Seam Carving Project

Run project

run with this command format:

```
python main.py input_file dx dy -o output_file
```

- input_file : path of input image
- dx : deacresing pixel number in width direction
- dy : deacresing pixel number in height direction
- output: path of output image and gif

like this:

```
python main.py data/In_1.jpg 10 10 -o data/Out_1.jpg
```

Examples

```
>>> python main.py input1.py 50 50 -o data/output_of_input1.jpg
```

input example

example_input

output example

example_output

output gif

example_gif

Energy algorithm

در این پروژه من از دو متد محاسبه انرژی متفاوت استفاده کردم.

اولی همان متد داخل داک پروژه بود که طبق محاسبات زمانی برای هر عکس ۴۰۰ در ۴۰۰ پیکسل ۴ ثانیه طول میکشید.علت طولانی شدن این الگوریتم تعداد محاسبات زیاد ریاضی شامل توان دو و جذر هست.که این محاسبات باید برا هر پیکسل انجام شود و در واقع باید به تعداد طول ضرب در عرض عکس اینکار تکرار شود.

دومی یک متد محاسبه ی انرژی از گیت هاب هست که با استفاده از دی پی و با سرعت خیلی بالا تری محاسبه را در کمتر از یک دهم ثانیه برای عکس ۴۰۰ در ۴۰۰ انجام میدهد.

Find seam algorithm

توضیح الگوریتم را از اینجا شروع میکنم.

برای هر پیکسل از عکس، این شرایط برقرار هست.کوتاهترین مسیری که میتواند وجود داشته باشد و این پیکسل از عبور کند به این شکل خواهد بود که باید از یکی از سه پیکسل بالایی به این پیکسل وارد شد.یعنی مثلا از آ به جی

```
A B C
 \|/
  G
```

روند حرکت از بالا به پایین هست.همسایگی هاهم حتمن به همین شکل هستند.پس میتوان از سطر اول شروع کرد و به پایین آمد و همیشه کمترین مسیر را برای هر نود انتخاب کرد.ما در واقع برای هر نود مشخص میکنیم که از کدام نود بالاسری باید به این نود رسید تا کمترین مقدار باشد.

ارزش تمام نود های سطر اول همان مقدار انرژی آنهاست.بعد از بالا شروع میکنیم و به سطر های دوم و سوم و پایین تر میرویم.در هر سطر برای هر پیکسل با استفاده از برنامه ریزی پویا و اطلاعاتی که برای نود های بالایی بدست اورده ایم تصمیم میگیریم که برای اینکه با کمترین مقدار از این پیکسل عبور کنیم باید از کدام پیکسل بالاسری به این پیکسل وارد شویم.

پیکسل بالا سری ای که مقدار کمتری از بقیه دارد به عنوان جد این پیکسل در حال بررسی انتخاب میشود.

یعنی فرمول ان به این شکل میشود

```
                          A
         <------------------------------------>
C[x, y] = Min(C[x-1, y-1], C[x-1, y], C[x-1, y+1]) + energy[x, y]

ancestor[x, y] = A

A is node with minimum value
```

حالا این کار را از بالا به پایین برای همه پیکسل ها انجام میدهیم

O(Height * Width)

و سپس از سطر آخر آن پیکسلی که مقدار محاسبه تا اینجا برای آن از همه کمتر هست را انتخاب میکنیم

O(Width)

و سپس با استفاده از آرایه پدر ها مسیر را از پایین به بالا پیدا میکنیم

O(Height)

پس اردر زمانی این الگوریتم

```
Time Complexity = O(Height * Width)
```

میباشد

- در طول اجرای الگوریتم یک آرایه ی دو بعدی برای نگهداری مقادیر انرژی ها نیاز داریم
- یک آرایه دو بعدی برای ذخیره کمترین مقداری که میتوان از بالا به پایین حرکت کرد و به این نود رسید
- و یک آرایه دو بعدی برای نگهداری پدر های هر پیکسل

```
Space Complexity = 3 * O(Height * Width) = O(Height * Width)
```