Filtragem 2D

Visão Computacional em Robótica (BLU3040)

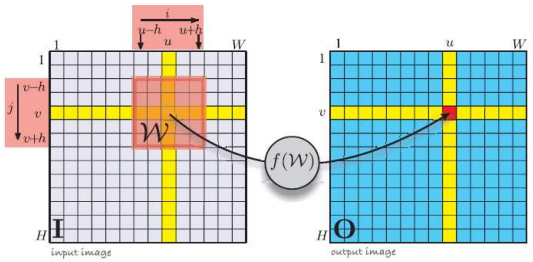
Prof. Marcos Matsuo (marcos.matsuo@ufsc.br)

Universidade Federal de Santa Catarina - UFSC

1. Operações Espaciais
2. Filtragem 2D
   - Correlação 2D
   - Convolução 2D
3. Aplicações da Filtragem Espacial
   - Suavização
   - Detecção de Borda
     - Algoritmo Básico

Operações Espaciais

Cada *pixel* (v, u) da imagem de saída $\mathbf{O}$ é resultado do processamento de todos os *pixels* em uma dada janela $\mathcal{W}$ da imagem de entrada $\mathbf{I}$.



1. Operações Espaciais

2. Filtragem 2D
   - Correlação 2D
   - Convolução 2D

3. Aplicações da Filtragem Espacial
   - Suavização
   - Detecção de Borda
     - Algoritmo Básico

Operações Espaciais

Características da janela $\mathcal{W}$

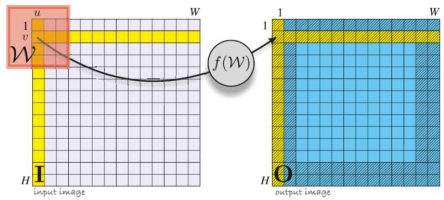
- Quadrada.
- Denota uma região de *pixels*.
- Centrada no *pixel* de entrada (v, u).
- Distância entre o *pixel* central e a borda da janela é dado por h.
- $h \in \mathbb{Z}^+$ (isto é, h é número inteiro não negativo).
- A dimensão da janela é $w \times w$, onde $w = 2h + 1$ é sempre ímpar.

Operações Espaciais

Problema: Nas regiões de borda da imagem de entrada a janela $\mathcal{W}$ abrange regiões sem *pixels*.

Soluções:

- Não calcular $f(\mathcal{W})$ nas bordas.
- Assumir que a imagem é cercada por *pixels* com valor zero.
- Replicar os *pixels* das bordas "para fora".



1. Operações Espaciais

2. Filtragem 2D
   - Correlação 2D
   - Convolução 2D

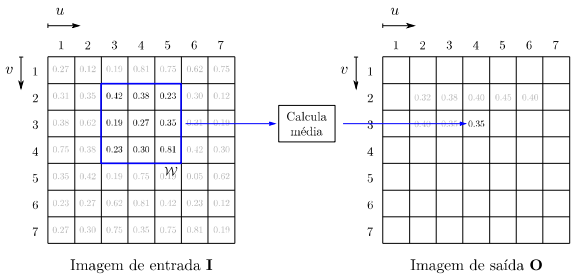
3. Aplicações da Filtragem Espacial
   - Suavização
   - Detecção de Borda
     - Algoritmo Básico

Filtragem 2D

Operações Espaciais

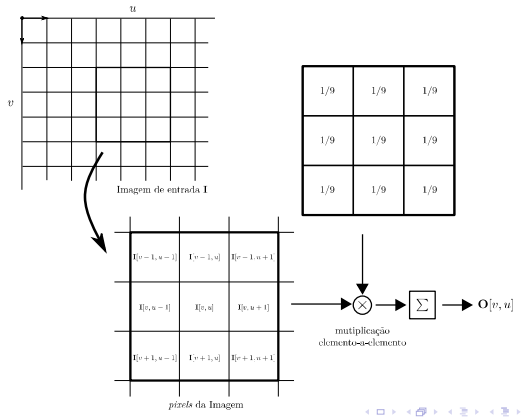
Uma operação muito importante em processamento de imagem é a **filtragem 2D**.

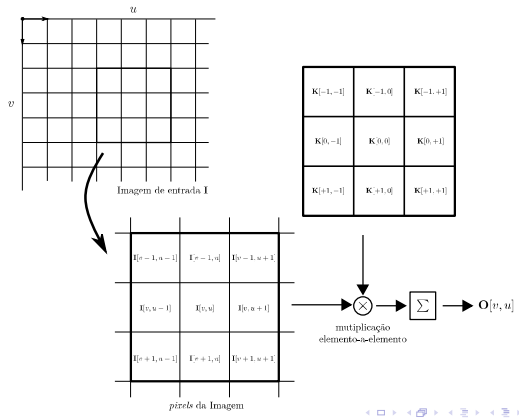
Um exemplo de filtragem 2D é a **operação de média**.



Filtragem 2D

Operações Espaciais





Matematicamente, cada *pixel* (v, u) da imagem de saída $\mathbf{O}$ é computado como

$$\mathbf{O}(v,u) = \sum_{(i,j)\in\mathcal{W}} \mathbf{I}(v+i, u+j)\mathbf{K}(i,j)$$
$$= \sum_{j=-1}^{+1}\sum_{i=-1}^{+1} \mathbf{I}(v+i, u+j)\mathbf{K}(i,j)$$

onde $\mathbf{K}(i,j) = 1/9$ para $i, j \in [-1, +1]$.

Na literatura da área a **filtragem 2D** é apresentada de duas formas distintas.

Filtragem espacial

Correlação 2D

$$\mathbf{O}[v,u] = \sum_{j=-h}^{+h}\sum_{i=-h}^{+h} \mathbf{I}[v+j, u+i]\mathbf{K}[j,i]$$

Convolução 2D

$$\mathbf{O}[v,u] = \sum_{j=-h}^{+h}\sum_{i=-h}^{+h} \mathbf{I}[v-j, u-i]\mathbf{K}[j,i]$$

- No Matlab a operação de **correlação 2D** é realizada através da função `filter2()`.
- Para a operação de **convolução 2D** tem-se a função `conv2()`.
- Os resultados numéricos da correlação e da convolução são iguais se o filtro $\mathbf{K}$ (também chamado de *kernel*) for simétrico.

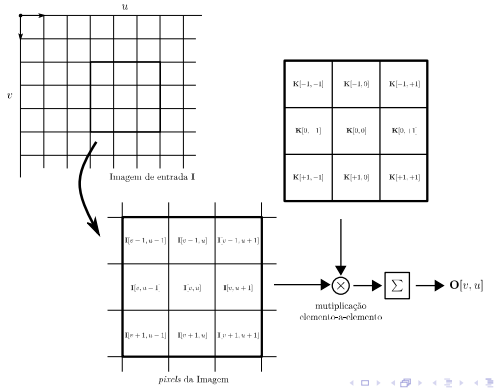
- Considere a operação de **correlação 2D**

$$\mathbf{O}(v,u) = \sum_{j=-h}^{+h}\sum_{i=-h}^{+h} \mathbf{I}(v+i, u+j)\mathbf{K}(i,j)$$

- Para fins de ilustração, assuma $h = 1$ (implicando em uma janela $\mathcal{W}$ de dimensão 3×3). Neste caso, tem-se

$$\mathbf{O}(v,u) = \sum_{j=-1}^{+1}\sum_{i=-1}^{+1} \mathbf{I}(v+i, u+j)\mathbf{K}(i,j)$$

Cálculo do pixel (v, u) da imagem de saída $\mathbf{O}$



Enunciado: Crie no Matlab um imagem de entrada $\mathbf{I}$ de dimensão 8×8, com pixels do tipo `uint8` com valores iguais a 100. Na sequência filtre a imagem utilizando um kernel $\mathbf{K}$ de dimensão 3×3, com cada elemento do kernel com valor igual a 1. Para fins de comparação, realize a filtragem a função `filter2` de duas formas. Primeiro, considere preenchimento por zero nas regiões de borda e, depois, desconsidere a região de borda.


```
1  % imagem de entrada
2  I = 100*ones(8,8,'uint8');
3
4  % kernel
5  K = ones(3,3);
6
7  % correlação 2D (com preenchimento de zeros nas regiões
8  % de borda)
9  I2 = filter2(K,I,'same');
10
11 % correlação 2D (com descarte da região de borda)
12 I3 = filter2(K,I,'valid');
```

Convolução 2D

Filtragem Espacial

- Considere a operação de **convolução 2D**

$$\mathbf{O}(v,u) = \sum_{j=-h}^{+h} \sum_{i=-h}^{+h} \mathbf{I}(v-i, u-j)\mathbf{K}(i,j)$$

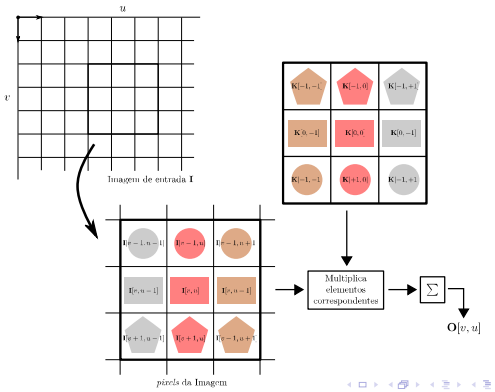
- Para fins de ilustração, assuma $h = 1$ (implicando em uma janela $\mathcal{W}$ de dimensão 3×3). Neste caso, tem-se

$$\mathbf{O}(v,u) = \sum_{j=-1}^{+1} \sum_{i=-1}^{+1} \mathbf{I}(v-i, u-j)\mathbf{K}(i,j)$$

Convolução 2D

Filtragem Espacial

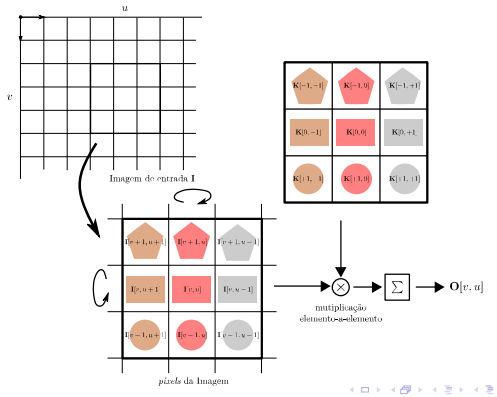
Cálculo do pixel (v,u) da imagem de saída $\mathbf{O}$



Convolução 2D

Filtragem Espacial

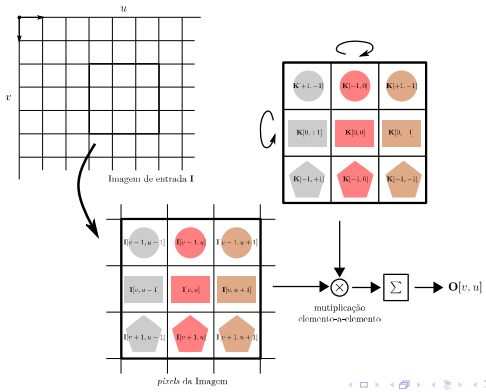
Forma alternativa 1



Convolução 2D

Filtragem Espacial

Forma alternativa 2 (preferencial)



Exemplo 2: Convolução 2D

Filtragem Espacial

Enunciado: Crie no Matlab um imagem de entrada $\mathbf{I}$ de dimensão 8×8, com pixels do tipo `uint8` com valores iguais a 100. Na sequência filtre a imagem utilizando um kernel $\mathbf{K}$ de dimensão 3×3, com cada elemento do kernel com valor igual a 1. Para fins de comparação, realize a filtragem com a função `conv2` de duas formas. Primeiro, considere preenchimento por zero nas regiões de borda e, depois, desconsidere a região de borda.

Exemplo 2: Convolução 2D

Filtragem Espacial

```
1  % imagem de entrada
2  I = 100*ones(8,8,'uint8');
3
4  % kernel
5  K = ones(3,3);
6
7  % convolução 2D (com preenchimento de zeros nas regiões
8  % de borda)
9  I2 = conv2(I,K,'same');
10
11 % convolução 2D (com descarte da região de borda)
12 I3 = conv2(I,K,'valid');
```


1. Operações Espaciais
2. Filtragem 2D
   - Correlação 2D
   - Convolução 2D
3. Aplicações da Filtragem Espacial
   - Suavização
   - Detecção de Borda
     - Algoritmo Básico

Aplicações

Filtragem Espacial

- Alterando-se os valores dos coeficientes do filtro $\mathbf{K}$, tem-se diferentes operações de filtragem.
- Exemplos de aplicações envolvendo filtragem:
  1. Suavização/redução de ruídos.
     - Filtro normalizado.
     - Filtro gaussiano.
  2. Detecção de borda.
     - Algoritmo básico.
     - Algoritmo de Canny.

Suavização

Aplicações da Filtragem Espacial

- Os filtros utilizados para suavização são também conhecidos como filtros passa-baixa.
- Reduzem os componentes de alta frequência.
- Os filtros utilizados realizam a média (aritmética ou ponderada) dos *pixels* dentro da janela $\mathcal{W}$.
- Quanto maior a janela, maior o efeito de suavização.

Suavização

Aplicações da Filtragem Espacial

Filtro Normalizado

$$\mathbf{K}[j,i] = \frac{1}{w^2}$$

$\forall j, i \in [-h, h]$, com $w = 2h + 1$ denotando a largura (ou altura) da janela quadrada $\mathcal{W}$.

Exemplos

$$\mathbf{K} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \qquad \mathbf{K} = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

Suavização

Aplicações da Filtragem Espacial

Filtro normalizado, $w = 3$

Imagem original

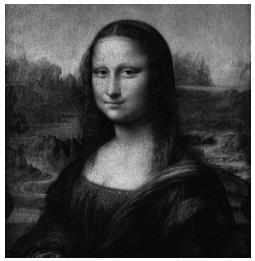


Imagem filtrada

Suavização

Aplicações da Filtragem Espacial

Filtro normalizado, $w = 5$

Imagem original

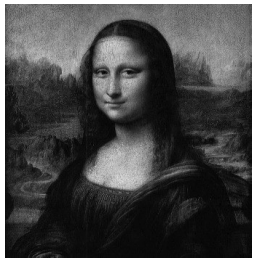


Imagem filtrada

Exemplo 3: filtragem de imagem

Aplicações da Filtragem Espacial

`Enunciado`: construa um código no Matlab para filtrar a imagem `castle.jpg` (disponível no Moodle e mostrada abaixo). Como kernel utilize um filtro normalizado com dimensão 4×4.

Exemplo 3: filtragem de imagem

Aplicações da Filtragem Espacial

Solução 1

```
I = imread('castle.jpg');
figure; imshow(I);

% kernel normalizado
N = 4;
K = 1/(N^2) * ones(N);

% filtragem 2D
I2 = filter2(K,I,'same');

figure; imagesc(I2);
colormap gray;
colorbar;
```

Exemplo 3: filtragem de imagem

Aplicações da Filtragem Espacial

Solução 2

```
I = imread('castle.jpg');
figure; imshow(I);

% kernel normalizado
N = 4;
K = 1/(N^2) * ones(N);

% filtragem 2D
I2 = filter2(K,I,'same');

% converte valores para uint8
I2 = uint8(I2);

figure; imshow(I2);
```

Exemplo 3: filtragem de imagem

Aplicações da Filtragem Espacial

Solução 3

```
I = imread('castle.jpg'); figure; imshow(I);

% kernel normalizado
N = 4;
K = 1/(N^2) * ones(N);

% filtragem 2D
I2 = filter2(K,I,'same');

% normalizada resultado pelo máximo valor para forçar
% os valores dos pixels entre 0 e 1;
I2 = I2/max(I2(:));

figure; imshow(I2);
```

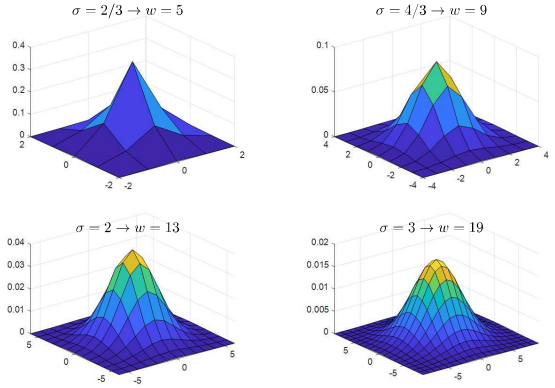
Filtro Gaussiano

$$\mathbf{K}[j,i] = \frac{1}{2\pi\sigma^2} e^{-\frac{(j^2+i^2)}{2\sigma^2}} \quad (1)$$

quanto maior σ, menos concentrados em torno do centro estão os coeficientes. Regra geral $h = 3\sigma$, ou seja, $w = 6\sigma + 1$.

Exemplo: $\sigma = 2/3 \rightarrow h = 2$, $w = 5$

$$\mathbf{K} = \begin{bmatrix} 0 & 0.0013 & 0.0040 & 0.0013 & 0 \\ 0.0013 & 0.0377 & 0.1163 & 0.0377 & 0.0013 \\ 0.0040 & 0.1163 & 0.3581 & 0.1163 & 0.0040 \\ 0.0013 & 0.0377 & 0.1163 & 0.0377 & 0.0013 \\ 0 & 0.0013 & 0.0040 & 0.0013 & 0 \end{bmatrix}$$



Filtro gaussiano, $\sigma = 2/3 \rightarrow w = 5$

Imagem original

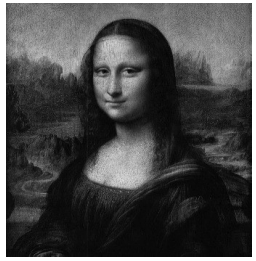
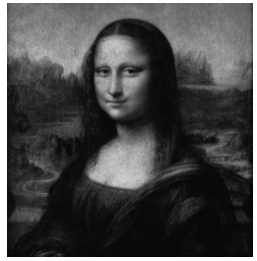


Imagem filtrada



Filtro gaussiano, $\sigma = 4/3 \rightarrow w = 9$

Imagem original

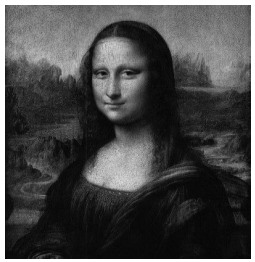
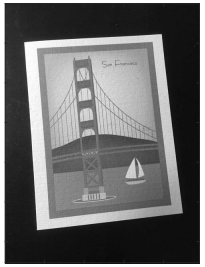


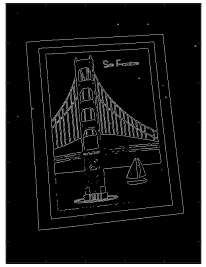
Imagem filtrada

Enunciado: construa um código no Matlab para filtrar a imagem `castle.jpg` (disponível no Moodle e mostrada abaixo). Utilize um kernel gaussiano com largura $w = 5$ e $\sigma = (w-1)/6$.

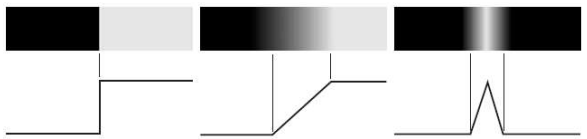

```
I = imread('castle.jpg');
figure; imshow(I);

% kernel gaussiano
w = 5;
sig = (w-1)/6;
K = fspecial('gaussian',w,sig);

% filtragem 2D
I2 = filter2(K,I,'same');
I2 = uint8(I2);

figure; imshow(I2);
```


Detecção de borda é uma abordagem utilizada mais frequentemente para segmentar imagens baseadas em mudanças abruptas de intensidade.

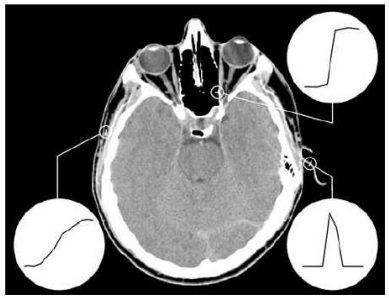




Modelos de bordas são classificadas de acordo com o perfil de variação da intensidade dos *pixels*.

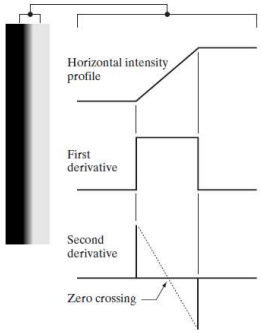


É importante destacar que os três modelos de bordas apresentados no *slide* anterior, são de fato observados em imagens práticas.

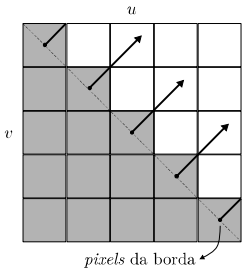


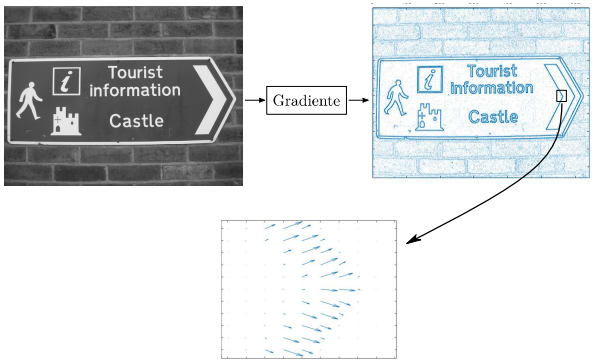
Observações:

- Bordas são carecterizadas por variações na intensidade dos *pixels*.
- Assim, o cáculo das derivadas de 1ª e 2ª ordem da imagem pode ser útil para detecção de bordas.



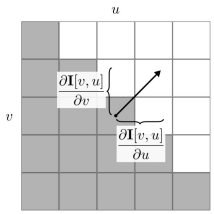
- Baseado no cálculo do gradiente da imagem.
- Para cada *pixel* determina-se um vetor indicando a direção e magnitude da maior variação da intensidade dos *pixels*.





Para cada *pixel* (v, u) da imagem $\mathbf{I}$, o vetor de gradiente é dado por

$$\nabla \mathbf{I}[v,u] = \begin{bmatrix} \dfrac{\partial \mathbf{I}[v,u]}{\partial u} \\ \dfrac{\partial \mathbf{I}[v,u]}{\partial v} \end{bmatrix}$$



onde $\partial \mathbf{I}[v,u]/\partial u$ denota a taxa de variação da intensidade do *pixel* no eixo horizontal u e $\partial \mathbf{I}[v,u]/\partial v$ representa a taxa de variação em relação ao eixo vertical v.

- Computando-se as derivadas parciais para todos os *pixels* (v, u) da imagem $\mathbf{I}$, tem-se que $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$ passam a ser imagens com a mesma dimensão de $\mathbf{I}$.
- Ou seja, o gradiente de $\mathbf{I}$ pode ser composto pelas imagens $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$.

Como calcular $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$?

1ª Opção: Para todo (v, u) calcula-se

$$\frac{\partial \mathbf{I}[v,u]}{\partial u} = \mathbf{I}[v, u+1] - \mathbf{I}[v,u]$$

$$\frac{\partial \mathbf{I}[v,u]}{\partial v} = \mathbf{I}[v+1, u] - \mathbf{I}[v,u]$$

Como calcular $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$?

2ª Opção: Para todo (v, u) calcula-se

$$\frac{\partial \mathbf{I}[v,u]}{\partial u} = \frac{1}{2}\left(\mathbf{I}[v, u+1] - \mathbf{I}[v, u-1]\right)$$

$$\frac{\partial \mathbf{I}[v,u]}{\partial v} = \frac{1}{2}\left(\mathbf{I}[v+1, u] - \mathbf{I}[v-1, u]\right)$$

Como calcular $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$?

3ª Opção:

$$\frac{\partial \mathbf{I}}{\partial u} = \mathbf{I} * \mathbf{K}_u$$

onde $*$ representa a operação de convolução e

$$\mathbf{K}_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

é conhecido como filtro de Sobel.

Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial

Como calcular $\partial \mathbf{I}/\partial u$ e $\partial \mathbf{I}/\partial v$?

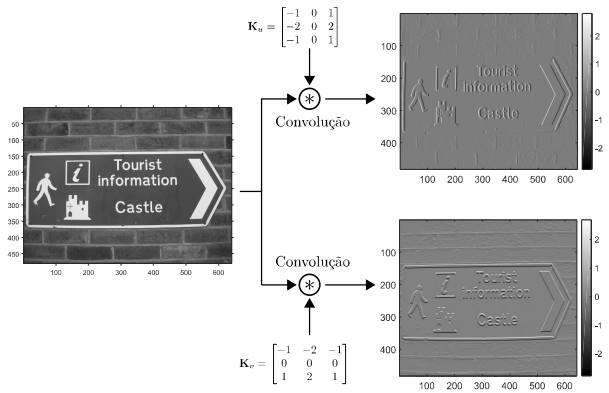
3ª Opção:

$$\frac{\partial \mathbf{I}}{\partial v} = \mathbf{I} * \mathbf{K}_v$$

onde

$$\mathbf{K}_v = \mathbf{K}_u^{\mathsf{T}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial



Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial

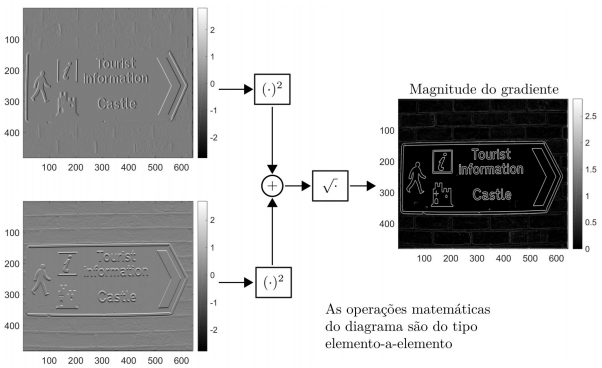
Note que o gradiente no *pixel* (v, u) também pode ser representado por da magnitude $\mathbf{M}[v, u]$ e fase $\alpha[v, u]$, definidas como

$$\mathbf{M}[v,u] = \sqrt{\left(\frac{\partial \mathbf{I}[v,u]}{\partial u}\right)^2 + \left(\frac{\partial \mathbf{I}[v,u]}{\partial v}\right)^2}$$

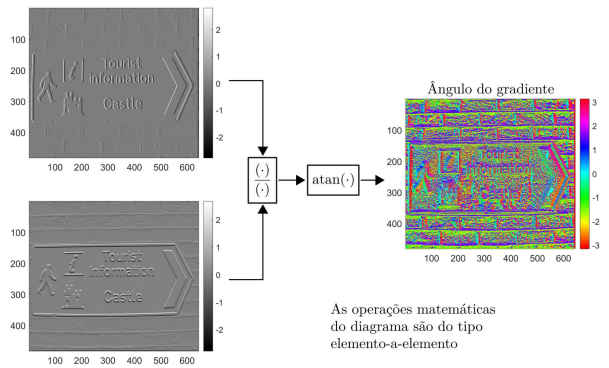
e

$$\alpha[v,u] = \operatorname{atan}\left(\frac{\partial \mathbf{I}[v,u]/\partial v}{\partial \mathbf{I}[v,u]/\partial u}\right)$$

Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial



Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial



Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial

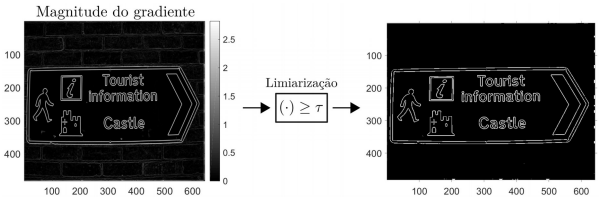
Note que na imagem de magnitude do gradiente (isto é, $\mathbf{M}[v, u]$), os *pixels* de maior valor tendem a corresponder as bordas da imagem.

Assim, uma imagem de borda pode ser obtida a partir da operação de limiarização, ou seja,

$$\mathbf{I}_b[v,u] = \begin{cases} 1, & \mathbf{M}[v,u] \geq \tau \\ 0, & \text{caso contrário} \end{cases}$$

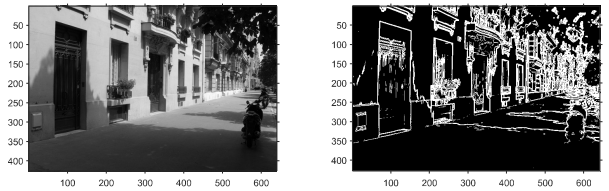
onde $\mathbf{I}_b$ é uma imagem binária com *pixels* de borda em branco e τ é um limiar de comparação.

Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial



Algoritmo Básico
Detecção de Borda - Aplicações da Filtragem Espacial

Se a imagem $\mathbf{I}$ tiver muitas variações de textura, o algoritmo básico apresentado pode gerar uma imagem de borda bastante ruidosa.



Em casos como o anterior, geralmente, a imagem $\mathbf{I}$ é primeiramente filtrada por um kernel de suavização para então ter suas bordas determinadas.