УДК 004.932 DOI: 10.31673/2412-4338.2024.022439

Крючкова Лариса Петрівна
Київський столичний університет імені Бориса Грінченка, Київ
ORCID 0000-0002-8509-6659

Башкевич Єгор Леонідович
Київський столичний університет імені Бориса Грінченка, Київ
ORCID 0000-0002-8509-6659

Куцибала Михайло Ігорович
Державний університет інформаційно-комунікаційних технологій, Київ
ORCID 0009-0002-4817-1653

ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ВИОКРЕМЛЕННЯ КОНТУРІВ ЗОБРАЖЕНЬ ОБ'ЄКТІВ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ ВІДЕОСПОСТЕРЕЖЕННЯ

Анотація. *Інтенсивний розвиток засобів прийому та передачі цифрових зображень створює проблему обробки величезних обсягів потоків відеоінформації. Існує широкий спектр завдань, в яких зображення розглядаються як джерело інформації, на основі якого приймається рішення. Важливі завдання, які вирішують інтелектуальні системи відеоспостереження: ідентифікація об'єктів і визначення їх траєкторій; вимірювання швидкості об'єктів; виявлення тривожних подій у завданнях охорони об'єктів-територій в режимі реального часу. Однією з основних операцій в інтелектуальних системах відеоспостереження при обробці зображень для подальшого аналізу є виділення контурів зображень об'єктів, оскільки контур містить всю необхідну інформацію для розпізнавання об'єктів за їх формою. Такий підхід дозволяє не враховувати внутрішні точки зображення і, таким чином, значно скоротити обсяг інформації, що обробляється. Це дає можливість аналізувати зображення в реальному часі. Контурний аналіз – це сукупність методів виділення, опису та обробки контурів зображення, що дозволяє описувати, зберігати, порівнювати та шукати об'єкти, представлені у вигляді їх зовнішніх контурів, а також ефективно вирішувати основні завдання розпізнавання образів – перенесення. , повертати та масштабувати зображення об'єкта. У цьому випадку контур означає просторово-довжинний розрив, різницю або різку зміну значень яскравості. У публікації розглянуто основні методи аналізу контурів, наведено програмну реалізацію алгоритмів виділення контурів зображень об'єктів в інтелектуальних системах відеоспостереження. Перспективним напрямком подальших досліджень є синтез алгоритмів виділення контурів зображень об'єктів, які реалізують двомасштабну статистичну модель зображення.*

Ключові слова: *контури зображень об'єктів, контурний аналіз, інтелектуальні системи відеоспостереження, виявлення об'єктів, відстеження об'єктів, цифрове оброблення зображень.*

Larysa Kriuchkova
Borys Grinchenko Kyiv Metropolitan University, Kyiv
ORCID 0000-0002-8509-6659

Yehor Bashkevich
Borys Grinchenko Kyiv Metropolitan University, Kyiv
ORCID 0000-0002-8509-6659

Mykhailo Kutsybala
State university of information and communication technologies, Kyiv
ORCID 0009-0002-4817-1653

SOFTWARE IMPLEMENTATION OF ALGORITHMS FOR SELECTING THE OUTLINES OF OBJECTS IMAGES IN INTELLECTUAL VIDEO SURVEILLANCE SYSTEMS

Abstract*. The rapid advancement in digital image acquisition and transmission technologies poses challenges in processing vast volumes of video information streams. Images serve as crucial sources of decision-making information across a wide range of tasks. Key objectives addressed by intelligent video surveillance systems include object identification, trajectory determination, object speed measurement, and real-time detection of critical events for securing premises. Among the primary operations in intelligent video surveillance systems is the extraction of object contours from images, as contours contain essential information for object recognition based on their shape. This approach excludes internal image points, significantly reducing the amount of processed information and enabling real-time image analysis. Contour analysis encompasses methods for selecting, describing, and processing image contours to describe, store, compare, and search for objects based on their external contours. This methodology effectively addresses fundamental challenges in pattern recognition, such as transferring, rotating, and scaling object images. The paper discusses prominent contour analysis methods and presents software implementations of algorithms for extracting object outlines in intelligent video surveillance systems. A promising future direction involves synthesizing algorithms that extract object contours using a two-scale statistical image model.*

Keywords: *object image contours, contour analysis, intelligent video surveillance systems, object detection, object tracking, digital image processing.*

1. Вступ

У теперішньому світі системи відеоспостереження мають одне з головних значень для сучасного життя. Швидкий розвиток технологій зумовлює створення все більшої кількості цифрових зображень, які у свою чергу являються важливим джерелом інформації, а також вагомим чинником для ухвалення конкретних рішень. Обробка цих даних стає складним завданням через те, що інтенсивний розвиток цифрових камер та інших пристроїв збільшує кількість відеоінформації, що постійно створюється. Робота операторів систем відеоспостереження пов'язана з монотонністю та тривалим спостереженням за відеопотоком. Оператор – це людина, тому він стомлюється, а наслідком цього є знижена концентрація уваги. Ці фактори чинять негативний вплив на ефективність їхньої роботи. Якщо людина безперервно стежить відеопотоком інформації, то після 12 хв вона починає пропускати 45% потенційно тривожних подій, а зі збільшенням часу спостереження до 22 хв відсоток пропуску зростає до 95% [1], що є достатньо суттєвою проблемою. Впровадження інтелектуальних систем відеоспостереження допоможуть автоматично виявляти події, що становлять важливе значення, і сповіщати про них операторів. Особливо ідентифікація об’єктів на цифрових зображеннях, які є значущим завданням для цих систем.

Давайте розглянемо дві важливі групи, на які можна поділити розпізнавання об'єктів на зображеннях [2]:

а) розпізнавання або класифікація зображень;

б) пошук і розпізнавання об'єктів (конкретних локальних ділянок) на зображеннях.

Для першої групи завдань передбачається розпізнавання або класифікація всього зображення, яке належить до одного з декількох класів. Рішення для завдання розпізнавання полягає в створенні відображення з умовним номером класу для кожного зображення.

Якщо розглядати другу групу завдань, то до неї належить процес розпізнавання у сфері обробки зображень. Він спрямований на виявлення геометричних об'єктів на всій області, яку ми розглядаємо. Також варто зазначити, що у такому випадку об'єкти мають малі розміри та зазвичай з'являються на обмежених ділянках зображення. У більшості випадків немає інформації про присутність об’єктів на зображенні. У загальному ми отримаємо результат характеристик та інших властивостей об'єктів у площині зображення.

Прикладом другої категорії є завдання об'єктно-територіального захисту, яке включає виявлення тривожних подій в реальному часі за допомогою відеокамер. Ці завдання є

складнішими в математичному і обчислювальному плані через невизначеність характеристик об'єктів.

Побудова опису зображення з використанням ознак є скданим завданням для будь-якої системи розпізнавання візуальної інформації.

Метою публікації є програмна реалізація алгоритмів виокремлення контурів зображень об’єктів в інтелектуальних системах відеоспостереження.

2. Виклад основного матеріалу

Контур є однією з головних характеристик будь-якого об'єкта на зображенні. Найчастіше за все він містить важливу інформацію для розпізнавання. Цей факт зумовив розвиток цілого напрямку в обробці та аналізі зображень - контурний аналіз. Він об'єднує методи виділення, опису, перетворення та аналізу контурів. Такий підхід дозволяє зробити систему, яка за допомогою скорочення кількості обчислень, працює у режимі реального часу [3].

Контурний аналіз об'єднує техніки виділення, характеризування та маніпулювання контурами на зображеннях. Ці методи дозволяють описати, зберегти, порівняти та знайти об'єкти, представлені через їхні зовнішні лінії-контурні обриси, і ефективно вирішувати ключові завдання розпізнавання образів, наприклад: переміщення, обертання та масштабування зображень об'єктів. Контур - це просторовий розрив, перепад або раптова зміна яскравості, який повністю характеризує форму зображення, а також володіє усією важливою інформацією для того, щоб ідентифікувати зображення на основі їх форми. Цей метод дозволяє ігнорувати внутрішні точки зображення, що значно зменшує кількість інформації, яку потрібно обробляти під час аналізу, тим самим забезпечуючи можливість проведення аналізу зображень у режимі реального часу.

Методи контурного аналізу

Методи контурного аналізу стають усе більш актуальними у сучасному етапі розвитку технологій. Вони часто застосовуються для розпізнавання тексту, які ми використовуємо у повсякденному житті. Також у системах розпізнавання об'єктів на відео, що знаходять застосування в області безпеки та в автомобільних автопілотах. Виділення контурів об'єктів на зображеннях дозволяє ефективно розпізнати їх за формою, зменшуючи при цьому обсяг опрацьовуваної інформації та забезпечуючи можливість для аналізу в реальному часі.

Методи контурного аналізу полягають у пошуку раптових змін яскравості або одномірних областей на зображенні. Серед визначених методик виділення контурів є метод змії, метод Кенні, а також фільтрація на основі операторів Собеля, Лапласа, Первітта [4-5]. Основою їх роботи є переходи яскравості. Цілісний контур отримуємо шляхом додаткової обробки, наприклад, застосуванням алгоритмів морфологічного аналізу.

Відомі алгоритми для виділення областей включають порогову сегментацію, кластеризацію, нарощування областей, алгоритм вододілу, блочну сегментацію та інші [4-5]. Ці алгоритми базуються на об'єднанні пікселів у однорідні області на основі певного закону однорідності або ознаки. Після їх використання ми отримуємо набір однорідних областей. Далі для отримання зв'язного контуру об'єкта ми використовуємо алгоритм проходження контуром. Проходження контуром — це послідовний перебір пікселів цифрового зображення за певними правилами з метою знаходження зв'язного контуру об'єкта.

Метод активних контурів

Метод активних контурів є важливим інструментом, що використовується для виділення меж, контурів і сегментації зображень. Він базується на використанні кривих мінімальної енергії, або змії, для виявлення контурів на зображенні. Процес методу полягає в тому, що спочатку визначається контур як проста лінія, яка потім деформується для створення області об'єкта. Кожна точка контуру намагається знаходитися на кордоні об'єкта, мінімізуючи енергію контуру, що обчислюється за допомогою функцій внутрішньої і зовнішньої енергії.

Енергія кожної точки v_i може бути розрахована за формулою:

$$E_i = \alpha E_{int}(v_i) + \beta E_{ext}(v_i), \quad (1)$$

де α, β – константи, які забезпечують відносну корекцію енергії; $E_{int}(v_i)$ – функція внутрішньої енергії, залежна від форми контуру; $E_{ext}(v_i)$ – функція зовнішньої енергії, залежна від властивостей зображення і типу градієнту в околиці точки v_i.

E_i, $E_{int}(v_i)$, $E_{ext}(v_i)$ – це величини, які є квадратними матрицями. Для позначки v_i енергія контуру відповідає значенню кожної матриці в центрі.

Кожна вершина v_i потенційно може перейти в будь-яку точку $v_i^{/}$, відповідну мінімальній енергії E_i.

Цей метод має такі недоліки:

1. Алгоритм не впорається коректно з задачею сегментації, якщо об’єкти без чітких меж або з неоднорідною площею та гладкими градієнтами.
2. Також, зміни у нормалі дотичної можуть призвести до злиття точок і створення грубого контуру, який може значно відрізнятися від фактичних меж об'єкта.
3. При автоматизованому застосуванні методу, складність полягає у виборі коефіцієнтів α і β, які коригуються оператором, а також у визначенні функцій енергії в межах та поза контуром.

З використанням цього методу розроблено різноманітні алгоритми для виділення контурів об'єктів різної природи. Наприклад, швидкий алгоритм активних контурів, запропонований Donna J. Williams і Mubarak Shah [6], дозволяє виділяти контури різних об'єктів, але вимагає прямої участі оператора у виборі коефіцієнтів. Компанія Biomedical Imagine Group (BIG) представила спосіб моделювання контурів біологічних об'єктів за допомогою методу змійки (E-Snake) та експоненційних сплайнів, що є найбільш підходящим для морфологічного аналізу клітин.

Метод деформованих шаблонів

Метод деформованих шаблонів розроблено для адаптації моделей об'єктів до вхідних зображень на основі їх внутрішньої енергії. Він передбачає використання вихідної моделі об'єкта та вхідного зображення, яке створює силове поле для деформації моделі. Ці деформовані моделі можуть змінювати свою форму залежно від характеристик кожної точки, що робить метод схожим на активні контури, проте більш ефективним у виявленні об'єктів з різноманітною структурою через використання різних типів енергій.

Наприклад, у якості деформованої моделі може слугувати "модель змії"[8], яка визначається сплайном. Контрольні точки цього сплайна переміщуються на поверхні зображення, а енергія визначається як зважена сума внутрішньої та зовнішньої енергій (схожий вигляд має формула (1)).

Детектор кордонів Кенні

Метод Кенні для виявлення країв, розроблений Джоном Канні у 1986 році, був спрямований на створення оптимального фільтра, що задовольняє критерії чіткості виділення країв, точності їх локалізації та мінімізації помилкових меж [9]. Цей детектор повинен забезпечувати високу чутливість до реальних меж, при цьому ігноруючи шум та інші помилкові сигнали, а також точно визначати положення контору і реагувати на кожен край лише один раз, уникаючи помилкового виявлення широких переходів яскравості як низки окремих країв.

Основні етапи алгоритму Кенні:

1. Згладжування: застосування Гауссового фільтра для усунення шуму.
2. Виявлення градієнтів: визначення країв у місцях з максимальним значенням градієнта.
3. Придушення не максимумів: відзначення лише тих країв, що є локальними максимумами.
4. Подвійна порогова фільтрація: встановлення потенційних країв за допомогою двох порогових значень.
5. Трасування областей неоднозначності: формування остаточних країв шляхом з'єднання слабких країв з сильними.

Ключовою особливістю алгоритму є використання принципу Non-Maximum Suppression, який полягає у визначенні країв як пікселів, що досягають локального максимуму градієнта у напрямку самого градієнта. Незважаючи на те, що детектор Канні був розроблений на ранніх етапах розвитку комп'ютерного зору, він досі вважається одним з найефективніших методів виявлення країв.

Згідно з дослідженнями порівняльного аналізу методів виділення контурів [10], метод Кенні може мати нижчу швидкість виявлення країв, якщо порівнювати з іншими методами, він показує кращі результати за критеріями точності та надійності, заснованими на середньоквадратичному відхиленні та піковому відношенні сигналу до шуму.

Відстеження контурів

Відстеження контурів має ключове значення для ефективної роботи алгоритмів, що працюють з контурними лініями зображень.

Визначальними для зупинки цих алгоритмів є спеціальні критерії [11]:

1. Алгоритм припиняє роботу після повернення до початкової точки задану n-разів. Цей метод ефективний, коли відомо, скільки разів алгоритм має повернутися до стартової точки.

2. Критерій Джакобса передбачає зупинку алгоритму після двох візитів до стартової точки. Це спрощений варіант попереднього критерію, який вирізняється високою продуктивністю при роботі з простими контурами та легкістю у впровадженні, але має обмеження у виборі початкової точки.

3. Інший критерій зупиняє алгоритм, коли він досягає точки, яка вже була визначена як частина контуру. Цей метод дозволяє вибрати будь-яку стартову точку, але вимагає маркування пройдених точок контуру та може бути неефективним при ідентифікації виступів контуру товщиною в один піксель.

Розглянемо кілька відомих алгоритмів відстеження контурів:

Алгоритм "жука" (Square Tracing Algorithm) [12]:

Цей метод полягає у послідовному обході контуру між об'єктом і фоном. Він використовує два простих правила:

1. Якщо активний піксель належить однорідній області (значення пікселя = 1), алгоритм робить поворот ліворуч.

2. Якщо активний піксель не належить однорідній області (значення пікселя = 0), алгоритм робить поворот праворуч.

Робота алгоритму завершується, коли він повертається до стартового пікселя.

Цей метод дозволяє ефективно відстежувати контури об'єктів на зображенні за допомогою простих локальних перевірок пікселів і правил руху.

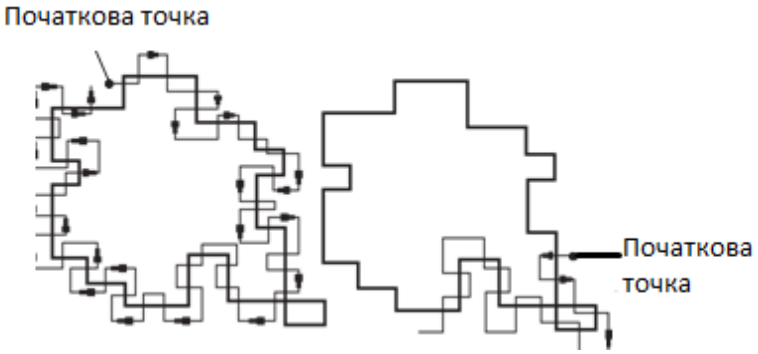


Рис.1. Ілюстрація методу простежування контурів [12]

2. Алгоритм «Moore-Neighbor Tracing» [12]. Метод використовується для пошуку контурів шляхом послідовної перевірки всіх сусідніх пікселів. Пошук наступного контурного пікселя розпочинається з пікселя, з якого відбувся перехід на активний піксель, і завершується при поверненні алгоритму в стартовий піксель.

3. «Redial Sweep» [14]. Метод є модифікацією попереднього алгоритму. Основна відмінність полягає в тому, що пошук наступного контурного пікселя починається з пікселя,

визнаного контурним на попередньому кроці алгоритму, а не з пікселя, з якого було зроблено перехід на активний піксель.

4. Алгоритм «Theo Pavlidi's Algorithm» [15] базується на використанні групи з трьох пікселів для визначення наступного контурного пікселя. Перевірка здійснюється згідно з такими правилами: спочатку перевіряється перший піксель P1 за годинниковою стрілкою (рис. 1.1); якщо він не є контурним, перевіряється другий піксель P2; якщо обидва не є контурними, перевіряється третій піксель P3. Якщо жоден з пікселів не є контурним, алгоритм здійснює поворот на 90 градусів за годинниковою стрілкою. Рух в іншому випадку здійснюється згідно зі схемою поворотів (рис. 1.1).

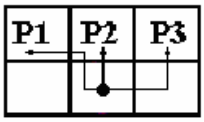


Рис. 1.1 Приклад групи з трьох точок для алгоритму «Theo Pavlidi's Algorithm»

Ці алгоритми широко використовуються для проходження контуру на зображеннях і мають реалізації в різних бібліотеках програмування.

Кластеризація

Кластеризація - це процес групування первинних елементів (в даному випадку пікселі) у декілька взаємовиключних категорій на основі їхніх схожих характеристик. Елементи в межах одного кластера мають схожі властивості, тоді як елементи з різних кластерів відрізняються помітно. Для простих зображень використовуються координати пікселів як вихідні дані, а для складніших, наприклад напівтонових зображень, беруться до уваги тривимірні вектори, що включають координати та градації сірого.

Однак, методи кластеризації часто не враховують точне просторове розташування елементів, або роблять це непрямо, через характеристики, такі як координати. Тому, після кластеризації, зазвичай потрібно виконати додаткову обробку для визначення зв'язкових компонентів зображення. Кластеризація може бути неефективною для зашумлених зображень, ведучи до втрати деталей та створення численних малих областей. Для великих зображень, які потребують розділення на багато класів, кластерний аналіз вимагає значних обчислювальних ресурсів. Один з способів зменшення обчислювального навантаження полягає в застосуванні методів зниження розмірності, наприклад, методу головних компонент (PCA). Втім, варто враховувати, що PCA також може збільшити обчислювальну складність.

Кластеризацію найближчого сусіда використовують для оцінки швидкості та відстані.Якщо показати цей процес математично, то він буде виглядати наступним чином: $\{p_1,...p_m\}\in S_1$, $\{q_1...q_n\}\in S_2$ - позначили двома лініями, але водночас вони мають задовольняти нерівності (1.2, 1.3, 1.4):

$$\left\{\left|x_p - x_q\right| + \left|y_p - y_q\right|\right\} \le \alpha_d \quad (1.2)$$

$$\left\{\left|u_{p_i} - u_{q_j}\right|\right\} \le \alpha_u \quad (1.3)$$

$$\left\{\left|v_{p_i} - v_{q_j}\right|\right\} \le \alpha_v \quad (1.4)$$

де α_d , α_u , α_v – порогові константи; u_p , v_p – складові швидкості точки p в координатах (x_p, y_p);

Алгоритми фільтрації на основі операторів

Просторова фільтрація включає переміщення спеціальної ковзкої маски по зображенню та обчислення градієнту яскравості в кожній точці зображення. Ковзка маска представляє собою квадратну матрицю коефіцієнтів, які відповідають певній групі пікселів на зображенні. Однотонні області на зображенні матимуть низький градієнт яскравості і відображатимуться як темні області на вихідному зображенні, а області з високим градієнтом матимуть яскравіші лінії на вихідному зображенні.

Відгук *R* фільтрації в точці (*x, y*) розраховується перед обрахунком градієнта за формулою (1.5):

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + ... + w(0,0)f(x,y) + ... + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \quad (1.5)$$

На значення пікселів, до яких ми застосовуємо маски, є сума, що обчислюється як добуток коефіцієнтів маски.

Для визначення градієнта яскравості зображення потрібно використовувати дискретні аналоги похідних першого і другого порядку. Першу похідну *f(x)* обчислюємо як різницю між значеннями сусідніх пікселів, а другу похідну — як різницю між значеннями сусідніх значень першої похідної.

У випадку двох змінних *(x, y)* необхідно обчислити часткові похідні по кожній з просторових осей, оскільки піксель зображення має дві координати. Для обчислення першої похідної зображення ми використовуємо дискретні наближення двовимірного градієнта. Градієнт зображення представляє собою вектор, що обчислюється за формулою (1.6):

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} f \\ \frac{\partial}{\partial y} f \end{bmatrix} \quad (1.6)$$

Використовуючи методи контурного аналізу, треба розуміти важливість модуля вектора градієнта функції:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2} \quad (1.7)$$

Знаходячи контури, однією з найважливіших характеристик є напрямок вектора градієнта:

$$a(x,y) = \operatorname{arctg}\left(\frac{G_y}{G_x}\right) \quad (1.8)$$

Обробка зображення буде відбуватись за допомогою маски, яку вибирають після обчислення відгуку фільтрації.ьРозглянемо маски на основі операторів Превітт, Кірша, Собеля, Шарра та Лапласа [16-20].

Оператор Превітт. Він використовує маску 3×3 для просторової фільтрації:

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \;.\; G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad (1.9)$$

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad (1.10)$$

Ці функції працюють шляхом визначення різниці між сумами значень по верхній та нижній рядках в області 3×3, що наближає похідну по вісі x. Різниця між сумами значень у першому та останньому стовпцях відображає похідну по вісі y. Ці функції реалізуються за допомогою маски фільтрації по вісі x та y.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}, \; G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \quad (1.11)$$

Оператор Собеля використовує маску 3×3 для просторової фільтрації.

Особливістю є те, що він використовує ваговий коефіцієнт для значень середніх елементів:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (1.12)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (1.13)$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (1.14)$$

Недостатком оператора Собеля є те,що він не демонструє доброї обертової симетрії.

Оператор Шарра. Ганно Шарр покращив оператора Собеля та знизив ефект від недостатків. У його матрицях вага центральних пікселів перевершує вагу крайніх пікселів у 3,3 раза:

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}, G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix} \quad (1.16)$$

Оператор Кірша використовує коефіцієнти для всіх значень, але не для середнього. Маска оператора Кірша:

$$G_x = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}; G_y = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & 3 & -3 \end{bmatrix} \quad (1.17)$$

Оператор Лапласа для виділення контурів є розширенням векторного оператора Лапласа. Він використовує однакову маску для просторової фільтрації як для осі *x*, так і для осі *y*:

$$G_{x,y} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad (1.18)$$

3. Програмна реалізація

Було розглянуто достатню кількість популярних алгоритмів виділення контурів зображень об'єктів. Для програмної реалізації було вибрано алгоритм Кенні та оператор Шарра. Також було вирішено порівняти їх виконання.

Метод Кенні, як було вище зазначено, має нижчу швидкість виявлення країв, але за цих обставин він показує кращі результати за критеріями точності та надійності.

Оператор Шарра вважається оптимальним за витратами ресурсів та якістю виділення.

4. Опис програми

Програмне забезпечення було розроблено мовою програмування Python, версія Python: 3.12. Середовище розробки - Visual Studio Code, яке забезпечує зручне та продуктивне програмування на мові Python.

Програма ContourDetectionApp є інтерактивним інструментом для детекції контурів на зображеннях за допомогою двох алгоритмів: Scharr та Canny. Вона використовує бібліотеку OpenCV для обробки зображень та PyQt5 для створення графічного інтерфейсу користувача.

5. Графічний інтерфейс

Головне вікно програми містить елементи керування для завантаження зображення, вибору алгоритму Canny та Scharr, а також відображення оригінального та обробленого зображень (Рис.2-2.1):

–original_label: Місце для відображення оригінального зображення.

–load_button: Кнопка "Load Image" для вибору та завантаження зображення.

–scharr_button: Кнопка "Scharr Operator" для застосування алгоритму Scharr до завантаженого зображення.

–canny_button: Кнопка "Canny Algorithm" для застосування алгоритму Canny до завантаженого зображення.

–processed_label: Місце для відображення обробленого зображення.

6. Структура коду та функціональність

Алгоритм Кенні (Рис.2)

Клас ContourDetectionApp**:**

1. Функція __init__**:** ініціалізує вікно, розміщення та мітки для відображення зображень.
2. Функція load_image**: в**ідкриває діалогове вікно вибору файлу, завантажує вибране зображення, відображає його в мітці "Оригінальне зображення" (масштабує до ширини 400 пікселів).
3. Функція detect_canny**:**

–Читає завантажене зображення за допомогою cv2.imread з OpenCV.

–Перетворює зображення у відтінки сірого за допомогою cv2.cvtColor.

–Застосовує алгоритм Кенні з стандартними порогами (50 і 150) за допомогою cv2.Canny.

–Перетворює відтінки сірого назад в кольорове зображення за допомогою cv2.cvtColor.

–Створює QPixmap з обробленого зображення для відображення.

4. Функція get_pixmap**:** перетворює зображення OpenCV в QPixmap для відображення в PyQt5 GUI.
5. Функція show_processed_image**:** відображає оброблене зображення у новому вікні.
6. Функція show_error_message: відображає повідомлення про помилку.

Оператор Шарра (Рис.2.1)

Клас ContourDetectionApp**:**

1. Функція __init__**: і**ніціалізує вікно, розміщення та мітки для відображення зображень.
2. Функція load_image**:** завантажує та відображає оригінальне зображення.
3. Функція detect_scharr**:**

– Читає зображення у відтінках сірого.

– Застосовує оператори Scharr по X та Y осях.

– Розраховує величину градієнта.

– Перетворює результат в формат RGB.

4. Функція get_pixmap**:** функція конвертації обробленого зображення у формат для відображення.
5. Функція show_processed_image**:** відображає оброблене зображення у новому вікні.
6. Функція show_error_message**:** відображає повідомлення про помилку.

4. Використання програми

–Користувач запускає програму, вибравши файл **ContourDetectionApp.py**.

–Натискає на кнопку "Load Image" для вибору та завантаження зображення (рис.2.3).

–Обирає один з алгоритмів: "Scharr Operator", "Canny Algorithm" (рис.2.4)

–Результат обробки відображається у новому вікні (рис.2.5)

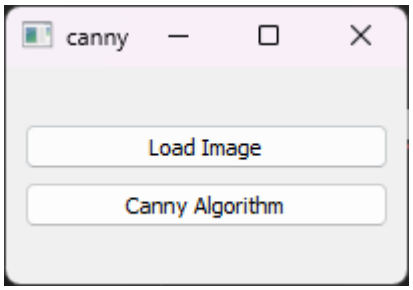
–Користувач може повторити обробку для іншого зображення або алгоритму.

```python
import sys
import cv2
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QPushButton, QFileDialog, QMessageBox, QMainWindow
from PyQt5.QtGui import QPixmap, QImage
from PyQt5.QtCore import Qt

class ContourDetectionApp(QMainWindow):
    def __init__(self):
        super().__init__()

        self.image_path = None

        self.central_widget = QWidget(self)
        self.setCentralWidget(self.central_widget)

        self.original_label = QLabel(self.central_widget)
        self.processed_label = QLabel(self.central_widget)

        self.load_button = QPushButton('Load Image', self.central_widget)
        self.canny_button = QPushButton('Canny Algorithm', self.central_widget)

        self.load_button.clicked.connect(self.load_image)
        self.canny_button.clicked.connect(self.detect_canny)

        layout = QVBoxLayout(self.central_widget)
        layout.addWidget(self.original_label)
        layout.addWidget(self.load_button)
        layout.addWidget(self.canny_button)
        layout.addWidget(self.processed_label)

        self.show()

    def load_image(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        self.image_path, _ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Image Files (*.png *.jpg *.bmp);;All Files (*)", options=options)

        if self.image_path:
            pixmap = QPixmap(self.image_path)
            self.original_label.setPixmap(pixmap.scaledToWidth(400, Qt.SmoothTransformation))

    def detect_canny(self):
        if self.image_path:
            try:
                image = cv2.imread(self.image_path)
                gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                edges = cv2.Canny(gray, 50, 150)
                processed_image = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)
                pixmap = self.get_pixmap(processed_image)
                self.show_processed_image(pixmap)
            except Exception as e:
                self.show_error_message(str(e))

    def get_pixmap(self, image):
        height, width, channel = image.shape
        bytes_per_line = 3 * width
        q_image = QImage(image.data, width, height, bytes_per_line, QImage.Format_RGB888)
        pixmap = QPixmap.fromImage(q_image)
        return pixmap

    def show_processed_image(self, pixmap):
        new_window = QMainWindow(self)
        new_central_widget = QWidget(new_window)
        new_window.setCentralWidget(new_central_widget)

        new_processed_label = QLabel(new_central_widget)
        new_processed_label.setPixmap(pixmap)

        new_layout = QVBoxLayout(new_central_widget)
        new_layout.addWidget(new_processed_label)
        new_window.show()

    def show_error_message(self, message):
        error_dialog = QMessageBox(self)
        error_dialog.setIcon(QMessageBox.Warning)
        error_dialog.setText(message)
        error_dialog.setWindowTitle("Error")
        error_dialog.exec_()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = ContourDetectionApp()
    sys.exit(app.exec_())
```

Рис.2. Програмна реалізація алгоритму Кенні

```python
import sys
import cv2
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QPushButton, QFileDialog, QMessageBox, QMainWindow
from PyQt5.QtGui import QPixmap, QImage
from PyQt5.QtCore import Qt

class ContourDetectionApp(QMainWindow):
    def __init__(self):
        super().__init__()

        # Ініціалізація вікна та важливих змінних
        self.image_path = None

        self.central_widget = QWidget(self)
        self.setCentralWidget(self.central_widget)

        # Віджети та їх розміщення в макеті
        self.original_label = QLabel(self.central_widget)
        self.processed_label = QLabel(self.central_widget)

        self.load_button = QPushButton('Load Image', self.central_widget)
        self.scharf_button = QPushButton('Scharf Operator', self.central_widget)

        self.load_button.clicked.connect(self.load_image)
        self.scharf_button.clicked.connect(self.detect_scharf)

        layout = QVBoxLayout(self.central_widget)
        layout.addWidget(self.original_label)
        layout.addWidget(self.load_button)
        layout.addWidget(self.scharf_button)
        layout.addWidget(self.processed_label)

        self.show()

    def load_image(self):
        # Функція завантаження зображення та відображення його на екрані
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        self.image_path, _ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Image Files (*.png *.jpg *.bmp);;All Files (*)", options=options)

        if self.image_path:
            pixmap = QPixmap(self.image_path)
            self.original_label.setPixmap(pixmap.scaledToWidth(400, Qt.SmoothTransformation))

    def detect_scharf(self):
        # Функція застосування оператора Scharf до завантаженого зображення
        if self.image_path:
            try:
                image = cv2.imread(self.image_path, cv2.IMREAD_GRAYSCALE)
                scharf_x = cv2.Scharr(image, cv2.CV_64F, 1, 0)
                scharf_y = cv2.Scharr(image, cv2.CV_64F, 0, 1)
                magnitude = cv2.magnitude(scharf_x, scharf_y)
                processed_image = cv2.cvtColor(magnitude.astype('uint8'), cv2.COLOR_GRAY2BGR)
                pixmap = self.get_pixmap(processed_image)
                self.show_processed_image(pixmap)
            except Exception as e:
                self.show_error_message(str(e))

    def get_pixmap(self, image):
        # Функція конвертації обробленого зображення у формат для відображення
        height, width, channel = image.shape
        bytes_per_line = 3 * width
        q_image = QImage(image.data, width, height, bytes_per_line, QImage.Format_RGB888)
        pixmap = QPixmap.fromImage(q_image)
        return pixmap

    def show_processed_image(self, pixmap):
        # Функція відображення обробленого зображення у новому вікні
        new_window = QMainWindow(self)
        new_central_widget = QWidget(new_window)
        new_window.setCentralWidget(new_central_widget)

        new_processed_label = QLabel(new_central_widget)
        new_processed_label.setPixmap(pixmap)

        new_layout = QVBoxLayout(new_central_widget)
        new_layout.addWidget(new_processed_label)
        new_window.show()

    def show_error_message(self, message):
        # Функція відображення повідомлення про помилку
        error_dialog = QMessageBox(self)
        error_dialog.setIcon(QMessageBox.Warning)
        error_dialog.setText(message)
        error_dialog.setWindowTitle("Error")
        error_dialog.exec_()

if __name__ == '__main__':
    # Запуск програми
    app = QApplication(sys.argv)
    window = ContourDetectionApp()
    sys.exit(app.exec_())
```

Рис.2.1. Програмна реалізація оператора Шарра

а) 

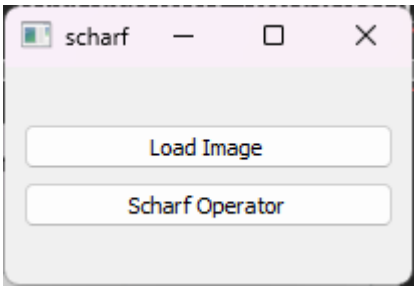
б) 

Рис.2.3 - а) алгоритм Кенні б) оператор Шарра

а) 

б) 

Рис.2.4 - а) алгоритм Кенні б) оператор Шарра

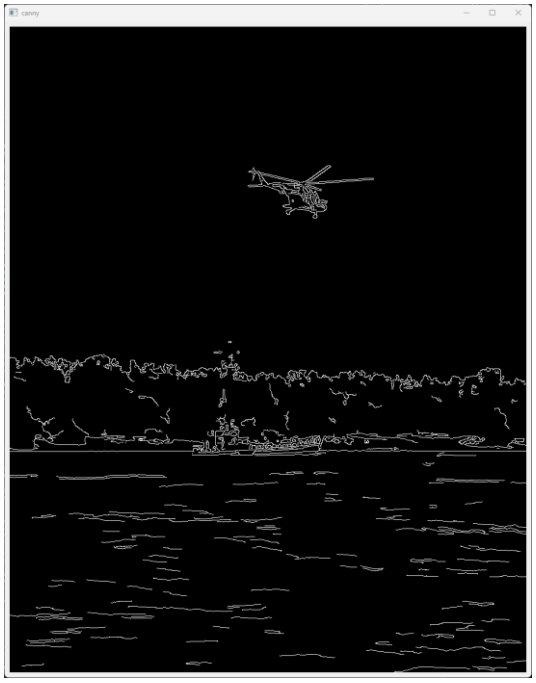
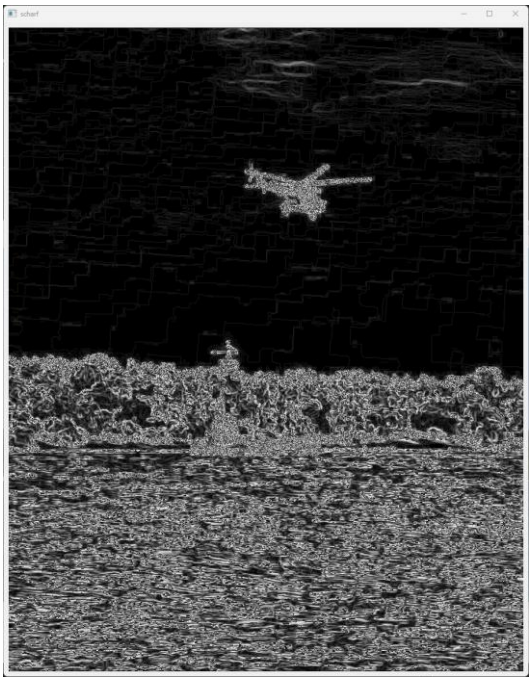
а) 
б) 

Рис.2.5 - а) алгоритм Кенні б) оператор Шарра

Для порівняння цих методів вибрано два об’єкта: гвинтокрил та човен (рис.2.6). Причому човен на тлі пляжу та лісу, що ускладнить його виокремлення.

Рис.2.6. Світлина з гвинтокрилом та човном

Спочатку розглянемо роботу алгоритму Кенні (рис.2.7):

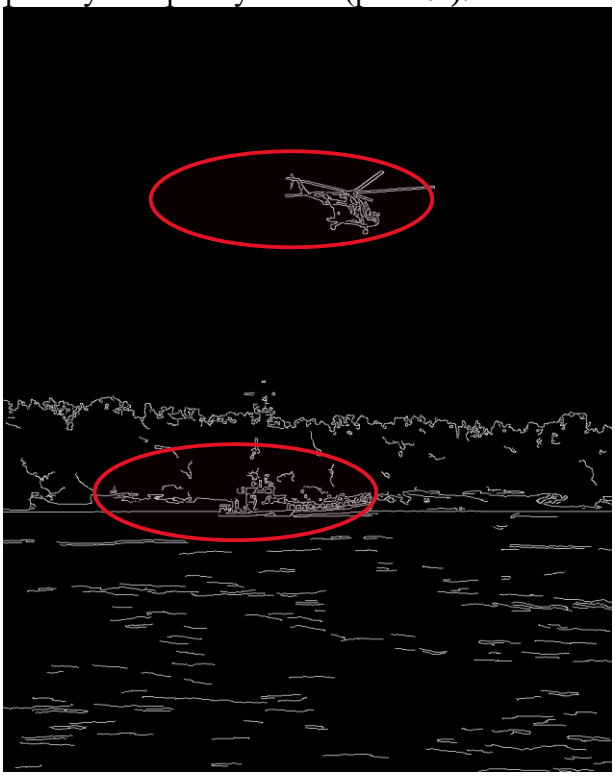


Рис.2.7. Результат роботи алгоритму Кенні

Ми можемо побачити, що цей метод чудово виділив контури гвинтокрила. Човен теж має доволі чіткий силует. Цей метод виокремив реальні межі об'єктів.

Зараз розглянемо роботу оператора Шарра (рис.2.8):

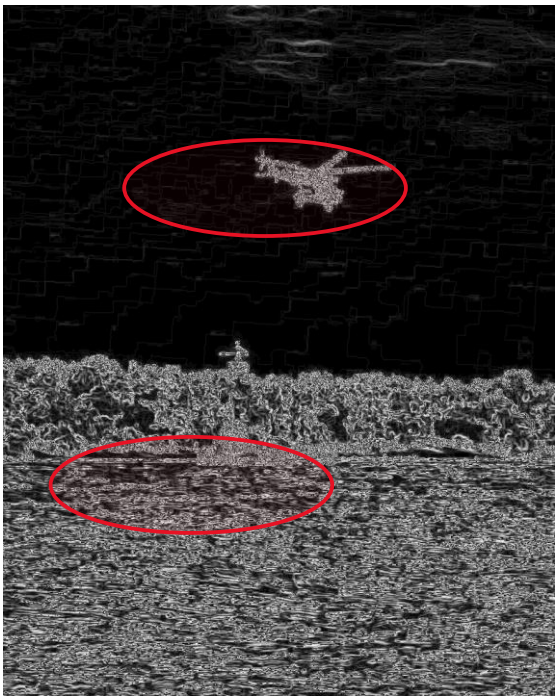


Рис.2.8. Результат роботи оператора Шарра

Можемо помітити, що виявлено більшу кількість контурів, але їх значимість не є суттєвою через наявність шуму, менша чіткість об’єктів. Вертоліт можна ще розпізнати, а човен зовсім губиться на фоні лісу та води: важко його виявити.

Підсумовуючи результати, можна зробити висновок, що алгоритм Кенні, на відміну від оператора Шарра, показав кращий результат роботи у виокремленні контурів об’єктів. Тому можна стверджувати, що алгоритм Кенні досі залишається чудовим варіантом для виділення контурів зображень об'єктів, хоча і має нижчу швидкість роботи, порівнюючи з Оператором Шарра.

7. Висновки та перспективи подальших досліджень

Узагальнюючи отримані результати, можна зазначити, що виокремлення контурів на зображеннях залишається актуальною проблемою через ряд труднощів, таких як розриви контурів, наявність помилкових контурів через шум, а також широкі контурні лінії через розмитість. Порівнюючи різні методи контурного аналізу, виявлено, що оператор Превітт має найнижчі ресурсні витрати, але не завжди ефективно виділяє контури. Оператор Собеля та Шарра вважаються оптимальними за витратами ресурсів та якістю виділення, тоді як оператор Лапласа є швидшим та менш витратним за рахунок меншої обчислювальної складності.

Один із значущих недоліків алгоритмів виокремлення контурів зображень об’єктів полягає в високій ймовірності помилок, особливо якщо на зображенні присутній шум або текстура. Практично всі розглянуті методи потребують наявності однорідного фону, на якому розташовані достатньо контрастні об’єкти. Помилки призводять до виникнення випадкових точок, які можуть бути ідентифіковані як контурні або як розриви справжніх протяжних контурів. Причина таких невдач знаходиться у самому підході, оскільки для прийняття рішення про наявність контуру в кожній точці зображення аналізується лише невелика околиця цієї точки.

Детектор контурів Кенні також вважається ефективним методом виявлення країв об’єкта завдяки своїй високій чутливості до реальних меж та точній локалізації. Хоча він може мати нижчу швидкість виявлення країв порівняно з іншими методами, він демонструє гарні результати за критеріями точності та надійності.

Існуючі спеціалізовані алгоритми, призначені для ефективного розв'язання конкретних завдань, потребують дотримання ряду умов обмежень.

Перспективним напрямком подальших досліджень є розробка алгоритмів иокремлення контурів зображень об'єктів, що базуються на двомасштабній статистичній моделі зображення.

Список використаної літератури

1. Ainsworth T. Buyer Beware // Security Oz. 2002. Vol. 19. P. 18–26.
2. Крючкова, Л. П., Стрельніков, В. І., Акулінічева, М. В., Бортник, О. С., & Дібрівний, О. А. (2020). Виокремлення контурів зображень об'єктів в інтелектуальних системах відеоспостереження. Зв'язок, № 5, 50-56.
3. Bastian, L., & Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. Computer Society Conference on Computer Vision and Pattern Recognition. (Vol. 2). Proceedings.
4. Polyakova M. Image contour segmentation in the space of coefficients of signal semantic wavelet transform / M. Polyakova, V. Krylov // Modern Problems of Radioengineering, Telecommunications and Computer Science: Proc. of Intern. Conf. TCSET 2008, Ukraine – 2008. – P. 280–283.
5. Polyakova M. Classification of methods of the signal semantic wavelet transform for image contour segmentation / M. Polyakova, V. Krylov // Int. Journal of Computing – 2008. – Vol. 7 – Issue 1– P. 51 – 57.
6. Williams, D. J., & Shah, M. (1992). A fast algorithm for active contours and its application to computer vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), 111-123.
7. Terzopoulos, D., Witkin, A., & Kass, M. (1988). Constraints on deformable models: Recovering 3D shape from 2D images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(4), 363-370.
8. Xu, C., & Prince, J. L. (1998). Snakes, shapes, and prior knowledge. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), 321-333.
9. Canny J.E. A computational approach to edge detection / J.E. Canny // IEEE Trans Pattern Analysis and Machine Intelligence. — 1986. — № 8. — P. 679 - 698.
10. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.
11. Jähne, B. (2002). Digital image processing. Springer.
12. Pratt W.K. Digital Image Processing: PIKS Inside, Third Edition. William K. Pratt / John Wiley and Sons, Inc., New York. – 2001 – 736 p.
13. Canny J. F. Finding edges and lines in images // Master’s thesis. MIT, Cambridge, USA, 1983. P. 50—67.
14. Rajashekar P. Evaluation of Stopping Criterion in Contour Tracing Algorithms / P.Rajashekar Reddy et al, / (IJCSIT) International Journal of Computer Science and Information Technologies. – Vol. 3 (3). – 2012. – P. 3888-3894.
15. Pavlidis T. Algorithms for Graphics and Image Processing / T. Pavlidis – US, Rockville, Maryland: Computer Science Press, 1982. – 438 p
16. Prewitt, J. M-S. (1970). Object enhancement and extraction. Picture processing and Psychopictorics, 10(1), 15–19.
17. Sobel, I., & Feldman, G. (1973). A 3x3 Isotropic Gradient Operator for Image Processing. Pattern Classification and Scene Analysis, 271–272
18. Kirsch, R. (1971). Computer determination of the constituent structure of biological images. Computers and Biomedical Research, 4, 315–328.
19. Moon, P., & Spencer, D. E. (1953). The Meaning of the Vector Laplacian. J. Franklin Inst., 551–558.
20. D.A. Sharr and H.S. Ahuja (1980). A New Method for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 210-216.

References

1. Ainsworth T. Buyer Beware // Security Oz. 2002. Vol. 19. P. 18–26.
2. L. P. Kryuchkova, V. I. Strelnikov, M. V. Akulinicheva, O. S. Bortnyk, O. A. Dibrivnyi (2020). Algorithms for selecting the outlines of objects images in intellectual video surveillance systems. Connectivity, № 5, 50-56.
3. Bastian, L., & Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. Computer Society Conference on Computer Vision and Pattern Recognition. (Vol. 2). Proceedings.
4. Polyakova M. Image contour segmentation in the space of coefficients of signal semantic wavelet transform / M. Polyakova, V. Krylov // Modern Problems of Radioengineering, Telecommunications and Computer Science: Proc. of Intern. Conf. TCSET 2008, Ukraine – 2008. – P. 280–283.
5. Polyakova M. Classification of methods of the signal semantic wavelet transform for image contour segmentation / M. Polyakova, V. Krylov // Int. Journal of Computing – 2008. – Vol. 7 – Issue 1– P. 51 – 57.
6. Williams, D. J., & Shah, M. (1992). A fast algorithm for active contours and its application to computer vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), 111-123.
7. Terzopoulos, D., Witkin, A., & Kass, M. (1988). Constraints on deformable models: Recovering 3D shape from 2D images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(4), 363-370.
8. Xu, C., & Prince, J. L. (1998). Snakes, shapes, and prior knowledge. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), 321-333.
9. Canny J.E. A computational approach to edge detection / J.E. Canny // IEEE Trans Pattern Analysis and Machine Intelligence. — 1986. — № 8. — P. 679 - 698.
10. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.
11. Jähne, B. (2002). Digital image processing. Springer.
12. Pratt W.K. Digital Image Processing: PIKS Inside, Third Edition. William K. Pratt / John Wiley and Sons, Inc., New York. – 2001 – 736 p.
13. Canny J. F. Finding edges and lines in images // Master’s thesis. MIT, Cambridge, USA, 1983. P. 50—67.
14. Rajashekar P. Evaluation of Stopping Criterion in Contour Tracing Algorithms / P.Rajashekar Reddy et al, / (IJCSIT) International Journal of Computer Science and Information Technologies. – Vol. 3 (3). – 2012. – P. 3888-3894.
15. Pavlidis T. Algorithms for Graphics and Image Processing / T. Pavlidis – US, Rockville, Maryland: Computer Science Press, 1982. – 438 p
16. Prewitt, J. M-S. (1970). Object enhancement and extraction. Picture processing and Psychopictorics, 10(1), 15–19.
17. Sobel, I., & Feldman, G. (1973). A 3x3 Isotropic Gradient Operator for Image Processing. Pattern Classification and Scene Analysis, 271–272
18. Kirsch, R. (1971). Computer determination of the constituent structure of biological images. Computers and Biomedical Research, 4, 315–328.
19. Moon, P., & Spencer, D. E. (1953). The Meaning of the Vector Laplacian. J. Franklin Inst., 551–558.
20. D.A. Sharr and H.S. Ahuja (1980). A New Method for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 210-216.