SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Zadanie k predmetu vizuálne systémy

**Bratislava 2017 Matúš Tinák, Ivan Petráň**

# Úloha 1

## Zadanie

Navrhnite a naprogramujte program (aplikáciu), ktorá bude schopná sledovať pohybujúce sa objekty Vami zvoleného tvaru (kruh) a farby v obraze z kamery.

Počas vypracovávania zadania budete prechádzať cez jednotlivé metódy spracovania obrazu , ktoré na seba nadväzujú (konverzie medzi farbenými modelmi, hranové operácie a iné filtrácie obrazu, segmentácia, detekcia objektov, ...).

## Vypracovanie

Ako prvé je potrebné výstup obrazu z kamery transformovať z RGB modelu do HSV:

while (1) {

capture.read(cameraFeed);

//konvercia snímky z BGR do HSV

cvtColor(cameraFeed, HSV, COLOR\_BGR2HSV);

inRange(HSV, Scalar(H\_MIN, S\_MIN, V\_MIN), Scalar(H\_MAX, S\_MAX, V\_MAX), threshold);

}

Ďalším krokom je v HSV obraze vyhľadať farbu, ktorú chceme sledovať. Toto je vykonané pomocou HSV slidera, v ktorom si vieme nastaviť MIN a MAX hodnoty jednotlivých hodnôt HSV (Hue, Saturation, Value). Pri zmene jednotlivých hodnôt je možné potom na výstupe obrazu vidieť iba farbu ktorú chceme. Napr. ak chceme na obraze vidieť iba červené objekty, tak budeme meniť hodnotu HSV, až kým na obrazu neostane iba červený objekt. Pre túto úlohu vieme sledovať modré, zelené, žlté a červené objekty.

if (clr == 0) {

H\_MIN = 89;

H\_MAX = 117;

S\_MIN = 146;

S\_MAX = 212;

V\_MIN = 110;

V\_MAX = 183;

}

if (clr == 1) {

H\_MIN = 51;

H\_MAX = 86;

S\_MIN = 105;

S\_MAX = 256;

V\_MIN = 137;

V\_MAX = 256;

}

if (clr == 2) {

H\_MIN = 0;

H\_MAX = 126;

S\_MIN = 126;

S\_MAX = 256;

V\_MIN = 214;

V\_MAX = 256;

}

if (clr == 3) {

H\_MIN = 140;

H\_MAX = 212;

S\_MIN = 142;

S\_MAX = 249;

V\_MIN = 186;

V\_MAX = 256;

}

Na to aby sme obraz vedeli s istotou zachytiť sme použili funkciu **erode** a **dilate**. Táto funckia zabezpečí,, že aj v prípade ak napr. kruh nebude na 100% správne zachytený, tieto funkcie ho dotvarujú do želanej podoby.

Mat erodeElement = getStructuringElement(MORPH\_RECT, Size(3, 3));

Mat dilateElement = getStructuringElement(MORPH\_RECT, Size(8, 8));

erode(thresh, thresh, erodeElement);

erode(thresh, thresh, erodeElement);

dilate(thresh, thresh, dilateElement);

dilate(thresh, thresh, dilateElement);

Ako posledné je na vyhľadanie a sledovanie nášho zvoleného objektu (kruh) najprv pomocou **contour** vykonané zistenie obrazu vo výstupe z kamery. Pokiaľ sa objekt hýbe, tak v strede tohto objektu sa objaví kurzor a zostane tam iba v prípade, ak sa objekt stále hýbe. V prípade, že objekt zastane, kurzor zmizne.

if (area>MIN\_OBJECT\_AREA && area<MAX\_OBJECT\_AREA && area>refArea) {

x = moment.m10 / area;

y = moment.m01 / area;

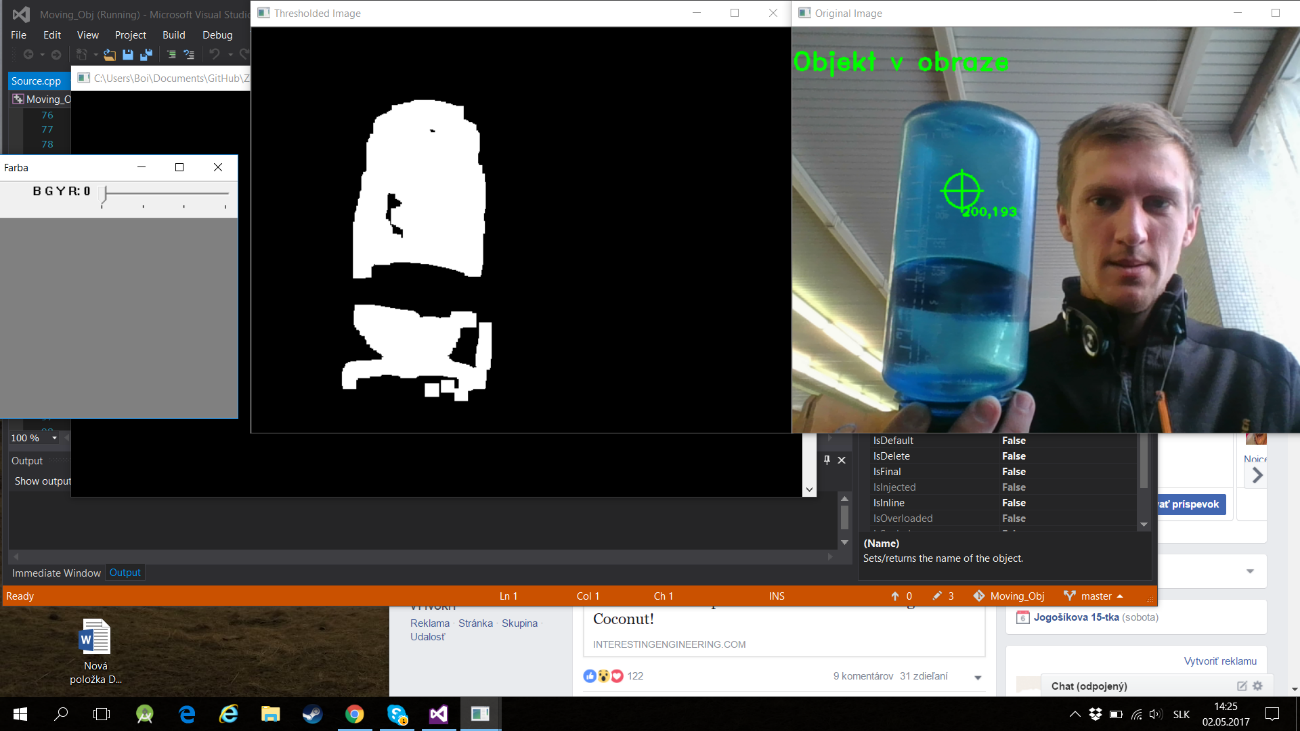
}

if ((objectFound == true)&&(abs(x-xold)>0.5)&&((abs(y - yold)>0.5))) {

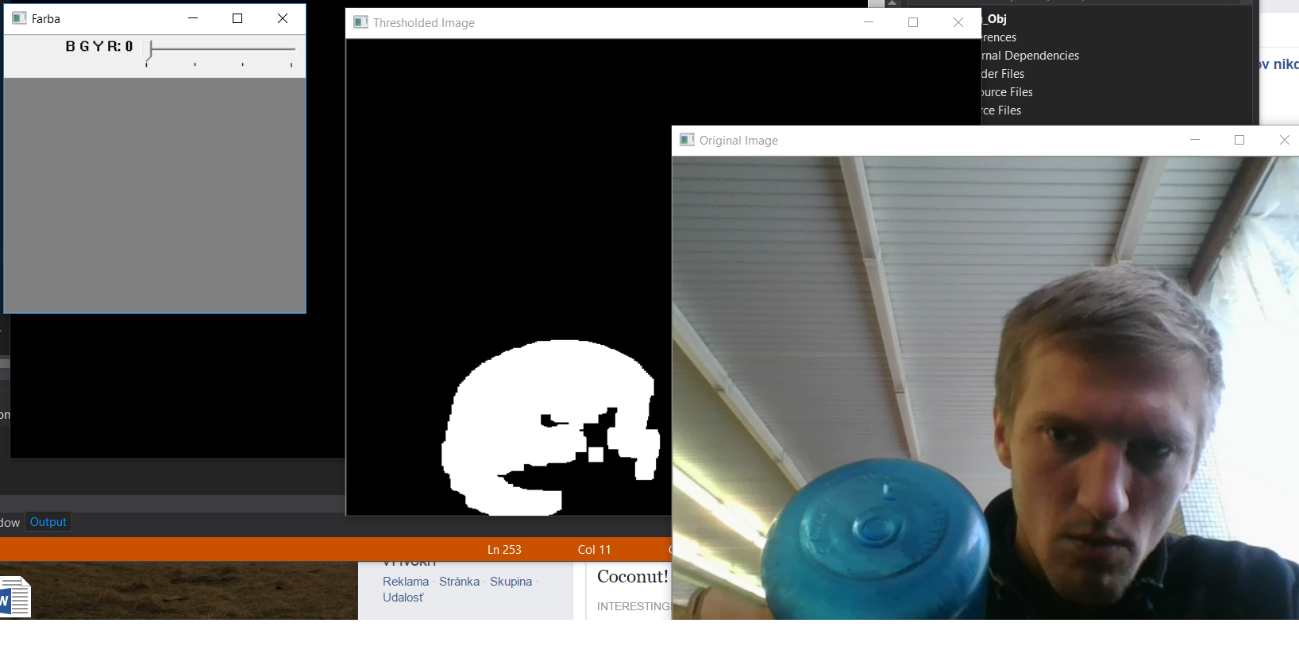
putText(cameraFeed, "Objekt v obraze", Point(0, 50), 2, 1, Scalar(0, 255, 0), 2);

drawObject(x, y, cameraFeed);

}



Obr. 1 Sledovanie pohyblivého objektu



Obr. 2 Statický objekt, kurzor zmizol

# Úloha 2

## Zadanie

Naprogramujte algoritmus detekcie hrán LoG operátorom, pričom ako vstup použijete obrázok vo formáte .bmp alebo .jpg.

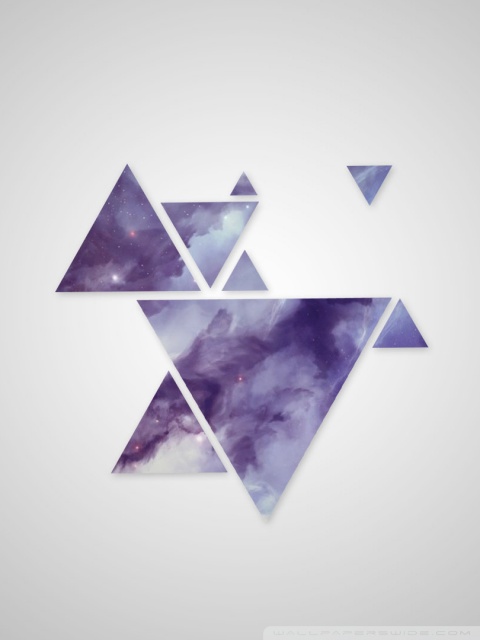
## Teória

Laplacian je 2-D isotropické opatrenie druhej priestorovej derivácie obrazu, ktoré zvýrazňuje oblasti silnej zmeny intenzity, preto je využívaný pre detekciu hrán. Laplacian je často aplikovaný na obraz ktorý bol najprv vyhladený s Gaussovým vyhladzovacím filtrom, aby sa zredukovala senzitivita na šum, preto je Laplacian a Gauss použitý spolu. Ako vstup sa väčšinou použije jeden šedý obraz z ktorého sa vyrobí nový šedý obraz.

Kedže vstupný obraz je reprezentovaný ako súbor diskrétnych pixelov, treba nájsť diskrétne konvolučné jadro, ktoré sa bude dať aproximovať druhou deriváciou v definícii Laplaca.

## Vypracovanie

Pre prácu s touto úlohou bol použitý nasledovný obrázok



Obr. 3. Pôvodný obrázok

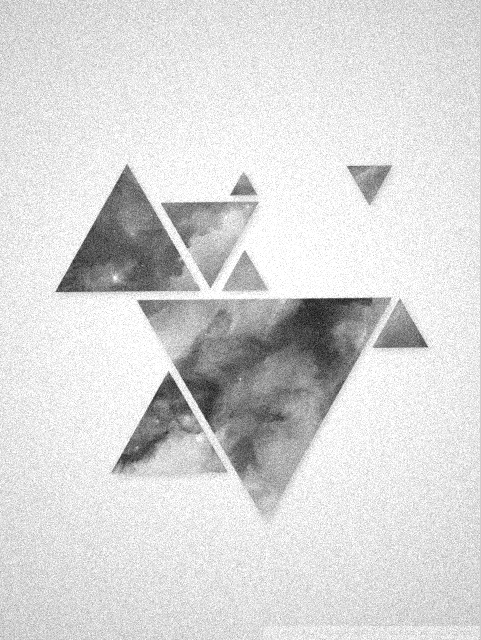
Ako prvé je potrebné obrázok transformovať na čierno biely obrázok.



Obr. 4. Čierno biely obrázok

Ďalej je na obrázok privedený šum pomocou príkazu :

grayNoise = imnoise(gray,'Gaussian',0.04,0.003);



Obr. 5. Čierno biely obrázok po privedení šumu

Po tejto úprave je obrázok potrebné upraviť pomocou Gaussového vyhladzovacieho filtra. Pri tejto úlohe je vytvorená 2D mriežka pomocou funkcie meshgrid, ktorá má rovnakú veľkosť ako Gaussova filtračná maska ( volená ).

ind = -floor(N/2) : floor(N/2);

[X Y] = meshgrid(ind, ind);

2D Gauss môže byť definovaný ako:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Konštanta pred exponentom slúži na to, aby plocha pod Gaussom bola vždy 1. V tejto úlohe táto konštanta nie je potrebná, pretože je zabezpečené, že plocha pod Gaussom je vždy 1, tj. všetky prvky matice sú vydelené tak, aby bol súčet všetkých prvkov matice 1. Za predpokladu, že filter je N x N s danou smerodajnou odchýlkou **σ** ( volená ) kód vyzerá nasledovne:

h = exp(-(X.^2 + Y.^2) / (2\*sigma\*sigma));

h = h / sum(h(:));

Kód pre samotnú filtráciu obrázka pomocou získanej matice h.

grayGauss=Funkcia\_konv(grayNoise,h,N)

kde funkcia Funkcia\_konv vyzerá nasledovne:

function [ image ] = Funkcia\_konv( img,matrix,rozsah )

[rows cols] = size(img)

G=floor(rozsah/2);

%// Vytvorenie novej matice, kde sa podla velkosti masky N pridaju do rohov matice 0

new\_img = zeros((rows+(2\*G)),(cols+(2\*G)));

new\_img = cast(new\_img, class(img));

new\_img((G+1):end-G,(G+1):end-G) = img;

image = zeros(size(new\_img));

image = cast(image, class(img));

%// 2-D konvolucia, blizsie vysvetlena pri Laplaciane

for i=(G+1):1:rows+G

for j=(G+1):1:cols+G

value=0;

for g=-G:1:G

for l=-G:1:G

value=value+new\_img(i+g,j+l)\*matrix(g+(G+1),l+(G+1));

end

end

image(i,j)=value;

end

end

image = image((G+1):end-G,(G+1):end-G);

end

Výsledný obrázok po Gaussovej filtrácii:



Obr. . Obrázok upravený pomocou Gaussovej filtrácie

Ako posledné je pomocou Laplaciána urobená detekcia hrán. Pre danú úlohu je použitá maska:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Ďalej je použitá funkcia, pomocou ktorej sú spravené potrebné úpravy pre dosiahnutie detekcie hrán pomocou použitia Laplaciána, kde sa najprv do matice z obr. 4 pridájú na okraj hodnoty 0 podľa toho, akú veľkosť má maska.

|  |  |  |
| --- | --- | --- |
|  |  |  |

[rows cols] = size(grayGauss)

new\_img = zeros(rows+2,cols+2);

new\_img = cast(new\_img, class(grayGauss));

new\_img(2:end-1,2:end-1) = grayGauss;

image = zeros(size(new\_img));

image = cast(grayGauss, class(img));

Ako posledné je potrebné vypočitať konvolúciu novo vzniknutej matice so zvolenou maskou, tj. treba pre každú bunku z matice zobrať 8 susedných buniek a prenásobiť ich zvolenou maskou. Súčtom všetkých výsledných 9 buniek sa potom získa nová hodnota výslednej bunky. Toto treba spraviť pre každý jeden pixel obrázka.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Pre vypočítanie konvolúcie sa použije funckia Funkcia\_konv, ktorej kód je zobrazený pri Gaussovej filtrácii.

Výsledný obrázok vyzerá potom nasledovne:

