Projeto Seam Carving: Documentação

Matheus Popst

14 de abril de 2019

1 Funções principais

1.1 opticl:read-jpeg-file

Função do pacote opticl que lê arquivos JPEG. A sintaxe é simples, basta colocar entre áspas o caminho do arquivo.

```
(opticl:read-jpeg-file "lenna.jpg")
(defvar lenna (opticl:read-jpeg-file "lenna"))
```

1.2 energy-calculation

É uma função que retorna uma matrix com a mesma altura e largura da matriz de entrada com o gradiente para cada ponto, segundo o site. Há de se mencionar aqui que fiz à luz dos conhecimentos proporcionados no texto, por isso as bordas podem parecer estranhas.

```
(energy-calculation image)
```

1.3 find-vertical-seam

Esta função irá achar uma seam vertical e retornar uma lista que contém a ordenada de cada um dos elementos, de cima pra baixo.

```
(find-vertical-seam image energy)
```

1.4 make-image

Não estava nos enunciados do problema, mas é uma função que cria a imagem energética de uma dada imagem. O retorno é uma imagem já no formato JPEG.

```
(make-image image)
```

Para se criar de fato a imagem, basta usar `(opticl:write-jpeg-file path (make-image image))`

1.5 make-it-black

A exemplo da função anterior, também não estava nos enunciados. Serve apenas para visualizar passos intermediários do problema. Ela irá retornar uma imagem JPEG com os pixels do seam que seria retirado pintados de preto. Serve para que entendamos como o algortimo funciona.

```
(make-it-black image vertical-seam)
(opticl:write-jpeg-file path (make-it-black image (find-vertical-seam image energy)))
```

1.6 reduce-horizontal

Função que dada uma imagem e a quantidade de pixels que você deseja reduzir na horizontal, retorna uma imagem com o `redue-horizontal` aplicado aquele tanto de vezes. A função tem complexidade $O(w \times h \times n)$ (onde n é a quantidade de pixels a serem reduzidos), mas mesmo assim pode demorar muito, quando w e n são grandes. Então criei um função para tentar resolver isso, mas ainda que seja a mesma complexidade da anterior, não é toda iteração que irá demorar tanto. Mas antes, a explicação de `reduce-horizontal`

```
(reduce-horizontal image qtd)
(reduce-horizontal lenna 100)
```

A função `fast-reduce-horizontal` irá tentar reaproveitar a matriz de energia usada no passo anterior. Assim, apenas em algumas iterações ela irá recalcular toda a matriz de energia. Nas outras, ele usará a função `fast-energy-calculation` que apenas recalcula o gradiente num determinado range do vertical-seam de onde foi tirado os pixels da iteração anterior. Assim ele não recalcula toda a matriz de energia (que é o passo mais custoso do programa).

```
(fast-reduce-horizontal image qtd passo &optional range)
```

O passo é a quantidade de vezes que o programa ficará sem calcular toda a matriz de energia. Se o passo for 5, ele reaproveitará a matriz de energia 4 vezes e na quinta, recalculará novamente. Já range é o raio do intervalo de pixels que você quer que sejam recalculados a energia, a cada iteração.

1.7 reduce-vertical

Mesma coisa que o `reduce-horizontal`. Basicamente a função transpõe a matriz e chama `reduce-horizontal` (que pode ser feito com custo menor que $O(w \times h \times n)$, usando apenas $O(w \times h)$, o que é facilmente pagável.