



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE  
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO  
E AUTOMAÇÃO  
COMPONENTE: SISTEMAS OPERACIONAIS

Marco Antonio Moreira Carujo

Professor: Diogo Pinheiro Fernandes Pedrosa

Natal / RN

Maio de 2017

<b>Sumário .....</b>	<b>2</b>
<b>1. A Atividade.....</b>	<b>3</b>
<b>2. O Código.....</b>	<b>6</b>
<b>3. A Lógica.....</b>	<b>8</b>
<b>4. Os Resultados.....</b>	<b>9</b>
<b>Referências Bibliográficas .....</b>	<b>10</b>

---

# 1

## A Atividade

---

A área de processamento automático de imagens digitais é particularmente importante para diversos segmentos. Os avanços obtidos com metodologias e técnicas para tratamento e extração de informação útil em imagens possibilita o desenvolvimento de estudos em saúde, engenharia, ensino, entre outros campos. O processamento de imagens busca justamente prover meios para aquisição, tratamento e extração de informação que possa ser relevante. Dentre várias técnicas envolvidas, o processamento de imagens inclui metodologias para detecção de bordas em imagens. A detecção de bordas tem por objetivo a detecção de descontinuidades em uma imagem com o intuito de determinação de objetos característicos. De forma simplificada, isto é obtido pela aplicação de operadores nas direções x e y na imagem. Estes operadores são pequenas matrizes que são usadas em uma operação de convolução com partes da imagem alvo.

Figura 1: Operação de convolução entre máscara e imagem alvo (extraído de [1]). Um tipo simples destes operadores para extração de bordas é o operador de Prewitt para as direções x e y.

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad (\text{direção vertical})$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (\text{direção horizontal})$$

Como exemplo, tem-se as figuras a seguir:

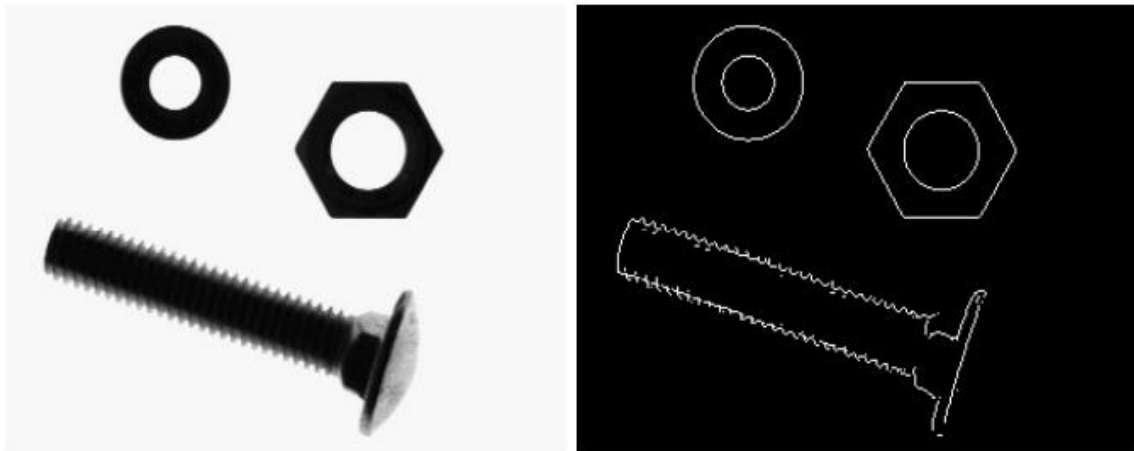


Figura 2: detecção de bordas com o operador de Prewitt (extraído de [2]).

Baseando-se nisto, elabore um programa em linguagem C ou C++ para que, além do processo pai, sejam criados mais dois processos filhos para que um deles proceda com a detecção de bordas no sentido horizontal e o outro, no sentido vertical. O processo pai deve esperar os filhos executarem e, após as suas finalizações, deve proceder com a junção das imagens resultantes em uma única imagem.

A imagem alvo é apresentada a seguir. Ela está disponível no SIGAA ou para download em [3] e está no formato PBM ascii. Maiores informações podem ser obtidas em [4] ou em [5].



#### Referências

[1] Teoria: Processamento de Imagens

(<http://www.dpi.inpe.br/spring/teoria/filtrage/filtragem.htm>)

[2] <http://www2.ic.uff.br/~jcarvalho/ai/lab8/index.htm>

[3]

<http://people.sc.fsu.edu/~jburkardt/data/pbma/washington.ascii.pbm>

[4] <http://netpbm.sourceforge.net/doc/pbm.html#plainpbm>

[5] <http://people.sc.fsu.edu/~jburkardt/data/pbma/pbma.html>

---

## 2

## O Código

---

Primeiramente houve a utilização de uma biblioteca : *fstream* a qual facilitou o trabalho de leitura e escrita de arquivos , e assim lemos os valores no formato *.pbm* e alocamos para um objeto que foi criado , chamado *imagem* a qual lia o tipo de arquivo , colunas , linhas e valor, para formar uma matriz dinamica e quando não precisar mais de ser manipulada, escrevia o arquivo.

```
class imagem{
public:
    string tipo,nome;
    int l,c;
    int **mat;

    imagem(char* nome);
    imagem(string a,int linha,int colunas);
    void salvarimagem(char* a);
    void fechar();
    void filtrar_horizontal(imagem A);
    void filtrar_vertical(imagem A);
    void somando_filtros(imagem A,imagem B);
};
```

O primeiro construtor, é o qual lê o arquivo e o segundo foi utilizado para uma criação de variavel auxiliar, para armazenar os resultados das minhas operações de filtragem.

Os metodos são descritos nos nomes: *salvar\_imagem* , é chamado emcima da imagem a ser salva, e recebe o nome do arquivo para salvamento. *fechar* desaloca a imagem (a matriz). *filtrar\_horizontal* faz a filtragem horizontal com o operador x de Prewitt recebendo a imagem a ser filtrada. *filtrar\_vertical* faz a filtragem vertical com o operador y de Prewitt recebendo a imagem a ser filtrada. E por ultimo *somando\_filtros* recebe duas imagens e simplesmente soma pixel a pixel resultando a imagem filtrada de Prewitt.

A criação dos processos ocorre com a utilização da chamada do sistema *fork*, utilizando a chamada de sistema primitiva *wait* fazemos o processo pai aguardar que o seu processo filho termine de executar sua função e ao fim do processo filho utilizamos outra chamada de sistema primitiva para encerrar o processo o *exit*.

```
if(fork()==0){
    cout << "\n\tProcesso filho de numero : " << getpid()
    imagem A("washington.ascii.pbm");
    imagem AH("P1",A.l,A.c);
    AH.filtrar_horizontal(A);
    AH.salvarimagem("washington.ascii_filtrado_horizontal.r
    A.fechar();
    AH.fechar();
    cout << "\tProcesso filho de numero : " << getpid() <<
    exit(0);
}
else {
    int ret1, status1;
    ret1 = wait(&status1) ;
```

O entendimento é que o *fork* é a única chamada de sistema que tem a capacidade de criação de um processo em **UNIX**. O processo pai e o processo filho, ambos possuem o mesmo código e ID's diferentes como está exemplificado no programa.

A chamada *wait* suspende a execução do processo pai até a morte de seu processo filho e retorna o ID do processo morto para a variável *ret1* e a variável *status1* recebe o parametro do *exit* para indentificar se o fim do processo filho ocorreu de maneira correta.

O *exit* tem a função de encerrar o processo, e passando um sinal que no exemplo aplicado é por convenção 0, o que significa que um código de retorno igual a 0 terminou normalmente e não possui retorno.

---

## 3

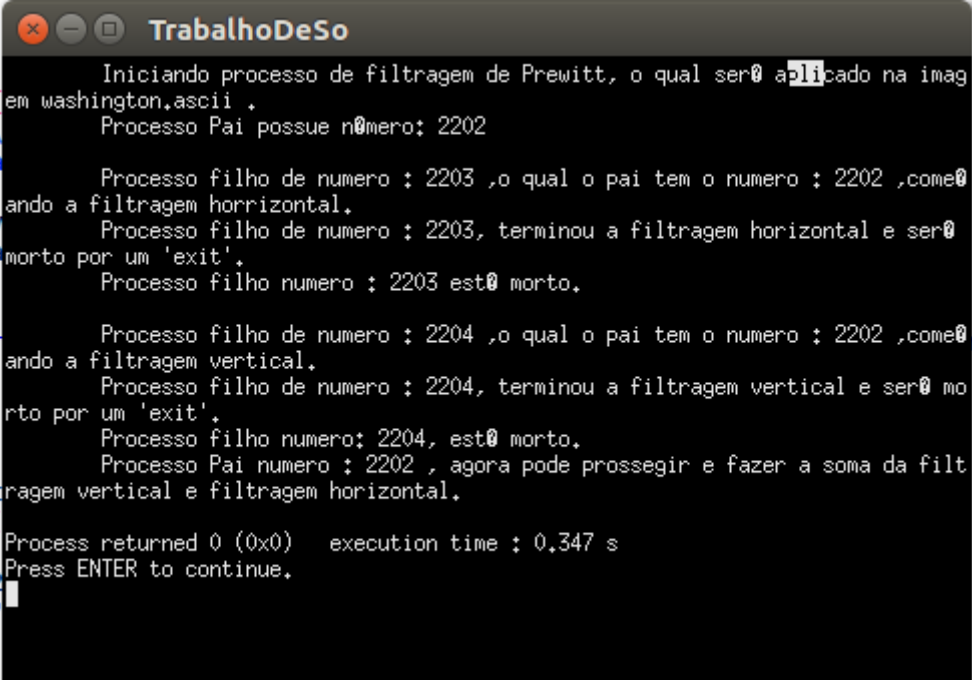
# A Lógica

---

A logica é bem simples, consiste em a inicialização do Processo Pai, e logo em seguida é criado um Primeiro Processo Filho que carrega a imagem a ser filtrada, faz a filtragem horizontal, salva e morre. Enquanto o Primeiro Filho faz o seu trabalho o Pai aguarda a morte do Primeiro Filho.

Com a morte do Primeiro Processo Filho ocorrida com sucesso, o Pai pode seguir e então cria novamente, um Segundo Processo Filho e ele irá carregar novamente a imagem a ser filtrada, fazer a filtragem vertical, salvar e morrer.

Novamente o Processo Pai irá aguardar que o Segundo Filho execute toda a sua função, e após a morte do Segundo Filho o Processo Pai poderá abrir as duas filtrações salvas por seus filhos, somar e salvar uma filtragem definitiva e final.



```
TrabalhoDeSo
Iniciando processo de filtragem de Prewitt, o qual ser@ aplicado na imag
em washington.ascii .
Processo Pai possui numero: 2202

Processo filho de numero : 2203 ,o qual o pai tem o numero : 2202 ,come@
ando a filtragem horizontal.
Processo filho de numero : 2203, terminou a filtragem horizontal e ser@
morto por um 'exit'.
Processo filho numero : 2203 est@ morto.

Processo filho de numero : 2204 ,o qual o pai tem o numero : 2202 ,come@
ando a filtragem vertical.
Processo filho de numero : 2204, terminou a filtragem vertical e ser@ mo
rto por um 'exit'.
Processo filho numero: 2204, est@ morto.
Processo Pai numero : 2202 , agora pode prosseguir e fazer a soma da filt
ragem vertical e filtragem horizontal.

Process returned 0 (0x0)   execution time : 0.347 s
Press ENTER to continue.
```



---

## 4

## Resultados

---

A figura um mostra a imagem a ser filtrada no nosso primeiro exemplo, e é a imagem requisitada na atividade.

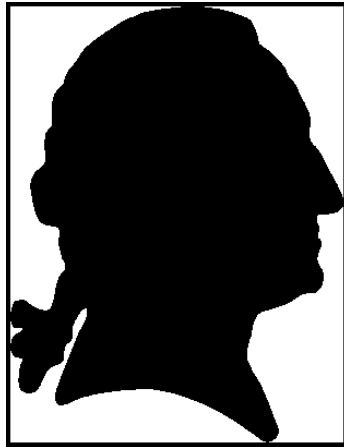


Figura 1 - Exemplo 1

Logo aqui está o resultado final, todas as bordas da imagem detectadas com sucesso pelo filtro de Prewitt.

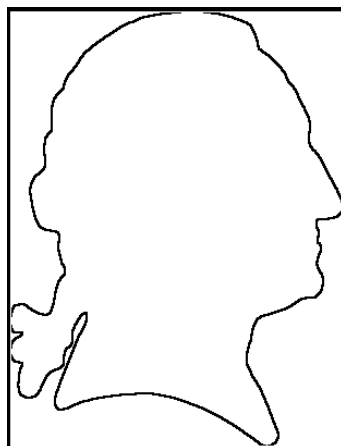
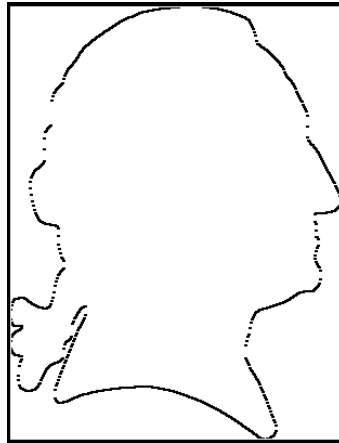


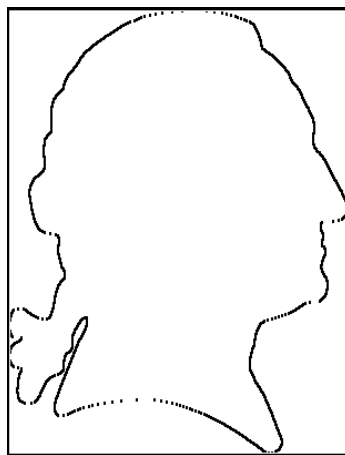
Figura 2 - Exemplo 1 com todas as bordas

Abaixo prestando atenção dos detalhes das imagens podemos ver que a imagem abaixo não pega bordas na parte horizontal, apenas na vertical, então temos nossa imagem filtrada apenas na posição vertical.



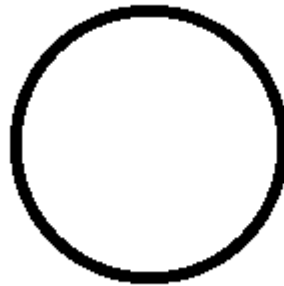
**Figura 3 - Exemplo 1 com apenas bordas verticais**

E por ultimo temos o exemplo com as bordas verticais não detectadas então temos aqui a imagem filtrada horizontalmente.



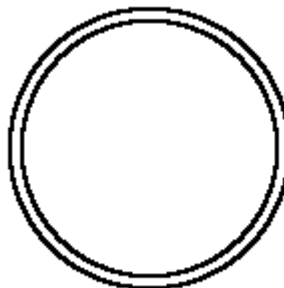
**Figura 4 - Exemplo 1 com bordas horizontais**

Repetindo o procedimento do exemplo 1, temos abaixo a imagem a ser filtrada, representando um círculo o que é interessante pois nele poderemos ver até onde o filtro entende como horizontal e até onde ele entende como vertical.



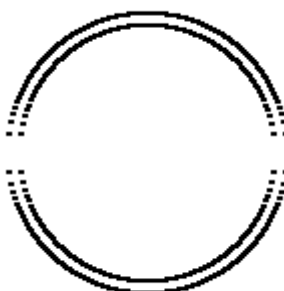
**Figura 5 - Exemplo 2**

O circulo com todas as bordas detectadas perfeitamente pelo operador de Prewitt.



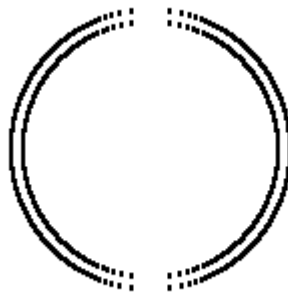
**Figura 6 - Exemplo 2 com todas as bordas**

O circulo filtrado apenas verticalmente e podemos ver que nos pontos em que o filtro deixou de entender como vertical e não detectou as bordas horizontais



**Figura 7 - Exemplo 2 com bordas vertical**

E por ultimo a imagem processada pelo filtro horizontal, onde não detecta os pontos mais acima e mais a abixo do circulo ( limite superior e inferior ), e podemos notar que treixos são considerados por ambos filtros, então treixos que são curvado, ou em diagonal podem ser dectados por qualquer um dos operadores do filtro de Prewiit.



**Figura 8 - Exemplo 2 com bordas horizontal**

---

## **Referências Bibliográficas**

---

- [1] Teoria: Processamento de Imagens  
(<http://www.dpi.inpe.br/spring/teoria/filtrage/filtragem.htm>)
- [2] <http://www2.ic.uff.br/~jcarvalho/ai/lab8/index.htm>
- [3] <http://people.sc.fsu.edu/~jburkardt/data/pbma/washington.ascii.pbm>
- [4] <http://netpbm.sourceforge.net/doc/pbm.html#plainpbm>
- [5] <http://people.sc.fsu.edu/~jburkardt/data/pbma/pbma.html>