دانشکده مهندسی کامپیوتر

تحلیل و طراحی الگوریتم

گزارش پروژه

علی صداقی

۹۷۵۲۱۳۷۸

موضوع

Seam Carving

۲ حالت مختلف Quality و Time:

در این برنامه دو حالت برای حذف کردن درزها در نظر گرفته‌شده‌است که در ادامه به بررسی هر یک می-پردازیم.

۱. حالت Quality:

در این حالت هر بار که یک Seam حذف می‌شود برنامه براساس عکس جدید آرایه Energy را دوباره محاسبه می‌کند و براساس آن درز بعدی یافت می‌شود. در این حالت برای هر درز عکس مربوط به آن نیز ذخیره می‌شود. نتیجه این کار بهبود کیفیت و افزایش زمان اجرای برنامه می‌باشد.

۲. حالت Time:

در این حالت تنها یک بار در ابتدای برنامه آرایه Energy محاسبه می‌شود. و بر اساس همان آرایه Seamها یافت می‌شوند. همچنین تنها عکس اولین Seam یافت شده ذخیره می‌شود. نتیجه این کار کاهش زمان اجرای برنامه است.

در ادامه به صورت کمی به مقایسه این دو حالت خواهیم پرداخت.

نحوه کار با برنامه:

ابتدا به مسیر زیر رفته:

```
Seam-Carving/src
```

سپس با دستور زیر برنامه را اجرا می‌کنیم.

```
python3 SeamCarving.py arg1 arg2 arg3 arg4 arg5
arg1: relative path of image
arg2: vertical pixels to remove
arg3: horizontal pixels to remove
arg4: mode: “quality” or “time
arg5: folder name to save the result
```

نکته: نتیجه در مسیر زیر ذخیره می‌شود.

```
Seam-Carving/Result/arg5
```

نکته: عکس های ورودی در مسیر زیر قرار گرفته‌اند.

```
Seam-Carving/Inputs
```

پس arg1 به صورت زیر خواهد بود:

```
../Inputs/In_0.png
```

مثال از یک دستور:

```
python3 SeamCarving.py ../Inputs/In_0.png 50 50 quality 0_q
```

توضیح بخش های مختلف Code:

- از کتاب‌خانه Pillow برای باز کردن عکس و ذخیره سازی عکس استفاده شده.
- از کتاب‌خانه NumPy برای نگهداری پیکسل های عکس و همچنین Vectorization استفاده شده.

کلاس MyImage با در ابتدا با ورودی گرفتن مسیر عکس، حالت پاک کردن، مسیر خروجی شروع به کار می‌کند.

فیلد های کلاس

- realImage

یک آرایه NumPy با شکل (height, width, 3) که مسئول نگهداری پیکسل های عکس می‌باشد. هرگاه که یک درز پاک شد این فیلد آپدیت می‌شود.

- height و width

طول عمودی و افقی عکس که پس از حذف هر درز آپدیت می‌شوند.

- colors

مشخص کننده تعداد رنگ های هر پیکسل برای عکس های rgb برابر ۳ و برای rgba برابر ۴

- energyMatrix

آرایه دو بعدی که مقدار انرژی هر پیکسل را در خود نگه می‌دارد. بسته به نوع mode برنامه این آرایه ممکن هست فقط یک بار محاسبه شود یا اینکه پس از حذف هر درز آپدیت شود.

- seamImage

یک آرایه NumPy با شکل (height, width, 3) که مسئول نگهداری پیکسل های عکس مربوط به انرژی می باشد. نحوه ساختن این آرایه از آرایه energyMatrix در ادامه آمده است. این آرایه نیز بسته به mode می‌تواند تنها یک بار ساخته شود یا اینکه پس از حذف یک درز دوباره ساخته شود.

- seamCoordinates

یک آرایه که اعضای آن یک تاپل (x, y) می‌باشند که مشخص کننده مختصات نقاط درز هستند.

- mark

که آرایه دو بعدی از مقادیر Boolean که اگر یک المنت برابر False بود بیانگر این است که آن پیکسل عضو درز است.

- mode

مشخص کننده حالت برنامه که میتواند به صورت time یا quality باشد.

- folderName

نام پوشه ای که میخواهیم خروجی در آن ذخیره شود.

متد های کلاس

- getPixelRGB

با گرفتن مختصات یک پیکسل، یک تاپل ۳ تایی از رنگ قرمز، سبز و آبی آن پیکسل را بر می‌گرداند.

- getPixelEnergy

با گرفتن مختصات یک پیکسل، انرژی آن پیکسل را محاسبه می‌کند. نحوه انجام این کار با الگوریتم Dual Gradient می‌باشد. برای افزایش سرعت برنامه پیکسل های همسایه درون آن کش می‌شوند تا از دسترسی مداوم و مستقیم به آرایه اصلی عکس جلوگیری کنیم.

برای پیکسل هایی که روی مرز هستند دو روش ممکن بود: ۱- Zero Pad (پیکسل های همسایه که وجود ندارند را 0 در نظر بگیریم) ۲- Max Border (پیکسل های روی مرز عکس را بیشترین انرژی ممکن بدهیم، درون عکس انرژی یک بوردر سفید دور عکس ایجاد می‌شد)

در اینجا از روش Zero Pad استفاده شده است زیرا کیفیت خروجی بهتر بود.

Max Border

Zero Pad

- updateImageDimension

هر گاه که یک درز پاک شد این تابع فیلد های height و width را آپدیت می‌کند.

- updateEnergyMatrix

فیلد energyMatrix را آپدیت می‌کند. نحوه انجام آن به صورت دو for یکی بر روی عرض و دیگری بر روی طول عکس است و به ازای هر پیکسل تابع getPixelEnergy صدا زده می‌شود.

این تابع به صورت Vectorization هم پیاده سازی شد اما خروجی تصویر انرژی به هیچ عنوان قابل قبول نبود به همین دلیل ترجیح داده شد به صورت For Loop پیاده سازی شود.

```
#calculate squared difference ((x-1, y) - (x+1, y))^2 for each R, G and B pixel
deltaX2 = np.square(np.roll(arr, -1, axis = 0) - np.roll(arr, 1, axis = 0))
#same for y axis
deltaY2 = np.square(np.roll(arr, -1, axis = 1) - np.roll(arr, 1, axis = 1))
#add R, G and B values for each pixel, then add x- and y-shifted values
de_gradient = np.sum(deltaX2, axis = 2) + np.sum(deltaY2, axis = 2)
```

پیاده سازی با Vectorization

- updateVerticalSeam

با استفاده از Dynamic Programming یک درز عمودی با کمینه انرژی محاسبه می‌شود.

M یک کپی از ماتریس Energy می‌باشد که هر خانه آن کمینه انرژی ممکن برای رسیدن به آن خانه از بالا به پایین است. نحوه محاسبه M به صورت زیر است:

$$M(x,y) = Energy(x,y) + \min\left(M(x-1,y-1), M(x-1,y), M(x-1,y+1)\right)$$

هنگامی که به پایین عکس رسیدیم شروع به Backtrack می‌کنیم تا کمینه مسیر رسیدن به هر نقطه بدست آید.

پس از اینکه عملیات بالا کامل شد شروع به ساختن دو متغیر mark و seamCoordinates می‌کنیم که در بخش معرفی فیلد ها و تابع زیر کاربرد هرکدام شرح داده شده است.

- updateSeamImage

ابتدا بر اساس ماتریس انرژی و به وسیله یک نگاشتی خطی مقدار انرژی هر پیکسل را به ۳ تاپل RGB تبدیل می‌کند نحوه انجام آن به صورت زیر است.

ابتدا مقدار maxEnergy به دست می‌آید که به دو استراتژی ممکن است که در اینجا از روش ۱ استفاده شده است :

۱. ماکسیموم انرژی موجود در فیلد energyMatrix: تولید عکس پر انرژی تر

۲. ماکسیموم انرژی که فرمول Dual Gradient می‌تواند تولید کند: تولید عکس کم انرژی تر

سپس یک نگاشت خطی بین مقدار انرژی هر پیکسل و عدد ۰ تا ۲۵۵ انجام می‌دهیم.

$$scale = \frac{255}{maxEnergy}$$

$$pixelRGBValue = scale * \ energyPixel$$

اگر پیکسل مورد نظر در آرایه seamCoordinates بود به جای روابط بالا مقدار (255, 0, 0) که بیانگر رنگ قرمز هست تولید می‌شود.
در نهایت به ازای تمامی پیکسل ها عملیات بالا صورت میگیرد و آرایه seamImage آپدیت می‌شود.

- updateAll

یک تابع که تمامی توابع update را به ترتیب زیر فراخوانی می‌کند.

```
updateImageDimension
updateEnergyMatrix
updateVerticalSeam
updateSeamImage
```

کاربرد آن در حالت quality است که در هر حذف درز تمامی فیلد ها دوباره آپدیت می‌شوند.

- deleteVerticalSeam

با استفاده از فیلد mark و تابع reshape کتاب‌خانه NumPy نقاطی که درون mark به صورت False می‌باشند حذف می‌شوند و فیلد realImage آپدیت می‌شود.
اگر در حالت time بودیم آرایه energyMatrix نیز باید آپدیت شود. زیرا در هر مرحله این آرایه دوباره از نوع ساخته نمی‌شود.

- saveSeamImage

براساس فیلد seamImage تصویر انرژی و درز را ذخیره می‌کند.

- saveRealImage

بر اساس فیلد realImage تصویر نهایی را ذخیره می‌کند.

- removeColumns

از این تابع در حالت quality استفاده می‌شود. به ازای هر ستونی که باید حذف شود ابتدا همه چیز آپدیت می‌شود. سپس تصویر درز و انرژی ذخیره می‌شود. درز حذف می‌شود.

- removeRows

از این تابع در حالت quality استفاده می‌شود. می‌توان حذف ردیف را مشابه حذف ستون در نظر گرفت، به شرطی که عکس را ۹۰ درجه دوران دهیم. پس ابتدا عکس را ۹۰ درجه می‌چرخانیم. سپس برای به ازای هر ردیف تمامی کار های متد بالا را انجام می‌دهیم اما قبل از ذخیره کردن عکس انرژی و شکاف باید دوران معکوس کنیم. در انتهای کار نیز عکس اصلی را دوران معکوس می‌کنیم.

- remove

یک تابع Wrapper که در حالت quality دو تابع بالا را صدا می‌زند و در حالت time مشابه بالا رفتار می‌کند با این تفاوت که دیگر در هر مرحله انرژی ها آپدیت نمی‌شوند و تصویر انرژی درز تنها یک بار ذخیره می‌شود.

بررسی عملکرد برنامه:

برای سه تصویر اول موجود در پوشه `Inputs` هر دو حالت `time` و `quality` و حذف ۵۰ پیکسل افقی و ۵۰ پیکسل عمودی اجرا شده است و نتایج به صورت زیر است:

Image	Width	Height	Time Mode (sec)	Quality Mode (sec)
In_0.png	612	345	145	610
In_1.jpg	985	771	584	3036
In_2.jpg	780	488	287	1320
In_3.jpg	2000	1328	2007	-
In_4.jpg	1125	610	511	2713
In_5.jpg	1379	1024	1112	-
In_6.jpg	1200	800	757	4251
In_7.jpg	1280	960	1009	-
In_8.jpg	770	563	315	1499
In_9.jpg	960	720	546	2690
In_10.jpg	600	403	174	733
In_11.jpg	945	1200	920	-
In_12.jpg	310	163	33	113

کیفیت خروجی تصاویر در پوشه `Result` موجود است. برای سایر تصاویر موجود در این پوشه تنها حالت `Time` اجرا شده است.

همان طور که در نمودار بالا مشاهده می‌شود در حالت Time Mode برای حذف تعداد ثابتی پیکسل ($c = 50$ و $r = 50$) زمان اجرا با حاصل $Height * Width$ رابطه خطی دارد. اما در حالت Quality Mode رابطه سهمی دارد.

➢ محاسبه پیچیدگی حافظه ای برنامه:

نگهداری عکس ها به صورت یک آرایه NumPy از مرتبه زمانی زیر می‌باشد:

$$0(Height * Width * 3)$$

آرایه mark، energyMatrix، M و backtrack نیز به صورت زیر می‌باشند:

$$0(Height * Width)$$

پس مرتبه حافظه ای برنامه به صورت زیر است:

$$0(Height * Width)$$

➢ محاسبه پیچیدگی زمانی برنامه:

با فرض اینکه width تصویر برابر w و height آن برابر h باشد همچنین تعداد درز عمودی برای دیلیت c و تعداد درز افقی برای دیلیت r باشد برای هر تابع پیچیدگی زمانی زیر را داریم.

Function	Time Complexity	توضیحات
getPixelRGB	$O(1)$	پیدا کردن مقدار یک خانه مشخص در آرایه نامپای
getPixelEnergy	$O(1)$	تعداد ثابتی دسترسی به تابع بالا و چندین محاسبه ضرب و تفریق
updateImageDimension	$O(1)$	پیچیدگی زمانی تابع shape در نامپای ثابت است.
updateEnergyMatrix	$O(h * w)$	دو حلقه تو در تو بر روی طول و عرض عکس. می-توانستیم به صورت Vectorize حساب کنیم اما تصویر خروجی از انرژی کیفیت خوبی نداشت.
updateVerticalSeam	$O(h * w) + O(h) = O(h * w)$	الگوریتم DP که دو حلقه تو در تو بر روی طول و عرض عکس و پس از آن Backtrack کردن بر روی ارتفاع عکس

updateSeamImage	$O(h*w)*O(\max(h,w))$ $=$ $O(h^2*w)$ یا $O(h*w^2)$	به دست آوردن ماکس انرژی موجود در ماتریس سپس دو حلقه تو در تو بر روی طول و عرض عکس و تبدیل انرژی هر پیکسل به RGB همچنین قرمز کردن پیکسل های موجود در درز
updateAll	$O(h*w*\max(h,w))$	حاصل جمع پیچیدگی ۴ تابع جمع می‌باشد.
deleteVerticalSeam	$O(1)$	مجموعه ای از توابع stack، reshape نامپای که پیچیدگی تمامی آن ها عدد ثابت است.
saveSeamImage	$O(1)$	تولید و ذخیره یک عکس از یک آرایه نامپای
saveRealImage	$O(1)$	تولید و ذخیره یک عکس از یک آرایه نامپای
removeColumns	$O(c*h*w*max\ (h,w)\ )$	یک حلقه c مرتبه ای که در آن توابع آپدیت، ذخیره عکس و حذف درز صدا می‌شوند.
removeRows	$O(r*\ h*w*max\ (h,w))$	مشابه تابع بالا اما این بار در هر مرحله یک بار آرایه نامپای دوران ۹۰ درجه میکند و چون این دوران ربطی به داده ها ندارد و وابسته به shape است پیچیدگی آن ثابت است.

جدول پیچیدگی زمانی تابع remove

Mode	Time Complexity	توضیحات
Quality Mode	$O((c+r)*h*w*\max(h,w))$	ابتدا تابع removeColumns سپس removeRows و درنهایت تصویر نهایی ذخیره می‌شود.
Time Mode	$O(h*w)+O(h*w)+$ $O(c+r)*\big(O(1)+O(h*w)+O(1)\big)=$ $O((c+r)*h*w)$	تنها یک‌بار توابع updateEnergyMatrix updateSeamImage صدا زده می‌شوند و در هر Iteration توابع updateImageDimension updateVerticalSeam deleteVerticalSeam فراخوانی می‌شوند.