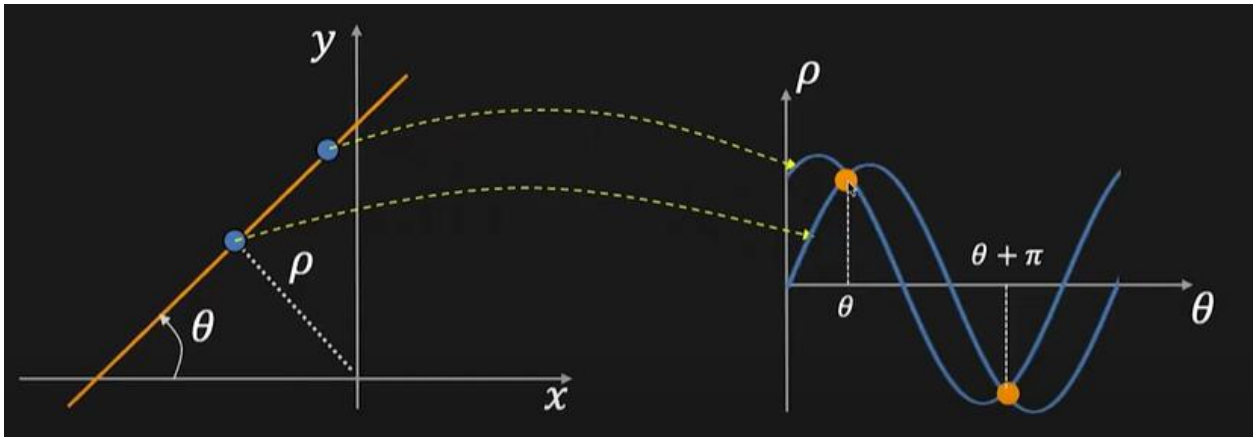
Пројекат: Детектовање правих линија на фотографији и њихово исцртавање

Документација за предмет: Пројектовање електронских уређаја на системском нивоу

1. Спецификација

Тема нашег пројекта јесте детектовање правих линија (праваца) на фотографији која се прослеђује алгоритму, те њихово исцртавање. Централни део алгоритма јесте „Hough“ трансформација, која представља једну линеарну трансформацију. Принцип трансформације јесте следећи: Свака тачка (пиксел) из једног домена (наша фотографија) се пресликава у праву у другом домену. У овом алгоритму се користи тригонометријски облик праве, тако да је она одређена са растојањем од координатног почетка (ро) и коефицијентом правца (тета). Информације о тим правама се складиште унутар акумулатора, који је представља матрицу, у којој је једна димензија додељена за тета, а друга за ро. Иницијална вредност свих поља унутар акумулатора је нула, а током рада алгоритма он се испуњава информацијама о трансформисаним линијама. На крају трансформације унутар акумулатора ће се наћи велике вредности у пољима са информацијама о правим уочљивих на оригиналној фотографији.



Слика 1. Hough трансформација

Овом алгоритму се прослеђује фотографија на којој су већ издвојене се ивице, те се под *тачком* из претходног пасуса подразумевају светли пиксели. На крају рада алгоритма, пронаћи ће се највећа вредност унутар акумулатора, која заправо представља правац који се најјасније уочава на фотографији. У односу на њега се врши исцртавање и осталих линија, а тим може да управља корисник, тако што при позиву програма, осим саме фотографије наводи и коефицијент исцртавања (реалан број 0 - 1). Отималан број зависи од фотографије, потребе корисника, али у општем случају, не прпоручује се вредност мања од 0.3, док је препоручена вредност од 0.5 до 0.7.

Улаз алгоритма представљају:

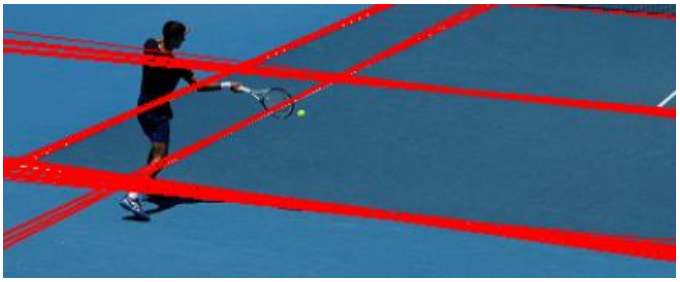
- фотографија над којом се врши поступак
- фактор исцртавања линија

Излаз алгорима представља:

- фотографија над којом су исртане уочљиве праве линије

Пример рада:

Слика 2. Улазна фотографија



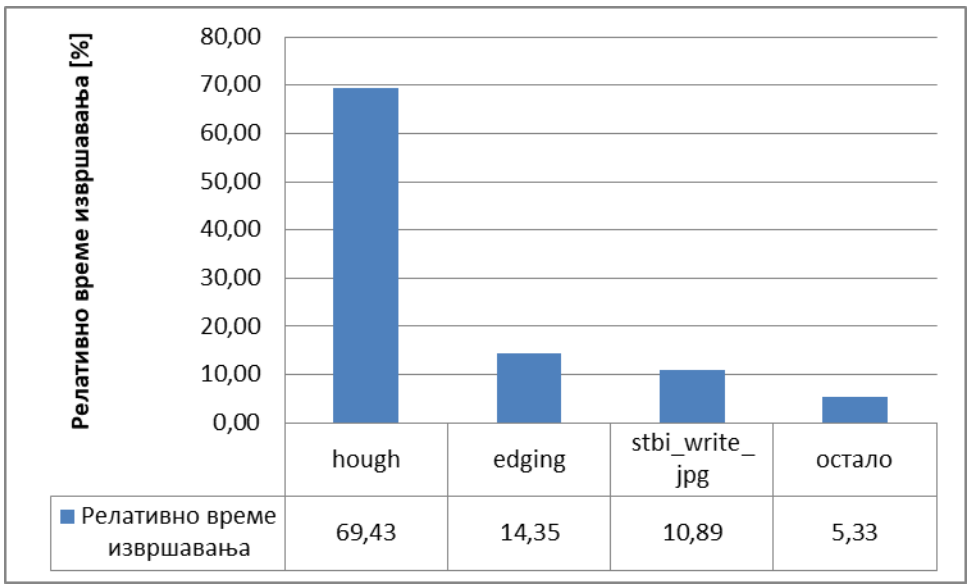
Слика 3. Излазна фотографија

2. Резултати профајлирања

Програм за детекцију линија на фотографији се састоји из неколико сегмената:

- учитавање фотографије (функција ***stbi_load***)
- копирања оригиналне фотографије (функција ***copying***)
- конверзија у монохроматску верзију (функција ***monochroming***)
- детекција ивица на фотографији (функција ***edging***)
- иницијализације таблице за акумулацију (функција ***acc_init***)
- „hough“ трансформација (функција ***hough***)
- проналазак највеће вредности у таблици за акумулацију (функција ***acc_max***)
- исцртавања линија на основу резултата (функција ***draw_lines***)
- чувања фотографије (функција ***stbi_write_jpg***)

Исход профајлирања у изворном ***C++*** коду, при чему смо за тестирање проследили 10 фотографија, је приказан на следећем графикону:



Графикон 1. Исход компајлирања

Значајан део времена приликом извршавања програма одлази на функцију *hough,* и због тога је одлучено да се акцелира тај део програма.

3. Битска анализа

Анализирајући наш програм у C++ коду, може се закључити да постоји неколико коришћених реалних променљивих. Како би се избегли типови са покретном тачком, за ове променљиве су дефинисани нови типови. Провером за разне дужине ових вектора, пронашли смо вредности за које алгоритам и даље даје задовољавајуће резултате детекције и исцртавања линија:

1. За вредности тригонометријских функција: **typedef sc_dt::sc_fixed_fast<16, 2> trig_t;** Пошто се овај тип користи унутар саме *hough* функције, односно касније у самом хардверу, посебно смо обратили пажњу да ова величина буде дужине једнаке степену двојке. Пошто за 8 бита алгоритам није био прецизан, следећа најмања задовољавајућа вредност је била 16 (2 испред зареза, за знак и вредност, а осталих 14 бита за реални остатак).

2. За вредност фактора исцртавања: **typedef sc_dt::sc_ufixed_fast<8, 1> dw_t**;
3. За вредност координата: **typedef sc_dt::sc_fixed_fast<16, 12> point_t;**
4. За вредност коефицијента праве: **typedef sc_dt::sc_fixed_fast<24, 12> slope_t;**
5. За вредност тачке пресека праве са y-осом: **typedef sc_dt::sc_fixed_fast<28, 24> intercept_t;**

Остале типови су мање важни, а употребљавају се унутар функције за исцртавање линија, *draw_lines***.**

Због ограничених ресурса на *Zybo* плочи и природе алгоритма услед којег се поменути акумулатор мора у потпуности складиштити унутар BRAM-ова, постоји ограничење на величину фотографија које се могу обрађивати, а то је у нашем случају формат такав да дијагонала фотографије није већа од 474 пиксела.

4. Виртуална платформа

На *слици 4* је приказана блок схема виртуалне платформе система који извршава функционалност, односно детектовање и исцртавање линија.

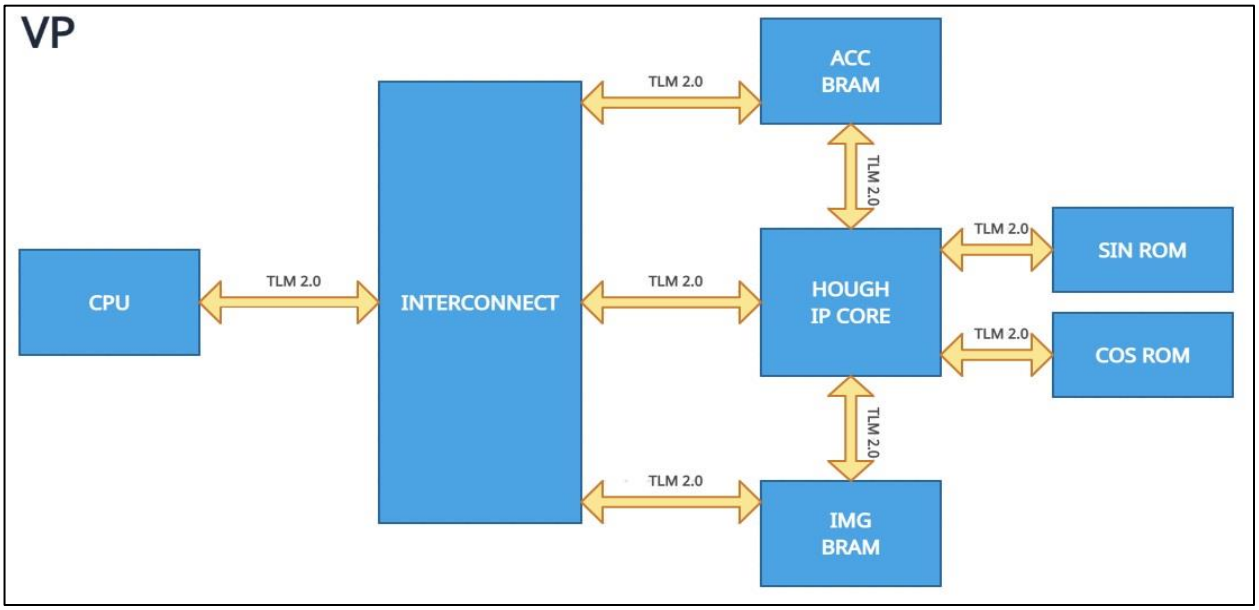
Платформа се састоји из неколико компоненти:

- ***CPU*** модул који представља софтвер
- ***Interconnect*** моудл којим се врши меморијско мапирање
- ***Hough IP Core*** модул који представља акцелатор критичног дела кода (*Hough* трансофмрација)
- ***Acc Bram*** (Block RAM) – део меморије за складиштење акумулатора
- ***Img Bram*** (Block RAM) – део меморије за складиштење фотографије
- ***Sin Rom*** – *look-up* табела за складиштење потребних вредности функције синус
- ***Cos Rom*** – *look-up* табела за складиштење потребних вредности функције косинус

Целокупни систем функционише на следећи начин:

Софтвер (односно *CPU*) врши учитавање фотографије и почетну обраду до *hough* трансформације. Пре саме хардверске операције, конфигуришу се регистри у *Hough IP Core-y* помоћу којих се сам постпуак извршава. Фотографија и акумулатор са иницијалним вредностима се из дељене меморије уписују у BRAM. Сетовањем *start* бита покреће се извршавање операције. Алгоритам пролази кроз

фотографију пиксел по пиксел и по потреби врши трансформацију тако што уз помоћ вредности синус и косинус функција из ROM-ова рачуна RO за свако TETA, те инкрементује одговарајућа поља у акумулатору. Након завршетка, језгро сетује *ready* бит те на тај начин обавештава софтвер да је акумулатор спреман за преузимање, што софтвер и ради, а на основу тих резултата се затим исцртавају линије.



Слика 4. Виртуална платформа

5. Hough IP Core модул

Назив регистра	Адресни офсет	Ширина регистра	Опис
width_reg	**0x00**	**10 бита**	ширина фотографије
height_reg	**0x02**	**10 бита**	висина фотографије
treshold_reg	**0x04**	**8 бита**	праг за „светли“ пиксел
rho_reg	**0x06**	**10 бита**	дијагонала фотографије
start_reg	**0x08**	**1 бит**	вредност 1 – команда за почетак рада језгра вредност 0 – модул је започео обраду или чека команду за почетак
ready_reg	**0x09**	**1 бит**	вредност 0 – модул није завршио обраду вредност 1 – модул је спреман за обраду

Табела 1. Регистарска мапа

На *слици 5* је приказан код из виртуалне платформе којим се моделује наш акцелератор. Овај модел се конфигурише из софтверске компоненте попуњавањем постојећих регистара (наведених у *табели 1*) и сетовањем start бита. *Ready* регистар обавештава да је операција завршена

```
// Prolazak po visini
for (sc_dt::sc_uint<16> y = 0; y < height_reg; y++)
{
    // Prolazak po sirini
    for (sc_dt::sc_uint<16> x = 0; x < width_reg; x++)
    {
        // Ucitavanje piksela
        current_pixel = read_bram(VP_ADDR_IMG_BRAM_L + y * width_reg + x);

        // Provera piksela
        if (current_pixel > threshold_reg)
        {
            // Ppounjavanje akumulatora
            for (theta = 0; theta < 180; theta++)
            {
                // Ucitavanje odgovarajucih vrednosti sinusa i kosinusa iz romova
                sin_pom = read_rom(VP_ADDR_SIN_ROM_L + theta);
                cos_pom = read_rom(VP_ADDR_COS_ROM_L + theta);

                // Proracun ro-a
                rho = x*sin_pom - y*cos_pom;

                // Popunjavanje akumulatora
                if (rho >= 0)
                {
                    cell_value = read_bram(VP_ADDR_ACC_BRAM_L + theta*rho_reg + rho);
                    cell_value++;
                    write_bram(VP_ADDR_ACC_BRAM_L + theta*rho_reg + rho, cell_value);
                    //cout << "acc[" << theta << "][" << rho << "] = " << cell_value << endl;
                }
                else
                {
                    cell_value = read_bram(VP_ADDR_ACC_BRAM_L + (THETA + theta)*rho_reg - rho);
                    cell_value++;
                    write_bram(VP_ADDR_ACC_BRAM_L + (THETA + theta)*rho_reg - rho, cell_value);
                    //cout << "acc[" << THETA + theta << "][" << -rho << "] = " << cell_value << endl;
                }

                offset += sc_core::sc_time(DELAY, sc_core::SC_NS);
                //cout << x << " " << y << " " << theta << " " << rho << endl;
            }
        }

        offset += sc_core::sc_time(DELAY, sc_core::SC_NS);
    }
}
```

Слика 5. Код за Hough IP Core модул

Као што се видело из претходне секције овог документа, језгро је повезано и на портове BRAM-ова преко којих преузима фотографију пиксел по пиксел, обрађује их и резултат складишти у акумулатор. И пиксели и акумулаторске вредности су 8-битне. Такође, комуницира и са ROM-овима из којих учитава неопходне тригонометријске вредности.

6. Перформансе система

Hough IP Core модул се састоји одоперација поређења, те сабирања и множења које су погодне за мапирање на DSP модуле на Zybo FPGA платформи. У првобитној верзији виртуалне платформе, са интуитивно процењеној фревенцији од **100MHz** (што је у коду моделовано са променљивом DELAY са вредношћу 10 и пратећом јединицом ns којим је представљено напредовање симулације), дошло се до резултата да је наш у систем могућности да обради у просеку око **90** фотографија у секунди (frame rate). (**Првобитна верзија**)