DOI: 10.36910/6775-2524-0560-2020-38-01
УДК: 004.93'1
Ліщина Наталія Миколаївна, к.т.н., доцент
https://orcid.org/0000-0002-5200-536X
Ліщина Валерій Олександрович, к.т.н., доцент
https://orcid.org/0000-0002-2371-3850
Повстяна Юлія Славомирівна, к.т.н., доцент
https://orcid.org/ 0000-0001-5426-4157
Луцький національний технічний університет

ПІДХОДИ ТА АЛГОРИТМИ ОБРОБКИ ТА РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ СКЛАДНОЇ СТРУКТУРИ

Ліщина Н.М., Ліщина В.О., Повстяна Ю.С. Підходи та алгоритми обробки та розпізнавання зображень складної структури. У статті описані високопродуктивні алгоритми та програмне забезпечення, що виконує одну із задач обробки зображень – аналітичного описання контурів об'єктів виділених на растрових зображеннях. Показано, що запропонована методика, алгоритми та розроблене програмне забезпечення повністю вирішує розглянуту задачу для зображень як штучного так і природного походження.
Ключові слова: контур зображення, алгоритм, аналітичний опис об'єктів, апроксимація, час обробки.

Лищина Н.М., Лищина В.А., Повстяная Ю.С. Подходы и методы обработки и распознавания изображений сложной структуры. В статье описанные высокопроизводительные алгоритмы и программное обеспечение, выполняющее одну из задач обработки изображений – аналитического описания контуров объектов выделенных на растровых изображениях. Показано, что предложенная методика, алгоритмы и разработанное программное обеспечение полностью решает рассматриваемую задачу для изображений как искусственного так и естественного происхождения.
Ключевые слова: контур изображения, алгоритм, аналитическое описание объектов, аппроксимация, время обработки.

Nataliia Lishchyna, Valeriy Lishchyna, Ju. Povstyana. Approaches and algorithms for processing and image recognition of complex structure. In article highly productive algorithms and software are described, that executes one of tasks of the image processing – analytical description of contours of objects of selected on the raster images. It is shown, that a method, algorithms and developed software, is offered fully decides the considered task for the images as artificial so natural origin.
Keywords: contour of image, algorithm, analytical description of objects, approximation, time of treatment.

Поставка проблеми. Аналіз та розпізнавання зображень складної структури різної фізичної природи широко використовується в різних областях науки і техніки, наприклад в медицині та біології для аналізу рентгенограм, тканин клітини і хромосом; в неруйнівному контролі матеріалів і середовищ; в робототехніці при аналізі динамічних тривимірних сцен; в криміналістиці для реставрації і відновлення низькоконтрастних документів; в лазерній локації для виявлення малорозмірних і великорозмірних об'єктів й у інших галузях.

Формування мети дослідження. Метою роботи є розробка нового підходу до створення високопродуктивних технологій синтезу, обробки та розпізнавання зображень складної структури різної фізичної природи, який дає можливість генерувати зображення тривимірних об'єктів, здійснювати геометричні перетворення для покращення якості зображення, формувати системи інформативних ознак для опису малорозмірних і великорозмірних рухомих і нерухомих образів, і на їх основі створювати високопродуктивні розпізнавальні системи реального часу.

Виклад основного матеріалу.

Існуючі алгоритми побудови контуру об'єктів використовують принцип трасування межі однорідних ділянок півтонових або дворівневих зображень. Основним їх недоліком є те, що при трасуванні меж ділянок, зображення неодноразово проглядається в довільних напрямках, які визначаються поведінкою межі об'єкту. А це приводить до зниження ефективності роботи програмного забезпечення, особливо, коли обробляються зображення великих розмірів. Тому доцільно розробити однопрохідний алгоритм, що позбавлений зазначених недоліків.

В основу алгоритму можна покласти принцип перевірки на парність, що застосовується в процедурах заповнення областей. Алгоритми заповнення областей, які використовують перевірку на парність, ґрунтуються на тому, що довільна пряма перетинає довільну замкнуту криву парну кількість разів.

Розглянемо суть однопрохідного алгоритму побудови контуру об'єкту на прикладі рис. 1. Нехай зображення сканується зліва-направо зверху-вниз апертурою певного розміру, доки пікселі однорідних ділянок мають однакові фіксовані значення. При досягненні межі об'єкту (точка A) породжуються дві дуги контуру об'єктів I та II ліва та права. Ці дуги заносяться у список контурів відповідних об'єктів як незамкнуті послідовності точок. Координати (x, y) точки A заносяться у списки точок цих дуг. В наступному скані при перетині меж об'єктів координати точок перетину добавляються у списки точок

найближчих дуг контурів. Оскільки пікселі тіла об'єкту є чотири-зв'язними, а контури – восьми-зв'язними, то відстань між сусідніми контурними точками буде рівна одиниці. Опираючись на принцип парності точок перетину деякої прямої лінії та замкнутого контуру, до списків дуг контурів буде добавлено однакову кількість точок. При подальшому скануванні зображення, у точці *B* буде породжено нові дуги контурів та добавлені до відповідних об'єктів. У точці *C* дуги *AC* та *BC* об'єднаються в єдиний сегмент контуру.

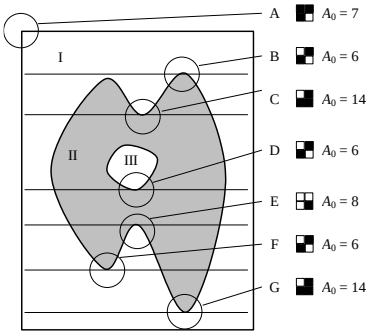

Рисунок 1 – Характерні ділянки контурів об'єктів

Продовжуючи обробку зображення за розглянутим підходом, нові дуги контурних ланцюгів будуть створені у точках *D* та *F*, а у точках *E, G* та *H* відповідні дуги будуть об'єднані в ланцюжки та утворять замкнуті контури ділянок.

Аналіз алгоритму формування контурів

Розглянутий алгоритм описання контурів об'єкта забезпечує побудову контурів (зовнішніх та внутрішніх) всіх об'єктів за один перегляд зображення з контурним препаратом. Необхідно виділити тільки дві ділянки пам'яті довжиною пропорційною ширині зображення для збереження маркування об'єктів та їх сусідства. Основні затрати внутрішньої пам'яті будуть припадати для збереження дуг та ланцюгів контурів. У процесі обробки об'єкту його контур містить тільки дуги, які поступово об'єднуються між собою. Після завершення обробки внутрішнього контуру або об'єкту в цілому всі дуги замикаються та утворюють замкнуті ланцюги.

Опис програм

Основою розробленого програмного забезпечення аналітичного описання контурів є функція ObjectPtContourDescr(). Відповідно до свого функціонування, вона викликає допоміжні підпрограми (функції), що реалізовують окремі дії над інформаційними об'єктами (дугами, ланцюгами та ін.). Відповідно до інформаційних об'єктів з якими оперують програми, функція розбиті на групи. Належність функції до певної групи визначається першим словом у її назві. А саме: ObjectPt... – функції працюють з об'єктами в цілому; Branch... – функції працюють з дугами контурів; Chain... – функції працюють з ланцюгами контурів; Neighbourhood... – функції відслідковують сусідство об'єктів.

Програма ObjectPtContourDescr() є базовою для аналітичного описання контурів об'єктів на бінарних зображеннях. Прототип функції міститься у файлі заголовків EdgeDescription.h, а повний текст у файлі EdgeDescription.c.

Основними вхідними параметрами функції є вказівник contF на структуру типу FILE, через який відбувається зчитування вхідних даних (бінарного зображення) з файлу. Зображення записано у файлі в форматі RAW з бітовою глибиною 8 бітів на піксель. Розміри зображення передаються через змінні imWidth та imHeight, що відповідно задають його ширину та висоту в пікселях. Для обчислення інтегральних характеристик об'єктів в процесі описання їх контурів та непохідних при класифікації, функції передаються вказівники типу FILE для доступу до первинного зображення та його градієнту.

Результатами роботи програми є вказівник на масив елементів struct SObjectPt, у якому розміщено описані об'єкти та довжина цього масиву ObjectCount. Після перегляду всіх рядків зображення програма звільняє використану пам'ять та повертає викликаючій програмі вказівник на масив описаних об'єктів та його розмір через формальний параметр ObjectCount.

Група функції ObjectPt… опрацювання об'єктів

Група функцій, що починається з префіксу ObjectPt призначена для виконання операцій над об'єктами в цілому. До неї входять функції: ObjectPtCreate() – створення та ініціалізації об'єкту; ObjectPtDelete() – знищення об'єкту; ObjectPtConcatenate() – об'єднання об'єктів; ObjectPtCleaning() – очистки порожніх об'єктів.

Функція ObjectPtCreate() призначена створення та ініціалізації нового об'єкту. Вона повертає нові значення вказівника на масив об'єктів, його максимальну довжину ObjectPtMax та індекс створеного об'єкту ObjectPtCnt.

```
struct SObjectPt* ObjectPtCreate(
struct SObjectPt* pObject,
int &ObjectPtMax,
int &ObjectPtCnt,
struct SBranch* pBranch);
```

У створений об'єкт переписується дуга контуру, вказівник на яку pBranch передається через формальні параметри. Пам'ять, що виділяється додатково ініціалізується нулем.

Функція ObjectPtDelete() призначена для знищення об'єкту на який вказує pObject. Спочатку переглядається внутрішня структура об'єкту та звільняється вся пам'ять, що виділена в процесі його побудови, а потім усім полям структури присвоюються нульові значення.

Функція ObjectPtConcatenate() об'єднує (зливає) об'єкти, коли у двох ділянок зображення, які оброблялися незалежно, появляється точка, що є чотирьох-зв'язною з обома ділянками. У випадку, коли "об'єднуються" об'єкти з однаковими індексами, то це значить, що замкнувся один із внутрішніх контурів.

Функція ObjectPtCleaning() призначена для видалення масиву порожніх об'єктів з метою його ущільнення.

Функція ObjectPtCleaning() повертає нову довжину щільно упакованого динамічного масиву.

Група функції Branch… опрацювання дуг контурів

У процесі побудови, контури спочатку представляються дугами, тобто незамкнутими ланцюжками контурних точок. Група функцій назви яких починаються з префіксу Branch призначена для роботи з дугами. До цього класу відносяться функції: BranchInsert() – вставлення нової дуги в об'єкт; BranchAddPoint() – додавання точки до біжучої дуги; BranchGrow() – нарощення вказаної дуги точками з інших дуг; BranchMove() – переміщення дуг з одного об'єкту в інший; BranchNext() – циклічний інкремент індексу біжучої дуги; BranchMove2Chain() – замикання дуг та переміщення в масив ланцюгів.

Функція BranchInsert() добавляє нові дуги до об'єкту. Вхідними параметрами функції є pObject вказівник на об'єкт до якого добавляються дуги на які вказує pBranch.

```
void BranchInsert(struct SObjectPt* pObject, struct SBranch* pBranch);
```

Функція BranchAddPoint() добавляє нову точку контуру з абсолютними координатами зображення (n, m) у біжучу дугу об'єкту pObject та переміщує індикатор біжучої дуги на наступну, якщо Next - істина.

```
void BranchAddPoint(struct SObjectPt* pObject, int, int);
```

Функція BranchGrow() призначена для нарощення дуги контуру точками з двох інших дуг.

```
void BranchGrow(struct SBranch* pDestBranch,
struct SBranch* pSrcBranchOne,
struct SBranch* pSrcBranchTwo, bool bFree);
```

У кінець дуги приймача pDestBranch добавляються точки з дуг джерел pSrcBranchOne та pSrcBranchTwo послідовно і, якщо bFree – істина, дуги джерела знищуються. Останній параметр добавлено для коректного опрацювання службового нульового об'єкту дуги якого не повинні знищуватися. Функція BranchGrow() використовується при об'єднанні об'єктів.

Функція BranchMove() переміщує Number послідовних дуг з динамічного масиву на який вказує pSourBranch у біжучу позицію масиву дуг об'єкту pObject.

```
void BranchMove(struct SObjectPt* pObject,
struct SBranch* pSourBranch,
int);
```

Функція використовується у програмах об'єднання об'єктів. Вона переміщає незамкнуті дуги контурів з одного об'єкту в інший. Спочатку перевіряється максимальний розмір масиву дур об'єкту приймача і при необхідності розширюється. Потім розсувається існуючий масив дуг на Number елементів та у порожнє місце переносяться дуги. Індикатор біжучої дуги інкрементується на Number.

Функція BranchMove2Chain() об'єднує дві сусідні дуги об'єкту pObject утворюючи внутрішній чи зовнішній контурний ланцюг та переміщує його в масив ланцюгів.

```
void BranchMove2Chain(struct SObjectPt* pObject);
```

Функція BranchNext() призначену для циклічного переходу на наступну дугу об'єкту pObject.

```
void BranchNext(struct SObjectPt* pObject);
```

Функція збільшує на одиницю індикатор біжучої дуги, якщо біжуча дуга є останньою в масиві, то індикатор встановлюється в нуль.

Група функції Chain… опрацювання ланцюгів контурів

Функції з цієї групи працюють з уже завершеними (замкнутими) ланцюгами контурів. До неї входить тільки дві функції: ChainSetRect() – визначення описуючого чотирикутника; ChainMove() – переміщення масиву ланцюгів з одного об'єкту в інший.

Функція ChainSetRect() визначає описуючий чотирикутник замкнутого ланцюга на який вказує змінна pChain.

int ChainSetRect(struct SChain* pChain);

Описуючий чотирикутник представляється через абсолютні координати лівого верхнього та правого нижнього кутів. Ці координати визначаються як мінімальне та максимальні значення координат точок, що входять в ланцюг за кожною з осей.

Функція ChainMove() призначена для переміщення Number ланцюгів з позиції масиву ланцюгів на яку вказує pSourChain у масив ланцюгів об'єкту pObject.

void ChainMove(struct SObjectPt* pObject,

struct SChain* pSourChain, int Number);

Оскільки, на відміну від дуг, порядок чергування внутрішніх ланцюгів не має принципового значення, то для їх перезапису використовується стандартна функція memmove(). Тільки, перед переміщення збільшується ємність масиву дуг об'єкту приймача, а після перезапису звільняється відповідна пам'ять в якій зберігалися ланцюги.

Група функції Neighbour… визначення сусідства об'єктів

Сусідство об'єктів має важливе значення на етапі маніпуляції з об'єктами, при визначенні їх взаємного положення, спільних точок контуру, об'єднання та ін. Визначення сусідства уже сформованих об'єктів є перебірковою та затратною задачею, тому сусідство доцільніше визначати на етапі формування їх контурів, коли є вся необхідна для цього інформація. До групи функцій визначення сусідства об'єктів входить: NeighbourObjAssign() – встановлення взаємного сусідства двох об'єктів; NeighbourObjChange() – заміна взаємного сусідства двох об'єктів; NeighbourObjMove() – переміщення даних про сусідство об'єкту; NeighbourObjDelete() – знищення даних про сусідство об'єкту.

Функція NeighbourObjAssign() призначена для встановлення даних про сусідство двох об'єктів, що задані індексами ObjHigher та ObjLower у масиві об'єктів pObject.

void NeighbourObjAssign(struct SObjectPt* pObject,

int ObjHigher, int ObjLower, int LoPointCount, int LoHighSum);

Через змінні LoPointCount та LoHighSum передається кількість точок контуру, що належать обом об'єктам та сумарне значення їх яскравостей.

Функція NeighbourObjMove() призначена для переміщення даних про сусідство об'єкту pObject в нову позицію, з позиції OldObjID у позицію NewObjID.

void NeighbourObjMove(struct SObjectPt* pObject, int NewObjID, int OldObjID);

Таке переміщення необхідно для збереження порядку сортування в масиві даних про сусідство. Потреба в переміщенні виникає у випадку коли індекс об'єкту міняється. Для переміщення даних відшукуються стара та нова позиція даних в масиві про сусідство об'єктів, відповідно до індексів OldObjID та NewObjID.

Функція NeighbourObjChange()призначена для вилучення даних про сусідство об'єкту, з індексом SourceObj, з масиву pObject та призначення об'єктам, які були сусідніми з SourceObj, сусідства з об'єктом з індексом DestinObj.

void NeighbourObjChange(struct SObjectPt* pObject, int DestinObj, int SourceObj);

Функція застосовується у випадках коли сусідні об'єкти об'єднуються (зливаються) і природно новоутворений об'єкт (DestinObj) перебирає до себе сусідство з об'єктами, що були суміжними з об'єктом (SourceObj), що приєднався до нього. Вона працює за простим алгоритмом. Спочатку знищуються дані про взаємне сусідство об'єктів DestinObj та SourceObj за допомогою функції NeighbourObjDelete(). Пізніше організовується цикл, у якому переглядаються всі сусідні з SourceObj об'єкти. В кожному з них функцією NeighbourObjDelete() знищується сусідство з об'єктом SourceObj а за допомогою функції NeighbourObjAssign() призначається сусідство з об'єктом DestinObj.

Функція NeighbourObjDelete()призначена для вилучення даних про сусідство об'єкту заданого індексом ObjDelete з об'єкту заданого вказівником pObject.

void NeighbourhoodObjDelete(struct SObjectPt* pObjDest, int ObjDelete);

Функція переглядає масив з даними про сусідство об'єктів, знаходить та вилучає дані про сусідство з об'єктом ObjDelete шляхом стискання масиву, зменшує на одиницю лічильник об'єктів сусідів та встановлює індекс наступного за останнім сусіда в –1. Встановлення індексу, наступного за останнім сусідом в масиві, в –1, забезпечує коректну зупинку алгоритму пошуку об'єктів сусідів у впорядкованому за не спаданням масиві.

Для тестування програмного забезпечення розроблено технологічні програми, на які покладалися функції забезпечення інтерфейсу з користувачем, відображення та аналізу результатів роботи розроблених програм.

Завантажувальний модуль інтерфейсної програми називається ContourDescribe.exe. У тому ж каталозі повинен бути ініціалізаційний файл ContourDescribe.ini в якому прописано завдання на обробку. Ініціалізаційний файл має наступну структуру:

ContourDescribe.imWidth=256

ContourDescribe.imHeight=256

ContourDescribe.InputFile=c:\HouseRoofs\Roofs\Test9\Test_1.raw

де ContourDescribe – кодове, що визначає належність ключів, що йдуть після крапки саме до цієї програми;

imWidth, imHeight – ширина та висота зображення в пікселях;

InputFile – повна назва файлу зі зображенням у RAW форматі.

Інтерфейсна програма читає ініціалізаційний файл та перевіряє повноту та коректність вхідних даних. Якщо перевірка завершилася успішно, викликаються функції аналітичного описання контурів, в інакшому випадку видається повідомлення про помилку та виконання програми переривається.

Для виводу результатів описання контурів об'єктів розроблено дві функції imDrawContourPoint() та ObjectPtTestPrint(). Перша відтворює бінарне зображення на основі аналітичного описання контурів та служить для візуальної оцінки роботи програми, а друга аналізує коректність контурних ланцюгів та виводить інформацію про об'єкти та їх контури в текстовому вигляді. За її допомогою можна точно оцінити коректність роботи програм описання контурів.

Прототип функції відтворення бінарних зображень об'єктів за їх контурами має вигляд:

void imDrawContourPoint(FILE *outF, int imWidth, int imHeight,

struct SObjectPt* pObject, int ObjectCount,

int ObjectDraw)

де outF – вказівник на відкритий файл, у який виводитиметься зображення;

imWidth, imHeight – ширина та висота зображення;

pObject – вказівник на масив об'єктів;

ObjectCount – довжина масиву об'єктів;

ObjectDraw – індекс об'єкту який необхідно вивести, якщо індекс є від'ємним числом, то виводяться всі об'єкти.

Функція аналізу та виводу структури об'єктів у текстовій формі викликається з параметрами

void ObjectPtTestPrint(FILE *logF,

struct SObjectPt* pObject, int ObjectCount,

int nObject, int bSquare, int bNeighbors,

int bChains, int bPoints, int bPointDiff)

де logF – вказівник на відкритий файл у який виводитиметься текстова інформація;

pObject – вказівник на масив об'єктів;

ObjectCount – довжина масиву об'єктів;

nObject – індекс об'єкту який необхідно вивести, якщо індекс є від'ємним числом, то виводяться всі об'єкти.

bSquare, bNeighbors, bChains, bPoints, bPointDiff – ключі, ненульове значення яких забезпечує друк у файл: площі об'єкту; списку сусідніх об'єктів та параметри їх спільних ділянок контурів; списку ланцюгів та їх описуючих чотирикутників; точок контуру; різниці між координатами сусідніх точок контуру, якщо вона перевищує одиницю.

Висновки. Відомі алгоритми обходу та аналітичного описання контурів вимагають багаторазового перегляду контурних точок зображення. На зображеннях складної структури це приводить до багаторазового поглядання зображення, що опрацьовується. Якщо, вихідне зображення має досить великі розміри і його не вдається повністю розмістити в оперативній пам'яті, то підчитування фрагментів зображення буде вимагати інтенсивної роботи із зовнішнім запам'ятовуючим пристроєм. Оскільки операція роботи із зовнішніми пристроями є повільною, то розглянуті алгоритми будуть критичними за часом виконання.

Список бібліографічного опису.

1.Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.

2. Фисенко В.Т., Фисенко Т.Ю. Компьютерная обработка и распознавание изображений. СПб.:, 2008.

References.

1.Honsales R., Vuds R. Tsyfrovaia obrabotka yzobrazhenyi. M.: Tekhnosfera, 2005. 1072 s.

2.Fysenko V.T., Fysenko T.Iu. Kompiuternaia obrabotka y raspoznavanye yzobrazhenyi. SPb.:, 2008.