Trabalho 2 - Introdução ao Processamento de Imagens Digitais

Nome

Victor Palmerini

RA

178061

Data

24/04/2020

Conteúdos

Introdução

Este é o Trabalho 2 da disciplina ***MC920*** - *Introdução ao Processamento de Imagens Digitais* da **Unicamp** - Universidade Estadual de Campinas.

O principal objetivo deste trabalho é desenvolver uma boa noção sobre alguns filtros e suas respectivas máscaras e quais os efeitos destes em imagens digitais monocromáticas.

O trabalho requer por parte do aluno um conhecimento básico da linguagem de programação `python` e bibliotecas que facilitem a manipulação das imagens digitais bem como seus processamentos. No caso deste trabalho, foram usados os pacotes `numpy`, `pillow` e `jupyter notebook`. Estes pacotes serão mencionados com mais detalhes no decorrer do relatório.

Desenvolvimento

Estrutura do Projeto

1. `/notebooks` - é a pasta que contém os notebooks com os algoritmos para serem executados pelo `jupyter notebook`
2. `requirements.txt` - é um arquivo texto que contém as dependências para executar a aplicação (foi gerado pelo pacote `pip` através do comando `pip freeze > requirements.txt`)

3. `Pipfile` e `Pipfile.lock` - são os arquivos relacionados ao `ambiente virtual` aonde as dependências são instaladas e executadas. Isso permite que as dependências sejam instaladas apenas no ambiente virtual e não no ambiente local.

4. `/images` - pasta com algumas imagens usadas como entrada para execução dos notebooks

Rodando Localmente

Dependências

Para executar os notebooks é necessário instalar as seguintes dependências:

1. `Python 3` - neste projeto foi usada a versão `3.7.3`. Qualquer versão do `python` a partir da `3` é suficiente.

2. `Pipenv` - gerenciador de ambientes virtuais. É equivalente ao `virtualenv`.

Inicialização do Ambiente

Na pasta root do projeto, execute os seguintes comandos em uma `shell`:

1. `pipenv shell` - para iniciar um ambiente virtual localmente

2. `pipenv install -r requirements.txt` - instala no ambiente virtual todas as dependências listadas no arquivo `requirements.txt`

Executando os notebooks

Para executar os notebooks, há 2 caminhos:

1. Executar `jupyter notebook` em uma `shell` para subir um servidor do `jupyterlab` que vai abrir automaticamente o navegador padrão com o ambiente `jupyter` e as pastas e arquivos do projeto.

2. Caso o projeto tenha sido aberto no `Visual Studio Code`, uma opção é instalar a extensão `VS Code Jupyter Notebook`, que permite executar os notebooks no próprio `VS Code`.

Entradas

As entradas dos notebooks serão o `path` da imagem de entrada e o `path` da **pasta** de saída. As imagens de entrada utilizadas são *monocromáticas*, no formato `.png` e com 256 níveis de intensidade. Como foi falado, na pasta `/images` há algumas imagens que podem ser usadas como entrada.

Saídas

As saídas também serão imagens *monocromáticas* no formato `.png` com o respectivo filtro aplicado.

Implementação dos Processamentos

Máscaras h1 a h8

Os notebooks que implementam a aplicação das máscaras nas imagens de entrada estão em `/notebooks`, como já foi falado. A estrutura desses notebooks é muito parecida, por isso essa estrutura será discutida apenas nesta seção.

Em termos de implementação, o algoritmo em questão resume-se a:

1. Importação de bibliotecas necessárias (no caso desses filtros apenas a biblioteca `pillow` foi utilizada).
2.
   - `path` da imagem de entrada (ex: `../images/city.png`)
   - `path` da pasta de saída - (ex: `../outputs`, a pasta `/outputs` deve existir)
3. Definir a estrutura da máscara (matriz de inteiros)
4. Aplicar o filtro a partir da utilização da função `Filter` da biblioteca `pillow`. Neste, especifica-se a imagem para a qual se quer aplicar o filtro, a máscara definida, um fator pra dividir os elementos junto com a aplicação da máscara e um fator pra ser somado com os elementos junto com a aplicação da máscara. Esses 2 fatores serão 1 e 0, respectivamente, se nenhum valor for especificado para eles.
5. Salva-se cada imagem de saída no formato `.png` na pasta de saída especificada. O nomes das imagens seguirão o padrão `h{k}.png`, em que `k` é o número associado ao filtro (`h5.png`, por exemplo).

Exemplo:

- Fitro com máscara `h6` (`/notebooks/h6.ipynb`)
- Imagem de Entrada: `../images/baboon.png`
- Path de Saída: `../outputs/`
- Imagem de Saída: `../outputs/h6.png`

 *Imagem Original*

 *Imagem com Filtro **h6** aplicado*

Filtro sqrt(h3^2 + h4^2)

Este notebook em específico segue uma estrutura um pouco distinta dos demais, por isso sua implementação será discutida numa seção a parte. Isso porque tal filtro utiliza 2 máscaras e realiza operações sobre os resultados dessas máscaras, o que será explicado a seguir.

Em termos de implementação, o algoritmo em questão resume-se a:

1. Importação de bibliotecas necessárias.
2. - `path` da imagem de entrada (ex: `../images/baboon.png`)
   - `path` da pasta de saída - (ex: `../outputs`, a pasta `/outputs` deve existir)
3. Definir a estrutura das máscaras (matriz de inteiros) `h3` e `h4`.
4. Aplicação dos filtros, para cada máscara, a partir da utilização da função `Filter` da biblioteca `pillow`. Os resultados são armazenados em estruturas auxiliares.
5. Após a aplicação dos filtros, transforma-se as imagens em *arrays* `numpy`.
6. Normaliza-se as imagens para a escala `[0,1]` para aplicação da transformação quadrática (impede que os valores ultrapassem o valor de intensidade).
7. Aplica-se a transformação quadrática nas 2 imagens.

8. Por fim, soma-se as estruturas resultantes e aplica-se a transformação da raiz quadrada.

9. Feitas todas as operações, normaliza-se as imagens de volta pra escala `[0,255]` e esta então é salva como `sqrt.png` na pasta de saída especificada.

Exemplo:

- Imagem de Entrada: `../images/baboon.png`
- Path de Saída: `../outputs/`
- Imagem de Saída: `../outputs/sqrt.png`

Imagem Original

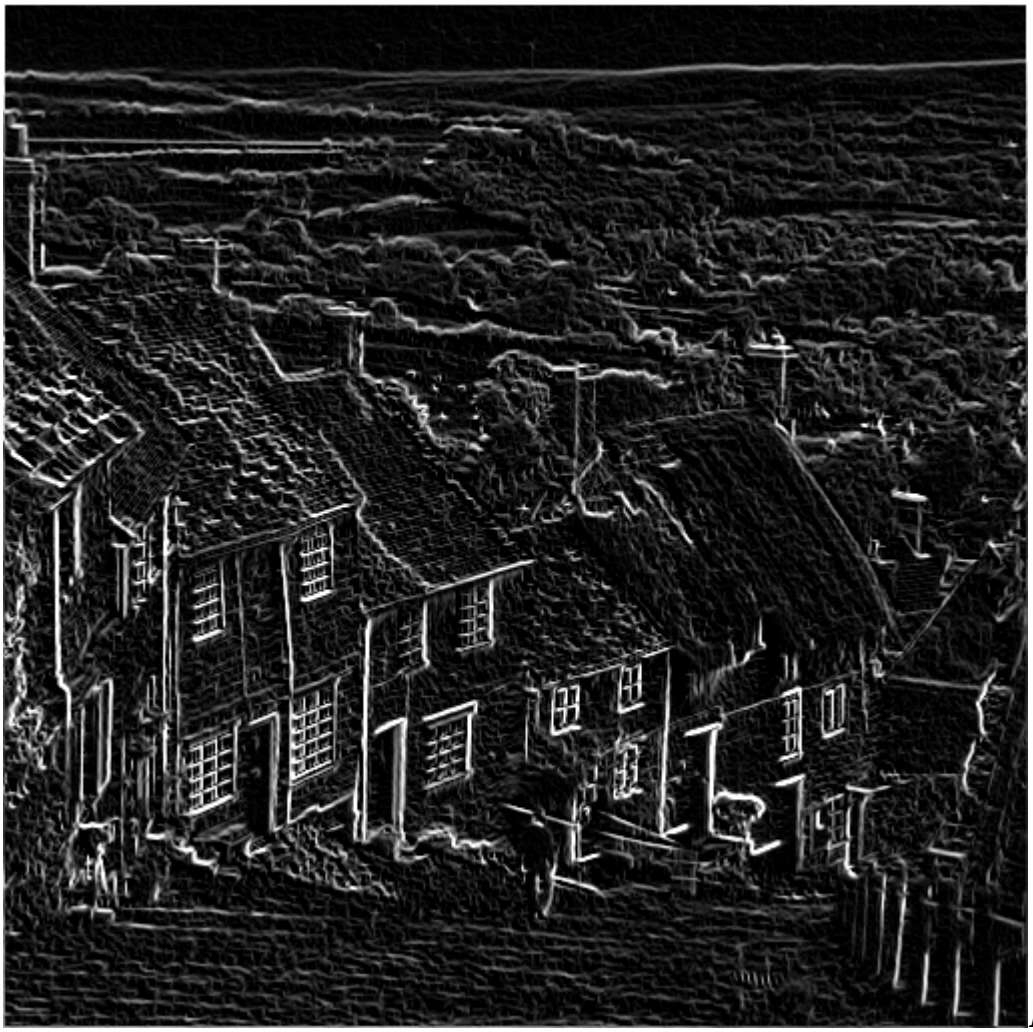


*Imagem com Filtro **sqrt(h3^2 + h4^2)** aplicado*

Análise

Como já foi dito, os filtros foram aplicados utilizando-se a função `Filter` da biblioteca `pillow`. Esta função define alguns padrões próprios que são refletidos nos efeitos aplicados sobre as imagens digitais. Um desses padrões, por exemplo, está relacionado à aplicação do filtro em pixels da borda. Nestes caso, a função `Filter` ignora esses pixels e aplica os filtros nos pixels em que a máscara consegue ser aplicada totalmente. Então pra máscaras `3x3` a função ignora *1 faixa* de pixels na borda e pra máscaras `5x5` a função ignora *2 faixas* de pixels na borda.

Segue-se então uma análise para o efeito obtido pela aplicação de cada filtro. A imagem de entrada utilizada foi a seguinte:

 *Imagem Original de Entrada*

Filtro h1

Este filtro utiliza a seguinte máscara:

```
h1 = (
    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
)
```

 *Imagem com Filtro h1*

O efeito obtido então é um realce bem destacado das regiões com transição de intensidades de cinza contrastantes. Isso já era esperado pela análise da máscara, que valoriza o pixel central e diminui o valor dos pixels vizinhos na vertical e horizontal.

Filtro h2

Este filtro utiliza a seguinte máscara:

```
h2 = 1/256 * (
    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
)
```

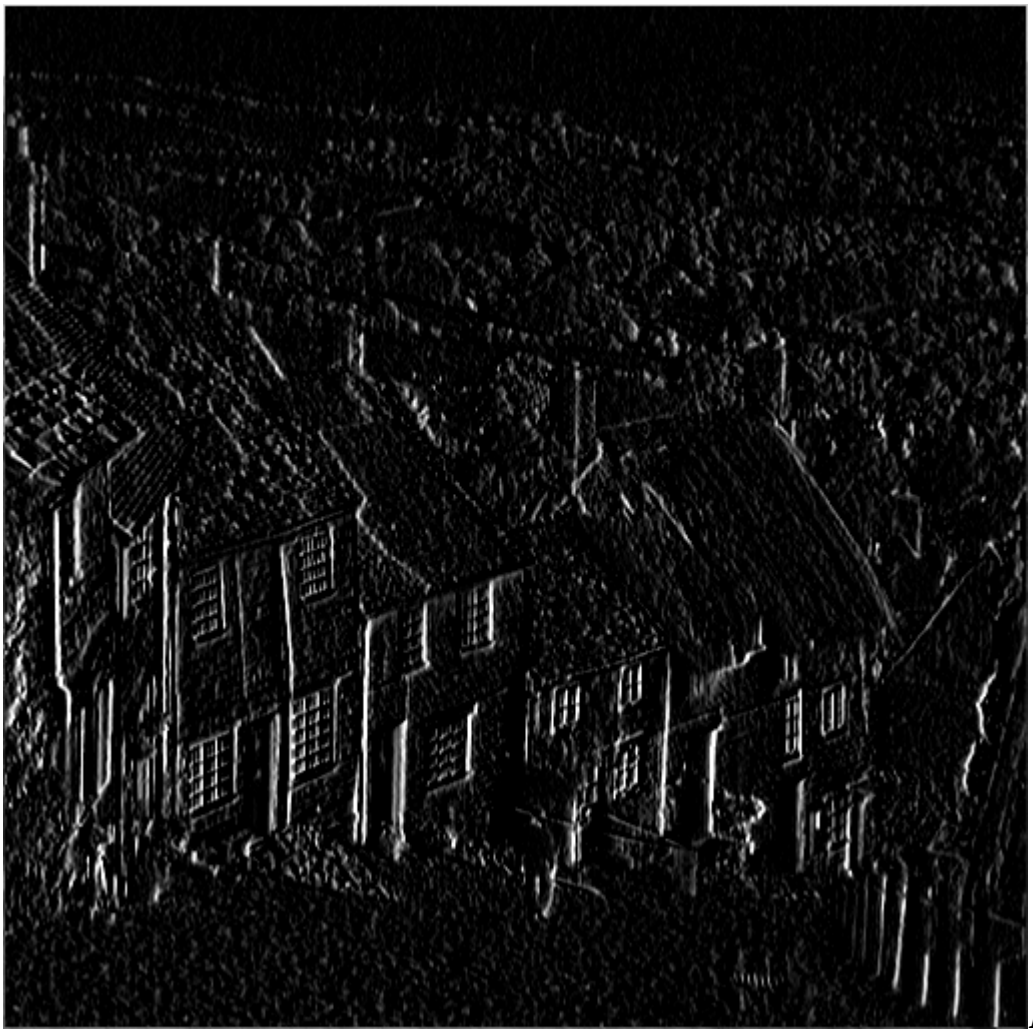

Imagem com Filtro h2

Com este filtro, o efeito obtido é o de *blur* da imagem como um todo. Observando-se a máscara utilizada, percebe-se que ela abrange toda a vizinhança do pixel central e ainda obtém um valor médio considerando esses vizinhos, suavizando então as transições de intensidade. Este filtro pode ser interessante para remover ruídos de uma imagem digital, por exemplo.

Filtro h3

Este filtro utiliza a seguinte máscara:

```
h3 = (
    -1, 0, 1,
    -2, 0, 2,
    -1, 0, 1,
)
```

 *Imagem com Filtro h3*

Percebe-se aqui que o efeito deste filtro é um realce bem destacado das regiões com transição de intensidades de cinza contrastantes, parecido com nesse sentido com o `Filtro 1`, mas aqui o filtro destaca os pixels vizinhos na direita e desvaloriza os vizinhos na esquerda. Dessa forma, o destaque fica mais nítido em regiões onde a transição de intensidades varia bastante.

Filtro h4

Este filtro utiliza a seguinte máscara:

```
h4 = (
    -1, -2, -1,
     0,  0,  0,
     1,  2,  1,
)
```

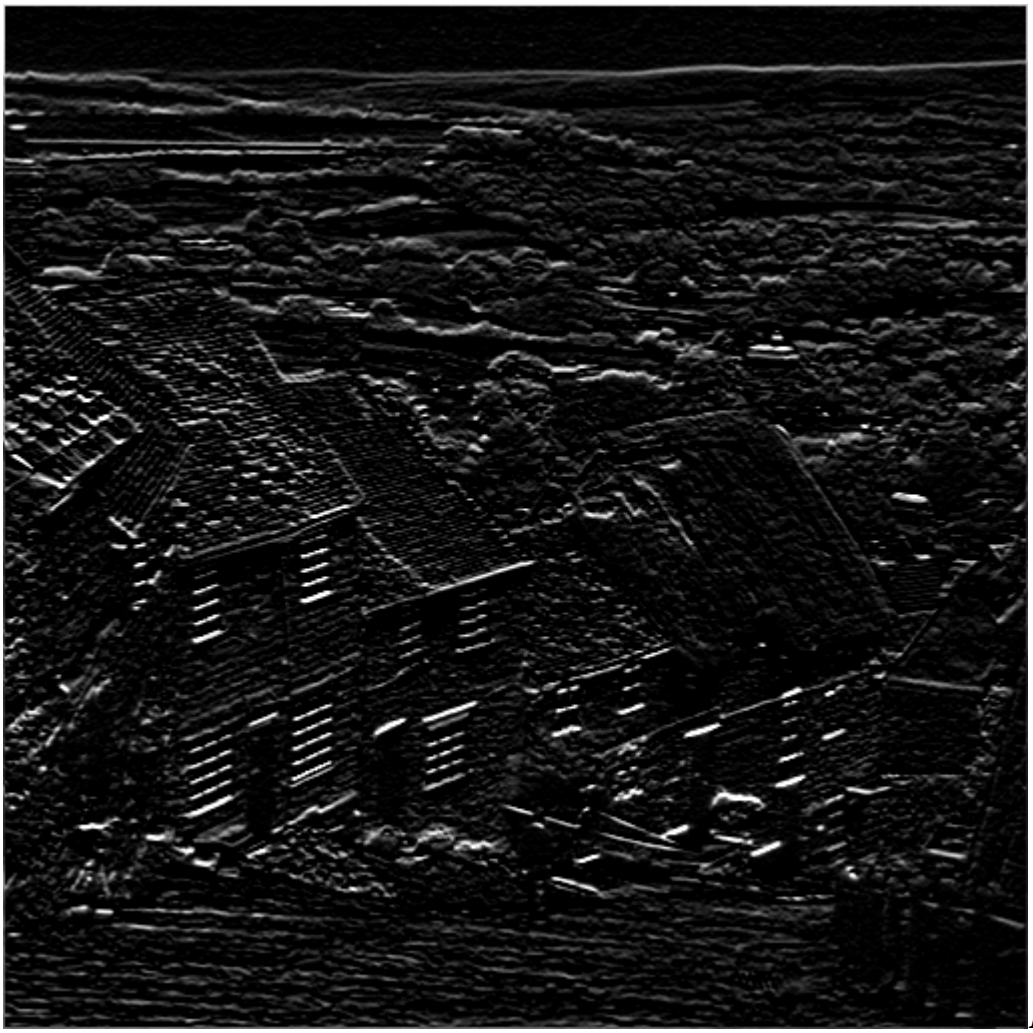


Imagem com Filtro h4

O efeito obtido aqui é extremamente parecido com o do `Filtro 3`, o qual possui um realce bem destacado das regiões com transição de intensidades de cinza contrastantes. Porém aqui esse efeito é aplicado na direção oposta. Enquanto no `Filtro 3` o efeito é observado na direção horizontal, aqui ele é observado na direção vertical.

Filtro h5

Este filtro utiliza a seguinte máscara:

```
h5 = (
    -1, -1, -1,
    -1,  8, -1,
    -1, -1, -1,
)
```

 *Imagem com Filtro h5*

Aqui percebe-se um efeito parecido com o do `Filtro h1`, o que é esperado quando se compara as respectivas máscaras. O efeito de destacar as bordas e transições de intensidade é atingido mas com menos intensidade que no `Filtro h1`.

Filtro h6

Este filtro utiliza a seguinte máscara:

```
h6 = 1/9 * (
    1, 1, 1,
    1, 1, 1,
    1, 1, 1,
)
```

 *Imagem com Filtro h6*

Tem-se aqui um efeito parecido com o do `Filtro h2`, o de *blur*. A intensidade dos pixels da imagem resultante é obtida a partir da média das intensidades dos pixels vizinhos.

Filtro h7

Este filtro utiliza a seguinte máscara:

```
h7 = (
    -1, -1,  2,
    -1,  2, -1,
     2, -1, -1,
)
```

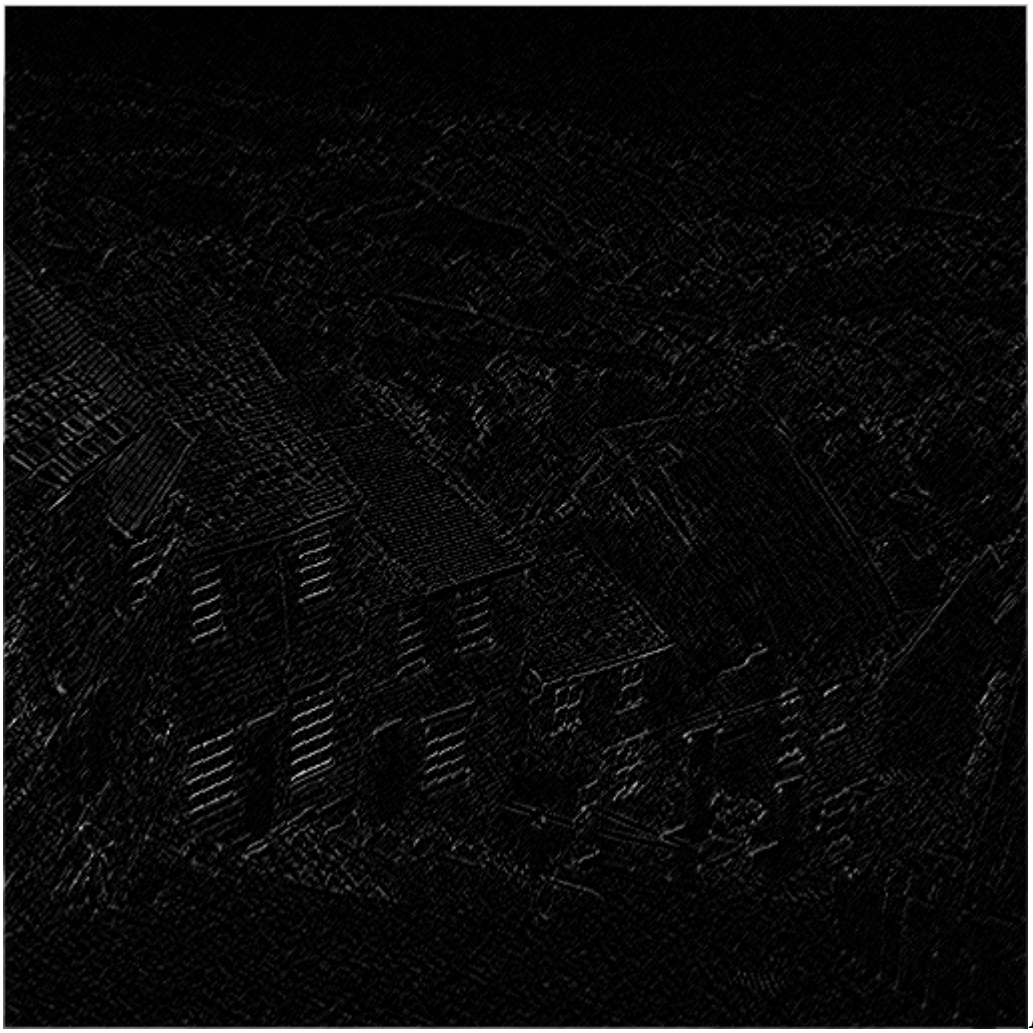
 *Imagem com Filtro h7*

Aqui percebe-se um efeito parecido com os do `Filtro h3` e `Filtro h4`, porém na direção diagonal. Isto poderia ficar intuitivo quando se compara as respectivas máscaras. O efeito de destacar as bordas e transições de intensidade é atingido mas com menos intensidade que nos filtros `h3` e `h4`.

Filtro h8

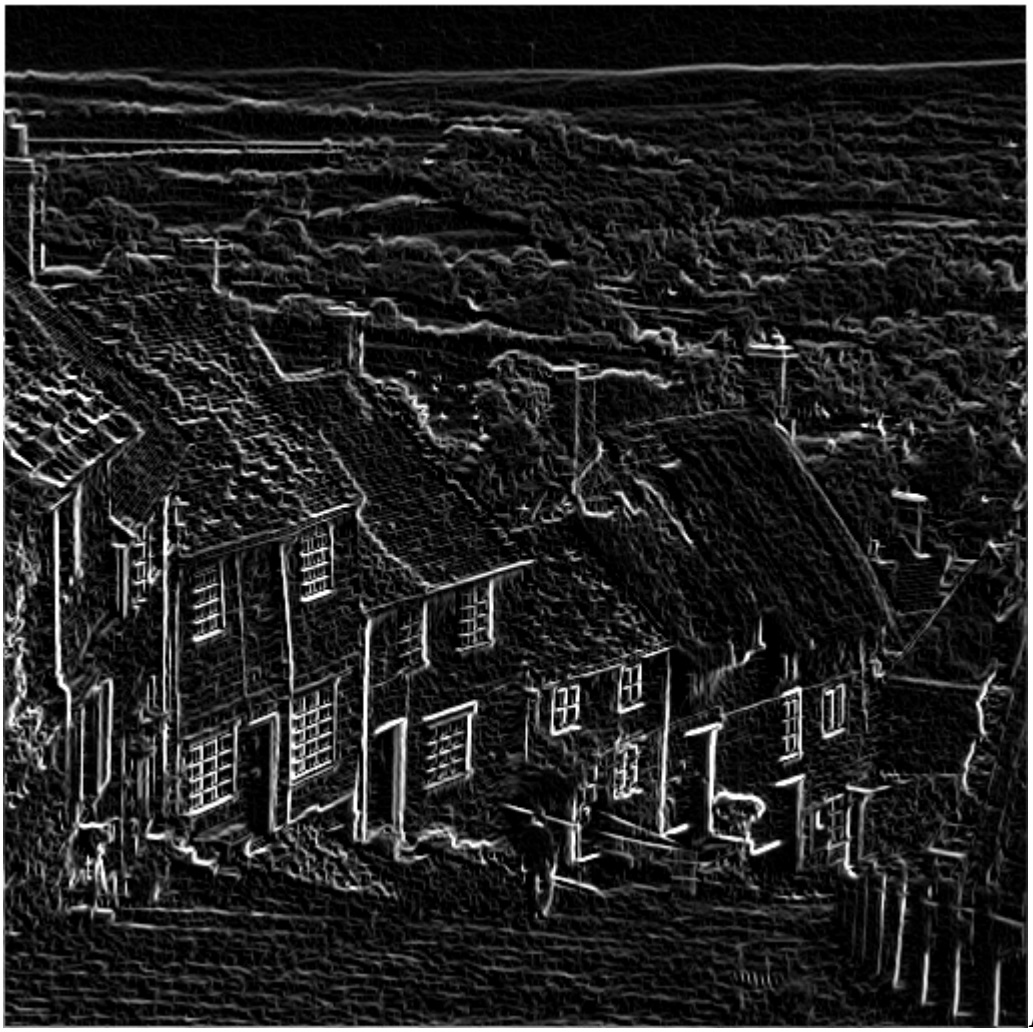
Este filtro utiliza a seguinte máscara:

```
h8 = (
     2, -1, -1,
    -1,  2, -1,
    -1, -1,  2,
)
```

 *Imagem com Filtro h8*

O efeito deste filtro é extremamente parecido com o do `Filtro h7`. A única diferença nítida é que aqui o efeito é atingido ainda na direção diagonal mas no sentido oposto.

Filtro sqrt(h3^2 + h4^2)

 *Imagem com Filtro sqrt(h3^2 + h4^2)*

Este filtro é uma mescla dos filtros `h3` e `h4` e seu efeito é muito parecido com os destes. Na operação optou-se por normalizar a intensidade pra escala `[0,1]` para evitar `overflow` dos valores e ao final da operação, volta-se pra escala `[0,255]`. Desta maneira a imagem resultante fica um pouco mais "clara" que as imagens dos filtros `h3` e `h4`. É possível também voltar pra escala `[0,255]` antes de aplicar a operação de raiz quadrada `sqrt()`. Neste caso a imagem resultante ficará um pouco mais "escura".

Limitações

Os resultados obtidos neste trabalho se aplicam e se limitam às imagens quadradas, monocromáticas e no formato `.png`.

Isto se deve principalmente aos tipos de processamentos implementados e às limitações das bibliotecas utilizadas.

Bibliografia

1. Documentação das bibliotecas usadas
    - Pillow
    - NumPy
    - Jupyter Notebook
2. *R.C. Gonzalez, R.E. Woods. Digital Image Processing. Prentice Hall, 2007.*
3. Material de aula fornecido pelo Professor