Estudo de comparação do descritor de imagens coloridas BIC empregando diferentes abordagens de classificação de detecção de bordas: Canny e Operador Laplaciano

Diego Martin Mancini

Orientador Prof. Dr. Paulo A. V. de Miranda

Trabalho de Formatura Supervisionado

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Objetivo

Este trabalho de formatura supervisionado tem como objetivo aplicar os conhecimentos adquiridos no curso de Bacharelado em Ciências da Computação do IME-USP para descrever e avaliar o desempenho com relação à eficácia e a eficiência do algoritmo *BIC (Border/Interior pixel Classification)* [13].

Tal avaliação será feita através da comparação entre variações do algoritmo supracitado com algoritmos clássicos da literatura de pré-processamento de imagens que podem ser empregados nas diferentes etapas do *BIC*.

1.2 Motivação

Com o advento de novas tecnologias de aquisição de imagens, a necessidade de gerenciar e manipular grandes volumes de informações de multimídia tem crescido nos últimos anos na internet. Para tal propósito, é necessário ter um sistema de recuperação de imagens baseado em conteúdo (*Content-Based Image Retrieval - CBIR [3] [12]*) que permite a recuperação de imagens com base nas características de cor, textura e forma dos objetos presentes em uma imagem de consulta.

Os descritores mais utilizados para a obtenção de domínios amplos de imagens usam informação de cor. Esses descritores de imagem baseados em cor são, geralmente, divididos em três principais categorias: com base global (GB) com base em partição (PB); e com base em região (RB) (veja Seção 2.1) .

Os primeiros incluem descritores espaço-invariantes, que globalmente representam a distribuição de cores das imagens, como os histogramas de cores globais (GCH) [6] [1], correlograma de cor, vetores de coerência de cor (CCV) [11] e classificação entre borda/interior (*BIC*) [13]. Por outro lado, descritores com base em partição capturam a distribuição espacial da cor, decompondo a imagem num número fixo de regiões (CCV e LCH [7]).

BIC é um dos mais proeminentes métodos para aplicações de domínio amplo, também

com várias aplicações bem sucedidas em domínio estreito de imagem, incluindo diagnóstico automático de parasitas intestinais, em combinação com outros descritores. Ele também foi combinado com sucesso com outros descritores mais sofisticados, consolidando uma referência para a extração de características de baixo nível. Neste sentido, também pode servir como um pilar de referência para outros descritores mais sofisticados com base em informação de médio - e alto - nível.

No entanto, algumas soluções adotadas nas difentes etapas do *BIC* não consideram o uso de algoritmos mais clássicos de processamento de imagens que poderiam em teoria ser utilizados com a mesma finalidade.

Sendo assim, este trabalho tem como objetivo realizar essa comparação com os algoritmos mainstream e testar, se de fato, o algoritmo *BIC*, tal como proposto em [13], é tão eficaz quanto sua versão utilizando os algoritmos de suporte mais tradicionais.

1.3 Justificativa

Entre as aplicações de técnicas *CBIR*, podemos citar a recuperação de imagens baseados em categorização, que em geral, necessita de um grande volume de imagens. Logo, é necessário que existam algoritmos de processamento de imagens eficientes e rápidos para realizar tal processamento.

Pensando nisso, a necessidade de realizar experimentos para detectar possíveis melhoras de métodos computacionais vem aumentando ao longo do tempo. Além de verificar a eficácia do descritor de cores *BIC*, o intuito desse trabalho é indicar possíveis novos métodos que visam melhorar esse descritor.

2 Descritores de imagens

2.1 O método CBIR

O método CBIR (Content-based image retrieval) ou simplesmente *Recuperação de Imagens Baseado em Conteúdo* é uma aplicação de técnicas de visão computacional que tem como objetivo resolver problemas de recuperação de imagens digitais em grandes bases de dados. O termo "Content-Base"significa que a recuperação de imagem analisa o conteúdo da imagem em vez de metadados, como keywords (palavras-chave) ou tags (termo muito usado em páginas web) que são descrições associadas à um determinado grupo de imagens. Mais especificamente, o termo "Content", nesse contexto, pode se referir a cores, formas, texturas ou qualquer outra informação que pode ser derivada da própria imagem. Por esse motivo, o método CBIR é vantajoso pois a recuperação de informação das imagens não depende somente de metadados, analisando a imagem em toda sua completude.

Existem várias técnicas CBIR que podem ser empregadas, tais como relevância de feedback (interação humana com máquina), técnicas de consulta em um sistema de recuperação de informação ou comparações de contéudo de imagens usando uma medida de distância. Nesse trabalho, será adotado a técnica de comparações de conteúdo de imagens usando uma medida de distância, pois a análise será feita através de recuperação de informação de cores das imagens, usando uma medida de distância entre várias imagens (sempre dois à dois). Tal distância é chamada de distância dLog, que será descrita na Seção 2.3.3.

2.2 Explicação conceitual dos descritores

Encontrar padrões em imagens digitais têm sido muito significativo para diversas aplicações computacionais. Os descritores de imagem são métodos computacionais que têm como objetivo extrair o conteúdo abstrato das imagens para variadas finalidades.

A avaliação experimental de métodos de *CBIR*, depende de algumas etapas importantes. Obviamente, a aquisição das imagens de teste é um passo fundamental e isso será feita através da base pública do *ETH* [8], que será descrito na Seção 3.2.

Depois de concluída essa etapa, é preciso realizar o pré-processamento dessas imagens para diminuir ruído, borramento e artefatos. Em seguida, será feita a escolha de uma metodologia para comparar diferentes imagens por alguma medida de semelhança. Essa metodologia envolve a extração de um vetor de características e a definição de uma função de distância entre vetores. Um vetor de características é um histograma de cores que será definido na Seção 2.3.2. O par vetor e função de distância definem um descritor de imagem.

Algoritmos CBIR baseiam-se em descritores abstratos durante a fase de análise das imagens e normalmente usam diferentes domínios de imagens. Tipicamente existem dois tipos de domínios: *Domínio estreito de imagem* (narrow image domain) e *Domínio amplo de imagem* (broad image domain).

Domínio estreito de imagen tem uma variabilidade limitada e previsível em todos os aspectos relevantes da sua aparência. Exemplos desse tipo de domínio são as coleções de impressões digitais e raios-X do cérebro humano.

Por outro lado, *Domínio amplo de imagem* é o oposto do domíno citado acima, possuindo uma variabilidade ilimitada e imprevisível com relação ao conteúdo das imagens. De modo geral, as interpretações desses conteúdos não são únicas e as coleções de imagens são muito grandes. Devido a esse detalhe, é impossível usar técnicas semi-automáticas para comparação de imagens durante a fase de análise.

Como o *BIC* é um descritor de propósito geral, voltado para uma coleção grande de imagens, o conceito de domínio amplo de imagem será adotado nesse trabalho. Tendo isso em vista, as características visuais de baixo nível (como cor e textura) são suficientemente úteis para a comparação das imagens.

A cor é o item mais utilizado por diversos grupos de descritores em sistemas CBIR, uma vez que o ser humano tem mais facilidade em diferenciar imagens através dessa característica. As imagens que são manipuladas por sistemas CBIR baseados em cores podem ser separadas em três categorias:

Localidade Global (Global based): descreve o conteúdo da imagem como um todo, considerando cada pixel de cor da imagem de uma só vez.

Localidade em Partições (Partition based): decompõe a imagem em um número fixo de regiões e analisa cada região separadamente.

Localidade em Regiões (Region based): encontra grupos similares de pixels a fim de decompor a imagem em regiões. Não há um padrão de tamanho para cada região. Em geral essa categoria busca a divisão em regiões que seriam facilmente distinguíveis por um humano ao olhar a imagem.

De maneira generalizada, é difícil usar algoritmos *CBIR* de Localidade em Regiões devido ao fato que o número das regiões segmentadas é muito alto e, consequentemente, torna as comparações envolvendo imagens com alto grau de precisão muito custosas computacionalmente, uma vez que o número de imagens processadas é muito elevado. Uma maneira de contornar esse problema é diminuir o número de segmentos durante a fase de análise. No entanto, isso pode acarretar um uma perda significativa de eficácia nas comparações.

Pelos motivos explicados nessa seção, foi escolhido um algoritmo *CBIR* que tenha um custo computacional relativamente baixo, rápido e sem perda de informações relevantes das imagens, ou seja, sem exclusão de regiões segmentadas por conta da simplificação. Nas seções seguintes será explicado o algoritmo *BIC* que contém essas propriedades.

Para além do que foi descrito acima sobre o propósito do trabalho, também serão descritos os algoritmos empregados nas variações do método *BIC*.

2.3 Descritor BIC

2.3.1 Espaço de cores RGB e definição de pixels vizinhos

O nome RGB é a abreviação para Red (vermelho), Green (verde) e Blue (azul) e, em processamento de imagens, as cores no espaço RGB de uma dada imagem podem ser descritas como:

1) As imagens são definidas por um conjunto de pixels. Cada pixel possui uma cor que possui três canais que são Vermelho, Verde e Azul. Cada canal tem um valor mínimo e máximo, que geralmente varia entre 0 (mínimo) à 255 (máximo).

2) Para obter a cor do pixel em questão, é necessário a combinação dos valores de cada canal, que geralmente é representada por (R,G,B), onde as letras R, G e B são valores que variam conforme descrito em 1).

Como pode-se perceber, o número total de combinações é da ordem de 256^3. Esse número contém cores que não foram citadas ainda como, por exemplo, o branco e amarelo. Com isso, podemos obter o vermelho completamente intenso com a combinação (255,0,0) ou ainda o branco, com (255,255,255).

Em processamento de imagem, é comum usar-se o termo "os n vizinhos do pixel", onde n pode ser qualquer número natural (em geral é 4 ou 8). Logo, o termo vizinho será definido da seguinte forma: fixado um pixel p_i, os pixels mais próximos p_j (ao redor) de p_i são seus vizinhos. Para 4 vizinhos de um pixel, serão considerados os pixels de cima, baixo, esquerda e direita.

2.3.2 BIC

Desenvolvido por *Stehling et. al.* [13], esse método utiliza o espaço de cores RGB uniformemente quantizado em um número de cores, onde será reduzido de $256 \times 256 \times 256$ para $4 \times 4 \times 4 = 64$. Este descritor gera uma representação compacta e separa os dados da imagem (que são as cores dos pixels após a quantização de cor) em dois histogramas de cor, um para os pixels de borda e outro para os pixels de interior. Um pixel é considerado interior se seus 4 vizinhos tiverem a mesma cor (após a requantização da imagem), caso contrário será borda (figura 1). A vizinhança considerando 4 elementos é usada em vez da vizinhança de 8 elementos, pois diminui o custo computacional sem perdas percentuais em termos de acurácia. Em seguida, os dois histogramas de cores são concatenados e o resultado disso - um vetor de características - representa as características extraídas pelo *BIC*.

Objetos grandes com formas regulares possuem, em geral, mais interior do que borda, conforme pode ser visto na figura 1. No entanto, se o número de pixels de interior de uma cor é menor que os de bordas, então pelo menos uma das características abaixo é verdadeira:

1) A cor é distribuída em regiões relativamente grandes e com formato irregular (figura 2).

2) A cor é distribuída em pequenas regiões conexas tais que a borda de cada região é maior que seu interior (figura 3).

3) A cor é parte de uma região rica em informações de textura (figura 4).

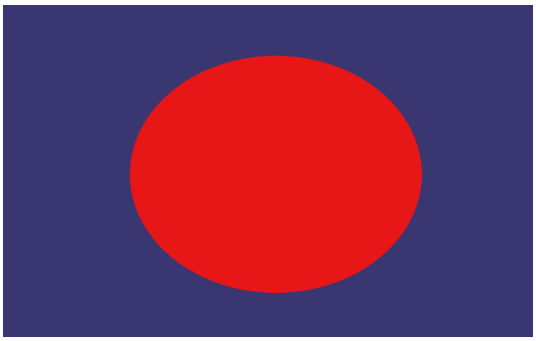

Figura 1: Por ser regular, o objeto de cor vermelho da figura acima possui muito mais interior que borda.

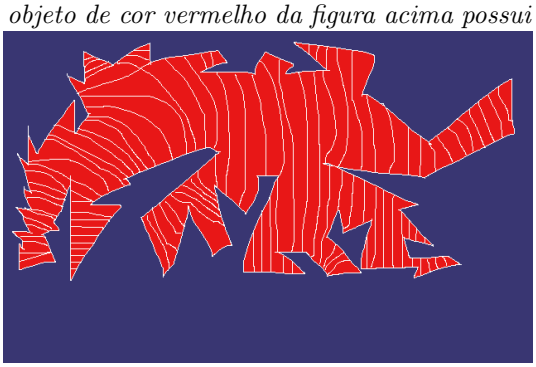

Figura 2: Nessa figura, a cor vermelha possui formatos irregulares e é distribuida em uma região relativamente grande.

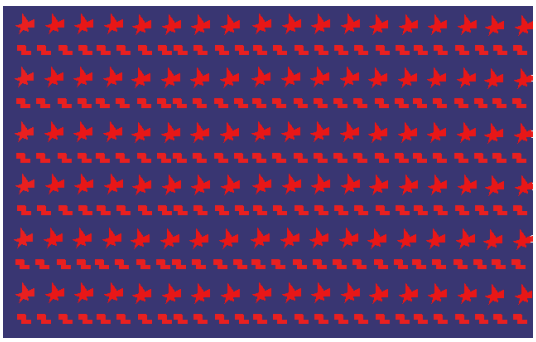

Figura 3: Na figura acima, a cor vermelha é distribuída em regiões conexas relativamente pequenas, onde possui um maior número de bordas que interior.

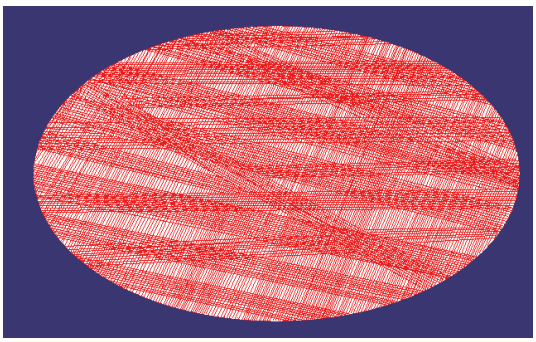

Figura 4: As texturas são representadas pelas várias linhas em vermelho dentro da elipse da figura acima

Depois de construído o vetor de características, é necessário um método para comparar os histogramas de cores. Esse método é a *distância dLog* que será descrito em seção posterior.

Abaixo seguem algumas figuras geradas pelo descritor de cores *BIC*, bem como o seu pseudocódigo.

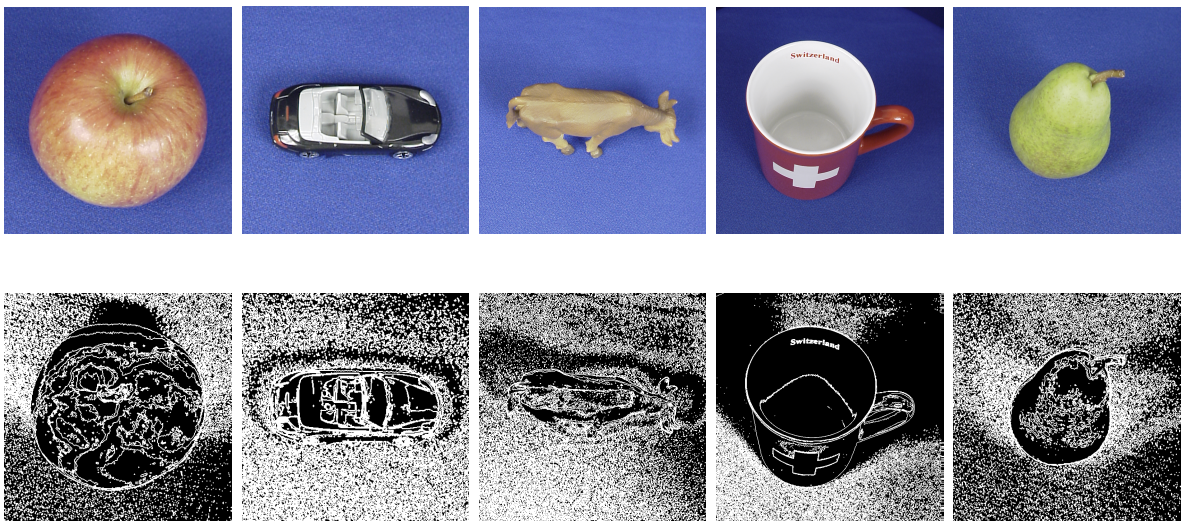

Figura 5: Resultados da classificação de detecção de bordas usado na 2º etapa do BIC (imagens inferiores) para diferentes entradas de imagens (imagens superiores).

Algorithm 1 BIC

Entrada:

img é a imagem de tamanho $n \times m$.

Saída:

o histograma de características de cor (vetor).

função BIC(img)

/*$prop$ é o vetor que conterá as cores em 64 níveis e a classificação de interior e borda.*/

$Properties\ prop$;

$prop \leftarrow quantiza\_pixels(img)$ /*Quantifica as cores em 64 níveis (4x4x4).*/

$prop \leftarrow borda\_interior(prop)$ /*Classifica os pixels em interior e borda.*/

$vetor \leftarrow cria\_histograma(prop)$ /*Cria o histograma de interior/borda.*/

devolve $vetor$

fim função

No algoritmo acima, a variável vetor é o histograma construído pelo BIC. Essa variável contém dois eixos x e y. Cada posição do eixo x representa uma cor quantizada em 64 níveis, de 0 à 63 para os pixels considerados interior e 64 à 127 para borda. Já o eixo y, cada posição desse eixo representa a quantidade da *i-ésima* cor representada no eixo x.

2.3.3 Distância dLog

A comparação entre dois histogramas de cores com valores muito distantes pode gerar um valor muito grande na distância, como por exemplo no caso de imagens que possuem uma cor predominante. Isso é notadamente possível quando se é empregado um método não apropriado para a função de distância como, por exemplo, o cálculo de distância Euclidiana.

Como o resultado da distância é a soma das diferenças, um valor muito alto tende a dominar a soma e encobrir resultados das outras cores, que apesar de serem menos frequentes, podem ser importantes para diferenciar imagens . A *dLog* (distância Logarítmica) tem como principal objetivo atenuar este problema, fazendo com que as diferenças fiquem sempre entre 0 e 9, dando igual importância para todos os valores de um histograma de cores na comparação.

Desta forma são gerados dados de menores valores, em geral da ordem de metade do espaço que seria ocupado pelo método da distância vetorial. Essa distância é dada pela

função:

$$dLog(q,d) = \sum_{i=0}^{i<M} [f(q[i]) - f(d[i])]$$

$$f(x) = \begin{cases} 0, & \text{se } x = 0 \\ 1, & \text{se } 0 < x \leq 1 \\ [log_2 x] + 1 & \text{caso contrário} \end{cases}$$

Onde q e d são histogramas de cores com M cores cada. A *dLog* é considerada a função de distância padrão para o descritor *BIC*.

2.4 Variações do BIC usando detector de bordas de Canny e Operador Laplaciano

Seguindo os passos descritos anteriormente - classificação do pixel como interior ou borda e construção de um histograma de cor quantizados em 64 níveis para cada classificação de interior/borda, ou seja, o histograma possuirá os primeiros 64 níveis classificados como interior e os 64 níveis restantes como borda, totalizando um histograma de 128 níveis. Esse histograma irá conter as características extraídas. Entretanto, o algoritmo para classificar em interior/borda do *BIC* será substituído pelos algoritmos de Canny e Operador Laplaciano. O resultado dessa substituição será usado para construir novos histogramas de cor para interior/borda. Nas duas seções seguintes serão descritos os métodos de Canny e Laplaciano.

2.4.1 Detector de bordas de Canny

O detector de bordas de Canny (Canny Edge Detection) foi desenvolvido por John F. Canny em 1986 [2] e segue os três critérios seguintes.

1) Boa Detecção - O algoritmo deve ser capaz de identificar todas as bordas possíveis na imagem.

2) Boa Localização - As bordas encontradas devem estar o mais próximas possíveis das bordas reais da imagem original.

3) Cada borda da imagem deve ser marcada apenas uma vez. O ruído da imagem não deve criar falsas bordas.

Para isso, qualquer tipo de ruído deve ser obrigatoriamente eliminado. Então será aplicado um filtro, que é uma técnica de transformação (máscaras) com o objetivo de corrigir (remoção de caractérísticas indesejáveis), suavizar ou realçar (atenuar) determinadas características de uma imagem, onde é feita pixel à pixel. O filtro mais adequado para o problema em questão é o filtro Gaussiano. Um exemplo desse tipo de filtro é dado abaixo:

$$B = \frac{1}{159} \times \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A$$

Onde A é a matriz de uma dada imagem e o operador * indica a convolução. O passo seguinte é encontrar o gradiente da imagem. Isso será feito seguindo um procedimento análogo ao de Sobel [4].

1) Aplicar um par de máscara de convolução nas direções (x e y), que será dado por:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * B$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * B$$

Onde o operador * denota a convolução.

2) Calcular a intensidade do gradiente e direção com as seguintes equações:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan(\frac{G_y}{G_x})$$

A direção deve ser arredondada para um dos ângulos a seguir : 0, 45, 90 ou 135. Esses ângulos são representados pelas direções verticais, horizontais e duas diagonais. A direção da borda deverá estar em uma região de cor definida por valores de ângulos especificados. Por exemplo, se um ângulo theta (θ) cair na região amarela (de 0º a 22,5º e 157,5 a 180º), a direção terá valor 0º.

A *supressão não máxima* deverá ser então aplicada ao longo da direção do vetor gradiente. Isso irá remover pixels que não são considerados parte da borda, permanecendo somente os candidatos à borda.

Essa *supressão não máxima* se baseia na eliminação de pixels cujos valores não são máximos locais, ou seja, busca-se na direção do gradiente da imagem e se o pixel não faz parte dos máximos locais, então ele é definido como zero, suprimindo assim toda informação que não faz parte do máximo local.

Finalmente, o Canny usará dois limiares (thresholds) superior e inferior (upper and lower), segundo as seguintes regras:

1) Se o gradiente do pixel é maior do que o limiar superior, o pixel em questão é considerado borda.

2) Se o valor do gradiente do pixel é abaixo do limiar inferior, então o pixel é rejeitado.

3) Se o gradiente do pixel está entre os dois thresholds, então este pixel somente será aceito se estiver conectado a um pixel que estiver acima do limiar superior.

Esses passos descritos acima classificarão os pixels em interior ou borda, conforme mostrado na figura 2 (logo a seguir).

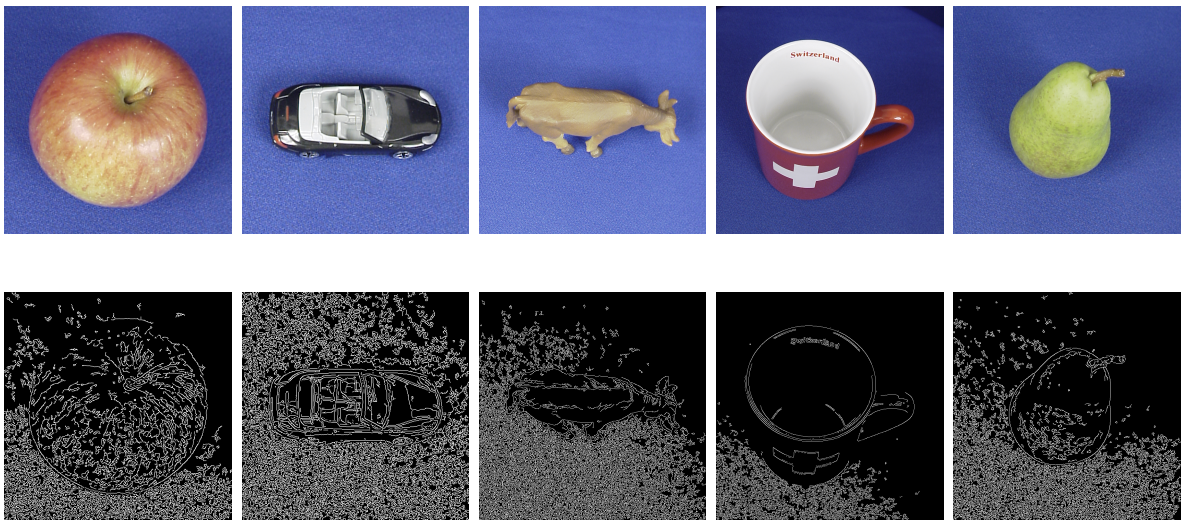

Figura 6: Resultados da classificação de detector de bordas Canny (himagens inferiores) para diferentes entradas de imagens (imagens superiores).

Os valores obtidos pelo detector de bordas de Canny já são uma representação binária de cores, ou seja, os pixels já são definidos como bordas ou interiores. Por essa razão, não será necessário demonstrar o código nesse trabalho, já que função de detecção de bordas de Canny é fornecida pela biblioteca do openCV [10].

2.4.2 Operador Laplaciano

Este método utiliza o operador de Sobel [4] internamente para classificar a intensidade de cada pixel. Essa intensidade calculada do pixel, que está na região de borda, resultará em uma alta variação de intensidade e será usada pelo operador Laplaciano. Essa intensidade é, na verdade, a magnitude do gradiente de Sobel, que reflete uma alta variação de brilho na região da imagem.

Obtendo a primeira derivada, podemos observar que a aresta têm um valor máximo, que é a alta variação da intensidade, conforme mostra a figura abaixo:

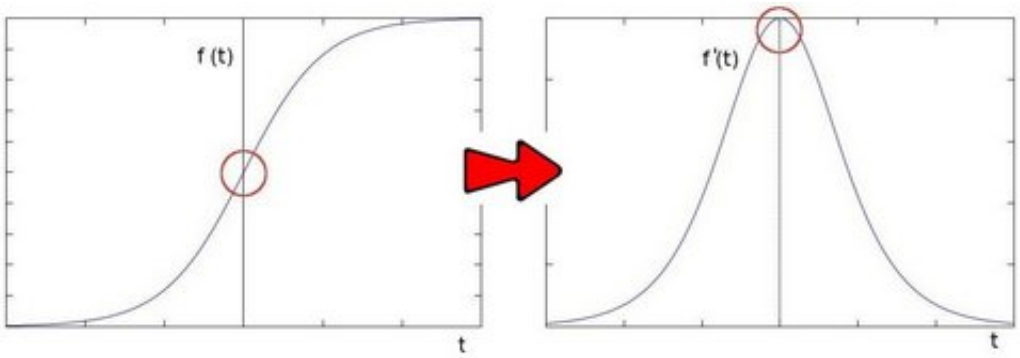

Figura 7: Primeira derivada referente à variação de intensidade do Operador Laplaciano.

Para a segunda derivada, temos o seguinte:

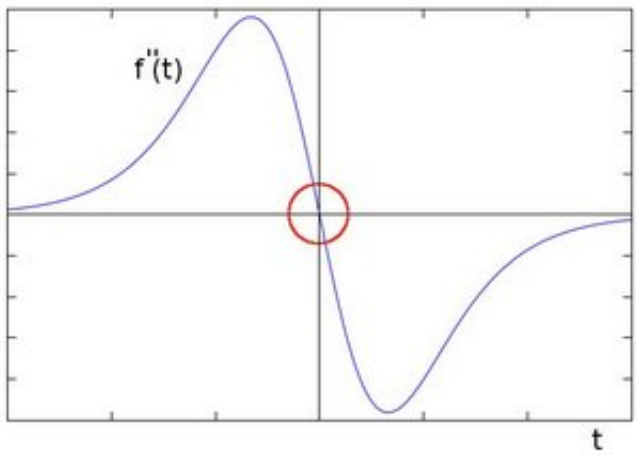

Figura 8: Segunda derivada referente à detecção de bordas do Operador Laplaciano

Pode-se observar que a segunda derivada é zero. Logo, é possível usar esse fato para tentar detectar as arestas na imagem. No entanto, os zeros não aparecem somente nas arestas, podendo aparecer em outros locais não relevantes. Para resolver esse problema, será aplicado um filtro onde for necessário.

Conforme explicado acima, a segunda derivada será usada para detectar as bordas. Como as imagens são em 2 dimensões (2D), teremos que usar a segunda derivada em ambas as dirmensões e o operador Laplaciano será determinado por:

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2}$$

Um exemplo do operador laplaciano é mostrado abaixo:

$$I = \begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 \\ 3 & 4 & 5 & 5 & 5 & 5 & 5 \\ 3 & 3 & 4 & 5 & 5 & 5 & 5 \\ 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

$$k_a = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \; I_k a = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ -2 & 0 & 2 & 1 & 1 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}$$

$$k_b = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \; I_k b = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 \\ 0 & 4 & 1 & 0 & 0 \\ -4 & 0 & 5 & 3 & 3 \\ -1 & -4 & 2 & 0 & 0 \\ 0 & -1 & -2 & -3 & -3 \end{bmatrix}$$

No exemplo acima, a image A (representada pela matriz I) é processada pelo operador Laplaciano utilizando dois kernels k_a e k_b, gerando duas novas matrizes que serão usadas para a classificação binária de interior/borda. Essa classificação será feita através de uma técnica denominada *zero crossing*, que irá ser detalhada na seção seguinte e demonstrado na figura 3 (logo a seguir).

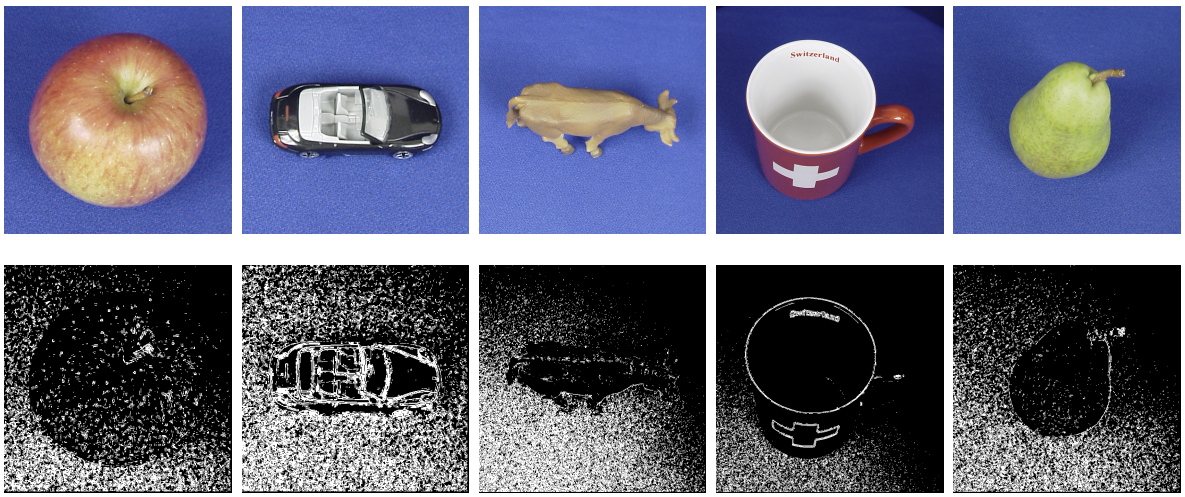

Figura 9: Resultados da classificação de detector de bordas usando o Operador Laplaciano para diferentes entradas de imagens.

2.4.3 Zero Crossing do Operador Laplaciano

O *zero crossing* [14] utiliza uma janela (kernel) para detectar se o pixel é interior ou borda. Essa classificação utiliza tal janela (que pode ser $3 \times 3, 5 \times 5, 7 \times 7$) para verificar os valores máximos e mínimos da vizinhança de um dado pixel. Esses valores são denomidados polaridade máxima e polaridade mínima. As respectivas polaridades serão utilizadas da seguinte forma:

1) Um limiar (threshold) é definido para verificar as polaridades.

2) Se a polaridade mínima for menor que o valor negativo do limiar e a polaridade máxima for maior que o valor positivo do limiar, então o pixel escolhido é zero crossing, ou seja, borda.

3) Caso contrário, o pixel será interior. Note que o pixel em questão não tem que ser igual a zero.

O limiar é necessário para evitar uma falsa detecção de um zero crossing. Isso pode ocorrer devido a uma eventual existência de um ruído aleatório. Sendo assim, se a condição (2) for verdadeira, o pixel é borda ou caso contrário o zero crossing que foi determinado pelo ruído é desconsiderado.

O pseudocódigo do zero crossing é definido abaixo:

Algorithm 2 Zero Crossing

Entrada:
imagem_entrada de tamanho $n \times m$.
MAX é o maior número inteiro.
th é o valor do limiar.

Saída: imagem_saida com zero crossing.

```
função ZERO_CROSSING
    para i ← 1 até m-L faça /*L é o tamanho da janela (kernel) L × L.*/
        para j ← 1 até n-L faça
            min ← MAX
            max ← −min
            para k ← 1 até L faça
                para l ← 1 até L faça
                    w ← imagem_entrada[i + k][j + l]
                    se w < min então
                        min ← w
                    fim se
                    se w > max então
                        max ← w
                    fim se
                fim para
                se min < −th&&max > th então
                    imagem_saida[i][j] ← 255
                senão
                    imagem_saida[i][j] ← 0
                fim se
            fim para
        fim para
    fim para
fim função
```

3 Descrição do Experimento

3.1 Experimento

O foco dessa seção será descrever inteiramente os 'passos' do experimento desse trabalho de uma maneira simples e de fácil compreensão.

Antes de tudo, precisaremos de um conjunto de imagens adotadas para a realização do experimento. Tais imagens possuem uma classificação previamente estabelecida, que será usada para calcular o gráfico de *precision* x *recall*. Esse gráfico será definido na Seçao 3.3.

Depois de selecionada o conjunto de imagens de tamanho n, serão aplicados os algoritmos *BIC* e suas variações para se obter os histogramas de cores de cada imagem. Para efetuar as comparações, os seguintes passos serão feitos:

1) Cada algoritmo (o próprio *BIC*, por exemplo) fixará uma imagem A do conjunto e obterá o seu respectivo histograma de cor h_1.

2) Em seguida, calculará o histograma de cor h_2 de uma imagem qualquer B do conjunto (podendo ser a própria imagem A) e utilizará os histogramas de cores h_1 e h_2 para calcular a distância *dLog*, sendo essa distância o critério de comparação entre as duas imagens.

3) Esse passo continuará até que o histograma de cor h_1 compare com todos os histogramas de cores $h_i, i = 1, ..., n$.

4) Os passos (1), (2) e (3) são repetidos para o histograma de cor $h_2, h_3, ..., h_n$.

Para cada imagem, é criado um arquivo texto (.txt) no passo (1) que será usado para guardar os valores dados pelos passos (2) (distância *dLog* dos histogramas de cores h_i e h_j) e (3).

Obtidos os valores das comparações, será criado o gráfico de *precision* x *recall* para cada algoritmo, onde teremos uma comparação definitiva do desempenho dos métodos *BIC* e suas variações. Essa análise pode ser feita de maneira diferente. Uma abordagem comumente utilizada é simplesmente analisar o gráfico (*precision* $\times$ *recall*) e verificar se há contraste visível. Outro tipo de análise seria calcular a área do gráfico de cada método e comparar seus valores. Ou, ainda, fixar um valor do eixo recall e obter o valor de precision de cada

método e, depois, fazer a comparação de seus valores.

A escolha entre esses tipos de abordagem será realizada somente depois de obtermos os gráficos de *precision* $\times$ *recall* de todos os algoritmos.

3.2 Imagens

O conjunto de imagens foi retirado do site *Max planck institut informatic* [8] e contém mais de 3500 imagens, divididas em 89 classes diferentes. Para o experimento, foi usado um sub-conjunto de mais de 2200 imagens com 54 classes, onde as imagens são de extensão png e foi necessário a utilização do software *Image Magic* [5] para converter as imagens de png para ppm, sem afetar sua qualidade e para evitar viés no resultado do experimento. Essa conversão foi necessária pois as estruturas de dados do código fonte do BIC citado no artigo original [13] só aceita imagens de entrada no formato ppm. A biblioteca *openCV* [10] foi utilizada para a utilização dos algoritmos de detecção de bordas Canny e o Operador Laplaciano.

Para cada classe de imagens, teremos 41 imagens de tamanhos $n \times n$, onde n está entre 447 e 800 pixels. Uma classe de imagem é definida como: um conjunto de m imagens de um mesmo objeto, onde $m = 41$ e cada uma das m imagens desse objeto são obtidas de vários ângulos (posições).

Os objetos de cada classe não precisam ser necessariamente distintos por completo. Podemos ter, por exemplo, uma classe A de imagens maçãs vermelhas e outra classe B de maçãs verdes.

3.3 Gráfico de Precision x Recall

Este gráfico [9] nos dará a comparação de acurácia dos algoritmos *BIC* e suas variações. A construção do gráfico consiste em dois eixos: precision e recall. Ambos os eixos tem valores reais entre 0 e 1. A definição de cada eixo segue da seguinte forma:

Precision: Seja n_i o número total de imagens processadas na i-ésima comparação. Conforme já mencionado em 3.1, uma imagem i_i terá seu respectivo histograma de cor h_i e, a partir deste histograma de cor h_i, será comparado com outros histogramas h_j de cores. A cada comparação será calculada a distância *dLog* e, ao final das comparações, teremos um vetor ordenado de distância $dist_i$ da imagem i_i com todas as outras i_j imagens do conjunto.

A partir daí, o eixo Precision será construído da seguinte forma:

1) Temos um contador nac_i, que representa o *número de acertos*, onde os *acertos* é definido da seguinte maneira: para cada par de imagens i_i e i_j pertencentes ao conjunto de imagens, é considerado *acerto* se ambas as imagens estão na mesma classe. No caso do vetor $dist_i$, a *i-ésima* posição desse vetor possui o nome da imagem i_j que foi comparada com a imagem i_i. O nome das imagens é necessário para identificar se as imagens i_i e i_j estão na mesma classe.

2) Se as imagens são da mesma classe, então o contador é incrementado, ou não é incrementado caso contrário. Nesse momento, para obter o valor do eixo precision, será feito

$$Precision(i) = \frac{nac_i}{n_i}$$

Recall: Já o eixo recall, simplesmente teremos n, que será o número total de imagens de cada classe (onde todas as classes possuirão o mesmo número de imagens) e o número de acertos. Logo o eixo recall será definido por

$$Recall(i) = \frac{nac_i}{n}$$

Teremos o i-ésimo valor do eixo recall correspondente ao i-ésimo valor de precision. No final, será obtido o gráfico de precision x recall para a imagem i_i do conjunto. Esse processo será repetido para todas as imagens do conjunto e, como queremos uma única curva de precision x recall para cada descritor, faremos a média das curvas. Sendo PR a média das curvas e pr_i a curva de cada imagem i_i, tal média será dada por :

$$PR = \sum_{j=0}^{j<n} (\frac{pr_i(j)}{n})$$

O gráfico terá uma função tendencialmente decrescente conforme demonstra a figura abaixo:

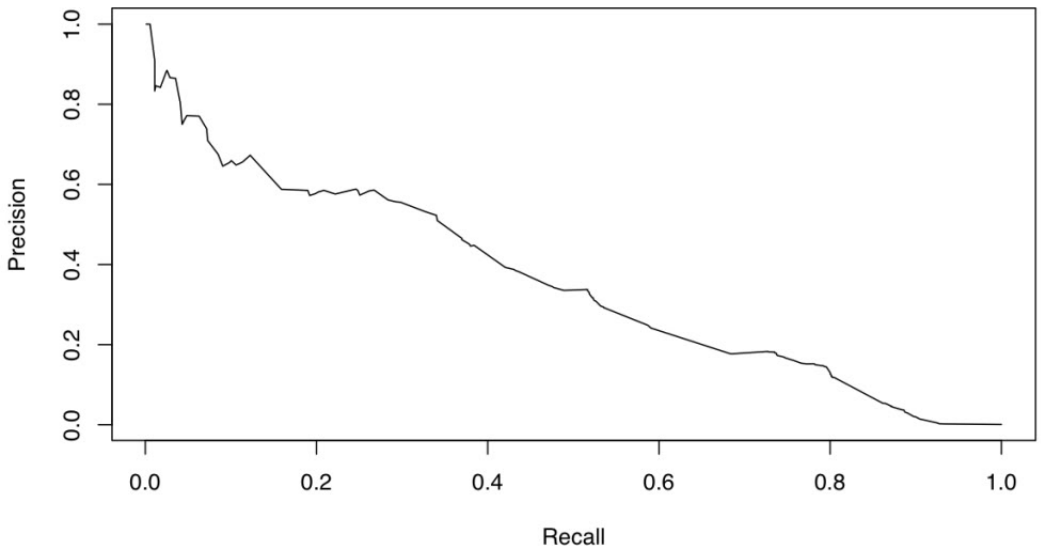

Figura 10: Exemplo de um gráfico de precision x recall

Note que os valores de ambos os eixos estão sempre no intervalo [0,1].

4 Análise do Experimento

Nesta seção discutiremos os resultados do experimento relacionados a acurácia do descritor *BIC*, já citados nas Seção 3.1. Inicialmente, iremos comparar o *BIC* em relação as suas variações com o detector de bordas de Canny e o operador Laplaciano com uma tabela, usando os gráficos de *precision* × *recall* vistos na Seção 3.3. Essa tabela terá alguns valores selecionados do eixo recall com os seus correspondentes valores do eixo precision, onde cada valor selecionado será, obviamente, igual para os três gráficos (*BIC* e suas variações). Essa tabela nos dará uma análise mais precisa na exibição dos valores absolutos.

Recall	BIC	BIC-CANNY	BIC-LAPLACIANO
0,152	0,887599	0,888918	0,888983
0,302	0,728247	0,740355	0,730174
0,441	0,565644	0,572003	0,553242
0,594	0,405760	0,412028	0,368140
0,653	0,347741	0,351984	0,311410
0,711	0,294488	0,297651	0,263169
0,790	0,228215	0,220075	0,203847
0,851	0,178862	0,178936	0,174392
0,920	0,115371	0,119419	0,103064
0,999	0,019069	0,020865	0,021501

Tabela 1: Tabela dos valores de Precision x Recall dos descritores

Em seguida será exibido os próprios gráficos de *precision* × *recall* do *BIC*, *BIC* usando Canny e *BIC* usando o operador Laplaciano. Para isso, exibiremos três gráficos comparando os descritores dois a dois e, por último, um único gráfico contendo as três curvas. Os gráficos mostrará de maneira mais ampla e informativa todo o conteúdo desse trabalho.

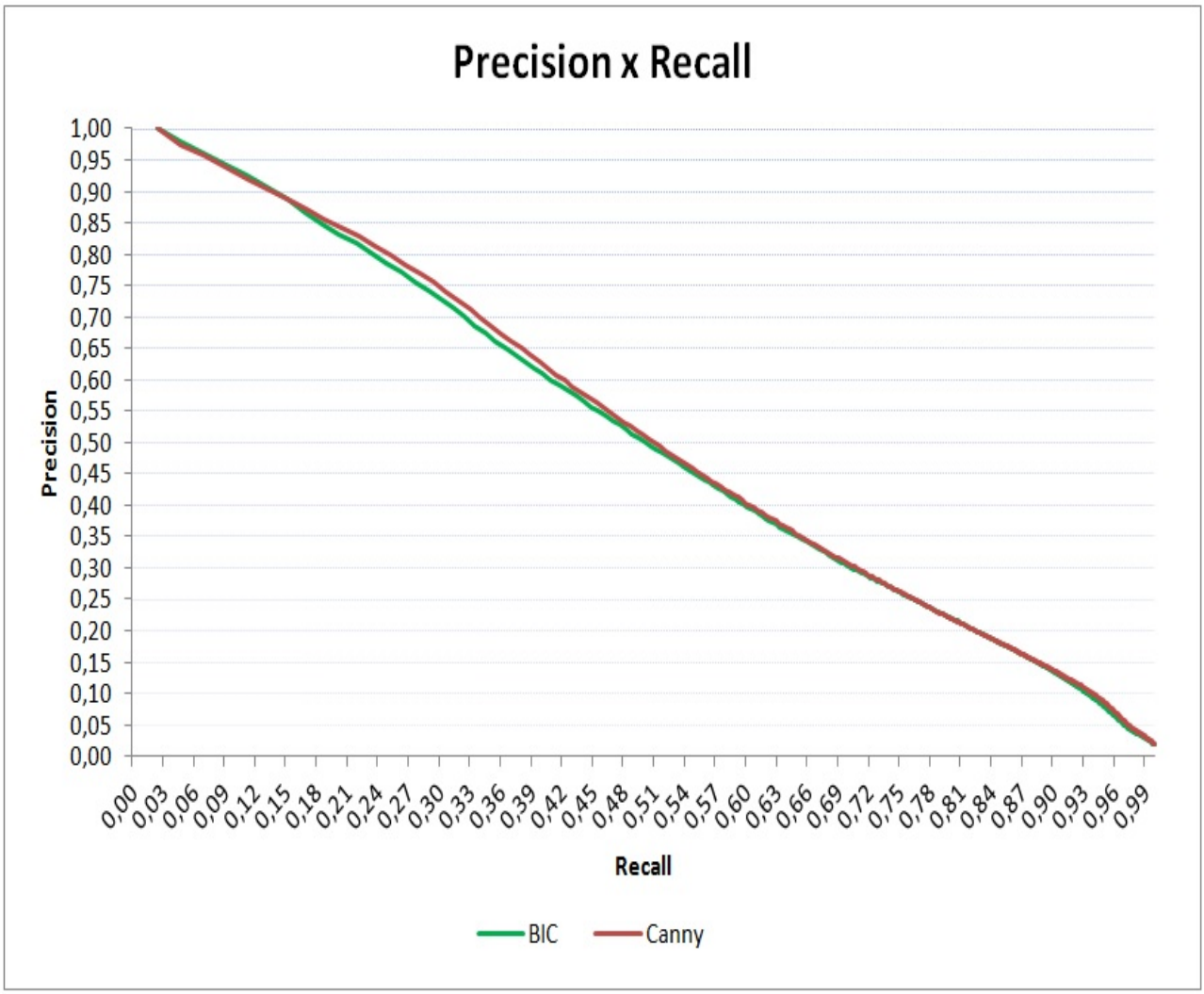

Figura 11: Gráfico de Precision x Recall dos Descritores de cores BIC e sua variação usando Canny

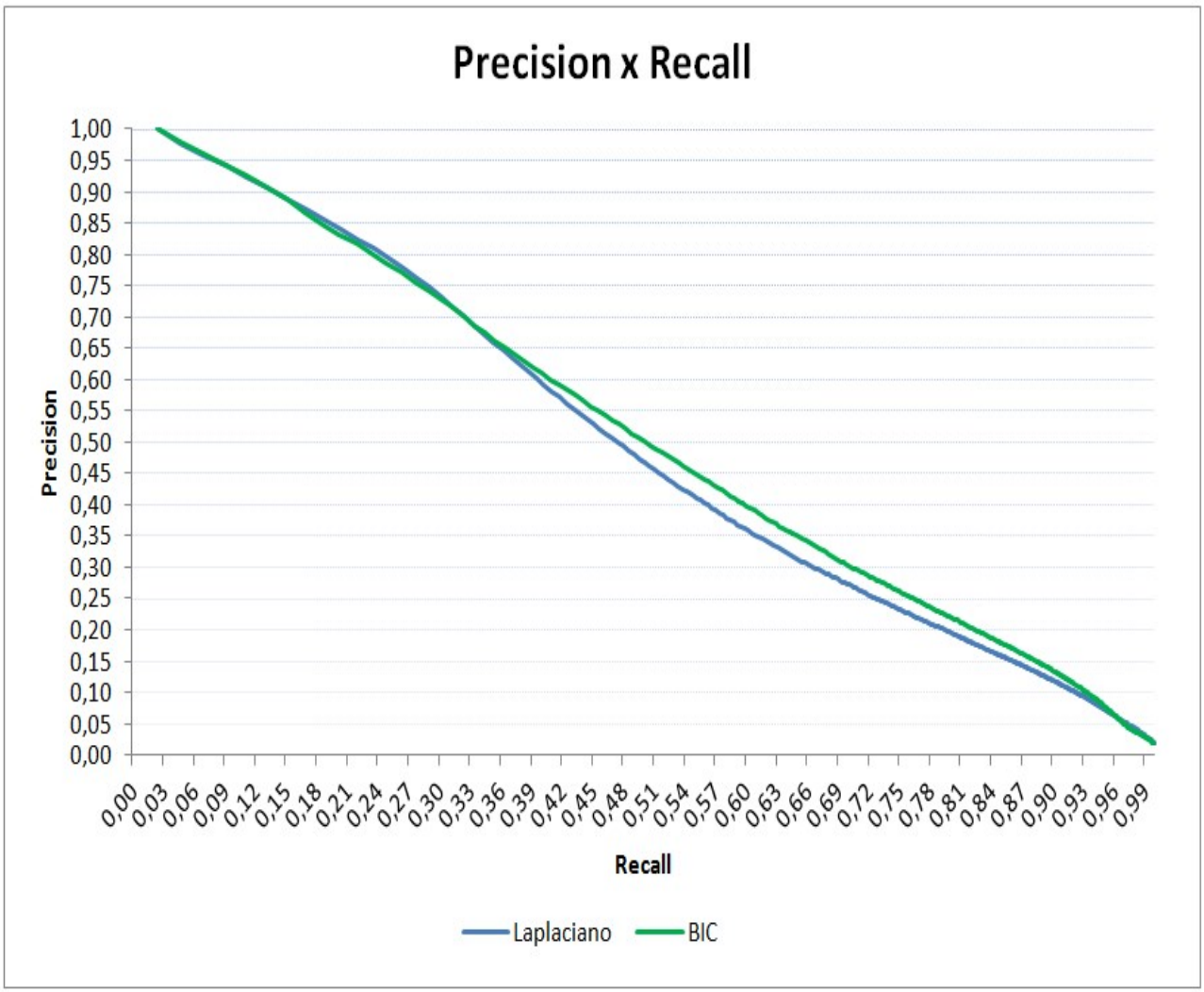

Figura 12: Gráfico de Precision x Recall dos Descritores de cores BIC e sua variação usando o Operador Laplaciano

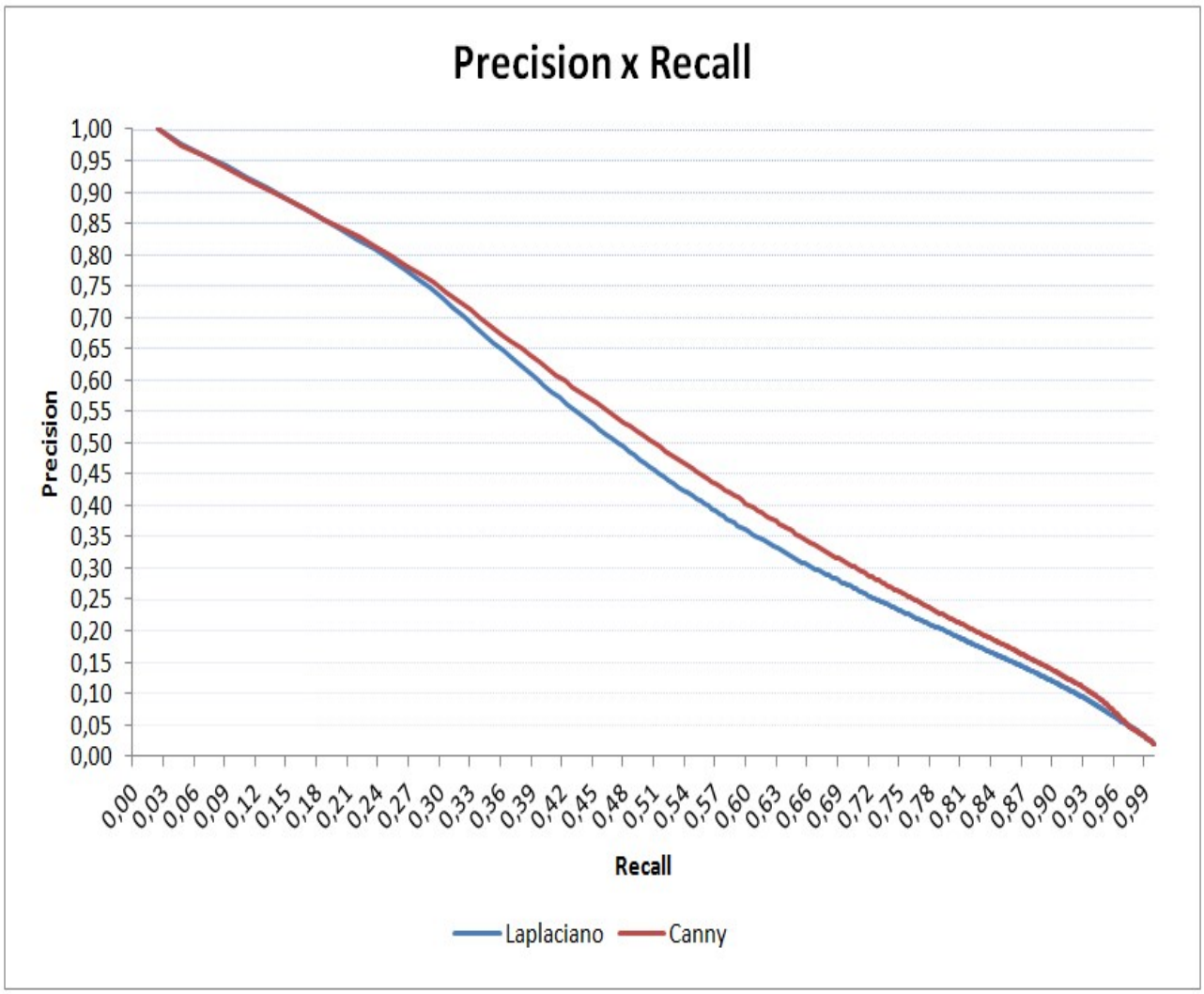

Figura 13: Gráfico de Precision x Recall das variações do Descritor de cores BIC usando os algoritmos de Canny e o Operador Laplaciano

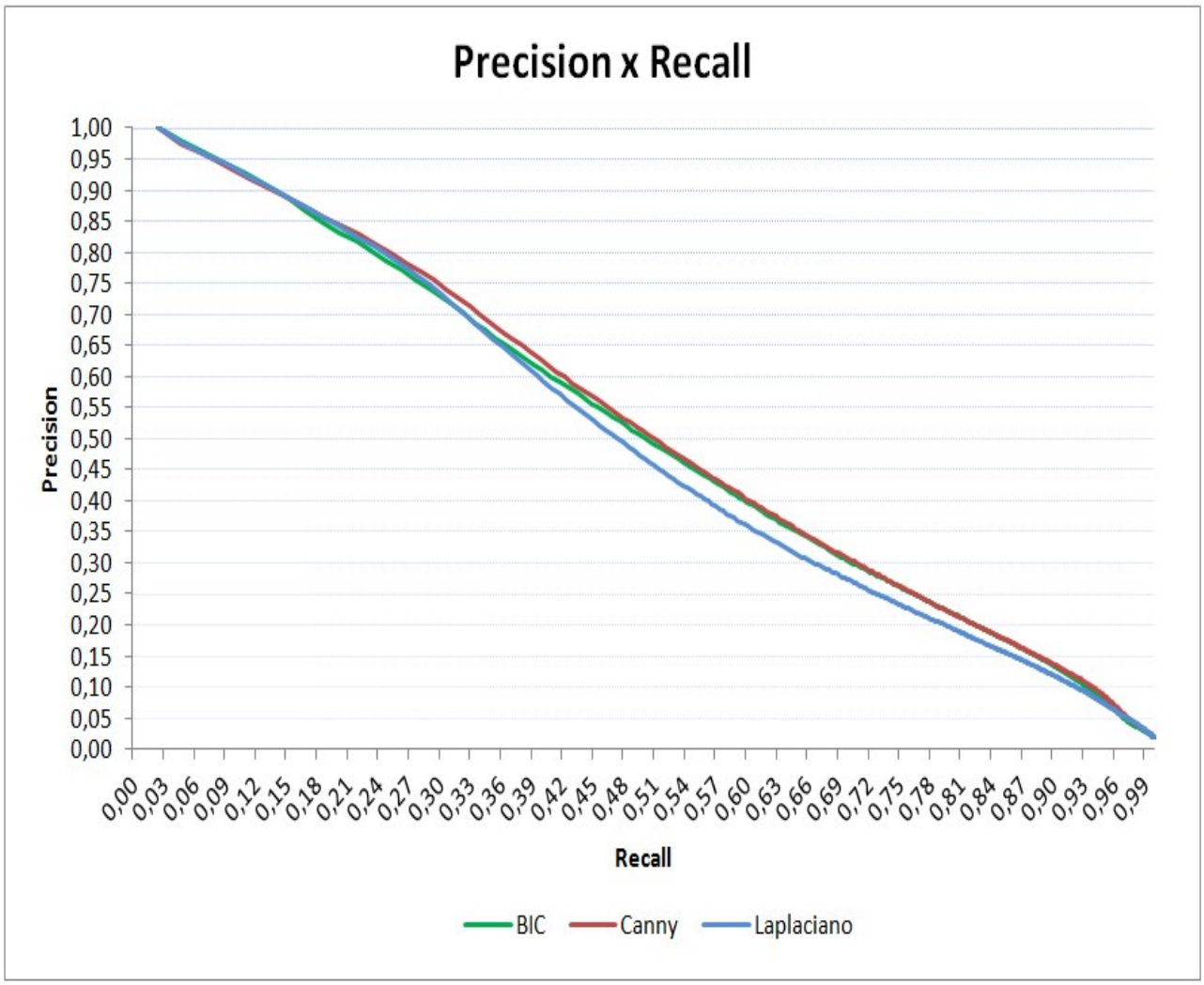

Figura 14: Gráfico de Precision x Recall com os três descritores

O *BIC* usando Operador Laplaciano demonstra uma menor acurácia que os outros dois. Ainda assim, os três descritores demonstram uma performance semelhantes entre si, concluindo-se que o algoritmo do descritor de cores BIC que classifica os pixels como interior e borda é tão eficiente quanto os algoritmos tidos como clássicos.

No entanto, os algoritmos de detecção de bordas Canny e o Operador Laplaciano possuem um limiar (threshold) que são idênticos para ambos os algoritmos nesse experimento, limitação que será melhor explorada na Seção 6.1.

5 Conclusão

Com o objetivo de verificar a acurácia do descritor de cores *BIC*, foram feitas comparações de suas versões alternativas utilizando os algoritmos de detecção de bordas *Canny* e o *Operador Laplaciano*. As substituições dos dois últimos algoritmos anteriormente citados pelo algoritmo de classificação de interior/borda do *BIC* mostraram desempenhos semelhantes nas comparações entre imagens, utilizando a distância dLog para comparar as imagens e, a partir dessas distâncias calculadas para cada descritor, criando-se os gráficos de *precision x recall* para efetuar tais comparações.

Os resultados extraídos tanto da tabela 1 e dos gráficos de *precision x recall* na Seção 3.3, nos revelou que o seu classificador de interior/borda possui uma simplicidade computacional poderosa e eficiente, pois os algoritmos de *Canny* e o *Operador Laplaciano* exigem um custo computacional maior que o do *BIC*, em teoria. No entanto, como será descrito na Seção 6.1, não foram explorados todos os recursos dos algoritmos de *Canny* e o *Laplaciano*. Tal limitação levanta indagações com relação a possíveis resultados diferentes daqueles obtidos nesse trabalho.

6 Trabalhos Futuros

6.1 Sugestões para Aperfeiçoamento do Trabalho

Não diferente de outros experimentos, este trabalho proporcionou alguns aspectos relevantes durante o seu desenvolvimento que não foram aprofundados e que podem ser importantes para a continuidade deste escopo. Com o objetivo de seguir a proposta inicial, ficam aqui listados alguns pontos que podem ser explorados posteriormente:

Implementação do detector de bordas fuzzy

Uma variação do descritor *BIC* que não foi mencionada neste trabalho foi usar o *BIC* somado a ideia de classificação de borda fuzzy ao invés de borda binária. Neste caso, cada pixel pode gerar uma contribuição tanto no histograma de borda como no histograma de região, sendo a contribuição no histograma de borda proporcional à magnitude do seu gradiente. Os gradientes que seriam usados são:

1) Magnitude do Gradiente de Sobel para definir a intensidade de borda,

2) Gradiente morfológico

Além disso, poderiam ser empregadas mais variações dos métodos acima levando em consideração algum esquema de votação interpolada na hora de calcular os histogramas. Por exemplo, os canais de cor no BIC são divididos em 4 níveis. Quando uma cor cai na fronteira de divisão entre dois bins do histograma, no esquema atual só é incrementado o contador de um dos bins. A ideia seria considerar uma votação parcelada de acordo com as distâncias aos centros dos bins (uma cor na fronteira de dois bins poderia contribuir com 0.6 para um e 0.4 para o outro).

Número de imagens

Para um trabalho de maior rigor científico, seria necessário usar um maior volume de imagens, tornando os dados mais consistentes na sua análise para possíveis sugestões de melhora do descritor *BIC*.

Opções variadas do limiar dos algoritmos Canny e laplaciano

Os algoritmos de detector de bordas de canny e o operador laplaciano exigem a escolha do parâmetro o limiar (thresholds) para a sua execução. O limiar já citado neste trabalho foi escolhido pela maior semelhança entre as imagens geradas pelo descritor *BIC* e suas respectivas variações, com o intuito de não enviesar os resultados dos dados. Em contrapartida, seria melhor ter usado limiares diferentes em ambos os algoritmos (Canny e Laplaciano) para ter uma análise mais rica e detalhada, aproveitando melhor os recursos oferecidos por esses algoritmos.

Testes e proposições de melhora de eficiência do descritor BIC

Os testes apresentados nesse trabalho visaram somente a acurácia dos descritores *BIC* e suas variações já mencionados (ver Seção 2.1). No entanto, os testes de eficiência computacional não foram levados em consideração, sendo esse ponto um motivo muito forte para um estudo detalhado em possíveis trabalhos futuros.

Referências

[1] A. D. Bimbo. Visual information retrieval. *Morgan Kaufmann*, 1999.

[2] Wikipedia. https://en.wikipedia.org/wiki/Canny_edge_detector. Acessado em 08-2015.

[3] Content-based image retrieval. https://en.wikipedia.org/wiki/Content-based_image_retrieval. Acessado em 01-2016.

[4] R. F. Gonzalez and R. E. Woods. Digital image processing. *Pearson PrenticeHall*, 2008.

[5] Image magic. http://www.imagemagick.org/script/index.php. Acessado em 05-2015.

[6] G. Lu. Multimedia database management systems. *Artech House*, 1999.

[7] H. Lu, B.-C. Ooi, and K.-L. Tan. Efficient image retrieval by color contents. *Applications of Databases*, pages 95–108, 1994.

[8] Max planck institut informatic. https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/object-recognition-and-scene-understanding/analyzing-appearance-and-contour-based-methods-for-object-categorization/. Acessado em 08-2015.

[9] H. Muller, W. Muller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Performance evaluation in content-based image retrieval: Overview and proposals. *Pattern Recognition Letters - Special issue on image/video indexing and retrieval*, 2001.

[10] Opencv (open source computer vision). http://opencv.org/. Acessado em 04-2015.

[11] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. *Proc. of ACM Multimedia Intl. Conf*, pages 65–73, 1996.

[12] J. Z. W. Ritendra Datta, Jia Li. Content-based image retrieval - approaches and trends of the new age. *MIR '05 Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, 2005.

[13] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. *CIKM '02 Proceedings of the eleventh international conference on Information and knowledge management*, 2002.

[14] Zero crossing laplacian operator. http://fourier.eng.hmc.edu/e161/lectures/gradient/node7.html. Acessado em 10-2015.