ДОКУМЕНТАЦИЈА ЗА ПРЕДМЕТНИ ПРОЈЕКАТ ИЗ ПРЕДМЕТА ПАРАЛЕЛНО ПРОГРАМИРАЊЕ

Аутор: Даница Газдић
Број индекса: SV 12/2020
Асистент: Стефан Пијетловић
Факултет техничких наука

Нови Сад, 2022.

Садржај

1. Анализа проблема

Пројекат се бави проблематиком дигиталне обраде слика, тачније, детекцијом ивица унутар слике идентификацијом оштрих промена у осветљењу. Операција се своди на аритметметичке операције двеју матрица од којих је једна матрица филтера, а друга подматрица улазне слике која садржи податке о вредностима боја пиксела слике.

$$Y[x,y] = \sum_{n=0}^{2} \sum_{m=0}^{2} X[x-1+m, y-1+n] * F[m,n]$$

Слика 1: Формула за филтрирање слике која се налази у матрици X, филтером језгра F, и смештање резултата у матрицу Y

Потребно је имплементирати детекцију ивица унутар слике помоћу „Prewitt“ оператора и помоћу најуже околине и паралелизовати их користећи OneTBB задатке.

2. Концепт решења

2.1. Детекција ивица слике помоћу „Prewitt” оператора

„Prewitt” оператор представља две матрице, вертикалну и хоризонталну (димензија 3x3, 5x5…) којима филтрирамо улазну слику тако што множимо елементе ових матрица са одговарајућим елементима подматрице улазне слике. Одговарајућу тачку излазне слике добијамо збиром апсолутних вредности хоризонталне и вертикалне компоненте. Да би излазна слика била црно – бела, потребно је резултујућу вредност скалирати у односу на глобални праг T = 128 (вредности испод прага постављамо на белу боју – 0, а вредности изнад глобалног прага на црну – 255).

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Слика 2: Вертикални и хоризонтални 3x3 "Prewitt" филтер (респективно)

2.2. Детекција ивица слике помоћу најуже околине

Детекција ивица слике помоћу најуже околине се одвија на следећи начин. За сваку тачку (пиксел) улазне слике потребно је претражити околину која је унапред задата и то:

- P(i, j) = 1, ако у околини тачке постоји тачка са вредношћу већом од глобалног прага
- P(i, j) = 0, ако у околини тачке не постоји тачка са вредношћу већом од глобалног прага
- O(i, j) = 0, ако у околини тачке постоји тачка са вредношћу мањом од глобалног прага
- O(i, j) = 1, ако у околини тачке не постоји тачка са вредношћу мањом од глобалног прага

Резултујућа вредност излазне тачке се рачуна као апсолутна вредност разлике вредности P(i, j) и O(i, j), а тако добијен резултат се скалира на вредности 0-255 (0 на 0, 1 на 255).

3. Програмско решење

3.1. Детекција ивица слике помоћу „Prewitt" оператора

За потребе рачунања ивица слике уз помоћ „Prewitt" оператора дефинисане су глобалне променљиве које представљају хоризонтални и вертикални филтер у димензијама 3x3 и 5x5.

```
// Prewitt operators
int filterHor5[FILTER_SIZE5 * FILTER_SIZE5] = {9, 9, 9, 9, 9, 9, 5, 5, 5, 9, -7, -3, 0, -3, -7, -7, -3, -3, -3, -7, -7, -7, -7, -7, -7};
int filterVer5[FILTER_SIZE5 * FILTER_SIZE5] = {9, 9, -7, -7, -7, 9, 5, -3, -3, -7, 9 ,5, 0 ,-3, -7, 9, 5, -3, -3, -7, 9, 9, -7, -7, -7};
int filterHor3[FILTER_SIZE3 * FILTER_SIZE3] = { -1, 0, 1, -1, 0, 1, -1, 0, 1 };
int filterVer3[FILTER_SIZE3 * FILTER_SIZE3] = { -1, -1, -1, 0, 0, 0, 1, 1, 1 };
```

Слика 3: Дефинисање глобалних променљивих "Prewitt" филтера

3.1.1. Серијска детекција ивица слике помоћу „Prewitt" оператора

Функција за серијску детекцију ивица уз помоћ „Prewitt" оператора (*Слика 4*) као параметре прима бафер улазне слике, бафер излазне слике, ширину и висину слике, хоризонтални и вертикални „Prewitt" филтер, као и величину већ поменутих филтера.

Функција има 4 „for" петље, које итерирају подматрицама улазне слике, односно матрицама глобално дефинисаних филтера. Вертикална и хоризонтална компонента излазног пиксела се рачунају у оквиру последње, унутрашње петље и то по горе наведеној формули (С*лика 1).*

Излазна вредност се рачуна као збир апсолутних вредности вертикалне и хоризонталне компоненте, а затим уписује на одговарајуће место у излазном баферу као скалирана вредност у односу на глобалну константу „THRESHOLD".

```
void filter_serial_prewitt(int *inBuffer, int *outBuffer, int width, int height, int *filterHor, int *filterVer, int f
{
    int margin = filterSize;
    for (int i = margin; i < width - margin; i++) {
        for (int j = margin; j < height - margin; j++) {
            int Gx = 0;
            int Gy = 0;
            int G = 0;
            for (int m = 0; m < filterSize; m++) {
                for (int n = 0; n < filterSize; n++) {
                    Gx += inBuffer[(j + n - 1) * width + (i + m - 1)] * filterHor[n * filterSize + m];
                    Gy += inBuffer[(j + n - 1) * width + (i + m - 1)] * filterVer[n * filterSize + m];
                }
            }

            G = abs(Gx) + abs(Gy);
            if (G > THRESHOLD) {
                outBuffer[j * width + i] = 255;
            }
            else {
                outBuffer[j * width + i] = 0;
            }
        }
    }
}
```

Слика 4: Функција за серијску детекцију ивица уз помоћ "Prewitt" оператора

3.1.2. Паралелна детекција ивица слике помоћу „Prewitt" оператора

Функција за паралелну детекцију ивица као параметре такође прима бафер улазне и излазне слике, тренутну висину и ширину дела слике, ред и колону од које је потребно започети рачунање, укупну ширину и висину целе слике, као и величину филтера и хоризонтални и вертикални филтер одређених димензија.

Функција, као што и сам назив каже, паралелизује функцију описану у тачки 3.1.1. и то уз помоћ TBB задатака.

На самом почетку се проверава да ли је тренутна ширина дела слике мања од унапред задате вредности за ширину. Уколико није, користимо TBB шаблонкласу „task_group" и њену методу „run" како бисмо измрестили четири нова задатка која деле тренутну слику на четири једнака дела (*Слика 5*), а уколико јесте, прво морамо подесити маргине. Маргине се подешавају како индексирање матрице улазне слике не би излазило ван опсега у ивичним случајевима. Након подешавања маргина, рачунање се одвија као у серијском примеру.

```
else {
    int widthMiddle = width / 2;
    int heightMiddle = height / 2;
    g.run([&] {filter_parallel_prewitt(inBuffer, outBuffer, widthMiddle, heightMiddle, row, col, fullWidth, fullHeight, filterHor, filterVer, filterSize);});
    g.run([&] {filter_parallel_prewitt(inBuffer, outBuffer, widthMiddle, heightMiddle, row + widthMiddle, col, fullWidth, fullHeight, filterHor, filterVer, filterSize);});
    g.run([&] {filter_parallel_prewitt(inBuffer, outBuffer, widthMiddle, heightMiddle, row, col + heightMiddle, fullWidth, fullHeight, filterHor, filterVer, filterSize);});
    g.run([&] {filter_parallel_prewitt(inBuffer, outBuffer, widthMiddle, heightMiddle, row + widthMiddle, col + heightMiddle, fullWidth, fullHeight, filterHor, filterVer, filterSize);});
    g.wait();
}
```

Слика 5: Мрешћење задатака TBB шаблонкласом task_group

3.2. Детекција ивица слике помоћу најуже околине

3.2.1. Серијска детекција ивица слике помоћу најуже околине

Функција за серијску детекцију ивица као параметре прима бафер улазне и излазне слике, ширину и висину слику и димензију матрице која представља околину пиксела за који рачунамо излазну вредност.

Анализа овог алгоритма је описана у тачки 2.2, а ова функција само преводи логику алгоритма у прогрмаски код.

```
void filter_serial_edge_detection(int *inBuffer, int *outBuffer, int width, int height, int surrounding)
{
    for (int i = surrounding; i < width - surrounding; i++) {
        for (int j = surrounding; j < height - surrounding; j++) {
            int P = 0;
            int O = 1;
            int F = 0;
            for (int m = 0; m < surrounding; m++) {
                for (int n = 0; n < surrounding; n++) {
                    if (inBuffer[(j + n - 1) * width + (i + m - 1)] > THRESHOLD) {
                        P = 1;
                    }
                    if (inBuffer[(j + n - 1) * width + (i + m - 1)] < THRESHOLD) {
                        O = 0;
                    }
                }
            }
            F = abs(P - O);
            if (F == 1) {
                outBuffer[j * width + i] = 255;
            }
            else {
                outBuffer[j * width + i] = 0;
            }
        }
    }
}
```

Слика 6: Функција за серијску детекцију ивица помоћу најуже околине

3.2.2. Паралелна детекција ивица слике помоћу најуже околине

Ова функција као параметре прима бафер улазне и излазне слике, тренутну ширину и висину слике, тренутни ред и колону од којих почиње обрада, укупну ширину и величину целе слике и димензију матрице која представља околину пиксела за који рачунамо излазну вредност.

Функција за детекцију ивица слике помоћу најуже околине је паралелизована на сличан начин као и функција за детекцију ивица слике помоћу „Prewitt“ оператора. (3.1.2.)

Прво се проверава да ли је тренутна ширина дела слике мања од унапред задате вредности ширине. Уколико није, измрешћују се четири задатка помоћу TBB шаблонкласе и њене методе „run”. Уколико ширина јесте мања од задате вредности, прво је потребно подесити маргине како ивични случајеви обраде не би излазили ван граница подматрице улазног бафера.
Након подешавања маргина рачунање се наставља као и у серијској верзији алгоритма.

```
else {
    int widthMiddle = width / 2;
    int heightMiddle = height / 2;
    g.run([&] {filter_parallel_edge_detection(inBuffer, outBuffer, widthMiddle, heightMiddle, row, col, fullWidth, fullHeight, surrounding);});
    g.run([&] {filter_parallel_edge_detection(inBuffer, outBuffer, widthMiddle, heightMiddle, row + widthMiddle, col, fullWidth, fullHeight, surrounding);});
    g.run([&] {filter_parallel_edge_detection(inBuffer, outBuffer, widthMiddle, heightMiddle, row, col + heightMiddle, fullWidth, fullHeight, surrounding);});
    g.run([&] {filter_parallel_edge_detection(inBuffer, outBuffer, widthMiddle, heightMiddle, row + widthMiddle, col + heightMiddle, fullWidth, fullHeight, surrounding);});
    g.wait();
}
```

Слика 7: Мрешћење задатака TBB шаблонкласом task group

4. Тестирање

4.1. Карактеристике рачунара

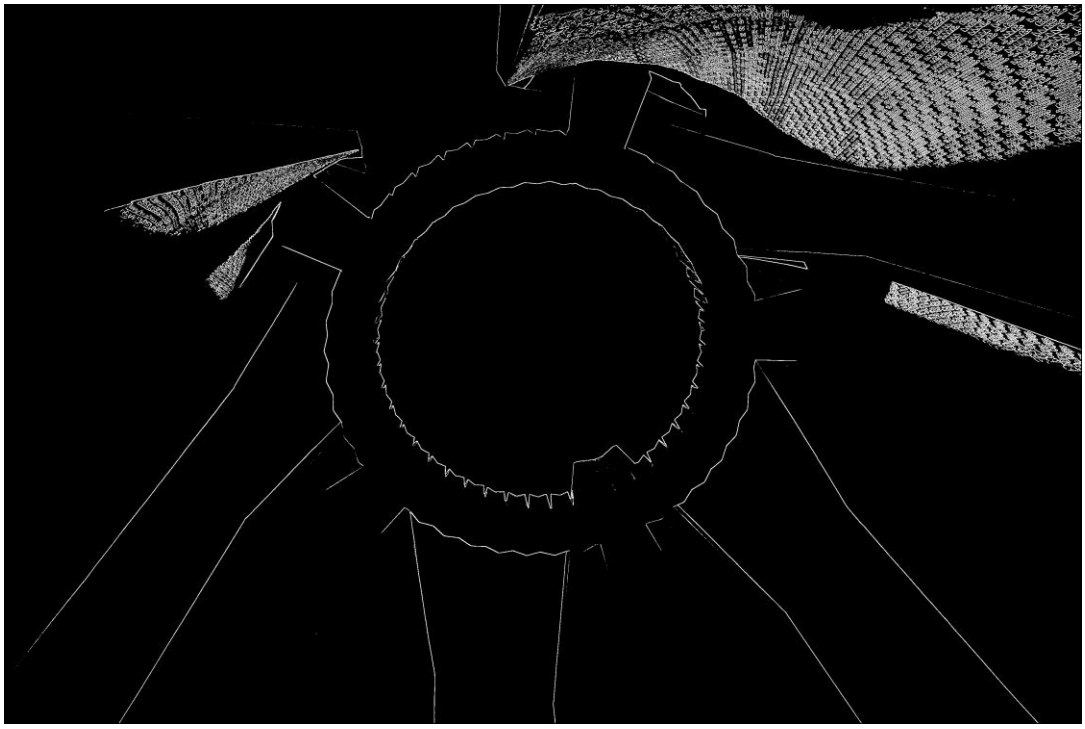
Рачунар на ком је писање и тестирање програма рађено има 16GB RAM меморије, шест физичких и дванаест логичких процесорских језгара.

4.2. Тестни случајеви

За тестирање коришћене су слике различитих димензија, као и различите димензије филтера.

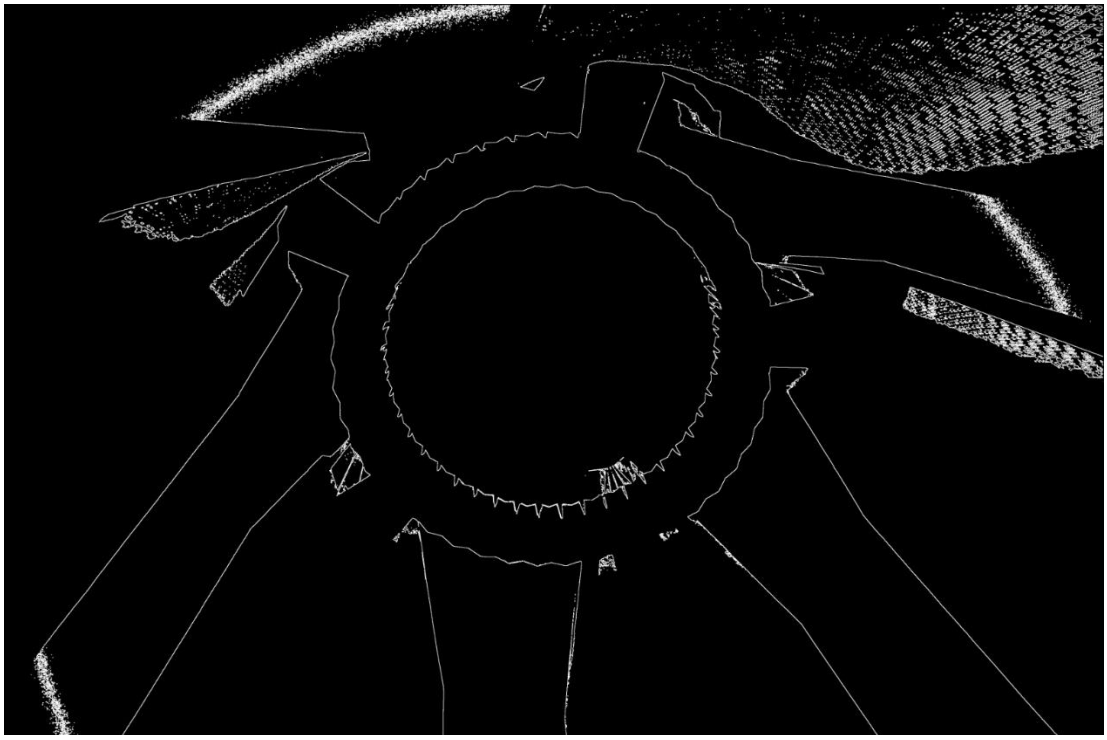
Следеће слике су приказ резултата за улазну слику димензије 3888x2592 (*Слика 8*).

Слика 8: Улазна слика

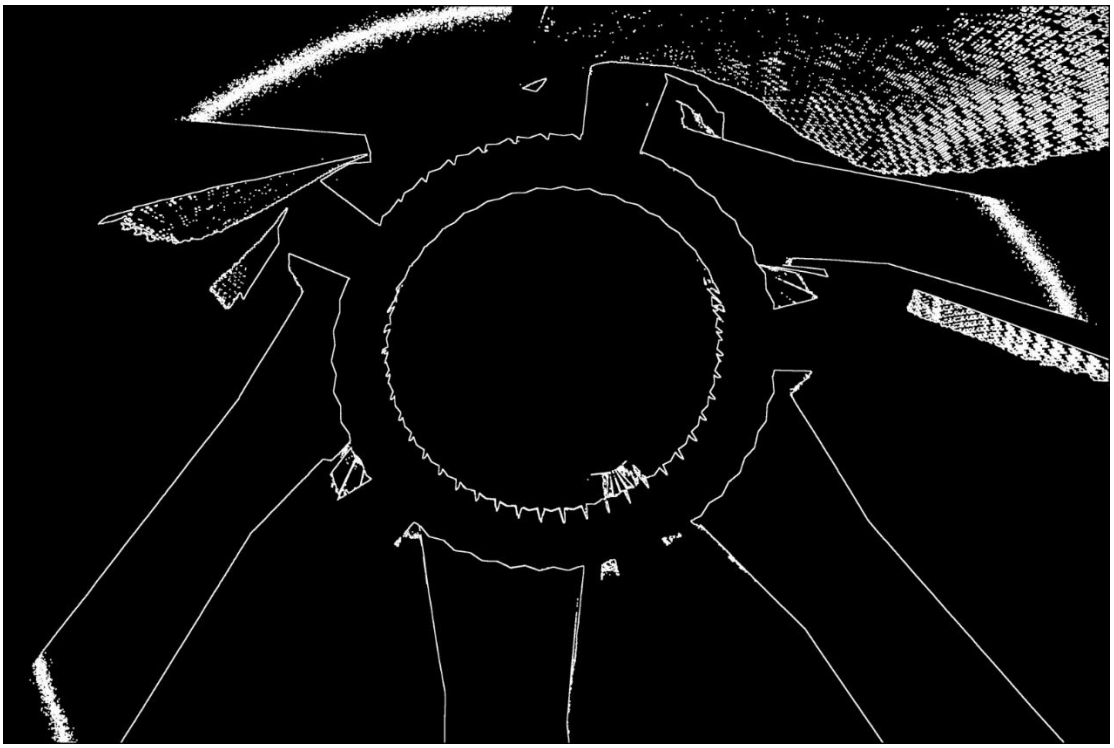


Слика 9: Резултат обраде "Prewitt" филтером димензије 3x3

Слика 10: Резултат обраде "Prewitt" филтером димензије 5x5



Слика 11: Резултат обраде најужом околином димензије 1



Слика 12: Резултат обраде најужом околином димензије 2

Следеће слике су приказ резултата за улазну слику димензије 5184x3456 (*Слика 13*)

Слика 13: Улазна слика

Слика 16: Резултат обраде најужом околином димензије 1

Слика 14: Резултат обраде "Prewitt" филтером димензије 3x3

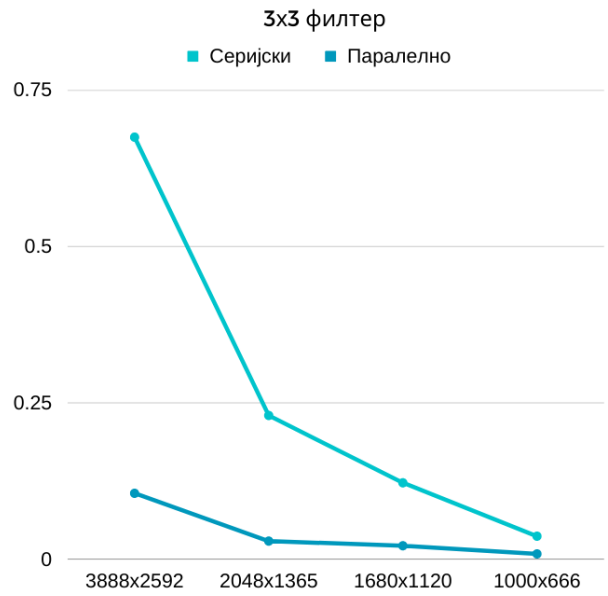
Слика 15: Резултат обраде "Prewitt" филтером димензије 5x5

Табела 1: Приказ времена у секундама за различите димензије слике и различите димензије "Prewitt" филтера

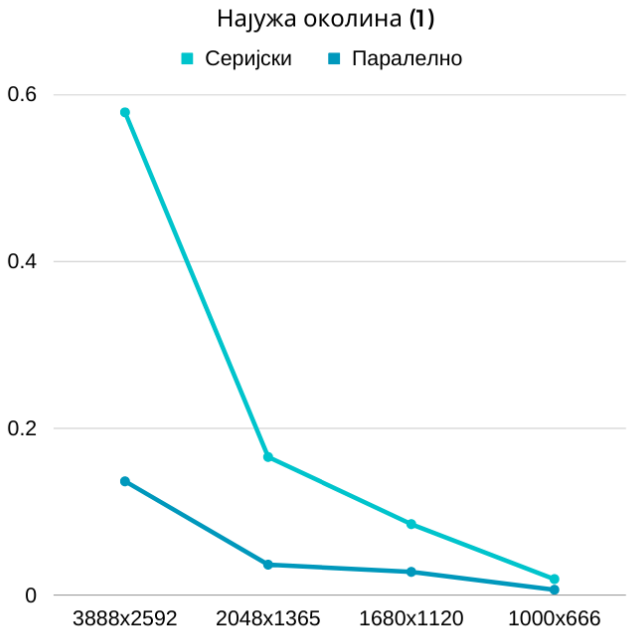
димензија слике/ имплементација	**Серијски 3x3 филтер**	**Паралелни 3x3 филтер**	**Серијски 5x5 филтер**	**Паралелни 5x5 филтер**
3888x2592	0.673025	0.105759	1.31252	0.238316
1680x1120	0.12691	0.0186685	0.237592	0.0438513

Приказ времена у секундама за различите димензије слике и различите димензије околине

димензија слике/ имплементација	**Серијски, околина 1**	**Паралелно, околина 1**	**Серијски, околина 2**	**Паралелно, околина 2**	**Серијски, околина 3**	**Паралелно, околина 3**
3888x2592	0.667865	0.146538	1.14057	0.290604	1.80423	0.462509
1680x1120	0.0768613	0.0199493	0.164698	0.0370726	0.282043	0.090113



Графикон 1: Зависност времена извршења "Prewitt" алгоритма са филтером 3x3 од димензије улазне слике



Графикон 2: Зависност времена извршења алгоритма најужа околина од димензије улазне слике

5. Анализа резултата

Сви паралелни резултати извршавају се задовољавајуће брже од серијских што је и био крајњи циљ

За границу одсецања узиман је однос ширине дела слике који се обрађује у том тренутку и укупне ширине слике. Тестирањем је извучен закључак да се највеће убрзање добија уколико се граница одсецања постави на 1/10 укупне ширине целе слике.

На основу резултата се види да се при смањењу димензија слике смањује и убрзање добијено паралелном обрадом.

Добијени резултати такође показују да се убрзање повећава повећањем филтера.