بسمه تعالی

تمرین سری دوم درس بینایی کامپیوتر

چگونه تبدیلات مولفه های اصلی(Principal Components Transforms) باعث میشوند شیء نسبت به تبدیلات چرخش، جابجایی و تغییر اندازه مستقل شوند؟

پوریا محمدی نسب

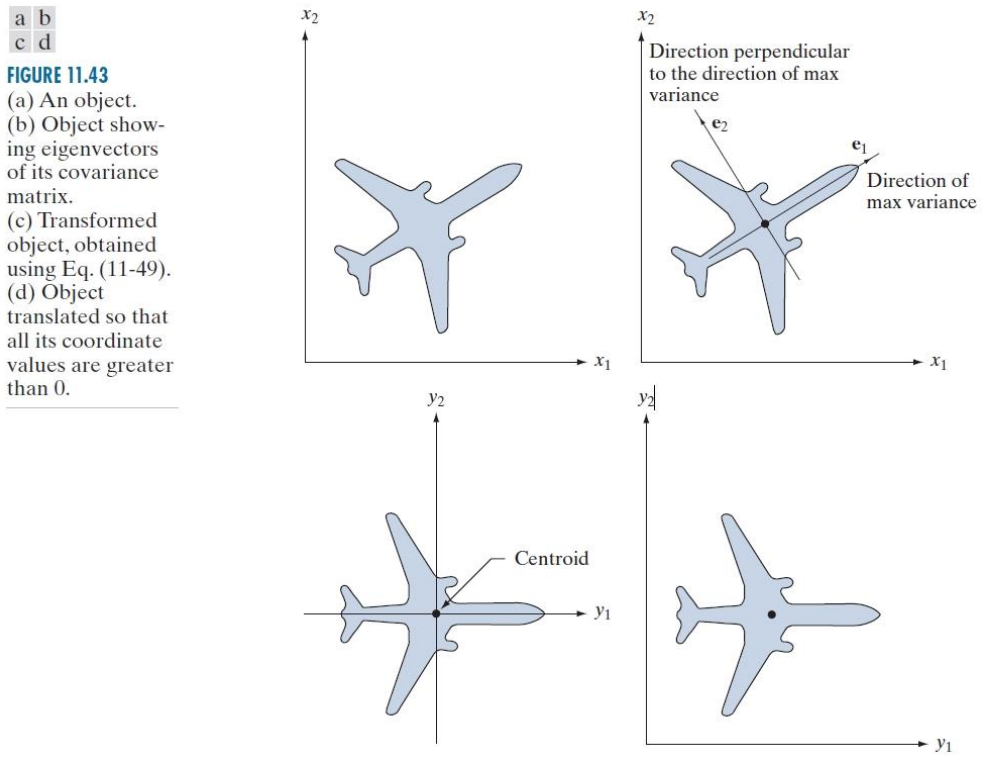
۴۰۰۷۲۲۱۳۸

هدف از معرفی توصیفگرهای ویژگی ها[1] این است که این توصیفات تا حد ممکن نسبت به تغییر اندازه[2]، جابجایی[3] و چرخش[4] مستقل[5] باشند به این معنی که اگر تبدیلی روی شیء و یا تصویر اعمال شد مقدار توصیفگر تغییر چندانی نکند. مولفه های اصلی[6] یک راه راحت و قابل اعمال فراهم میکنند تا تغییرات این سه نوع تبدیل را برای مرزها و نواحی شکل نرمال کنیم. برای مثال فرض میکنیم شکلی داریم که اندازه، مکان و میزان چرخش آن هر مقداری میتواند باشد و دلخواه است. نقاط روی مرز و یا درون ناحیه را میتوانیم مانند یک وکتور ۲ بعدی به صورت $(x_1, x_2)^T$ در نظر بگیریم. پس میتوان از این وکتور ۲ بعدی برای محاسبه ماتریس کواریانس(C_x) و وکتور میانگین (m_x) استفاده کنیم. در این حالت اولین بردار ویژه[7] در جهتی است که نقاط درون شکل یا روی مرز بیشترین تغییرات(واریانس) را دارند و بردار ویژه دوم عمود بر بردار اول است. حال با توجه به رابطه

$$y = A(x - m_x)$$

دو کار اساسی انجام میدهد. ۱) مختصات مرکز نقاط تبدیل شده را محاسبه میکند. ۲) شکل را آنقدر چرخش میدهد تا بردار ویژه اول در راستای y_1 و بردار ویژه دوم در راستای y_2 قرار گیرد. بدین ترتیب ناحیه و یا شیء مورد نظر نسبت به تبدیل چرخش مستقل میشود. در مرحله بعدی شکلی که هم راستای y_1 و y_2 شده است را با انجام یک انتقال به قسمتی از محورهای مختصات میبریم تا تمام مقادیر نقاط مثبت باشند و به این صورت شیء مورد نظر نسبت به تبدیل انتقال نیز مستقل میشود. در [1] ذکر شده است که مولفه های اصلی نمیتوانند نسبت به تغییر اندازه ی شیء مستقل باشند زیرا تغییرات اندازه منجر به تغییر در نقاط ناحیه میشود و به همین ترتیب مقادیر و نسبت های بردارهای ویژه تغییر میکنند.

شکل زیر شکلی از کتاب است که توضیحات بالا را به صورت مصور بیان میکند.



1) Rencher, A.C. (2002). Principal Component Analysis. In Methods of Multivariate Analysis, A.C. Rencher (Ed.). https://doi.org/10.1002/0471271357.ch12

[1] Feature descriptors
[2] Scale
[3] Translation
[4] Rotation
[5] Invariant
[6] Principal components
[7] eigenvector

Principal components transform

February 27, 2022

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
from sklearn.decomposition import PCA
```

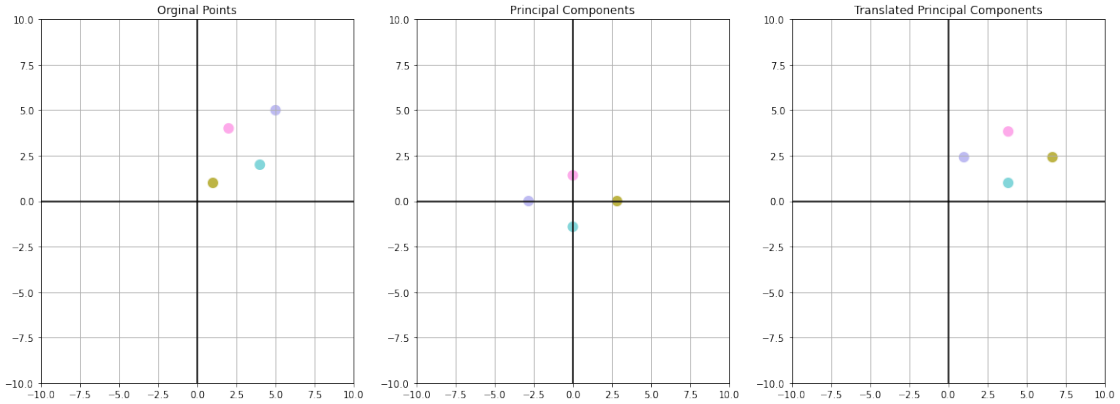
[ ]:
```
def random_colors(n):
    colors = []
    for i in range(n):
        colors.append("#%06x" % random.randint(0, 0xFFFFFF))
    return colors
```

[ ]:
```
def apply_PCA(Points):

    # set a unique color for each point
    colors = random_colors(len(Points))

    # PCA model
    pca = PCA(n_components = 2)
    principalComponents = pca.fit_transform(Points)

    # translation
    principalComponents_translated = np.zeros_like(principalComponents)
    min_x = principalComponents[:,0].min()
    principalComponents_translated[:,0] = principalComponents[:,0] - min_x + 1

    min_y = principalComponents[:,1].min()
    principalComponents_translated[:,1] = principalComponents[:,1] - min_y + 1


    # ploting

    fig, axs = plt.subplots(1, 3,figsize=(20,7))

    axs[0].scatter(Points[:,0],Points[:,1],marker='o',linewidths=5,color=colors)
    axs[0].grid()
    axs[0].axis(xmin=-10,xmax=10,ymin=-10, ymax=10)
    axs[0].hlines(y=0,xmin=-10,xmax=10,colors='k')
    axs[0].vlines(x=0,ymin=-10,ymax=10,colors='k')
    axs[0].set_title("Orginal Points")

    axs[1].scatter(principalComponents[:,0],
    principalComponents[:,1],marker='o',linewidths=5,color=colors)
    axs[1].grid()
```

```
    axs[1].axis(xmin=-10,xmax=10,ymin=-10, ymax=10)
    axs[1].hlines(y=0,xmin=-10,xmax=10,colors='k')
    axs[1].vlines(x=0,ymin=-10,ymax=10,colors='k')
    axs[1].set_title("Principal Components")

    axs[2].scatter(principalComponents_translated[:,0],
    principalComponents_translated[:,1],marker='o',linewidths=5,color=colors)
    axs[2].grid()
    axs[2].axis(xmin=-10,xmax=10,ymin=-10, ymax=10)
    axs[2].hlines(y=0,xmin=-10,xmax=10,colors='k')
    axs[2].vlines(x=0,ymin=-10,ymax=10,colors='k')
    axs[2].set_title("Translated Principal Components")
```

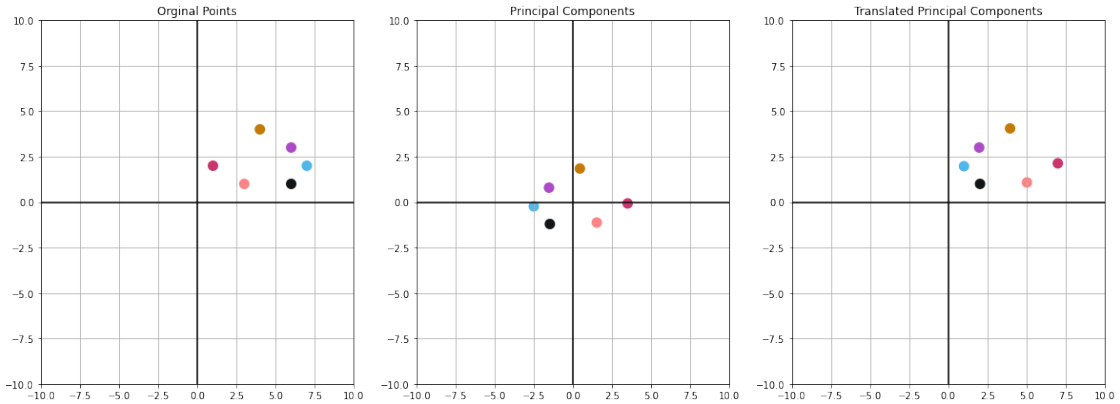
[ ]:
```
Points1 = np.array([[1,1],[2,4],[4,2],[5,5]])
```

[ ]:
```
apply_PCA(Points1)
```



[ ]:
```
Points2 = np.array([[1,2],[3,1],[4,4],[6,1],[6,3],[7,2]])
```

[ ]:
```
apply_PCA(Points2)
```



[ ]:
```
Points3 = np.random.randint(-4,4,(6,2))
```

[ ]:
```
apply_PCA(Points3)
```