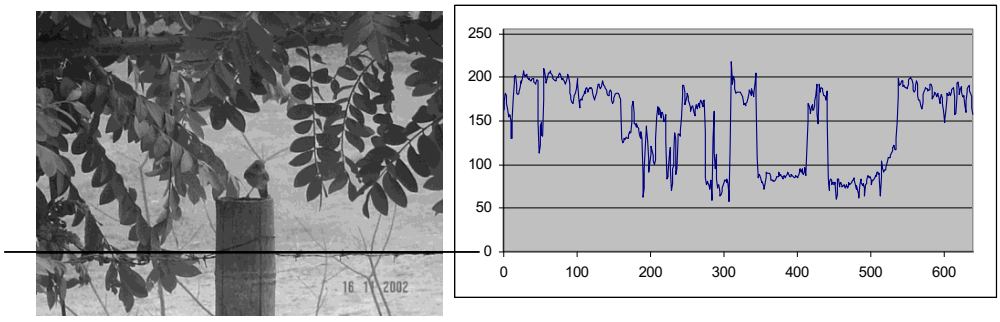
Imagem Digital

Depois da cor, o segundo elemento fundamental da Computação Gráfica é a Imagem Digital. Mas, antes de discutirmos a versão digital, vamos conceituar melhor o que entendemos por imagem. No sentido comum, imagem é uma representação gráfica de objetos que nos cercam ou que criamos. No contexto deste capítulo, a imagem que nos interessa é aquela produzida por computador que se assemelha a uma foto – ou seja, é uma região retangular do espaço na qual em cada ponto percebemos uma cor ou uma intensidade de cinza.

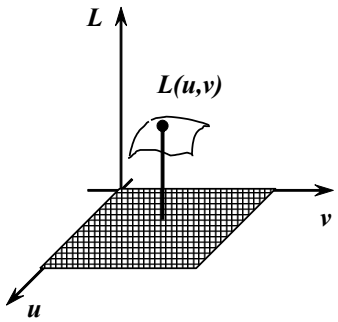
Por questão de simplicidade, vamos inicialmente tratar do modelo das fotos monocromáticas ou, como se diz comumente, fotos em preto e branco. A figura abaixo mostra uma foto de um passarinho num poste com uma linha marcada. O valor da intensidade de cinza, medida de 0 a 255, ao longo desta linha está mostrada no lado direito da figura.



(a) **(b)**

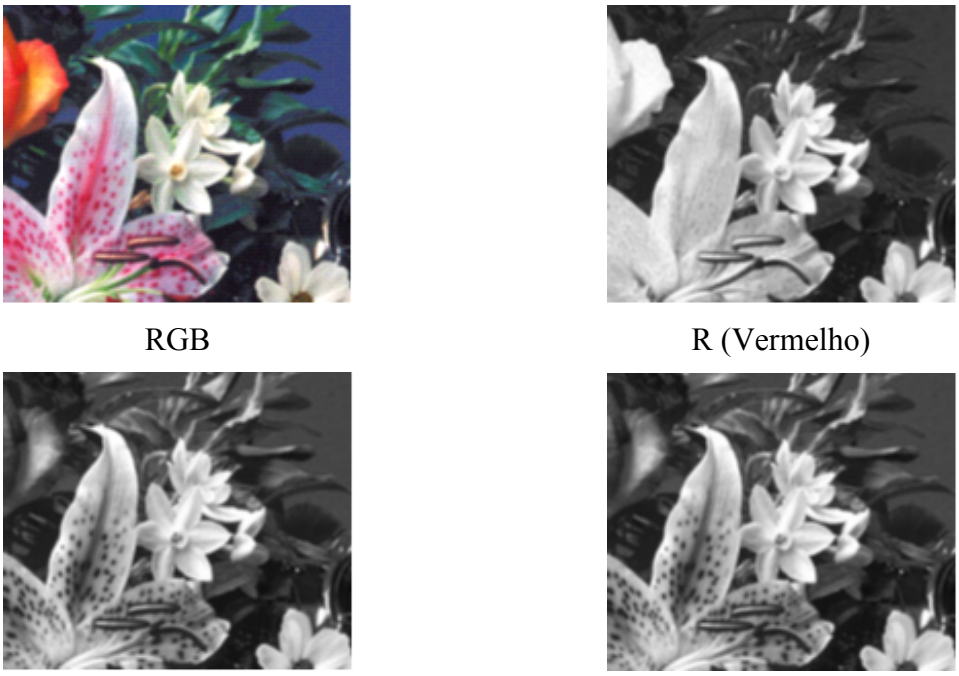
Fig. 3.1 – Valor da intensidade de cinza ao longo de uma linha da foto.

Se considerarmos que, para cada valor no interior do retângulo da foto, temos uma intensidade de cinza de 0 a 255, podemos, abstratamente, pensar em uma imagem monocromática como sendo uma função do R^2 em R ou uma superfície no R^3, como ilustra a figura abaixo.



Modelo matemático para uma imagem: função do R^2 em R.

Se a foto for colorida, capturada num sistema tipo o sRGB discutido no capítulo de cores, não podemos medir a cor por apenas um número real, são necessários mais valores. Uma maneira simples de evoluirmos o modelo monocromático descrito acima consiste em considerarmos, em vez de uma, três funções distintas: uma para cada canal RGB. A figura abaixo ilustra esta decomposição, na qual para cada canal o valor da intensidade é convertido numa escala de preto (zero) para branco (255 ou 1.0, se a imagem for armazenada em ponto flutuante).



RGB R (Vermelho)

G (Verde) B (Azul)

Decomposição de uma imagem colorida em três canais: RGB.

Naturalmente, também podemos pensar na imagem colorida como sendo uma função que atribui a cada ponto do domínio retangular um ponto no espaço de cores. Como visto no capítulo de cores, este espaço pode ser RGB, RGBA, CMY, CMYK, etc. A figura abaixo ilustra esta idéia para o espaço RGB.

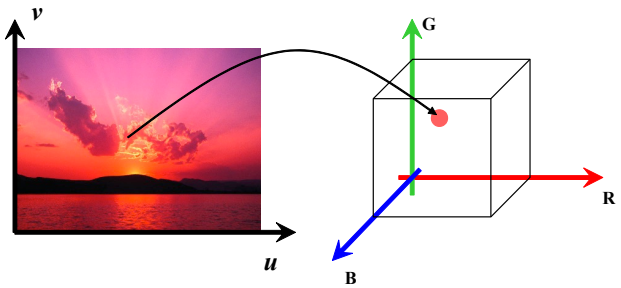


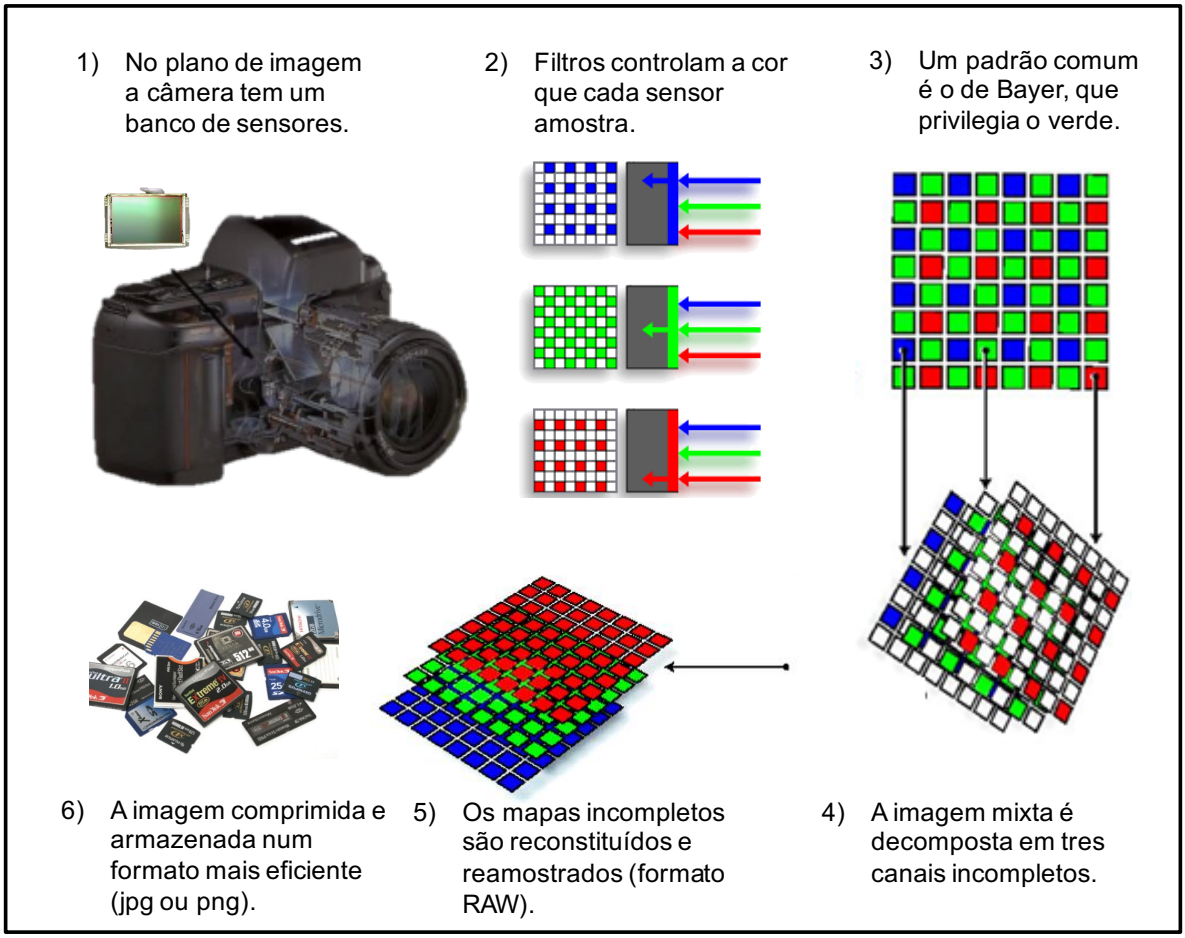
Imagem colorida como um mapeamento do R^2 em R^3.

A conveniência de escolhermos este ou aquele modelo para imagens coloridas depende da aplicação e é, de certa forma, uma decisão arbitrária. Para efeito de introdução ao assunto

de processamentos de imagem, o tratamento de cada canal de cor em separado, R, G, B, ou L (luminância) facilita.

Processos envolvidos na captura e reprodução de imagens

A natureza de uma imagem digital pode ser melhor compreendida se analisamos o seu processo de aquisição. Numa máquina fotográfica digital, um sistema de lentes projeta a cena 3D no plano correspondente ao filme. O assunto de como esta projeção é modelada é deixado para os capítulos de síntese de imagens ou *rendering*. Este capítulo foca nos processos que ocorrem no plano onde a imagem é formada ilustrados na figura abaixo.



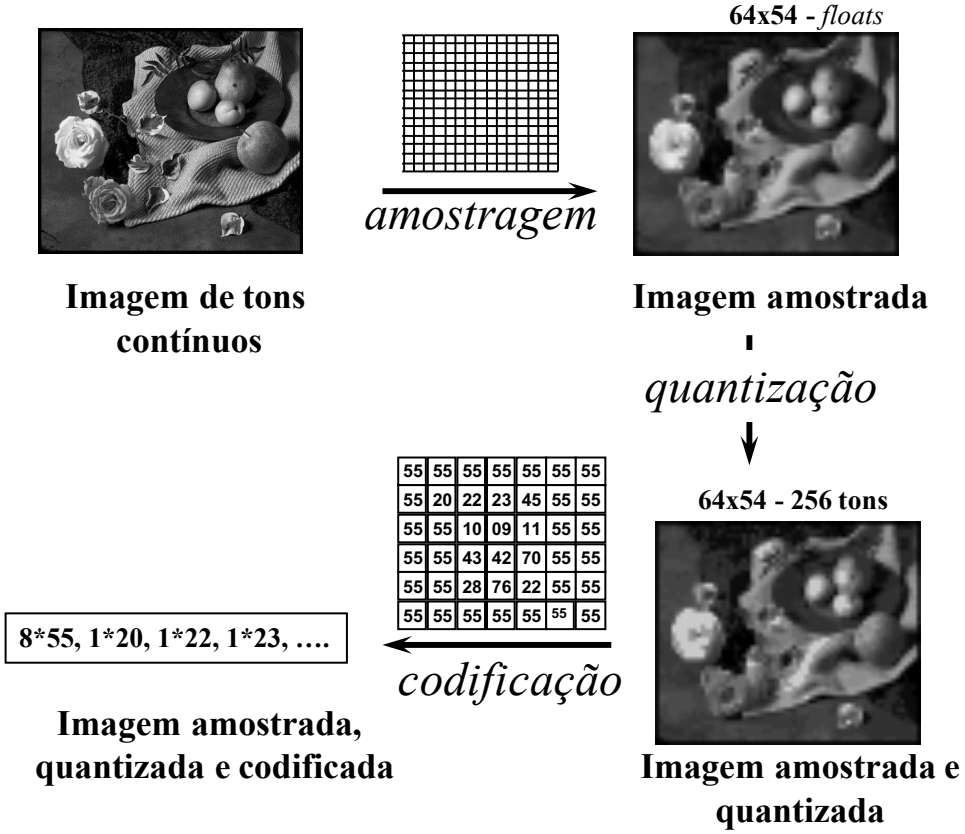
Aquisição de uma imagem numa câmera fotográfica.

No passo 4 da figura acima, os canais vermelho, verde e azul de uma imagem são amostrados no plano da imagem em pontos esparsos da matriz de pixels. Para obter os valores em todos os pixels é necessário reconstruímos a função e reamostrá-la nos pontos faltantes.

Os valores de radiância luminosa lidos são mapeados em valores de intensidade e armazenados como matrizes de números. Para que este armazenamento ocupe apenas um byte por canal e por pixel, os valores reais das intensidades precisam ser quantizados em 256 níveis. Mesmo com esta redução as imagens ocupam muita memória e por isto são

comumente armazenadas em formatos mais comprimidos como o jpeg e ou png. Por exemplo, no formato RAW uma imagem em FullHD, 1920×1080, ocupa quase 6.220.800 bytes. No formato jpg estes 5,9 MB são reduzidos para aproximadamente 372KB.

Três classes de problemas importantes ocorrem na captura de uma imagem: (1) **amostragem e reconstrução**, (2) **quantização** e (3) **codificação**. Uma visão simplificada do processo de captura da imagem é mostrada na figura abaixo.

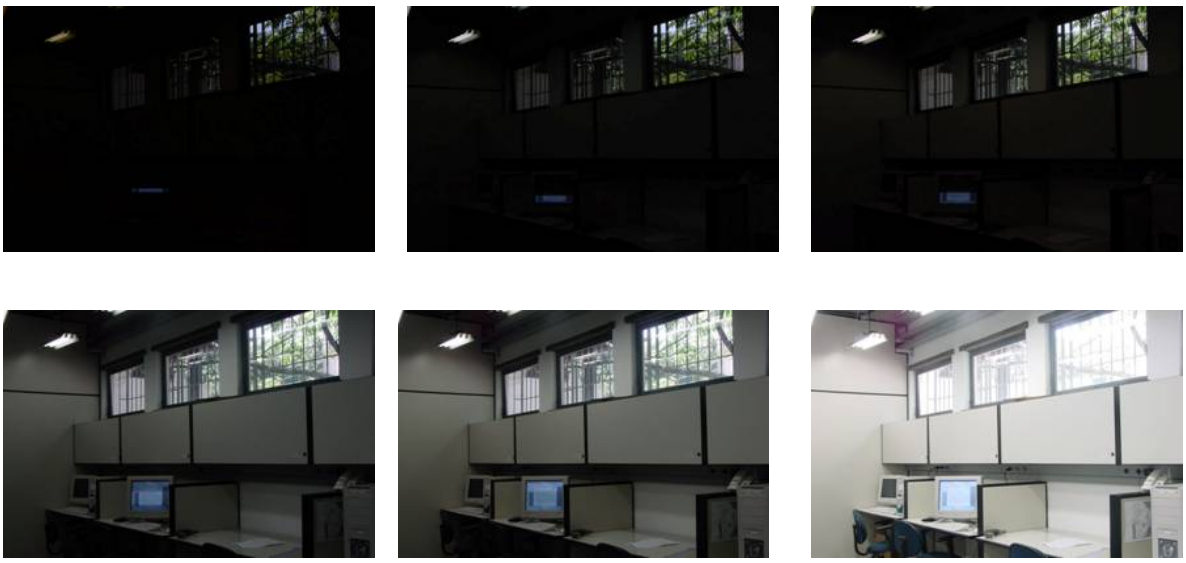


Amostragem, reconstrução e quantização na captura de uma imagem.

O valor amostrado pode ser adquirido como um tipo ponto flutuante ou inteiro longo, que requerem mais *bits* do que podemos dispor tanto na memória quanto em banda no canal de transmissão e por isto comumente são armazenados em 1 byte que pode representar apenas 256 níveis distintos.

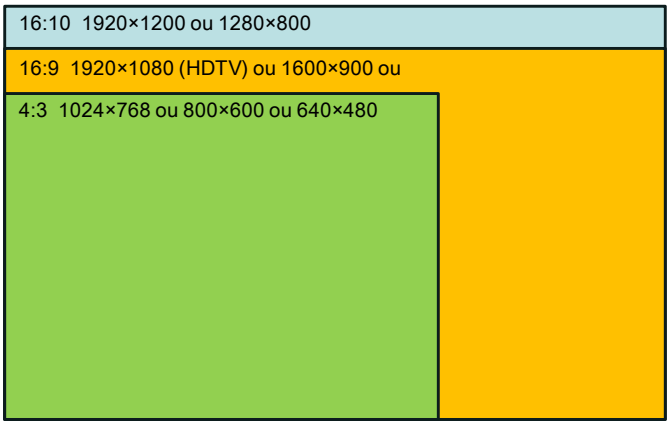
O processo de captura da luz, entretanto, tem suas peculiaridades. A escala linear não é adequada para transformar valores de luminância em tons de um canal de cor. A figura abaixo mostra uma função tipo S, comumente utilizada na transformação de radiância (quantidade de luz) em tons de um determinado canal de cor, denominado Mapeamento de Tons (*Tone Mapping*).

O estudo aprofundado de Mapeamento de Tons foge ao escopo deste capítulo, mas é importante termos em mente que a medida de RGB fornecida pela câmera não é uma medida de quantidade de luz, mas sim o resultado de um processamento que ocorre dentro da própria câmera digital. A figura abaixo mostra seis fotos da mesma cena com diferentes tempos de exposição. Note que as primeiras, no canto superior esquerdo, a vegetação é nítida enquanto nas últimas o que é nítido é a cadeira da última mesa. A limitação de 256 níveis impede que a câmera consiga capturar na mesma foto as duas regiões com luminosidades tão diferentes.



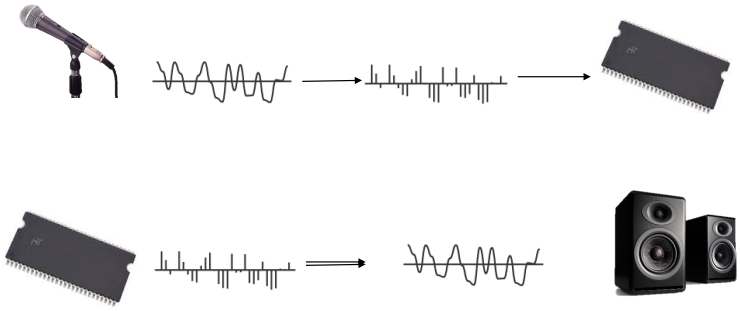
Seis fotos da mesma cena com variação do tempo de abertura da câmera.

Dois assuntos importantes no estudo de imagens digitais são: (a) resolução $w \times h$, número de amostras largura e altura, respectivamente, e (b) razão de lados $w:h$, (*aspect ratio*). Em princípio eles podem ser quaisquer, mas, a produção industrial de equipamentos precisa de padrões. Valores típicos destes resolução e razão de lados são ilustrados na figura abaixo. Mudanças de resolução são mais facilmente acomodadas que mudanças de razão de lados. Um filme produzido em uma razão de lados não consegue ser reproduzido em um equipamento com outra sem uma certa perda.



Razões de lados e resoluções típicas.

Os processos de **amostragem e reconstrução**, **quantização** e **codificação** são estudados na Teoria do Sinais. Para compreendermos melhor estes processos vamos iniciar o estudo com sinais 1D que são mais simples. A partir dos resultados em 1D podemos expandir este estudo para imagens (2D). É como se estivéssemos estudando estes conceitos com base na captura e reprodução de som ilustrada na figura abaixo.

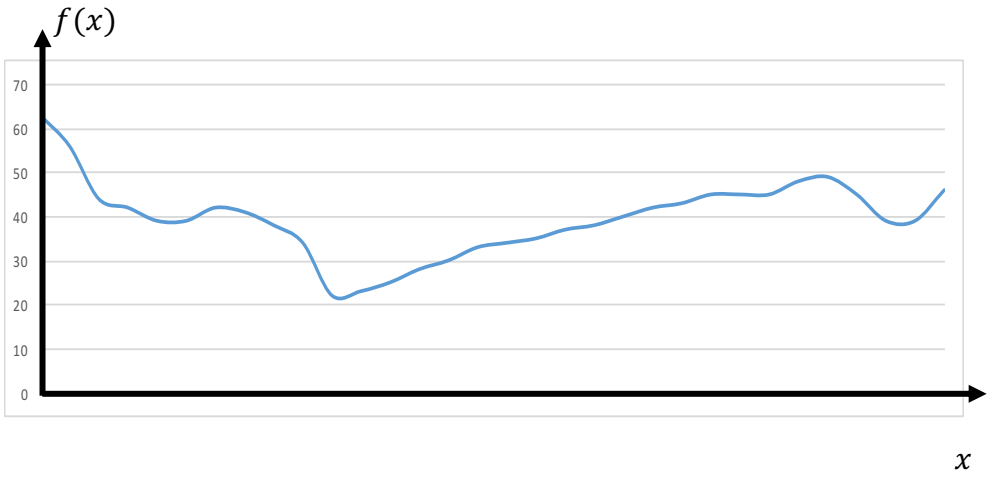


Captura e reprodução do som.

Amostragem e reconstrução de sinais 1D

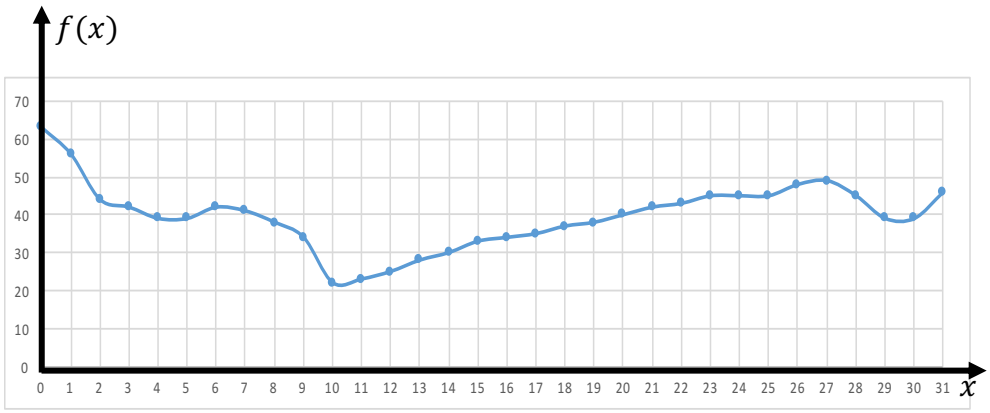
A figura abaixo ilustra um sinal 1D onde o domínio, *x,* é o tempo no caso de estarmos amostrando um som. No caso de estarmos amostrando uma imagem podemos pensar em *x* como sendo um eixo de espaço numa linha da matriz de cores.

A amostragem é normalmente feita dividindo o domínio, eixo *x* em uma partição uniforme e dentro de cada intervalo desta partição escolher um valor para representar a função naquele trecho.



Sinal 1D.

A figura abaixo ilustra a partição do eixo x em 32 células. A figura também mostra as amostras, que no caso, são os valores instantâneos da função no início do intervalo.



Amostragem: partição e escolha de representante.

Uma maneira mais precisa de modelar matematicamente o processo de amostragem consiste em escrever a função $f(x)$ no intervalo de $[0, n)$ em numa base de funções $h_i(x)$ que representem a sensibilidade do sensor em cada ponto.

Ou seja:

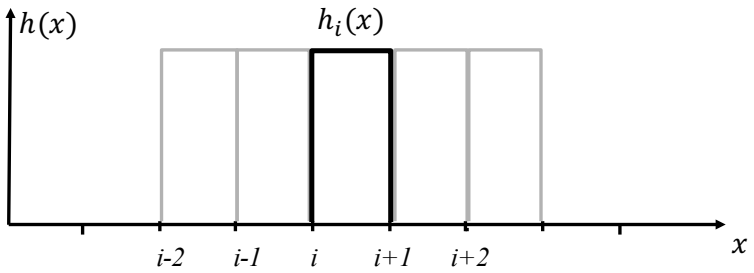
$$f(x) = \sum_i f_i \, h_i(x) \qquad \text{eq. (1)}$$

Uma possibilidade simples para $h_i(x)$ seria a função caixa (*box*) no intervalo $[i, i+1)$:

$$h_i(x) = \begin{cases} 0, se\ x < i \\ 1, se\ i \leq x < i+1 \\ 0, se\ x > i \end{cases}$$

Estas funções $h_i(x)$ formam uma base para escrevermos, de forma aproximada, todas as funções que estejam no intervalo de $[0, n)$ quando ele é particionado em células de tamanho um.

A figura abaixo ilustra um conjunto de funções $h_i(x)$.



Funções caixa cobrindo um intervalo particionado.

Para calcularmos os coeficientes f_i, basta multiplicamos $f(x)$ por $h_k(x)$ e integramos o produto no domínio, ou seja:

$$\int h_k(x) f(x) dx = \int h_k(x) \left(\sum_i f_i \, h_i(x) \right) dx = \sum_i f_i \int h_k(x) \, h_i(x) dx$$

Como, as funções caixa são ortogonais no sentido de:

$$\int h_k(x) \, h_i(x) dx = \begin{cases} 0, se \; i \neq j \\ i, se \; i = j \end{cases}$$

a determinação dos coeficientes f_k é trivial:

$$f_k = \int f(x) \, h_k(x) \, dx = \int_k^{k+1} f(x) dx$$

Ou seja, os coeficientes f_k, que escrevem a função contínua de forma discreta das funções caixa de base um é simplesmente a área da curva no intervalo $[k, k+1)$. Desta forma dizemos que:

$$(f_o, f_1, f_2, \ldots, f_{n-2}, f_{n-1})$$

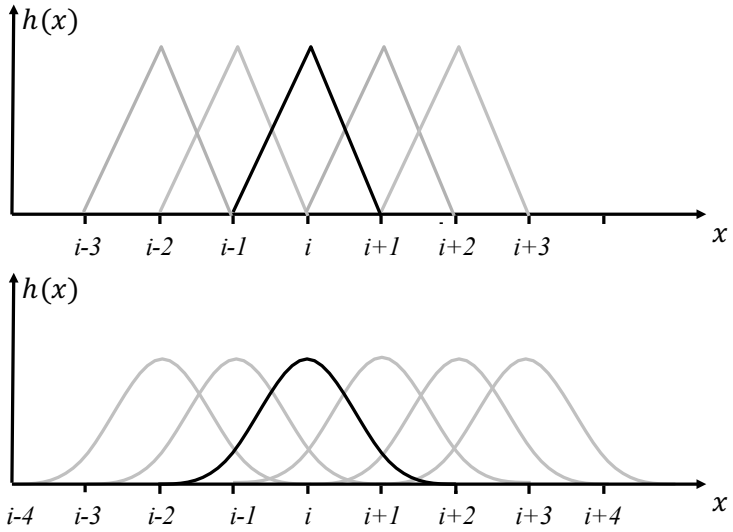
é a representação discreta de $f(x)$ no intervalo $[0, n)$ com base na partição uniforme unitária e nas funções de amostragem caixa. Note também que:

$$\sum_{i=0}^{n-1} f_i = \int_0^n f(x) dx$$

Ou seja, a forma discreta preserva a área abaixo da curva do sinal. Note entretanto que neste caso f_i não é, necessariamente, o valor instantâneo da função no início do intervalo.

Note que na derivação apresentada acima, usamos o resultado da Álgebra Linear de como escrever um vetor qualquer numa base de funções ortonormais.

Outras funções que também são comumente utilizadas para $h_i\ (x)$ são as lineares e cúbicas ilustradas abaixo.



Funções lineares e cúbicas cobrindo um intervalo particionado.

Essas funções, entretanto, não são ortogonais no sentido de

$$\int h_k\ (x)\ h_i\ (x)dx = \begin{cases} 0, se\ i \neq j \\ i, se\ i = j \end{cases}$$

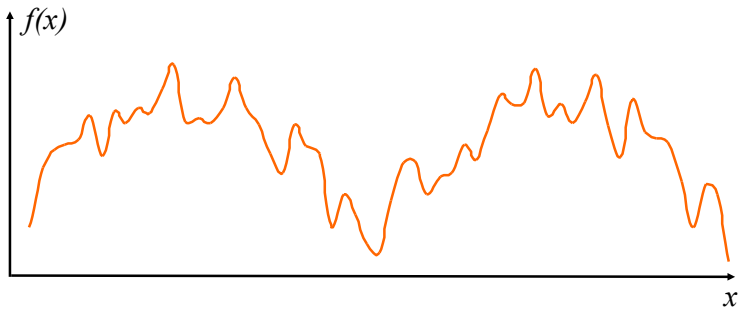
e o cálculo dos coeficientes f_i é um pouco mais envolvente pois recai num sistema de equações linear. Este sistema, entretanto, é simples de ser resolvido.

Se desejarmos mudar o número de amostra do domínio (aumentar ou diminuir a resolução) podemos reconstruir a função contínua através da eq. (1)

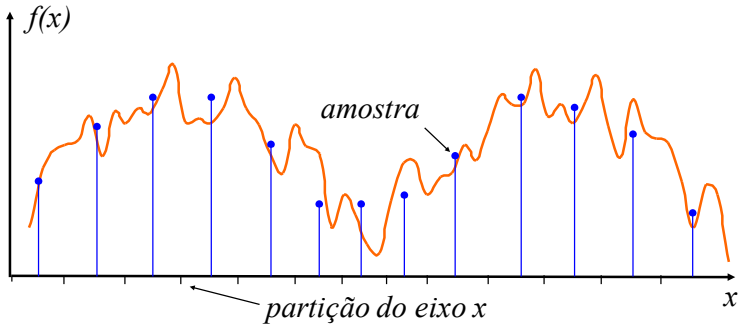
$$\tilde{f}(x) = \sum_{i=0}^{n-1} f_i h_i(x)$$

e reamostradá-la com a nova partição na resolução desejada.

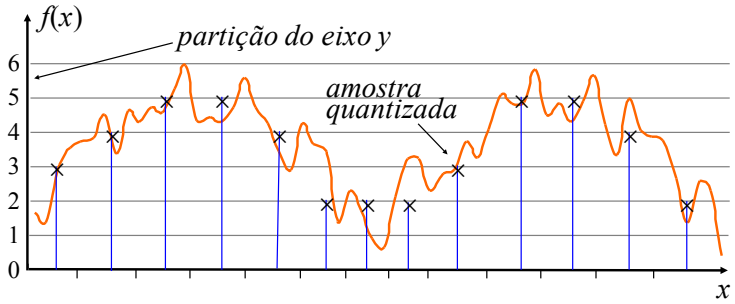
Note, entretanto que a reconstrução é apenas uma aproximação da função original. Existe um erro que pode ou não ser aceitável. As figuras abaixo ilustram o processo de amostragem e reconstrução num caso onde a função reconstituída é muito diferente da função original. Nela parte da variação da função original é perdida.



Sinal original.

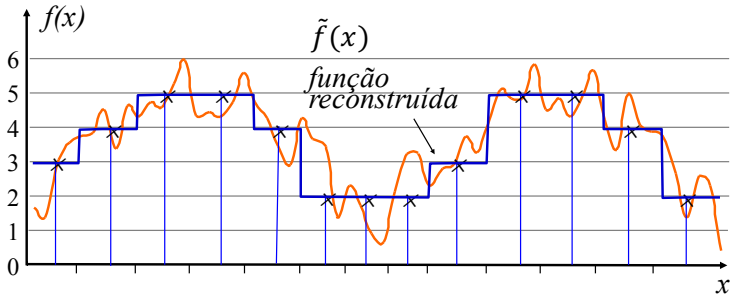


Sinal amostrado.



$f_i = (3,4,5,5,4,2,2,3,5,5,4,2)$

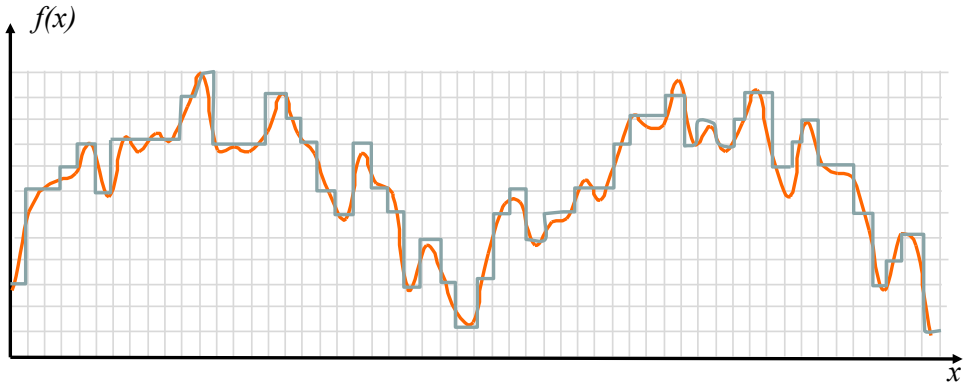
Sinal discreto: amostrado e quantizado.



Sinal reconstruído.

Reconstrução de $f(x)$, $\tilde{f}(x)$ com base dos níveis do sinal discreto f_i

Se examinarmos a reconstrução da função *f(x)* ilustrada acima, vemos que ela incorre em erros nas partes do domínio onde *f(x)* varia muito rápido. Este fato advém da baixa taxa de amostragem (número de amostra no por intervalo) nestas regiões de grande variação. Se aumentarmos a taxa de amostragem podemos capturar melhor a função como um todo. A figura abaixo ilustra a amostragem da mesma função com mais amostras (maior *n*) e com uma quantificação mais representantes. Note pelo gráfico que intuitivamente a aproximação nesta nova taxa de amostragem é bem melhor que a anterior.



Sinal com boa amostragem.

A determinação da taxa mínima de amostragem capaz de aproximar bem um sinal pode ser feita a partir do Teorema da amostragem de Nyquist–Shannon que se baseia na Transformada de Fourier. Para entendermos a idéia deste Teorema vamos composição do sinal em uma base de senos e cossenos ilustrada a seguir.

Vamos reescrever a eq. (1) com as seguintes escolhas de $h_i(x)$

$$f(x) = a_0 + \sum_{k=1}^{\infty} (a_k \cos\left(\frac{2\pi kx}{n}\right) + b_k \sin\left(\frac{2\pi kx}{n}\right))$$

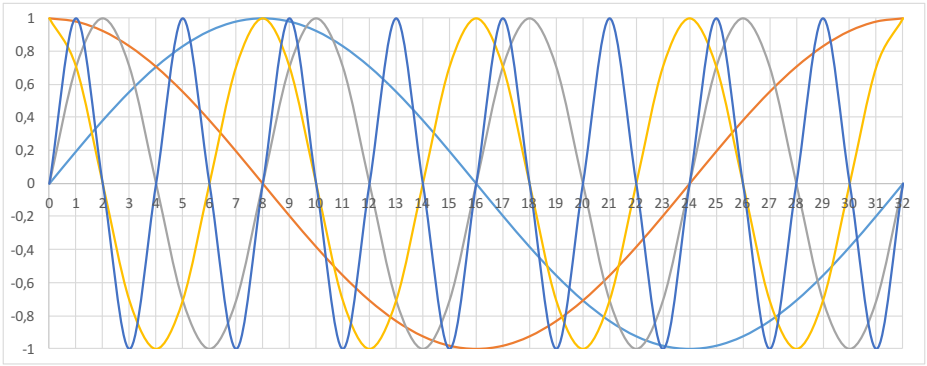
onde:

$$a_0 = \frac{1}{n}\int f(x)dx$$

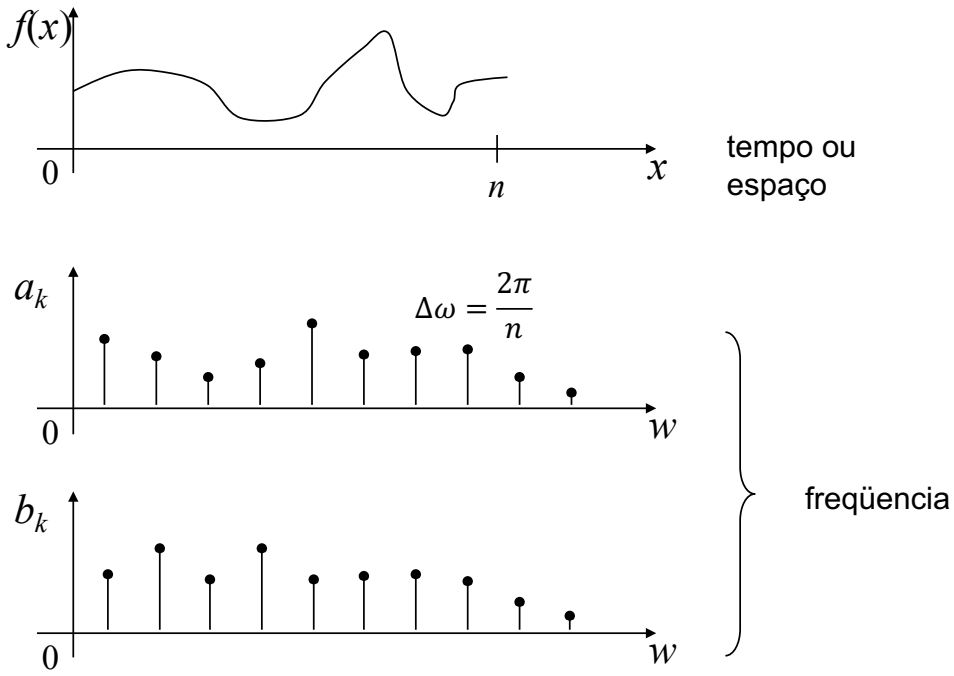
$$a_k = \frac{1}{n}\int f(x)cos\left(\frac{2\pi kx}{n}\right)dx$$

$$b_k = \frac{1}{n}\int f(x)sin\left(\frac{2\pi kx}{n}\right)dx$$

Esta série, proposta por Fourier em 1807 escreve uma função qualquer como uma combinação linear das funções seno e cosseno. Os primeiros termos da série são o seno e cosseno que tem um período, T, correspondente ao intervalo escolhido para amostrar a função, no nosso caso [0,*n*). Os demais senos e cossenos tem frequências que são múltiplas destas. Ou seja, enquanto a série se expande temos termos com frequências cada vez maiores. A figura abaixo mostra as funções básicas de seno e cosseno para um *n* igual a 32. Alguns senos e cossenos de frequências mais altas são também mostrados para ilustrar a tendência da série.



O coeficiente a_k *e* b_k da série de Fourier são denominados componentes do sinal na frequência ω_k. Note que ao definirmos o intervalo [0, *n*) da função *f*(*x*) estamos implicitamente definindo a escala de frequência em que vamos fazer a amostram, como ilustra a figura abaixo. Quanto maior o número de amostras *n*, menor a taxa de amostragem $\Delta\omega$.



Domínios de amostragem de um sinal.

As funções que representam sinais físicos tendem a ser "bem-comportadas" e, a partir de uma certa frequência, não tem mais componentes. Ou seja, a partir de um certo ω_{max} a série de Fourier não tem mais termos. O Teorema da amostragem de Nyquist–Shannon afirma que se amostrarmos o sinal com uma taxa de amostragem igual ou superior duas vezes a frequência máxima podemos reconstruir exatamente o sinal contínuo a partir das *n* amostras. Esse resultado faz a ponte entre a matemática contínua e discreta na Teoria de Sinais e, por isto, é considerado um teorema fundamental deste estudo. Uma maneira

"força bruta" de amostrarmos corretamente seria calcularmos a série de Fourier para um n bem grande e desprezarmos os coeficientes próximos de zero do final da série.

Se temos uma função já amostrada, o cálculo da série de Fourier é denominado Transformada de Fourier e pode ser feita através das aproximações indicadas abaixo:

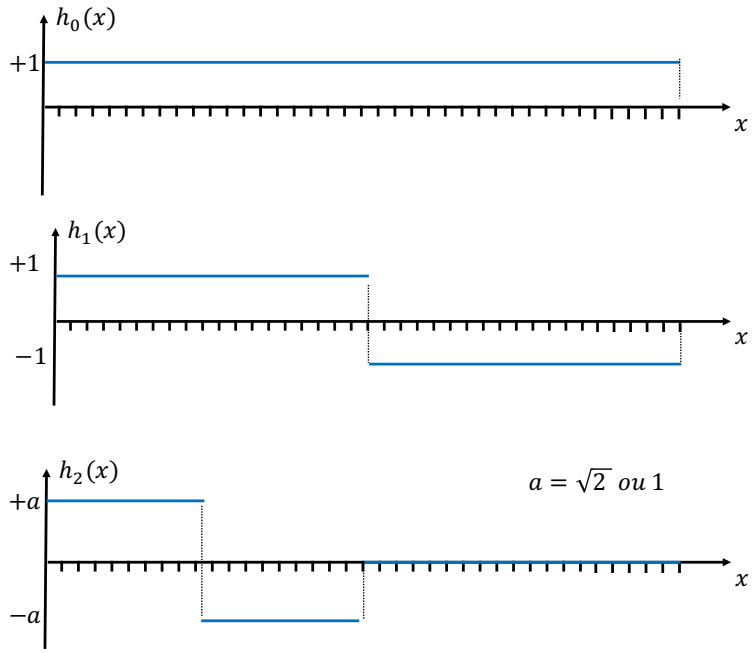
$$a_k = \frac{1}{n}\int f(x)cos\left(\frac{2\pi kx}{n}\right)dx \cong \frac{1}{n}\sum_{i=0}^{n-1} f_i cos\left(\frac{2\pi ki}{n}\right)$$

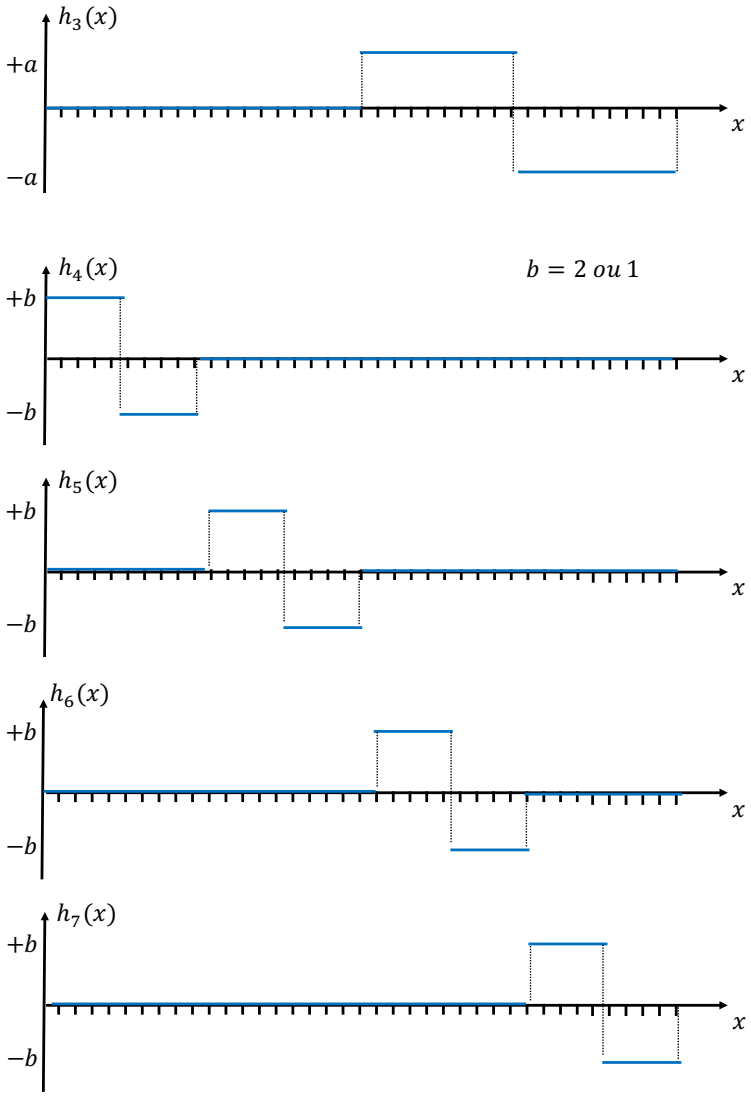
$$b_k = \frac{1}{n}\int f(x)sin\left(\frac{2\pi kx}{n}\right)dx \cong \frac{1}{n}\sum_{i=0}^{n-1} f_i sin\left(\frac{2\pi ki}{n}\right)$$

A Transformada de Fourier é a base de muitos processos da Teoria de Sinais, mas tem uma deficiência grave. Quando num sinal temporal ou espacial identificamos uma componente importante de frequência não sabemos quando (ou onde) está frequência ocorre. Isto porque os senos e cossenos cobrem todo o domínio.

As wavelets formam uma base funções que tem a propriedade de capturar não somente as frequências presentes num sinal, mas também a localidade (quando ou onde) estas frequências ocorrem. Dada a natureza introdutória destas notas vamos nos restringir a wavelt de Haar que é a mais simples delas, mas que pode dar resultados muito bons.

A idéia é novamente voltarmos a reescrever a eq. (1) com as novas escolhas de $h_i(x)$, no caso utilizando as wavelets de Haar ilustradas na figura abaixo.



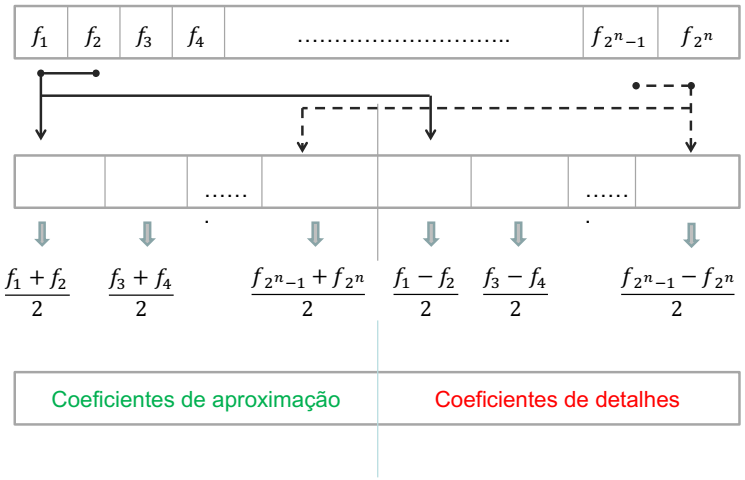


Gráficos das sete primeiras funções de Haar.

A escolha de *a* e *b* iguais a $\sqrt{2}$ e 2, respectivamente, faz com que esta base seja ortonormal, ou seja, de funções (vetores) ortogonais e unitárias, segundo o produto . Optamos pela escolha de *a* e *b* iguais a 1, para evitar o produto por um número irracional e, consequentemente, tornar a decomposição computacionalmente mais eficiente. Os vetores permanecem ortogonais, mas não tem mais norma um.

O esquema ilustrado a seguir mostra a implementação simples e eficiente da transformada de Haar a escolha de *a* e *b* iguais a 1.

Inicialmente calculamos os coeficientes de aproximação e os de detalhes da forma ilustrada na figura abaixo.

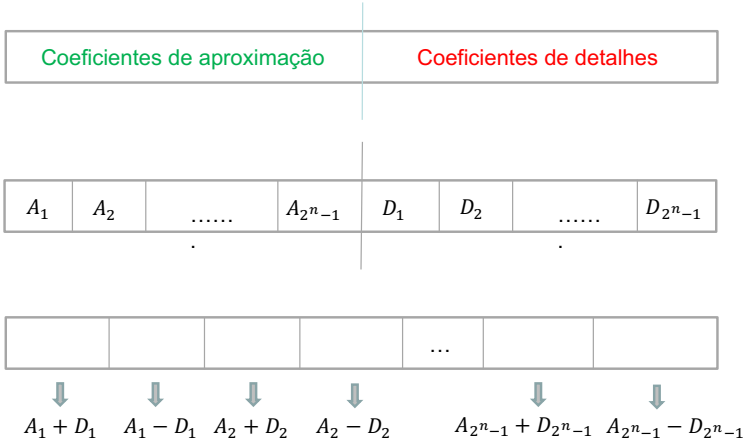


Um passo da transformada de Haar.

A seguir aplicamos recursivamente o passo acima nos coeficientes de aproximação. Os coeficientes de detalhes são deixados intactos. O algoritmo para quando chegamos a um coeficiente de aproximação apenas. O exemplo mostrado na figura abaixo ilustra o algoritmo.

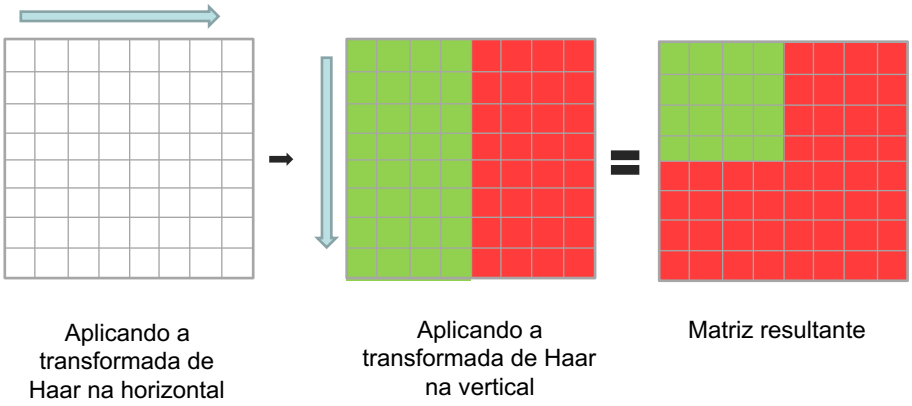
Transformada de Haar.

Não é difícil deduzir a transformada inversa. A figura abaixo ilustra o passo inverso que leva dos coeficientes de aproximação e detalhes para os coeficientes originais.



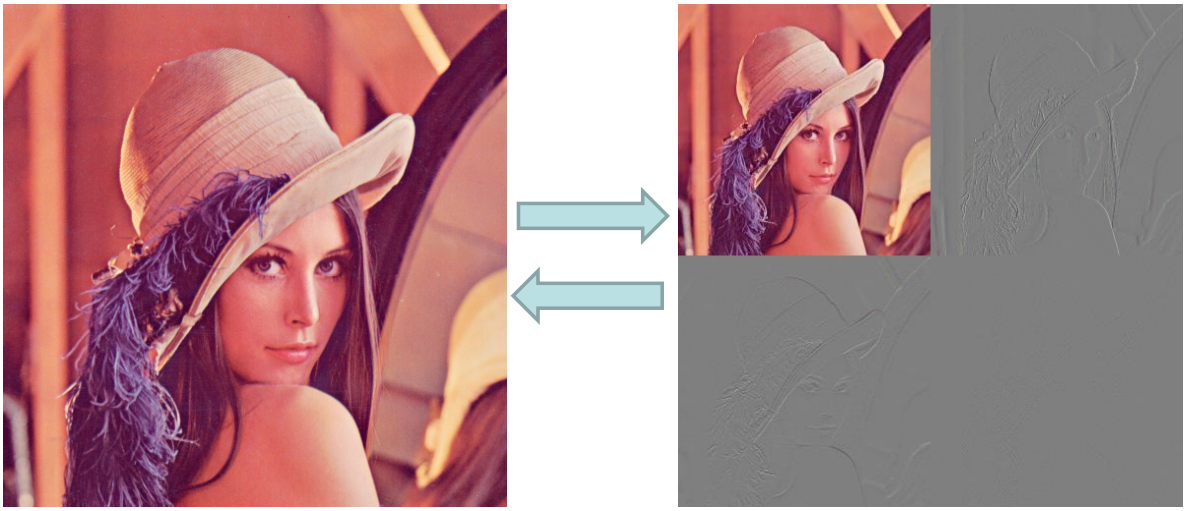
Um passo da inversa da transformada de Haar.

A Transformada de Haar em uma imagem pode ser implementada como uma generalização do esquema mostrado acima. Basta tratarmos inicialmente cada linha da imagem como um vetor de amostras independente uma das outras. Após este passo, repetimos o processo para cada coluna como ilustra afigura abaixo.



Um passo da transformada de Haar em imagens.

A figura abaixo ilustra duas iterações na transformada de Haar e sua inversa que retorna a imagem original.



Exemplo de duas iterações da transformada de Haar e sua inversa.

Redução de ruídos

A idéia central dos algoritmos de redução de ruídos é a seguinte: dada uma imagem $I(x,y)$ com um ruído $N(x,y)$, reduza $N(x,y)$ o máximo que puder sem alterar significativamente $I(x,y)$.

O modelo aditivo de ruído pressupõe que uma imagem com ruído seja formada por:

$$\tilde{I}(x,y) = I(x,y) + N(x,y)$$

Os dois principais tipos de ruídos presentes na captura de imagens são: o **ruído branco** e o **ruído sal e pimenta**. O ruído branco, também chamado de ruído Gaussiano, é um processo estocástico de média zero, independente do tempo e do espaço. Ou seja, é uma variação randômica no valor da imagem para mais e para menos que ocorre em todos os pixels sem diferenciar partes da imagem nem variar com o tempo. Note que a função aleatória não varia, mas os valores são aleatórios e não são constantes.

Um modelo matemático que simula a geração do ruído branco é consiste em gerarmos para cada pixel valores de $N(x, y)$ sorteando uma variável aleatória de distribuição Gaussiana de média zero.

O ruído sal e pimenta, também chamado de ruído impulsivo, é normalmente causado por erro de transmissão e defeito nas componentes. Também é independente do tempo e do espaço, mas não é um processo que ocorre em todos os pixels nem, necessariamente, tem média zero.

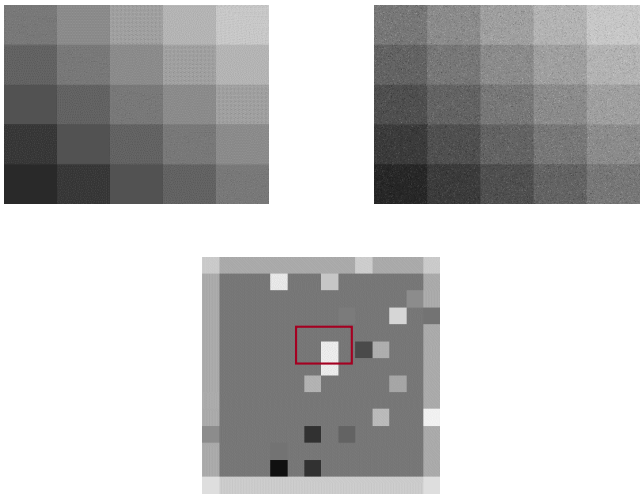


Imagem original e com ruído sal e pimenta. Abaixo um detalhe da vizinhança de um *pixel*.

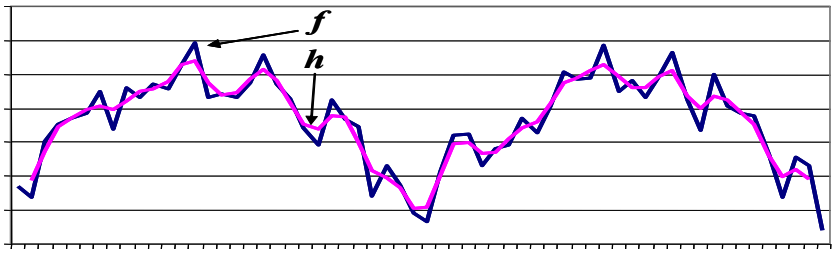
Um modelo matemático para simular a geração de um ruído sal e pimenta é o dado por:

$$N(x,y) = \begin{cases} 0, & se\ t < \alpha \\ N_{min} + u(N_{max} - N_{min}), & se\ t \geq \alpha \end{cases}$$

onde α é um parâmetros que determina se o ruído ocorre ou não, N_{min}e N_{max} definem as intensidades e t, u é são variáveis aleatórias uniforme no intervalo zero a um.

Ruído branco

Para reduzir ruído branco, buscamos procedimentos que atenuam as variações localizadas. Para ilustrar esta ideia considere a função *f(x)* da figura abaixo.



Suavização de função.

Se substituirmos cada valor f_i do interior do domínio por uma média ponderada do tipo:

$$h_i = \frac{f_{i-1} + 2f_i + f_{i+1}}{4}$$

obtemos a função h_i também mostrada na figura. Note como após a aplicação da equação a função se torna mais suave. Esta propriedade é geral nestas transformações que substituem o valor local por uma média ponderada da vizinhança. Os valores de máximos são naturalmente reduzidos, uma vez que fazem média com valores menores que eles. O mesmo raciocínio se aplica para explicar o aumento dos valores de mínimos.

O cálculo de $h_i$deve ser feito para $i = 1,2,\ldots,n-2$. Uma outra maneira de escrever algebricamente este processo consiste em definir uma função auxiliar g_l como:

$$g_l = \begin{cases} 0 & se \quad l < -1 \\ 1/4 & se \quad l = -1 \\ 2/4 & se \quad l = 0 \\ 1/4 & se \quad l = +1 \\ 0 & se \quad l > +1 \end{cases}$$

A partir daí o cálculo da função suavizada pode ser escrito por:

$$h_i = \sum_{k=0}^{n-1} g_{(k-i)} f_i$$

Se considerarmos as funções *f* e *g* contínuas, temos:

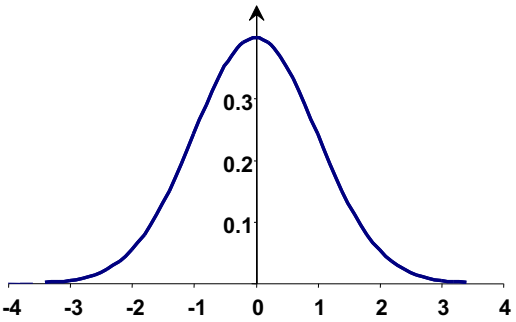
$$h(x) = \int_{t=-\infty}^{t=\infty} g(t-x) f(x) dt$$

que é a convolução da função *f(x)* com o a função *g(x)*.

Quando a função *g* for um tipo média ponderada com a vizinhança ela elimina as variações de maior frequência da função *f*, ela é designada de "filtro passa-baixa". É comum utilizarmos filtros passa-baixa baseados na função de distribuição de Gauss:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

onde σ é o desvio padrão da distribuição. O gráfico desta função está ilustrado na Fig. 3.19.



Gaussiana de média 0 e desvio padrão 1.

Esta função tem duas propriedades importantes para um filtro: é simétrica e a integral dela em todo o domínio é 1.0.

Voltando à notação discreta, temos que a função *g* pode ser representada pela matriz:

$$\frac{1}{4}[1 \quad 2 \quad 1]$$

entendendo que o ponto central 2/4 é o peso da amostra no local em que está sendo computada a função suavizada e os valores à direita e à esquerda da matriz correspondem a amostras à direita e à esquerda da função, respectivamente. Esta forma permite generalizar estas convoluções discretas, bastando para isto definir a matriz a ela associada.

Outras formas discretas da distribuição gaussiana, de maior desvio padrão, seriam:

$$\frac{1}{16}[1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]$$

ou

$$\frac{1}{64}[1 \;\; 6 \;\; 15 \;\; 20 \;\; 15 \;\; 6 \;\;]$$

O efeito de duas passadas do filtro $\frac{1}{4}[1 \quad 2 \quad 1]$ é equivalente a uma passada do filtro $\frac{1}{16}[1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]$ como ilustra o exemplo a seguir. Isto reforça a idéia que a convolução com Gaussianas de desvio padrão maiores aceleram a suavização da imagem. A suavização reduz o ruído branco, mas, em contrapartida, tira a nitidez das bordas e arestas, onde a variação da imagem é grande.

Exemplo:

Com base no sinal discreto unidimensional: (u_0, u_1, u_2, ... , u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, ... , u_{n-1}), mostre que duas passadas do filtro gaussiano $\frac{1}{4}[1 \quad 2 \quad 1]$ é equivalente a uma passada do filtro $\frac{1}{16}[1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]$.

Resp:

Uma passada do filtro de maior base num ponto *i* genérico leva a:

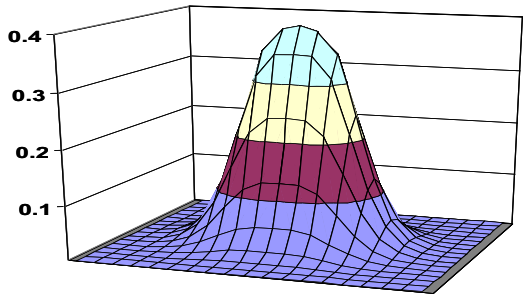
$$u_i' = \frac{1}{16}\left(u_{i-2} + 4u_{i-1} + 6u_i + 4u_{i+1} + u_{i+2}\right) =$$
$$= \frac{1}{4}\left[\frac{1}{4}\left(u_{i-2} + 2u_{i-1} + u_i\right)\right] + \frac{2}{4}\left[\frac{1}{4}\left(u_{i-1} + 2u_i + u_{i+1}\right)\right] + \frac{1}{4}\left[\frac{1}{4}\left(u_i + 2u_{i+1} + u_{i+2_i}\right)\right]$$
$$= \frac{1}{4}\left[u_{i-1}'\right] + \frac{2}{4}\left[u_i'\right] + \frac{1}{4}\left[u_{i+1}'\right]$$

Que corresponde a duas passadas do filtro.

No caso de uma imagem, a função *g* é bidimensional e a matriz é geralmente quadrada. A escolha desta matriz depende do efeito desejado na imagem. Para suavização, podemos adotar a distribuição de Gauss no plano:

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

cuja imagem para $\sigma = 1$ e média zero está ilustrada na figura abaixo.



Gaussiana com media (0,0) e σ=1.

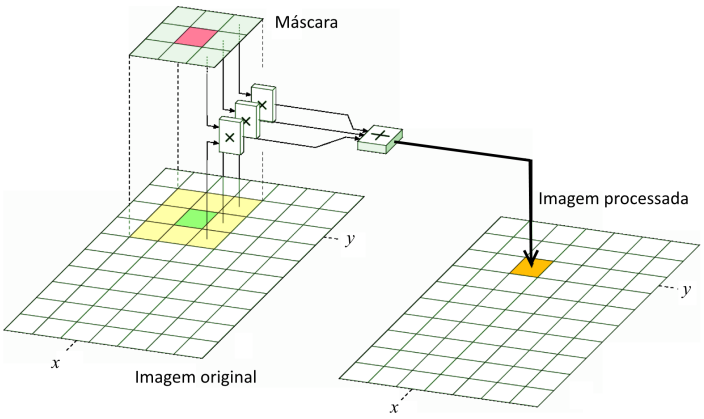
Formas discretas desta função podem ser escritas como:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

ou

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

para um valor maior de desvio padrão. A figura abaixo ilustra o processo de aplicar a máscara num pixel.



Convolução com o uso de uma máscara.

Resumindo, a suavização pode ser útil para retirarmos ruídos de uma imagem. Como, geralmente, o ruído tem uma média zero (tanto adiciona quanto subtrai), e tem alta frequência (varia localmente em torno de um *pixel*), a média ponderada tende a reduzir mais o ruído que a informação da imagem. Note que, em contrapartida à diminuição do ruído, temos uma perda em nitidez, porque as bordas dos objetos da imagem são também altas frequências. Os filtros mais comuns são as formas discretas da função Gaussiana mas existem muitos outros.

Ruído impulsivo

Um algoritmo comumente utilizado para reduzir o ruído impulsivo é o denominado filtro de mediana que substitui o valor de cada *pixel* da imagem pelo valor da mediana do conjunto de valores de seus vizinhos. A figura abaixo ilustra este processo aplicado a um pixel de valor 179 numa vizinhança de tamanho 3× 3.

223	204	204	204	204	204	204	204	204	204	204	204	204	223
171	120	120	120	18	120	50	120	120	120	120	120	120	171
171	120	120	120	116	120	120	120	120	120	120	120	120	171
138	120	120	120	120	120	50	120	97	120	120	120	120	171
171	120	120	120	120	120	120	120	120	120	187	120	120	242
172	120	120	120	120	120	120	120	120	120	120	120	120	171
171	120	120	120	120	120	**179**	120	120	120	120	167	120	171
171	120	120	120	120	120	120	235	120	120	120	120	120	171
171	120	120	120	120	120	120	235	120	76	175	120	120	171
171	120	120	120	120	120	120	120	120	120	120	120	120	171
171	120	120	120	120	120	120	120	123	120	120	214	120	114
171	120	120	120	120	120	120	120	120	120	120	120	143	171
171	120	120	120	232	120	120	198	120	120	120	120	120	171
203	171	171	171	171	171	171	171	171	205	171	171	171	203

Ω_{ij}

Exemplo de filtro de mediana.

O valor da mediana pode ser calculado ordenando o conjunto e tomando valor do elemento no meio do vetor.

179	120	120	120	**120**	120	120	235	235

No caso ilustrado acima o valor do elemento que fica no meio do vetor ordenado é 120.

Correção da gama

Consideremos, por exemplo, a imagem de um jogo de futebol ilustrada na figura abaixo.

Imagem clara.

Imagem após a correção gama γ=0.42.

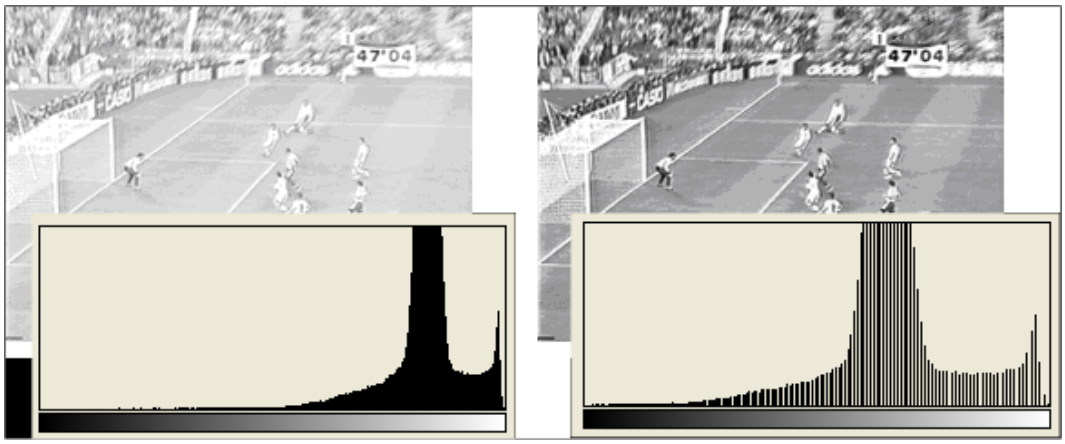
Mudança da luminosidade de cada *pixel* da imagem.

Como a imagem da esquerda está muito clara, podemos, por exemplo, utilizar um processo chamado de *correção gama*, para melhorá-la. Esta correção substitui o valor da luminosidade, L, na escala de zero a um de cada *pixel* por outro valor calculado por:

$$L \leftarrow \sqrt[\gamma]{L}$$

Se tomarmos o valor γ=0.42, obteremos o resultado indicado no lado direito da figura. Note que nesta expressão os valores de L estão necessariamente entre 0 e 1. Se os valores da imagem estiverem na escala 0, 255 é preciso normalizá-la antes.

Uma característica que permite um melhor entendimento da distribuição de tons numa imagem digital é o seu histograma. O histograma de uma imagem é uma função que, para cada valor possível de cor, associa o número de *pixels* em que ela ocorre ou a sua frequência na imagem. A abaixo mostra os histogramas das imagens do campo de futebol.



Histogramas das imagens.

A média, o desvio padrão e a mediana dos valores destes histogramas estão mostrados na Tabela abaixo. A transformação gama com um valor de $\gamma < 1.0$ tende a reduzir a intensidade luminosa, enquanto as transformações com valores maiores que 1.0 fazem o

inverso, ou seja, tornam a imagem mais clara. Note também a natureza não linear da transformação gama, que afeta mais os valores próximos de zero e um.

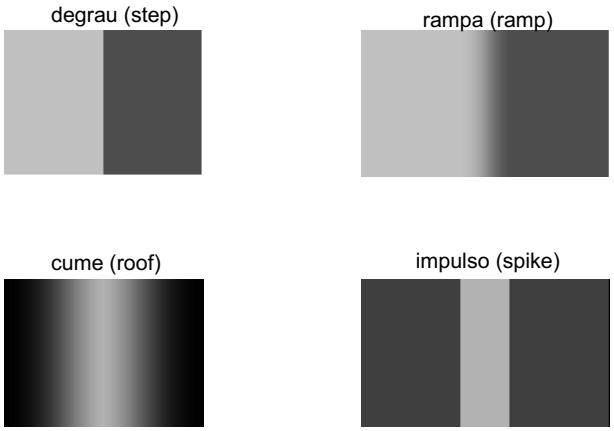
	Média	Desvio Padrão	Mediana
Imagem clara	212	19	212
Imagem transformada	166	33	164

Valores característicos dos histogramas da imagem do campo de futebol.

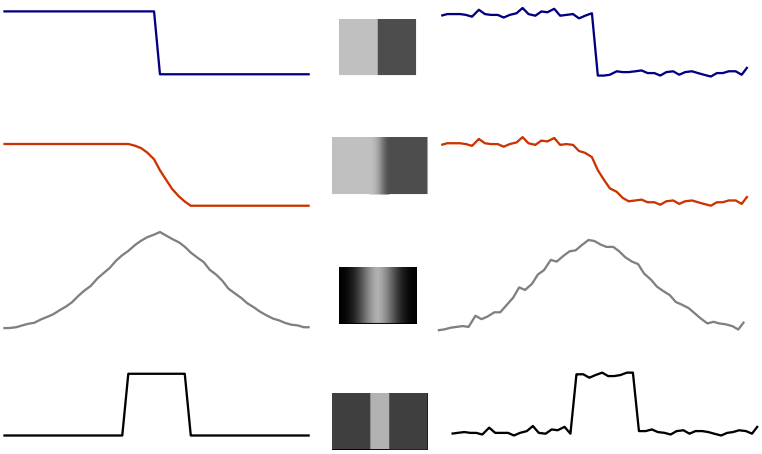
A correção gama é uma das transformações que atuam sobre o valor de cada um dos *pixels* individualmente. Outras correções que servem para ajustar o brilho e o contraste também modificam os valores dos *pixels* transformando o histograma da imagem.

Detecção dos pixels nas bordas de objetos da imagem

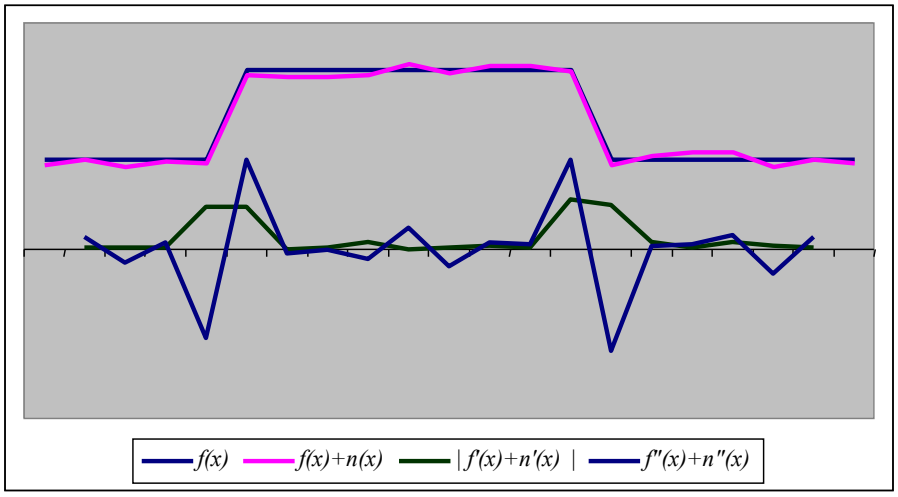
Para exemplificar os algoritmos de realce de arestas, considere a figura abaixo onde na parte inferior está mostrada uma imagem bem simples que é composta apenas de três faixas de cinza. A função *f(x)*, também mostrada na figura, representa os valores dos *pixels* ao longo do eixo que está sobre as faixas, acrescidos de um certo ruído sempre presente nas imagens reais. Como a faixa mais clara tem maior intensidade luminosa o valor da função *f(x)* é mais alto. Na borda entre o cinza escuro e o cinza claro temos uma variação alta num intervalo pequeno, ou seja uma taxa de variação grande.



Tipos de bordas em imagens.



Gráficos da luminosidade das bordas com e sem ruído a direita e esquerda, respectivamente.



Função, função com ruído, módulo da derivada e segunda derivada.

Para destacar bordas e arestas de uma imagem geralmente utilizamos operadores que procuram avaliar taxas de variação da intensidade luminosa. O cálculo de taxas de variação se faz com o uso de derivadas da função de luminosidade. Como numa imagem não dispomos da expressão analítica destas funções, mas sim de valores amostrados em intervalos iguais, os cálculos de derivadas seguem as aproximações de *diferenças finitas*. Para ilustrar como estas aproximações são obtidas, considere a série de Taylor de uma função $f(x)$:

$$f(x+\Delta x) = f(x) + (\Delta x) f'(x) + \frac{(\Delta x^2)}{2} f''(x) + O(\Delta x^3)$$

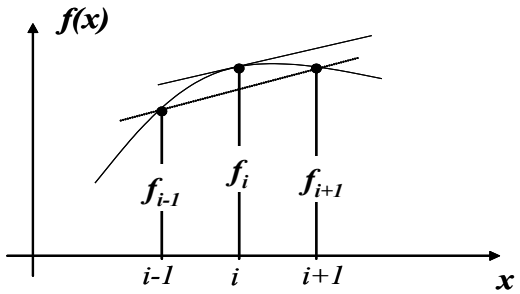
Com $\Delta x = 1$, $f(x) = f_i$ e $f(x+\Delta x) = f_{i+1}$ e a expressão (3.13) pode ser escrita como:

$$f_{i+1} \cong f_i + f'_i + \frac{1}{2} f''_i$$

Com $\Delta x = -1$, $f(x) = f_i$ e $f(x+\Delta x) = f_{i-1}$ temos:

$$f_{i-1} \cong f_i - f'_i + \frac{1}{2} f''_i$$

A figura abaixo ilustra estes valores discretos.



Cálculo de derivada por diferenças finitas.

Subtraindo a equação de f_{i-1} da equação de f_{i+1} podemos avaliar a derivada em x_i como sendo:

$$f'_i \cong (f_{i+1} - f_{i-1})/2$$

Se somarmos as equações de f_{i-1} e de f_{i+1} podemos obter a seguinte aproximação para a segunda:

$$f''_i \cong -(-f_{i+1} + 2f_i - f_{i-1})$$

Como uma imagem é uma função de duas variáveis as derivadas mais comumente utilizadas no realce de arestas são o gradiente e o laplaciano. Dada uma função $f(x,y)$ a expressão analítica do gradiente de f no ponto (x,y) é da dada por:

$$\nabla f(x,y) = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix}$$

Na grade regular este gradiente pode ser estimado por:

$$\nabla f_{ij} = \nabla f(x_i, y_j) = \begin{pmatrix} \big(f(x_{i+1}, y_j) - f(x_{i-1}, y_j)\big)/2 \\ \big(f(x_i, y_{j+1}) - f(x_i, y_{j-1})\big)/2 \end{pmatrix} = \begin{pmatrix} \big(f_{(i+1)j} - f_{(i-1)j}\big)/2 \\ \big(f_{i(j+1)} - f_{i(j-1)}\big)/2 \end{pmatrix}$$

A magnitude deste vetor estima a taxa de variação de f no ponto (x,y) e é, pode ser escrita como:

$$\left\|\nabla f_{ij}\right\| = \tfrac{1}{2}\sqrt{\big(f_{(i+1)j} - f_{(i-1)j}\big)^2 + \big(f_{i(j+1)} - f_{i(j-1)}\big)^2}$$

Os algoritmos se baseiam em comparações deste valor de magnitude nos diversos *pixels*. Como o que interessa são os valores relativos e não absolutos, para reduzir o esforço computacional é comum estimar esta taxa de variação abandonando os quadrados, as raízes e o fator ½. Assim a estimativa de taxa de variação pode ser feita por:

$$\left\|\nabla f_{ij}\right\| = \left|f_{(i+1)j} - f_{(i-1)j}\right| + \left|f_{i(j+1)} - f_{i(j-1)}\right|$$

Uma outra medida de taxa de variação importante é o Laplaciano, que analiticamente se escreve como sendo:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Em diferenças finitas temos:

$$\nabla^2 f_{ij} = 4f_{ij} - (f_{(i+1)j} + f_{(i-1)j} + f_{i(j+1)} + f_{i(j+1)})$$

Uma forma simples de escrever esta fórmula, consiste em fornecer, em uma matriz, os coeficientes que multiplicam os valores dos *pixels*, ou seja:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

O elemento central da matriz corresponde ao *pixel ij* e os demais elementos aos seus vizinhos correspondentes. Esta forma permite uma generalização da convolução sobre uma imagem. Outros operadores, poderiam ser implementados mudando apenas a matriz de coeficientes.

Ocorre, entretanto, que os operados de derivada que detectam as bordas são muito sensíveis ao ruído branco que também tem alta frequência. Para detectarmos a borda com menos interferência do ruído branco é comum acoplarmos ao filtro de detecção de borda outro de suavização. Assim por exemplo, a mascara:

$$\frac{\partial f}{\partial x} \approx \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

representa o filtro de Sobel para calcular $\partial f/\partial x$. Note que o valor que representa o *pixel* a direita é a média ponderada com os coeficientes 1, 2 e1 que aproximam a gaussiana. O mesmo ocorre com o *pixel* a esquerda. Naturalmente o mesmo raciocínio se aplica para calcularmos a aproximação de $\partial f/\partial y$ que resulta em:

$$\frac{\partial f}{\partial y} \approx \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Após o cálculo das derivadas parciais em relação a *x* e *y* podemos calcular o módulo do gradiente por:

$$\|\nabla f\| = \sqrt[2]{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Se desejarmos saber se um *pixel* está na borda de um objeto da imagem a maneira mais simples é comparar o valor de $\|\nabla f\|$ com um certo valor limite . Se estiver acima está na aresta, caso contrário, não.

Para calcularmos a inclinação da aresta podemos utilizar:

$$\theta = atan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

Para as segundas derivadas o operado de Laplaciano suavizado pode ser dado por:

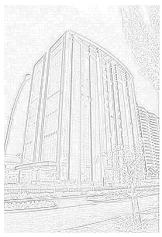
$$\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}$$

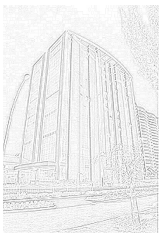
As figuras do prédio abaixo ilustram o uso destes operadores para destacar as arestas de uma imagem. As figuras do centro e da direita mostram resultado da aplicação do gradiente e do laplaciano, respectivamente, após a aplicação de uma correção gama. Note nestes resultados que as regiões de maior taxa de variação estão representadas por um valor mais preto, ou seja, de menor intensidade luminosa. Isto ocorre porque as cores nas figuras b e c estão invertidas, ou seja, para cada *pixel* a intensidade do canal de luminosidade foi transformada por:

$$L \leftarrow 255 - L$$

transformando preto em branco e claro em escuro. O resultado "preto sobre branco" gasta menos tinta na impressão e fica melhor visualmente.


(a) foto

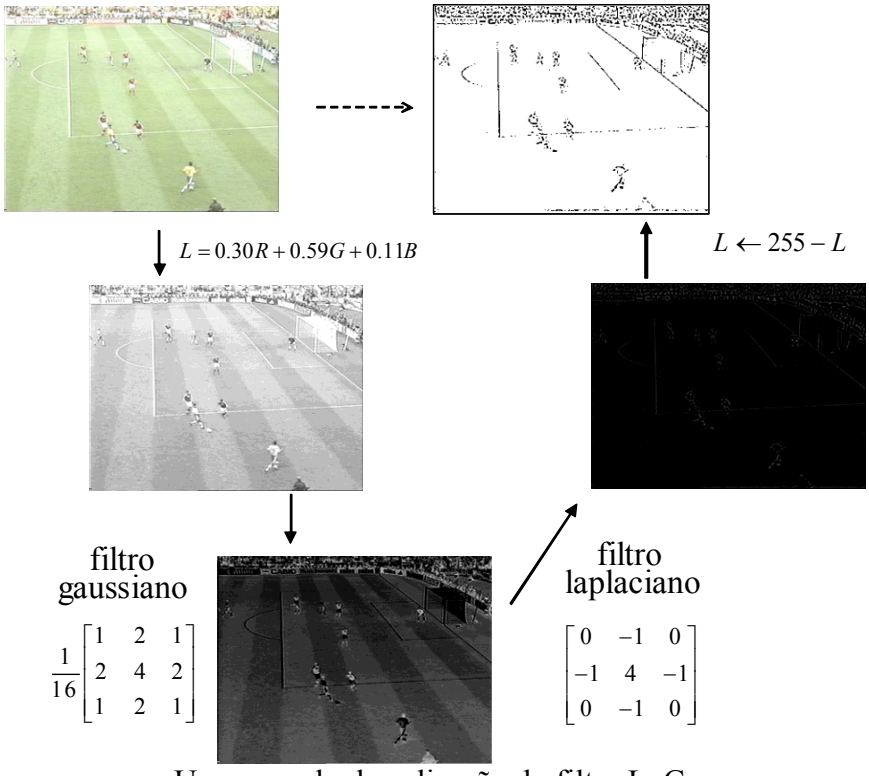

(c) gradiente


(d) laplaciano

Gradiente e laplaciano no uso de detecção de arestas.

A figura abaixo ilustra o processo de realce dos *pixels* que estão sobre as linhas de campo. Este processo de aplicar o filtro Gaussiano antes do Laplaciano é tão comum que os operadores combinados recebem o nome de "LoG". Note que nesta figura a imagem colorida é transformada em preto e branco através do operador de luminosidade:

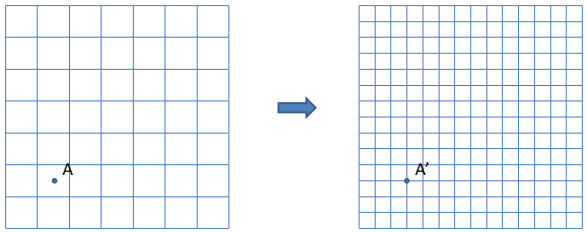
$$L = 0.30R + 0.59G + 0.11B$$



Um exemplo de aplicação do filtro LoG.

Exercícios resolvidos

1. Explique os processos de reconstrução e amostragem que ocorrem nos algoritmos para aumentar a resolução de uma imagem. De duas opções de estratégias de calcular a cor do *pixel* "A" mostrado na figura abaixo?



Resp.:

Para aumentar a resolução de uma imagem precisamos conhecer o valor de cor em pontos do plano onde não temos amostras. Para estimar estes valores temos que reconstruir a função de cor da imagem original para então podermos amostrá-la. No caso do ponto A da figura temos podemos seguir diversas estratégias. Uma seria simplesmente atribuir ao ponto A' a cor do pixel mais próximo de A. Outra seria utilizar uma interpolação linear obtendo a cor em A em função das quatro amostras mais próximas dele.

---

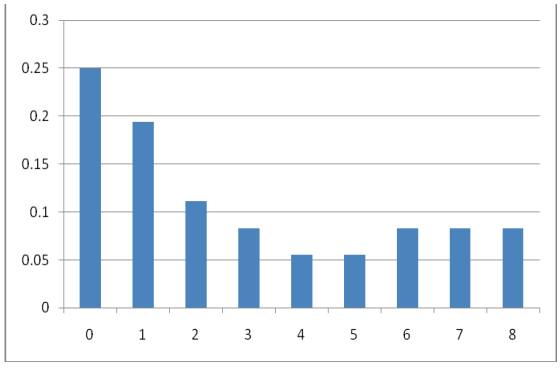
2. Considere a imagem 6×6 com 9 tons de luminosidade mostrada abaixo.

1	0	1	1	0	1
0	5	7	8	4	2
1	4	6	8	7	2
0	5	8	6	7	6
2	1	0	2	1	0
0	3	3	0	3	0

Calcule a probabilidade de um *pixel* aleatoriamente escolhido desta imagem ter uma determinada luminosidade. Faça um gráfico com estas probabilidades para cada um dos 9 tons.

Resp.:

Valor	Freq.	Prob.
0	9	0.25
1	7	0.1944
2	4	0.1111
3	3	0.0833
4	2	0.0556
5	2	0.0556
6	3	0.0833
7	3	0.0833
8	3	0.0833
	36	1



3. A uma imagem 6×6 mostrada abaixo, ilustra os dois tipos de ruídos mais comuns nas imagens capturadas. Pergunta-se que tipo de ruído são eles e que tipo de processo se utiliza para reduzi-los? De um exemplo de aplicação para cada um deles.

$$\begin{bmatrix} 19 & 20 & 21 & 20 & 18 & 20 \\ 20 & 18 & 19 & 255 & 20 & 18 \\ 22 & 20 & 18 & 20 & 19 & 21 \\ 20 & 19 & 21 & 22 & 18 & 19 \\ 18 & 20 & 22 & 20 & 19 & 20 \\ 21 & 20 & 19 & 20 & 19 & 21 \end{bmatrix}$$

Resp.:
A imagem apresenta um ruído branco (pequenas variações oscilatórias com média zero) e um pixel com ruído impulsivo, tipo "sal e pimenta" (o que tem valor 255).

A maneira de reduzir os ruídos seria aplicarmos filtros.

Um filtro apropriado para o ruído branco é o filtro Gaussiano. Para o pixel de valor 18 no canto superior esquerdo da imagem o filtro Gaussiano 3×3 o transformaria para:

$$v = \frac{1}{16}\left(19 + 2\cdot 20 + 21 + 2\cdot 20 + 4\cdot 18 + 2\cdot 19 + 22 + 2\cdot 20 + 18\right) = 19.375 \cong 19$$

se aproximando mais do valor 20.

Um filtro para atenuar o ruído impulsivo é o filtro de mediana. O pixel com valor 255 submetido a este filtro numa janela 3×3 se transforma em 20 pelo algoritmo.

Vizinhos em ordem = (18, 18, 19, 19, 20, 20, 21, 255). Mediana = 19.5

4. Considere uma imagem em tons de cinza representada pela matriz abaixo:

$$\begin{bmatrix} 20 & 20 & 45 & 50 \\ 15 & 18 & 40 & 54 \\ 10 & [16] & 8 & 26 \\ 10 & 22 & 20 & 30 \end{bmatrix}$$

Determine qual o valor do pixel indicado na figura, atualmente de valor 16, depois de passarmos o filtro de Sobel, para acharmos arestas na imagem.

Resp.:

$$\Delta_x = \frac{1}{4}\begin{bmatrix} 1\times 15 & 0 & -1\times 40 \\ 2\times 10 & 0 & -2\times 8 \\ 1\times 10 & 0 & -1\times 20 \end{bmatrix} = \frac{15-40+20-16+10-20}{4} = \frac{-31}{4} = -7.75$$

$$\Delta_y = \frac{1}{4}\begin{bmatrix} 1\times 15 & 2\times 18 & 1\times 40 \\ 0 & 0 & 0 \\ -1\times 10 & -2\times 22 & -1\times 20 \end{bmatrix} = \frac{15+36+40-10-44-20}{4} = \frac{17}{4} = 4.25$$

Novo valor = $\sqrt{7.75^2 + 4.25^2} = 8.839$

5. Considere a imagem 7×7em tons de cinza sem ruídos representada pela matriz abaixo:

6	64	64	64	64	64	64	64
5	0	62	64	64	64	64	64
4	0	0	64	64	64	64	64
3	0	0	0	64	64	64	64
2	0	0	0	0	64	64	64
1	0	0	0	0	0	64	64
0	0	0	0	0	0	0	64
	0	1	2	3	4	5	6

Que filtro você utilizaria para avaliar se um *pixel* está ou não sobre uma aresta? Como você poderia determinar a direção da aresta? De um exemplo de um *pixel* que está sobre uma aresta e outro que não está. Qual a direção desta aresta?

Resp:

Uma maneira simples de determinar se um *pixel* está sobre uma aresta seria calcularmos o módulo do gradiente da imagem nele. Um filtro de Sobel calcula as derivadas com uma suavização gaussiana prévia:

$$\frac{\partial f}{\partial x} \approx \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \frac{\partial f}{\partial y} \approx \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \|\nabla f\| = \sqrt[2]{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

A direção da aresta pode ser calculada por:

$$\theta = atan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

Qualquer dos *pixels* com a vizinhança:

64	64	64
0	***64***	64
0	0	64

está sobre uma aresta, e :

$$\frac{\partial f}{\partial x} \approx 4 \times 64 - 64 = 192 \qquad \frac{\partial f}{\partial y} \approx 4 \times 64 - 64 = 192 \qquad \|\nabla f\| = \sqrt[2]{(192)^2 + (192)^2 = 192}$$

$$\theta = atan\left(\frac{192}{192}\right) = \frac{\pi}{8}$$

Os *pixels* do tipo dos sublinhados a seguir ***não*** estão sobre uma aresta.

0	0	0
0	***0***	0
0	0	0

64	64	64
64	***64***	64
64	64	64

Os *pixels* abaixo não estão sobre uma aresta, mas devido a proximidade dela possuem gradiente significativo:

0	0	64
0	***0***	0
0	0	0

0	64	64
0	**0**	64
0	0	0

64	64	64
64	***64***	64
0	64	64

Uma outra resposta para esta pergunta consiste em calcularmos o Laplaciano de

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A aplicação do Laplaciano resulta em:

	-120	-2	0	0	0	
	126	-128	0	0	0	
	0	128	-128	0	0	
	0	0	128	-128	0	
	0	0	0	128	-128	

A aresta está no cruzamento de zero entre os *pixels* positivos e negativos como ilustra a figura acima.

---

6. Considere uma imagem em tons de cinza representada pela matriz abaixo. Aplique os filtros de Sobel e Laplaciano. Discuta como eles determinam a borda.

$$\begin{bmatrix} 20 & 20 & 20 & 100 & 100 & 100 \\ 20 & 20 & 20 & 100 & 100 & 100 \\ 20 & 20 & 20 & 100 & 100 & 100 \\ 20 & 20 & 20 & 100 & 100 & 100 \\ 20 & 20 & 20 & 100 & 100 & 100 \\ 20 & 20 & 20 & 100 & 100 & 100 \end{bmatrix}$$

Resp.:

Utilizando os filtros de Sobel para capturar uma linha vertical temos:

$$\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}\otimes\begin{bmatrix}20 & 20 & 20\\ 20 & 20 & 20\\ 20 & 20 & 20\end{bmatrix}=\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}\otimes\begin{bmatrix}100 & 100 & 100\\ 100 & 100 & 100\\ 100 & 100 & 100\end{bmatrix}=0$$

$$\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}\otimes\begin{bmatrix}20 & 20 & 100\\ 20 & 20 & 100\\ 20 & 20 & 100\end{bmatrix}=\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}\otimes\begin{bmatrix}20 & 100 & 100\\ 20 & 100 & 100\\ 20 & 100 & 100\end{bmatrix}=80$$

resultando em:

20	20	20	100	100	100
20	0	80	80	0	100
20	0	80	80	0	100
20	0	80	80	0	100
20	0	80	80	0	100
20	20	20	100	100	100

onde a aresta vertical aparece nas duas colunas de *pixels* que fazem fronteira com ela.

Utilizando o Laplaciano temos:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 20 & 20 & 20 \\ 20 & 20 & 20 \\ 20 & 20 & 20 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 100 & 100 & 100 \\ 100 & 100 & 100 \\ 100 & 100 & 100 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 20 & 20 & 100 \\ 20 & 20 & 100 \\ 20 & 20 & 100 \end{bmatrix} = -80, \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 20 & 100 & 100 \\ 20 & 100 & 100 \\ 20 & 100 & 100 \end{bmatrix} = 80$$

20	20	20	100	100	100
20	0	-80	80	0	100
20	0	-80	80	0	100
20	0	-80	80	0	100
20	0	-80	80	0	100
20	20	20	100	100	100

A aresta está no cruzamento de zero entre -80 e 80.

---

7. Calcule a inversa da transformada de Haar de: (40, 16, 32,16, 6, 1, 1, 1) seguindo o algoritmo simplificado apresentado em aula que se baseia em funções que variam entre +1 e -1. Desenhe as oito primeiras funções da base de Haar.

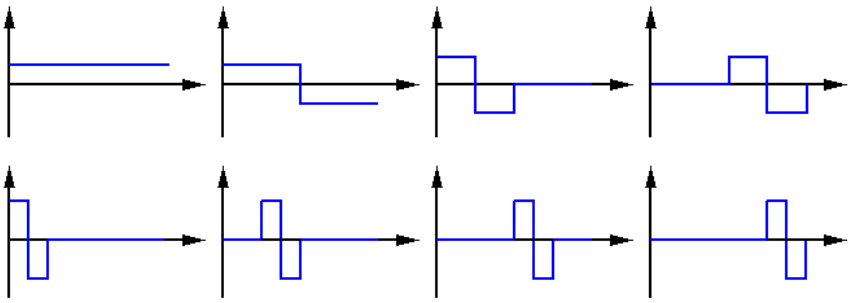
Resp: (use a tabela para auxiliar a minha correção. A **transformada de Haar** está copiada na 1ª linha. Note que o que é pedido é a função original.)

40	16	32	16	6	1	1	1
56	24	32	16				
88	24	40	8	6	1	1	1
94	**82**	**25**	**23**	**41**	**39**	**9**	**7**

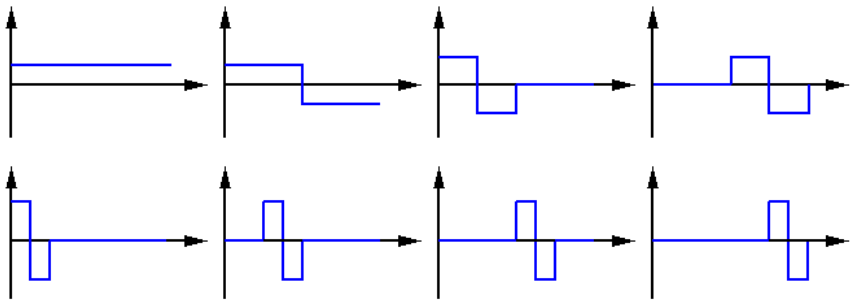
Fazendo a ida e a volta:

40	16	32	16	6	1	1	1
56	24	32	16	6	1	1	1
88	24	40	8	6	1	1	1
94	**82**	**25**	**23**	**41**	**39**	**9**	**7**
88	24	40	8	6	1	1	1
56	24	32	16	6	1	1	1
40	16	32	16	6	1	1	1

As funções são dados por:



8. Calcule a transformada de Haar sem perda do sinal discreto 18,14,6,10. Coloque a resposta na ordem da base ilustrada abaixo onde os valores das ordenadas das funções são iguais a 1:



Resp:

18,14,6,10

(18+14)/2,(6+10)/2, (18-14)/2,(6-10)/2

16,8, 2,-2,

(16+8)/2, (16-8)/2, 2, -2

12, 4, 2, -2

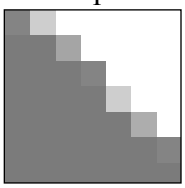
Exercícios propostos

1. A matriz mostrada abaixo com elementos que variam de 0 à 255 representa uma imagem de luminância 6×6. Qual seria a luminância do pixel de valor 118 marcado por [ ]'s, se sobre a imagem fosse aplicado:
   i) um filtro Gaussiano 3×3;
   ii) um filtro de Mediana.

$$\begin{bmatrix} 85 & 134 & 169 & 181 & 184 & 165 \\ 71 & 120 & 152 & 176 & 183 & 182 \\ 56 & 74 & [118] & 158 & 179 & 183 \\ 58 & 62 & 92 & 134 & 169 & 179 \\ 71 & 63 & 73 & 105 & 150 & 172 \\ 69 & 63 & 55 & 76 & 119 & 159 \end{bmatrix}$$

2. Calcule o módulo do gradiente e do laplaciano no pixel em questão na imagem da questão anterior.

3. Considere a imagem em tons de cinza representada pela matriz abaixo:



132	206	255	255	255	255	255
123	123	165	255	255	255	255
123	123	123	**132**	255	255	255
123	123	123	125	206	255	255
123	**123**	123	123	123	173	255
123	123	123	123	123	123	132
123	123	123	123	123	123	123

   Que filtro você utilizaria para indicar se um *pixel* está sobre uma aresta? Qual o valor deste operador aplicado aos *pixels* de valor 132 e 123, marcados na matriz em negrito e sublinhado?

4. Dada função
   $f(x) = 3sen(\pi x) + 0.2\cos(\pi x/8)$
   qual o maior intervalo de amostragem Δx de forma a podermos reconstruí-la corretamente?

5. Considere uma imagem em tons de cinza representada pela matriz abaixo:

$$\begin{bmatrix} 20 & 20 & 20 & 21 \\ 15 & 18 & 28 & 24 \\ 10 & 16 & 8 & 26 \\ 10 & 22 & 20 & 30 \end{bmatrix}$$

Sem modificar os *pixels* da borda, calcule a matriz que representa a imagem suavizada pelo filtro Gaussiano.

6. Explique o que é resolução espacial de uma imagem.
7. O que é quantização e quando este processo ocorre na aquisição de uma imagem digital?
8. Explique os processos de reconstrução e amostragem que ocorrem quando mudamos a resolução espacial de uma imagem.

Trabalhos propostos

Algoritmo de HDRI

A foto abaixo é o resultado de um algoritmo de HDRI (*High Dinamic Range Image*) que consegue mapear melhor os tons de diversas imagens, como as mostradas acima, em um único conjunto de 256 níveis que combina todas as fotos em uma apenas.

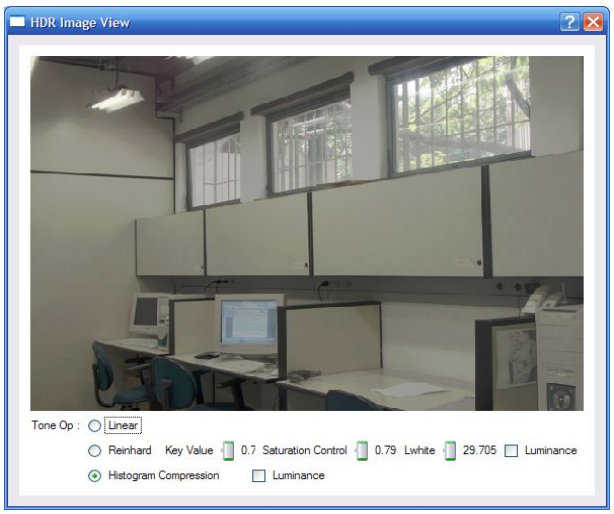


Imagem resultante de um algoritmo de HDRI aplicado nas imagens acima.

Referencias básicas de HDRI:

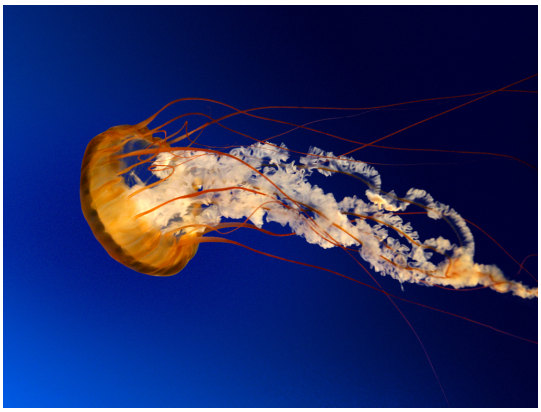
1) DEBEVEC, P., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proc. of ACM SIG- GRAPH '97*, 369–378.
2) M. Robertson, S. Borman, and R. Stevenson, "Dynamic range improvement through multiple exposures," in Proceedings of the 1999 International Conference on Image Processing (ICIP-99), pp. 159–163, (Los Alamitos, CA), Oct. 24–28 1999.
3) www.debevec.org
4) www.hdrshop.com

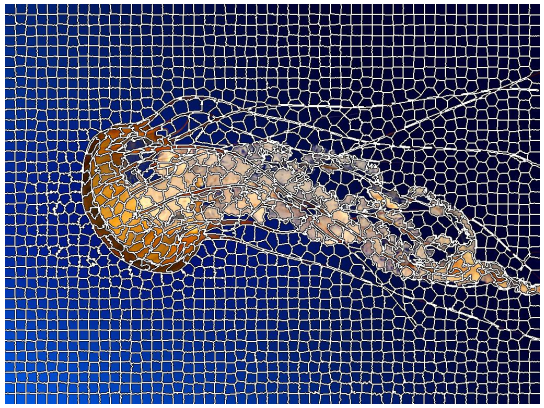
A partir destas busque as mais recentes.

Algoritmo de Super Pixels

Implemente o algoritmo de superpixel SLICO descrito em:

1) https://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf
2) https://infoscience.epfl.ch/record/177415





Bibliografia do capítulo

1. Foley, J. D., Van Dam, A., Feiner, S. K., e Huhes, J. F., **Computer Graphics: Principles and Practices**, (Systems Programming), 2nd edition in C, Addison-Wesley, 1995, ISBN 0-201-84840-6.
2. Gomes, J.M. e Velho, L., **Image Processing for Computer Graphics**, Springer, 1997, ISBN 0-387-94854-6
3. Gonzalez, R.C., and Woods, R.E, **Digital Image Processing**, Addison-Wesley, 1992.
4. Baxes, G. A., **Digital Image Processing: principles and applications**, John Wiley & Sons, New York, 1994, ISBN 0-471-00949-0