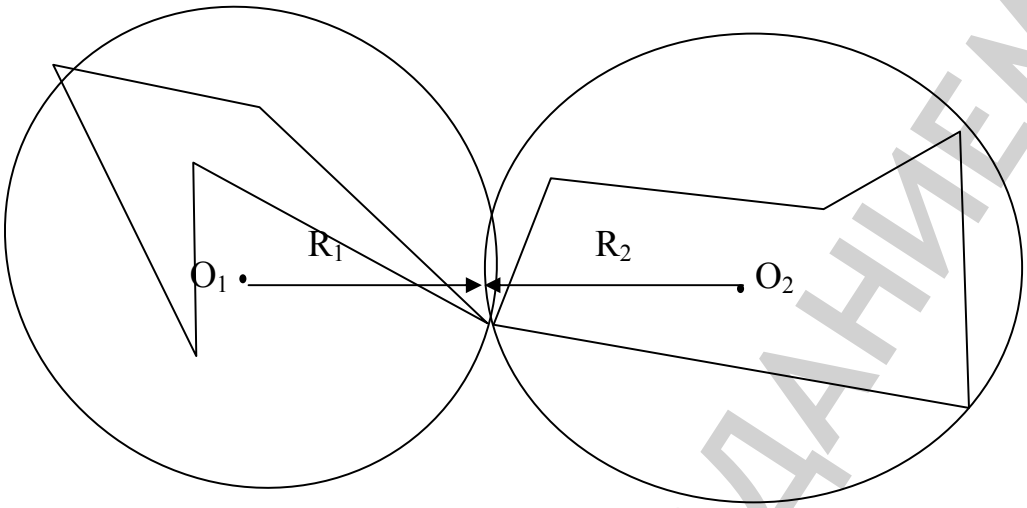
Отже, якщо координати точки D задовольнятимуть рівнянню (3), перетин є, але невідомо, належить точка багатокутнику чи ні. Для цього знаходимо таку точку D для кожного ребра багатокутника, та, в кожний раз інкрементуємо кількість перетинів. По загальній кількості перетинів дивимось, належить точка багатокутнику чи ні. Якщо кількість перетинів – непарна, то, як очевидно з рис. 2, точка С належить багатокутнику.

Взаємний неперетин деталей можливий при умові неперетину описаних навколо них кіл, тобто:

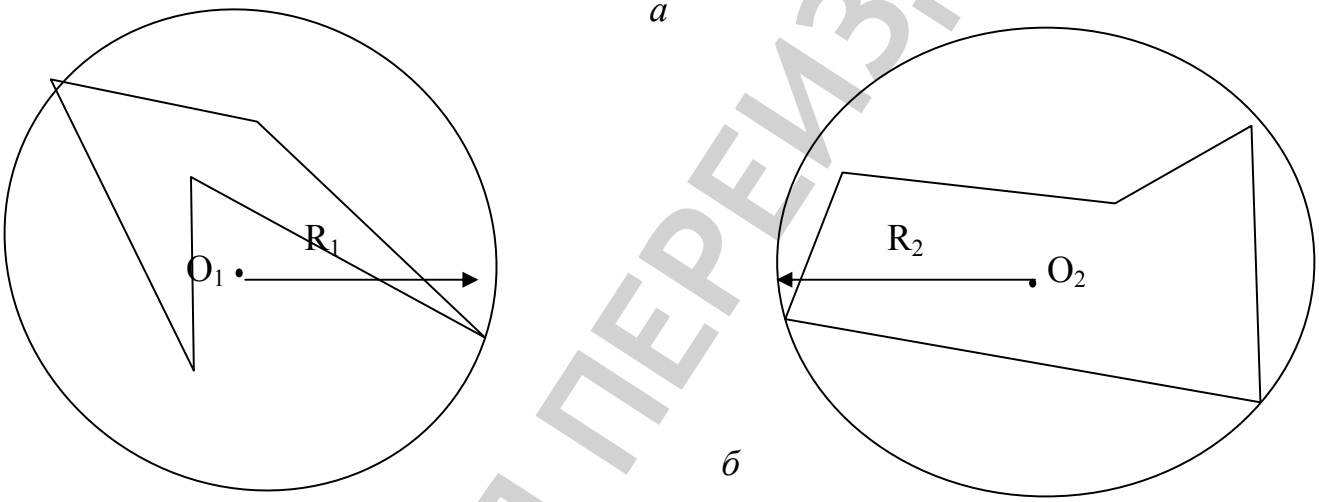
$R_1+R_2>=D,$ (4)

де R_1 та R_2 – радіуси кіл, описаних навколо вибраних деталей, а D – відстань між їхніми реальними полюсами.

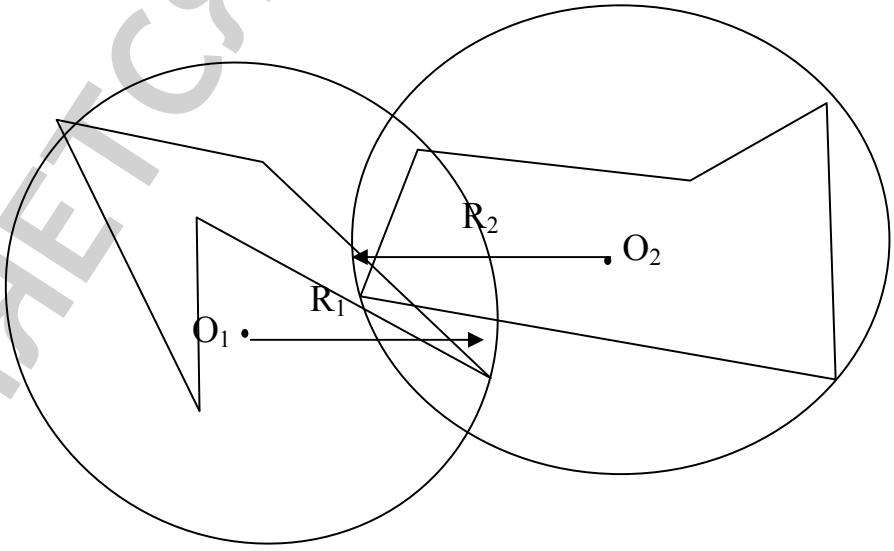
Проілюструємо виконання даної умови (рис. 4).



а



б



в

Рис. 4. Виконання умови неперетину: *а* – щільне розміщення; *б* – простий неперетин; *в* – перетин

На рис. 4, *а* показано ідеальний варіант, коли деталі лише доторкаються. На рис. 4, *б* показано варіант, коли деталі розміщені занадто далеко одна від одної. Проте, умова (4) виконується і, значить, таке розташування деталей задовольнятиме задачу про неперетин. На рис. 4, *в* показано варіант, коли деталі розміщені занадто близько і перетинаються.

Рис. 4 більш наглядно ілюструє критерій, за яким стає зрозумілим, які деталі слід вилучати зі схеми.

5.2. Алгоритм процедури

Як вже було відмічено, стандартний алгоритм методу трасування променя модифіковано згідно умов поставленої задачі. Підкреслимо, що ребро в даному алгоритмі виступає в якості прямої, а точки на промені будуть по одну або по іншу сторону від ребра. Таким чином, для вирішення завдання по вилученню деталей застосуємо вже описані формули:

$$D_1=A_1x_{di}+B_1y_{di}+C_1 \quad (5)$$
$$D_2=A_1x_{di+1}+B_1y_{di+1}+C_1 \quad (6)$$
$$D_3=A_2x_p+B_2y_p+C_2 \quad (7)$$
$$D_4=A_2x_{ex}+B_2y_{ex}+C_2 \quad (8)$$

де x_{di}, y_{di}, x_{di+1}, y_{di+1} – координати точок початку і кінця ребра;

x_p, y_p – координати полюсу деталі;

x_{ex}, y_{ex} – координати точки вилучення.

Отже, алгоритм виглядає наступним чином:

1) обнуляємо вміст лічильника перетинів;

2) проводимо пряму $A_1x+B_1y+C_1$ між точкою вилучення та полюсом деталі (1);

3) пряма $A_2x+B_2y+C_2$ знаходиться на ребрі багатокутника (2);

4) знаходимо значення D_1,D_2,D_3,D_4 по формулам (5)–(8);

5) якщо добутки $D_1*D_2<0$ та $D_3*D_4<0$, то перетин є і додаємо до вмісту лічильника перетинів одиницю та переходимо на крок 2, а інакше – пропускаємо крок;

6) повторюємо кроки з другого по п’ятий для кожного ребра багатокутника;

7) аналізуємо вміст лічильника перетинів та вилучаємо або не вилучаємо дану деталь.

5.3. Лістинг процедури

Дана процедура описує кроки з другого по п’ятий вищеописаного алгоритму.

```
{=========================================================}
function peretyn(x1,y1,x2,y2,x3,y3,x4,y4:integer): boolean;
var A1,B1,C1,A2,B2,C2,D1,D2,D3,D4:integer;
```

```
begin
peretyn:=false;
 A1:=y2-y1;
 B1:=x1-x2;
 C1:=x2*y1-x1*y2;
 A2:=y4-y3;
 B2:=x3-x4;
 C2:=x4*y3-x3*y4;
 if (A1*B2-A2*B1<>0) then begin
  D1:=A1*x3+B1*y3+C1;
  D2:=A1*x4+B1*y4+C1;
  D3:=A2*x1+B2*y1+C2;
  D4:=A2*x2+B2*y2+C2;
  if (D1*D2<0) and (D3*D4<0) then peretyn:=true;
                                end;
end;
{=========================================================}
```

Це лише програмний модуль, по якому, якщо peretyn=true, головна програма видалить деталь.

6. Результати дослідження

Процедура вилучення деталей входить до складу розробленої програми інтерактивного формування схем розкрою. Це програмне забезпечення реалізовувалось в програмному середовищі Delphi 7. Дана програма забезпечує інтерактивне додавання та вилучення деталей не тільки в вихідному положенні і повернутих на 180°, а й під довільним кутом. На рис. 5, б показано, як програма працює в цих двох режимах: додавання та вилучення деталей.

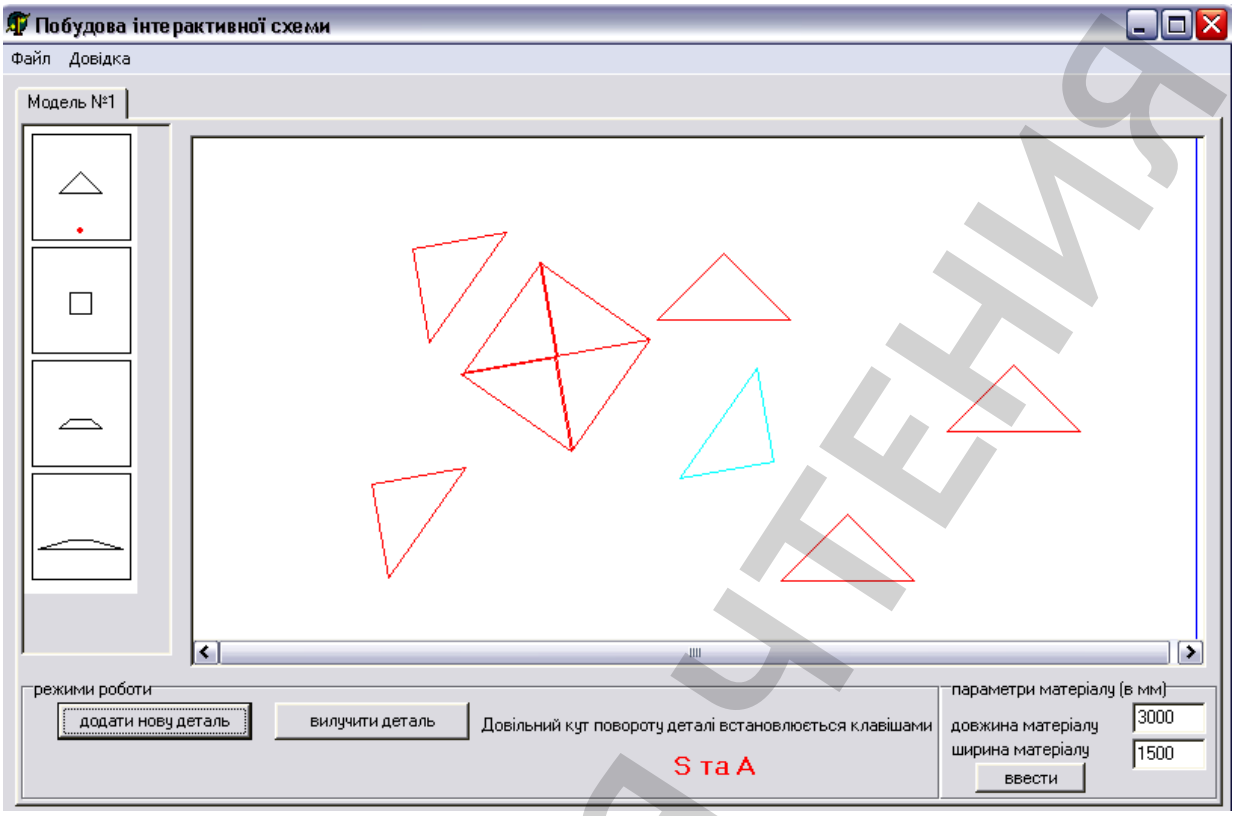


Рис. 5. Робота програми в режимі інтерактивного додавання деталей

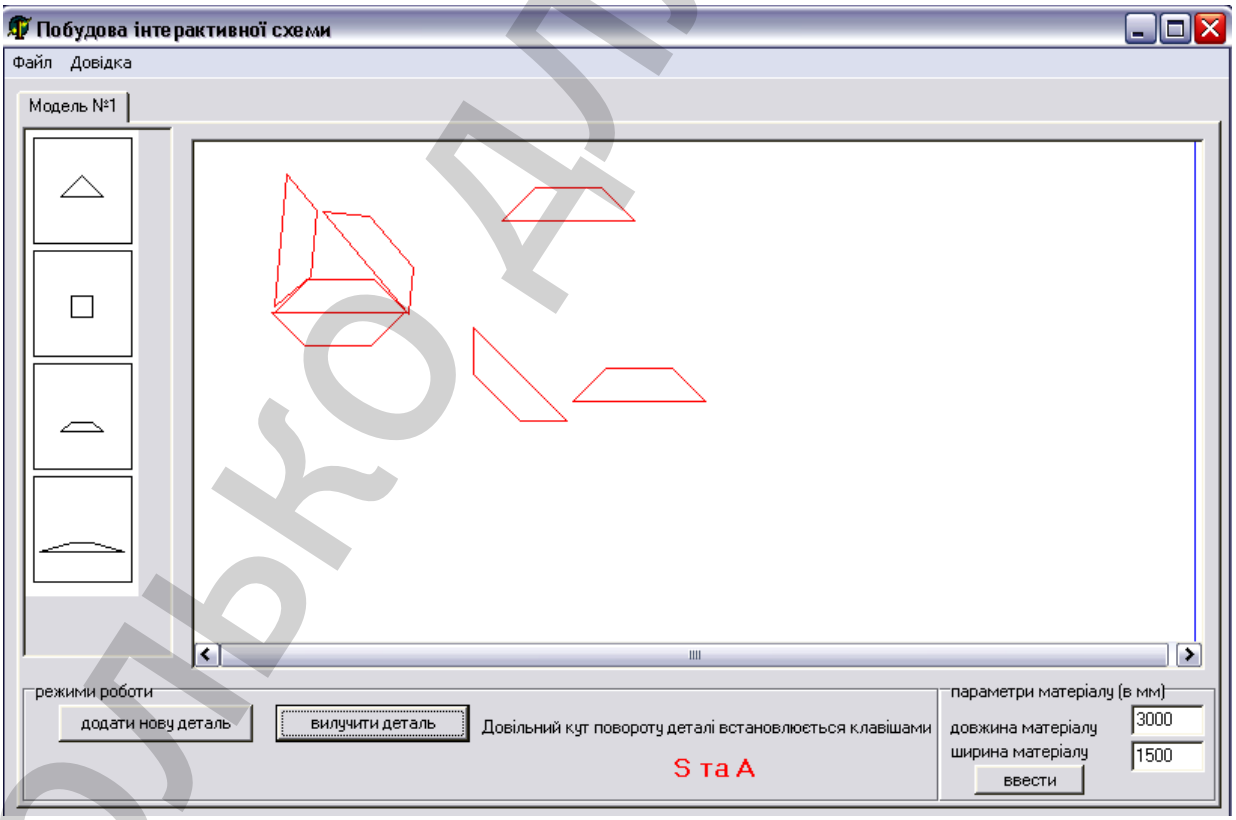


Рис. 6. Робота програми в режимі інтерактивного вилучення деталей

Рис. 5, 6 показують два суттєвих моменти:

1. Описані навколо деталей кола не дають їм перетинатися.
2. За обраним алгоритмом деталі вилучаються.

7. SWOT-аналіз результатів досліджень

Strengths. Вищеописаний алгоритм призначений для спрощення роботи розкрійника. Програмна реалізація алгоритму дозволить зменшити кількість трудових ресурсів та скоротити час виробництва схем розкрою.

Weaknesses. Якщо на вхід даного програмного модуля надійдуть некоректні дані, це може призвести до зупинки роботи програми, що, в свою чергу, призведе, як мінімум, до збільшення часу виробництва, а як максимум, до збільшення кількості трудових ресурсів.

Opportunities. Модернізація обладнання на підприємстві підвищить потужність обчислювальної техніки, а значить підвищення ефективності роботи програми, що, в свою чергу, призведе до ще більшого зменшення кількості трудових ресурсів та скорочення часу виробництва схем розкрою.

Threats. В роботі [6] запропоновано використання модифікованого методу підсумовування кутів. Цей метод має більшу ефективність та менший час виконання, однак на даний момент не існує розкрійного обладнання для роботи за цим методом.

8. Висновки

1. Проаналізовано три основних алгоритми пошуку приналежності точки багатокутнику. Алгоритмів, які вирішують дану задачу, існує набагато більше, проте дані алгоритми мають оптимальні параметри складності: O(2*N) проти O(N^2) у інших алгоритмів. Це означає, що даний алгоритм виконується швидше.
2. На основі аналізу трьох основних алгоритмів створено алгоритм вилучення деталі зі схеми розкрою. Даний алгоритм модифікований і підлаштований під запити поставленої задачі, тобто залишити по 4–6 деталей одної конфігурації в схемі розкрою.
3. На основі аналізу трьох основних алгоритмів створено алгоритм вилучення деталі зі схеми розкрою. Даний алгоритм модифікований і підлаштований під запити поставленої задачі, тобто залишити по 4–6 деталей одної конфігурації в схемі розкрою. В алгоритмі замість побудови окремої точки для променя використано полюс деталі.

Література

1. Haines, E. Point in Polygon Strategies [Text] / E. Haines // Graphics Gems. – Elsevier, 1994. – P. 24–46. doi:10.1016/b978-0-12-336156-1.50013-6
2. Weiler, K. An Incremental Angle Point in Polygon Test [Text] / K. Weiler // Graphics Gems. – Elsevier, 1994. – P. 16–23. doi:10.1016/b978-0-12-336156-1.50012-4
3. Foley, J. D. Computer Graphics: Principles and Practice [Text] / J. D. Foley, S. K. Feiner, J. F. Hughes, A. Van Dam. – Ed. 2. – Addison-Wesley, 1990. – 1200 p.
4. Har-Peled, S. Approximating the Maximum Overlap of Polygons under Translation [Text] / S. Har-Peled, S. Roy // Algorithmica. – 2016. – Vol. 78, № 1. – P. 147–165. doi:10.1007/s00453-016-0152-9

5. Landier, S. Boolean operations on arbitrary polygonal and polyhedral meshes [Text] / S. Landier // Computer-Aided Design. – 2017. – Vol. 85. – P. 138–153. doi:10.1016/j.cad.2016.07.013

6. Wang, Z.-J. Re2l: An efficient output-sensitive algorithm for computing Boolean operations on circular-arc polygons and its applications [Text] / Z.-J. Wang, X. Lin, M.-E. Fang, B. Yao, Y. Peng, H. Guan, M. Guo // Computer-Aided Design. – 2017. – Vol. 83. – P. 1–14. doi:10.1016/j.cad.2016.07.004

7. Chen, D. Z. Computing the Visibility Polygon of an Island in a Polygonal Domain [Text] / D. Z. Chen, H. Wang // Algorithmica. – 2015. – Vol. 77, № 1. – P. 40–64. doi:10.1007/s00453-015-0058-y

8. Чертенко, Л. П. Математичне задання контурів внутрішньої форми взуття [Текст] / Л. П. Чертенко, В. П. Коновал // Вісник КНУТД. – 2002. – № 1. – С. 15–19.

9. Залгаллер, В. А. Об одном необходимом признаке плотнейшего расположения фигур [Текст] / В. А. Залгаллер // УМН. – 1953. – Т. 8, № 4 (56). – С. 153–162.

10. Гаврилов, Т. М. Технологія підрахунку комплексного показника якості матеріалів для взуття за допомогою експертних оцінок [Текст] / Т. М. Гаврилов // Легка промисловість. – 2011. – № 1. – С. 27–29.