فایل helper.py

همراه تمامی کدها، یک فایل helper.py قرار دارد که به منظور خوانایی کدهاست. تابع هایی که در فایل اصلی کدها هستند (Qi.py) در فایل helper پیاده سازی میشوند تا کد اصلی کوتاه تر و خواناتر باشد. در توضیحات سوال ها هم کدهای اصلی هم تابع های پیاده سازی شده را تا جای ممکن توضیح میدهم.

در این تمرین هدف به کار بردن k-means و mean-shift برروی نقاط داده شده بود. نقاط به گونه ای انتخاب شده اند که این دو روش در حالت عادی نمیتوانند به خوبی کلاستر کنند که این به نحوه کارکرد این دو برمیگردد .

ابتدا نقاط را با کمک دستورات پایتون از فایل خوانده ام و در دو آرایه x ها و y ها را ذخیره کردم. همینطور بیشترین و کمترین x و بیشترین و کمترین y را هم پیدا کرده ام. نقاط در فضای دو بعدی معمولی هستند. ولی برای نمایش به صورت عکس باید مختصات آن ها را بدین صورت تغییر داد:

```
for i in range(len(x)):
    x[i] += -min_x

for j in range(len(y)):
    y[j] += -min_y
```

بدین ترتیب نقاط را به صورت یک عکس نشان میدهیم.

تابع get_array_points مقادیر xوy نقطه ها راگرفته و از روی آن ها یک آرایه numpy میسازد که این آرایه را به تابع های آماده k-means و mean-shift پاس میدهیم.

ابتدا k-means را بر بروی نقاط اعمال میکنیم و خروجی می گیریم. برای گرفتن خروجی چون نقاط هر کدام یک پیکسل هستند و دیدنشان کمی مشکل بود، هر نقطه را ۳ برابر کردم (به شعاع ۳ رو هم رنگی کردم) تا بهتر دیده بشه.

سپس mean-shift را اعمال کردم و همچون قسمت قبل نقطه ها رو نشون دادم.

دلیل اینکه هر بار که k-means را اجرا میکنیم نتیجه ای کمی متفاوت تر از قبلی به ما میدهد نحوه اجرای این الگوریتم است. در وهله اول طبق تعداد کلاسترهایی که براش مشخص کردیم(k) نقطه رندوم انتخاب میکند و سپس سعی میکند نقاطی که با این نقطه در یک کلاستر قرار دارند را بیابد و ... . این انتخاب اولیه رندوم میتواند باعث نتایج کمی متفاوت شود.

در انتها سوال خواسته شده که نقاط را طوری تغییر دهیم که k-means بتواند آن را به درستی کلاستر بندی کند. خب به وضح دو کلاستر ، یکی نقطه های وسطی است که فاصله شان از مرکز تصویر کم است. کلاستر دوم، نقاطی است که فاصله بیشتری از مرکز دارند. بدین ترتیب اگر نقاط را به مختصات قطبی شان ببریم مشکل حل خواهد شد:

```
r_array = []
theta_array = []
for i in range(1, len(file_lines)):
    xx, yy = file_lines[i].split(" ")
    point_r = math.floor(math.sqrt(float(xx) ** 2 + float(yy) ** 2) * 100)
    r_array.append(point_r)
    point_theta = math.floor(math.atan2(float(yy), float(xx)) * 100)
    theta_array.append(point_theta)
```

که نمایش این نقاط در تصویر im04-r-theta آمده است.

سپس k-means را روی این نقاط همچون بخش اول اجرا میکنیم و نتیجه را ذخیره میکنیم.