تمرين 1.1.1

نسيم فاني

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: 13/8/99** |  | در اين تمرين، به بررسي روش affine mapping در ثبت تصوير (image registration) پرداخته شده است. |
| **واژگان كليدي:**  ثبت تصوير  مدل affine  نقاط كنترل |  |

1-مقدمه[[1]](#footnote-1)

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

2-توضيحات تكنيكال

براي اين منظور از چهار مرحله زير استفاده مي­كنيم:

1. شناسايي تعدادي نقطه­ي مشترك از هر دو تصوير
   1. به اين نقاط، نقاط كنترل مي­گوييم.
   2. نقاط كنترل بايد داراي ويژگي هاي يكسان باشند ( براي مثال اگر تصاوير ما نقشه­ي يك شهر هستند، تقاطع دو خيابان، گوشه يك ساختمان و ... مي­تواند بهعنوان يك نقطه كنترل انتخاب شود )
   3. مختصات نقاط كنترل بايد از هردو تصوير استخراج شود.
2. انتخاب يك مدل براي تغيير مختصات ( در اين تمرين از مدل affine استفادي مي­كنيم)
3. اعمال نقاط كنترل به مدل موردنظر و اجراي least-squares adjustment براي پيدا كردن پارامترهاي تبديل
4. درنهايت، استفاده از پارامترهاي تبديل براي تبديل مختصات جغرافيايي تصوير

3-شکل‏ها، جدول‏ها و روابط (فرمول‏ها)

براي مرحله اول فرض شده كه ما بيش از سه نقطه كنترل از هردو تصوير را داريم:

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 ميخواهيم از مدل affine استفاده كنيم. تبديل affine با در نظرگرفتن rotation، translation، scale و shear دو سيستم مختصاتي را به يكديگر تبديل مي­كند:

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

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

در مرحله سوم نقاط كنترل را به فرم ماتريس درآورده و در معادله قرار مي­دهيم :

Eq.1:

= .

. =

. . = . W1 . = Z1

= W1-1 . Z1

عينا همين روش را براي معادله دوم نيز تكرار مي­ كنيم. در نهايت خواهيم داشت :

Eq.2:

= W2-1 . Z2

قرار مي­دهيم.affine اين مقادير را در مدل

در آخرين مرحله، به ازاي تمامي مقادير تصوير ورودي، مي­توانيم مقادير مورد نظرمان براي تصوير خروجي را محاسبه نماييم.

4-نتيجه گیری

براي پيدا كردن تابع انتقال جهت ثبت يك تصوير، استفاده از روش affine ساده و كارآمد است. لازم به ذكر است كه روش هاي دقيق تري نيز وجود دارند كه خارج از بحث اين تمرين هستند.

**مراجع**

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

تمرين 1.1.2

نسيم فاني

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: 13/8/99** |  | در اين تمرين، ميخواهيم دو تصوير را كه داراي overlap هستند به يكديگر متصل كرده (بدوزيم) و يك تصوير panoramic ايجاد نماييم. |
| **واژگان كليدي:**  اتصال تصوير  دوخت تصاویر  تصوير panoramic  Image stitching‎ |  |

1-مقدمه[[2]](#footnote-2)

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-توضيحات تكنيكال

براي اين منظور از چهار مرحله زير استفاده مي­كنيم:

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

قسمت اول به کمک همبستگیcorrelation)) 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-نتيجه گیری

براي اتصال بهتر تصاوير، مي­توان از روش­هاي ديگري براي پيدا كردن همپوشاني آن­ها استفاده كرد.

در روشي كه در حل اين تمرين استفاده شده است، اگر تصاوير كاملا در يك راستا گرفته شوند، نتيجه­ي 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

تمرين 1.1.3

نسيم فاني

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: 13/8/99** |  | در اين تمرين، ميخواهيم يك تصوير را حول نقطه مركزي­اش، 80º، 45º ، 30º بچرخانیم. |
| **واژگان كليدي:**  چرخش  نقطه مرکزی |  |

1-مقدمه[[3]](#footnote-3)

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

2-توضيحات تكنيكال

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

داریم:

x = v cos(a) – w sin(a)

y = v cos(a) + w sin(a)

3- -شکل‏ها، جدول‏ها و روابط (فرمول‏ها)

تصویر ورودی و تصاویر خروجی :



تصویر 1 - ورودی



تصویر 2 – چرخش 80 درجه



تصویر 3 – چرخش 30 درجه



تصویر 4 – چرخش 45 درجه

4-نتيجه 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

تمرين 1.2.1

نسيم فاني

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: 13/8/99** |  | در اين تمرين، مي­خواهيم يك تصوير را در دوحالت equalized و non-equalized در سطح­های مختلف ( 8، 16، 32، 64، 128) quantizeکنیم. سپس MSE را برای هر یک از تصاویر محاسبه نماییم. |
| **واژگان كليدي:**  هیستوگرام  histogram equalization  mse  quantize |  |

1-مقدمه[[4]](#footnote-4)

منظور از quantize کردن تصویر، کاهش تعداد سطوح خاکستری است.

2-توضيحات تكنيكال

براي 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.

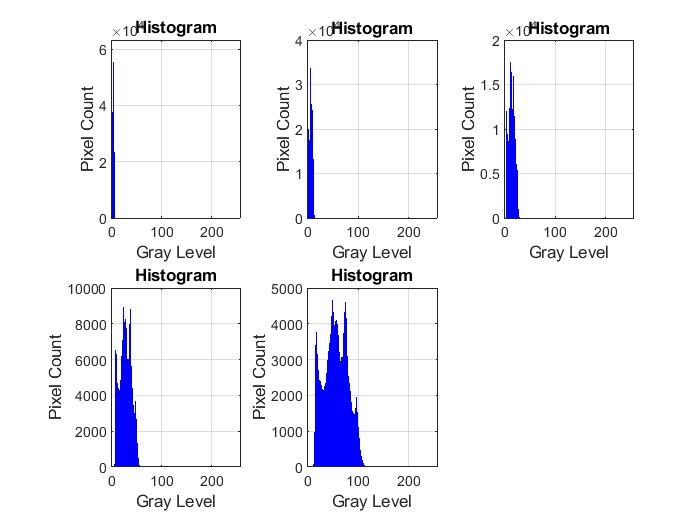
سپس به مقایسه نتایج حاصل می­پردازیم.

3--شکل‏ها، جدول‏ها و روابط (فرمول‏ها)

تصاویر و هیستوگرام­های خروجی :



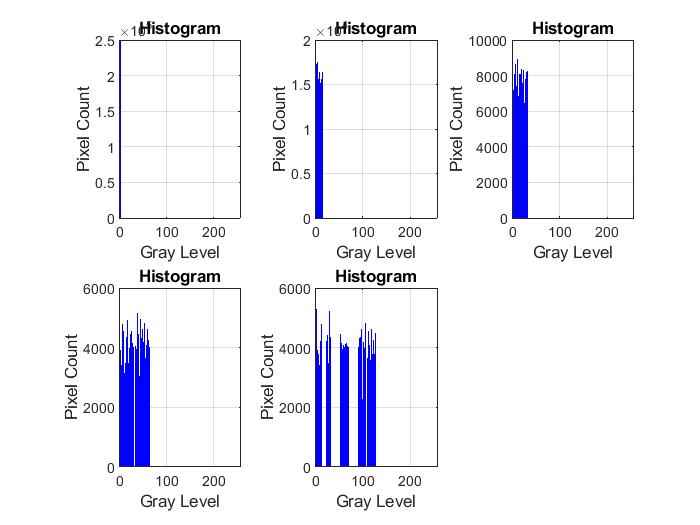
تصویر 1 – تصاویر non-equalized با quantization از 1 تا 5 به ترتیب 8مرحله، 16 مرحله، 32 مرحله، 64 مرحله و 128 مرحله



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

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

4-نتيجه 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(1, 256);

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

تمرين 1.2.2

نسيم فاني

|  |  |  |
| --- | --- | --- |
| اطلاعات گزارش |  | چکیده |
| **تاریخ: 13/8/99** |  | در اين تمرين، مي­خواهيم مقایسه­ای بین روش­های مختلف upsampling و downsampling داشته باشیم. |
| **واژگان كليدي:**  upsampling  downsamplin |  |

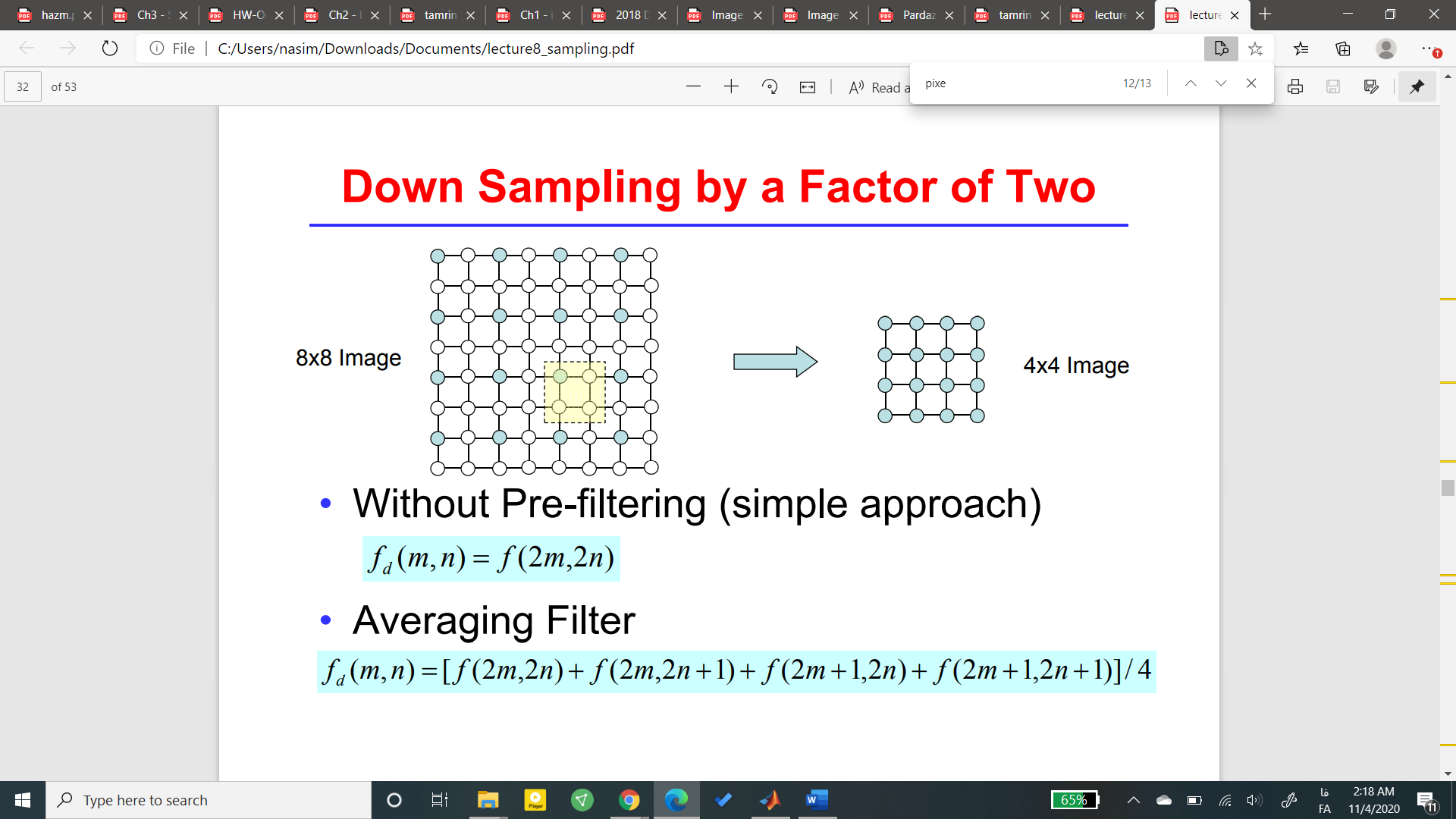
1-مقدمه[[5]](#footnote-5)

روش­های مختلف upsampling و downsampling را بررسی می­کنیم.

2-توضيحات تكنيكال

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:

Fd (m,n) = f (2m , 2n)

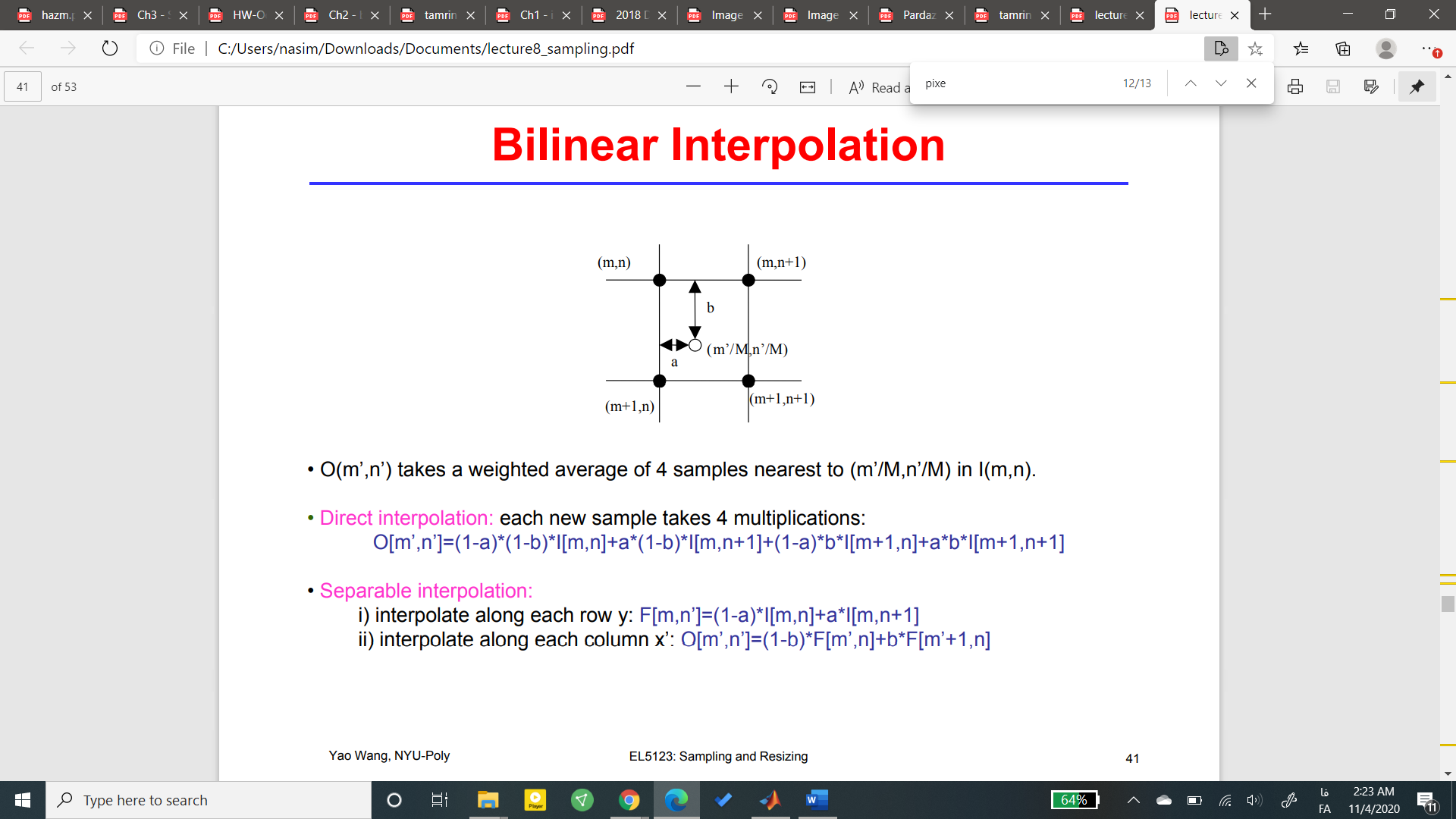
دوم- در این روش ابتدا فیلتر میانگین را بر روی تصویر اعمال کرده و سپس آن را با حذف سطر و ستون down-sample می­کنیم.

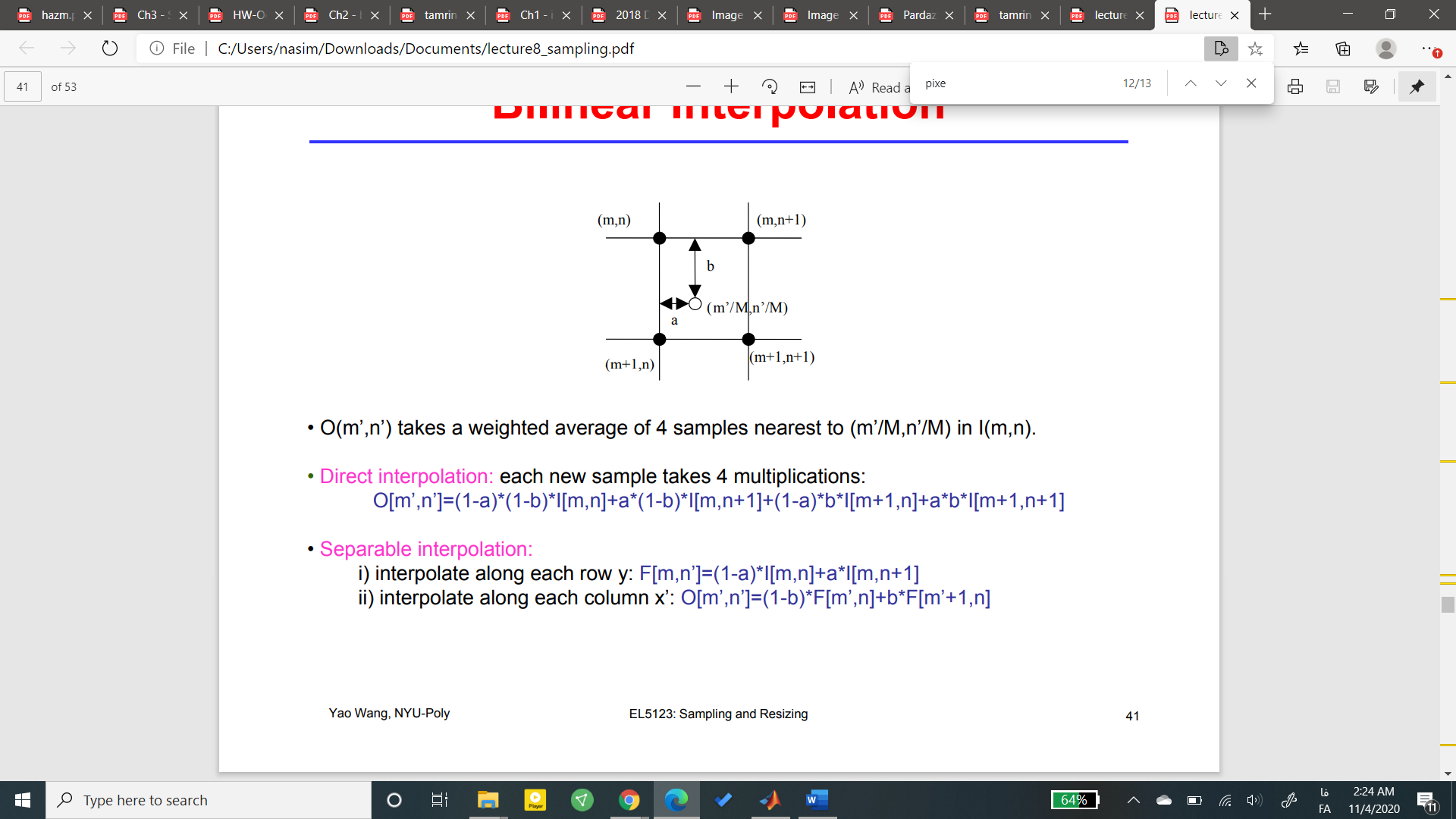
در این حالت داریم :

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

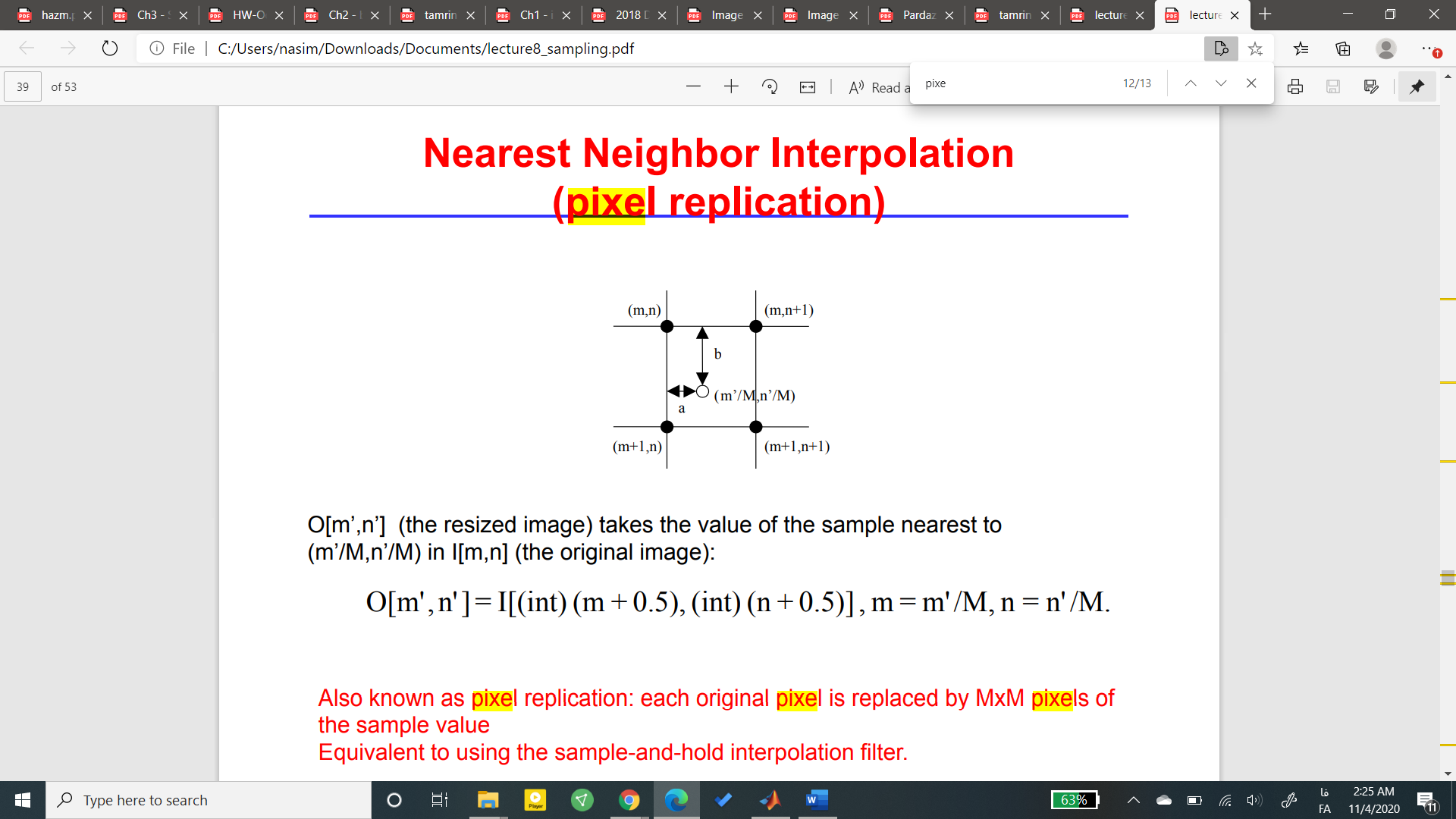
برایup-sample نیز از دو روش استفاده میکنیم :

اول- با کمک bilinear interpolation





دوم- نزدیک ترین همسایه



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

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

4-نتيجه گیری

هر چه MSE بیشتر 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);

i=1;j=1;

imgOut = zeros(rows/factor, columns/factor);

for x=1:factor:rows

for y=1:factor:columns

imgOut(i,j)=img(x,y);

j=j+1;

end

i = i+1;

j=1;

end

تابع averaging

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 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-

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)