hackday
by Sistemas de Informação / Ciência da Computação

Computer Vision with Java by OpenCV

Prof. Marcos Roberto dos Santos (marcos.santos@imed.edu.br)

IMED – COMPLEXO DE ENSINO SUPERIOR MERIDIONAL

Passo Fundo - RS
Rua Senador Pinheiro, 304 - 99072-220

Porto Alegre - RS
Av. Bastian, 121 - Menino Deus - 90130-021

IMED

Descrição do Evento

Evento prático de desenvolvimento de projetos com objetivo de apresentar tecnologias ou metodologias inovadoras que estimulem a pesquisa e experimentação pelos estudantes.

Público Alvo

Estudantes de graduação em TI ou convidados externos com nível básico de conhecimento em programação, robótica e ou redes.

Responsabilidade e Contato

Evento sob a responsabilidade das escolas de Sistemas de Informação e Ciência da Computação da Faculdade IMED de Passo Fundo. Dúvidas, questionamentos e indicações no e-mail:

- si@imed.edu.br
- marcos.santos@imed.edu.br

GitHub (Codes + Tutorial)

Abaixo o conteúdo desenvolvido no projeto base de visão computacional para reconhecimento facial através de vídeo e ou foto.

Computer-Vision-with-Java-by-OpenCV:

https://github.com/profmarcossantos/Computer-Vision-with-Java-by-OpenCV

Open CV

http://opencv.org/

Computer Vision

Conforme Silva *et al.* [14], visão computacional ou *computer vision* (CV) é a ciência responsável pela forma com que o computador enxerga o meio a sua volta, extraindo informações úteis através de imagens capturadas por câmeras de vídeo, satélites, sensores, scanners, entre outros. A utilização destes métodos tem aumentado consideravelmente nos últimos anos em diversas áreas de aplicação, como na indústria, na lavoura, no trânsito, entre outros, pois fornecem quantidades significativas de informação [13]. Além disso, esta tecnologia possibilita analisar informações do espectro eletromagnético em que o olho humano não é sensível, tal como ultravioleta (UV) ou regiões espectrais de infravermelho.

A seguir serão abordados métodos e técnicas que correlacionam com as teorias e aplicações de CV e a extração de informações úteis para o processamento dos dados. Ainda será explanado sobre a biblioteca de código aberto OpenCV e sua possível aplicação em sistemas de visão computacional.

1.1. CAPTURA DE IMAGENS

A aquisição de imagens é o primeiro passo para se aplicar técnicas de visão computacional e justapor o devido tratamento para a obtenção de informações. Segundo Knob [5], a captura de imagens depende de dois componentes.

O primeiro é um dispositivo físico que seja sensível a uma banda do espectro de energia eletromagnética e que produza um sinal elétrico de saída proporcional a um nível de energia percebida. Como exemplos, têm-se equipamentos de ultrassom, radiografia, microscópios eletrônicos, radares, equipamento de ultrassom, câmeras digitais. O segundo é um dispositivo digitalizador para a conversão da saída elétrica para a forma digital.

Ainda conforme a proposição de Maruyama [9], a aquisição é o processo de obtenção da imagem por meio de um sensor de imageamento capaz de digitalizar o sinal produzido. Para tanto, algumas técnicas de pré-processamento são necessárias para melhorar a qualidade da imagem, com o intuito de facilitar a aplicação das etapas subsequentes, que serão dirimidas nos itens a seguir.

1.1.1. Iluminação

Dentre as principais características que determinam a qualidade que uma imagem digital pode apresentar, a iluminação destaca-se por estar relacionado à clara definição dos itens de uma cena. Segundo Silva [15], o termo imagem refere-se a uma função de intensidade luminosa bidimensional, denotada por $f(x,y)$, em que o valor da amplitude de f nas coordenadas espaciais (x,y) dá a intensidade (brilho) da imagem naquele ponto. Como a luz é uma forma de energia, $f(x,y)$ deve ser uma quantia positiva e finita, ou seja, $0< f(x,y) < \infty$.

A importância da iluminação é retratada por Koga *et al.* [7]. Eles definem que uma boa prática para capturar imagens é colocar os itens fotografados em um ambiente controlado, a fim de evitar variações de iluminação entre as imagens coletadas. A Figura 1 retrata um exemplo de ambiente para a coleta das imagens.

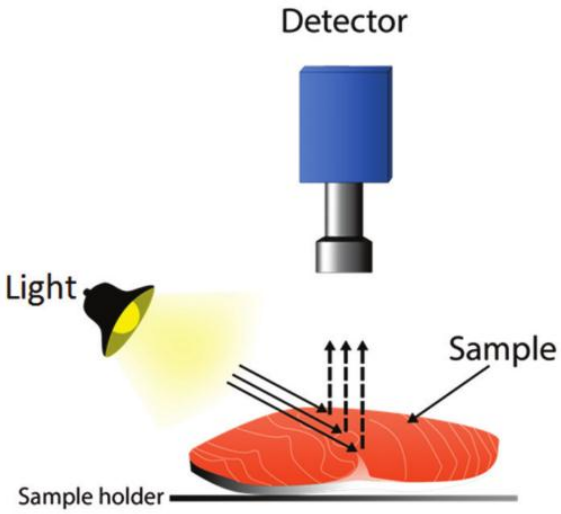


Figura 1. Esquema para coleta de imagens com iluminação controlada [13].

Ainda citando Silva [15], as imagens que as pessoas percebem em atividades visuais corriqueiras consistem de luz refletida dos objetos. A natureza básica de *f (x,y)* pode ser caracterizada por dois componentes: (1) a quantidade de luz incidindo na cena observada e (2) a quantidade de luz refletida pelos objetos da cena. Esses componentes são chamados de iluminação e refletância, respectivamente, e são representados por *i (x,y)* e *r (x,y)*, onde o produto das funções *i (x,y)* e *r (x,y)* resulta *f (x,y)*:

$$f(x,y) = i(x,y) \cdot r(x,y)$$

$$0 < i(x,y) < \infty$$

$$0 < r(x,y) < 1$$

Esta última equação indica que a refletância é limitada entre 0 (absorção total) e 1 (refletância total). A natureza de *i (x,y)* é determinada pela fonte de luz, e *r (x,y)* é determinada pelas características dos objetos observados.

1.1.2. Imagem de Reamostragem

Segundo Ferreira *et al.* [3], uma imagem digital pode passar por transformações em relação a sua resolução espacial antes de ser processada e analisada. A essas mutações dá-se o nome de imagem de reamostragem, que tem por finalidade o aumento ou a redução da resolução. No caso de redução tem-se uma subamostragem e no caso de aumento de resolução tem-se uma interpolação. Esta técnica é utilizada para ampliar, reduzir e rotacionar imagens digitais, eliminando informações irrelevantes e indesejáveis presente na imagem.

Na prática, este método permite que algoritmos de processamento de imagens, baseado na exploração de pixels, tenham um tempo de resposta relativo ao tamanho da imagem ou área foco que necessita ser analisada.

Conforme a hipótese de pesquisa de Ferreira *et al.* [3], a construção de uma imagem de reamostragem é realizada pelas aplicações de técnicas como detecção de linhas, armazenamento das linhas detectadas, eliminação de pontos inválidos, redimensionamento, equalização de contraste, entre outros.

Freitas *et al.* [4] também comentam que algumas imagens podem possuir variações de iluminação, ruídos, sombras e outras diversificações que afetam diretamente a análise. Na Figura 2 é demonstrado um exemplo de imagem de reamostragem diminuindo consideravelmente a área a ser processada e consequentemente o tempo.

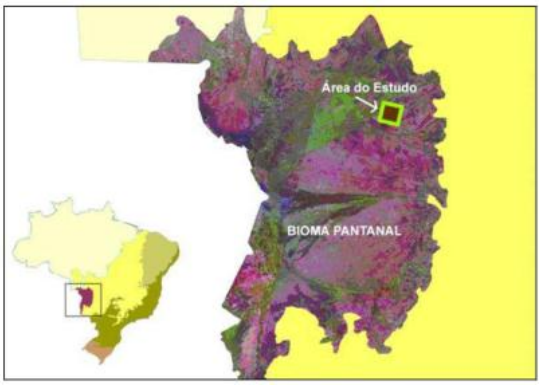
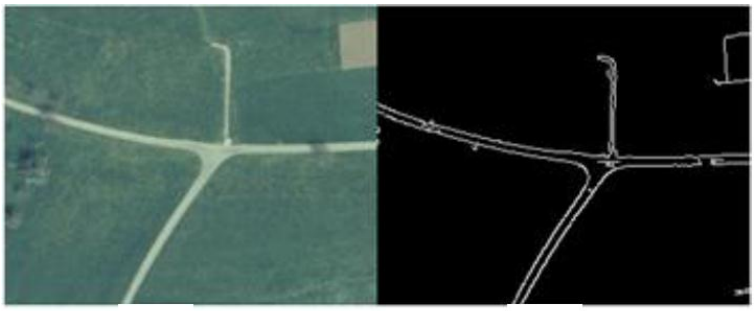


Figura 2. Imagem exemplo após aplicações de técnicas de reamostragem [4].

1.1.3. Construção de Imagem Base

De acordo com Ferreira *et al.* [3], imagem base é criada a partir da imagem de reamostragem, aplicando-se métodos de detecção de bordas. A finalidade da detecção de borda, em geral, é reduzir significativamente a quantidade de dados em uma imagem, enquanto preserva as propriedades estruturais a serem utilizadas para o processamento da gravura.

A Figura 3 demonstra um exemplo de imagem base processada por meio de um algoritmo de detecção de bordas.



(a) (b)

Figura 3. (a) Imagem de Entrada e (b) Imagem Base. [3]

Para a execução de algoritmos de detecção de bordas, Neves e Pelaes [10] destacam que dois filtros espaciais lineares podem ser aplicados: baseado no gradiente da função luminosidade da imagem e baseados no operador laplaciano.

O filtro gradiente de uma imagem *f(x, y)* na localização *(x, y)* é o vetor:

$\nabla = [G_x , G_y] = [\partial f, \partial x \partial f, \partial y]$

Sabe-se da análise vetorial que o vetor gradiente aponta na direção da máxima taxa de variação de *f* em *(x, y)*. Na detecção de borda uma quantidade importante é o módulo desse vetor, geralmente referido simplesmente como gradiente e denotado ∇f, onde:

$\nabla f = mag (\nabla f = [G^2_x + G^2_y]^{½}$

Essa quantidade é igual à máxima taxa de crescimento de *f(x, y)* por unidade de distância na direção de $f\nabla$. Uma prática comum é aproximar o filtro gradiente do valor absoluto: $\nabla f \approx |G_x| + |G_y|$ o qual é muito mais simples de se implementar.

Já o operador laplaciano se baseia em uma função 2-D *f(x, y)*, derivada de segunda ordem definida como: $\nabla^2 f = [\partial^2 f / \partial x^2] + [\partial^2 f / \partial y^2]$. Embora o laplaciano seja insensível à rotação, e, portanto capaz de realçar ou detectar bordas em qualquer direção, seu uso é restrito devido a sua grande suscetibilidade a ruído.

1.1.4. Armazenamento de Linhas Detectadas

Conforme Ferreira *et al.* [3], as informações geradas pelos métodos de detecção de bordas possuem coordenadas dos pixels necessários para os processamentos subsequentes. O armazenamento depende de uma estrutura de dados onde cada cadeia de pixels contíguos é armazenada em estruturas distintas do tipo *array* (vetor), representando segmentos da imagem.

Na Figura 4, são apresentados diversos segmentos de objetos na imagem, resultado de um pré-processamento.

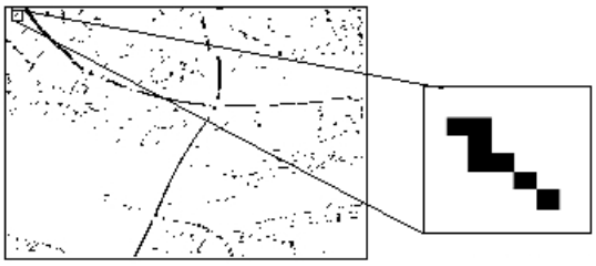


Figura 4. Segmentação de objetos de uma imagem [3].

Uma análise detalhada do objeto em questão pode ser exemplificada na Figura 5, incluindo o pixel correspondente de cada marcação:

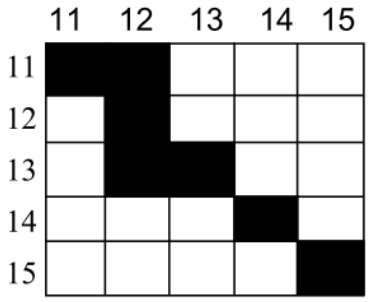


Figura 5. Detalhamento de objeto pixel a pixel [3].

Propondo a estrutura de dados para armazenar as informações do objeto selecionado, tem-se o seguinte *array* de inteiros*: [11,11], [11,12], [12,12], [13,12], [13,13], [14,14], [15,15]*.

1.2. IMAGEM DIGITAL

Uma imagem digital pode ser considerada uma matriz cujos índices de linhas (N) e de colunas (M) identificam um ponto na imagem, e o correspondente valor do elemento da matriz identifica a cor naquele ponto [5]. Os elementos básicos dessa matriz digital são chamados de elementos da imagem, elementos da figura ou "pixels" (abreviação de "*picture elements*"). Cada pixel representa uma medida que dependem de variáveis como cor, profundidade e tempo, é o menor ponto que forma uma imagem digital.

Pode-se entender uma imagem digital como um agrupamento de sensores que, quando atingidos por raios de luzes, gravam a tonalidade de cor que o atingiu. Logo, quanto mais pixels (sensores) uma imagem tem, melhor é sua qualidade e maior é a sua resolução.

As próximas subseções explicarão as principais etapas aplicadas ao processamento de imagens digitais, como técnicas que realçam as características da figura e diminuem imperfeições, bem como operações matemáticas utilizadas para extrair informações úteis do conjunto final de pixels.

1.2.1. Etapas de Processamento de Imagens

Conforme Araújo e Freire [1], o principal objetivo do processamento de imagens é a remoção de barreiras inerentes ao sistema visual humano, para facilitar a extração de informações. Segundo Knob [5], o reconhecimento de padrões encontrados em imagens pode ser dividido em cinco etapas: aquisição da imagem, pré-processamento, segmentação, representação/descrição e reconhecimento/interpretação, conforme o esquema apresentado na Figura 6.

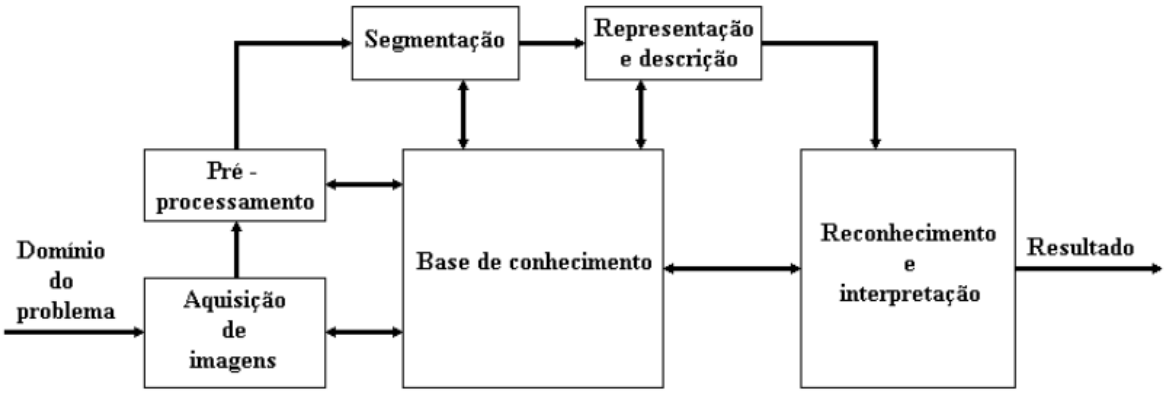


Figura 6. Etapas fundamentais do processamento de imagens [5].

1.2.2. Tamanho e Formato de Imagens Digitais

Segundo Peixoto *et al.* [12], o tamanho de uma imagem está altamente correlacionado com o tempo de processamento e o custo computacional em aplicações com sistemas de visão computacional. Conforme Koga *et al.* [7], para se obter estimativas do tamanho real de objetos em uma cena é necessário fazer a relação entre a média em milímetros deste objeto e o número de linhas e colunas de pixels que este

possui na imagem digital. Com essa técnica é possível analisarmos os *feedbacks* dos sistemas de visão computacional com unidades de medida entendíveis pelos seres humanos como centímetros e ou metros.

Ainda na detecção do tamanho de objetos na cena, um dos principais modelos utilizados é localizar o primeiro pixel que corresponde ao objeto, assim o algoritmo varre recursivamente sua vizinhança buscando por pixels de valores iguais. De acordo com Souza [16], esse processo acaba apenas quando não há mais pixels de valor igual adjacentes ao ruído. No item a seguir será explanado sobre essa técnica.

Conforme os formatos de arquivos de imagens, Knob [5] define que existe necessidade de se designar padrões de armazenamento de imagens de forma que se possa conseguir a interação destas entre diferentes sistemas. Outro fator importante é a codificação das imagens, uma vez que estas normalmente ocupam muito espaço de memória e por isso, necessitam do emprego de alguma forma de compressão de dados para o seu armazenamento. Existem diferentes formatos de arquivos para o armazenamento de imagens, e os principais são apresentados na Tabela 1.

Tabela 1. Formatos de arquivos para armazenamento de imagens [2].

Formato / Nome	Descrição	Reconhecimento Extensão
TIFF	Tagged Image File Format	.tif; .tiff
JPEG	Joint Photographic Experts Group	.jpg; .jpeg
GIF	Graphics Interchange Format	.gif
BMP	Windows Bitmap PNG	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

1.2.3. Os Vizinhos do Pixel

Conforme Silva [15], um pixel *p* nas coordenadas *(x,y)* possui quatro vizinhos horizontais e verticais, cujas coordenadas são dadas por *(x+1,y) (x-1,y) (x,y+1) (x,y-1)*. Esse conjunto de pixel chamado vizinhança-de-4 de *p*, sendo que alguns dos vizinhos de *p* ficarão fora da imagem digital se *(x,y)* estiverem na borda da imagem.

Um problema corriqueiro é quando um pixel é anisotrópico, ou seja, ele não apresenta as mesmas propriedades em todas as direções. Esta característica faz com que um pixel na sua coordenada possua quatro vizinhos de borda (dois verticais e dois horizontais), cujas coordenadas estão representadas na Figura 7.

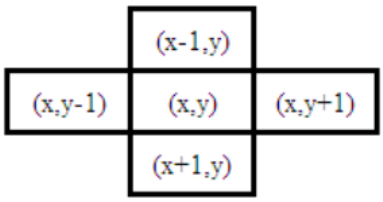


Figura 7. Vizinhança-de-4. [5]

Para maiores precisões analíticas, um pixel também pode possuir oito vizinhos, e nesse caso teremos quatro vizinhos de borda e quatro vizinhos de diagonal, como podemos observar na Figura 8, e nesse caso será chamado de vizinhança-de-8.

(x-1,y-1)	(x-1,y)	(x-1,y+1)
(x,y-1)	(x,y)	(x,y+1)
(x+1,y-1)	(x+1,y)	(x+1,y+1)

Figura 8. Vizinhança-de-8. [5]

1.2.4. Operações Lógicas e Aritméticas

Conforme Silva [15], uma imagem após ter sido adquirida e digitalizada pode ser vista como uma matriz de inteiros em um vetor e, portanto, pode ser manipulada numericamente utilizando operações lógicas e/ou aritméticas. Estas operações podem ser efetuadas pixel a pixel ou orientadas a vizinhança. No primeiro caso, elas podem ser descritas pela seguinte notação: $x\ opn\ y = Z$.

Sejam duas imagens X e Y de igual tamanho. Estas imagens podem ser processadas pixel a pixel utilizando um operador aritmético ou lógico, produzindo uma terceira imagem Z, cujos pixels correspondem ao resultado de *x opn y* para cada elemento de X e Y, conforme ilustra a Figura 9.

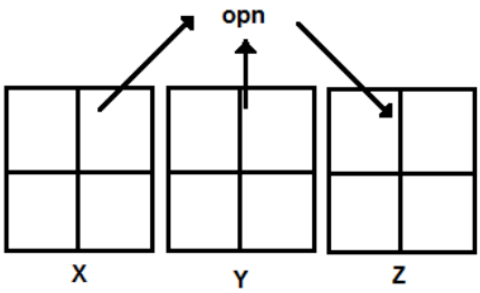


Figura 9. Operações lógicas e aritméticas pixel a pixel [15].

Operações aritméticas em imagens inteiras são desempenhadas pixel a pixel. Segundo Silva [15], o principal uso da adição de imagens ocorre ao se fazer média de imagens para redução de ruído. A subtração de imagens é um exemplo de recurso utilizado em imagens para remover informação estática de fundo. A multiplicação (ou divisão) de imagens é empregada para corrigir sombras de níveis de cinza produzidas por não uniformidades da iluminação ou sensor utilizado para a aquisição da imagem.

Ao executar operações aritméticas sobre imagens, deve-se tomar cuidado com os problemas de *underflow* ou *overflow* do resultado. A adição de duas imagens de 256 tons de cinza, por exemplo, pode resultar em um número maior de tons, ao mesmo tempo em que a subtração de duas imagens pode resultar em valores negativos para alguns elementos. Para contornar estes problemas, existem basicamente duas alternativas: (1) manter os resultados intermediários em uma matriz na qual o espaço em memória alocado para cada pixel permita a representação de valores negativos e/ou maiores que o intervalo de tons disponível, e normalizar; (2) truncar os valores maiores que o máximo valor permitido, bem como os valores negativos, igualando-os aos limites inferior e superior, respectivamente.

Já quando retratamos operações lógicas pixel a pixel utilizadas em processamento de imagens denotamos com E, OU e COMPLEMENTO.

Operações lógicas podem ser efetuadas em imagens com qualquer número de níveis de cinza, mas são mais bem compreendidas quando vistas em imagens binárias. São usadas para tarefas tais como mascaramento, detecção de características e análise de forma. Como as operações lógicas envolvem apenas uma posição de pixel de cada vez, podem ser feitas nas coordenadas do próprio pixel, como no caso das operações aritméticas.

Ainda falando em operações lógicas e aritméticas, estas são usadas em operações orientadas à vizinhança. O processamento da vizinhança é tipicamente formulado num contexto das assim denominadas operações por máscara. A ideia por trás das operações por máscara é modificar o valor de um pixel em função do seu próprio nível de cinza e o de seus vizinhos. Como, por exemplo, subtração de imagens, filtragem espacial e filtragem por mediana.

1.2.5. Imagens em Tons de Cinza

As imagens são processadas e transformadas para tons de cinza principalmente para amenizar os ruídos e eliminar informações irrelevantes [14]. Desse moto é possível facilitar a execução do algoritmo de detecção de bordas para gerar a imagem base que será analisada pelas operações lógicas e matemáticas.

Conforme Silva [15], para obter a imagem em tons de cinza utiliza-se a intensidade luminosa de uma figura monocromática *f* nas coordenadas *(x,y)*, de nível de cinza *(l)* da imagem naquele ponto, onde $L_{min} \leq l \leq L_{máx.}$

Em teoria, a única limitação sobre L_{min} é que seja um valor positivo e sobre $L_{máx}$ é que seja finito. Na prática $L_{min} = i_{min}\ r_{min}$ e $L_{máx} = i_{máx}\ r_{máx.}$

O intervalo $[L_{min},L_{máx}]$ é denominado escala de cinza. A prática comum é deslocar esse intervalo para $[0, L]$, onde $l = 0$ é considerado preto e $l = L$ é considerado branco. Todos os valores intermediários são tons de cinza variando continuamente entre o branco e o preto.

A Figura 10 representa a conversão de uma imagem para tons de cinza, onde muitos detalhes de iluminação são abstraídos da imagem, facilitando a criação da imagem base e consequentemente a redução do tempo de processamento da mesma.

Figura 10. Conversão de uma imagem original para tons de cinza e imagem base [5].

1.2.6. Binarização de uma Imagem Digital

Uma imagem binária é uma transformação que permite associar cada par de inteiros *(x,y)* a um único elemento natural pertencente ao intervalo [0,1] . Em outras palavras, estas imagens digitais usam apenas as cores preta e branca e sua representação matricial é composta apenas por elementos 0 e 1, onde o número 0 indica a cor preta e o número 1 indica a cor branca.

De acordo com Koga *et al.* [7], a binarização das imagens consiste na determinação de quais regiões da imagem pertencem ao fundo, e quais são os objetos de interesse.

1.2.7. Ajuste de Contraste

Este recurso do processamento de imagem digital é usado para realçar características visíveis em diferentes contrastes. Conforme Silva [15], pode-se utilizar equalização de histograma para este fim. Na Figura 11é possível perceber após o ajuste de contraste o aumento de detalhes na imagem final. Com essa técnica é possível melhorar as possibilidades de detecção de formas e consequentemente a obtenção de informações.

Figura 11. Operações lógicas e aritméticas pixel a pixel [15].

Equalizar o histograma significa obter a máxima variância do histograma de uma imagem, obtendo assim uma imagem com melhor contraste. O contraste é uma medida qualitativa e que está relacionada com a distribuição dos tons de cinza em uma imagem.

1.3. OPEN CV

Segundo Lima [6], a biblioteca OpenCV (*Open Source Computer Vision* – Visão Computacional de Código Aberto) foi desenvolvida pela Intel, escrita em Linguagem C/C++ e com suporte a diversas linguagens como Java e Python. A OpenCV procura emular a visão humana realizando o processamento de imagens mediante uma interpretação parcial de objetos de interesse na imagem capturada [11]. Além de estar livremente disponível, a OpenCV possui ampla documentação disponível em livros e na internet.

Dentre as funções do OpenCV, destacam-se as principais [8]:

- cvNamedWindow – cria uma janela onde será capturado o vídeo;
- cvCaptureFromCAM – função que retorna o dispositivo de captura esteja conectado ao computador, retornando falso caso contrário;
- cvQueryFrame – função que retorna frames capturados. O retorno é uma imagem colorida, que possui um registro que armazena uma matriz com as dimensões [0..Altura, 0..Largura, [R,G,B]];
- cvFlip – faz o espelhamento de uma imagem;
- cvCvtColor – função que converte uma imagem de um modelo de cor para outro; cvPtr2D – função que retorna as cores de um determinado pixel (x, y) da imagem;
- cvErode – função que aplica a operação de erosão da morfologia matemática;
- cvDilate – função que aplica a operação de dilatação da morfologia matemática;
- cvMorphologyEx – função que aplica a operação de abertura ou fechamento da morfologia matemática;
- cvFindContours – função que identifica contornos em imagens;
- cvConvexHull – função que encontra o fecho convexo;
- cvConvexityDefects - função que identifica os defeitos convexos dado um fecho convexo por parâmetro, bem como o ponto de maior distância.

Para se alcançar um melhor paralelismo com as linguagens, a biblioteca OpenCV entrega junto com sua distribuição plug-ins, como por exemplo, o JavaCV [6]. Esse *plugin* oferece várias funções existentes no OpenCV, as quais são acessadas via JNI (*Java Native Interface* – interface nativa Java).

Referências

[1] Araújo, M.V. and Freire, G.S.S. 2007. Utilização de SIG nos Estudos Ambientais do Estuário do Rio Acaraú/Ceará. *Revista Geonomos*.

[2] Casanova, M.L.S. et al. 2010. Avaliação da qualidade das imagens digitais panorâmicas adquiridas com diferentes resoluções. *Brazilian Dental Science*.

[3] Ferreira, E.F. dos S. et al. 2013. Extração de Sementes de Rodovias a partir de Imagens Aéreas Digitais. *Colloquium Exactarum*.

[4] Freitas, D.M. de et al. 2009. Fusão de Imagens Cbers-CCD com Cbers-HRC para obter uma melhor interpretação das sub-regiões e áreas antrópicas do Pantanal. *2009*. 2º Simpósio de Geotecnologias no Pantanal.

[5] Knob, A.H. 2010. *Aplicação do processamento de imagens digitais para análise da anisotropia da massa de grãos*. Universidade Regional do Noroeste do Estado do Rio Grande do Sul.

[6] De Lima, J.R. et al. 2012. Comparação de Histogramas de Imagens Digitais para Determinação de Similaridade em Sementes de Milho. *Revista de Engenharia e Tecnologia*.

[7] LOLA Silva, ML Koga, LBV Boas, CE Cugnasca, A.C. 2013. Automated visual quality sorting of agricultural seedlings. *IX Workshop de Visao Computacional (WVC 2013)*. (2013).

[8] Masutani, V.H. et al. Reconhecimento de Objetos Coloridos e Mãos Usando Cores e Formas. *Colloquium Exactarum*.

[9] Matos Maruyama, T. et al. 2015. Estudo Comparativo de Opções de Software de Processamento Digital de Imagens para Análise de Sementes. *Anais SULCOMP*.

[10] Neves, S.C.M. and Pelaes, P.D.E.G. 2001. Estudo e Implementação de Técnicas de Segmentação de Imagens. *Revista Virtual de Iniciação Acadêmica da UFPA*. Vol 1, (2001), Página 1 de 11.

[11] Paiva, J.P. and Yepes, I. 2014. Uso de Visão Computacional para Controle Autônomo de Câmera Embarcada em Veículo Aéreo não Tripulado. *Anais do Computer on the Beach*.

[12] Peixoto, F. de M. et al. 2015. Desenvolvimento de um Software para cálculo da densidade de nódulos de grafita em ferro fundido nodular através de Processamento Digital de Imagens. *Matéria (Rio de Janeiro)*. 20, 1 (Mar. 2015), 262–272.

[13] Saldaña, E. et al. 2013. Review: computer vision applied to the inspection and quality control of fruits and vegetables. *Brazilian Journal of Food Technology*. 16, 4 (Dec. 2013), 254–272.

[14] Silva, G.G. da et al. 2014. Veículos Aéreos não Tripulados com Visão Computacional na Agricultura: Aplicações, Desafios e Perspectivas. *Anais do Encontro Científico de Administração, Economia e Contabilidade*.

[15] SILVA, J.D.O. DA 2011. *Processamento de Imagens Digitais e o Ensino de Matrizes*. Universidade Federal do Oeste do Pará.

[16] SOUSA, D.F. 2014. *Uma abordagem híbrida e semiautomática para estimativa de regiões cobertas por nuvens e sombras em imagens de satélite: análise e avaliação*. UNIVERSIDADE FEDERAL DO PARÁ.