DETECÇÃO DE BORDAS EM IMAGENS DIGITAIS

Helder C. R. De Oliveira, Marco A. Piteri, Aylton Pagamisse

Departamento de Matemática, Estatística e Computação, FCT, UNESP

19.060-900, Presidente Prudente, SP

heldercro@gmail.com; {piteri, aylton}@fct.unesp.br

Resumo: Este artigo apresenta os resultados preliminares do protótipo de um sistema de software voltado para a manipulação de imagens que tem como propósito dar suporte as principais classes de algoritmos na área de *processamento de imagens*, como, binarização, detecção de bordas, afinamento (*thinning*) e segmentação. Outro objetivo primário no desenvolvimento desta ferramenta é que ela possa ser empregada de forma didática no contexto de disciplinas comuns aos cursos da área de computação e de engenharia, tais como, *processamento digital de imagens*, *reconhecimento de padrões* e *visão computacional*, não só para ilustrar, sistematicamente, o efeito das operações aplicadas às imagens, mas também para permitir que os alunos possam acrescentar novas funcionalidades ao sistema. Com o intuito de viabilizar a proposta de elaboração do sistema, foi escolhido inicialmente o problema detecção de bordas e, até o presente momento, três métodos baseados em operadores de derivação de primeira e segunda ordem foram implementados. O software está sendo elaborado nas linguagens C/C++ e faz uso das bibliotecas Qt e OpenCV, que são de código aberto e amplamente utilizadas pela indústria e acadêmia. Em síntese, o trabalho fornece uma visão geral sobre o sistema e apresenta os resultados preliminares já obtidos.

Palavras-Chave: *Processamento de imagens, Detecção de bordas, OpenCV, Qt.*

Duas das principais ambições dos pesquisadores, entre eles os da área de ciência da computação, são reproduzir modelos que descrevem e simulem o comportamento do cérebro e da visão, dois dos sistemas mais complexos existentes no ser humano. Em particular, o desenvolvimento de metodologias e técnicas para reprodução da visão humana em máquinas, denominada *visão computacional*, permitirá automatizar uma série de tarefas, como por exemplo: reconhecimento de objetos/pessoas em ambientes diversos, auxílio na análise e diagnóstico de imagens médicas (raios-X, ultrassom, tomografia, ressonância magnética), análise de imagens de satélites (sensoriamento remoto), navegação autônoma de robôs, veículos aéreos, terrestres e marítimos, entre outras.

É importante realçar que independente do problema que se busca resolver na área de visão computacional, a imagem ou sequência de imagens (vídeo) que está sendo processada deve passar por uma etapa de pré-processamento que consiste num conjunto de operações que objetivam melhorar a qualidade final da imagem, eliminando ruídos indesejados e/ou, destacando alguns de seus elementos (objetos, partes) baseados em alterações de suas características relevantes, como cor e textura.

Uma das principais operações passível de ser feita sobre uma imagem é, indiscutivelmente, a *detecção de bordas* e muitas vezes, uma das primeiras tarefas a ser realizada em muitas aplicações [1], [2]. Neste sentido, a qualidade dos resultados obtido nesta etapa influencia decisivamente no sucesso da resolução do problema em estudo, razão pela qual esta classe de algoritmos foi escolhida como objeto primário deste estudo.

Uma das motivações que levaram ao desenvolvimento do presente projeto, é que os principais sistemas comerciais existentes no mercado para a manipulação de imagens (Photoshop, CorelDraw, MatLab, ...) não possuem código aberto. Dessa forma, o usuário realiza a operação, vê os resultados, sabe “o que” foi feito, mas não sabe “como” foi feito.

Em geral, os aspectos conceituais que fundamentam a área de processamento digital

de sinais e que são largamente empregados na área de processamento de imagens, utilizam um arcabouço teórico derivado da matemática contínua, que afasta os alunos de atividades práticas de desenvolvimento de códigos. Nesse sentido, os autores acreditam que, após o estudo dos elementos teóricos que fundamentam os algoritmos para o problema de detecção de bordas, o contato do aluno com o respectivo código que materializa os conceitos, diminui a distância entre a prática e a teoria, facilita o seu aprendizado e pode atrair interessados na elaboração de novas funcionalidades ao sistema.

Com esta nova perspectiva, agora o aluno, além de saber "o que" faz, passa a ter conhecimento "de como" fazer, e, consequentemente, dominar a tecnologia envolvida.

O objetivo fundamental do presente trabalho consiste no desenvolvimento de um sistema de software para dar suporte as principais classes de algoritmos associados à manipulação de imagens bidimensionais. O sistema está sendo desenvolvido na linguagem C/C++ com o uso do compilador GCC, as bibliotecas Qt e OpeCV.

A biblioteca Qt, escrita originariamente em C++, mas disponível em outras linguagens como Python (PyQt, PySide) e Java (Qt Jambi), possui 800 classes e tem como propósito facilitar a construção de interfaces gráficas ricas e intuitivas, libertando o programador para se concentrar no foco principal, que é o desenvolvimento da aplicação. Permite um alto nível de abstração na manipulação dos pixels das imagens processadas, independente dos principais formatos utilizados no mercado, como: JPG, PNG, BMP, GIF, entre outros, mantendo a mesma uniformidade na aparência, independente da plataforma em que o sistema seja executado. Outro dado interessante é que esta biblioteca é à base do ambiente *desktop* gráfico KDE[1], e utilizada pelas principais empresas do mercado de desenvolvedoras de software, como por exemplo, Google e Adobe, entre outras, tornando-se um padrão "de facto" para a elaboração de interfaces.

Por outro lado, a biblioteca OpenCV[2], desenvolvida originariamente pela Intel em C/C++ e posteriormente doada à comunidade de programadores, possui um conjunto de recursos de alto nível para a manipulação de imagens e vídeos. Atualmente existem versões desta biblioteca para as linguagens Java, Python e Visual Basic. Até o presente momento, seu uso no protótipo desenvolvido se limita a captura de imagens e ao uso da técnica detecção de bordas de Canny, de modo a permitir realizar comparações com os resultados obtidos pelos algoritmos já implementados no sistema.

É importante salientar que a captura de imagens por meio de *webcams* é uma tarefa um tanto complicada, sendo assim, esta facilidade disponível junto a OpenCV foi bastante útil. No futuro, as demais funcionalidades existentes junto a esta biblioteca serão exploradas.

Os **detectores de bordas** são métodos locais de processamento de imagem elaborados para identificarem grupos de *pixels* que definem uma borda, que pode ser vista como uma linha em que a intensidade do fundo de cada um de seus lados, ou, é muito superior, ou muito inferior à intensidade dos *pixels* da linha.

Em geral os passos fundamentais relativos aos algoritmos de detecção de bordas, são:

- **Suavizar a imagem para reduzir o ruído**: A segunda derivada de um ponto qualquer da imagem é muito sensível a ruídos. Se uma imagem possui ruídos em grande quantidade, pode ser praticamente impossível detectar suas bordas;
- **Detecção dos pontos da borda**: Esta operação ocorre localmente na imagem e determina os pontos candidatos a serem pontos da borda;
- **Localização da borda**: Esta operação retira os pontos (falsos candidatos) que não pertencem à borda, deixando apenas os que realmente interessam.

Do ponto de vista histórico, o *algoritmo* ou *operador de Sobel*, foi um dos primeiros no sentido de resolver o problema de detecção bordas. Para este propósito, filtra a imagem de entrada com máscaras de dimensão 3x3, nas direções x e y, respectivamente, conforme ilustrado na

[1] http://kde.org

[2] http://opencvlibrary.sourceforge.net

Figura 1, onde cada quadrado corresponde a um *pixel* na imagem.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

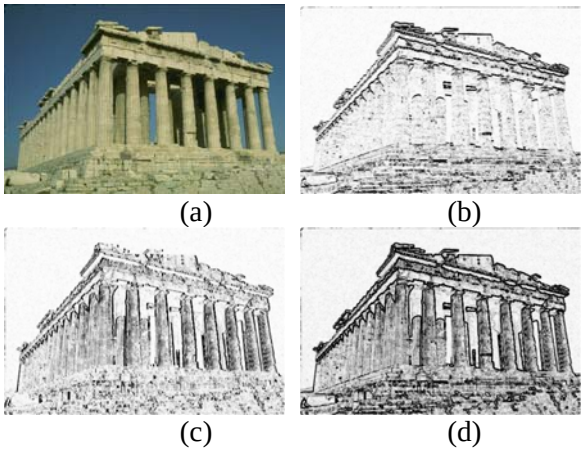
Figura 1: Modelo de máscara 3x3 de Sobel.

O cálculo das derivadas parciais nas direções *x* e *y* com a máscara centrada em z_5 para todos os pixels da imagem, é dado, respectivamente, pelas Equações (1) e (2).

$$\frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad (1)$$

$$\frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \qquad (2)$$

As Figuras 2b e 2c ilustram os resultados da aplicação do filtro de Sobel nas direções *x* ou *y* a partir da imagem original dada pela Figura 2a, enquanto a Figura 2d mostra os resultados em ambas direções.



(a) (b)

(c) (d)

Figura 2: Filtro de Sobel. (a) imagem original; (b) eixo-x; (b) eixo-y e (d) ambas as direções.

Os operadores de Sobel são fáceis de serem implementados e obtêm resultados bastante satisfatórios, mesmo na presença de imagens ruidosas, razões pelas quais continuam a ser usados [1], [2].

A partir da abordagem de Sobel, Marr e Hildreth desenvolveram uma alternativa mais elaborada, argumentando que:

- A mudança de intensidade na imagem é dependente de sua escala. Portanto, a detecção de bordas requer o uso de operadores de tamanhos diferentes;

- Caso ocorra uma alteração brusca de intensidade na imagem, este efeito irá originar um pico ou um vale na primeira derivada ou, um cruzamento por zero na segunda derivada, fazendo com que ela atravesse o eixo-*x*.

Os autores argumentaram ainda que o operador que poderia satisfazer as premissas acima seria o filtro $\nabla^2 G$, também chamado de **LoG** (*Laplacian of Gaussian*), ou seja, Laplaciano da gaussiana, onde ∇^2 é o operador laplaciano (derivada de segunda ordem) e G é a função gaussiana bidimensional dada pela Equação 4, onde σ é o desvio padrão.

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (4)$$

É importante observar que ao usarmos uma máscara associada à superfície $\nabla^2 G$ (tamanho 5x5, por exemplo), o elemento central possui um valor positivo, os elementos "mais próximos" valores negativos e os mais distantes (> 3), estabilizam-se em zero.

Agora, a borda a ser identificada, se existir, deve ser procurada na vizinhança do elemento central (núcleo), já que é exatamente nestes pontos que o gráfico da função cruza o eixo-*x*.

O que ocorre na prática é que o operador $\nabla^2 G$ borra a imagem atenuando a intensidade dos níveis de cinza (mesmo nos ruídos) para uma escala da ordem de σ, pois a gaussiana é suave nos domínio espacial.

Logo, a primeira derivada pode ser usada para encontrar variações bruscas de intensidade, mas elas são operadores direcionais. Por sua vez, o laplaciano tem a propriedade de ser isotrópico (invariante a rotação, ou seja, preserva as propriedades independentes das direções), assim como o sistema de visão humano, o que é uma vantagem em relação ao uso da primeira derivada. Os passos fundamentais do método são

- Aplicar na imagem original I uma função gaussiana G bidimensional (filtrar);
- Calcular o laplaciano da imagem resultante com uma máscara 3x3 isotrópica, referencie isto por L;
- Encontrar os *pixels* de L que cruzam o zero. Estes *pixels* pertencem às bordas.

A Figura 3b, 3c e 3d ilustram os resultados da aplicação do algoritmo de Marr e Hildreth na Figura 3a original, com diferentes valores de desvio padrão. É fácil observar na Figura 4 que o aumento do valor de σ, elimina a detecção de "falsos positivos" devido à presença de ruídos na imagem, realçando as bordas mais bem definidas, e, consequentemente, fornecendo resultados mais precisos.
Com o passar do tempo o problema de detecção de bordas assumiu uma importância ainda maior e novas abordagens algorítmicas surgiram, entre elas, o algoritmo de Canny, que é decorrente da abordagem anterior e considerado muito eficiente. Canny enuncia três premissas básicas necessárias a um método detector de bordas ótimo:



Figura 3: Aplicação do algoritmo de Marr e Hildreth. (a) imagem original; (b) $\sigma = 2$; (c) $\sigma = 3$; (d) $\sigma = 4$.

- **Taxa de Erro**: O detector de bordas deve identificar somente bordas e ser capaz de encontrar, absolutamente, todas elas;
- **Localização**: A distância entre os *pixels* de uma borda e o centro da mesma ("largura") deve ser mínima;
- **Unicidade**: O detector não deve identificar múltiplos *pixels* numa mesma borda, onde apenas um existe.

Além dos critérios enunciados, Canny [1] consegue expressões matemáticas e busca encontrar soluções "ótimas" para cada um deles por meio de métodos de otimização numérica e defini um detector de bordas baseado na primeira derivada de uma gaussiana. Após a convolução da imagem com a primeira derivada da gaussiana, estima-se a intensidade da borda e a direção de cada ponto por meio de um cálculo da magnitude e da direção do vetor gradiente.
A seguir, é necessário afinar as bordas que foram criadas em torno dos máximos locais pelo uso do gradiente, por meio do método da *supressão dos não máximos* [1]. Na sequência, a imagem passa por um processo de limiarização por histerese, objetivando remover os *pixels* redundantes, onde então são classificados em *fracos* e *fortes*, sendo estes últimos marcados como pertencentes à borda. Em linhas gerais, o algoritmo de Canny pode ser sintetizado nos seguintes passos:

- Suavizar a imagem por meio de máscaras gaussianas unidimensionais (ambas as direções);
- Separa a imagem uniformizada (estágio anterior) em ambas as direções para se computar o gradiente da superfície uniforme;
- Fazer a aplicação do método de supressão dos não máximos na imagem resultante do passo anterior, ou seja, eliminar os pontos de mínima intensidade;
- Aplicar a limiarização (*thresholding*), aceitando todos os *pixels* com valores acima de um limite superior e, rejeitando, todos aqueles com valores abaixo de um limite inferior fixado. Os demais pixels (entre os limites inferior e superior) serão sob certas condições.

Embora o algoritmo de supressão dos não máximos afine as bordas, podem restar algumas que possuem largura maior que um *pixel*, o que pode ser facilmente resolvido com a aplicação de algum algoritmo de afinamento

[2].
A Figura 4a ilustra o resultados obtidos a partir da aplicação do algoritmo de Canny com $\sigma = 1$ e limiares inferior e superior iguais a 0 e -1, na imagem original dada na Figura 3a.



(a) (b)

Figura 4: Resultados da aplicação dos detectores de: (a) Canny; (b) Shen e Castan.

Por outro lado, Shen e Castan corroboram com o método proposto anteriormente em relação à forma geral de um detector de bordas que resulta numa convolução com um núcleo de suavização, seguido por uma busca pelos *pixels* que definem a borda, mas propõem uma nova alternativa de otimização, resultando numa abordagem que é referenciada na literatura por *Infinite Symmetric Exponential Filter* (ISEF) [1].
Esta nova proposta oferece uma melhor relação sinal-ruído e, portanto, consegue uma melhor localização dos *pixels* de interesse. Entretanto, o método não aborda o problema de múltiplos pixels numa mesma borda, o que pode ocasionar resultados falsos na presença de bordas borradas (turvas).
Na Figura 4b é possível observar a aplicação do método devido a Shen e Castan na imagem original dada na Figura 3a.
Por sua vez, a Figura 5 ilustra a interface principal do protótipo desenvolvido. É importante realçar que todos os métodos discutidos neste artigo foram devidamente implementados e estão disponíveis no sistema, além de outras funcionalidades básicas, como, por exemplo, o cálculo do histograma.
Este trabalho apresenta o protótipo de um sistema de software voltado para o processamento de imagens bidimensionais e que integra algumas das principais tecnologias disponíveis no mercado, como o Qt e OpenCV, ambas desenvolvidas sob os paradigmas de programação orientada a objetos e programação genérica.
O protótipo elaborado também poderá servir como uma plataforma de software voltada para o ensino na disciplina de ***processamento de imagens***. Dessa forma, os alunos poderão usá-la como fonte de novos aprendizados, quer do ponto vista técnico-acadêmico ou de pesquisa, na medida em que será possível adicionar outras funcionalidades e categorias de algoritmos (binarização, afinamento, segmentação, ...), por meio de novas classes. O ambiente também é adequado para se explorar conceitos oriundos de diferentes disciplinas do curso, tais como: estrutura de dados, engenharia de software, álgebra linear, etc.

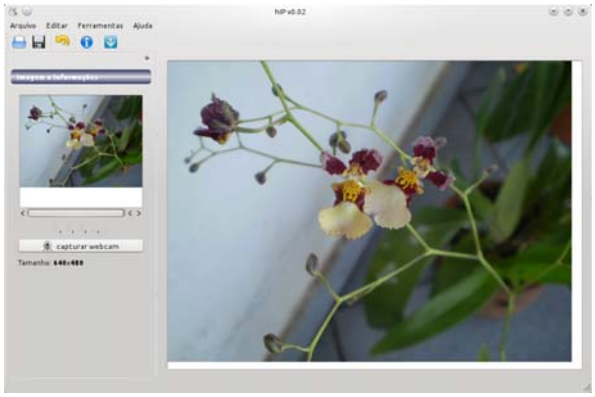


Figura 5: Tela principal do sistema.

Conforme ilustram os exemplos apresentados e dentro do que era esperado, o sistema está operacional. Em síntese, pode-se afirmar que os objetivos iniciais foram alcançados e os resultados preliminares estimula a continuidade do trabalho com a incorporação de novas funcionalidades, entre elas, a classe de algoritmos para tratar o problema de afinamento (*thinning*) alargando ainda mais o escopo e usabilidade do sistema, para num momento posterior, disponibilizá-lo publicamente, já que ambas as bibliotecas usadas são livres e de código aberto.

Referências

[1] GONZALEZ R. C.; WOODS, R. E. **Digital image processing**. 3rd edition. ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2006.

[2] PEDRINI H.; SCHWARTZ, W. R. **Análise de imagens digitais**: Princípios, algoritmos e aplicações.

São Paulo: Thomson Learning, 2007.