УДК 656.61

МАТЕМАТИЧНА МОДЕЛЬ ВИЗНАЧЕННЯ ЗОН ПЕРЕКРИТТЯ ПОЛІГОНІВ З РІЗНОЮ ПРІОРИТЕТНІСТЮ ІНФОРМАЦІЇ У ECDIS

Петровський А. В., *к.т.н., доцент кафедри судноводіння Херсонської державної морської академії, м. Херсон, Україна, e-mail: andreyanybody@gmail.com, ORCID: 0000-0002-3337-9577.*

Запропоновано метод аналітичного визначення контурів перетину двовимірних полігонів – Intersection Polygon Contour Analysis (IPCA), що включає алгоритм знаходження множини контурів перетину, які можуть бути як зовнішніми, так і вкладеними. Отримані результати можуть бути використані у навігаційних і геоінформаційних системах. У математичній моделі IPCA все формується через: аналітичну перевірку перетинів лінійних ребер, вставлення точок у ребра, послідовне збирання циклів через орієнтований граф, обробку вкладеності з позиційної аналітики. IPCA враховує вкладені контури, дірки, орієнтацію, цикли, працює з полігонами, дірками, можливе розширення на криві, але тоді потрібно вручну обробляти всі ребра. Алгоритм, побудований на основі моделі IPCA чітко і поетапно аналізує вкладеність, за допомогою аналітичних перевірок положення контурів один відносно одного. Метод IPCA найкраще підходить для задач, де потрібна висока точність, важливо враховувати структуру вкладеності або потрібен точний контроль над геометрією результату і має однозначні переваги, коли: потрібний повний контур перетину, не тільки площа чи факт перетину; є вкладені контури / дірки, і критично важливо зберегти топологію; можуть бути вироджені випадки і потрібно обробити їх точно.

Ключові слова: *перетин полігонів; Intersection Polygon Contour Analysis; ECDIS; SENC.*

DOI: 10.33815/2313-4763.2025.2.31.156-165

Вступ. Розвинення навігаційних інформаційних систем, Electronic Chart Display and Information System (ECDIS), надає все більше даних та інструментів для реалізації задач судноводія на навігаційному містку. Все це розширює можливості судноводія і підвищує адекватність прийняття рішень завдяки більшій повноті навігаційної інформації. Зростання кількості отриманої інформації з різних джерел і можливість відображення полігонів по координатах у повідомленнях одразу, одночасно ускладнює процес аналізу отриманої інформації судноводієм. Особливо це стосується полігонів даних, які побудовані в автоматизованому режимі після отримання такої інформації, а саме: з різною пріоритетністю приділення уваги до небезпеки інформації. Прикладом може стати отримання повідомлення від NavTex та AIO, якщо вони при відображенні на карті, мають полігони даних з перетинанням. Якщо перехід планується у досить жвавому трафіку, з наявністю обмеженого навігаційного простору і характеризується багатьма інформаційними накладаннями разом з тимчасовими об’єктами, автоматизація підсвічування спірних за пріоритетністю приділення уваги контурів/площ ручної коректури/AIO/NavTex поглибить ступінь пророоблення маршруту на навігаційні перешкоди.

Постановка проблеми. Накладання додаткової інформації на System Electronic Navigational Chart (SENC) звичайно є геометричними полігонами. Перетин геометричних фігур – це ключова задача у багатьох прикладних сферах: навігації, картографії, комп'ютерній графіці. Складність полягає в обробці полігонів та визначенні вкладених контурів для автоматизованого візуального виявлення загальних контурів після коректури навігаційних карт SENC/накладання полігонів даних з AIO/NavTex у ECDIS при попередній прокладці маршруту руху судна (рис. 1).

Необхідні вхідні умови реалізації такої задачі:

– невелика кількість і складність перетинів, оскільки всі площадні об’єкти при коректурі карти SENC мають в основному прості фігури (полігони, кола) з власним пріоритетом при оцінці навігаційної небезпеки;

– наявність внутрішніх вкладених контурів, які можуть бути відображені на карті від різних типів повідомлень;

– необхідність точного визначення загальних площ при перетинанні для визначення пріоритету такої ділянки у подальшому аналізі;

– немає значення, як швидко діє алгоритм, оскільки використання його планується у статичній обстановці, на кроці побудови попередньої прокладки.

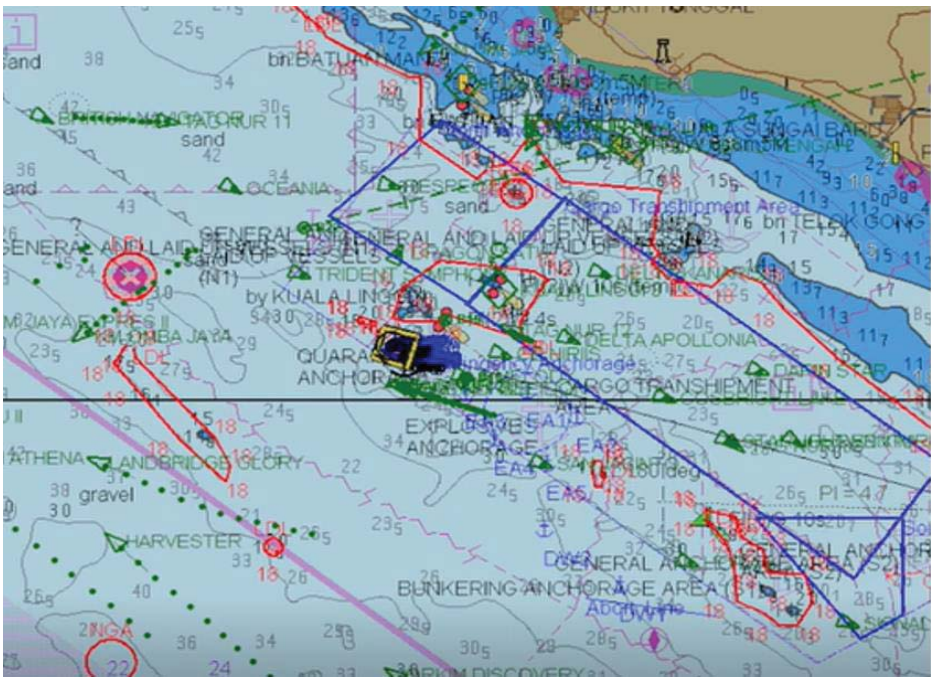

Рисунок 1 – Приклад Overlap навігаційної карти
(перетину полігонів синього кольору – результатом є трикутник)

Актуальність дослідження. Чинні системи ECDIS не надають автоматично загальних контурів при перетинанні полігонів з різним пріоритетом небезпеки після коректури карти SENC. Іноді, оператор ECDIS може взагалі не побачити наявність перетину, оскільки такі площі можуть візуалізуватися однаково при відсутності зміни налаштувань для різних типів повідомлень, які розглядаються штурманом при попередній прокладці. У такому випадку пріоритет небезпеки загальної зони перетину буде завжди відповідати останньому контуру, що в деяких умовах не є коректним. Автоматична візуалізація таких зон для оператора однозначно приверне увагу і надасть можливість правильно призначити пріоритет.

На цей час досить багато науковців світу проводили дослідження зі схожого напряму. У [1] досліджено перетин лінії з опуклим полігоном, використання проєктивної просторової репрезентації для підвищення точності. Однак залишається складність реалізації через проєктивну геометрію, модель орієнтована тільки на опуклі полігони, є непридатною для довільних, може бути повільніше за прості евклідові методи на малих наборах даних. Робота [2] надає модель, яка працює тільки для опуклих полігонів, присутній складніший математичний апарат підвищує поріг знань для soft-розробників. Дослідження [3] реалізують модель обчислення площі перетину довільних полігонів із великими/складними формами за допомогою Graphics Processing Unit (GPU) та методів апроксимації, що накладає вимоги до наявності у системі продуктивного GPU, крім того, апроксимації знижують точність. У [4] показано розширення алгоритму Greiner–Hormann, щоб обробляти вироджені перетини коректно, але має підвищену складність реалізації через обробку численних винятків, високі вимоги до числової стабільності, також бажаним є подвійна точність обчислень. Алгоритм для відсікання полігона прямокутним вікном, із параметричним представленням ребер представлено у [5]. Він жорстко прив’язаний до прямокутного вікна і непридатний для довільних контурів, а параметричний підхід може втрачати точність при дуже малих/дуже великих координатах. Робота [6] надає алгоритм, який реконструює полігони з контурів,

працює паралельно з GPU, з акцентом на збір сегментів і порівняння контурів внутрішніх/зовнішніх. Потребує GPU та відповідної мікропроцесорної інфраструктури, складно узгоджувати внутрішні та зовнішні контури у складних топологіях, паралельні алгоритми ускладнюють налаштування й тестування. У дослідженнях [7] описаний метод є спеціалізованим: працює лише для сіток однакової топології, і, використання методу Монте-Карло й растеризації, призводять до апроксимаційної похибки при складних межах, але головне – потребує потужного GPU, погана масштабованість на дуже великих моделях із високою роздільністю. Робота [8] демонструє метод для накладання двох квадратних сіток з однаковою топологією й оптимізує кількість випадків перетинів ребер. Присутні обмеження на топологію сіток, ускладнена обробка дірок, особливо при складних внутрішніх контурах. У роботі [9] надані дослідження з продуктивності просторової індексації для швидкого пошуку кандидатів на перетин полігонів. Запропонована індексація не дає точного результату, а лише список кандидатів, що потребує другої стадію перевірки. Представлений у [10] базовий алгоритм визначення належності точки та взаємної диз'юнктності фігур, теж має недоліки: не описана масштабованість для багатокутників великої складності, чутливість до вироджених випадків.

Таким чином багато наукових робіт не дають контур перетину як результат, а тільки площу або факт перетину, не у всіх є підтримка вкладених контурів або дірок.

Метою дослідження є розробка формальної математичної моделі для 2D полігонів із вкладеними контурами та алгоритму, який формує множину контурів перетину разом із вкладеними контурами. Об'єктом дослідження є процес визначення зон (контурів) перетину двовимірних полігонів із різною пріоритетністю в SENC.

Задачі дослідження:

– розробити формальну математичну модель для 2D полігонів із вкладеними контурами;

– створити алгоритм, що формує множину контурів перетину разом із вкладеними контурами;

– протестувати модель на прикладі 2D з метою подальшого застосування для векторних карт.

Методи дослідження: аналітичні методи (побудова й розв'язання систем рівнянь для точок перетину ребер); параметризація контурів полігонів; алгоритмічний аналіз графів (для збору сегментів у замкнені цикли); аналітичні перевірки положення контурів один відносно одного (приналежність/вкладеність).

Основна частина. Пропонується метод аналітичного визначення контурів перетину двовимірних полігонів – Intersection Polygon Contour Analysis (IPCA), що включає алгоритм знаходження множини контурів перетину, які можуть бути як зовнішніми, так і вкладеними.

Вхідними даними математичної моделі є полігон A з множиною замкнутих контурів:

$$A = \left\{C_A^{(0)}, C_A^{(1)}, \dots, C_A^{(k)}, \right\},$$

де $C_A^{(0)}$ – зовнішній контур (порядок обходу проти годинникової стрілки);

$C_A^{(i)}, i = 1..k$ – внутрішні контури (дірки, порядок обходу за годинниковою стрілкою);

і полігон B: $B = \left\{C_B^{(0)}, C_B^{(1)}, \dots, C_B^{(m)}, \right\}$.

Аналітичний опис контурів.

Кожен контур C параметризується функцією: $r(t) = \big(x(t), y(t)\big), t \in [0,1]$

Області, що відповідають полігонам з дірками:

$$W_A = \left\{x \in R^2 : x \in int\left(C_A^{(0)}\right) \text{ та } x \notin \bigcup_{i=1}^{k} int(C_A^{(i)})\right\};$$
$$W_B = \left\{x \in R^2 : x \in int\left(C_B^{(0)}\right) \text{ та } x \notin \bigcup_{i=1}^{m} int(C_B^{(j)})\right\}. \quad (1)$$

Кроки побудови контуру перетину:

1. Першим кроком є пошук точок перетину ребер.

Для кожної пари ребер:

$$e_{A,i}^{(p)}: r_{A,i}^{(p)}(t), t \in [t_1, t_2];$$
$$e_{B,j}^{(q)}: r_{B,j}^{(q)}(s), s \in [s_1, s_2],$$

розв'язують систему рівнянь:

$$r_{A,i}^{(p)}(t) = r_{B,j}^{(q)}(s); \qquad t \in [t_1, t_2], s \in [s_1, s_2], \tag{2}$$

що еквівалентно:

$$\begin{cases} x_{A,i}^{(p)}(t) = x_{B,j}^{(q)}(s); \\ y_{A,i}^{(p)}(t) = y_{B,j}^{(q)}(s). \end{cases} \tag{3}$$

2. Далі визначають приналежності точок ребрам полігонів.

Для будь-якої точки p, що належить ребру, перевіряється приналежність до W_A та аналогічно W_B:

$$p \in int\left(C_A^{(0)}\right) \text{ і } p \notin \bigcup_{i=1}^{k} int\left(C_A^{(i)}\right);$$
$$q \in int\left(C_B^{(0)}\right) \text{ і } q \notin \bigcup_{i=1}^{m} int\left(C_B^{(j)}\right). \tag{4}$$

3. Формування контурів перетину $R = A \cap B$.

Контур R складається з ребер e, для яких $\forall q \in e: q \in W_A \cap W_B$.

4. Обробка вкладених контурів.

Кожен контур $C_R^{(m)}$ визначається як цикл ребер, що складають замкнену ламану.

Орієнтація контуру задає його тип:

– проти годинникової стрілки (зовнішній контур);
– за годинниковою стрілкою (дірка – вкладений контур).

Визначення вкладеності базується на перевірці належності точок одного контуру до області обмеженої іншим.

Алгоритм побудови множини контурів перетину з вкладеними контурами.

1. Проведення параметризації ребер полігонів A і B.
2. Пошук точок перетину ребер шляхом розв'язання системи рівнянь.
3. Вставлення точок перетину у контури полігонів A і B – розбиття ребер.

У кожному контурі $C_A^{(i)}$ для ребра $e_{A,i}^{(k)}$точки перетину $P_{A,i}^{(k)} \subset e_{A,i}^{(k)}$впорядковуються по параметру τ. Ребро розбивається на сегменти:

$$S_{A,i}^{(k,r)} = \left\{ r_{A,i}^{(k)}(\tau): \tau \in [\tau_r, \tau_{r+1}] \right\}, \tag{5}$$

де τ_r – параметри точок перетину та кінців ребра з вершинами (0 і 1).

Аналогічно для ребер $e_{B,j}^{(m)}$.

4. Перевірка належності кожного сегменту полігонам A і B за допомогою прямо-променевого тесту.

Перевірка приналежності точки q до *int(C)* здійснюється через алгебраїчний прямо-променевий тест:

– надається прямий промінь з q у напрямку, наприклад, *x*.
– визначається кількість перетинів променю з ребрами контуру C.
– якщо кількість непарна, то $q \in int(C)$, інакше – зовні.

5. Вибір сегментів, що належать перетину $W_A \cap W_B$.

Обирають сегменти S таких ребер полігонів A і B, для яких $\forall p \in S: p \in W_A \cap W_B$.

6. Формування циклів контурів з послідовності сегментів утворених точками перетину.

Побудова множини контурів $R = C_R^{(m)}$ відбувається шляхом послідовного з'єднання сегментів кроку 5, враховуючи точки перетину як «вершини» нового графу.

Вершинами такого графу є точки:

$V = \{$всі вершини контурів A і B та точки перетину $P\}$.

Ребрами графу є сегменти, які належать перетину $W_A \cap W_B$.

Тоді ставиться задача – знайти всі цикли, як замкнені шляхи у графі $G = (V, E)$. Знайдені цикли відповідають контурам $C_R^{(m)}$.

7. Визначення орієнтації контурів і класифікація на зовнішні та вкладені.

Для кожного контуру $C_R^{(m)}$, заданого послідовністю вершин $\{V_1, V_2, \dots, V_n\}$ обчислюють орієнтацію через функцію:

$$orient\left(C_R^{(m)}\right) = \sum_{k=1}^{n} (x_k y_{k+1} - y_k x_{k+1}), \quad V_{n+1} = V_1 \,. \tag{6}$$

Якщо при цьому $orient\left(C_R^{(m)}\right) > 0$, орієнтація проти годинникової стрілки – зовнішній контур, якщо $orient\left(C_R^{(m)}\right) < 0$, то це є дірка – вкладений контур.

Для визначення вкладеності між контурами для кожного контуру $C_R^{(m)}$ і $C_R^{(n)}$, $m \neq n$, потрібно перевірити, чи виконується умова: $C_R^{(n)} \subset int\left(C_R^{(m)}\right)$, тестуючи, наприклад, одну вершину $V \in C_R^{(n)}$ на приналежність до $int\left(C_R^{(m)}\right)$.

8. Побудова множини результатних контурів з координатами вершин $R = \left\{C_R^{(m)} : m = 1, \dots, M\right\}$, де кожен контур $C_R^{(m)} = \left\{V_1^{(m)}, V_2^{(m)}, \dots, V_{n_m}^{(m)}, \right\}$ має таку послідовність координат вершин контуру.

Результати і аналіз використання. Реалізація запропонованого методу мовою програмування Python 3.12 у середовищі Spider 6.0 здійснена на площині (рис. 2).

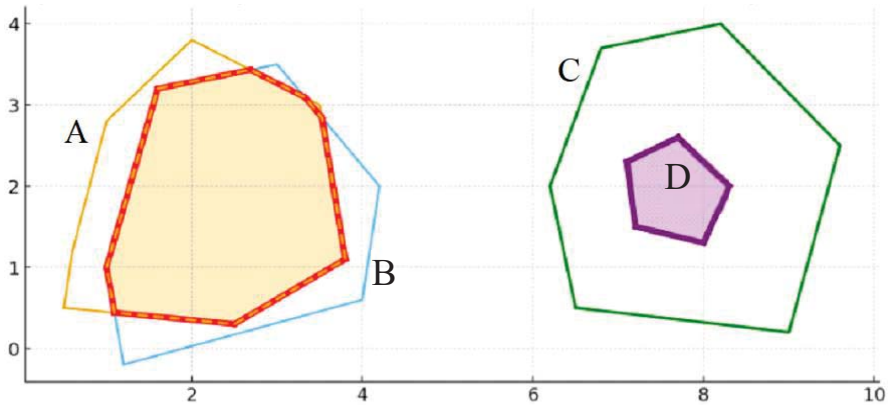

Рисунок 2 – Результат моделювання виділення загальних контурів і площ (червона лінія та фіолетова) при перетину полігонів A∩B із зовнішнім полігоном C і вкладеним полігоном D

Результати показали достатню ефективність запропонованої моделі. Подальше використання на навігаційних картах буде здійснено в рамках побудови загальної системи візуалізації окремого функціоналу ECDIS стосовно попередньої прокладки, для чого і створюються окремі модулі (відповідно до теми науково-дослідної роботи за №0125U003422) [11–14].

Розроблений метод IPCA має низку переваг і недоліків, залежно від контексту його застосування (навігаційні карти, ГІС, САПР, геометричне моделювання тощо).

Переваги методу IPCA:

1. Аналітична точність. Вся побудова базується на системах рівнянь, а не евристичних операціях. Немає втрати точності при роботі з лінійними сегментами – обчислення відбувається точно в координатній формі.

2. Придатність для вкладених структур. Модель повністю підтримує вкладені контури (дірки, багатоконтурні структури), на відміну від простих Boolean-операцій.

3. Контроль над кожним етапом. Контроль над кожним кроком: перетин, розбиття ребер, перевірка належності, формування циклів. Це зручно для налаштування, дослідження граничних випадків (наприклад, торкання ребер або точок), інтеграції з користувальницькою логікою.

4. Не потребує сітки або растеризації. На відміну від методів, які вимагають побудови піксельної сітки (растеризації), IPCA повністю векторний.

5. Гнучкість розширення. Метод легко адаптується для обробки:

– множин полігонів;
– полігонів із криволінійними краями (наприклад, через сплайни);
– інших геометричних фігур (кола, дуги тощо).

Недоліки методу IPCA:

1. Обчислювальна складність. При великій кількості вершин (особливо з багатьма точками перетину) обсяг обчислень різко зростає.

2. Потребує точного чисельного обчислення. У випадку дуже близьких або вирівняних точок (наприклад, майже паралельні відрізки) накопичення похибок може викликати артефакти – потрібні механізми округлення / епсилон-пороги.

3. Неінтуїтивність реалізації. На відміну від бібліотек типу Clipper або GEOS (наприклад для реалізації мовою програмування Python), реалізація вимагає повного контролю над:

– розбиттям ребер;
– побудовою графу;
– циклічною перевіркою вкладеності.

4. Необхідність роботи з топологією. Потрібно правильно впорядковувати вершини, уникати самоперетинів, дубльованих точок і т. д. Помилка в побудові графа → некоректний контур → порушення геометрії.

5. Порівняно важкий старт. Для простих задач (наприклад, перевірка перетину двох квадратів) IPCA може виявитися "надто складним" – немає сенсу, якщо не потрібна деталізація результату.

Аналіз теоретичної результативності зведений до таблиці 1.

Таблиця 1 – Порівняння IPCA з наявними дослідженнями

Критерій	***IPCA***	***Наявні підходи***
1	2	3
Аналітичність / точність	Дуже висока: системи рівнянь, точно відрізки, точки перетину, контури, вкладеність	Результати деяких досліджень (наприклад, “degenerate intersections”, “projective space”) прагнуть підвищити точність або обробку кордонів, але часто з обмеженнями (опуклі полігони, лінія полігон, або апроксимації)
Підтримка вкладених контурів / дірок	Так, IPCA враховує вкладені контури, дірки, орієнтацію, цикли	У багатьох дослідженнях розглядаються полігони із дірками, але рідше дуже деталізовано цикли та вкладеність у результаті перетину (особливо при вироджених перетинах)

Продовження таблиці 1

1	2	3
Обробка вироджених випадків	IPCA може бути адаптований до вироджених випадків (точки/краї, що дотикаються тощо)	Так. «Clipping simple polygons with degenerate intersections» та «projective space representation»
Складність / ефективність	IPCA може бути значно повільніший, якщо вхід великий і багато перетинів; алгоритмічна складність висока для великих полігонів	Методи GPU прискорення, O(log N) алгоритми (для опуклих випадків), спеціалізовані бібліотеки часто оптимізовані для практичних задач
Універсальність типів об'єктів	Полігони, з дірками, можливо розширення на криві; треба вручну обробляти всі ребра	Багато методів обмежені: опуклі полігони, лінія полігон, або просто полігон прямокутне вікно, або апроксимації

Метод IPCA найкраще підходить для задач, де:
– потрібна висока точність;
– важливо враховувати структуру вкладеності;
– потрібен точний контроль над геометрією результату.

Не рекомендується застосовувати IPCA там, де достатньо приблизної побудови або потрібна висока швидкодія без деталізації (наприклад, у real-time графіці без дірок). Разом з тим вказане обмеження може з часом бути не суттєвим залежно від задач і умов використання. Наприклад, визначення точного контуру враження при накладанні контуру враження від боєзапасу БПЛА на ймовірний контур маневрування піхоти (який теж можливо будувати полігоном залежно від наявних укрить). При цьому швидкодія є головною для штучного інтелекту БПЛА, а точність визначення контуру враження у режимі реального часу не є обов'язковою, сенсу використання розробленого методу немає. Але, якщо будуть використовуватися швидкі елементи мікропроцесорної техніки: процесор, пам'ять, шина даних, то звісно, користі від отримання точного контуру буде більше, оскільки буде можливість, як подовження розробленого методу, здійснити моделювання різних позицій скиду заряду БПЛА для отримання найбільш точного контуру враження при можливих маневрах цілей.

Таким чином, метод IPCA має однозначні переваги, коли: потрібний повний контур перетину, не тільки площа чи факт перетину; є вкладені контури / дірки, і критично важливо зберегти топологію; можуть бути вироджені випадки і потрібно обробити їх точно.

Висновки. Запропонована модель IPCA дозволяє ефективно аналітичним чином знаходити множини контурів перетину двовимірних полігонів з урахуванням вкладених контурів. Метод базується на параметризації, а також на тестах приналежності для побудови конкретних контурів.

Перспективи подальших досліджень. Сучасні роботи по перетинах і булевим операціям підкреслюють тренд до поєднання *точності* і *масштабованості* – наприклад, 3D-EPUG-Overlay [15] демонструє, що exact-алгоритми (Simulation of Simplicity, multi-precision rational arithmetic) можна зробити паралельними для великих задач з гарантіями коректності, хоча це й ускладнює реалізацію. Паралельно розвиваються підходи, які дають інтерактивну робастність при виконанні булевих операцій на великих наборах даних [16]. З іншого боку, GPU-растерні підходи дають велику швидкість у задачах обчислення площ та попередньої фільтрації кандидатів, але вимагають спеціалізованої інфраструктури і додаткових кроків для відновлення топологічно коректних векторних контурів [17]. Сучасні «фільтри предикатів» дозволяють зменшити навантаження на exact-арифметику, зберігаючи

надійність у більшості випадків [18]. Ці напрямки показують можливі шляхи розвитку IPCA: інтегрувати адаптивні фільтри та/або елементи паралельної індексації для кращої масштабованості при збереженні аналітичної точності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Skala, V. Robust line-convex polygon intersection computation in E^2 using projective space representation. Machine Graphics & Vision, 32(3/4), 3–16. https://doi.org/10.22630/MGV.2023.32.3.1.

2. Scala V. A new fully projective O(log N) point-in-convex polygon algorithm: a new strategy, The Visual Computer 41(7):4839-4850, 2024, https://doi.org/10.1007/s00371-024-03693-9.

3. Yi Gao, Bo Wu, Jianxin Luo, Hangping Qiu, GPU-based Arbitrary Polygon Intersection Area Algorithm, DEStech Transactions on Engineering and Technology Research, 2017, https://doi.org/10.12783/dtetr/ismii2017/16652.

4. Erich L Foster, Kai Hormann, Romeo Traian Popa, Clipping simple polygons with degenerate intersections, Computers & Graphics: X, Volume 2, 2019, 100007, ISSN 2590-1486, https://doi.org/10.1016/j.cagx.2019.100007.

5. Sushil Chandra Dimri, Umesh Kumar Tiwari, Mangey Ram, An efficient algorithm to clip a 2D-polygon against a rectangular clip window, Applied Mathematics, Volume 37, pages 147–158, 2022, https://doi.org/10.1007/s11766-022-4556-0.

6. Ji R, Niu Z, Chen L. GPU-Accelerated Algorithm for Polygon Reconstruction. Applied Sciences. 2025; 15(3):1111. https://doi.org/10.3390/app15031111.

7. Xihua Xu, Shengxin Zhu, Symmetric Sweeping Algorithms for Overlaps of Quadrilateral Meshes of the Same Connectivity, Lecture Notes in Computer Science In book: Computational Science – ICCS, 2018, pp.61-75, https://doi.org/10.1007/978-3-319-93713-7_5.

8. Molano R., Sancho J. C., Ávila M.M., Rodríguez P. G., Caro A. Obtaining the user-defined polygons inside a closed contour with holes, Science and Business Media LLC, The Visual Computer, Volume 40, Springer Science and Business Media LLC, 2024, pages 6369–6387, https://doi.org/10.1007/s00371-023-03170-9.

9. Zhou, Chen & Li, Manchun. Performance evaluation of spatial indexing to identify polygon intersection. Geocarto International. Volume 35. pp.1–18. https://doi.org/10.1080/10106049.2019.1624987.

10. Gavrilov, T. Programming of the search algorithm for point belonging to the polygon and the mutual non-intersection of the figures. Technology Audit and Production Reserves, 3(3(35), 29–32. https://doi.org/10.15587/2312-8372.2017.105505.

11. Петровський А. В., Авторське свідоцтво № 134311 Комп'ютерна програма «WeatherCheckRoute» 12.03.2025.

12. Петровський А. В., Авторське свідоцтво № 133972 Комп'ютерна програма «S57ViewerVisualCatzocPointEnc» 03.03.2025.

13. Петровський А. В., Авторське свідоцтво № 135551 Комп'ютерна програма «WaveCheckAnalyze» 29.04.2025.

14. Петровський А. В., Авторське свідоцтво № 135550 Комп'ютерна програма «SimulationBridgeMasterE» 29.04.2025.

15. Magalhães S. V. G., Franklin W. R., Andrade M. V. A. An efficient and exact parallel algorithm for intersecting large 3-D triangular meshes using arithmetic filters. Computer-Aided Design. 2019. Vol. 111. 102801. https://doi.org/10.1016/j.cad.2019.102801.

16. Cherchi G., Pellacini F., Attene M., Livesu M. Interactive and Robust Mesh Booleans. ACM Transactions on Graphics. 2022. https://doi.org/10.1145/3550454.3555460.

17. Gao Y., Luo J. et al. A GPU-Based Rasterization Algorithm for Boolean Operations on Polygons. IEICE Transactions on Information and Systems. 2017. https://doi.org/10.1587/transinf.2017EDL8119.

18. Bartels T., Fisikopoulos V., Weiser M. Fast floating-point filters for robust predicates. Bit Numer Math. 2023. Vol. 63. P. 31. https://doi.org/10.1007/s10543-023-00975-x.

REFERENCES

1. Skala, V. (2023). Robust line-convex polygon intersection computation in E^2 using projective space representation. Machine Graphics & Vision. Vol. 32, No. 3/4. P. 3–16. https://doi.org/10.22630/MGV.2023.32.3.1.

2. Scala, V. (2024). A new fully projective O(log N) point-in-convex polygon algorithm: a new strategy. The Visual Computer. Vol. 41, No. 7. P. 4839–4850. https://doi.org/10.1007/s00371-024-03693-9.

3. Gao, Y., Wu, B., Luo, J., Qiu, H. (2017). GPU-based Arbitrary Polygon Intersection Area Algorithm. DEStech Transactions on Engineering and Technology Research. https://doi.org/10.12783/dtetr/ismii2017/16652.

4. Foster, E. L., Hormann, K., Popa, R. T. (2019). Clipping simple polygons with degenerate intersections. Computers & Graphics: X. Vol. 2. 100007. https://doi.org/10.1016/j.cagx.2019.100007.

5. Dimri, S. C., Tiwari, U. K., Ram, M. (2022). An efficient algorithm to clip a 2D-polygon against a rectangular clip window. Applied Mathematics. Vol. 37. P. 147–158. https://doi.org/10.1007/s11766-022-4556-0.

6. Ji, R., Niu, Z., Chen, L. (2025). GPU-Accelerated Algorithm for Polygon Reconstruction. Applied Sciences. 2025. Vol. 15, No. 3. 1111. https://doi.org/10.3390/app15031111.

7. Xu, X., Zhu, S. (2018). Symmetric Sweeping Algorithms for Overlaps of Quadrilateral Meshes of the Same Connectivity. In: Computational Science – ICCS. P. 61–75. https://doi.org/10.1007/978-3-319-93713-7_5.

8. Molano, R., Sancho, J. C., Ávila, M. M., Rodríguez, P. G., Caro, A. (2024). Obtaining the user-defined polygons inside a closed contour with holes. The Visual Computer. 2024. Vol. 40. P. 6369–6387. https://doi.org/10.1007/s00371-023-03170-9.

9. Zhou, C., Li, M. (2019). Performance evaluation of spatial indexing to identify polygon intersection. Geocarto International. 2019. Vol. 35. P. 1–18. https://doi.org/10.1080/10106049.2019.1624987.

10. Gavrilov T. (2017). Programming of the search algorithm for point belonging to the polygon and the mutual non-intersection of the figures. Technology Audit and Production Reserves. 3(3(35)). P. 29–32. https://doi.org/10.15587/2312-8372.2017.105505.

11. Petrovskyi, A. V. (2025). Avtorske svidotstvo №134311 Komp′yuterna programa «WeatherCheckRoute». 12.03.2025.

12. Petrovskyi, A. V. (2025). Avtorske svidotstvo №133972 Komp′yuterna programa «S57ViewerVisualCatzocPointEnc». 03.03.2025.

13. Petrovskyi, A. V. (2025). Avtorske svidotstvo №135551 Komp′yuterna programa «WaveCheckAnalyze». 29.04.2025.

14. Petrovskyi, A. V. (2025). Avtorske svidotstvo №135550 Komp′yuterna programa «SimulationBridgeMasterE». 29.04.2025.

15. Magalhães, S. V. G., Franklin, W. R., Andrade, M. V. A. (2019). An efficient and exact parallel algorithm for intersecting large 3-D triangular meshes using arithmetic filters. Computer-Aided Design. 2019. Vol. 111. 102801. https://doi.org/10.1016/j.cad.2019.102801.

16. Cherchi, G., Pellacini, F., Attene, M., Livesu, M. (2022). Interactive and Robust Mesh Booleans. ACM Transactions on Graphics. 2022. https://doi.org/10.1145/3550454.3555460.

17. Gao, Y., Luo, J. et al. (2017). A GPU-Based Rasterization Algorithm for Boolean Operations on Polygons. IEICE Transactions on Information and Systems. https://doi.org/10.1587/transinf.2017EDL8119.

18. Bartels, T., Fisikopoulos, V., Weiser, M. (2023). Fast floating-point filters for robust predicates. Bit Numer Math. Vol. 63. P. 31. https://doi.org/10.1007/s10543-023-00975-x.

Petrovskyi A. MATHEMATICAL MODEL FOR DETERMINING OVERLAPPING ZONES OF POLYGONS WITH DIFFERENT INFORMATION PRIORITIES IN ECDIS

This paper introduces Intersection Polygon Contour Analysis (IPCA), an analytical method for detecting and accurately reconstructing the intersection contours of two-dimensional polygons with different information priorities in Electronic Chart Display and Information Systems (ECDIS). As the volume of navigational data from diverse sources such as NavTex and Admiralty Information Overlay (AIO) grows, ship navigators increasingly face overlapping hazard polygons that may differ in priority. For safe voyage planning, it is not enough to know that polygons overlap—the exact geometry of their shared boundaries must also be defined. The proposed IPCA approach is grounded in analytical geometry and delivers precise coordinates for every intersection vertex. Each polygon is first represented as a set of closed contours: external boundaries oriented counter-clockwise and internal holes oriented clockwise. Polygon edges are parameterized, and systems of linear equations are solved to find all intersection points. These points are inserted into the original contours, subdividing edges into ordered segments. Using an oriented-graph framework, valid segments are assembled into closed cycles that capture every intersection contour, including nested holes. Orientation tests then distinguish exterior boundaries from interior voids, thereby preserving full topological accuracy. Unlike many existing methods that rely on rasterization, convexity assumptions, or GPU-based approximations, IPCA provides exact vector geometry and maintains complete nesting relationships, enabling accurate hazard-priority assessment in ECDIS. A prototype Python implementation demonstrates the method's effectiveness on representative navigation scenarios, where execution speed is sufficient for static pre-voyage planning. IPCA thus offers a robust solution whenever high analytical precision and faithful topological reconstruction are essential. Beyond maritime navigation, it can be applied to geographic information systems, computer-aided design, and other domains that require accurate, detailed intersection contours for complex polygonal data.

Key words: *polygon intersection; Intersection Polygon Contour Analysis; ECDIS; SENC.*

Статтю прийнято до редакції 16.11.2025