SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

VIZUÁLNE SYSTÉMY

DOKUMENTÁCIA: ÚLOHA 2

Bratislava 2017 **Michal Pavlac, Filip Miklík**

Zadanie:

Úloha 2 (10 bodov):

Naprogramujte Houghovu transformáciu na detekciu kružníc, pričom ako vstup použijete obrázok vo formáte .bmp alebo . jpg.

Nemôžete použiť žiadnu knižnicu pre spracovanie obrazu!
***Programovacie prostredia**: Microsoft Visual Studio, Android Studio, Matlab,...*

Na zápočet je potrebné vypracovať obidva príklady aspoň s úspešnosťou 50%.

Vypracovanie:

Na začiatku sme si vyhľadali informácie o téme ktorú budeme spracovávať. Ako prvý sme riešili problém obrázku, ktorý použijeme na vstupe Houghovej transformácie. Musí to byť binary image. Ten sme získali s použitím nasledovných funkcií:

imread – načítanie obrázka do Matlabu uvedením jeho umiestnenia na disku

rgb2gray – zmena obrázku na tzv. gray resp. čiernobiely

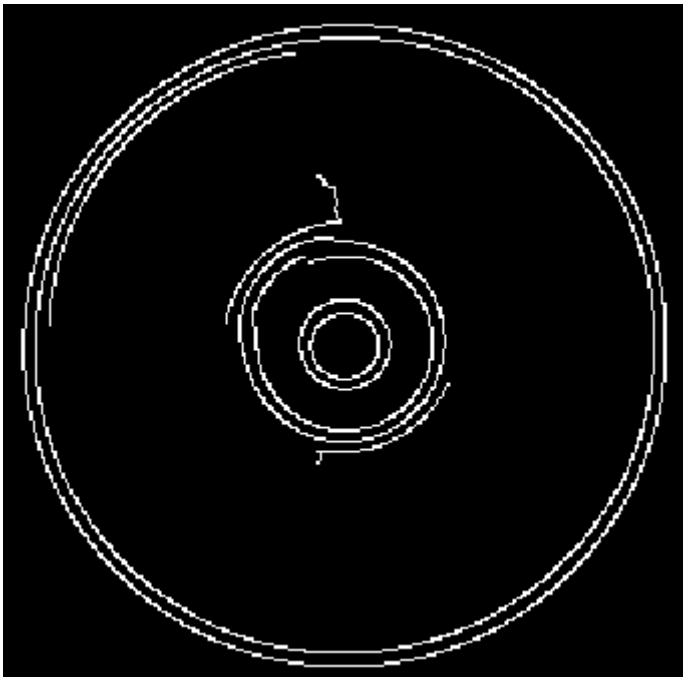
fspecial, imfilter – funkcie využité pri aplikácií gaussovho filtra, ktorý upravuje obrázok, aby bol vhodnejší na ďalšie operácie

edge – detekcia hrán v obrázku, výstupom je matica ktorá predstavuje jednotlivé pixely pôvodného obrázku, 1-tky v matici znamenajú nájdenú hranu, nuly prázdny priestor (použili sme typ detekcie ‘canny’)

Na nasledujúcom obrázku môžeme vidieť pôvodný obrázok:

Ďalej môžeme vidieť obrázok po aplikácií gaussovho filtra (je mierne rozmazaný):

A nakoniec môžeme vidieť binary image s detegovanými hranami, s ktorým budeme ďalej pokračovať:



Po týchto úpravách je už obrázok pripravený na samotnú Houghovu transformáciu. Na začiatku si určíme radius a prahovú hodnotu ktorá určuje minimálny počet pixelov, ktoré patria kruhu v priestore obrázku.

Ďalej si vytvoríme akumulátor, čo bude matica o rozmeroch rozlíšenia obrázku, ktorú zatiaľ naplníme nulami. Potom si v binary image nájdeme x-ové a y-ové súradnice pixelov ktoré tvoria hrany (pixely s hodnotou 1). Zistíme si počet riadkov a stĺpcov v binary image.

Ďalej vstupujeme do cyklu, v ktorom prechádzame hodnotami od 1 do veľkosti počtu prvkov v poli v ktorom máme zapísané x-ové súradnice nenulových pixelov. Vytvoríme si dolnú a hornú hodnotu, ktorej veľkosť nám bude určovať pole x-ových súradníc nenulových pixelov mínus radius (pre dolnú hodnotu) resp. plus radius (pre hornú hondotu). Ešte si saturujeme tieto hodnoty, aby neboli menšie ako 1 resp. aby nepresahovali počet stĺpcov v binary image. Teraz prechádzame (stále v rámci prvého cyklu) vo vnorenom cykle všetky hodnoty od dolnej po hornú. Pre tieto hodnoty počítame odchýlku ktorú potom odčítame a potom aj pričítame k aktuálnemu prvku poľa nenulových pixelov. Dostaneme premenné y1 resp. y2. Nasledujú podmienky, ak je y1 v rozmedzí medzi 1 a počtom riadkov v binary image, tak zvýšime číslo v akumulátore na danej pozícií o 1. To isté spravíme aj pre y2. Skončíme cyklus.

Teraz, keď máme akumulátor naplnený číslami, hľadáme lokálne maximá, z ktorých potom vytvoríme možné y a x. Vytvoríme tiež dočasný akumulátor tak, že od aktuálneho akumulátora odčítame nami zadanú prahovú hodnotu. Ďalej vykonávame cyklus od 1 po počet prvkov v možnom y. V cykle vyhodnocujeme podmienku, ak dočasný akumulátor má hodnotu viac ako 0 na aktuálnych súradniciach (možné x a možné y), potom pridáme do y, možné y resp. do x, možné x. Použili sme tieto funkcie:

zeros – vytvorí maticu núl

size – vráti vektor obsahujúci počty prvkov v každom rozmere matice

find – nájde nenulové prvky v poli

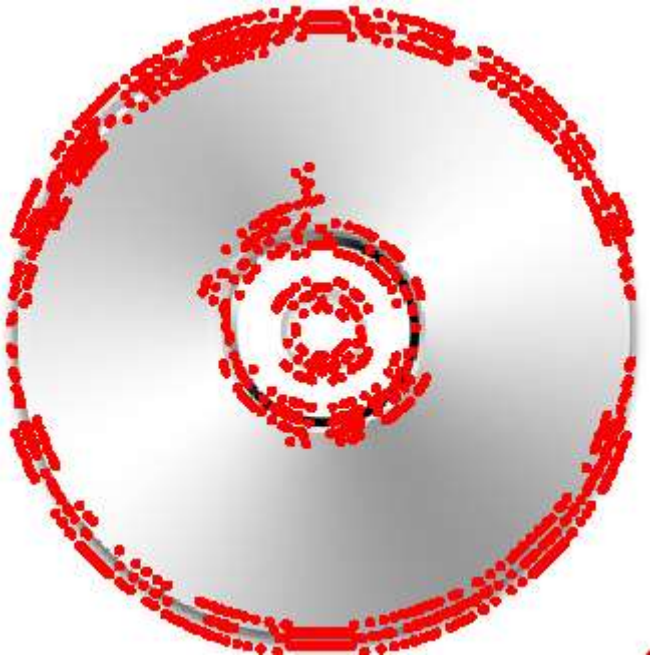
numel – vráti počet prvkov v poli

round – zaokrúhli každý prvok na najbližšiu celočíselnú hodnotu

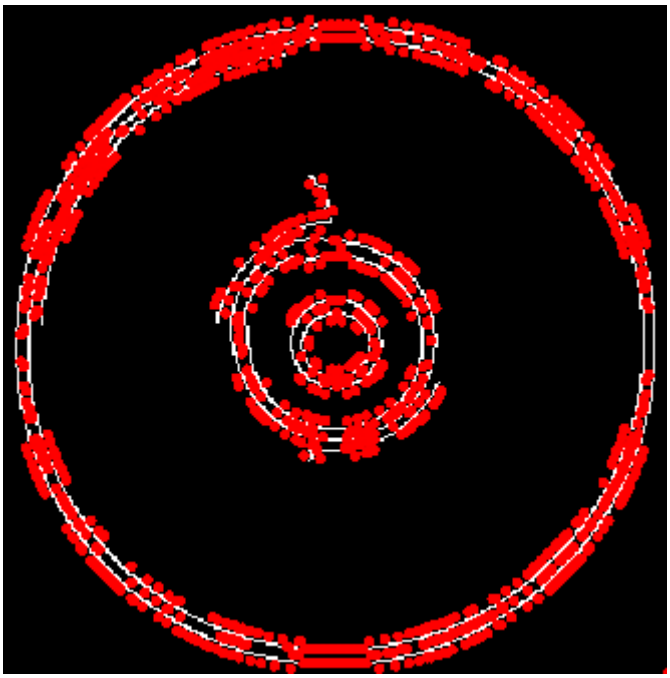
imregionalmax – nájde lokálne maximá v matici

Teraz, keď už máme x a y, môžeme vykresliť do pôvodného obrázku (alebo do binary image, ako si zvolíme) súradnice obkresľujúce nájdene kružnice.

Vykreslenie do pôvodného obrázku:



Vykreslenie do binary image:



Poznámka: Rôznou modifikáciou parametrov radius a prahová hodnota by bolo teoreticky možné dosiahnúť lepšie výsledky, toto sú najlepšie, ktoré sa podarili nám pri našich experimentoch.