Практична робота №6

Тема: Дослідження методів виділення контурів об'єктів на цифрових відеозображеннях

Мета роботи: Дослідити найпопулярніші методи визначення контурів на цифрових відеозображеннях.

6.1 Теоретична інформація

Є велика група методів заснована на використанні математичних моделей. Всі вони полягають у статистичному аналізу фрагментів зображення із метою дослідження змін кольору та освітленості. Найзагальнішим методом пошуку змін є обробка зображень за допомогою ковзкої маси, яку називають також фільтром, ядром, вікном чи шаблоном, яка представляє собою деяку квадратну матрицю. Процес базується на простому переміщенні маски фільтра від точки до точки по зображенню.

Оператор Робертса використовують в обробці зображень та комп'ютерному баченні для виявляння контурів. Він був одним із перших детекторів контурів, первинно запропонованим Лоуренсом Робертсом 1963 року. Як у різницевого оператора ідея оператора Робертса полягає в наближенні градієнта зображення шляхом дискретного диференціювання, якого досягають обчисленням суми квадратів різниць між діагонально сусідніми пікселями.

Згідно з Робертсом, детектор контурів повинен мати наступні властивості: видавані контури повинні бути чітко визначеними, тло повинне вносити якомога менше шуму, а яскравість контурів повинна якомога ближче відповідати людському сприйняттю. З огляду на ці критерії та на основі переважної на той час психофізичної теорії Робертс запропонував такі рівняння:

$$y_{i,j} = \sqrt{x_{i,j}}$$

$$z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2}$$

де x — первинне значення яскравості на зображенні, z — обчислювана похідна, а i,j подають розташування на зображенні.

Результати цієї операції висвітлюватимуть зміни яскравості в діагональному напрямку. Одним із найпривабливіших аспектів цієї операції є її простота; ядро невелике й містить лише цілі числа. Оператор Робертса сильно страждає на чутливість до шуму.

Щоби виконувати виявляння контурів за допомогою оператора Робертса, ми спершу згортаємо первинне зображення з наступними двома ядрами:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{та} \quad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Нехай *I(x,y)* — точка на первинному зображенні, *Gx(x,y)* — точка на зображенні, утвореному згортанням із першим ядром, а *Gy(x,y)* — точка на зображенні, утвореному згортанням із другим ядром. Тоді градієнт можливо визначити як

$$\nabla I(x,y) = G(x,y) = \sqrt{G_x^2 + G_y^2}$$

Напрямок градієнта також можливо визначити наступним чином:

$$\Theta(x,y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) - \frac{3\pi}{4}$$

Оператор Прюітт оператор дискретного диференціювання, який обчислює наближення градієнта функції яскравості зображення. Оператор Прюітт ґрунтується на згортанні зображення з невеликим роздільним цілочисловим фільтром в горизонтальному та вертикальному напрямках. З іншого боку, наближення градієнтна, яке він створює, відносно грубе, зокрема для високочастотних мінливостей зображення. Оператор Прюітт розробила Джудіт М. С. Прюітт.

Цей оператор обчислює градієнт яскравості зображення в кожній точці, вказуючи напрямок найбільшого збільшення яскравості від темного до світлого та швидкість цієї зміни. Таким чином, результат показує, наскільки різко або плавно змінюється зображення в конкретній точці, що дає можливість оцінити, чи є це частина контуром, і визначити напрямок контуру. Практично легше та надійніше інтерпретувати величину градієнта (яка показує правдоподібність наявності контуру), ніж напрямок.

Математично, градієнт функції двох змінних (яскравості зображення) у кожній точці зображення — це двовимірний вектор, складові якого визначаються похідними в горизонтальному та вертикальному напрямках. У кожній точці цей вектор вказує напрямок найбільшого збільшення яскравості, а його довжина — швидкість цієї зміни. Тобто, на області з сталою яскравістю вектор градієнта буде нульовим, а на контурі — вектор, що вказує перпендикулярно до контуру, від темніших до світліших областей.

З математичної точки зору цей оператор використовує два ядра 3×3, які згортають із первинним зображенням, щоб обчислити наближення похідних — одне для горизонтальних змін, інше — для вертикальних. Якщо ми визначимо **A** як первинне зображення, а *Gx* та *Gy* — два зображення, які в кожній точці містять наближення горизонтальної та вертикальної похідних, то їх обчислюють так:

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}, \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Де * позначує операцію двовимірної згортки.

Оскільки ядра Прюітт можливо розкласти як добутки усереднювального та диференціювального ядер, вони обчислюють градієнт зі згладжуванням. Отже, це роздільний фільтр. Наприклад, *Gx* можливо записати як

$$\begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

Координату x тут визначено як зростальну «ліворуч», а координату y — як зростальну «вгору». У кожній точці зображення отримані наближення градієнта можливо об'єднувати, щоб отримувати величину градієнта, використовуючи

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Використовуючи цю інформацію, ми також можемо обчислювати напрямок градієнта:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

де, наприклад, Θ дорівнює 0 для вертикального контуру, темнішого праворуч.

Оператор Собеля (англ. Sobel operator), який іноді називають оператором Собеля — Фельдмана - названо на честь Ірвіна Собеля та Гері Фельдмана, колег зі Стенфордської лабораторії штучного інтелекту. Собель та Фельдман запропонували ідею «ізотропного оператора градієнта зображення 3 × 3» на виступі в SAIL у 1968 році. Технічно це оператор дискретного диференціювання , який обчислює наближення градієнта функції яскравості зображення. У кожній точці зображення результат оператора Собеля — Фельдмана — або відповідний вектор градієнта, або норма цього вектора. Оператор Собеля — Фельдмана ґрунтується на згортанні зображення з невеликим роздільним цілочисловим фільтром в горизонтальному та вертикальному напрямках, і тому відносно невитратний з погляду обчислень. З іншого боку, наближення градієнтна, яке він створює, відносно грубе, зокрема для високочастотних мінливостей на зображенні.

Цей оператор використовує два ядра 3×3, які згортають з первинним зображенням, щоб обчислювати наближення похідних — однієї для горизонтальних змін, та однієї для вертикальних. Якщо ми визначимо *A* як первинне зображення, а *Gx* та *Gy* — два зображення, що в кожній точці містять наближення горизонтальної та вертикальної похідних відповідно, то ці обчислення такі:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}, \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

де ∗ позначує операцію згортки у двовимірній обробці сигналу.

Координату x тут визначено як зростальну «праворуч», а координату y — як зростальну «донизу». У кожній точці зображення отримані наближення градієнта можливо об'єднувати, щоб отримувати величину градієнта, використовуючи

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Використовуючи цю інформацію, ми також можемо обчислювати напрямок градієнта:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

де, наприклад, Θ дорівнює 0 для вертикального контуру, світлішого з правого боку.

Лапласіан-Гаусіан похідний оператор. Використовує маску другого порядку. До цього всі попередні методи визначали маски першого порядку. Головним завданням є підкреслення розривів в інтенсивності зображень та відсіювати області з помірною зміною. В масці є дві класифікації: позитивний та негативний оператор Лапласа. В цій згортці ми знаходимо другу похідну по x та y. Але наші краї можуть знаходитись не тільки горизонтально або вертикально, а в будь-якій орієнтації. За допомогою цієї згортки неможливо визначити відстані якщо вони знаходяться під кутом 45° відносно пікселю, бо наші кутові елементи матриці дорівнюють 0.

$$\nabla^2(\mathrm{I}) = \frac{\partial^2 \mathrm{I}}{\partial \mathrm{x}^2} + \frac{\partial^2 \mathrm{I}}{\partial \mathrm{y}^2}$$

$$\nabla^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Для того щоб виправити ситуацію та знаходити краї в незалежності від їх положення найчастіше використовують наступну покращену згортку (1.8):

$$\nabla^2 = \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 0 & 4 & 0 \end{bmatrix}$$

Canny Edge Detection — це популярний алгоритм виявлення країв. Він був розроблений Джоном Ф. Канні Це багатоетапний алгоритм, і ми пройдемо кожен етап. Першим кроком у алгоритмі виявлення країв Canny є перетворення вхідного зображення в градації сірого. Зображення у відтінках сірого мають єдиний канал, що відображає інтенсивність кожного пікселя, що спрощує процес виявлення країв і зменшує складність обчислень. Перетворення градацій сірого видаляє інформацію про колір із зображення, зберігаючи відносні рівні яскравості.

Кольорові зображення зазвичай представлені трьома каналами: червоним, зеленим і синім (RGB). Кожен канал містить інтенсивність відповідного колірного компонента в кожному пікселі. На відміну від цього, зображення у градаціях сірого містять лише один канал, де значення пікселів представляють рівні яскравості або яскравості.

Перетворення у відтінки сірого досягається за допомогою зваженої суми каналів RGB для обчислення значення інтенсивності відтінків сірого для кожного пікселя. Вагові коефіцієнти, які використовуються в цьому перетворенні, можуть змінюватися залежно від програми чи умов.

Першим кроком у алгоритмі виявлення країв Canny є застосування фільтра Гауса до вхідного зображення. Фільтр Гаусса — це операція згладжування, яка допомагає зменшити шум у зображенні. Шум може створити хибні краї, що може поставити під загрозу точність процесу виявлення країв. Фільтр Гауса згладжує зображення, згортаючи його за допомогою ядра Гауса, ефективно зменшуючи високочастотний шум, зберігаючи чіткість країв.

Математично ядро Гауса визначається як:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \cdot \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

де:

- x і y– просторові координати ядра.
- П є математичною константою Пі (приблизно 3,14159).
- σ є стандартним відхиленням, що контролює ширину розподілу Гауса.

Фільтр Гауса застосовується до кожного пікселя на зображенні шляхом ковзання ядра по всьому зображенню та отримання середнього зваженого значення інтенсивності сусідніх пікселів. Це означає, що він враховує яскравість навколишніх пікселів і надає більше значення ближчим. Отже, якщо ядро більше, воно включає більше пікселів у обчислення, що призводить до сильнішого ефекту розмиття зображення.

Після зменшення шуму алгоритм Canny переходить до обчислення градієнта згладженого зображення. Градієнт вимірює швидкість зміни інтенсивності в місці розташування кожного пікселя. Алгоритм використовує концепцію похідних, як правило, оператор Собеля, щоб визначити як величину градієнта, так і орієнтацію для кожного пікселя. Величина градієнта вказує на силу зміни інтенсивності, а орієнтація градієнта визначає напрямок найкрутішої зміни.

Тепер, коли ми обчислили величину градієнта та орієнтацію кожного пікселя, ми переходимо до критичного кроку немаксимального придушення. Цей крок ефективно стоншує краї та створює чіткіше представлення фактичних країв на зображенні.

Немаксимальне придушення в алгоритмі виявлення країв Canny працює, досліджуючи величину та орієнтацію градієнта кожного пікселя та порівнюючи його з сусідніми пікселями вздовж напрямку градієнта. Якщо величина градієнта центрального пікселя є найбільшою серед його сусідів, це означає, що цей піксель, ймовірно, є частиною краю, і ми зберігаємо його. Якщо ні, ми пригнічуємо його, встановлюючи його інтенсивність на нуль і видаляючи його з розгляду як крайовий піксель.

Наступний крок включає подвійне порогове значення для класифікації країв на три категорії: сильні краї, слабкі краї та некраї.

Для цього використовуються високий поріг і низький поріг.

Пікселі з величиною градієнта вище високого порогу вважаються сильними краями, що вказує на значні зміни інтенсивності.

Пікселі з градієнтними величинами між низьким порогом і високим порогом класифікуються як слабкі краї. Ці слабкі краї можуть представляти реальні краї або шум, і вони потребують подальшої перевірки.

Пікселі з величиною градієнта нижче нижнього порогу вважаються некраями та відкидаються.

Останнім кроком алгоритму виявлення краю Canny є відстеження краю за гістерезисом. Гістерезис означає «згадування минулого», щоб зробити наші краї більш точними та надійними. Цей крок має на меті зв’язати слабкі грані, які, ймовірно, є частиною справжніх ребер, із сильними. Починаючи з кожного сильного крайового пікселя, алгоритм відстежує край, враховуючи його сусідні слабкі крайові пікселі, які з’єднані. Якщо слабкий крайовий піксель з’єднаний із сильним крайовим пікселем, він також вважається частиною краю та зберігається. Цей процес триває до тих пір, поки слабкі краї більше не з’єднаються. Це гарантує безперервність і чіткість країв.

6.2 Порядок виконання роботи

6.2.1. Завантажити зображення;
6.2.2. Запустити код без додавання шуму та отримати зображення знаходження контурів;
6.2.3. Налаштувати детектор Canny так, щоб детекція контурів була відповідною до контурів на зображені.
6.2.4. Додати випадковий шум до зображення(Гаусівський шум, Сіль і перець, Спекл-шум) для кожного шуму запустити код один раз.
6.2.5. Повторити корок 6.2.4, але до зображення використати гаусівське згладжування(значення розмиття обрати самостійно, для коректної детекції контурів).

6.3 Зміст звіту

6.3.1. Тема лабораторної;
6.3.2. Мета лабораторної роботи;
6.3.3. Оригінальне зображення;
6.3.4. Зображення знайдених контурів відповідно до 5х методів детекції.
6.3.5. Зображення знайдених контурів відповідно до 5х методів детекції з Гаусівським/сіль і перець/спекл шумом.
6.3.6. Зображення знайдених контурів відповідно до 5х методів детекції з Гаусівським/сіль і перець/спекл шумом з Гаусівським згладжуванням.
6.3.7. Висновки.

6.4 Контрольні запитання

6.4.1. Яка основна мета використання операторів виявлення країв у зображеннях?
6.4.2. Як обчислюється градієнтне зображення у фільтрі Робертса?
6.4.3. Як можна нормалізувати результати роботи оператора Собеля?
6.4.4. Які розміри ядер використовуються в операторі Собеля?
6.4.5. Як оператор Превіта реагує на горизонтальні та вертикальні градієнти?
6.4.6. Чому перед застосуванням оператора виявлення країв рекомендується згладжувати зображення?

```
% Завантажуємо зображення та перетворюємо в відтінки сірого
image = imread('cat.jpg');
gray_image = rgb2gray(image);

% Додавання випадкового шуму
noisy_image = imnoise(gray_image, 'gaussian', 0, 0.01); % Гаусівський шум
% noisy_image = imnoise(gray_image, 'salt & pepper', 0.02); % Сіль і перець
% noisy_image = imnoise(gray_image, 'speckle', 0.05); % Спекл-шум

% Гаусовське розмиття
gaussian_blurred = imgaussfilt(noisy_image, 1);

% Фільтр Робертса
roberts_x = [1 0; 0 -1];
roberts_y = [0 1; -1 0];
edge_x_roberts = imfilter(double(gaussian_blurred), roberts_x);
edge_y_roberts = imfilter(double(gaussian_blurred), roberts_y);
edges_roberts = sqrt(edge_x_roberts.^2 + edge_y_roberts.^2);

% Фільтр Превіта
prewitt_x = [-1 0 1; -1 0 1; -1 0 1];
prewitt_y = [-1 -1 -1; 0 0 0; 1 1 1];
edge_x_prewitt = imfilter(double(gaussian_blurred), prewitt_x);
edge_y_prewitt = imfilter(double(gaussian_blurred), prewitt_y);
edges_prewitt = sqrt(edge_x_prewitt.^2 + edge_y_prewitt.^2);

% Фільтр Собеля
sobel_x = fspecial('sobel');
sobel_y = sobel_x';
edge_x_sobel = imfilter(double(gaussian_blurred), sobel_x);
edge_y_sobel = imfilter(double(gaussian_blurred), sobel_y);
edges_sobel = sqrt(edge_x_sobel.^2 + edge_y_sobel.^2);

% Лапласіан-Гаусіан
laplacian_gaussian = imfilter(gaussian_blurred, fspecial('laplacian'));

% Метод Канні
low_threshold = 100;
high_threshold = 150;
edges_canny = edge(gaussian_blurred, 'Canny', [low_threshold/255, high_threshold/255]);

% Відображення результатів
figure; imshow(noisy_image, []); title('Зображення з шумом');
figure; imshow(edges_roberts, []); title('Фільтр Робертса');
figure; imshow(edges_prewitt, []); title('Фільтр Превіта');
figure; imshow(edges_sobel, []); title('Фільтр Собеля');
figure; imshow(laplacian_gaussian, []); title('Лапласіан-Гаусіан');
figure; imshow(edges_canny); title('Метод Канні');
```