Implemenetacja filtru medianowego

Akceleracja Algorytmów Wizyjnych

Jan Rosa 19.03.2025

Obrazy wejściowe

Figure 1: Salted Input 1024x1024

Figure 2: Salted Input 256x256

Figure 3: Salted Input 4096x4096

Kernel

```
void bubbleSort(int arr[], int n);

void swap(int* arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {

        // Last i elements are already in place, so the loop
        // will only num n - i - 1 times
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1])
                swap(arr, j, j + 1);
        }
    }
}

#define ELLIPSE 1
#define ELLIPSE_DIMX 7u
#define ELLIPSE_DIMY 7u

__constant sampler_t imageSampler = CLK_NORMALIZED_COORDS_FALSE |
CLK_ADDRESS_CLAMP | CLK_FILTER_LINEAR;

__kernel void sobel_filter(__read_only image2d_t inputImage, __write_only
image2d_t outputImage)
{
      const int2 coord = (int2)(get_global_id(0), get_global_id(1));

      int2 square_dim = (int2)(ELLIPSE_DIMX, ELLIPSE_DIMY);

      int sort_buffer[ELLIPSE_DIMX*ELLIPSE_DIMY] = {0};

      uint4 pixel = (uint4)(0);
      //if(coord.x == 50 && coord.y == 50 )
      {
            for(int row = (coord.x) - (square_dim.x/(int)2); row <= (coord.x) +
(square_dim.x/(int)2); ++row){
                  for(int col = coord.y - (square_dim.y/(int)2); col <= coord.y
+ (square_dim.y/(int)2); ++col){

                        pixel = read_imageui(inputImage, imageSampler, (int2)
(row, col));
#if ELLIPSE
                        int x2 = (row - coord.x)*(row - coord.x);
                        int a2 = (square_dim.x*square_dim.x/4);
                        int y2 = (col - coord.y)*(col - coord.y);
                        int b2 = (square_dim.y*square_dim.y/4);

                        if(   ((float)x2)/((float)a2) + ((float)y2)/((float)b2)
<= 1)
#endif
                        {
                              size_t index = ELLIPSE_DIMX*(row-coord.x+
(square_dim.x/(int)2))+(col-coord.y+(square_dim.y/(int)2));
                              sort_buffer[index] = (int)pixel.x;
                        }
                  }
```

```
            }
        }

        bubbleSort(sort_buffer, ELLIPSE_DIMX*ELLIPSE_DIMY);
        size_t middle = sizeof(sort_buffer)/(2*sizeof(sort_buffer[0]))+1 +
ELLIPSE_DIMX*ELLIPSE_DIMY*ELLIPSE/4;
        int median_val = sort_buffer[middle];


        uint4 outpixel = (uint4){median_val, median_val, median_val, 0};

        //write_imageui(outputImage, coord, convert_uint4(pixel));
        write_imageui(outputImage, coord, outpixel);

}
```

Obrazy wyjściowe różny rozmiar

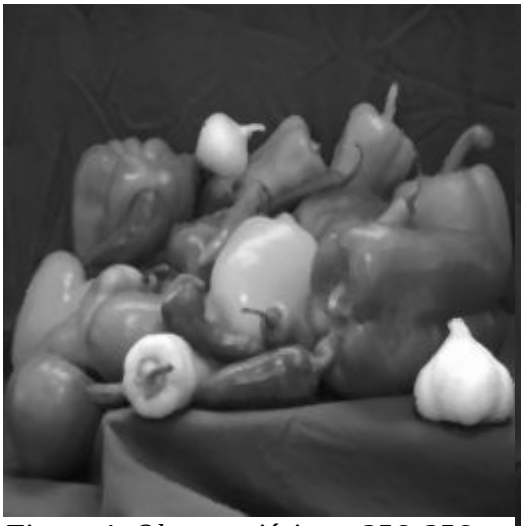

Figure 4: Obraz wyjściowy 256x256

Figure 5: Obraz wyjściowy 1024x1024

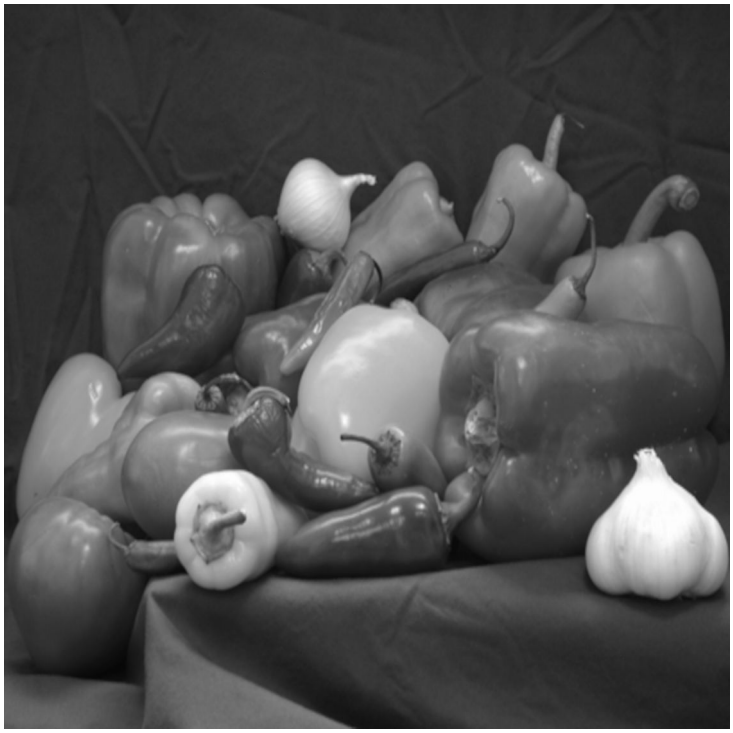

Figure 6: Obraz wyjściowy 4096x4096

Obrazy wyjściowe różny rozmiar okien mediany

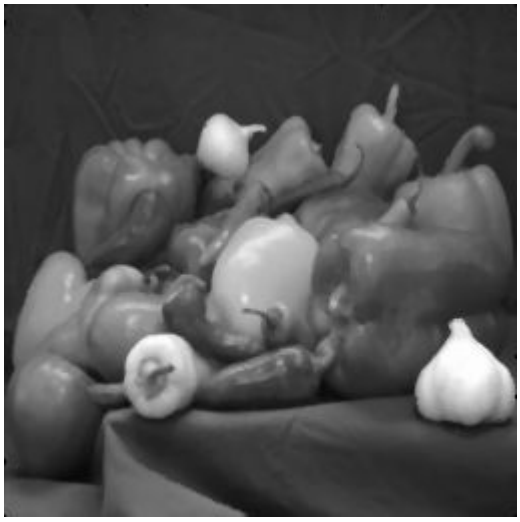

Figure 7: Obraz wyjściowy - mediana o promieniu 2px

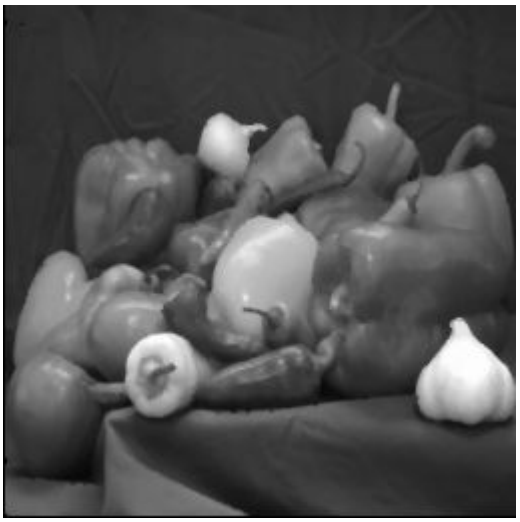

Figure 8: Obraz wyjściowy - mediana o promieniu 3px

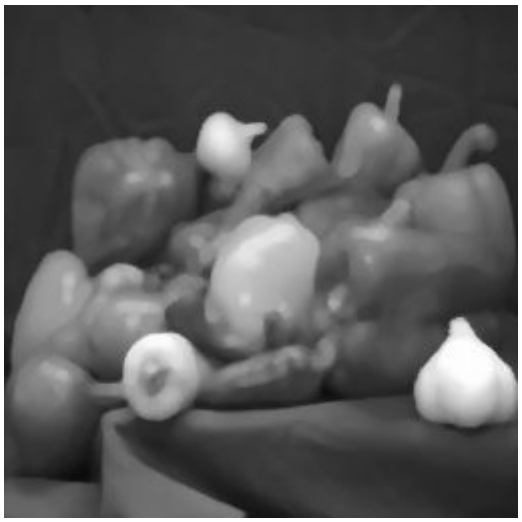

Figure 9: Obraz wyjściowy - mediana o promieniu 4px

Pomiar czasu

Dla okna medianowego o promieniu 4px.

Rozdzielczość	Czas (s) MATLAB	Czas (s) OpenCL CPU	[Transfer+Kernel] Czas (s) OpenCL CPU
256x256	0.099491	0.073843	0.023945
1024x1024	0.489014	0.380892	0.333842
4096x4096	6.254640	5.261800	5.215170