**Зчитування зображень на C#.docx -** <https://habr.com/ru/post/196578/>

Перш ніж реалізовувати алгоритм сегментації на мові програмування високого рівня c#, необхідно було дослідити різні методи зчитування зображення, тобто його перетворення до матриці структур Color.

В усіх методах використовується класичний клас Bitmap (System.Drawing.Bitmap). Даний клас зручний тим, що приховує від нас деталі кодування растрових форматів, підтримуючи всі поширені формати, типу BMP, GIF, JPEG, PNG.

При тестах послідовно виконуються 100 зчитувань зображення розмірами 1000\*1000 пікселів. Процесор – Intel Core i5-8600k. При інтервальному оцінюванні, ймовірність похибки першого роду

**«Стандартний» метод**

Суть методу полягає у використанні методу Bitmap.GetPixel(x, y). Наведемо повністю код методу, який конвертує вміст Bitmap в тривимірний масив. При цьому перша розмірність - це колірна компонента (від 0 до 2), друга - позиція y, третя - позиція x.

public static double[,,] BitmapToDoubleRgbNaive(Bitmap bmp)

{

int width = bmp.Width,

height = bmp.Height;

double [,,] res = new double[3, height, width];

for (int y = 0; y < height; ++y)

{

for (int x = 0; x < width; ++x)

{

Color color = bmp.GetPixel(x, y);

res[0, y, x] = color.R;

res[1, y, x] = color.G;

res[2, y, x] = color.B;

}

}

return res;

}

Результати

**Загальний час тесту: 72,4922 секунди.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Нижнє значення** | **Значення** | **Верхнє значення** | **Sigma** |
| **Мат. Сподівання** | 0.7247 | 0.7262 | 0.7278 | 0.0008 |
| **Середньоквадратичне відхилення** | 0.0070 | 0.0081 | 0.0092 | 0.0005 |

**Пряма робота з даними Bitmap за допомогою вказівників**

Клас Bitmap надає більш швидкий спосіб звернутися до своїх даних. Для цього нам необхідно скористатися посиланнями, наданими класом BitmapData та арифметикою вказівників:

public unsafe static double[,,] BitmapToDoubleRgb(Bitmap bmp)

{

int width = bmp.Width,

height = bmp.Height;

double[,,] res = new double[3, height, width];

BitmapData bd = bmp.LockBits(new Rectangle(0, 0, width, height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);

try

{

byte\* curpos;

fixed (double\* \_res = res)

{

double\* \_r = \_res,

\_g = \_res + width \* height,

\_b = \_res + 2 \* width \* height;

for (int h = 0; h < height; h++)

{

curpos = ((byte\*)bd.Scan0) + h \* bd.Stride;

for (int w = 0; w < width; w++)

{

\*\_b = \*(curpos++); ++\_b;

\*\_g = \*(curpos++); ++\_g;

\*\_r = \*(curpos++); ++\_r;

}

}

}

}

finally

{

bmp.UnlockBits(bd);

}

return res;

}

Результати

**Загальний час тесту:** 1,7823 **секунди.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Нижнє значення** | **Значення** | **Верхнє значення** | **Sigma** |
| **Мат. Сподівання** | 0.0168 | 0.0178 | 0.0189 | 0.0006 |
| **Середньоквадратичне відхилення** | 0.0048 | 0.0056 | 0.0063 | 0.0004 |

**Незважаючи на те, що перший метод є дуже простим у застосуванні, використовувати його як модуль у сегментації зображень немає сенсу, через його низьку ефективність порівняно з другим методом.**