Universidade Estadual de Campinas

MC920 - Introdução ao processamento de imagem digital

Trabalho 1

Felipe Hideki Matoba
RA: 196767

Sumário

1 Introdução

1.1 Resumo

Neste trabalho serão implementadas e discutidas as primeiras operações em imagens. Para as imagens coloridas serão feitas duas modificações, em uma os valores de RGB serão alterados segundo uma fórmula e na outra a imagem será convertida para monocromática. Nas imagens monocromáticas será aplicada a operação de correlação segundo as máscaras encontradas na seção 5.

1.2 Ambiente de execução

O código foi feito utilizando o Jupyter Notebook, com a inclusão das seguintes bibliotecas:

- skimage, versão 0.17.2
- numpy, versão 1.19.2
- matplotlib, versão 3.3.2

Para executar o código, basta executar cada célula do notebook, com o cuidado de rodar a importação das bibliotecas pertinentes e a definição das funções (especificamente para o caso das imagens monocromáticas) previamente.

2 Imagens coloridas

2.a Item a

Neste item, para cada pixel da imagem os valores de R, G e B foram alterados segundo a fórmula a seguir:

$$
\begin{aligned}
R' &= 0.393R + 0.769G + 0.189B \\
G' &= 0.349R + 0.686G + 0.168B \\
B' &= 0.272R + 0.534G + 0.131B
\end{aligned}
$$

A imagem original e a resultande podem ser vistas a seguir:

(a) Imagem original (b) Imagem alterada

Figura 1: Comparação entre a imagem original e a resultante do primeiro filtro

A imagem gerada pela aplicação do filtro assemelha-se bastante com o que se espera da aplicação de um filtro do tipo sépia, que confere à imagem um tom fortemente amarelado.

Observando a fórmula aplicada, nota-se que os novos valores para as cores da imagem atribuem um peso maior para o R (vermelho) e G (verde), especialmente esta última. Além disso, temos que o valor de R da nova imagem será sempre o maior, seguido do G e, por último, do B, devido a diferença dos pesos em cada fórmula.

De fato, calculando os valores médios dos pixels RGB obtemos 173.3, 154.8 e 120.5, respectivamente. Temos, então, que o tom amarelado origina de valores de azul mais baixos.

2.b Item b

Desta vez, a imagem resultante possuirá apenas uma banda, resultado de uma média ponderada entre os valores de RGB, dada pela seguinte fórmula:

$$I = 0.2989R + 0.5870G + 0.1140B$$

Intuitivamente, teremos uma imagem monocromática, que pode ser observada a seguir:

(a) Imagem original (b) Imagem alterada

Figura 2: Comparação entre a imagem original e a resultante do segundo filtro

Destaco que, nesse caso, não é necessário se preocupar com valores acima de 255 (máximo valor na escala de cinza), pois os valores de RGB também são limitados a 255 e uma operação de média ponderada não poderá gerá-los.

3 Imagens monocromáticas

Para a aplicação dos filtros foi feita uma função *filter*, onde o tratamento aplicado para valores de pixels fora do intervalo esperado, de [0, 255], foi simplesmente limitá-los a esse intervalo, ou seja, valores negativos foram transformados em zero, enquanto números acima de 255 tornaram-se 255. Como será discutido adiante, isso gerou problemas na combinação dos filtros h1 e h2, então foi feita uma segunda função, *filterNoTreatment*, que não realizava este tratamento previamente.

Além disso, foi necessário a aplicação de *padding* nas imagens para que a máscara pudesse ser aplicada nas regiões de borda da imagem. Para isso, foi utilizada a função *pad* da biblioteca numpy, que realizou o *pad* com zeros. Também foi testado com o valor de 255, mas não houveram mudanças significativas no resultado.

Todas as funções foram feitas de maneira vetorizada, sendo que para a aplicação das máscaras nas imagens monocromáticas foi utilizada a funcção *view_as_windows* do sklearn para auxiliar na subdivisão da matriz da imagem original. Com isso, o desempenho do código aparenta estar bem satisfatório, não demorando mais que poucos segundos para executar com os exemplos utilizados, geralmente de tamanho 512x512.

(a) Imagem original (b) Filtro h1 (c) Filtro h2

Figura 3: Comparação entre a imagem original e as imagens filtradas com máscaras h1 e h2

3.1 Filtros h1 e h2

Uma análise inicial dos filtros nos permite perceber que em h1 é feita a diferença dos pixels da direita com os da esquerda, enquanto em h2 essa diferença é entre os pixels de baixo com os de cima. Ao observar as imagens resultantes, fica claro o efeito dessa operação na imagem.

É fácil perceber que na figura 3b (filtro h1) temos várias linhas brancas que separam os objetos, orientadas verticalmente, enquanto na 3c (filtro h2) a orientação é na horizontal. Esses filtros detectam, portanto, bordas verticais e horizontais, respectivamente.

De maneira intuitiva, uma combinação desses dois filtros geraria uma imagem com todas as bordas destacadas, o que pode ser feito por meio da operação $\sqrt{(h1)^2 + (h2)^2}$, onde h1 e h2 são as imagens resultantes da aplicação de h1 e h2.

Vale ressaltar que, como comentado no início da seção, o ajuste dos valores da imagem deve ser feito no momento correto. Caso isso seja feito antes de se realizar a conta que gera a imagem combinada, muita informação é perdida e a imagem não tem o resultado esperado. Abaixo seguem os resultados do ajuste antes e depois do cálculo da nova imagem.

(a) Combinação dos filtros com perda

(b) Combinação dos filtros sem perda

Figura 4: Comparação entre as imagens resultantes da combinação do filtro, com e sem perda de informação

Perceba que a imagem gerada com perda de informação é extremamente escura, o que pode ser explicado pelo fato de que as imagens geradas por h1 e h2 possuíam muitos valores negativos que foram zerados e, portanto, geram valores muito baixos na imagem combinada, que correspondem a tons escuros.

Quando esse corte não é feito, os valores negativos, elevados ao quadrado, geram números mais altos para a imagem 4b, que tem as bordas em branco, ou seja, com valores próximos a 255, como era esperado.

3.2 Filtros h3 e h8

Esses dois filtros foram agrupados nessa seção por possuírem um formato bem semelhante: consistem na diferença entre o pixel central e os na vizinhança, sendo que a soma dos pesos dos pixels vizinhos é igual ao do central.

(a) Imagem original (b) Filtro h3 (c) Filtro h8

Figura 5: Comparação entre a imagem original e as imagens filtradas com as máscaras h3 e h8

Lembrando do resultado dos filtros h1 e h2, que consistiam na diferença entre os pixels na horizontal e vertical, nesse caso podemos esperar uma detecção de borda em todas as orientações, semelhante com o resultado obtido pela combinação dos filtros h1 e h2, o que de fato ocorre.

Observe que o filtro h8 aparenta ser mais sensível do que o h3, o que pode ser explicado pelo fato de que a matriz utilizada é maior (5x5).

3.3 Filtros h4 e h9

Dessa vez, temos dois filtros que tiram uma média entre o pixel central e aqueles em sua vizinhança.

(a) Imagem original (b) Filtro h4 (c) Filtro h9

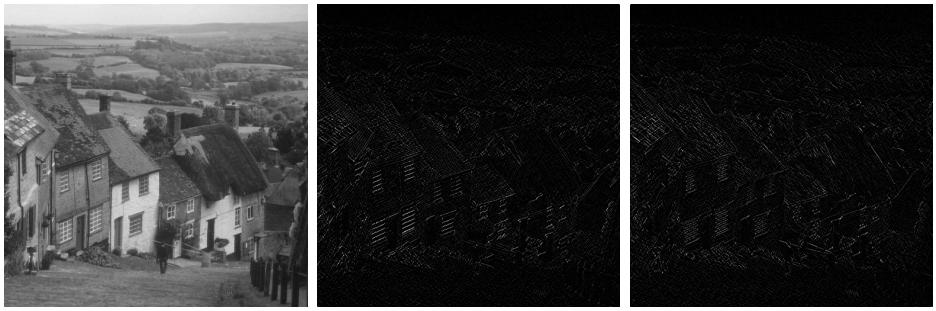
Figura 6: Comparação entre a imagem original e as imagens filtradas com as máscaras h4 e h9

O efeito desses filtros é de reduzir a definição das imagens, pois a diferença entre pixels vizinhos é diminuída pelo cálculo da média, aritimética em h4 e ponderada em h9.

A diferença entre as imagens filtradas não é realmente perceptível, apesar do tamanho da máscara do segundo ser maior, possivelmente porque o pixel central possui um peso maior no cálculo da média, o que diminuiria o efeito de borramento.

3.4 Filtros h5 e h6

Novamente, temos uma máscara que considera a diferença entre pixels, dessa vez entre a diagonal secundária e os demais pixels no caso de h5 e a diagonal principal e os outros pixels para h6. O resultado é, como se espera, semelhante a uma detecção de borda, dessa vez nas diagonais.



(a) Imagem original (b) Filtro h5 (c) Filtro h6

Figura 7: Comparação entre a imagem original e as imagens filtradas com as máscaras h5 e h6

3.5 Filtro h7

Finalmente, o filtro h7 simplesmente subtrai o pixel inferior esquerdo do pixel superior direito, considerando uma matriz 3x3, sendo que podemos esperar, novamente, algum tipo de detecção de borda desse filtro.

(a) Imagem original (b) Filtro h7

(a) Imagem original (b) Filtro h7

Figura 9: Comparação entre a imagem original e a imagem gerada pelo filtro h7

Esse filtro aparenta ser especialmente bom para detectar as texturas dos exemplos apresentados, mas sua função primária provavelmente é de detecção de bordas.

4 Conclusão

Neste trabalho foram realizadas as primeiras operações em imagens, que nos ajuda a entender melhor como funciona o processo de filtragem, tão presente no cotidiano moderno após a popularização dos smartphones.

Também foi adquirida uma intuição melhor com o uso do numpy e, portanto, da vetorização do código, algo que não é usual de se utilizar em outras disciplinas, o que será muito importante para o futuro desse curso e da carreira profissional.

Por fim, com a aplicação de máscaras nas imagens, também foi possível visualizar o efeito que diferentes pesos e operações aplicados em regiões específicas das máscaras pode ter no resultado, seja por meio de médias, adições ou subtrações.

5 Máscaras

$$h1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad h2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad h3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$h4 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad h5 = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad h6 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \quad h7 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$h8 = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \quad h9 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$