به نام خدا

تمرین سری دوم درس دید کامپیوتری

**دانشجو: محمدجواد قربانعلی وکیلی**

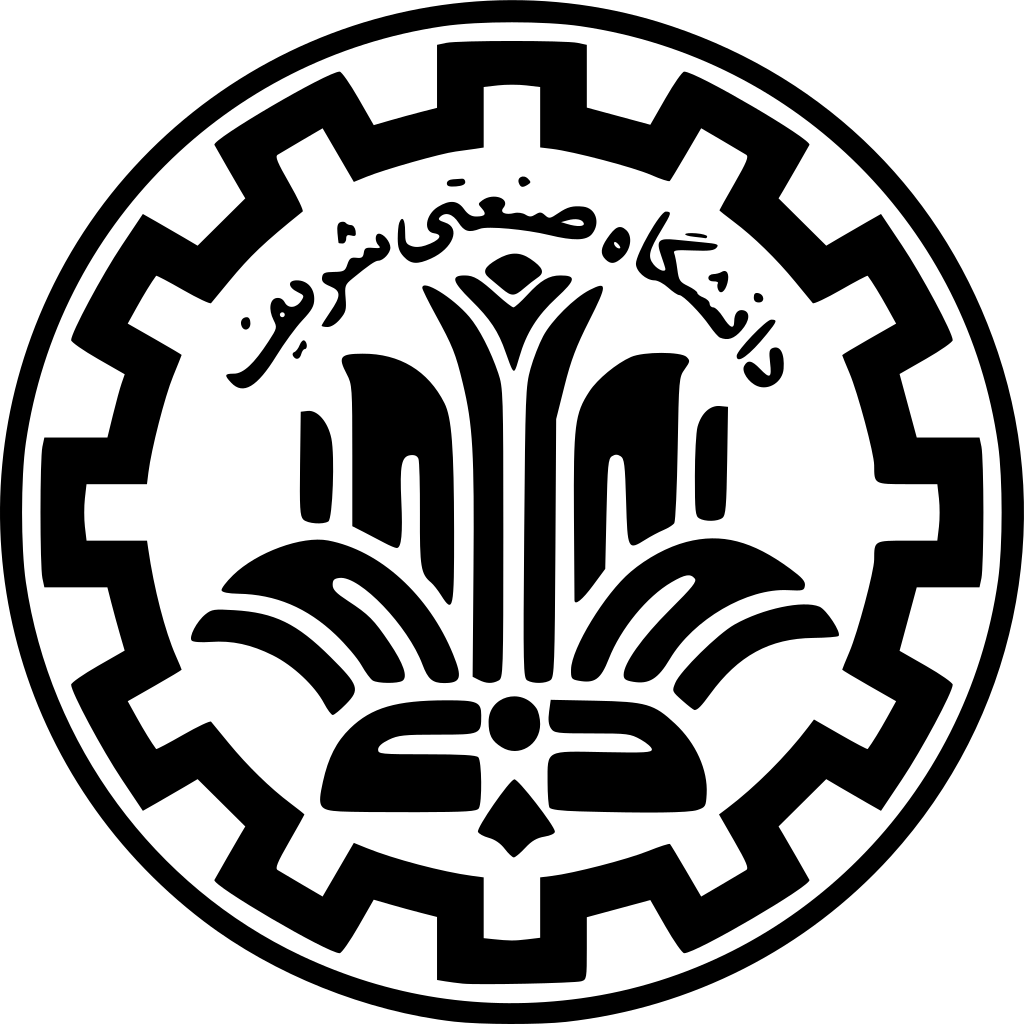
**شماره دانشجویی: 98210292**

**استاد: دکتر نرجس‌الهدی محمدزاده**

**نیمسال تحصیلی دوم 99-98**

**دانشکده مهندسی مکانیک**

**دانشگاه صنعتی شریف**



**تمرین الف-3**

در این تمرین، هدف تغییر سایز شکل 1 و اعمال یک فیلتر پایین‌گذر به آن و سپس طراحی کرنل فیلترهای لبه‌یاب عمودی و افقی و اعمال آن به شکل مورد نظر می‌باشد. اعمال این فیلترها می‌‍‌بایست بدون استفاده از توابع آماده opencv باشد. مراحل انجام این عملیات به تفصیل بیان می‌گردد.



**بخش اول:** در اولین بخش از این تمرین، هدف اعمال فیلتر میانگین به عنوان فیلتر پایین‌گذر به تصویر در فرمت RGB می‌باشد. گام‌های انجام این عملیات عبارتند از:

1. خواندن تصویر و تغییر سایز آن.

#############################################LPF in RGB image##

pic=cv2.imread('2.jpg')

pic=cv2.resize(pic,(400,300))

1. طراحی کرنل فیلتر میانگین 3×3 و سپس ایجاد یک numpy array با المان‌های سه بعدی. علت ایجاد این آرایه، همانطور که در بخش بعدی ذکر خواهد شد، zero pad کردن تصویر اولیه است. هم‌چنین با توجه به آن‌که تصویر اولیه در فرمت RGB است، المان‌های numpy array نیز می‌بایست سه بعدی باشند.

#kernel for mean filter

kernel = (1/9)\*np.ones((3,3), np.uint8)

#define for zero padding

pic\_pad = np.zeros((302,402,3), np.uint8)

1. Zero pad کردن تصویر اولیه

#zero padded pic

pic\_pad[1:301,1:401]=pic

1. در این گام، هدف اعمال کرنل تعریف شده به تصویر zero pad شده می‌باشد. بدین منظور برای فیلتر کردن هر پیکسل،ابتدا در کد، مرکز کرنل بر روی پیکسل مورد نظر قرار داده شده و سپس عملیات ضرب درایه‌ای و جمع انجام می‌پذیرد. در نهایت نیز تصویر فیلتر شده به صورت شکل 2 تولید می‌‌گردد.

for x in range(1,301):

for y in range(1,401):

#picks a 3\*3 section

section=pic\_pad[x-1:x+2,y-1:y+2]

#element-wise multiplication

q=kernel\*section

#sum of elements

pic\_pad[x,y]=q[0,0]+q[0,1]+q[0,2]+q[1,0]+q[1,1]+q[1,2]+q[2,0]+q[2,1]+q[2,2]

pic\_LPF=pic\_pad[1:301,1:401]

**بخش دوم:** در این بخش، هدف اعمال فیلتر میانگین به عنوان فیلتر پایین‌گذر، به شکل 1 در فرمت gray می‌باشد. لذا تمامی مراحل مشابه مراحل بخش اول می‌باشد، با این تفاوت که در این قسمت با توجه به فرمت تصویر، المان‌های numpy array بایستی یک بعدی باشند. در نهایت تصویر نهایی به صورت شکل 3 می‌گردد.

############################################LPF in gray scale##

pic\_gray=cv2.cvtColor(pic,cv2.COLOR\_BGR2GRAY)

kernel = (1/9)\*np.ones((3,3), np.uint8)

pic\_pad = np.zeros((302,402), np.uint8)

pic\_pad[1:301,1:401]=pic\_gray

for x in range(1,301):

for y in range(1,401):

section=pic\_pad[x-1:x+2,y-1:y+2]

q=kernel\*section

pic\_pad[x,y]=q[0,0]+q[0,1]+q[0,2]+q[1,0]+q[1,1]+q[1,2]+q[2,0]+q[2,1]+q[2,2]

pic\_lpf=pic\_pad[1:301,1:401]

**بخش سوم:** در این بخش، هدف اعمال یک فیلتر لبه‌یاب در جهت y (افقی) به تصویر فیلتر شده بخش دوم (شکل 3) می‌باشد. گام‌های انجام این عملیات به شرح زیر می‌باشد.

1. Zero pad کردن تصویر اولیه و تعریف کرنل به منظور انجام لبه‌یابی در جهت y. لبه‌های یک تصویر، پیکسل‌هایی هستند که در آن‌ها intensity پیکسل به طور ناگهانی و با شیب زیاد نسبت به پیکسل مجاور تغییر می‌کند. به همین دلیل برای یافتن این لبه‌ها، کرنل مورد نظر طوری طراحی شده که اختلاف intensity یک پیکسل را با پیکسل مجاور در جهت y محاسبه می‌نماید.

#######################################detect edge in y direction#

pic\_pad = np.zeros((302,402), np.uint8)

pic\_pad[1:301,1:401]=pic\_lpf

kernel = np.array([[0,-1,1]])

1. در این گام، هدف اعمال کرنل تعریف شده به تصویر zero pad شده می‌باشد. بدین منظور برای فیلتر کردن هر پیکسل،ابتدا در کد مرکز کرنل بر روی پیکسل مورد نظر قرار داده شده و سپس عملیات ضرب درایه‌ای و جمع انجام می‌پذیرد. در نهایت تصویری تولید می‌گردد که intensity هر پیکسل از آن، متناسب با آهنگ تغییر intensity همان پیکسل در تصویر اولیه است. لذا هرچه این لبه‌ها تیز تر باشند، در تصویر نهایی روشن‌تر خواهند بود. پس از اعمال threshold (که در ادامه شرح داده خواهد شد) تصویر نهایی به صورت شکل 4 خواهد بود.

for x in range(1,301):

for y in range(1,401):

section=pic\_pad[x,y-1:y+2]

q=kernel\*section

s=q[0,0]+q[0,1]+q[0,2]

pic\_pad[x,y]=abs(s)

pic\_y=pic\_pad[1:301,1:401]

**بخش چهارم:** در این بخش، هدف اعمال یک فیلتر لبه‌یاب در جهت x (عمودی) به تصویر فیلتر شده بخش دوم (شکل 3) می‌باشد. گام‌های انجام این عملیات کاملاً مشابه بخش سوم می‌باشد، با این تفاوت که این‌بار عملیات اعمال کرنل به تصویر به صورت عمودی انجام می‌پذیرد. پس از اعمال threshold (که در ادامه شرح داده خواهد شد) تصویر نهایی به صورت شکل 5 خواهد بود.

#######################################detect edge in x direction

pic\_pad = np.zeros((302,402), np.uint8)

pic\_pad[1:301,1:401]=pic\_lpf

kernel = np.array([[0,-1,1]])

for x in range(1,301):

for y in range(1,401):

section=pic\_pad[x-1:x+2,y]

q=kernel\*section

s=q[0,0]+q[0,1]+q[0,2]

pic\_pad[x,y]=abs(s)

pic\_x=pic\_pad[1:301,1:401]

**بخش پنجم:** فیلترهای لبه یاب بخش‌های سوم و چهارم، اثر تغییر شدید intensity را فقط در یک راستا بررسی می‌کنند و لذا اگر در راستای دیگر لبه وجود داشته باشد (تغییر ناگهانی intensity)، آن را در نظر نمی‌گیرند. لذا در این بخش، هدف در نظر گرفتن هم‌زمان اثر تغییر ناگهانی intensity در هر دو راستا به منظور شناسایی بهینه لبه‌ها می‌باشد. به منظور انجام این کار، مجذور تغییر intensity در هر دو راستا با یکدیگر جمع شده و سپس از عبارت حاصل جذر گرفته می‌شود. پس از اعمال threshold (که در ادامه شرح داده خواهد شد) تصویر نهایی به صورت شکل 6 خواهد بود.

##################################################edge detection#

edge = np.zeros((300,400), np.uint8)

for x in range(0,300):

for y in range(0,400):

a=int((pic\_x[x,y]\*\*2 + pic\_y[x,y]\*\*2)\*\*0.5)

if a>255:

edge[x,y]=255

else:

edge[x,y]=a

**بخش ششم:** در این بخش، ابتدا تصاویر حاصل از اعمال فیلترهای لبه‌یاب، به منظور بهبود وضوح تصویر، به کمک دستور threshold به تصویر binary تبدیل شده و سپس دستور نمایش تمام تصاویر نوشته می‌شود. هم‌چنین کاربر با فشردن دکمه e تمام پنجره‌ها را بسته و یا با فشردن دکمه s، ضمن بستن تمامی پنجره‌ها، تصاویر حاصل از فیلترهای لبه‌یاب عمودی و افقی را در directory با نام‌های نوشته شده ذخیره می‌نماید.

\_,edge=cv2.threshold(edge,10,255,cv2.THRESH\_BINARY)

\_,pic\_y=cv2.threshold(pic\_y,10,255,cv2.THRESH\_BINARY)

\_,pic\_x=cv2.threshold(pic\_x,10,255,cv2.THRESH\_BINARY)

cv2.imshow('INITIAL IMAGE',pic)

cv2.imshow('EDGE',edge)

cv2.imshow('INITIAL GRAY IMAGE',pic\_gray)

cv2.imshow('LPF GRAY',pic\_lpf)

cv2.imshow('Y EDGE',pic\_y)

cv2.imshow('X EDGE',pic\_x)

cv2.imshow('LPF RGB',pic\_LPF)

while True:

key=cv2.waitKey(0)

if key==ord('e'):

cv2.destroyAllWindows()

break

elif key==ord('s'):

cv2.imwrite('X.jpg',pic\_x)

cv2.imwrite('Y.jpg',pic\_y)

cv2.destroyAllWindows()

break

else:

 pass





در شکل 4 که فیلتر در جهت y (افقی) اعمال شده است، ملاحظه می‌گردد لبه‌هایی که در این جهت قرار دارند به خوبی نمایان شده‌اند در حالی که در شکل 5 که فیلتر در جهت x (قائم) اعمال شده، لبه‌ها در جهت x به وضوح مشخص هستند (به عنوان مثال در شکل 4 بر خلاف شکل 5، ساقه گل به خوبی نمایان است در حالی که در شکل 5 بر خلاف شکل 4، قسمت انتهایی گلبرگ سمت چپ به خوبی نمایان می‌باشد). حال در شکل 6 که اثر توأم این دو فیلتر لبه‌یاب اعمال شده است، ملاحظه می‌گردد که تمامی لبه‌ها با وضوح بیشتری قابل رؤیت هستند. در واقع شکل 6، تصویر بالاگذر شده شکل 1 در فرمت binary می‌باشد.