MAC317 – Introdução ao Processamento de Sinais Digitais
Prof. Marcel Parolin Jackowski
Departamento de Ciência da Computação
IME/USP – Segundo Semestre de 2019
Segundo Exercício-Programa
Data de entrega: até 8/11/2019 às 23h55

Descritores de Fourier

Neste segundo exercício-programa (EP), exercitaremos uma aplicação real da transformada discreta de Fourier (DFT), a representação de contornos digitais de formas bidimensionais. Esta é uma técnica bem estabelecida na literatura para representação e avaliação de similaridade entre formas [1].

Um contorno digital no plano contendo N pontos, tendo como origem um ponto arbitrário (x_0, y_0), pode ser descrito por uma sequência de coordenadas $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_{N-1}, y_{N-1})$. Esta sequência é construída após um processo de rastreamento no sentido horário ou antihorário do contorno (do contorno mais externo, no caso de buracos) de uma forma bidimensional. Um contorno digital é representado pela sequência de coordenadas $s(n) = [x(n), y(n)]$, para $n = 0, 1, 2, ..., N-1$. Sem perda para a representação, cada coordenada pode ser representada por um número complexo tal que:

$$s(n) = x(n) + jy(n) \tag{1}$$

para $n = 0, 1, 2, ..., N-1$. Desta forma, o eixo x é representado pela parte real e o eixo y pela parte imaginária. Esta representação possui uma grande vantagem, ela nos propicia reduzir a dimensionalidade da representação para uma dimensão. Assumindo que a transformada discreta de Fourier de s seja descrita como:

$$S(k) = \sum_{n=0}^{N-1} s(n)e^{-2i\pi k\frac{n}{N}} \tag{2}$$

onde $k \in [0, N-1]$, os coeficientes complexos $S(k)$ são chamados de descritores de Fourier do contorno digital s. A transformada inversa destes descritores que restauram os valores de $s(k)$ pode ser descrita como:

$$s(n) = \frac{1}{N}\sum_{k=0}^{N-1} S(k)e^{2i\pi k\frac{n}{N}} \tag{3}$$

Agora, suponha que utilizemos somente os primeiros P coeficientes. Isso é equivalente a atribuir $S(k) = 0$ para $k > P-1$ na equação 3. O resultado gera a seguinte aproximação de $s(n)$:

$$\hat{s}(n) = \frac{1}{P}\sum_{k=0}^{P-1} S(k)e^{2i\pi k\frac{n}{P}} \tag{4}$$

para $n \in [0, N-1]$. Embora somente P termos são utilizados para a obtenção de cada componente $\hat{s}(n)$, n ainda está no intervalo entre 0 e $N-1$. Isto é, o mesmo número de pontos existirão no contorno aproximado, mas contando com um número diferente de termos para a reconstrução de cada ponto. Conforme vimos em aula, componentes de alta frequência determinam detalhes em um sinal, enquanto componentes de baixa frequência representam a forma global do sinal. Assim, a utilização de valores pequenos de P resultarão na perda de detalhes presentes no contorno digital.

Neste EP, você deverá demonstrar esse efeito, primeiramente rastreando o contorno de uma figura bidimensional e depois se utilizar de diferentes valores de P para a filtragem e posterior reconstrução do contorno.

Enunciado

Você deverá criar um programa em Python que recebará como entrada um arquivo de imagem do tipo `.png`, irá rastrear um contorno de um de seus canais C, de intensidade prescrita I, e irá produzir duas imagens resultantes `.png`, uma contendo o contorno rastreado, e outra contendo o contorno após filtragem utilizando-se de $p\%$ para das N possíveis frequências, onde $p \in (0, 100]$ ou $p \in [-100, 0)$. Valores positivos de p representam as $p\%$ menores frequências (e.g. filtro passa-baixa), e valores negativos de p representam as $|p|\%$ maiores frequências (e.g. filtro passa-alta).

Em ambas as imagens de saída, os contornos deverão ter intensidades 255 e o fundo intensidades 0, e devem ter 3 (RGB) ou 4 canais (RGBA). Para esta tarefa, você poderá utilizar somente as bibliotecas Numpy, Scipy, e Imageio. Você deverá consultar o professor em relação à outras bibiliotecas. Abaixo segue um exemplo de execução:

```
$ ./main.py fox.png 3 255 10.5 fox_ctr.png fox_ctr_filtered.png
```

onde $C = 3$ (opacidade), $I = 255$ e $p = 10.5\%$ das frequências. Os nomes `fox_ctr.png` e `fox_ctr_filtered.png` representam os arquivos com os contornos após o rastreamento e após a filtragem, respectivamente. Você deverá verificar e abortar seu programa graciosamente quando eventuais valores fora da faixa para cada parâmetro forem especificados.

Rastreamento

Uma vez de posse da matriz (Numpy) contendo somente o canal prescrito da imagem, você deverá varrer a matriz, de cima para baixo, da esquerda para a direita, localizando o primeiro ponto (x_0, y_0) com intensidade I. Defina uma variável chamada `dir` que guardará a direção do movimento anterior que nos trouxe para o pixel atual. Inicialmente atribua `dir=7` (Figura 1).

Assumindo esse ponto como centro de uma janela 3x3, você deverá agora escolher o próximo pixel, como sendo o primeiro vizinho de intensidade I no sentido **anti-horário**, começando na direção:

1. `(dir + 7) mod 8` se `dir` for par
2. `(dir + 6) mod 8` se `dir` for ímpar

O primeiro pixel com intensidade I agora é escolhido e atualiza-se o valor de `dir`. Se o pixel atual for igual ao segundo pixel rastreado (x_1, y_1) e o pixel anterior for igual a (x_0, y_0), pare, e termine, caso contrário continue o processo de rastreamento.

Este vetor de posições após o rastreamento (excluindo os dois últimos pontos visitados) representará a sequência de coordenadas desejada $s(n)$. Você poderá utilizar funções do Numpy/Scipy para eftuar a localização de pontos (reduzindo o número de laços), mas não funções já concebidas para retornar contornos ou regiões.

Filtragem

Para efetuar a conversão de $s(n)$ para o domínio de frequência, e de volta para o domínio do espaço, você não deverá implementar a DFT, mas sim utilizar-se do algoritmo FFT já presente em funções na biblioteca Numpy/Scipy, conforme utilizados em aula.

Questionário

Após a implementação, utilize o arquivo `fox.png` para responder as seguintes perguntas, salvando os arquivos de saída correspondentes às suas respostas.

1. Se você somente mantiver a frequência fundamental (DC) após a filtragem, o que acontece com o contorno digital após a reconstrução?

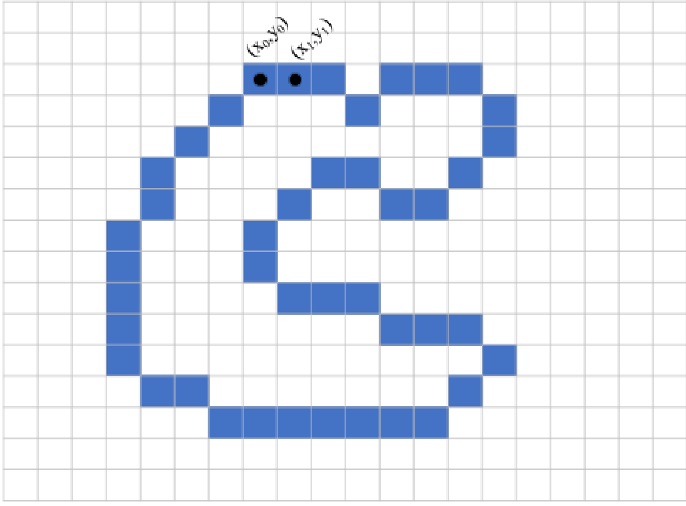

Figura 1: Contorno digital com ponto inicial (x_0, y_0) e segundo ponto da sequência (x_1, y_1). Você deverá rastrear todos os pontos atingíveis a partir do ponto inicial no sentido horário até chegar novamente ao ponto inicial ou até não puder mais prosseguir.

2. Se você mantiver todas as frequências, exceto a fundamental (DC), o que acontece com o contorno após a reconstrução?
3. Para o arquivo de entrada `fox.png`, qual o menor valor de p, onde $p > 0$, para que o contorno mantenha a sua geometria original ? Neste item você precisará calcular o somatório das distâncias Euclideanas em pares de pontos entre filtragens sucessivas e estabelecer um valor de tolerância para dizer que duas reconstruções são iguais (e.g. $\epsilon = 10^{-3}$).

Resultados a serem entregues

- Código-fonte interpretável (Python)
- Respostas e imagens correspondentes ao questionário;

Avaliação

Você poderá fazer o EP individualmente ou em grupo de até 2 pessoas. Você deverá entregar um arquivo `.zip` contendo toda a sua solução, incluindo um arquivo `README` que mencione os membros da equipe e descreva concisamente a sua solução. Favor entregar somente um arquivo por grupo. Qualquer sinal de plágio resultará em nota 0 para todos os envolvidos, então não mostre seu código a ninguém.

Tabela de avaliação:

1. **(1,0 pts)** Programa é imune à valores fora da faixa tolerável para cada parâmetro
2. **(2,0 pts)** Programa gera imagem de saída com contorno rastreado correto, dado conjunto de parâmetros
3. **(2,5 pts)** Programa gera imagem de saída com contorno filtrado correto, dado conjunto de parâmetros
4. **(2,5 pts)** Questionário respondido e arquivos de imagens corretamente corroboram as respostas.
5. **(2,0 pts)** Arquivo `README` incluso contendo descrição da implementação e código-fonte documentado.

Referências

[1] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.