Формирование четырёх компонент-изображений с помощью разложения Хаара

**Математическая постановка**

Изображение делится на блоки 2x2, далее в каждом блоке вычисляется значение нового изображение, ширина и высота которого будут в 2 раза меньше исходного.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |

4 компонента формируются следующим образом:

1. Берётся среднее арифметическое значение из блока 2x2, т.е. изображение просто сжимается (LL.bmp на выходе в программе)

|  |  |  |
| --- | --- | --- |
| (1 + 2 + 7 + 8) / 4 | (3 + 4 + 9 + 10) / 4 | (5 + 6 + 11 + 12) / 4 |
| (13 + 14 + 19 + 20) / 4 | (15 + 16 + 21 + 22) / 4 | (17 + 18 + 23 + 24) / 4 |

1. Разложение по строке. Значения формируются из модуля разности «правого верхнего» и «левого верхнего» пикселя в блоке.

|  |  |  |
| --- | --- | --- |
| |2 – 1| | |4 – 3| | |6 - 5| |
| |14 – 13| | |16 - 15| | |18 - 17| |

1. Разложение по столбцу. Значения формируются из модуля разности «левого нижнего» и «левого верхнего» пикселя в блоке

|  |  |  |
| --- | --- | --- |
| |7 – 1| | |9 – 3| | |11 - 5| |
| |19 – 13| | |21 - 15| | |23 - 17| |

1. Разложение по диагонали. Значения формируются из модуля разности «правого нижнего» и «правого верхнего» пикселя в блоке

|  |  |  |
| --- | --- | --- |
| |8 – 1| | |10 - 3| | |12 - 5| |
| |14 – 7| | |16 - 9| | |18 - 11| |

**Функция разложения Хаара на C++**

void haar(

    std::ofstream &output\_LL,

    std::ofstream &output\_LH,

    std::ofstream &output\_HL,

    std::ofstream &output\_HH,

    const std::vector<char> &pixels)

{

    std::vector<char> LL\_pixels(new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    std::vector<char> LH\_pixels(new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    std::vector<char> HL\_pixels(new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    std::vector<char> HH\_pixels(new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    for (int i = 0; i < height - 2; i += 2)

    {

        for (int j = 0; j < width - 2; j += 2)

        {

            for (int k = 0; k < colorsChannels; k++)

            {

                int a = (int)(unsigned char)pixels[i \* row\_size + j \* colorsChannels + k];

                int b = (int)(unsigned char)pixels[i \* row\_size + (j + 1) \* colorsChannels + k];

                int c = (int)(unsigned char)pixels[(i + 1) \* row\_size + j \* colorsChannels + k];

                int d = (int)(unsigned char)pixels[(i + 1) \* row\_size + (j + 1) \* colorsChannels + k];

                int LL = round((a + b + c + d) / 4.0);

                int LH = abs(b - a);

                int HL = abs(c - a);

                int HH = abs(d - a);

                LL\_pixels[(i / 2) \* new\_row\_size + (j / 2) \* colorsChannels + k] = static\_cast<char>(LL);

                LH\_pixels[(i / 2) \* new\_row\_size + (j / 2) \* colorsChannels + k] = static\_cast<char>(LH);

                HL\_pixels[(i / 2) \* new\_row\_size + (j / 2) \* colorsChannels + k] = static\_cast<char>(HL);

                HH\_pixels[(i / 2) \* new\_row\_size + (j / 2) \* colorsChannels + k] = static\_cast<char>(HH);

            }

        }

    }

    output\_LL.write(LL\_pixels.data(), new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    output\_LH.write(LH\_pixels.data(), new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    output\_HL.write(HL\_pixels.data(), new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    output\_HH.write(HH\_pixels.data(), new\_row\_size \* (height / 2) + (width / 2) \* colorsChannels);

    output\_LL.close();

    output\_LH.close();

    output\_HL.close();

    output\_HH.close();

}

**Результаты**



*Обрабатываемое чёрно-белое изображение:*



1. *Разложение как среднее арифметическое*



1. *Разложение по строкам*

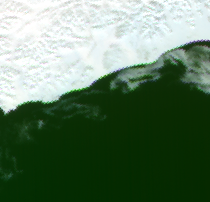


1. *Разложение по столбцам*

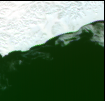


1. *Разложение по диагоналям*

Результат для цветного трёхканального изображения



*Обрабатываемое изображение*



1. *Разложение как среднее арифметическое*



1. *Разложение по строкам*



1. *Разложение по столбцам*



1. *Разложение по диагоналям*