نقاب گذاری غیر تیز

منا رداد

اطلاعات گزارش

تاریخ:
1400/9/5

واژگان کلیدی:
فیلتر هموار ساز
فیلتر میانگین
لاپلاسین
فیلتر میانه
نقاب گذاری
کلیشه

چکیده

در این مقاله به بررسی و کاربرد نقاب گذاری غیرتیز (Un-sharp masking) پرداختیم. *به منظور انجام این فرایند از فیلتر های جعبه ای و فیلتر میانگین وزن دار در اندازه ها و ابعاد مختلف استفاده می کنیم. برای هموارسازی تصاویر باید از فیلتر هموارساز استفاده کنیم که فیلتر میانه را تحت عنوان فیلتر هموارساز به تصاویر اعمال می کنیم. در پایان به کمک یک مقدار مشخص تحت عنوان آستانه گذاری و به صورت وفقی و غیر وفقی کلیشه های لاپلاسین را به تصویر اعمال کردیم و به بررسی آن پرداختیم.*

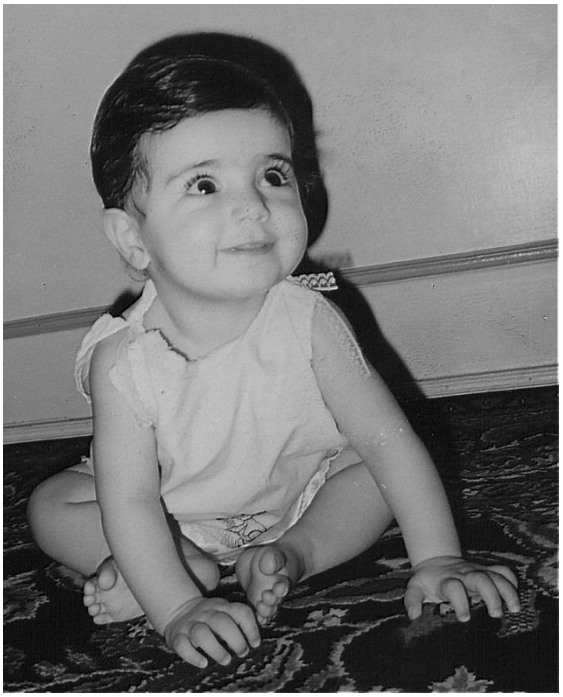
1-مقدمه

هدف از انجام این تمرین بررسی عملیات Un-sharp Masking یا نقاب گذاری غیر تیز و اعمال آن بر روی تصاویر است. نقاب گذاری غیر تیز تحت عنوان تکنیکی برای شفاف سازی تصاویر استفاده می کنیم که در ابتدا آن را در عکاسی تاریک خانه ها اجرا کردند، امروزه از این تکنیک در نرم افزار پردازش تصویر دیجیتال استفاده می شود. این روش یک تصویر شارپ نشده و تار را برای ساخت ماسک از تصویر اصلی به کار میگیرد و علت نامگذاری آن هم به همین خاطر است. برای اینکه تصویر از تصویر اصلی میزان تار بودن کمتری داشته باشد از ترکیب ماسک غیر واضح را با تصویر مثبت اصلی استفاده میشود. تصویر نهایی بدست آمده اگرچه واضح تر است، اما ممکن است نمایش دقیق تری از موضوع تصویر باشد.

1-2-فیلتر جعبه ای و Un-sharp Masking

برای انجام این قسمت از تمرین فیلتر هموارسازی را به تصویر child اعمال می کنیم که این فیلتر هموارساز یک فیلتر هموار ساز میانگین از نوع فیلتر جعبه ای و با ابعاد 3*3 می باشد که همانطور که در نتایج بدست آمد تصویر جدید child در مقایسه با تصویر اولیه اش دارای وضوح بیشتر است و در آن لبه ها بیشتر قابل مشاهده و مشخص تر هستند که نتایج بدست آمده را در زیر مشاهده می کنید:

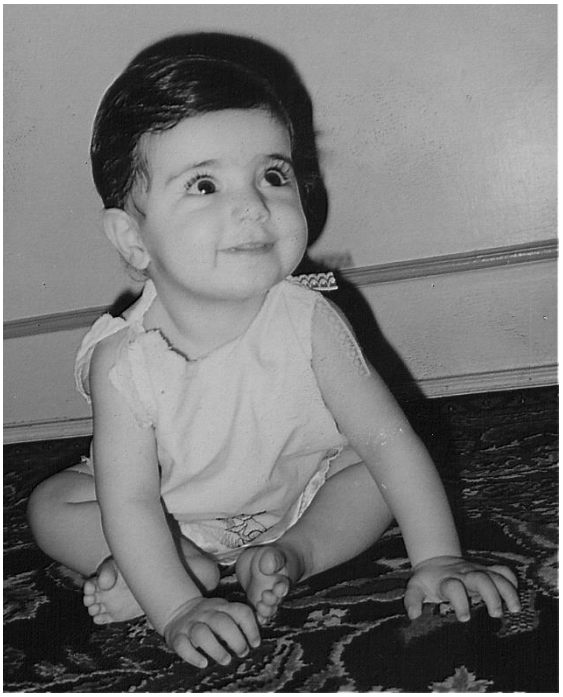
تصویر اصلی



تصویر با اعمال فیلتر میانگین box filter با ابعاد 3*3

2-2-میانگین وزن دار و Un-sharp Masking

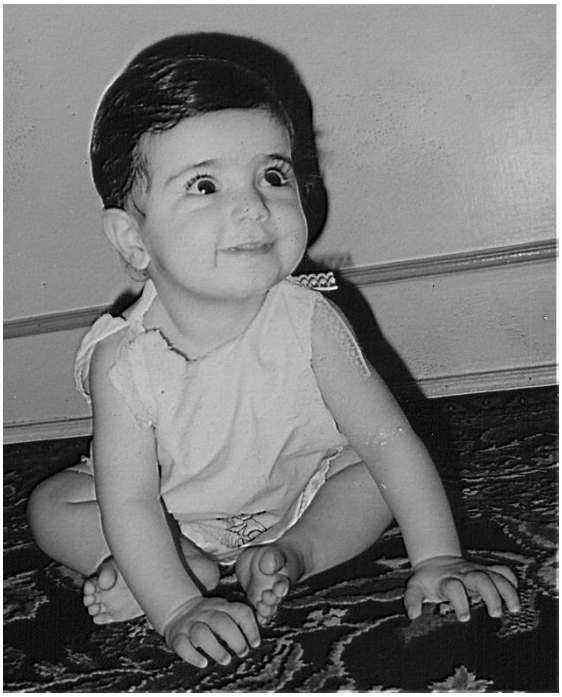
تمامی مراحل انجام شده در قسمت قبل را در این قسمت با اعمال فیلتر هموارساز میانگین وزن دار به تصویر اعمال می کنیم که نتایج آن به این صورت می باشد :



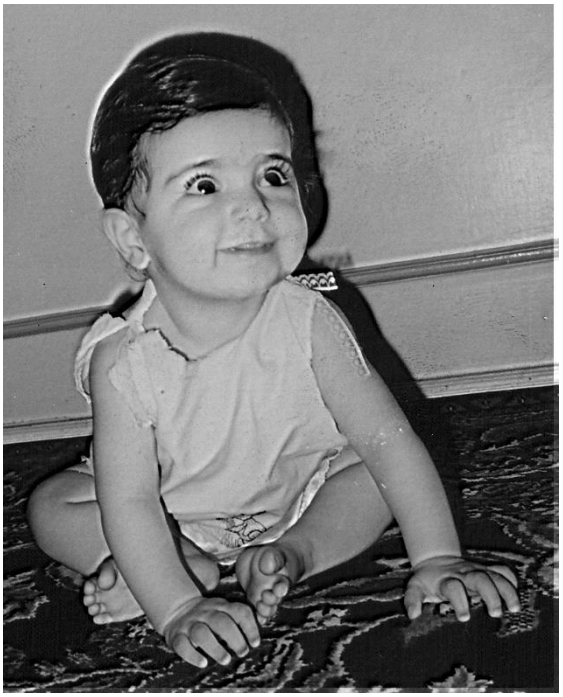
از نتیجه بدست آمده و مقایسه آن با تصویر اصلی میتوان مشاهده کرد که این دو تصویر دارای تفاوت های اندکی با یکدیگر می باشند و حال اگر این نتیجه را با تصویر مرحله قبل که از فیلتر جعبه ای استفاده کرده ایم مقایسه کنیم متوجه میشویم که این تصویر هرچد اندک اما دارای کنتراست بیشتر می باشد و لبه ها در آن بیشتر مشخص شده اند.

3-2- اعمال فیلتر های میانگین با ابعاد مختلف

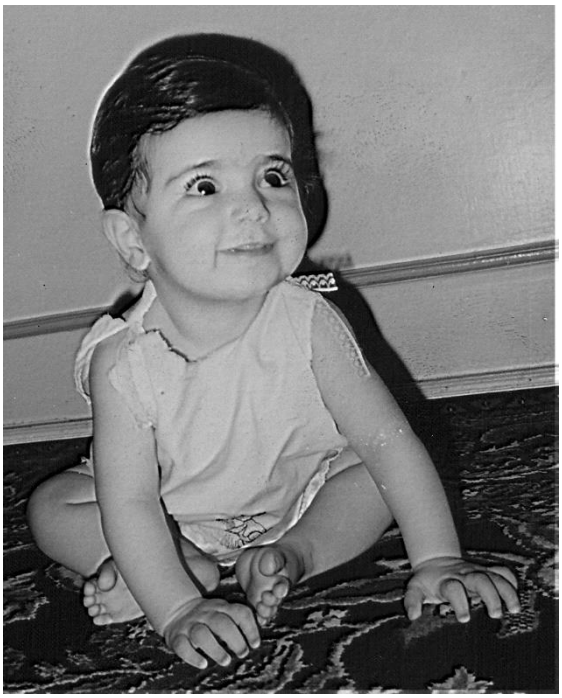
در این قسمت فیلتر میانگین جعبه ای را با ابعاد مختلف که در صورت تمرین گفته شده است (5*5و 7*7و9*9) به تصویر child اعمال می کنیم و نتایج زیر حاصل انجام این عملیات می باشد :



Box filter 5*5



Box filter 9*9

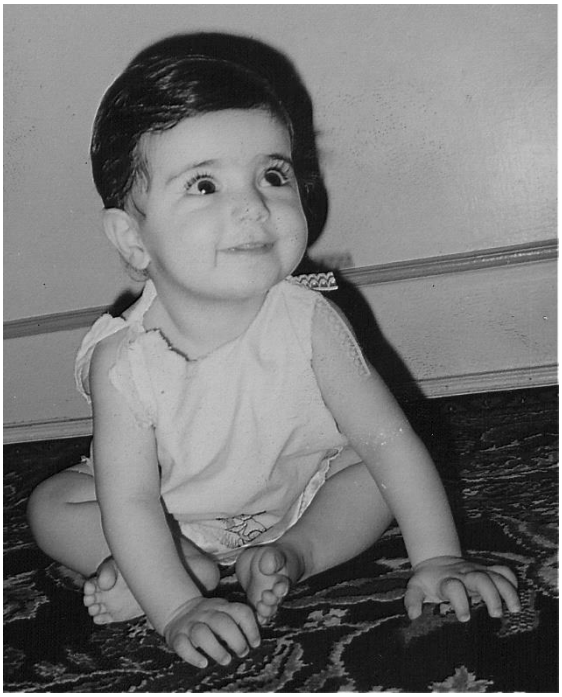


Box filter 7*7

نتایج بدست آمده نشان دهنده این موضوع هستند که هرچه ابعاد و سایز فیلتر جعبه ای ما بیشتر باشد وضوح تصویر به همان میزان افزایش می یابد، نویز های موجود در تصویر کاهش یافته و لبه ها در تصاویر بیشتر مشخص میشوند.

4-2-کلیشه لاپلاسین

برای انجام این بخش از تمرین ما فیلتر هموارسازی را که به تصویر اعمال کرده ایم را از نوع فیلتر میانه انتخاب می کنیم و آن را به تصویر اعمال می کنیم و نتیجه زیر را در آن مشاهده می کنیم:



از نتیجه بدست آمده می توان متوجه شد که تصویر به نسبت تصویر اصلی دارای وضوح بیشتر و نویز کمتر و لبه های مشخص تر می باشد.

حال در ادامه 1/16 کلیشه داده شده در صورت تمرین که

-1	-2	-1
-2	12	-2
-1	-2	-1

می باشد را تحت عنوان کلیشه لاپلاسین در نظر می گیریم و از رابطه $g(x,y) = f(x,y) + \nabla 2 f(x,y)$ برای تیز کردن تصویر استفاده می کنیم. این کلیشه از مجموع 1/16 کلیشه های زیر بدست می آید

0	0	0
-2	2	0
0	0	0

-1	0	0
0	1	0
0	0	0

0	-2	0
0	2	0
0	0	0

0	0	-1
0	1	0
0	0	0

0	0	0
0	2	-2
0	0	0

0	0	0
0	1	0
0	0	-1

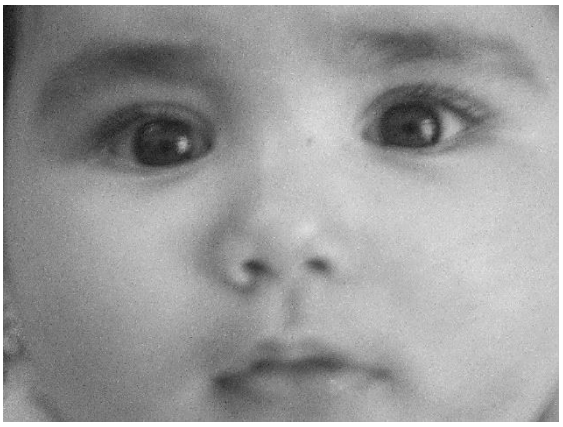
0	0	0
0	2	0
0	-2	0

0	0	0
0	1	0
-1	0	0

برای اینکه تصویر را به صورت وفقی تیز کنیم بایستی این کلیشه های داده شده را به هر نقطه از تصویر اعمال کنیم و با تعیین یک مقدار مشخص تحت عنوان آستانه گذاری که ما در این قسمت بزرگتر از صد هزار آن را در نظر گرفتیم حاصل تمام فیلتر هایی که بزرگتر از این مقدار هستند را با هم جمع کرده و نتیجه غایی را به تصویر اضافه کنیم که

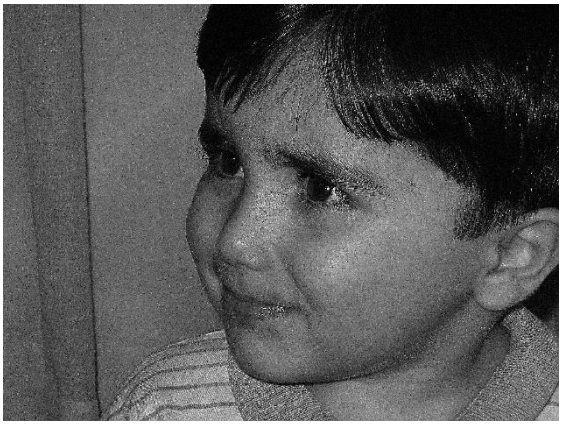
نتیجه تیز کردن تصویر به صورت وفقی را در زیر مشاهده می کنید :

face1



تصویر تیز شده به صورت وفقی face1

face2



تصویر تیز شده به صورت وفقی face2

در صورتی که بخواهیم تصویر را به صورت غیروفقی تیز کنیم باید بار دیگر 1/16 کلیشه لاپلاسین را به تصویر اصلی اعمال کنیم و نتیجه به دست آمده را به تصویر اصلی خود اضافه کنیم که حاصل انجام این عملیات تصاویر زیر می باشد:

تصویر تیز شده به صورت غیر وفقی face1

تصویر تیز شده به صورت غیر وفقی face2

همانطور که از نتایج بدست آمده پیدا است تیز کردن تصاویر به صورت وفقی نسبت به تیز کردن تصاویر به صورت غیر وفقی دارای نتایج بهتری است چراکه لبه ها در آن تیز تر شده و بهتر قابل تشخیص می باشد اما تصویر face1 با وجود اینکه یکنواخت تر است اما باز هم اعمال روش تیز کردن به صورت وفقی نسبت به تیز کردن به صورت غیروفقی بهتر بوده است و لبه ها به نسبت در آن بهتر مشخص شده و دیده میشوند.

.

مراجع

[1] https://www.geeksforgeeks.org

[2] https://stackoverflow.com/questions/53098631

[3] https://gisman.ir/image-processing-filter-2/

2-6- تصاویر کد های ضمیمه شده

```
[ ]  gussian_3x3 = np.array([[1,2,1],
                             [2,4,2],
                             [1,2,1]
                             ])/16


     def gussian_filter(image,kernel):

         n = kernel.shape[0]
         G = np.zeros((image.shape[0],image.shape[1]))

         for i in range(image.shape[0]):
           if image.shape[0] - i < n:
             break
           for j in range(image.shape[1]):
             if image.shape[1] - j < n:
               break
             G[i][j] = (kernel * image[i:i+n,j:j+n]).sum()

         return G
```

```
# 4
child = cv2.imread("child.jpg")
child = cv2.cvtColor(child,cv2.COLOR_BGR2GRAY)

def mean_filter(image,n):

  mean_filter = np.ones((n,n))/(n*n)
  G = np.zeros((image.shape[0],image.shape[1]))

  for i in range(image.shape[0]):
    if image.shape[0] - i < n:
      break
    for j in range(image.shape[1]):
      if image.shape[1] - j < n:
        break
      G[i][j] = (mean_filter * image[i:i+n,j:j+n]).sum()


  return G

child_unsharp = mean_filter(child,3)
unsharp_mask = child - child_unsharp
result1 = child + unsharp_mask
cv2_imshow(result1)

child_unsharp_gussian = gussian_filter(child,gussian_3x3)
unsharp_mask_gussian = child - child_unsharp_gussian
result2 = child + unsharp_mask_gussian
cv2_imshow(result2)
```

```
child_unsharp = mean_filter(child,5)
unsharp_mask = child - child_unsharp
result3 = child + unsharp_mask
cv2_imshow(result3)


child_unsharp = mean_filter(child,7)
unsharp_mask = child - child_unsharp
result5 = child + unsharp_mask
cv2_imshow(result5)


child_unsharp = mean_filter(child,9)
unsharp_mask = child - child_unsharp
result9 = child + unsharp_mask
cv2_imshow(result9)

def median_filter(image):


  G = np.zeros((image.shape[0],image.shape[1]))
  temp = np.zeros((3,3))
  for i in range(image.shape[0]):
    if image.shape[0] - i < 3:
      break
    for j in range(image.shape[1]):
      if image.shape[1] - j < 3:
        break
      temp = image[i:i+3,j:j+3]
      G[i][j] = np.median(temp)


  return G
```

```
# static gussian filter

kernel = np.array([[-1,-2, -1],[-2, 12, -2],[-1,-2,-1]])/16
derivative = gussian_filter(child,kernel)

res = derivative+child
cv2_imshow(res)

# dynamic gaussian filter

kernel1 = np.array([[0,0, 0],[-2, 2, 0],[0,0,0]])
derivative1 = gussian_filter(child,kernel1)/16
print(derivative1.sum())

kernel2 = np.array([[-1,0, 0],[0, 1, 0],[0,0,0]])
derivative2 = gussian_filter(child,kernel2)
print(derivative2.sum())


# lower than threshold
# kernel3 = np.array([[0,-2, 0],[0, 2, 0],[0,0,0]])
# derivative3 = gussian_filter(child,kernel3)
# print(derivative3.sum())

# lower than threshold
# kernel4 = np.array([[0,0, -1],[0, 1, 0],[0,0,0]])
# derivative4 = gussian_filter(child,kernel4)
# print(derivative4.sum())

# lower than threshold
# kernel5 = np.array([[0,0, 0],[0, 2, -2],[0,0,0]])
# derivative5 = gussian_filter(child,kernel5)
# print(derivative5.sum())
```

```
# lower than threshold
# kernel6 = np.array([[0,0, 0],[0, 1, 0],[0,0,-1]])
# derivative6 = gussian_filter(child,kernel6)
# print(derivative6.sum())

kernel7 = np.array([[0,0, 0],[0, 2, 0],[0,-2,0]])
derivative7 = gussian_filter(child,kernel7)
print(derivative7.sum())

kernel8 = np.array([[0,0, 0],[0, 1, 0],[-1,0,0]])
derivative8 = gussian_filter(child,kernel8)
print(derivative8.sum())

cv2_imshow(derivative1+derivative2+derivative7+derivative8+child)
```