מעבדה 3– HOG, PCA ו-SIFT

(1) ישום PCA במטלב – pca\_example.m

clear, clc, close all

rng(0)

data(:,1) = rand(30,1);

data(:,2) = 3.4 + 1.2 \* data(:,1);

data(:,2) = data(:,2) + 0.2\*rand(size(data(:,1)));

data = sortrows(data,1);

% Original Data

figure

axes('LineWidth',0.6,...

'FontName','Helvetica',...

'FontSize',8,...

'XAxisLocation','Origin',...

'YAxisLocation','Origin');

line(data(:,1),data(:,2),...

'LineStyle','None',...

'Marker','o');

axis equal

*data(:,1) = data(:,1)-mean(data(:,1));*

*data(:,2) = data(:,2)-mean(data(:,2));*

*% The task is to find the unit vector pointing into the direction with the largest variance*

*% within the bivariate data set data.*

*C = cov(data)*

*[V,D] = eig(C)*

*% The rotation helps to create new variables which are uncorrelated,*

*% i.e. the covariance is zero for all pairs of the new variables.*

*% The decorrelation is achieved by diagonalizing the covariance matrix C.*

*% Display the data together with the eigenvectors representing the new coordinate system.*

*figure*

*axes('LineWidth',0.6,...*

*'FontName','Helvetica',...*

*'FontSize',8,...*

*'XAxisLocation','Origin',...*

*'YAxisLocation','Origin');*

*line(data(:,1),data(:,2),...*

*'LineStyle','None',...*

*'Marker','o');*

*line([0 V(1,1)],[0 V(2,1)],...*

*'Color',[0.8 0.5 0.3],...*

*'LineWidth',0.75);*

*line([0 V(1,2)],[0 V(2,2)],...*

*'Color',[0.8 0.5 0.3],...*

*'LineWidth',0.75);*

*axis equal*

*norm(V(:,1))*

*norm(V(:,2))*

*dot(V(:,1),V(:,2))*

*% Calculating the data set in the new coordinate system.*

*% We need to flip newdata left/right since the second column is the one with the larges eigenvalue.*

*newdata = V \* data';*

*newdata = newdata';*

*newdata = fliplr(newdata)*

*% Display the newdata together with the new coordinate system.*

*figure*

*axes('LineWidth',0.6,...*

*'FontName','Helvetica',...*

*'FontSize',8,...*

*'XAxisLocation','Origin',...*

*'YAxisLocation','Origin')*

*line(newdata(:,1),newdata(:,2),...*

*'LineStyle','None',...*

*'Marker','o');*

*axis equal*

*% Do the same experiment using the MATLAB function PCA. We get same values for newdata and variance.*

*[coeff,newdatapca,latend,tsquared,variance] = pca(data);*

*newdatapca*

*variance*

*% The new data are decorrelated.*

*corrcoef(newdata)*

(2) HOG- hog\_example.m

clear

close ALL

img = imread('cameraman.tif');

%img = imread('gantrycrane.png');

[m,n,k]=size(img) ;

[featureVector,hogVisualization] = extractHOGFeatures(img,'CellSize',[16 16]);

figure, imshow(img);

hold on;

plot(hogVisualization);

featureVector הוא וקטור המאפיינים – כל מקטע מיוצג על ידי היסטוגרמת כיוונים באורך 9

CellSize מציין מהו גודל המקטע שעבורו מחושבת יסטוגרמת הכיוונים. שנה אותו ל- [64,64] ובחן את התוצאה

חזור על ההרצה עם קובץ התמונה 'gantrycrane.png' ואשר שהסטוגרמת הכיוונים מתאימה לכיוון קוי הקצה

(3) SIFT - SiftFeatures.m

%% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*This program finds the key points and their descriptors\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%% Important Variables

% a : Input image

% kpmag : keypoints magnitude

% kpori : keypoints orientation

% kpd : key point descriptors

% kp : keypoints

% kpl : keypoint locations

clc;

close all;

clear all;

fprintf('Some of the images available for checking are:\n');

fprintf('1.lena.jpg(512x512)\n2.lena1.jpg(256x256)\n3.peppers256.png(256x256)\n4.testfi.png(256x256)\n5.baboon.bmp(256x256)\n6.testfile.jpg(color)(1920x1200)\n');

image=input('Choose and enter the image name from above : ','s');

a=imread(image);

figure, imshow(a), hold on

title('Selected image');

[m,n,plane]=size(a);

if plane==3

a=rgb2gray(a);

end

a=im2double(a);

original=a;

store1=[];

store2=[];

store3=[];

tic

%% 1st octave generation

k2=0;

a(m:m+6,n:n+6)=0;

clear c;

for k1=0:3

k=sqrt(2);

sigma=(k^(k1+(2\*k2)))\*1.6;

for x=-3:3

for y=-3:3

h(x+4,y+4)=(1/((2\*pi)\*((k\*sigma)\*(k\*sigma))))\*exp(-((x\*x)+(y\*y))/(2\*(k\*k)\*(sigma\*sigma)));

end

end

for i=1:m

for j=1:n

t=a(i:i+6,j:j+6)'.\*h;

c(i,j)=sum(sum(t));

end

end

store1=[store1 c];

end

clear a;

a=imresize(original,1/((k2+1)\*2));

%% 2nd Octave generation

k2=1;

[m,n]=size(a);

a(m:m+6,n:n+6)=0;

clear c;

for k1=0:3

k=sqrt(2);

sigma=(k^(k1+(2\*k2)))\*1.6;

for x=-3:3

for y=-3:3

h(x+4,y+4)=(1/((2\*pi)\*((k\*sigma)\*(k\*sigma))))\*exp(-((x\*x)+(y\*y))/(2\*(k\*k)\*(sigma\*sigma)));

end

end

for i=1:m

for j=1:n

t=a(i:i+6,j:j+6)'.\*h;

c(i,j)=sum(sum(t));

end

end

store2=[store2 c];

end

clear a;

a=imresize(original,1/((k2+1)\*2));

%% 3rd octave generation

k2=2;

[m,n]=size(a);

a(m:m+6,n:n+6)=0;

clear c;

for k1=0:3

k=sqrt(2);

sigma=(k^(k1+(2\*k2)))\*1.6;

for x=-3:3

for y=-3:3

h(x+4,y+4)=(1/((2\*pi)\*((k\*sigma)\*(k\*sigma))))\*exp(-((x\*x)+(y\*y))/(2\*(k\*k)\*(sigma\*sigma)));

end

end

for i=1:m

for j=1:n

t=a(i:i+6,j:j+6)'.\*h;

c(i,j)=sum(sum(t));

end

end

store3=[store3 c];

end

[m,n]=size(original);

fprintf('\nTime taken for Pyramid level generation is :%f\n',toc);

%% Obtaining key point from the image

i1=store1(1:m,1:n)-store1(1:m,n+1:2\*n);

i2=store1(1:m,n+1:2\*n)-store1(1:m,2\*n+1:3\*n);

i3=store1(1:m,2\*n+1:3\*n)-store1(1:m,3\*n+1:4\*n);

[m,n]=size(i2);

kp=[];

kpl=[];

tic

for i=2:m-1

for j=2:n-1

x=i1(i-1:i+1,j-1:j+1);

y=i2(i-1:i+1,j-1:j+1);

z=i3(i-1:i+1,j-1:j+1);

y(1:4)=y(1:4);

y(5:8)=y(6:9);

mx=max(max(x));

mz=max(max(z));

mix=min(min(x));

miz=min(min(z));

my=max(max(y));

miy=min(min(y));

if (i2(i,j)>my && i2(i,j)>mz) || (i2(i,j)<miy && i2(i,j)<miz)

kp=[kp i2(i,j)];

kpl=[kpl i j];

end

end

end

fprintf('\nTime taken for finding the key points is :%f\n',toc);

%% Key points plotting on to the image

for i=1:2:length(kpl)

k1=kpl(i);

j1=kpl(i+1);

i2(k1,j1)=1;

plot(j1,k1,'yo')

end

hold off

figure, imshow(i2);

title('Image with key points mapped onto it');

%%

for i=1:m-1

for j=1:n-1

mag(i,j)=sqrt(((i2(i+1,j)-i2(i,j))^2)+((i2(i,j+1)-i2(i,j))^2));

oric(i,j)=atan2(((i2(i+1,j)-i2(i,j))),(i2(i,j+1)-i2(i,j)))\*(180/pi);

end

end

%% Forming key point neighbourhooods

kpmag=[];

kpori=[];

for x1=1:2:length(kpl)

k1=kpl(x1);

j1=kpl(x1+1);

if k1 > 2 && j1 > 2 && k1 < m-2 && j1 < n-2

p1=mag(k1-2:k1+2,j1-2:j1+2);

q1=oric(k1-2:k1+2,j1-2:j1+2);

else

continue;

end

%% Finding orientation and magnitude for the key point

[m1,n1]=size(p1);

magcounts=[];

for x=0:10:359

magcount=0;

for i=1:m1

for j=1:n1

ch1=-180+x;

ch2=-171+x;

if ch1<0 || ch2<0

if abs(q1(i,j))<abs(ch1) && abs(q1(i,j))>=abs(ch2)

ori(i,j)=(ch1+ch2+1)/2;

magcount=magcount+p1(i,j);

end

else

if abs(q1(i,j))>abs(ch1) && abs(q1(i,j))<=abs(ch2)

ori(i,j)=(ch1+ch2+1)/2;

magcount=magcount+p1(i,j);

end

end

end

end

magcounts=[magcounts magcount];

end

[maxvm maxvp]=max(magcounts);

kmag=maxvm;

kori=(((maxvp\*10)+((maxvp-1)\*10))/2)-180;

kpmag=[kpmag kmag];

kpori=[kpori kori];

% maxstore=[];

% for i=1:length(magcounts)

% if magcounts(i)>=0.8\*maxvm

% maxstore=[maxstore magcounts(i) i];

% end

% end

%

% if maxstore > 2

% kmag=maxstore(1:2:length(maxstore));

% maxvp1=maxstore(2:2:length(maxstore));

% temp=(countl((2\*maxvp1)-1)+countl(2\*maxvp1)+1)/2;

% kori=temp;

% end

end

fprintf('\nTime taken for magnitude and orientation assignment is :%f\n',toc);

%% Forming key point Descriptors

kpd=[];

for x1=1:2:length(kpl)

k1=kpl(x1);

j1=kpl(x1+1);

if k1 > 7 && j1 > 7 && k1 < m-8 && j1 < n-8

p2=mag(k1-7:k1+8,j1-7:j1+8);

q2=oric(k1-7:k1+8,j1-7:j1+8);

else

continue;

end

kpmagd=[];

kporid=[];

%% Dividing into 4x4 blocks

for k1=1:4

for j1=1:4

p1=p2(1+(k1-1)\*4:k1\*4,1+(j1-1)\*4:j1\*4);

q1=q2(1+(k1-1)\*4:k1\*4,1+(j1-1)\*4:j1\*4);

[m1,n1]=size(p1);

magcounts=[];

for x=0:45:359

magcount=0;

for i=1:m1

for j=1:n1

ch1=-180+x;

ch2=-180+45+x;

if ch1<0 || ch2<0

if abs(q1(i,j))<abs(ch1) && abs(q1(i,j))>=abs(ch2)

ori(i,j)=(ch1+ch2+1)/2;

magcount=magcount+p1(i,j);

end

else

if abs(q1(i,j))>abs(ch1) && abs(q1(i,j))<=abs(ch2)

ori(i,j)=(ch1+ch2+1)/2;

magcount=magcount+p1(i,j);

end

end

end

end

magcounts=[magcounts magcount];

end

kpmagd=[kpmagd magcounts];

end

end

kpd=[kpd kpmagd];

end

fprintf('\nTime taken for finding key point desctiptors is :%f\n',toc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END OF THE PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%