Лабораторна робота. Виділення контурів (перепадів яскравості) зі згладжуванням поверхнею другого порядку

Краснопір Тимур, ІПС-32

Вступ

Для того, аби отримати найкращий результат виділення контурів зображення, до зображення спочатку треба застосувати згладження. Згладження зображення зменшує чутливість до шуму. А це, в свою чергу, зменшує кількість фіктивних контурів, тобто контурів, отриманих з шуму.

Нижче наведено покроковий опис алгоритму виділення контурів зі згладжуванням поверхнею другого порядку. Цей алгоритм згладжує вихідне зображення поверхнею другого порядку, а вже потім виділяє контури зображення.

Побудова апроксимуючої поверхні другого порядку

Побудуємо апроксимуючу поверхню другого порядку.

```
% f(x, y) is an approximating surface of the second order
syms f(x, y) a b c alpha beta gamma
f(x, y) = a*x^2 + b*y^2 + c*x*y + alpha*x + beta*y + gamma;
```

Для побудови апроксимуючої поверхні другого порядку використаємо метод найменших квадратів. При пошуку невідомих коефіцієнтів (`a`, `b`, `c`, `alpha`, `beta` та `gamma`) будемо мінімізувати квадратичну похибку. Квадратична похибка наведена нижче.

```
% g(x, y) is a window
syms g(x, y)
epsilon = (f(x-1, y-1) - g(x-1, y-1))^2 + (f(x-1, y) - g(x-1, y))^2 + (f(x-1, y+1) - g(x-1, y+1))^2 ...
    + (f(x, y-1) - g(x, y-1))^2 + (f(x, y) - g(x, y))^2 + (f(x, y+1) - g(x, y+1))^2 ...
    + (f(x+1, y-1) - g(x+1, y-1))^2 + (f(x+1, y) - g(x+1, y))^2 + (f(x+1, y+1) - g(x+1, y+1))^2;
```

Наступну систему лінійних алгебраїчних рівнянь відносно невідомих коефіцієнтів `a`, `b`, `c`, `alpha`, `beta` та `gamma` отримаємо з необхідних умов мінімуму квадратичної похибки.

```
eqn1 = diff(epsilon, a) == 0;
eqn2 = diff(epsilon, b) == 0;
eqn3 = diff(epsilon, c) == 0;
eqn4 = diff(epsilon, alpha) == 0;
eqn5 = diff(epsilon, beta) == 0;
eqn6 = diff(epsilon, gamma) == 0;
```

Матриця A цієї системи лінійних алгебраїчних рівнянь наведена нижче.

```
[A, B] = equationsToMatrix([eqn1, eqn2, eqn3, eqn4, eqn5, eqn6], [a, b, c, alpha, beta, gamma]);
A
```

A =

$$\begin{pmatrix} 6\,(x-1)^4+6\,(x+1)^4+6\,x^4 & \sigma_1 & \sigma_5 & \sigma_9 & \sigma_2 & \sigma_7 \\ \sigma_1 & 6\,(y-1)^4+6\,(y+1)^4+6\,y^4 & \sigma_6 & \sigma_3 & \sigma_{10} & \sigma_8 \\ \sigma_5 & \sigma_6 & \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \\ \sigma_9 & \sigma_3 & \sigma_2 & \sigma_7 & \sigma_4 & 18\,x \\ \sigma_2 & \sigma_{10} & \sigma_3 & \sigma_4 & \sigma_8 & 18\,y \\ \sigma_7 & \sigma_8 & \sigma_4 & 18\,x & 18\,y & 18 \end{pmatrix}$$

where

$$\sigma_1 = 2\,x^2\,y^2+2\,x^2\,(y-1)^2+2\,x^2\,(y+1)^2+2\,y^2\,(x-1)^2+2\,y^2\,(x+1)^2+2\,(x-1)^2\,(y-1)^2+2\,(x-1)^2\,(y+1)^2+2\,(x+1)^2\,(y-1)^2+2\,(x+1)^2\,(y+1)^2$$

$$\sigma_2 = 2\,y\,(x-1)^2+2\,y\,(x+1)^2+2\,x^2\,(y-1)+2\,x^2\,(y+1)+2\,(x-1)^2\,(y-1)+2\,(x-1)^2\,(y+1)+2\,(x+1)^2\,(y-1)+2\,(x+1)^2\,(y+1)+2\,x^2\,y$$

$$\sigma_3 = 2\,x\,(y-1)^2+2\,x\,(y+1)^2+2\,y^2\,(x-1)+2\,y^2\,(x+1)+2\,(x-1)\,(y-1)^2+2\,(x-1)\,(y+1)^2+2\,(x+1)\,(y-1)^2+2\,(x+1)\,(y+1)^2+2\,x\,y^2$$

$$\sigma_4 = 2\,(x-1)\,(y-1)+2\,(x-1)\,(y+1)+2\,(x+1)\,(y-1)+2\,(x+1)\,(y+1)+2\,x\,y+2\,x\,(y-1)+2\,x\,(y+1)+2\,y\,(x-1)+2\,y\,(x+1)$$

$$\sigma_5 = 2\,y\,(x-1)^3+2\,y\,(x+1)^3+2\,x^3\,(y-1)+2\,x^3\,(y+1)+2\,(x-1)^3\,(y-1)+2\,(x-1)^3\,(y+1)+2\,(x+1)^3\,(y-1)+2\,(x+1)^3\,(y+1)+2\,x^3\,y$$

$$\sigma_6 = 2\,x\,(y-1)^3+2\,x\,(y+1)^3+2\,y^3\,(x-1)+2\,y^3\,(x+1)+2\,(x-1)\,(y-1)^3+2\,(x-1)\,(y+1)^3+2\,(x+1)\,(y-1)^3+2\,(x+1)\,(y+1)^3+2\,x\,y^3$$

$$\sigma_7 = 6\,(x-1)^2+6\,(x+1)^2+6\,x^2$$

$$\sigma_8 = 6\,(y-1)^2+6\,(y+1)^2+6\,y^2$$

$$\sigma_9 = 6\,(x-1)^3+6\,(x+1)^3+6\,x^3$$

$$\sigma_{10} = 6\,(y-1)^3+6\,(y+1)^3+6\,y^3$$

Розв'яжемо наведену вище систему лінійних алгебраїчних рівнянь.

```
coefficients = solve([eqn1, eqn2, eqn3, eqn4, eqn5, eqn6], [a, b, c, alpha, beta, gamma]);
```

Отримаємо наступний розв'язок цієї системи лінійних алгебраїчних рівнянь.

```
coefficients.a
```

```
ans =
```

$$\frac{g(x-1,y-1)}{6}+\frac{g(x-1,y+1)}{6}+\frac{g(x+1,y-1)}{6}+\frac{g(x+1,y+1)}{6}-\frac{g(x,y)}{3}-\frac{g(x,y-1)}{3}-\frac{g(x,y+1)}{3}+\frac{g(x-1,y)}{6}+\frac{g(x+1,y)}{6}$$

```
coefficients.b
```

```
ans =
```

$$\frac{g(x-1,y-1)}{6}+\frac{g(x-1,y+1)}{6}+\frac{g(x+1,y-1)}{6}+\frac{g(x+1,y+1)}{6}-\frac{g(x,y)}{3}+\frac{g(x,y-1)}{6}+\frac{g(x,y+1)}{6}-\frac{g(x-1,y)}{3}-\frac{g(x+1,y)}{3}$$

```
coefficients.c
```

```
ans =
```

$$\frac{g(x-1,y-1)}{4}-\frac{g(x-1,y+1)}{4}-\frac{g(x+1,y-1)}{4}+\frac{g(x+1,y+1)}{4}$$

```
coefficients.alpha
```

```
ans =
```

$$\frac{\sigma_3}{6}-\frac{\sigma_2}{6}-\frac{\sigma_1}{6}+\frac{\sigma_4}{6}+\frac{2\,x\,g(x,y)}{3}+\frac{2\,x\,g(x,y-1)}{3}+\frac{2\,x\,g(x,y+1)}{3}-\frac{x\,\sigma_5}{3}-\frac{x\,\sigma_6}{3}-\frac{\sigma_5}{6}+\frac{\sigma_6}{6}-\frac{x\,\sigma_1}{3}-\frac{x\,\sigma_2}{3}-\frac{x\,\sigma_3}{3}-\frac{x\,\sigma_4}{3}-\frac{y\,\sigma_1}{4}+\frac{y\,\sigma_2}{4}+\frac{y\,\sigma_3}{4}-\frac{y\,\sigma_4}{4}$$

where

$$\sigma_1 = g(x-1, y-1)$$

$$\sigma_2 = g(x-1, y+1)$$

$$\sigma_3 = g(x+1, y-1)$$

$$\sigma_4 = g(x+1, y+1)$$

$$\sigma_5 = g(x-1, y)$$

$$\sigma_6 = g(x+1, y)$$

```
coefficients.beta
```

```
ans =
```

$$\frac{\sigma_2}{6}-\frac{\sigma_1}{6}-\frac{\sigma_3}{6}+\frac{\sigma_4}{6}+\frac{2\,y\,g(x,y)}{3}-\frac{y\,\sigma_5}{3}-\frac{y\,\sigma_6}{3}+\frac{2\,y\,g(x-1,y)}{3}+\frac{2\,y\,g(x+1,y)}{3}-\frac{\sigma_5}{6}+\frac{\sigma_6}{6}-\frac{x\,\sigma_1}{4}+\frac{x\,\sigma_2}{4}+\frac{x\,\sigma_3}{4}-\frac{x\,\sigma_4}{4}-\frac{y\,\sigma_1}{3}-\frac{y\,\sigma_2}{3}-\frac{y\,\sigma_3}{3}-\frac{y\,\sigma_4}{3}$$

where

$$\sigma_1 = g(x-1, y-1)$$

$$\sigma_2 = g(x-1, y+1)$$

$$\sigma_3 = g(x+1, y-1)$$

$$\sigma_4 = g(x+1, y+1)$$

$$\sigma_5 = g(x, y-1)$$

$$\sigma_6 = g(x, y+1)$$

```
coefficients.gamma
```

```
ans =
```

$$\frac{x^2\,\sigma_1}{6}-\frac{\sigma_2}{9}-\frac{\sigma_3}{9}-\frac{\sigma_4}{9}-\frac{\sigma_1}{9}+\frac{x^2\,\sigma_2}{6}+\frac{x^2\,\sigma_3}{6}+\frac{x^2\,\sigma_4}{6}+\frac{y^2\,\sigma_1}{6}+\frac{y^2\,\sigma_2}{6}+\frac{y^2\,\sigma_3}{6}+\frac{y^2\,\sigma_4}{6}+\frac{5\,g(x,\,y)}{9}+\frac{x\,\sigma_5}{6}-\frac{x\,\sigma_6}{6}-\frac{x^2\,g(x,\,y)}{3}+\frac{y\,\sigma_7}{6}-\frac{y\,\sigma_8}{6}-\frac{y^2\,g(x,\,y)}{3}+\frac{2\,\sigma_7}{9}+\frac{2\,\sigma_8}{9}+\frac{2\,\sigma_5}{9}+\frac{2\,\sigma_6}{9}+\frac{x\,\sigma}{6}$$

where

$$\sigma_1 = g(x-1,\,y-1)$$

$$\sigma_2 = g(x-1,\,y+1)$$

$$\sigma_3 = g(x+1,\,y-1)$$

$$\sigma_4 = g(x+1,\,y+1)$$

$$\sigma_5 = g(x-1,\,y)$$

$$\sigma_6 = g(x+1,\,y)$$

$$\sigma_7 = g(x,\,y-1)$$

$$\sigma_8 = g(x,\,y+1)$$

Модуль градієнта

Формула модуля градієнта наведена нижче.

```
% grad(x, y) is |grad(x, y)|
syms grad(x, y)
grad(x, y) = sqrt((2*(coefficients.a)*x + (coefficients.c)*y + coefficients.alpha)^2 ...
    + (2*(coefficients.b)*y + (coefficients.c)*x + coefficients.beta)^2);
```

Застосування алгоритму до зображення `cameraman.tif`

Зчитаємо зображення з графічного файлу **`cameraman.tif`**.

```
originalImage = imread('cameraman.tif');
imshow(originalImage);
```

```
originalImage = double(originalImage);
```

Створимо нове зображення, у яке запишемо результат.

```
resultImage = zeros(size(originalImage), class(originalImage));
```

Для кожної точки вихідного зображення `originalImage` обрахуємо модуль градієнта (за наведеної вище формулою). Запишемо отримані значення у нове зображення `resultImage`.

Нижче формула модуля градієнта записана після підстановки у цю формулу знайдених значень відповідних коефіцієнтів.

```
% x is a row, y is a column
for x = 2:(size(originalImage, 1) - 1)
    for y = 2:(size(originalImage, 2) - 1)
        % the statement below is equivalent to: resultImage(x, y) = grad(x, y);
        resultImage(x, y) = sqrt((originalImage(x - 1, y - 1)/6 - y*(originalImage(x - 1, y - 1)/4 ...
                - originalImage(x - 1, y + 1)/4 - originalImage(x + 1, y - 1)/4 + originalImage(x + 1, y + 1)/4) ...
                + originalImage(x - 1, y + 1)/6 - originalImage(x + 1, y - 1)/6 - originalImage(x + 1, y + 1)/6 ...
                - x*(originalImage(x - 1, y - 1)/3 + originalImage(x - 1, y + 1)/3 + originalImage(x + 1, y - 1)/3 ...
                + originalImage(x + 1, y + 1)/3 - (2*originalImage(x, y))/3 - (2*originalImage(x, y - 1))/3 ...
                - (2*originalImage(x, y + 1))/3 + originalImage(x - 1, y)/3 + originalImage(x + 1, y)/3) ...
                - (2*x*originalImage(x, y))/3 - (2*x*originalImage(x, y - 1))/3 - (2*x*originalImage(x, y + 1))/3 ...
                + (x*originalImage(x - 1, y))/3 + (x*originalImage(x + 1, y))/3 + originalImage(x - 1, y)/6 ...
                - originalImage(x + 1, y)/6 + (x*originalImage(x - 1, y - 1))/3 + (x*originalImage(x - 1, y + 1))/3 ...
                + (x*originalImage(x + 1, y - 1))/3 + (x*originalImage(x + 1, y + 1))/3 + (y*originalImage(x - 1, y - 1))/4 ...
                - (y*originalImage(x - 1, y + 1))/4 - (y*originalImage(x + 1, y - 1))/4 + (y*originalImage(x + 1, y + 1))/4)^2 ...
                + (originalImage(x - 1, y - 1)/6 - x*(originalImage(x - 1, y - 1)/4 - originalImage(x - 1, y + 1)/4 ...
                - originalImage(x + 1, y - 1)/4 + originalImage(x + 1, y + 1)/4) - originalImage(x - 1, y + 1)/6 ...
                + originalImage(x + 1, y - 1)/6 - originalImage(x + 1, y + 1)/6 - y*(originalImage(x - 1, y - 1)/3 ...
                + originalImage(x - 1, y + 1)/3 + originalImage(x + 1, y - 1)/3 + originalImage(x + 1, y + 1)/3 ...
                - (2*originalImage(x, y))/3 + originalImage(x, y - 1)/3 + originalImage(x, y + 1)/3 ...
                - (2*originalImage(x - 1, y))/3 - (2*originalImage(x + 1, y))/3) - (2*y*originalImage(x, y))/3 ...
                + (y*originalImage(x, y - 1))/3 + (y*originalImage(x, y + 1))/3 - (2*y*originalImage(x - 1, y))/3 ...
                - (2*y*originalImage(x + 1, y))/3 + originalImage(x, y - 1)/6 - originalImage(x, y + 1)/6 ...
                + (x*originalImage(x - 1, y - 1))/4 - (x*originalImage(x - 1, y + 1))/4 - (x*originalImage(x + 1, y - 1))/4 ...
                + (x*originalImage(x + 1, y + 1))/4 + (y*originalImage(x - 1, y - 1))/3 + (y*originalImage(x - 1, y + 1))/3 ...
                + (y*originalImage(x + 1, y - 1))/3 + (y*originalImage(x + 1, y + 1))/3)^2);
    end
end
```

Результат застосування алгоритму до зображення **`cameraman.tif`** наведено нижче.

```
imshow(uint8(resultImage));
```

Результат у негативі наведено нижче.

```
imshow(255 - uint8(resultImage)); % negative resultImage
```

Визначення часу роботи програми

Загальний час (тобто з урахуванням часу на знаходження невідомих коефіцієнтів та формули модуля градієнта) роботи програми складає `0.7193` секунди (визначено за допомогою функції `timeit`).

Але знаходження формули модуля градієнта (що включає знаходження невідомих коефіцієнтів) необхідно виконати лише один раз. А вже далі застосовувати цю формулу для виділення контурів того чи іншого зображення. Тобто при виділенні контурів того чи іншого зображення підставляти у цю формулу модуля градієнта конкретні значення з вікна та конкретні координати `x` та `y`.

Тому доцільно за загальний час роботи програми обрати час роботи функції **`applyEdgeDetectionAlgorithm`**, код якої наведено нижче. Тобто не враховувати час, що затрачено на знаходження формули модуля градієнта.

Нижче наведено час роботи функції `applyEdgeDetectionAlgorithm` (час зазначено в секундах).

```
edgeDetectionAlgorithm = @() applyEdgeDetectionAlgorithm(originalImage);
time = timeit(edgeDetectionAlgorithm)
```

```
time = 0.0092
```

Код функції **`applyEdgeDetectionAlgorithm`** наведено нижче.

```
function resultImage = applyEdgeDetectionAlgorithm(originalImage)
    resultImage = zeros(size(originalImage), class(originalImage));
    % x is a row, y is a column
    for x = 2:(size(originalImage, 1) - 1)
        for y = 2:(size(originalImage, 2) - 1)
            % the statement below is equivalent to: resultImage(x, y) = grad(x, y);
            resultImage(x, y) = sqrt((originalImage(x - 1, y - 1)/6 - y*(originalImage(x - 1, y - 1)/4 ...
                - originalImage(x - 1, y + 1)/4 - originalImage(x + 1, y - 1)/4 + originalImage(x + 1, y + 1)/4) ...
                + originalImage(x - 1, y + 1)/6 - originalImage(x + 1, y - 1)/6 - originalImage(x + 1, y + 1)/6 ...
                - x*(originalImage(x - 1, y - 1)/3 + originalImage(x - 1, y + 1)/3 + originalImage(x + 1, y - 1)/3 ...
                + originalImage(x + 1, y + 1)/3 - (2*originalImage(x, y))/3 - (2*originalImage(x, y - 1))/3 ...
                - (2*originalImage(x, y + 1))/3 + originalImage(x - 1, y)/3 + originalImage(x + 1, y)/3) ...
                - (2*x*originalImage(x, y))/3 - (2*x*originalImage(x, y - 1))/3 - (2*x*originalImage(x, y + 1))/3 ...
                + (x*originalImage(x - 1, y))/3 + (x*originalImage(x + 1, y))/3 + originalImage(x - 1, y)/6 ...
                - originalImage(x + 1, y)/6 + (x*originalImage(x - 1, y - 1))/3 + (x*originalImage(x - 1, y + 1))/3 ...
                + (x*originalImage(x + 1, y - 1))/3 + (x*originalImage(x + 1, y + 1))/3 + (y*originalImage(x - 1, y - 1))/4 ...
                - (y*originalImage(x - 1, y + 1))/4 - (y*originalImage(x + 1, y - 1))/4 + (y*originalImage(x + 1, y + 1))/4)^2 ...
                + (originalImage(x - 1, y - 1)/6 - x*(originalImage(x - 1, y - 1)/4 - originalImage(x - 1, y + 1)/4 ...
                - originalImage(x + 1, y - 1)/4 + originalImage(x + 1, y + 1)/4) - originalImage(x - 1, y + 1)/6 ...
                + originalImage(x + 1, y - 1)/6 - originalImage(x + 1, y + 1)/6 - y*(originalImage(x - 1, y - 1)/3 ...
                + originalImage(x - 1, y + 1)/3 + originalImage(x + 1, y - 1)/3 + originalImage(x + 1, y + 1)/3 ...
                - (2*originalImage(x, y))/3 + originalImage(x, y - 1)/3 + originalImage(x, y + 1)/3 ...
                - (2*originalImage(x - 1, y))/3 - (2*originalImage(x + 1, y))/3) - (2*y*originalImage(x, y))/3 ...
                + (y*originalImage(x, y - 1))/3 + (y*originalImage(x, y + 1))/3 - (2*y*originalImage(x - 1, y))/3 ...
                - (2*y*originalImage(x + 1, y))/3 + originalImage(x, y - 1)/6 - originalImage(x, y + 1)/6 ...
                + (x*originalImage(x - 1, y - 1))/4 - (x*originalImage(x - 1, y + 1))/4 - (x*originalImage(x + 1, y - 1))/4 ...
                + (x*originalImage(x + 1, y + 1))/4 + (y*originalImage(x - 1, y - 1))/3 + (y*originalImage(x - 1, y + 1))/3 ...
                + (y*originalImage(x + 1, y - 1))/3 + (y*originalImage(x + 1, y + 1))/3)^2);
        end
    end
end
```