تمرین 1.1.1

نسیم فانی

اطلاعات گزارش

تاریخ: 99/8/13

واژگان کلیدی:
ثبت تصویر
مدل affine
نقاط کنترل

چکیده

در این تمرین، به بررسی روش affine mapping در ثبت تصویر (image registration) پرداخته شده است.

1-مقدمه

ثبت تصویر (انگلیسی: Image registration) یک فرایند تبدیل است که دسته‌های مختلف داده‌ها را به یک دستگاه مختصات دیگر منتقل می‌کند.

2-توضیحات تکنیکال

برای این منظور از چهار مرحله زیر استفاده می‌کنیم:

1. شناسایی تعدادی نقطه‌ی مشترک از هر دو تصویر
   a. به این نقاط، نقاط کنترل می‌گوییم.
   b. نقاط کنترل باید دارای ویژگی های یکسان باشند (برای مثال اگر تصاویر ما نقشه‌ی یک شهر هستند، تقاطع دو خیابان، گوشه یک ساختمان و ... می-تواند بهعنوان یک نقطه کنترل انتخاب شود)
   c. مختصات نقاط کنترل باید از هردو تصویر استخراج شود.
2. انتخاب یک مدل برای تغییر مختصات (در این تمرین از مدل affine استفادی می‌کنیم)
3. اعمال نقاط کنترل به مدل موردنظر و اجرای least-squares adjustment برای پیدا کردن پارامترهای تبدیل
4. درنهایت، استفاده از پارامترهای تبدیل برای تبدیل مختصات جغرافیایی تصویر

3-شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

برای مرحله اول فرض شده که ما بیش از سه نقطه کنترل از هردو تصویر را داریم:

	Target image	Input image
Point 1	(A,B)	(a,b)
Point 2	(C,D)	(c,d)
Point 3	(E,F)	(e,f)
Point 4	(G,H)	(g,h)
Point 5	(I,J)	(i,j)
Point 6	(K,L)	(k,l)

در مرحله دوم میخواهیم از مدل affine استفاده کنیم. تبدیل affine با در نظرگرفتن rotation، translation، scale و shear دو سیستم مختصاتی را به یکدیگر تبدیل می‌کند:

$X = Mx + Ny + P$ (Eq.1)

$Y = Qx + Ry + S$ (Eq.2)

در مرحله سوم نقاط کنترل را به فرم ماتریس درآورده و در معادله قرار می‌دهیم :

Eq.1:

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \\ \boldsymbol{E} \\ \boldsymbol{G} \\ I \\ K \end{bmatrix} = \begin{bmatrix} a & b & 1 \\ c & d & 1 \\ \boldsymbol{e} & f & 1 \\ g & h & 1 \\ i & j & 1 \\ k & l & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ \boldsymbol{P} \end{bmatrix} \longrightarrow$$

$$\begin{bmatrix} a & b & 1 \\ c & d & 1 \\ \boldsymbol{e} & f & 1 \\ g & h & 1 \\ i & j & 1 \\ k & l & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \\ \boldsymbol{E} \\ \boldsymbol{G} \\ I \\ K \end{bmatrix} \longrightarrow$$

$$\begin{bmatrix} a & b & 1 \\ c & d & 1 \\ \boldsymbol{e} & f & 1 \\ g & h & 1 \\ i & j & 1 \\ k & l & 1 \end{bmatrix}^{\boldsymbol{T}} \cdot \begin{bmatrix} a & b & 1 \\ c & d & 1 \\ \boldsymbol{e} & f & 1 \\ g & h & 1 \\ i & j & 1 \\ k & l & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ \boldsymbol{P} \end{bmatrix} =$$

$$\begin{bmatrix} a & b & 1 \\ c & d & 1 \\ \boldsymbol{e} & f & 1 \\ g & h & 1 \\ i & j & 1 \\ k & l & 1 \end{bmatrix}^{\boldsymbol{T}} \cdot \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \\ \boldsymbol{E} \\ \boldsymbol{G} \\ I \\ K \end{bmatrix} \longrightarrow W_1 \cdot \begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ \boldsymbol{P} \end{bmatrix} = Z_1$$

$$\begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{N} \\ \boldsymbol{P} \end{bmatrix} = W_1^{-1} \cdot Z_1$$

عینا همین روش را برای معادله دوم نیز تکرار می کنیم. در نهایت خواهیم داشت :

Eq.2:

$$\begin{bmatrix} Q \\ R \\ S \end{bmatrix} = W_2^{-1} \cdot Z_2$$

این مقادیر را در مدلaffine قرار می‌دهیم.

در آخرین مرحله، به ازای تمامی مقادیر تصویر ورودی، می‌توانیم مقادیر مورد نظرمان برای تصویر خروجی را محاسبه نماییم.

4-نتیجه گیری

برای پیدا کردن تابع انتقال جهت ثبت یک تصویر، استفاده از روش affine ساده و کارآمد است. لازم به ذکر است که روش های دقیق تری نیز وجود دارند که خارج از بحث این تمرین هستند.

مراجع

- https://www.youtube.com/watch?v=aDPJ2BzDMa4
- Digital Image Processing / Rafael C. Gonzalez, 4th Edition

تمرین 1.1.2

نسیم فانی

اطلاعات گزارش

تاریخ: 99/8/13

واژگان کلیدی:
اتصال تصویر
دوخت تصاویر
تصویر panoramic
Image stitching

چکیده

در این تمرین، میخواهیم دو تصویر را که دارای overlap هستند به یکدیگر متصل کرده (بدوزیم) و یک تصویر panoramic ایجاد نماییم.

1-مقدمه

دوخت تصاویر (انگلیسی: Image stitching) فرایندی است برای ترکیب چندین عکس که مناظر تصویربرداری‌شده توسط آن‌ها، همپوشانی یا پیوستگی دارند. هدف از انجام این کار، ایجاد یک منظره کامل‌تر و یک سراسرنما از مناظر است

در این کار، تصاویر باید با دقت بالا به هم وصل شده و در اتصال تصاویر نباید خطای اندازه یا پیکسل در تصویر بزرگ نهایی رخ بدهد.

در بیشتر موارد، برای دوخت تصویر نیاز به همپوشانی تقریباً دقیقی بین تصاویر هست تا نتیجه یکپارچه‌ای به‌دست آید.

2-توضیحات تکنیکال

برای این منظور از چهار مرحله زیر استفاده می‌کنیم:

1. شناسایی بخش overlap شده تصاویر
2. حذف قسمت overlap شده از یک تصویر
3. اضافه کردن تصاویر به یکدیگر

قسمت اول به کمک همبستگی(correlation) بین دو تصویر انجام شده است.

سپس با کمک index به دست آمده، می‌توان تشخیص داد که هر تصویر را تا چه مقداری در تصویر نهایی قرار دهیم .

در نهایت با کپی کردن مقادیر تصاویر ورودی (تا حد معین شده) تصویر پانورومای موردنظر ایجاد می‌شود.

3-شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

در شکل 1 تصویر ورودی اول، در شکل 2، تصویر ورودی دوم و در شکل 3 تصویر نهایی که حاصل از به هم دوختن تصاویر 1 و 2 است نمایش داده شده اند.

تصویر 1 - عکس ورودی اول

تصویر 2 - عکس ورودی دوم

تصویر 3 - عکس نهایی 1

4-نتیجه گیری

برای اتصال بهتر تصاویر، می‌توان از روش‌های دیگری برای پیدا کردن همپوشانی آن‌ها استفاده کرد.
در روشی که در حل این تمرین استفاده شده است، اگر تصاویر کاملا در یک راستا گرفته شوند، نتیجه‌ی اتصال بهتر خواهد شد.

پیوست

پیدا کردن همبستگی برای محاسبه index

تصاویر ورودی: F و S

```
for k = 0:cols-5  % to prevent j to go beyond boundaries.
   for j = 1:5
      F1(:,j) = F(:,k+j);
   end
   temp = corr2(F1,S1);
   Tmp = [Tmp temp];  % Tmp keeps growing, forming a matrix of 1*cols
   temp = 0;
end
```

اتصال تصاویر به یکدیگر

```
[Min_value, Index] = max(Tmp);
n_cols = Index + cols - 1; % New column of output image.
Opimg = [];
for i = 1:rows
   for j = 1:Index-1
      Opimg(i,j) = F(i,j); % First image is pasted till Index.
   end
   for k = Index:n_cols
      Opimg(i,k) = S(i,k-Index+1);%Second image is pasted after Index.
   end
end
```

مراجع

- https://en.wikipedia.org/wiki/Image_stitching
- https://uk.mathworks.com/matlabcentral/fileexchange/40848-image-stitching-using-correlation#overview_tab
- Digital Image Processing / Rafael C. Gonzalez, 4th Edition

تمرین 1.1.3

نسیم فانی

اطلاعات گزارش	چکیده
تاریخ: 99/8/13	در این تمرین، میخواهیم یک تصویر را حول نقطه مرکزی‌اش، 30^o ، 45^o ،80^o بچرخانیم.
واژگان کلیدی: چرخش نقطه مرکزی	

1-مقدمه

به کمک Affine transformations می‌توان برخی از تبدیلات ساده مانند rotation را بر روی تصاویر اعمال کرد.

2-توضیحات تکنیکال

برای چرخش تصاویر به اندازه‌ی a، از ماتریس زیر استفاده می‌کنیم:

$$\begin{bmatrix} cos(a) & sin(a) & 0 \\ -sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

داریم:

$x = v\cos(a) - w\sin(a)$
$y = v\cos(a) + w\sin(a)$

تصویر 1 - ورودی

3- -شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

تصویر ورودی و تصاویر خروجی :

تصویر 2 – چرخش 80 درجه

تصویر 3 – چرخش 30 درجه

تصویر 4 – چرخش 45 درجه

4-نتیجه گیری

چرخاندن تصاویر با این روش، موجب از دست رفتن بخشی از تصویر می‌شود.
ضمن اینکه بخش هایی از تصویر به دلیل عدم وجود اطلاعات کافی برای نمایش، به رنگ سیاه در میاید.

پیوست

تابع rotate

ورودی های این تابع تصویر موردنظر و درجه rotate است و خروجی آن تصویر چرخانده شده می‌باشد.

```
function image2=Rotate(image, angle)
[h,w,c]=size(image);
midx=round(h/2); %finding center of image
midy=round(w/2) ;
%rotate image around its center
for cc=1:c
   for ii=1:h
      for jj=1:w
      x_idx=round((ii-midx)*cosd(angle)+sind(angle)*(jj-midy))+midx ;
      y_idx=round(cosd(angle)*(jj-midy)-sind(angle)*(ii-midx))+midy;
      if(x_idx>1 && y_idx>1 && x_idx<=h && y_idx<=w)
      image2(ii,jj,cc)=image(x_idx,y_idx,cc)   ;
      end
      end
   end
end
```

استفاده از این تابع در فایل main

```
f = imread('1.bmp');
image = Rotate(f,30);  % also for 45 and 80
imshow(image);
```

مراجع

- https://en.wikipedia.org/wiki/Digital_image_processing
- Digital Image Processing / Rafael C. Gonzalez, 4th Edition

تمرین 1.2.1

نسیم فانی

اطلاعات گزارش

تاریخ: 99/8/13

واژگان کلیدی:
هیستوگرام
histogram
equalization
mse
quantize

چکیده

در این تمرین، می‌خواهیم یک تصویر را در دوحالت equalized و non-equalized در سطح‌های مختلف (8، 16، 32، 64، 128) quantize کنیم. سپس MSE را برای هر یک از تصاویر محاسبه نماییم.

1-مقدمه

منظور از quantizeکردن تصویر، کاهش تعداد سطوح خاکستری است.

2-توضیحات تکنیکال

برای این کار نیاز است تا کل level ها را بر تعداد سطوحی که میخواهیم تقسیم نماییم و سپس یک نگاشت بین سطوح خاکستری قبل و سطوح خاکستری جدید ایجاد کنیم.

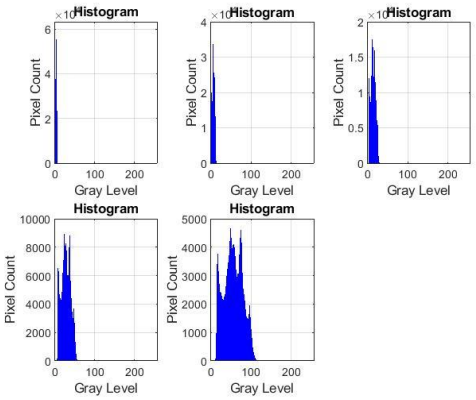
در ادامه ابتدا تصاویر را equalize کنیم و سپس مجددا عمل quantize را انجام دهیم.

سپس به مقایسه نتایج حاصل می‌پردازیم.

تصویر 1 – تصاویر non-equalized با quantization از 1 تا 5 به ترتیب 8مرحله، 16 مرحله، 32 مرحله، 64 مرحله و 128 مرحله

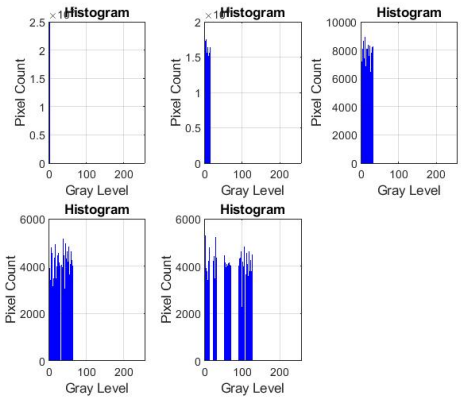
3--شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

تصاویر و هیستوگرام‌های خروجی :



تصویر 2 – هیستوگرام تصاویر non-equalized با quantization از 1 تا 5 به ترتیب 8مرحله، 16 مرحله، 32 مرحله، 64 مرحله و 128 مرحله

تصویر 3 – تصاویر equalized با quantization از 1 تا 5 به ترتیب 8مرحله، 16 مرحله، 32 مرحله، 64 مرحله و 128 مرحله



تصویر 4 – هیستوگرام تصاویر equalized با quantization از 1 تا 5 به ترتیب 8مرحله، 16 مرحله، 32 مرحله، 64 مرحله و 128 مرحله

جدول مقادیر mse

level	8	16	32	64	128
Without histeq	13865.97	12984.52	11303.66	8292.71	3667.05
With histeq	22554.23	12638.43	10671.93	7224.09	2432.85

4-نتیجه گیری

اعمال equalization باعث افزایش کنتراست تصویر و در نتیجه بالا رفتن کیفیت می‌شود.

MSE نشان دهنده‌ی میزان تفاوت تصاویر است. هر چه مقدارآن بیشتر باشد، یعنی تفاوت دو تصویر بیشتر است.

به همین دلیل مقادیر MSE در تصاویری که equalize شده اند عدد بزرگتری را نشان می‌دهد.

پیوست

تابع quantize

ورودی های این تابع تصویر موردنظر و level است و خروجی آن تصویر quantize شده می‌باشد.

```
function out = quantize(img, level)
out = floor(img ./ (256/level));
end
```

تابع MyHistogram برای نمایش هیستوگرام تصاویر

```
function [counts, grayLevels] = MyHistogram(grayImage)
[rows, columns, numberOfColorChannels] = size(grayImage);
counts = zeros;(256 ,1)
for col = 1 : columns
        for row = 1 : rows
                %Get the gray level.
                grayLevel = grayImage(row, col);
                %Add 1 because graylevel zero goes into index 1 and so on.
                counts(grayLevel+ 1) = counts(grayLevel+1) + 1;
        end
end
grayLevels = 0 : 255;
bar(grayLevels, counts, 'BarWidth', 1, 'FaceColor', 'b');
xlabel('Gray Level', 'FontSize', 10);
ylabel('Pixel Count', 'FontSize', 10);
title('Histogram', 'FontSize', 10);
grid on;
end
```

تابع Main

```
clc;
clear all;
close all;
img = imread('Barbara.bmp');
f = rgb2gray(img);
q8 = quantize(f,8);
q16 = quantize(f,16);
q32 = quantize(f,32);
q64 = quantize(f,64);
q128 = quantize(f,128) ;

histf = histeq(f);
eq8 = imquantize(histf,8);
eq16 = quantize(histf,16);
eq32 = quantize(histf,32);
eq64 = quantize(histf,64);
eq128 = quantize(histf,128);

err8 = immse(q8,f);
err16 = immse(q16,f);
err32 = immse(q32,f);
err64 = immse(q64,f);
err128 = immse(q128,f);

erreq8 = immse(im2uint8(eq8),f);
erreq16 = immse(im2uint8(eq16),f);
erreq32 = immse(eq32,f);
erreq64 = immse(eq64,f);
erreq128 = immse(eq128,f);
```

مراجع

- Image Processing / Rafael C. Gonzalez, 4th Edition
- https://uk.mathworks.com/help/images/ref/imquantize.html
- https://uk.mathworks.com/matlabcentral/answers/249632-quantize-a-greyscale-image-by-5-levels
- https://uk.mathworks.com/matlabcentral/answers/330385-find-the-histogram-of-the-image-cameraman-without-using-the-matlab-built-in-functions-for-histogram

تمرین 1.2.2

نسیم فانی

اطلاعات گزارش

تاریخ: 99/8/13

واژگان کلیدی:
upsampling
downsamplin

چکیده

در این تمرین، می‌خواهیم مقایسه‌ای بین روش‌های مختلف upsampling و downsampling داشته باشیم.

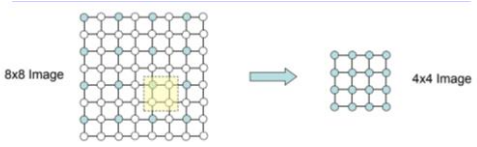
1-مقدمه

روش‌های مختلف upsampling و downsampling را بررسی می‌کنیم.

2-توضیحات تکنیکال

برای down-sample کردن دو روش استفاده می‌کنیم:

اول- در این روش بدون استفاده از فیلتر و با حذف سطها و ستون ها، تصویر را down-sample می‌کنیم.



در این حالت داریم:

$F_d(m,n) = f(2m, 2n)$

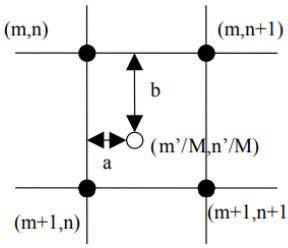
دوم- در این روش ابتدا فیلتر میانگین را بر روی تصویر اعمال کرده و سپس آن را با حذف سطر و ستون down-sample می‌کنیم.

در این حالت داریم :

$F_d(m,n) = [f(2m,2n) + f(2m,2n+1) + f(2m+1, 2n) + f(2m+1, 2n+1)]/4$

برای up-sample نیز از دو روش استفاده میکنیم :

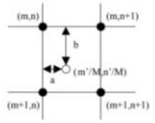
اول- با کمک bilinear interpolation



• Direct interpolation: each new sample takes 4 multiplications:
O[m',n']=(1-a)*(1-b)*I[m,n]+a*(1-b)*I[m,n+1]+(1-a)*b*I[m+1,n]+a*b*I[m+1,n+1]

• Separable interpolation:
i) interpolate along each row y: F[m,n']=(1-a)*I[m,n]+a*I[m,n+1]
ii) interpolate along each column x': O[m',n']=(1-b)*F[m',n]+b*F[m'+1,n]

دوم- نزدیک ترین همسایه



O[m',n'] (the resized image) takes the value of the sample nearest to (m'/M,n'/M) in I[m,n] (the original image):

$$O[m', n'] = I[(\text{int})(m + 0.5), (\text{int})(n + 0.5)], m = m'/M, n = n'/M.$$

3--شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

تصاویر و هیستوگرام‌های خروجی :

تصویر 1 – تصویر down-sample بدون فیلتر

تصویر 2 – تصویر down-sample با فیلتر

تصویر 3 – تصویر up-sample شده‌ی تصویر اول با نزدیک ترین همسایه

تصویر 4 – تصویر up-sample شده‌ی تصویر دوم با نزدیک ترین همسایه

تصویر 5 – تصویر up-sample شده‌ی تصویر اول با bilinear interpolation

تصویر 6 – تصویر up-sample شده‌ی تصویر دوم با bilinear interpolation

جدول مقادیر mse

4-نتیجه گیری

	Pixel Replication	Bilinear Interpolation
Averaging	13865.97	12984.52
Remove Row&Column	22554.23	12638.43

هر چه MSEبیشتر باشد یعنی تفاوت تصویر ما بیشتر است.

با افزایش کنتراست، کیفیت تصویر افزایش می‌بد

پیوست

تابع down-sample

```
function imgOut = myDownsample(img,factor)
[rows,columns]=size(img);
```

```
function res = averaging(img)
[x,y] = size(img);
res = zeros(x,y);
windows = 8;
for i=1:windows:x
   for j=1:windows:y
      res(i:i+windows-1,j:j+windows-1)...=
         mean2(img(i:i+windows-1,j:j+windows-1));
   end
end

end
```

تابع averaging

تابع نزدیک ترین همسایه

```
function outputImage = upsample_nearest(inputImage)
scale = [2 2];              %# The resolution scale factors: [rows columns]
oldSize = size(inputImage);                 %# Get the size of your image
newSize = max(floor(scale.*oldSize(1:2)),1);  %# Compute the new image size

 #%Compute an upsampled set of indices:

rowIndex = min(round(((1:newSize(1))-0.5)./scale(1)+0.5),oldSize(1));
colIndex = min(round(((1:newSize(2))-0.5)./scale(2)+0.5),oldSize(2));

 #%Index old image to get new image:

outputImage = inputImage(rowIndex,colIndex,:);
end
```

تابع lineaer - interpolation

```
function [Y] = bi_inter(I, ratio)
[h, w] = size(I);
H = (ratio * h);
W = (ratio * w);
Y = zeros(H,W);
hs = (h/H);
ws = (w/W);
   for i=1:H
     y = (hs * i) + (0.5 * (1 - 1/ratio));
       for j=1:W
         x = (ws * j) + (0.5 * (1 - 1/ratio));
 //%      Any values out of acceptable range
        x(x < 1) = 1;
        x(x > h - 0.001) = h - 0.001;
        x1 = floor(x);
        x2 = x1 + 1;
        y(y < 1) = 1;
        y(y > w - 0.001) = w - 0.001;
        y1 = floor(y);
        y2 = y1 + 1;
 4 //%      Neighboring Pixels
        NP1 = I(y1,x1);
        NP2 = I(y1,x2);
        NP3 = I(y2,x1) ;
        NP4 = I(y2,x2);
 4 //%      Pixels Weights
        PW1 = (y2-y)*(x2-x);
        PW2 = (y2-y)*(x-x1);
        PW3 = (x2-x)*(y-y1);
        PW4 = (y-y1)*(x-x1);
        Y(i,j) = PW1 * NP1 + PW2 * NP2 + PW3 * NP3 + PW4 * NP4;
       end
   end
 end
```

مراجع

- Image Processing / Rafael C. Gonzalez, 4th Edition
- https://uk.mathworks.com/help/images/ref/imresize.html

- https://stackoverflow.com/questions/1550878/nearest-neighbor-interpolation-algorithm-in-matlab-