تبدیل هندسی

منا رداد

اطلاعات گزارش

تاریخ:
1400/9/3

واژگان کلیدی:
درونیابی
انطباق
تبدیل هندسی
affine
registration

چکیده

در این گزارش می خواهیم به کمک تبدیلات هندسی مناسب و با بکار گرفتن درونیابی و استفاده از تابع affine سه تصویر lena,Barbara,girl را بر روی سه وجه مکعب بی اندازیم و همچنین این کار را با استفاده از درونیابی دو خطی نیز انجام دهیم. سپس دو تصویر map1,map2 را بر هم منطبق می کنیم و پارامتر های مدل را نیز ارائه می دهیم.

1-مقدمه

انطباق به فرایند منطبق کردن دو یا چند تصویر برهم گفته می شود و نیز مجموع مختلف داده ها را به یک سیستم مختصات تبدیل می کنیم که این داده ها شامل حسگر ها، زمان ها ، عکس های متعدد و ... می باشد. که از این داده ها میتوان در زمینه های مختلف از جمله اهداف نظامی و تشخیص خودکار آن ها، آنالیز تصاویر ماهواه ای، بررسی و تصویر برداری پزشکی و به طور کل در بینایی کامپیوتری میتوان استفاده کرد. در این بخش تبدیل های متفاوت تصاویر شامل rotate و scaling و distoration تصاویر را در نظر می گیریم.از این تغییرات اغلب تحت عنوان مراحل پیش پردازش در برنامه هایی همچون درک سند استفاده می شود، در اینجا ممکن است تصویر اسکن شده اشتباها تراز شود. در تبدیل های هندسی دو مرحله اساسی وجود دارد : در مرحله اول از لحاظ فضایی پیکسل های موجود در تصاویر را به صورت فیزیکی دوباره می چینیم و در مرحله دوم درون یابی سطح خاکستری، که gray scale ها را اختصاص به تصویر تبدیل شده میدهیم.

1-2-قرار دادن تصاویر بر سه وجه مکعب

در این قسمت به کمک تبدیلات هندسی و با استفاده از درون یابی نزدیک ترین همسایه عکس های داده شده که شامل تصاویر lena,Barbara,girl را روی وجه های مکعب قرار می دهیم. برای انجام این کار از توابعی استفاده می کنیم جهت بزرگ کردن و کوچک کردن تصاویر ، زاویه دادن و چرخش آن ها. این توابع به این صورت هستند که در تابع shear ابتدا تصویر را نسبت به اندازه اصلی آن بزرگتر می کنیم این کار به این دلیل است که هنگامی که تصویر را می

چرخانیم تصویر از دامنه تعیین شده ردیف ها و سر های آن خارج نشود،سپس به کمک تابع affine تغییرات را بسته به شرایط حول محور x یا y انجام می دهیم. پارامتر های ورودی این تابع شامل عکس موردنظر، درجه shear و محور مورد نظر می باشد و به عنوان خروجی ماتریسی از تصویر تغییر یافته برمیگرداند.در تابع دیگر عرض یا طول تصویر را تغییر سایز داده و کوچک یا بزرگ می کنیم تا هم اندازه با وجه مورد نظر در مکعب شود که این تابع به عنوان پارامتر ورودی تصویر موردنظر و نسبت موردنظر برای اعمال بر طول و عرض را میگیرد.

برای انجام این کار به مقدار حدودی تصویر زمانی که میچرخد تا در قسمت مورد نظر در مکعب قرار بگیرد، مختصات مرکز هر وجه مکعب را در کنار تصویر موردنظر و تصویر مکعب به تابع merge می دهیم و هنگامی که عکس در مکان نسبی قرار گرفت دوباره به کمک توابع توضیح داده شده عکس اولیه را چرخانده و تغییرات طول و عرض را اعمال می کنیم تا بدین ترتیب تصویر در مکان مناسب خود قرار گیرد. نتیجه این امر قرار گرفتن تصاویر در وجوه مورد نظر در تصویر مکعب می باشد که در زیر مشاهده می کنید:

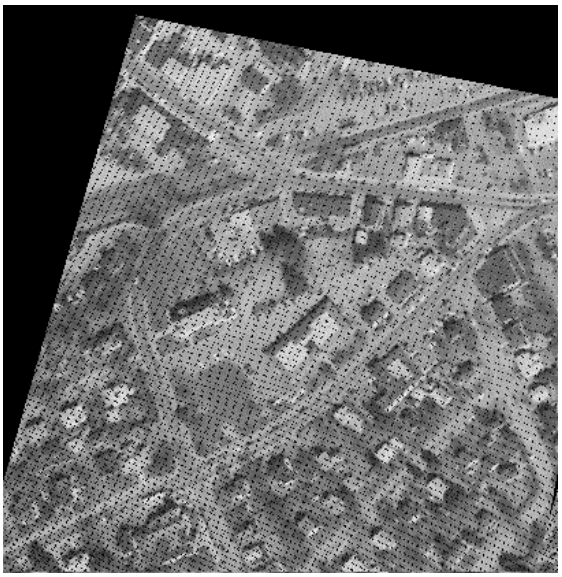
همانطور که مشاهده میشود با تنظیم آرگومان های هر یک از توابع مشاهده میشود که تصویر در محل مورد نظر در وجوه مکعب به خوبی قرار گرفته است.

2-2-انطباق تصاویر

در این قسمت از ما خواسته شده است که با استفاده از عملیات انطباق تصویر map1 را به map2 منطبق کنیم. انطباق تصویر یا registeration به فرایندی گفته میشود دو یا چند تصویر بر هم منطبق شوند. برای انجام این کار ما 4 نقطه از تصویر اول را در نظر میگیریم و مختصات نقاط متناظر آن در تصویر دیگر را نیز بدست آورده و معادله 8 مجهولی زیر را حل کرده و ضرائب را بدست می آوریم

$$\begin{cases} x = c_1 + c_2 v + c_3 w + c_4 vw \\ y = c_5 + c_6 v + c_7 w + c_8 vw \end{cases}$$

حال با بدست آوردن ضرائب هر پیکسل از map2 را به هر پیکسل از map1 منطبق می کنیم که نتایج آن به صورت زیر می باشد.



تصویر بالا نشان دهنده ی نتیجه ی انطباق تصاویر نقشه ها می باشد که اگر ما نقشه منطبق شده را با نقشه قبل از

انطباق مقایسه کنیم قادر خواهیم بود پیکسل به پیکسل تغییرات در این 2 عکس را پیدا کنیم که یکی از کاربرد های مهم registration یا انطباق نیز همین است.

مراجع

[1] https://gisman.ir/

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

[3] https://stackoverflow.com

4-2-کد های ضمیمه شده

```
def shear(image,degree,direction):
  res = np.zeros((2*image.shape[0],2*image.shape[1],3))

  for i in range(image.shape[0]):
    for j in range(image.shape[1]):
      v = (i - image.shape[1]/2)
      w = (j - image.shape[0]/2)
      if(direction=="y"):
        x = v
        y = w + v*degree
      elif(direction=="x"):
        x = v + w*degree
        y = w
      res[int(y + res.shape[0]/2),int(x + res.shape[1]/2)]=image[j,i]
  return res


def scale(image,qX,qY):

  res = np.zeros((image.shape[0],image.shape[1],3))
  for i in range(image.shape[0]):
    for j in range(image.shape[1]):
      v = i - image.shape[1]/2
      w = j - image.shape[0]/2
      x = v*qX
      y = w*qY
      res[int(y + res.shape[0]/2), int( x + res.shape[1]/2)] = image[j,i]
  return res

def merge(image,x,y,cube):
  for i in range(image.shape[0]):
    for j in range(image.shape[1]):
      newX = int(x-(image.shape[0]/2)+i)
      newY = int(y-(image.shape[1]/2)+j)
      if(newX>=0 and newX<cube.shape[0] and newY>=0 and newY<cube.shape[1]):
        if(np.all(image[i,j])):
         cube[newX,newY]= image[i,j]
  # cv2_imshow(cube)
  return cube
```

```
cube=cv2.imread("Cube.png")
# cube = cv2.cvtColor(cube,cv2.COLOR_BGR2GRAY)
# 1.1
lena = cv2.imread("lena.bmp")
# lena = cv2.cvtColor(lena,cv2.COLOR_BGR2GRAY)

girl = cv2.imread("girl.bmp")
# girl= cv2.cvtColor(girl,cv2.COLOR_BGR2GRAY)
barbara = cv2.imread("barbara.bmp")


sheared_barbara = shear(barbara,0.43,"x")
scaled_barbara = scale(sheared_barbara,.99,0.45)
cube = merge(scaled_barbara,232,472,cube)


#  scaling lena:
scaling = [[0.9765625,0,0],[0,0.9765625,0],[0,0,1]]
lena_scaled = np.zeros((512,512,3))
barbara_scaled = np.zeros((1000,1000,3))

for i in range(501):
  for j in range(501):
    for k in range(3):
      lena_scaled[i][j][k]= lena[int(i*0.9765625)][int(j*0.9765625)][k]

for i in range(500):
  for j in range(500):
    for k in range(3):
      cube[346+i,333+j,k]=lena_scaled[i][j][k]


image = shear(girl,0.45,"y")
image2 = scale(image,0.43,0.99)
result = merge(image2,481,222,cube)
```

```
cv2_imshow(result)

im = Image.open("Map1.gif")
im.seek(0)
im.save("map1.png")

im = Image.open("Map2.gif")
im.seek(0)
im.save("map2.png")


map1 = cv2.imread("map1.png")
map1 = cv2.cvtColor(map1,cv2.COLOR_BGR2GRAY)
map2 = cv2.imread("map2.png")
map2 = cv2.cvtColor(map2,cv2.COLOR_BGR2GRAY)
print(type(map1))
print(map1.shape,map2.shape)

def line_interpolation_map(v,w):
  y = 0.97153*v-0.29462*w+0.00028696*v*w+97.625
  if y > 405:
    y = 405
  x = 0.1915*v+1.0358*w-0.0001089*w*v+7.9987
  if x > 414:
    x = 414
  return x,y

map12 = np.zeros((415,406))
print(map12.shape)

for i in range(394):
  for j in range(369):
    x,y = line_interpolation_map(j,i)
    map12[int(x)][int(y)] = map2[i][j]

cv2_imshow(map12)
```