دانشگاه فردوسی مشهد
16 اسفند 1396
فهیم جعفری

Image fundamentals

Geometric Transformation

Quantiztion & Interpolation

چکیده

تبدیلات هندسی روابط مکانی بین پیکسل ها را در یک تصویر، تبدیل اصلاح می کنند. این تبدیلات را اغلب تبدیلات صفحه کائوچو مینامند ، زیرا مشابه "چاپ" تصویر در صفحه ی کائوچوبی و سپس گسترش این صفحه بر اساس مجموعه ای از قوانین از پیش تعریف شده است. بر اساس تعاریف پردازش تصویر دیجیتال هندسی شامل دو عملیات (1) تبدیل مکانی مختصات و(2) درونیابی شدتت که مقادیر شدت را به پیکسل هایی که به صورت تبدیل شدند ، نسبت میدهد.

کلمات کلیدی

01 : 1.1.1
02 : 1.1.2
03 : 1.1.3
04 : 1.2.1
05 : 1.2.2

01

برای image registrartion دو تصویر با توابع affine باید تمام جایگشت های ضرب ماتریس های affine را با تمام مقادیر امتحان کردن و سرچ کرد تا زمانی که 3 یا بیشتر نقطه که در دو تصویر انتخاب کردیم بر روی هم منطبق شوند و هنگامی که ماتریس تبدیل نهایی بدست امد از ان برای انتقال سایر پیکسل ها استفاده میکنیم به اینصورت دو تصویر بر روی هم منطبق میشوند اگر تعداد نقاط انتخاب شده بیشتر باشد زمان پیدا کردن بیشتر ولی دقت ان نیز بیشتر میشود

02

یکی از روش های اصلی برای حل این مسئله ، استفاده از نقاط کنترلی است ، که نقاط متناظری هستند که مکان ان ها در تصاویر ورودی و مرجع دقیقا شناخته شده اند. روش های متعددی برای انتخاب نقاط کنترلی وجود دارد، مثل انتخاب محاوره ای انها تا اجرای الگوریتم هایی که سعی میکنند این نقاط را به طور خودکار تعیین کنند(sift). مسئله براورد تابع تبدیل ،یکی از موضوعات مدل سازی است . به عنوان مثال ،فرض کنید مجموعه ای از چهار نقطه کنترلی داریم که هر کدام در یک تصویر مرجع و ورودی قرار دارند.یک مدل ساده بر اساس براورد دو خطی که در حل همین تمرین استفاده شده است به صورت های زیر است:

$$X= C_1V + c_2w + c_3vw + c_4 \qquad y = c_5v + c_6w + c_7vw + c_8$$

که در مرحله فاز براورد ،(v,w)و(x,y) به ترتیب مختصات نقاط کنترلی در تصاویر ورودی و مرجع اند . اگر چهار جفت نقطه کنترلی در هر تصویر داشته باشیم ، می توانیم چهار معادله با استفاده از معادلات بالا بنویسیم و ار انها برای حل هشت ضریب مجهول c1,...,c8 استفاده کنیم.این ضرایب مدلی را میسازند که پیکسل های یک تصویر را به مکان هایی از پیکسل های نصویر دیگر تبدیل میکند تا ثبت صورت گیرد

03

یکی از روش های چرخش عکس استفاده از مختصات قطبی میباشد به اینصورت که ابتدا مختصات کارتزین هر پیکسل را تبدیل به مختصات قطبی میکنیم سپس این مختصات را به اندازه زاویه دلخواه چرخش داده و دوباره به مختصات کارتزین برمیگردانیم و در تصویر خروجی قرار میدهیم البته در بعضی از پیکسل در تصویر خروجی مقدار ندارند به علت گرد کردن اعداد در تبدیل قطبی به کارتزین که این مشکل را با درونیابی نزدیک ترین همسایه یا ... حل میکنیم

$$R = x^2 + y^2 \qquad \theta = \tan^{-1}\frac{y}{x}$$

$$X = r\cos\theta \quad y = r\sin\theta$$

04

گسسته سازی مپ کردن یک مجموعه بزرگ به یک مجموعه کوچک میباشد در این تمرین مجموعه اعداد از رنج پیوسته نامحدود 0تا 255 که بینهایت اعداد در ان وجود دارد به مجموعه اعداد محدود با تعداد دلخواه از 0 تا 255 مپ شده است .به این صورت که هر رنج خاصی از مجموعه نامحدود به عدد مورد نظر مپ میشود. که در این صورت اطلاعات تصویر از بین میرود و با تصویر اصلی تفاوت دارد.

برای اینکه تصویر کیفیت بهتری پیدا کند و کنتراست ان بهتر شود از histequalization استفاده میکنیم ولی با این که کیفیت تصویر را زیاد کردیم ولی خسارت هایی هم وارد میشود که ان هم تفاوت بیشتر تصویر با تصویر اصلی برای اینکه این تفاوت را نشان دهیم از msse استفاده میکنیم

05

از روش های تغییر سایز تصویر میتوان به حذف ردیف و ستون بصورت یکی در میان و درونیابی با استفاده از میانگین گیری همسایه ها و تکرار پیکسل ها و درونیابی دو خطی میباشد که هرکدام بصورت زیر میباشد:

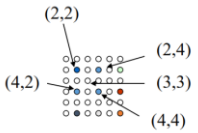
درونیابی دوخطی:

ابتدا چهار مجهول a,b,c,d را به دست اورده و برای پیکسل جدید استفاده میکنیم

$$v(x,y) = ax + by + cxy + d$$

$$125 = a(2) + b(2) + c(4) + d$$
$$170 = a(2) + b(4) + c(8) + d$$
$$172 = a(4) + b(2) + c(8) + d$$
$$170 = a(4) + b(4) + c(16) + d$$

$$v(3,3) = a(3) + b(3) + c(9) + d$$



درونیابی میانگین گیری:

مقادیر همسایه های مجاور با هم جمع میشود وبر تعداد انها تقسیم میکنیم

تکرار پیکسل ها :

هر پیکسل را یکباردر راستای x ویکبار در راستای y تکرار میکنیم و در تصویر خروجی قرار میدهیم

حذف ستون و ردیف:

یکی در میان از ستون ها و ردیف ها را حذف کرده و در تصویر خروجی قرار میدهیم

شرح نتایج

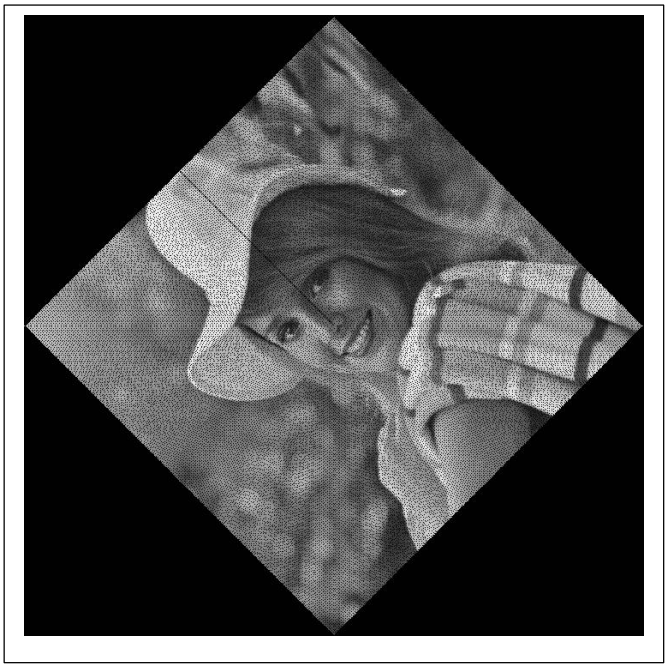
تصویر قبل از interpolation علت وجود نقطه در تصویر که تابع تبدیل بعضی مختصات ها را به علت گرد کردن در یک جا مپ کرده است

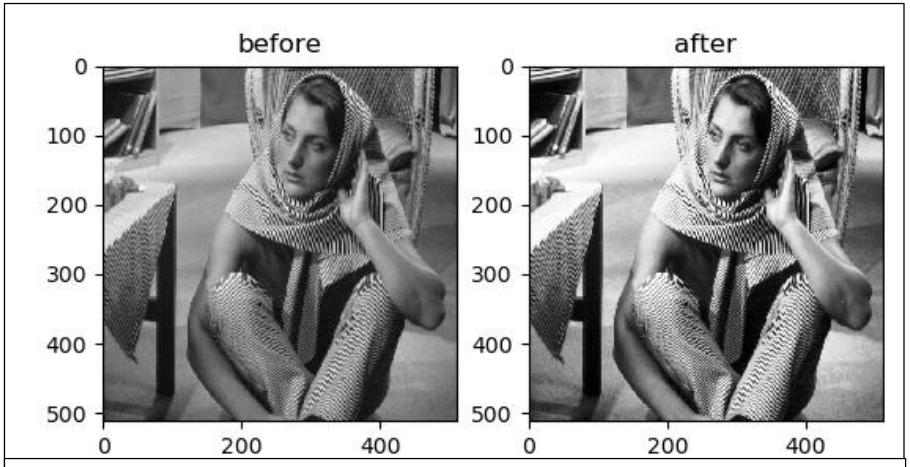
تصویر بعد از درونیابی

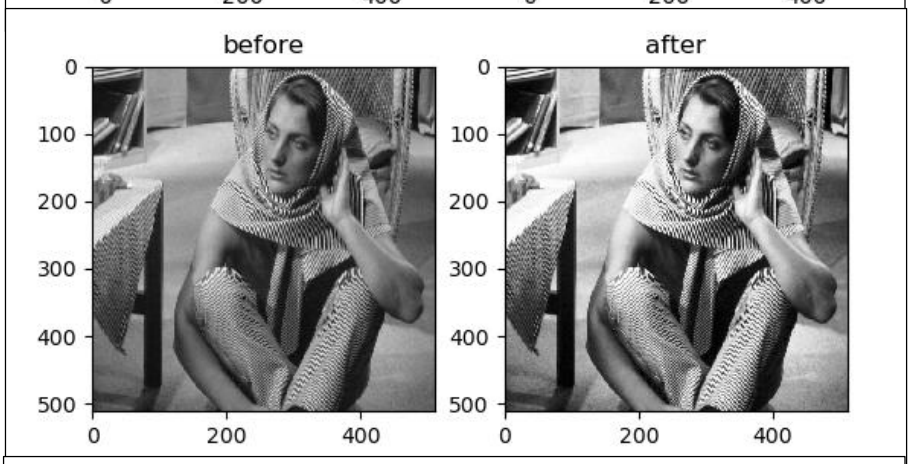
تصویر قبل از interpolation

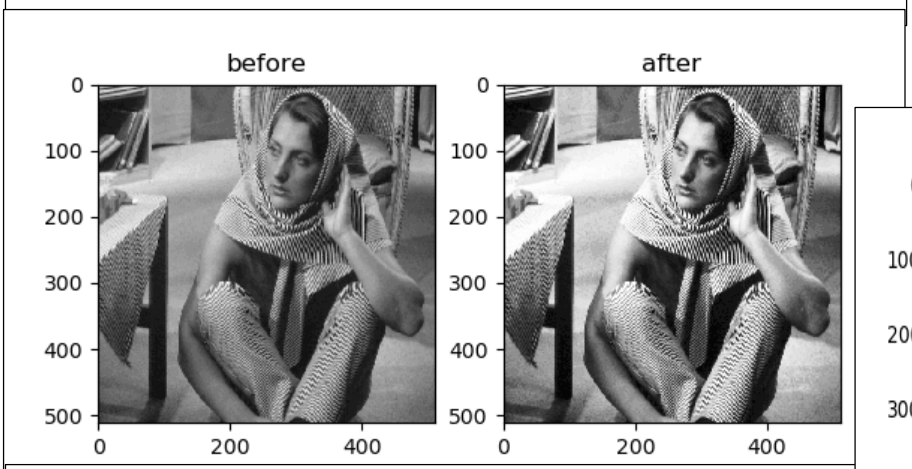
علت وجود نقطه در تصویر که تابع تبدیل بعضی مختصات ها را به علت گرد کردن در یک جا مپ کرده است

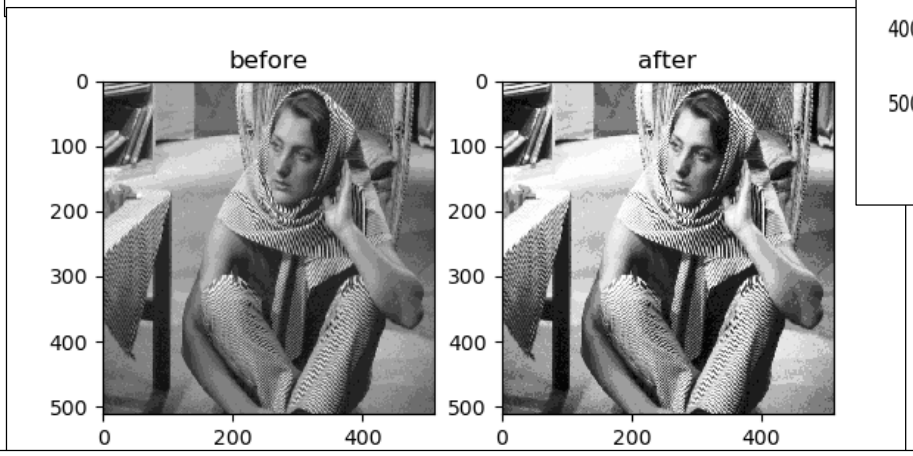
تصویر بعد از درونیابی







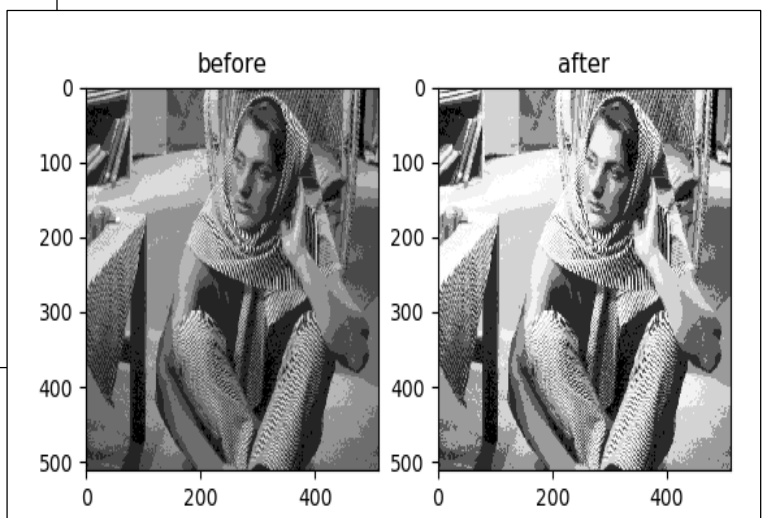




تصاویر از بالا به پایین به ترتیب 128 quantization و64و32و16و8 را نشان میدهند .

تصاویر سمت چپ قبل از اعمال hiseq میباشد و تصاویر سمت راست پس از اعمال .

همانطور که مشاهده میشود پس از اعمال histeq تصاویر دارای کنتراست بالاتر و تصویر با کیفیت بهتری نمایش داده میشود و هر چه به سمت پایین میرویم تفاوت بین دو تصویر کم میشود چرا که تفاوت level هایی که histeq تغییر میدهد کمتر میشود جدول مربوط به immse هم این امر را نشان میدهد (هر چه level ها کمتر میشود تفاوت mmse هم بین without و with هم کمتر میشود و از level16 کیفیت تصویر بشدت خراب میشود به علت Isopreference .



level	128	64	32	16	8
without histeq	0.4999809265136719	3.4942588806152344	17.51070785522461	76.13644027709961	91.59872055053711
with histeq	106.75680160522461	104.39081954956055	94.77213668823242	130.0508689880371	140.38868713378906

تصویر کاهش نمونه با حذف ستون و ردیف که کمی لبه ها تیز میباشد

تصویر upsample pixel replication شده ای که توسط میانگیری کاهش نمونه شده ولبه بیشتر نرم تر شده است و تفاوت با تصویر اصلی زیاد میباشد

تصویر کاهش نمونه با میانگین گیری که کمی لبه ها نرم تر به علت تاثیر پیکسل های مجاور میباشد

تصویر upsample bilinear شده ای که توسط میانگیری کاهش نمونه شده ولبه ها بیشترین نرم شدگی را دارند و تفاوت با تصویر اصلی از همه حالت های دیگر بیشتر میباشد

تصویر upsample bilinear شده ای که توسط حذف ردیف و ستون کاهش نمونه شده

تصویر upsample pixel replication شده ای که توسط حذف ستون و ردیف ولبه ها بیشترین تیز بودن را نسبت به بقیه دارد به علت اینکه بقیه پیکسل ها تاثیر در ان داده نشده است

	Pixel Replication	Bilinear Interpolation
averaging	169.99910481770831	339.2133513085716
Remove Row&Column	133.075927734375	209.96007537842195

شرح کد:

02 stitching

```python
import numpy as np
import cv2
import copy
xponits_img1 = [557,566,687,751]
ypoints_img1 = [492,428,418,489]
xponits_img2 = [133,144,269,334]
ypoints img2 = [514,447,437,506]

def interpolation(img):
    cop_im = copy.deepcopy(img)
    for i in range(0,img.shape[0]):
        for j in range(0,img.shape[1]):
            for k in range(0,img.shape[2]):
                if (img[i,j,k] == 1):
                    find = False
                    radius = 1
                    while(not find):
                        try:
                            if (img[i+radius,j,k] != 1):
                                cop_im[i,j] = img[i+radius,j]
                                find = True
                        except:
                            pass
                        try:
                            if img[i-radius,j,k] != 1:
                                cop_im[i,j] = img[i-radius,j]
                                find = True
                        except:
                            pass
                        try:
                            if img[i,j+radius,k] != 1:
                                cop_im[i,j] = img[i,j+radius]
                                find = True
                        except:
                            pass
                        try:
                            if img[i,j-radius,k] != 1:
                                cop_im[i,j] = img[i,j-radius]
                                find = True
                        except:
                            pass
                        if radius == 5:
                            find = True
                        radius += 1

    return cop_im

rgb im1 = cv2.imread('Car1.jpg')
size im1 = rgb im1.shape

#  read image 2 and convert to monochrome
rgb_im2 = cv2.imread('Car2.jpg')
size_im2 = rgb_im2.shape


ax = np.array([[xponits_img2[0],ypoints_img2[0],xponits_img2[0]*ypoints_img2[0],1],
               [xponits_img2[1],ypoints_img2[1],xponits_img2[1]*ypoints_img2[1],1],
               [xponits_img2[2],ypoints_img2[2],xponits_img2[2]*ypoints_img2[2],1],
               [xponits_img2[3],ypoints_img2[3],xponits_img2[3]*ypoints_img2[3],1]])
bx = np.array([xponits_img1[0],xponits_img1[1],xponits_img1[2],xponits_img1[3]])
x = np.linalg.solve(ax,bx)

ay = np.array([[xponits img2[0],ypoints img2[0],xponits img2[0]*ypoints img2[0],1],
               [xponits_img2[1],ypoints_img2[1],xponits_img2[1]*ypoints_img2[1],1],
               [xponits_img2[2],ypoints_img2[2],xponits_img2[2]*ypoints_img2[2],1],
               [xponits_img2[3],ypoints_img2[3],xponits_img2[3]*ypoints_img2[3],1]])
by = np.array([ypoints_img1[0],ypoints_img1[1],ypoints_img1[2],ypoints_img1[3]])
y = np.linalg.solve(ay,by)

res_im = np.ones((size_im1[0]+2*size_im2[0],size_im1[1]+2*size_im2[1],3))

for i in range(0,size_im1[0]):
    for j in range(0,size_im1[1]):
        for k in range(3):
            res im[i+size im1[0],j+size im1[1],k] = rgb im1[i,j,k]

for i in range(0,size im2[0]):
    for j in range(0,size_im2[1]):
        for k in range(3):
            xnew = int((x[0]*j)+(x[1]*i)+(x[2]*i*j)+x[3])
            ynew = int((y[0]*j)+(y[1]*i)+(y[2]*i*j)+y[3])
            xnew += size im1[1]
            ynew += size im1[0]
            res_im[ynew,xnew,k] = rgb_im2[i,j,k]

res_im = interpolation(res_im)
cv2.imwrite("res.jpg",res_im)
```

نقاط کنترلی

تابع درون یابی نزدیک ترین نقطه برای نقاطی که در تصویر به علت گرد کردن مختصات مقدار نگرفته اند

خواندن عکس ها

درست کردن 2 دستگاه 4 معادله و 4 مجهول مربوط به فرمول دوخطی

درست کردن تصویر خروجی

گزاشتن تصویر اول در تصویر خروجی

گزاشتن تصویر دوم در تصویر خروجی

درونیابی

چاپ تصویر خروجی

03 rotation

```python
import numpy as np
import cv2
import copy

def convert2Polar(x,y):
    r = int( ( (x**2) + (y**2) )**(1/2) )
    theta = np.arctan(y/x) if x!=0 else (np.arctan(-np.inf) if y < 0 else np.arctan(+np.pi))
    theta = theta+np.pi if x < 0 else theta
    return r,theta

def conver2Coordinate(r,theta):
    x = r*np.cos(theta)
    y = r*np.sin(theta)
    return int(x),int(y)

def interpolation(img):
    cop im = copy.deepcopy(img)
    for i in range(0,img.shape[0]):
        for j in range(0,img.shape[1]):
            for k in range(0,img.shape[2]):
                if (img[i,j,k] == 1):
                    find = False
                    radius = 1
                    while(not find):
                        try:
                            if (img[i+radius,j,k] != 1):
                                cop im[i,j] = img[i+radius,j]
                                find = True
                        except:
                            pass
                        try:
                            if img[i-radius,j,k] != 1:
                                cop im[i,j] = img[i-radius,j]
                                find = True
                        except:
                            pass
                        try:
                            if img[i,j+radius,k] != 1:
                                cop im[i,j] = img[i,j+radius]
                                find = True
                        except:
                            pass
                        try:
                            if img[i,j-radius,k] != 1:
                                cop_im[i,j] = img[i,j-radius]
                                find = True
                        except:
                            pass
                        if radius == 5:
                            find = True
                        radius += 1

    return cop_im



rgb im1 = cv2.imread('Elaine.bmp')
w = int(rgb_im1.shape[1])
h = int(rgb_im1.shape[0])
size = int( (w**2+h**2)**(1/2) )
X = np.ones((size,size,rgb_im1.shape[2]),dtype='uint8')

print(X.shape)
rotated_theta_degree = 80
rotate theta = rotated theta degree/180*np.pi
mid_w = rgb_im1.shape[1]/2
mid h = rgb im1.shape[0]/2

for i in range(0,rgb_im1.shape[0]):
    for j in range(0,rgb_im1.shape[1]):
        r,theta = convert2Polar(j-mid_w,-i+mid_h)
        theta +=  rotate_theta
        xpos,ypos = conver2Coordinate(r,theta)
        xpos = round(xpos+size/2)
        ypos = round((-ypos)+size/2)
        X[ypos,xpos] = rgb_im1[i,j]

# X = interpolation(X)
cv2.imwrite('re2.jpg',X)
```

تبدیل مختصات کارتزین به مختصات قطبی

تبدیل مختصات قطبی به کارتزین

درونیابی به نزدیک ترین همسایه

اندازه لازم برای تصویر خروجی برای تمام چرخش هاو تشکیل ماتریس ان

تبدیل زوایه به گرادیان

تبدیل کارتزین به قطبی و چرخش با زوایه مربوطه و تبدیل به کارتزین

04 quantization

```python
import cv2
import copy
import matplotlib.pyplot as plt
import numpy as np

def quantizeImage(img, level):
    cop_img = copy.deepcopy(img)
    size_segment = int(256/level)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            cop img[i, j] = int(img[i, j]/size segment)*size segment
    return cop img


img = cv2.imread('..\Barbara.bmp')
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

img128o = quantizeImage(img, 128)
img64o = quantizeImage(img, 64)
img32o = quantizeImage(img, 32)
img16o = quantizeImage(img, 16)
img8o = quantizeImage(img, 8)


img128w = cv2.equalizeHist(img128o)
img64w = cv2.equalizeHist(img64o)
img32w = cv2.equalizeHist(img32o)
img16w = cv2.equalizeHist(img16o)
img8w = cv2.equalizeHist(img8o)


mmse1280 = np.square(np.subtract(img,img128o)).mean()
mmse640 = np.square(np.subtract(img,img64o)).mean()
mmse320 = np.square(np.subtract(img,img32o)).mean()
mmse160 = np.square(np.subtract(img,img16o)).mean()
mmse80 = np.square(np.subtract(img,img8o)).mean()

mmse128w = np.square(np.subtract(img128o,img128w)).mean()
mmse64w = np.square(np.subtract(img64o,img64w)).mean()
mmse32w = np.square(np.subtract(img32o,img32w)).mean()
mmse16w = np.square(np.subtract(img16o,img16w)).mean()
mmse8w = np.square(np.subtract(img8o,img8w)).mean()


data = [["without histeq", mmse1280, mmse640, mmse320, mmse160, mmse80],
        ["with histeq", mmse128w, mmse64w, mmse32w, mmse16w,mmse8w]]

col_labels = ("level",128, 64, 32, 16, 8)

fig, ax = plt.subplots(dpi=300, figsize=(5,1))
ax.axis('off')
ax.table(cellText=data, colLabels=col labels, loc='center')
fig.savefig('../images/re6.png')

fig1, axs = plt.subplots(1, 2)
axs[0].imshow(img128o, cmap='gray')
axs[0].set_title("before")
axs[1].imshow(img128w, cmap='gray')
axs[1].set_title("after")
fig1.savefig('../images/re1.png')

fig1, axs = plt.subplots(1, 2)
axs[0].imshow(img64o, cmap='gray')
axs[0].set_title("before")
axs[1].imshow(img64w, cmap='gray')
axs[1].set title("after")
fig1.savefig('../images/re2.png')

fig1, axs = plt.subplots(1, 2)
axs[0].imshow(img32o, cmap='gray')
axs[0].set_title("before")
axs[1].imshow(img32w, cmap='gray')
axs[1].set_title("after")
fig1.savefig('../images/re3.png')

fig1, axs = plt.subplots(1, 2)
axs[0].imshow(img16o, cmap='gray')
axs[0].set title("before")
axs[1].imshow(img16w, cmap='gray')
axs[1].set title("after")
fig1.savefig('../images/re4.png')

fig1, axs = plt.subplots(1, 2)
axs[0].imshow(img8o, cmap='gray')
axs[0].set title("before")
axs[1].imshow(img8w, cmap='gray')
axs[1].set_title("after")
fig1.savefig('../images/re5.png')
```

تابع مربوط به گسسته سازی با تعداد سطح دلخواه

گسسته سازی تصویر با levelهای128و64و32و16و8 بدون histeq

اعمال تابع histeq به تصاویر گسسته شده با سطح های مختلف

اندازه گیری mmse تصاویر گسسته شده بدون تعادل هستوگرام و با تعادل هیستوگرام با تصویر اصلی

درست کردن ردیف و ستون های جدول مربوط به گزارش mmse با mmseهای اندازه گیری شده

ذخیره تصاویر گسسته شده بدون histeq و با histeq

```python
import cv2
import copy
import matplotlib.pyplot as plt
import numpy as np

def dwnsmple_averaging(img):
    width = int(img.shape[1]/2)
    height = int(img.shape[0]/2)
    dwn smpl img = np.zeros((height,width))
    for i in range(0,height-1):
        for j in range(0,width-1):
            dwn_smpl_img[i, j] = (int(img[i*2, j*2])+int(img[i*2, j*2+1]) + \
                                  int(img[i*2+1, j*2])+int(img[i*2+1, j*2+1]) )/4
    return dwn_smpl_img

def dwnsmple_rmvRowAndCol(img):
    width = int(img.shape[1]/2)
    height = int(img.shape[0]/2)
    dwn_smpl_img = np.zeros((height,width))
    for i in range(0,height):
        for j in range(0,width):
            dwn smpl img[i, j] = img[i*2, j*2]
    return dwn smpl img


def upsmple_pRreplication(dwn_smpl_img):
    width = dwn_smpl_img.shape[1]*2
    height = dwn smpl img.shape[0]*2
    up smpl img pRplctn = np.ones((height,width))
    for i in range(0,height):
        for j in range(0,width):
            up_smpl_img_pRplctn[i, j] = dwn_smpl_img[int(i/2), int(j/2)]

    return up_smpl_img_pRplctn

def upsmple bilinear(dwn smpl img):
    width = dwn smpl img.shape[1]*2
    height = dwn_smpl_img.shape[0]*2
    up_smpl_img = np.ones((height+2,width+2))*128
    for i in range(2,height,2):
        for j in range(2,width,2):
            up smpl img[i, j] = dwn smpl img[int(i/2), int(j/2)]
    for i in range(1,height,2):
        for j in range(2,width,2):
            a = np.array([[i+1,j-1,(i+1)*(j-1),1],
                 [i+1,j+1,(i+1)*(j+1),1],
                 [i-1,j+1,(i-1)*(j+1),1],
                 [i-1,j-1,(i-1)*(j-1),1]])
            b = np.array([up smpl img[i+1, j-2], up smpl img[i+1, j+2],
                 up_smpl_img[i-1, j+2], up_smpl_img[i-1, j-2]])
            x = np.linalg.solve(a,b)

            up_smpl_img[i, j] = x[0]*i + x[1]*j + x[2]*i*j + x[3]
    for i in range(1,height):
        for j in range(1,width,2):
            a = np.array([[i+1,j-1,(i+1)*(j-1),1],
                          [i+1,j+1,(i+1)*(j+1),1],
                          [i-1,j+1,(i-1)*(j+1),1],
                          [i-1,j-1,(i-1)*(j-1),1]])
            b = np.array([up_smpl_img[i+1, j-1], up_smpl_img[i+1, j+1],
                          up_smpl_img[i-1, j+1], up_smpl_img[i-1, j-1]])
            x = np.linalg.solve(a,b)

            up_smpl_img[i, j] = x[0]*(i) + x[1]*j + x[2]*i*j + x[3]


    return up_smpl_img


img = cv2.imread('..\Goldhill.bmp')
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

dwn_smpl_img_averaging = dwnsmple_averaging(img)
cv2.imwrite('down_sample_img_averaging.jpg', dwn_smpl_img_averaging)

dwn_smpl_img = dwnsmple_rmvRowAndCol(img)
cv2.imwrite('down_sample_img_rmv_row.jpg', dwn_smpl_img)


avg2pp = upsmple_pRreplication(dwn_smpl_img_averaging)
avg2bi = upsmple_bilinear(dwn_smpl_img_averaging)
rrc2pp = upsmple pRreplication(dwn smpl img)
rrc2bi = upsmple bilinear(dwn smpl img)
cv2.imwrite('avg2pp.jpg',avg2pp )
cv2.imwrite('avg2bi.jpg', avg2bi )
cv2.imwrite('rrc2pp.jpg', rrc2pp)
cv2.imwrite('rrc2bi.jpg', rrc2bi)


mmse_avg2pp = np.square(np.subtract(img,avg2pp)).mean()
mmse_avg2bi = np.square(np.subtract(img,avg2bi[1:avg2bi.shape[0]-1,1:avg2bi.shape[1]-1])).mean()
mmse_rrc2pp = np.square(np.subtract(img,rrc2pp)).mean()
mmse_rrc2bi = np.square(np.subtract(img,rrc2bi[1:rrc2bi.shape[0]-1,1:rrc2bi.shape[1]-1])).mean()


data = [["averaging", mmse_avg2pp, mmse_avg2bi],
```

تابع مربوط بهکاهش نمونه با میانگیری از 4 همسایه مجاور

کاهش نمونه با حذف ردیف و ستون به صورت یکی درمیان

افزایش سایز با تکرار پیکسل ها کل تصویر یکی برای ردیف و یکی برای ستون

افزایش سایز با فرمول دوخطی که دو دستگاه چهار معادله و چهار مجهول در ان حل شده ابتدا برای افزایش ردیف و سپس افزایش برای ستون

کاهش نمونه میانگینگیری

کاهش نمونه حذف ستون و.

افزایش نمونه با دوخطی و تکرار پیکسل تصاویر کاهش نمونه یافته قبلی

بدست اوردن mmse

```
        ["Remove Row&Column", mmse_rrc2pp, mmse_rrc2bi]]

col_labels = (" ", "Pixel Replication", "Bilinear Interpolation ")
fig, ax = plt.subplots(dpi=300, figsize=(5,1))
ax.axis('off')
ax.table(cellText=data, colLabels=col_labels, loc='center')
fig.savefig('re.png')
```