CENTRO UNIVERSITÁRIO DA FEI
FABIO PANTANO DE LUCA

# SEGMENTAÇÃO DE IMAGENS COM BAIXA PROFUNDIDADE DE CAMPO PARA APLICAÇÕES DE TEMPO REAL

São Bernardo do Campo
2016

FABIO PANTANO DE LUCA

# SEGMENTAÇÃO DE IMAGENS COM BAIXA PROFUNDIDADE DE CAMPO PARA APLICAÇÕES DE TEMPO REAL

Dissertação apresentada ao Centro Universitário da FEI, como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica. Orientada pelo professor Carlos Eduardo Thomaz.

São Bernardo do Campo
2016

Pantano de Luca, Fabio.
Segmentação de imagens com baixa profundidade de campo para aplicações de tempo real / Fabio Pantano de Luca. São Bernardo do Campo, 2016.
167 p. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Carlos Eduardo Thomaz.

1. Segmentação de imagens. 2. Baixa profundidade de campo. 3. Processamento em tempo real. I. Thomaz, Carlos Eduardo, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

# APRESENTAÇÃO DE DISSERTAÇÃO
# ATA DA BANCA EXAMINADORA

**Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica**

**Mestrado**

**PGE-10**

**Aluno:** Fabio Pantano de Luca **Matrícula:** 114107-6

**Título do Trabalho:** Segmentação de imagens com baixa profundidade de campo para aplicações de tempo real.

**Área de Concentração:** Processamento de Sinais

**Orientador:** Prof. Dr. Carlos Eduardo Thomaz

**Data da realização da defesa:** 29/03/2016

ORIGINAL ASSINADA

**Avaliação da Banca Examinadora:**

São Bernardo do Campo, 29 / 03 / 2016 .

| MEMBROS DA BANCA EXAMINADORA | |
|---|---|
| Prof. Dr. Carlos Eduardo Thomaz | Ass.: _____ |
| Prof. Dr. Hae Yong Kim | Ass.: _____ |
| Prof. Dr. Paulo Sérgio Silva Rodrigues | Ass.: _____ |

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

**VERSÃO FINAL DA DISSERTAÇÃO**

**APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA**

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Este trabalho é dedicado aos meus pais por terem me transmitido o gene do descontentamento, buscando sempre ir além.

## AGRADECIMENTOS

À minha esposa, pelo afeto, pelo companheirismo e pela paciência. Por ter compartilhado as dificuldades desses dois últimos anos.

Aos meus pais e família, pelo exemplo.

Ao meu orientador, Carlos Thomaz, e professores, desta e de demais instituições, pela transmissão do conhecimento.

À CAPES pelo apoio financeiro recebido por meio da bolsa modalidade ”taxas escolares” nestes dois anos de Mestrado.

*O mostrengo que está no fim do mar*
*Na noite de breu ergueu-se a voar;*
*A roda da nau voou três vezes,*
*Voou três vezes a chiar,*
*E disse: "Quem é que ousou entrar*
*Nas minhas cavernas que não desvendo,*
*Meus tectos negros do fim do mundo?"*
*E o homem do leme disse, tremendo:*
*"El-Rei D. João Segundo!"*

*"De quem são as velas onde me roço?*
*De quem as quilhas que vejo e ouço?"*
*Disse o mostrengo, e rodou três vezes,*
*Três vezes rodou imundo e grosso.*
*"Quem vem poder o que eu só posso,*
*Que moro onde nunca ninguém me visse*
*E escorro os medos do mar sem fundo?"*
*E o homem do leme tremeu e disse:*
*"El-Rei D. João Segundo!"*

*Três vezes do leme as mãos ergueu,*
*Três vezes ao leme as reprendeu,*
*E disse no fim de tremer três vezes:*
*"Aqui ao leme sou mais do que eu:*
*Sou um povo que quer o mar que é teu;*
*E mais que o mostrengo, que me a alma teme*
*E roda nas trevas do fim do mundo,*
*Manda a vontade, que me ata ao leme,*
*De El-Rei D. João Segundo!"*

(O Monstrengo, Fernando Pessoa (PESSOA, 2013))

## RESUMO

Com o objetivo de tornar dispositivos parcial ou completamente autônomos, hoje é imperativo extrair informações relevantes de uma vasta quantidade de dados disponíveis e com recursos computacionais adequados para operação em tempo real. Nos últimos anos, se tornou comum o processamento de imagens como uma solução possível para esse problema. Além disso, o uso de análise de imagens imita em partes a referência de sucesso em muitos casos práticos: a visão humana. Reconhecimento de imagens pode ser usado com altas taxas de acurácia quando o objeto de interesse ou o ambiente são bem conhecidos. Porém, em ambientes pouco controláveis, como espaços abertos urbanos, onde existe uma grande quantidade e variedade de artefatos e estímulos (informação), a separação ou segmentação do objeto de interesse (*foreground*) do restante da imagem (*background*) é ainda uma ação desafiadora cientificamente. Uma forma possível de abordar esse problema é por meio de alterações na forma de captura das imagens, possibilitando que mais informação relevante esteja disponível para os algoritmos de segmentação. Uma abordagem atual é o uso de imagens com baixa profundidade de campo, que, analogamente à percepção humana, pode discriminar o objeto de interesse do restante da cena através de uma parte principal de atenção visual (foco). Essa dissertação propõe o uso de duas imagens para análise, sendo uma delas com baixa profundidade de campo (pequena região em foco) e a outra com alta profundidade de campo (região maior em foco). Neste contexto, descreve-se e implementa-se uma nova técnica para segmentação de imagens, com resultados promissores de acurácia e tempo de processamento, sobressaindo-se às demais técnicas de segmentação baseadas em foco já existentes e discutidas ao longo do trabalho.



**Palavras-chave**: segmentação de imagens. Baixa profundidade de campo. Processamento em tempo real.

## ABSTRACT

In order to make devices partially or completely autonomous, it is imperative nowadays to extract relevant information from a large set of data available and using adequate computational resources for real-time operation. In the last years, it has become common to use images processing as a feasible solution to this problem. Additionally, the use of image analysis imitates the successful reference: the human vision. Image recognition can be used with high accuracy rates when the object of interest or the surrounding environment are well known. However, for the opposite situation, like open urban spaces, where a large number visual artifacts and stimuli (information) from many sorts are present, the segmentation of the object of interest (foreground) from the rest of the scene (background) is a challenging issue. One possible way to tackle this problem is changing how images are captured, increasing the quantity of relevant information in an image for the segmentation algorithms. A feasible approach is using low-depth of field images, which, analogous to the human vision, can discriminate the object of interest from the rest of the scene through a limitation of image elements having visual attention (focus). This dissertation proposes the use of two image for the analysis: the first of with low-depth of field (fewer elements into focus) and the other one with a high depth of field (more elements into focus). In this context, a new image segmentation method for focused elements is described and implemented, having promising results for both accuracy and processing time, over standing the other existing focus-based segmentation methods that were discussed along this work.



**Keywords**: image segmentation. Low depth of field. Real time processing.

**SUMÁRIO**

## LISTA DE FIGURAS

# 1 INTRODUÇÃO

Diversos obstáculos estão presentes na busca por tornar dispositivos, objetos e demais equipamentos existentes hoje em dia parcial ou completamente autônomos em sua operação. Uma dessas questões é a necessidade que informações relevantes sejam entregues aos mesmos, a fim de que um sistema decisório realize as ações pertinentes a que se propõe. Em razão dos tempos de processamento, a dificuldade da obtenção dos dados relevantes tem uma particular importância se há operação em tempo real. Sendo assim, esse problema possui dois obstáculos centrais:

a) Primeiramente, a quantidade e qualidade da informação entregue para a tomada de decisão do sistema. Por quantidade deve ser entendido justamente como o mínimo de informação necessária para que o sistema opere da maneira esperada. Mais dados redundantes implicam em desperdício, por si só algo negativo para a captura, além de exigir processamento dessa informação excedente, o que resulta em operações desnecessárias. Pela qualidade, são sinais livres de ruídos e com informações relevantes ao processo total. Em suma, pode-se definir como uma busca por eficiência máxima;

b) Segundo, a velocidade com que os dados são passados ao processo decisor. Mesmo dados que sejam de completa relevância para uma decisão, são inúteis se não forem entregues no tempo necessário.

Esses obstáculos são simultâneos. Consequentemente, a solução de apenas um deles não garante a validade da abordagem. De fato, uma solução que consiga apenas resolver uma das questões não poderia ser aplicada pela sua incompletude.

Apenas como um exemplo ilustrativo de diversas situações possíveis e críticas de sistemas com operação de tempo real, dotar um veículo automotor com controle de velocidade automático a partir da leitura da sinalização do tráfego, a não detecção das placas, o falso reconhecimento das mesmas, ou um atraso muito grande na sua interpretação, podem gerar prejuízos, seja pela quebra da expectativa dos usuários, sejam por prejuízos financeiros ao proprietário através de multas ou acidentes de trânsito causados.

O exemplo apresenta as duas dificuldades citadas considerando o uso de visão computacional. Primeiramente, a captação da informação das placas via análise de imagens pode ser uma tarefa árdua, em virtude da falta de padronização de sinalização e da dispersão dos posicionamentos possíveis. Nesse contexto, imagens abertas do ambiente possuem muita informação desnecessária, sendo imprescindível a segmentação como forma de aumento de eficiência dos dados. Já a presença de inúmeras sinalizações em casos urbanos, muitas vezes irrelevantes, como placas de referências ou direção de locais, que não são necessários para as decisões do sistema, exigem que haja uma grande velocidade da análise de cada quadro.

Demais situações de aplicação e que compartilham as mesmas dificuldades de extração de informação válida do todo, e que trabalham em altas taxas de amostragem, poderiam ser

a detecção de faces em meio a locais super populados, ou veículos dotados de frenagem semi-autônoma ou autônoma, que precisam localizar outros veículos automotores nas vias, ou objetos que apresentem uma situação de risco, ou ainda veículos de transporte completamente autônomos. Apesar de serem citados apenas como exemplos de aplicações, não são casos distantes da realidade atual, podendo se tratar de tecnologias já necessárias por empresas ou ao menos num futuro próximo. A frenagem autônoma (completa ou parcial) passa a ser obrigatória para que veículos comercializados na Europa obtenham a nota máxima (5 estrelas) de segurança EuroNCAP[1](sigla para *European New Car Assessment Programme*) a partir de 2018 (European New Car Assessment Programme (Euro NCAP), 2014). Ainda reforçando a importância de tais sistemas, em anos recentes é nítido o crescente interesse nos veículos de direção autônoma, representando um dos próximos passos no futuro da industria automotiva (THE ECONOMIST, 2015).

As grandezas físicas representadas nos dados avaliados podem ser vastas, e dependem do meio em que pretende-se inserir o sistema ou dos objetivos estabelecidos para o mesmo. Para cada caso os problemas anteriormente citados podem se apresentar em maior ou menor grau. Para o exemplo de detecção de sinalização de trânsito, uma forma de atacar o problema é analisar como seres humanos conseguem realizar a mesma tarefa de forma trivial e com grande sucesso, valendo-se basicamente de sua visão[2] (ANWER; VÁZQUEZ; LÓPEZ, 2011). Logicamente, essa mesma análise é válida para muitas outras situações e surge dai a motivação nessa abordagem: o grande desequilíbrio dos resultados que ocorre quando uma tarefa é executada por um sistema autônomo ou por um ser humano valendo-se de análise de imagens. Também a favor do uso de imagens como fonte de informação, em muitos casos o sensoriamento do ambiente de operação não é possível, pela incompatibilidade dos sensores com o meio (WHITAKER, 1996) ou por interferências (KIM; EOM; PARK, 2015). Acrescentam-se ainda os sucessos recentes do uso de imagens em geral (SALDAÑA et al., 2013).

A comparação entre humanos e máquinas pode ser feita direcionada aos "defeitos" da visão humana. O termo defeito é grafado de tal maneira para indicar que é desta forma que algumas características ou limitações são interpretadas (JAVARAN; HASSANPOUR; ABOLGHASEMI, 2015) em geral, mas que aqui não serão classificadas, mas apenas usadas. Sendo elas classificadas de forma negativa, podem passar despercebidas ou serem negligenciadas pelo público em geral.

A visão humana apresenta uma limitação na capacidade de foco em vista de possuir uma profundidade de campo finita (MARCOS; MORENO; NAVARRO, 1999), inerente a qualquer sistema óptico. Essa é a caraterística que pode-se aplicar aos sistemas autônomos ao fazer alterações de abertura ótica e consequente redução da profundidade de campo.

Reforçando esse conceito da limitação da profundidade de campo (imagem com foco e

[1]Programa europeu que avalia o desempenho de segurança de novos carros e atribui aos mesmos notas conforme seus resultados. A nota máxima são 5 estrelas.

[2]Apesar da cognição humana usar outros dados no geral, como audição, tato ou olfato, no caso da direção, as entradas de informações durante a direção são predominantes via a visão.

desfoque) para seres humanos, estudos recentes (KHAN; DINET; KONIK, 2011; PETERSON et al., 2015) demonstram por meio de experimentos de rastreamento ocular (ou, em inglês, *eye tracking*) com imagens controladas que o foco é um fator que deve ser considerado dentre os elementos de atratividade da visão humana, sendo que tradicionalmente dentre tais elementos estão cores, texturas, intensidade e contraste. Assim, nos modelos de visão humana, o foco ou não foco devem ser integrados. Essa proposta vem se evidenciando apenas mais recentemente em trabalhos científicos (KHAN; DINET; KONIK, 2011).

Conceitualmente, imagens são normalmente sinais em duas dimensões, representando a amplitude pontual de componentes espectrais sobrepostos num intervalo de tempo definido. Apesar de recentes avanços em imagens com três dimensões, a forma mais popular e simples de obter imagens são aquelas em 2D ainda. Mesmo com essa aparente limitação dimensional, existem formas de mapear a informação de modo que as imagens contenham mais dados presentes em seu conteúdo, além do que a luminância e cores de cada pixel, chegando a um número de dimensões intermediário entre 2 e 3. Algumas das possibilidades são o uso de câmeras que medem a distância pela emissão de luz, ou *time-of-flight camera* (ToF *camera*) (GOKTURK; YALCIN; BAMJI, 2004), emprego de codificação no obturador da câmera (LEVIN et al., 2007), e a redução da profundidade de campo (LUCA; THOMAZ, 2015; WANG et al., 2001).

No caso geral de uma imagem qualquer com foco seletivo (limitado), apesar do grau de desfoque ter relação com a distância entre o objeto e o plano de captura (sensor ou filme fotográfico), não é possível medir ou mesmo estimar a distância (profundidade) de cada objeto na imagem apenas pelo quão desfocado o mesmo é representado por não haver informação do conteúdo original. No entanto, existe a possibilidade de usar a informação de um objeto estar em foco ou não como recurso para saber se o objeto está dentro de uma área de interesse. Os benefícios de tal separação são basicamente dois: obter uma melhor segmentação (separação entre *foreground* e *background*), que se torna especialmente interessante em ambientes com alto grau de poluição visual, e estimar a distância entre o objeto focado e a câmera, quando parâmetros do sistema óptico são conhecidos. Adicionalmente, a mesma técnica pode ser empregada para fazer uma análise 3D, desde que seja possível fazer a captura de uma sequência de imagens com a variação da distância focal entre elas, num processo análogo ao empregado por técnicas de ressonância magnética (KUPERMAN, 2000).

Outros benefícios trazidos pelo uso de imagens com baixa profundidade de campo e o aperfeiçoamento da segmentação de área de interesse (salientes) são a otimização de buscas baseadas no conteúdo, aumento de eficiência em processos de corte automático das partes de interesse da imagem e melhorias nos processos de reconhecimento de objetos. Além disso, há melhorias no processo de compressão de imagens (NEVEROVA; KONIK, 2012). A base de algumas dessas possibilidades está no conceito largamente empregado (NEVEROVA; KONIK, 2012) de uma relação direta entre a quantidade de informação nas partes de uma imagem e as partes salientes dessa. Levando-se em conta que tais concentrações de informações significam a presença de mais componentes de alta frequência no domínio espectral, fica assim caracterizada a presença de alta frequência (detalhes) na região em foco, e da falta de tais componentes

espectrais nas partes sem foco (não-detalhes).

Imagens com baixa profundidade de campo são largamente encontradas em fotografias profissionais e imagens microscópicas (WANG et al., 2001). Quando do interesse e necessárias, essas imagens podem ser também criadas, tendo o aparato fotográfico e configurações adequados. Por sua vez, tais equipamentos se popularizaram ao longo dos últimos anos, o que torna mais acessível e barata tal tecnologia (Camera and Imaging Products Association (CIPA), 2014). Apesar dessa relativa facilidade na criação das imagens, a separação entre regiões com foco e sem foco não é simples do ponto de vista computacional. Para isso, há inúmeras linhas de pesquisa ao redor do mundo, cada qual com uma metodologia própria e também com resultados diversos.

## 1.1 OBJETIVO

O presente trabalho tem por objetivo reforçar a importância do uso de imagens como fonte de dados para sistemas de tempo real, mas com a mudança conceitual de uso de imagens com baixa profundidade de campo para segmentação e análise das regiões de interesse. Propõe-se que, diferentemente das técnicas atuais, utilizam-se duas imagens para a segmentação, ao invés de apenas uma. Como consequência direta dessa mudança de paradigma novas abordagens são permitidas e um novo algoritmo é apresentado e avaliado. Adicionalmente, uma técnica de criação de imagens artificiais para a geração de um banco de dados de testes é também descrita e implementada em virtude da inexistência de tais bancos publicamente.

## 1.2 ORGANIZAÇÃO

O trabalho está dividido em cinco capítulos. No capítulo 2, é feita uma revisão teórica dos conceitos da ótica necessários para embasar o uso da baixa profundidade de campo das imagens como fator de discerimento para os algoritmos. No capítulo 3, são apresentados métodos existentes na literatura para segmentação de imagens com baixa profundidade de campo a partir de uma única imagem, e também é apresentada a proposta de um novo método que faz uso de duas imagens. Resultados experimentais comparativos estão presentes nesse capítulo. No capítulo 4, é apresentada uma abordagem de criação de imagens artificiais para testes e resultados mais extensivos do novo método de segmentação proposto empregando tais imagens. Por fim, no capítulo 5, estão as conclusões e os possíveis trabalhos futuros desde estudo.

# 2 FUNDAMENTAÇÃO TEÓRICA

Diversos conceitos empregados ao longo desta dissertação baseiam-se em princípios de ótica que normalmente estão fora do domínio de pessoas ligadas às questões de visão computacional e áreas correlatas. Desta forma, um detalhamento dessas questões, que passarão a ser empregadas ao longo do trabalho, são necessárias. Os conceitos apresentados de ótica são limitados ao necessário para um melhor embasamento do trabalho em questão, sem extrapolações.

## 2.1 ÓTICA GEOMÉTRICA E ÓTICA PARAXIAL

Apesar do entendimento a cerca da real natureza da luz ser vastamente conhecido, ou seja, sabe-se que a luz é uma onda eletromagnética, seu estudo como raios (partícula) é uma aproximação adequada para diversas aplicações. Quando os elementos de estudo estão dentro do limite em que suas dimensões são muito maiores do que o comprimento de onda da luz[1], muitos efeitos são neglicenciáveis, tais como a difração. Para tais casos clássicos, o estudo da ótica na forma geométrica é adequada.

Dentro da ótica geométrica uma segunda aproximação é possível, sem que isso cause perdas da qualidade de soluções. Trata-se da ótica paraxial ou gaussiana. Nesta aproximação, consideram-se apenas raios (observe que já está embutido neste termo a aplicação da ótica geométrica) de luz que formem um pequeno ângulo com o eixo ótico do sistema (LIPSON; LIPSON; LIPSON, 2010). Essa segunda aproximação tem um carácter mais matemático do que físico, que foi o caso do entendimento como uma análise de partículas. Nesses termos, e definindo $\theta$ como sendo o ângulo entre o raio de luz e o eixo do sistema, a relação matemática

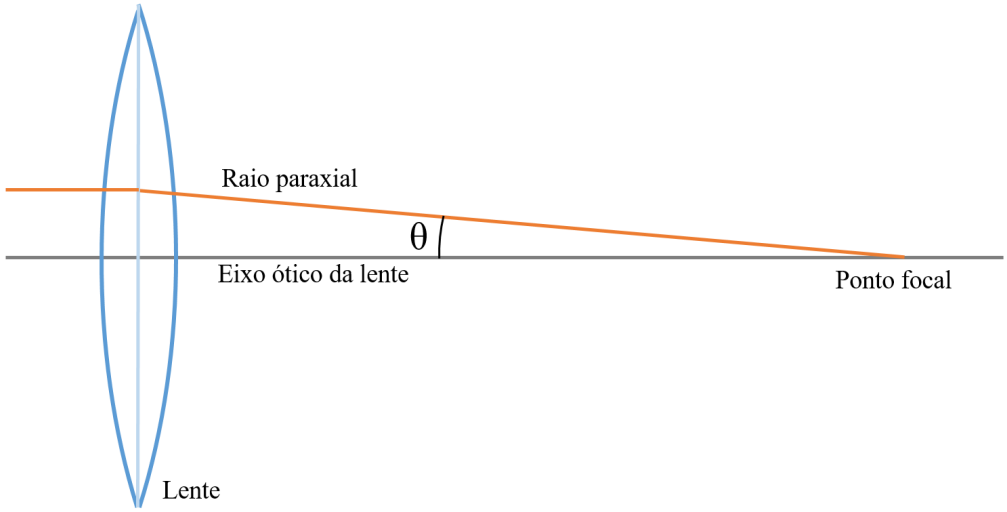
$$\theta \approx \sin\theta \approx \tan\theta \tag{1}$$

passa a ser válida e empregada para $\theta$ muito pequenos. A ilustração do ângulo $\theta$ é feita na Figura 1.

A importância da relação (1) não será explicitamente demonstrada, mas é relacionada com os cálculos dos ângulos de difração pela lei de Snell–Descartes (SERWAY; JEWETT, 2013).

[1] Para referência, os comprimentos de onda da luz visível, tanto superior quanto inferior, variam para cada indivíduo. No entanto alguns valores são determinados como usuais (médios), mas mesmo assim não há consenso. Uma faixa de trabalho aceita é entre 430 e 690 nanômetros, com centro em 555 nanômetros (HALLIDAY; RESNICK; WALKER, 2010).

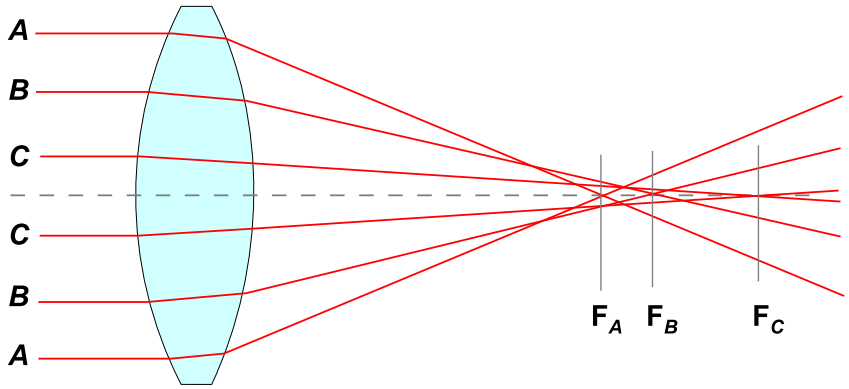
Figura 1 – Ilustração do conceito de ótica paraxial




Fonte: Autor, 2015
Notas: Representação do angulo $\theta$ num sistema ótico simples (lente convexa simples).

Como consequência também da aproximação gaussiana ou paraxial, uma outra limitação comum em lentes reais deixa de ser considerada. Trata-se da aberração esférica, na qual diferentes pontos da lente (distâncias medidas relativamente ao eixo ótico) e os respectivos raios que atravessam nesses locais terão diferentes pontos focais, resultando numa imagem não perfeita. A ilustração da aberração esférica é feita na Figura 2. Esse tipo de aberração é comum em sistemas óticos reais que empregam lentes esféricas, mas será negligenciado para os cálculos deste trabalho, sem que isso afete os resultados obtidos. Na prática existem compensações possíveis para mitigar ou anular o efeito indesejado. No caso de lentes compostas, isso pode ser feito compensando os efeitos em alguma das componentes do conjunto (BASS et al., 1994). De forma geral, uma opção é através da limitação da área efetiva da lente, para que apenas a região central tenha uso. Essa segunda alternativa é comum em câmeras e lentes que possuam um diafragma de controle de abertura, por exemplo (SERWAY; JEWETT, 2013). Assim, para as análises feitas, será considerado um foco ideal.

Figura 2 – Ilustração da aberração esférica



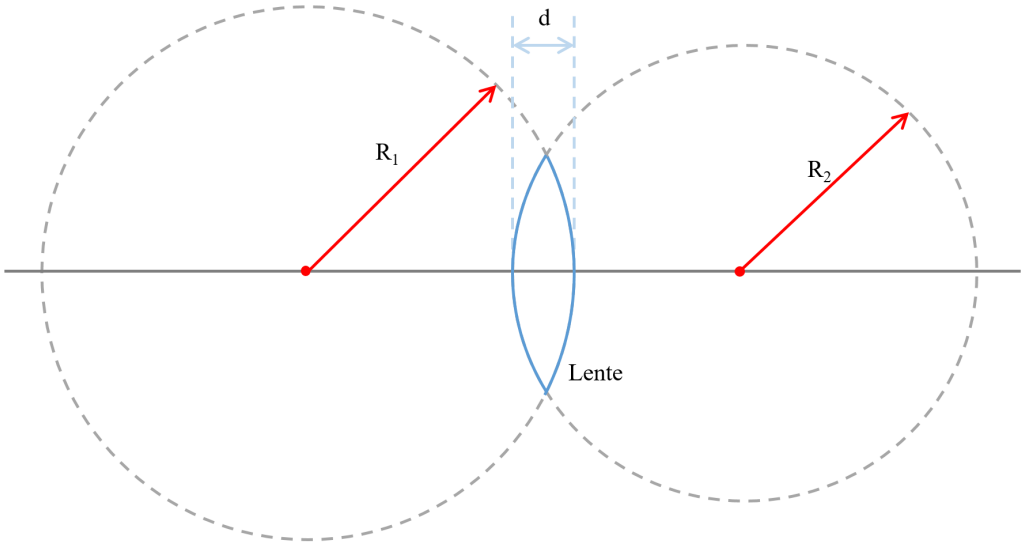

Fonte: (WIKIMEDIA COMMONS, 2011)
Notas: Ilustração do fenômeno da aberração esférica que ocorre em lentes reais. Os pares de raio $A$ , $B$ e $C$ possuem cada qual um ponto focal diferente, respectivamente $F_A$ , $F_B$ e $F_C$, criando uma distorção na imagem formada.

## 2.2 LENTES FINAS

A forma mais comum de entender lentes é através da união de duas superfícies esféricas limitando um material de índice de refração $n$ (constante[2]) diferente do meio externo. Normalmente, vidro e ar, respectivamente. Sendo os raios que definem ambas as superfícies da lente denominados de $R_1$ e $R_2$, conforme ilustrado na Figura 3. A distância entre os pontos das superfícies da lente, que pertencem simultaneamente ao eixo ótico da mesma, expressa a espessura da lente $d$. Quando os raios $R_1$ e $R_2$ são muito maiores do que a espessura $d$, passa-se a denominar o sistema por lente fina. O uso dessa forma de lente possui relação com uma aproximação anteriormente feita, que se trata da ótica paraxial. Quando as condições de lentes finas são aplicadas, os raios que incidem sobre o material das mesmas sofrem difração em ângulos próximos a 90° praticamente ao longo de toda a superfície considerada (lembrando aqui que, em muitos sistema óticos, a área útil da lente é limitada a uma região próxima ao centro da mesma, seja para redução da aberração esférica, seja por necessidade de obtenção de outros efeitos, como será explorado adiante), ou seja, os raios de luz na análise geométrica trabalham em uma direção quase paralela ao do eixo ótico, e consequentemente possuem um pequeno valor do ângulo $\theta$.

[2]Para obtenção de um índice de refração $n$ constante, é necessário que o material em questão seja homogêneo. Em livros textos (BASS et al., 2009), a consideração mais comum é a condição da homogeneidade do material.

Figura 3 – Ilustração de construção de lentes finas através da intersecção de esferas




Fonte: Autor, 2015

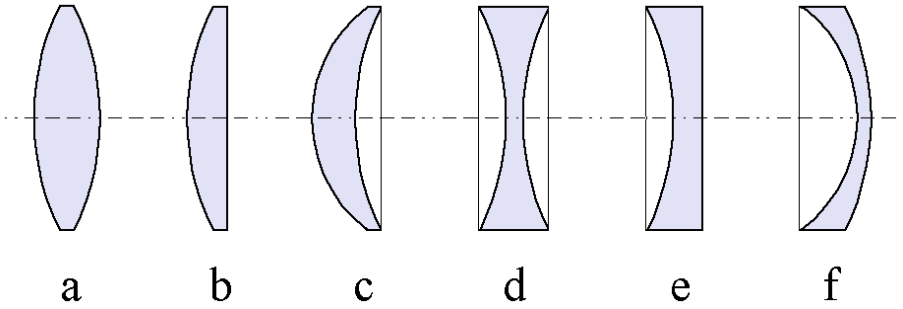
Notas: Ilustração de construção de lentes finas através da intersecção de esferas de raios $R_1$ e $R_2$ apresentando da forma de medição da largura da lente de modo a definir sua espessura $d$.

Apesar da Figura 3 ilustrar apenas um caso possível de lentes esféricas, outros casos existem, como será visto adiante. Essas construções diversas são feitas variando a posição dos centros das esferas ao longo do eixo ótico e o valor do raio (inclusive chegando ao caso limite de raio infinito, o que resulta num plano). Tais construções estão demonstradas na Figura 4. Para cada construção exibida, o resultado da passagem da luz pelas mesmas poderá ter resultados muito distintos. Uma forma de classificação é através da convergência ou divergência da lente. No primeiro caso, a luz composta de raios paralelos (feixe de luz colimado) ao atravessar a lente será direcionado para o ponto focal[3] do lado oposto. Lentes com tais características são referidas como convergentes. Em contra partida, no caso das lentes divergentes, um mesmo feixe colimado irá se espalhar ao passar pela mesma. Analisando a Figura 4, as lentes $a$ e $b$ se enquadram na classificação de convergentes e as lentes $d$ e $e$ como divergentes. As lentes $c$ e $f$ podem pertencer a qualquer um dos grupos, conforme os raios de construção $R_1$ e $R_2$ e a espessura $d$.

Um resultado da definição das lentes como convergente e divergente são os tipos de imagens produzidas. Como o estudo da ótica aqui está voltado para a fotografia tradicional, apenas as lentes convergentes são de interesse, pois são as únicas que formam a imagem do lado oposto ao objeto (no caso de uma câmera, sobre o sensor ou filme da mesma). As lentes divergentes formarão a imagem entre o próprio objeto e a lente. Também possuem aplicações, mas não para fins dessa dissertação.

[3] Fica evidente nesse ponto que os efeitos da aberração esférica foram descartados, conforme discussão anterior.

Figura 4 – Tipos de lentes




Fonte: (WIKIMEDIA COMMONS, 2005)

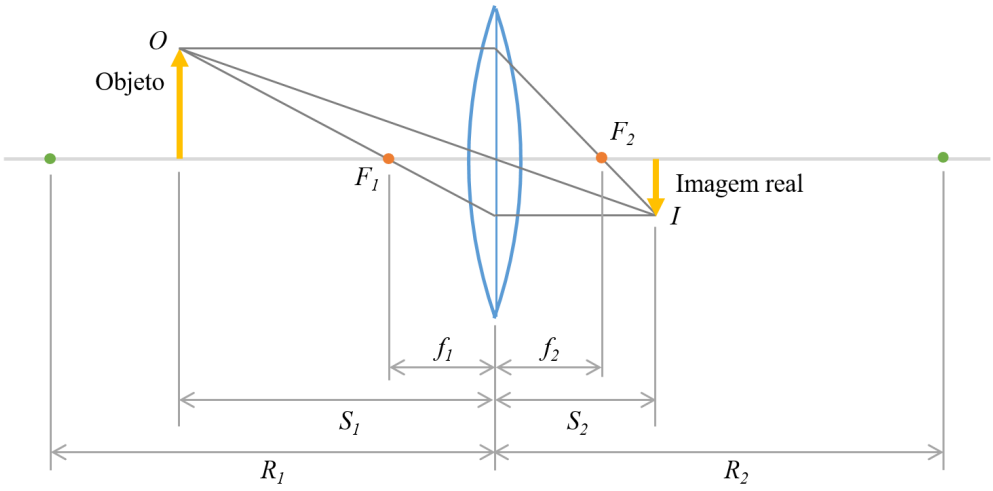
Notas: Representação dos tipos de lentes esféricas possíveis de serem construídas: a) biconvexa; b) plano-convexa; c) côncavo-convexa; d) bicôncava; e) plano-côncava; f) convexo-côncava.

A formação da imagem por uma lente convergente é esquematiza na Figura 5. A situação ilustrada é o caso em que a distância do objeto à lente (expresso por $S_1$) é maior do que a distância $f_1$. Os casos em que a distância $S_1$ é igual ou menor do que $f_1$ resultam na formação de nenhuma imagem ou numa imagem virtual, e que não se aplica ao caso de uma câmera fotográfica. A variação da posição relativa do objeto à lente muda as características da imagem formada tanto em tamanho quanto em posição, medida por $S_2$. A relação entre $S_1$ e $S_2$ para o caso de lentes finas descritas previamente, e ainda supondo que a mesma se encontre imersa em ar[4], o qual possui índice de refração muito próximo a 1, é descrita pela equação das lentes finas ou também conhecida como fórmula dos fabricantes de lentes (HECHT, 2002):

$$\frac{1}{S_1} + \frac{1}{S_2} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right) . \tag{2}$$

[4]A suposição de que a lente esteja imersa em ar não representa uma perda de generalidade grande, visto que é a situação mais comum de ocorrer (KINGSLAKE, 1983).

Figura 5 – Formação de imagem em lentes biconvexas




Fonte: Autor, 2015

Notas: Representação da formação de uma imagem em lentes biconvexas, determinando os principais parâmetros no processo, como os pontos focais, dimensão da imagem, posição da imagem relativa ao objeto.

Para calcular a relação entre as distâncias focais e a posição entre os objetos, faz-se a suposição de que o objeto $O$ seja deslocado para infinito, de tal maneira que distanciando-se da lente sua imagem $I$ tenderá a se tornar um ponto sobre o ponto focal $F_2$. Essa análise é um resultado direto de uma das condições apresentadas anteriormente, de que raios paralelos (colimados) serão direcionados ao ponto focal da lente. É possível ter a mesma conclusão analisando a geometria da Figura 5. Analogamente, para distanciar a imagem $I$ para o infinito, requer que o objeto $O$ seja deslocado para cima do ponto focal $F_1$. Ambas as relações são expressas como:

$$\lim_{S_1 \to \infty} S_2 = F_2 \,, \tag{3}$$

$$\lim_{S_2 \to \infty} S_1 = F_1 \,. \tag{4}$$

Aplicando individualmente (3) e (4) na equação (2), considerando a distância em si, e não a referência à posição, tem-se as seguintes expressões respectivamente:

$$\frac{1}{f_2} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \,, \tag{5}$$

$$\frac{1}{f_1} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \,. \tag{6}$$

Analisando (5) e (6) pelos lados direitos, é simples concluir que:

$$f_1 = f_2 = f \,. \tag{7}$$

Dessa expressão, um fato importante fica em evidencia. A distância focal da lente em ambos os lados é igual, mesmo que os raios de construção da lente sejam diferentes. Ainda, aplicando o resultado em (2), tem-se:

$$\frac{1}{f} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right) . \tag{8}$$

O resultado da comparação entre (2) e (8) é uma das relações mais importantes na ótica geométrica, e também conhecida como formula de lentes de Gauss (HECHT, 2002):

$$\frac{1}{f} = \frac{1}{S_1} + \frac{1}{S_2} . \tag{9}$$

Uma segunda equação importante é aquela que define o tamanho entre imagem e objeto, que é a amplificação $M$. A determinação matemática de tal relação é decorrência natural da análise da semelhança entre triângulos vistos na Figura 5. Denominando o tamanho do objeto por $o$ e o da imagem por $i$, fica evidente que a relação entre ambos são:

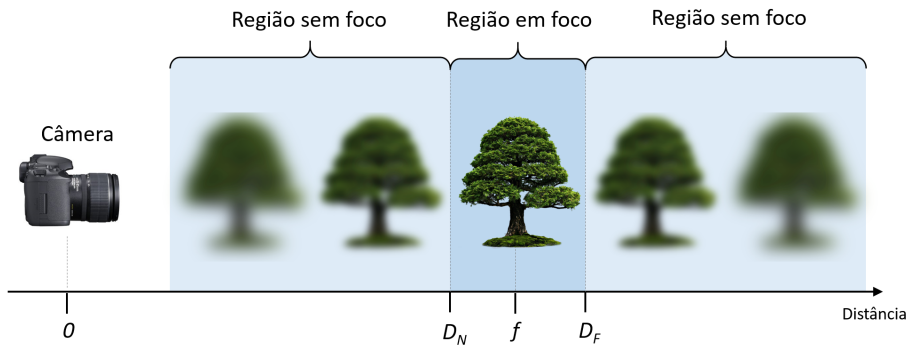
$$M = \frac{o}{i} = \frac{S_1}{S_2} . \tag{10}$$

A equação acima representa a relação de amplificação em termos de módulo apenas (HALLIDAY; RESNICK; WALKER, 2010).

## 2.3 PROFUNDIDADE DE CAMPO

A profundidade de campo é uma limitação inerente a sistemas óticos de lentes, e define a região que está em foco para o mesmo, delimitada por uma distância máxima e mínima. A profundidade de campo é comumente referenciada pela abreviatura DOF do termo em inglês (*depth of fiel*d), e é algo corriqueiro em fotografia, pois é um efeito muitas vezes desejável por quem registra uma imagem (LONG, 2012). Contrário à expectativa comum, uma imagem dificilmente possui foco em todos os seus pontos. O foco completo está presente em um dos planos (ponto focal) do objeto registrado, e a medida que a distância dos demais objetos fotografados aumenta em relação a esse, o grau de desfoque cresce. No entanto, o desfoque nem sempre é perceptível, por ser muito pequeno. A Figura 6 ilustra a questão, enquanto a Figura 7 é um exemplo de imagem com pouca profundidade de campo, resultando numa maior percepção do desfoque à medida que os objetos se distanciam do plano focal (sobre o moedor de pimenta).

Figura 6 – Ilustração de percepção da profundidade de campo através de uma câmera fotográfica (ou outro sistema ótico)




Fonte: Autor, 2015

Notas: Destaca-se na imagem que apenas uma região delimitada por $D_N$ e $D_f$ está em foco, e todas as demais regiões, tanto mais próximas como mais distantes, sem foco.

Figura 7 – Exemplo de foto com baixa profundidade de campo

Fonte: Autor, 2015

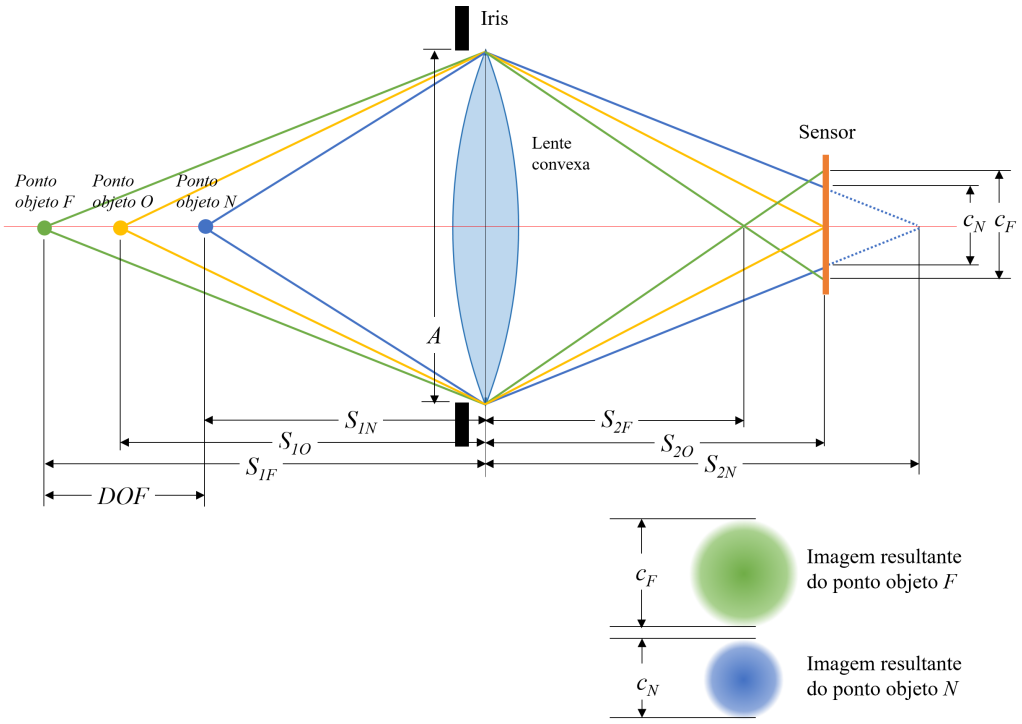
Notas: Foto de exemplo de baixa profundidade de campo, na qual apenas o moedor de pimentas ao lado direito está em foco, e objetos no mesmo plano, como parte da mesa. Os demais objetos da imagem (segundo moedor ou parede) estão fora de foco. É possível observar nessa imagem que ao aumentar à distância do plano em foco, maior é o desfoque da imagem.

A razão da ocorrência desse efeito é mais facilmente entendida ao observar a Figura 8. Nessa imagem, fica evidente que apenas o ponto objeto[5] $O$ está em foco sobre o sensor (ou filme) que registra a imagem. Os demais pontos objetos ($F$ e $N$) possuem foco em outros lugares, e sobre o sensor se tornam uma imagem maior, denominada de círculo de confusão. Apesar de seu formato não ser sempre circular, pois isso depende da forma de limitação que foi realizada

[5]Ponto objeto é um termo que designa um ponto qualquer no espaço que pertença a um objeto material. O ponto é um elemento ideal que representa o menor emissor de luz possível dentro de um objeto real (maior).

sobre a lente (iris), o nome é empregado de forma genérica (PRÄKEL, 2009). Ainda é comum um segundo abuso de nomenclatura, pois o termo círculo de confusão também se refere ao seu próprio diâmetro, e abreviado pelas iniciais CoC do inglês (*circle of confusion*). O ponto objeto $F$ está em foco na distância $S_{2F}$, antes de $S_{2O}$ e o ponto objeto $N$, com foco $S_{2N}$, após $S_{2O}$. Ambas imagens chegam ao sensor como círculos de diâmetros $c_F$ e $c_N$, respectivamente. Na figura, o diâmetro da iris que limita a entrada de luz (e limita a área útil da lente) é dado por $A$.

Figura 8 – Ilustração da origem da profundidade de campo




Fonte: Autor, 2015

Notas: Ilustração da origem da profundidade de campo através de três pontos objetos e suas imagens. É possível ver que apenas o ponto objeto $O$ está em foco, enquanto os demais $N$ e $F$ formam um círculo sobre o sensor, com diâmetro que determina o tamanho do círculo de confusão.

É importante notar que os pontos objetos, tanto ao lado próximo ($S_{1N}$), como ao lado distante ($S_{1F}$), são representados por círculos de diâmetros maiores a medida que tais pontos se afastam de $S_{1O}$. Até o momento não foi definido ainda o DOF, pois há uma subjetividade no mesmo. O aumento dos diâmetros dos círculos de confusão formados só passa a ser sensível ao observador a partir de um determinado valor que depende da sua capacidade de resolução. Assim, os valores dados como limites para que haja foco ou não (ou seja, um valor limite do raio do círculo de confusão) dependem do observador (acuidade visual de cada individuo) e da forma que a mesma é apresentada (exemplo: distância da imagem, uso de auxilio como lupas, presença ou não de ampliação na imagem). Para se ter uma ordem de grandeza, um valor comumente aceito é 0,2 mm (JACOBSON et al., 2000) para um observador a 25 cm da imagem final de tamanho aproximado de 8x10". Esse valor de CoC é menor quando analisado do ponto de vista do sensor ou filme da câmera por ter dimensões menores do que a imagem

final (JACOBSON et al., 2000; HIRSCH, 2012; HESS; MCLERNON, 2012). Ainda assim, um valor preciso de CoC para esse trabalho pode divergir: valor limite é uma questão de capacidade de reconhecimento do algoritmo e da sua metodologia aplicada, portanto, não se trata da mesma forma de determinação de um valor limite para o olho humano.

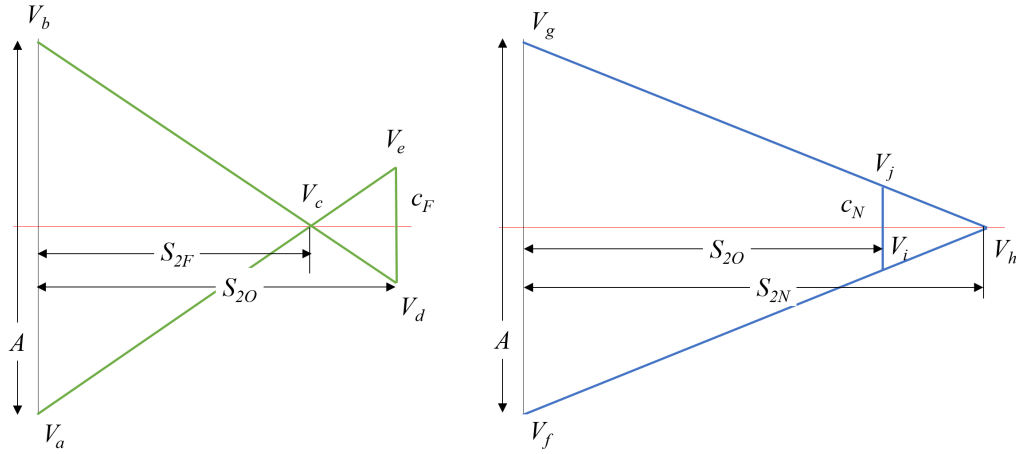
## 2.4 CÁLCULO DO CÍRCULO DE CONFUSÃO

Para determinar a forma com que o valor de CoC varia dentro de uma imagem, pode-se considerar a ilustração da Figura 8. Iniciando pelo uso de relações de semelhança entre triângulos, conforme destacados na Figura 9, nota-se claramente que:

$$\triangle V_a V_b V_c \sim \triangle V_c V_d V_e \,, \tag{11}$$

$$\triangle V_f V_g V_h \sim \triangle V_h V_i V_j \,. \tag{12}$$

Figura 9 – Ilustração de triângulos para determinação do diâmetro do círculo




Fonte: Autor, 2015

Notas: Ilustração de triângulos para determinação do diâmetro do círculo de confusão do sistema ótico sobre um sensor ou filme, via semelhança entre os mesmos: $\triangle V_a V_b V_c \sim \triangle V_c V_d V_e$ e $\triangle V_f V_g V_h \sim \triangle V_h V_i V_j$.

Derivam-se dai as seguintes relações:

$$\begin{aligned} \frac{S_{2F}}{A} &= \frac{S_{2O} - S_{2F}}{c_F} \,, \\ \frac{c_F}{A} &= \frac{S_{2O} - S_{2F}}{S_{2F}} \,, \\ \frac{S_{2N}}{A} &= \frac{S_{2N} - S_{2O}}{c_N} \,, \end{aligned} \tag{13}$$

$$\frac{c_N}{A} = \frac{S_{2N} - S_{2O}}{S_{2N}} \,. \tag{14}$$

Usando o mesmo valor de CoC tanto para o lado longe, quanto para o lado próximo, pois o interesse é definir o valor geral do mesmo $c$ para ambos os lados, tem-se:

$$c = c_N = c_F \,.$$

Aplicando tal relação e igualando (13) e (14), obtêm-se:

$$\frac{S_{2O} - S_{2F}}{S_{2F}} = \frac{S_{2N} - S_{2O}}{S_{2N}} = \frac{c}{A} \,. \tag{15}$$

Como é comum em fotografia, o valor de $A$ é expresso em termos do seu *f-number*, que é dada pela razão entre a distância focal de uma lente e o tamanho da abertura:

$$N = \frac{f}{A} \,. \tag{16}$$

Aplicando então em (15):

$$\frac{S_{2O} - S_{2F}}{S_{2F}} = \frac{S_{2N} - S_{2O}}{S_{2N}} = \frac{N.c}{f} \,.$$

Rearranjando:

$$S_{2F} = \frac{f}{(f + N.c)}.S_{2O} \,, \tag{17}$$

$$S_{2N} = \frac{f}{(f - N.c)}.S_{2O} \,. \tag{18}$$

Assim, usando a Equação (9) e manipulando de forma a ter o termo referente à posição do objeto isolado:

$$\frac{1}{S_1} = \frac{1}{f} - \frac{1}{S_2} \,,$$

$$\frac{1}{S_1} = \frac{S_2 - f}{f.S_2} \,,$$

$$S_1 = \frac{f.S_2}{S_2 - f} \,.$$

Aplicando a ambas as regiões de foco, ou seja, próximo e longe:

$$S_{1F} = \frac{f.S_{2F}}{S_{2F} - f} \,, \tag{19}$$

$$S_{1N} = \frac{f.S_{2N}}{S_{2N} - f} \,. \tag{20}$$

Para o ponto em foco, a equação será escrita de modo a isolar a posição da imagem. Essa necessidade diferente será vista adiante. É simples ver que há uma simetria nas equações, o que resulta em:

$$S_{2O} = \frac{f.S_{1O}}{S_{1O} - f} \,. \tag{21}$$

Aplicando (21) em (17):

$$S_{2F} = \frac{f}{(f + N.c)} . \frac{f.S_{1O}}{S_{1O} - f} \,.$$

E usando esse resultado em (19):

$$S_{1F} = \frac{f.\left(\frac{f}{(f+N.c)} . \frac{f.S_{1O}}{S_{1O}-f}\right)}{\frac{f}{(f+N.c)} . \frac{f.S_{1O}}{S_{1O}-f} - f} = \frac{\frac{f^3.S_{1O}}{(f+N.c).(S_{1O}-f)}}{\frac{f^2.S_{1O}-f.(f+N.c).(S_{1O}-f)}{(f+N.c).(S_{1O}-f)}} \,,$$

$$S_{1F} = \frac{f^3.S_{1O}}{f^2.S_{1O} - f.\left(f + N.c\right).\left(S_{1O} - f\right)} = \frac{f^2 S_{1O}}{f.S_{1O} - \left(f.S_{1O} - f^2 + N.c.S_{1O} - N.c.f\right)} \,,$$

$$S_{1F} = \frac{f^3.S_{1O}}{f^2.S_{1O} - f.\left(f + N.c\right).\left(S_{1O} - f\right)} \,,$$

$$S_{1F} = \frac{f^2.S_{1O}}{f^2 - N.c.\left(S_{1O} - f\right)} \,. \tag{22}$$

Analogamente, para o objeto ao lado mais próximo:

$$S_{1N} = \frac{f^2.S_{1O}}{f^2 + N.c.\left(S_{1O} - f\right)} \,. \tag{23}$$

A profundidade de campo é definida como a diferença entre ambas as equações, como é visto na Figura 8:

$$DOF = S_{1F} - S_{1N} = \frac{f^2.S_{1O}}{f^2 - N.c.\left(S_{1O} - f\right)} - \frac{f^2.S_{1O}}{f^2 + N.c.\left(S_{1O} - f\right)} \,,$$

$$DOF = \frac{f^2.S_{1O}.\left[f^2 + N.c.\left(S_{1O} - f\right)\right] - f^2.S_{1O}.\left[f^2 - N.c.\left(S_{1O} - f\right)\right]}{\left[f^2 - N.c.\left(S_{1O} - f\right)\right].\left[f^2 + N.c.\left(S_{1O} - f\right)\right]} \,.$$

Para o denominador é evidente a presença de um produto notável da álgebra. Para o numerador, é feita a aplicação da propriedade distributiva:

$$DOF = \frac{2.f^2.S_{1O}.N.c.\left(S_{1O} - f\right)}{f^4 - N^2.c^2.\left(S_{1O} - f\right)^2} \,. \tag{24}$$

Apesar do método diferenciado e da ligeira divergência de nomenclatura, a fórmula está de acordo com a comumente empregada (JACOBSON et al., 2000). Com esse resultado é possível calcular o tamanho da região que estará com um foco aceitável, dentro de um valor de $c$

especifico. Os demais parâmetros da equação são calibrados conforme o aparato usado (abertura da lente $N$, distância focal da lente $f$ e distância focada $S_{1O}$). Outra relação importante é saber a variação do CoC conforme a distância ao sistema ótico. Para isso, manipulando a Equação (22) de modo a isolar o diâmetro do círculo de confusão:

$$S_{1F}.\left[f^2 - N.c.\left(S_{1O} - f\right)\right] = f^2 S_{1O}\,,$$

$$S_{1F}.N.c.\left(S_{1O} - f\right) = -f^2 S_{1O} + f^2.S_{1F}\,,$$

$$c = \frac{f^2}{N.\left(S_{1O} - f\right)}\left(1 - \frac{S_{1O}}{S_{1F}}\right)\,.$$

Igualmente para a Equação (23):

$$c = \frac{f^2}{N.\left(S_{1O} - f\right)}\left(-1 + \frac{S_{1O}}{S_{1N}}\right)\,.$$

É possível reunir ambas as equações em um só resultado, reescrevendo através do módulo. Isso é facilmente observado por meio do fator ao lado direito de cada equação, em que eles são semelhantes, mas com sinais invertidos. Ao fazer isso, o resultado torna-se geral, e não há mais necessidade de diferenciar entre os lados em que estão em foco (próximo ou longe):

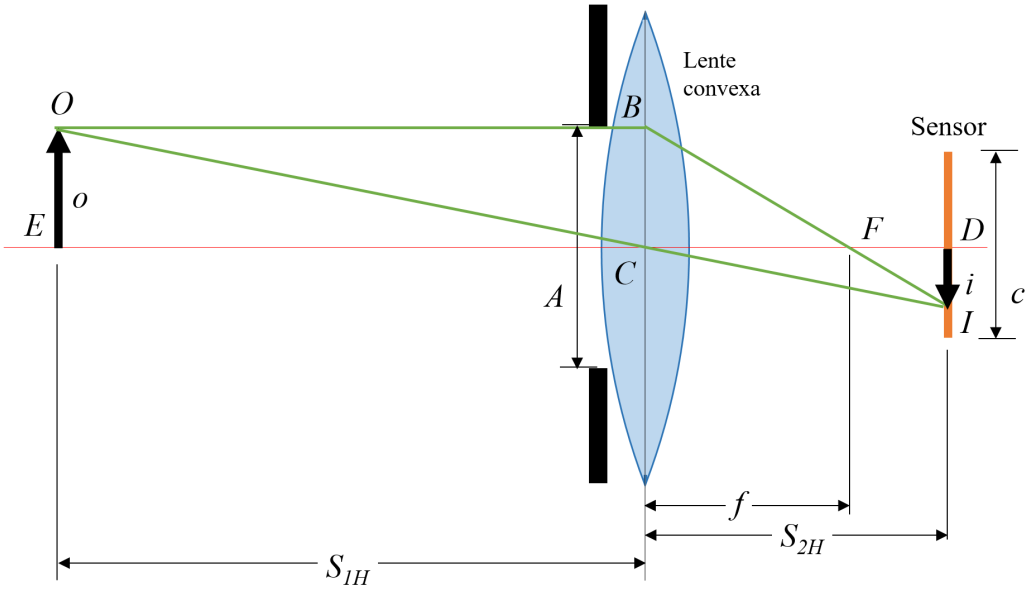
$$c = \frac{f^2}{N.\left(S_{1O} - f\right)}.\left|1 - \frac{S_{1O}}{S_{1NF}}\right|\,. \tag{25}$$

## 2.5 DISTÂNCIA HIPERFOCAL

Antes de analisar qualitativamente os resultados (Equações (24) e (25)), ainda são necessárias algumas obervações sobre um caso especial, mas que poderá ter influencia conforme a abrangência da análise desejada . Existe um caso especifico em que a distância focada irá otimizar a profundidade de campo, particularmente abrangendo o infinito. Essa posição de foco é denominada de distância hiperfocal (SAVAZZI, 2010).

Definindo de forma mais precisa, a distância hiperfocal é aquela mais próxima possível, em que os objetos no infinito ainda encontram-se focados. É interessante usar um contraexemplo para auxiliar tal compreensão: caso o foco fosse colocado exatamente no infinito, o DOF estaria abrangendo posições além desse, o que se torna algo desnecessário. Assim, tomando como referência a Figura 10, onde uma lente convexa é representada com o objeto na distância hiperfocal $S_{1H}$, e analisando por meio de semelhança de triângulos, obtém-se:

Figura 10 – Ilustração da determinação da distância hiperfocal




Fonte: Autor, 2015

Notas: Ilustração da determinação da distância hiperfocal através da ótica geométrica paraxial. A comprovação da magnitude da distância hiperfocal é realizada via análise da semelhança entre triângulos $\triangle OBC \sim \triangle CDI$ e $\triangle BCF \sim \triangle FDI$.

$$\triangle OBC \sim \triangle CDI \, ,$$

$$\triangle BCF \sim \triangle FDI \, .$$

Assim, para o primeiro triângulo:

$$\frac{f}{A/2} = \frac{S_{2H} - f}{c/2} \, ,$$

$$S_{2H} = \frac{c.f}{A} + f \, .$$

E para o segundo triângulo:

$$\frac{S_{1H}}{A/2} = \frac{S_{2H}}{c/2} \, ,$$

$$S_{1H} = \frac{A.S_{2H}}{c} \, .$$

Juntando os resultados e empregando a relação (16), chega-se a equação para a distância hiperfocal:

$$S_{1H} = \frac{A}{c}.\left(\frac{i.f}{A} + f\right) = f + \frac{A}{c}.f\,,$$

$$S_{1H} = f + \frac{f^2}{c.N}\,. \tag{26}$$

Usando o resultado recém obtido em (23):

$$S_{1NH} = \frac{f^2.S_{1H}}{f^2 + N.c.\left(S_{1H} - f\right)}\,,$$

$$S_{1NH} = \frac{f^2.\left(f + \frac{f^2}{c.N}\right)}{f^2 + N.c.\left(\left(f + \frac{f^2}{c.N}\right) - f\right)} = \frac{f^3 + \frac{f^4}{c.N}}{f^2 + N.c.\left(\frac{f^2}{c.N}\right)} = \frac{f + \frac{f^2}{c.N}}{1 + N.c.\left(\frac{1}{c.N}\right)}\,,$$

$$S_{1NH} = \frac{f + \frac{f^2}{c.N}}{2}\,,$$

$$S_{1NH} = \frac{1}{2}.\left(f + \frac{f^2}{c.N}\right)\,, \tag{27}$$

$$S_{1NH} = \frac{1}{2}.S_{1H}\,. \tag{28}$$

Para o lado mais distante $S_{1F}$, fica claro pela própria definição que a mesma está no infinito (SAVAZZI, 2010). Portanto:

$$S_{1FH} \to \infty\,. \tag{29}$$

O resultado prévio deve ser usado em conjunto com (24), delimitados pelo valor de $S_{1H}$ (maiores ou menores).

## 2.6 ANÁLISE DOS FATORES DA PROFUNDIDADE DE CAMPO

O primeiro passo a ser feito com os resultados teóricos anteriores é analisar quais parâmetros influenciam na dimensão da profundidade de campo. Pela Equação (24), e como já mencionado, a medida do DOF varia conforme a abertura da lente $N$, a distância focal da lente $f$, a distância focada $S_{1O}$ e o diâmetro adotado para o círculo de confusão $c$, caso a distância do objeto esteja dentro do limite hiperfocal. Caso a distância seja maior do que esse limite, os parâmetros serão os mesmos, com exceção da distância à lente, podendo ser avaliados na Equação (27), lembrando que a variável $S_{1O}$ passa a ser referida como $S_{1H}$ no caso hiperfocal. No entanto, não é possível determinar de forma simples como varia tal comportamento, pela complexidade da primeira equação.

Inicialmente, os dois parâmetros mais simples de serem avaliados são a abertura da lente $N$ e o diâmetro $c$. É nítido que o $DOF$ será maior quanto maiores forem $N$ ou $c$, pois sua presença no numerador da equação força o aumento do valor do $DOF$ e a outra presença, no

segundo termo do denominador, por estarem todos os fatores elevados ao quadrado, será um número positivo, e pela subtração, fará com que o valor final seja menor conforme tanto $N$ ou $c$ aumentem. O resultado não pode ser negativo, por uma limitação física do problema e geométrica do sistema, então não há necessidade de avaliar tal limite.

O resultado acima é totalmente condizente com a prática ou com a lógica. Para o valor de $c$, é direta a relação que quanto maior for a tolerância ao diâmetro de círculo de confusão, maior será a área dentro desse foco considerado. Já para $N$, é conhecimento comum entre os fotógrafos que, para atingirem o maior *bokeh*[6] possível, devem usar a maior abertura possível de suas lentes e câmeras.

Para os demais parâmetros, é mais interessante a obtenção de uma relação detalhada, e para isso avalia-se conforme uma prática comum em economia: *Ceteris paribus*[7]. Nesse caso, os valores de $N$ e $c$ serão fixados em valores plausíveis, práticos. Sendo assim, definindo $c$ como 5 µm e $N$ como $f/2.8$ [8] o resultado da equação é descrito na Figura 12. Primeiramente, o domínio da Equação (24) é limitado para dentro da região abaixo de 90% da distância hiperfocal:

$$\left\{ DOF\left(S_{1O}, f\right) = \frac{2.f^2.S_{1O}.N.c.\left(S_{1O}-f\right)}{f^4 - N^2.c^2.\left(S_{1O}-f\right)^2} \mid S_{1O} < 0,9.\left(f + \frac{f^2}{c.N}\right) \right\} .$$

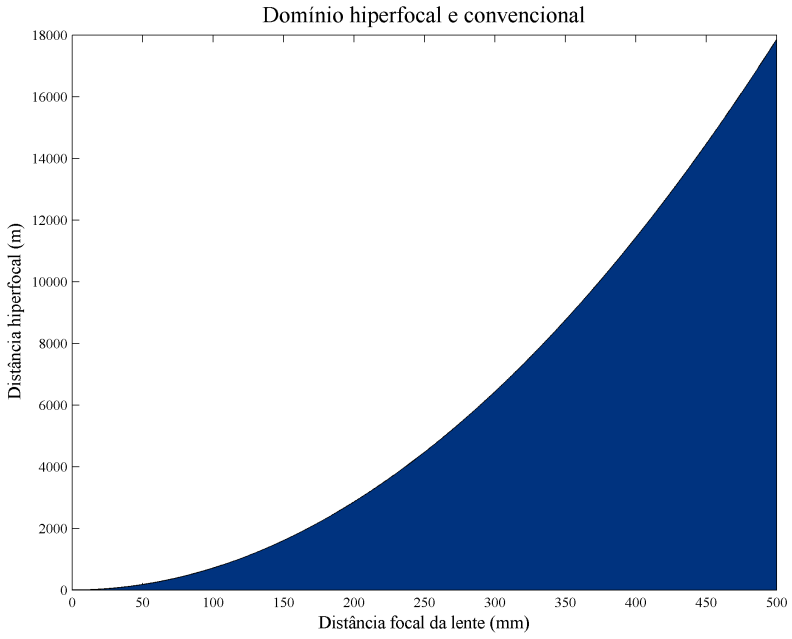
Esse domínio limitado é demonstrado na Figura 11. Na mesma, a região convencional do sistema ótico é destacada em azul, enquanto a região hiperfocal em branco. Essa limitação do domínio será evidente adiante, quando a profundidade de campo for colocada em forma gráfica.

[6] *Bokeh* é um termo de origem japonesa, e que se refere em fotografia a uma composição em que o primeiro plano da foto está em foco, enquanto os demais planos ficam fora de foco, conforme o exemplo na Figura 7 anterior.

[7] Termo em latim que significa "todo os demais são constantes", usual em economia, mas aplicável em diversos campos de estudo.

[8] A abertura de lentes, conforme empregado nesse trabalho e por câmeras fotográficas, é dada pela fração descrita em (16), com a letra $f$ inicial.

Figura 11 – Exemplo de separação entre domínio regular e domínio hiperfocal

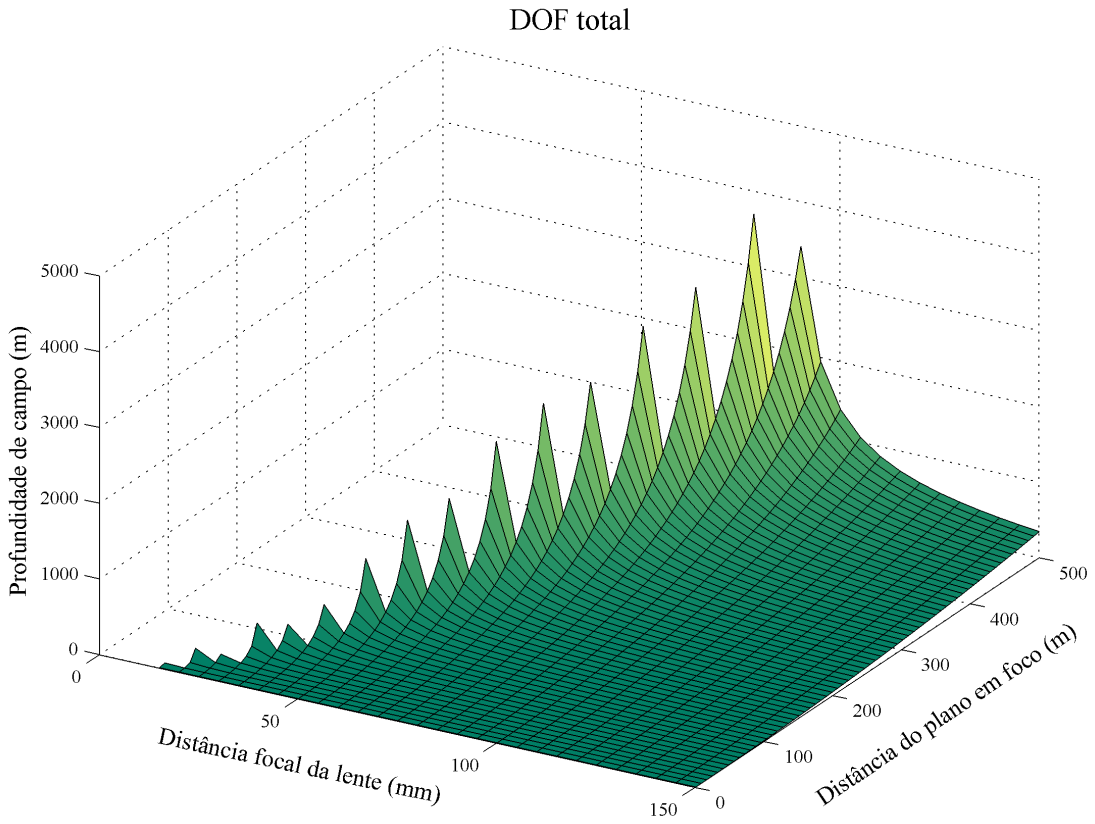



Fonte: Autor, 2015

Notas: Exemplo de separação entre domínio regular e domínio hiperfocal para um sistema ótico, em função da sua distância focal. Para os demais parâmetros conforme Equação (26), foram fixados respectivamente em $f/2.8$ e 5 μm para a abertura e para o diâmetro do círculo de confusão.

Pelo resultado da Figura 12 fica evidente o comportamento monotônico da função para ambas as variáveis $f$ e $S_{1O}$. Para a distância focal, a função é monotonicamente decrescente, o que significa que quanto maior a distância focal da lente, menor é a dimensão da profundidade de campo. De forma oposta, para $S_{1O}$, a função é monotonicamente crescente, o que implica que quanto mais longe o plano de foco estiver da lente, maior será a extensão da profundidade de campo. No gráfico apresentado os valores dos parâmetros usados possuem um intervalo de análise muito além da aplicação de poucos metros com o intuito de ficar mais evidente o comportamento das curvas e a relação entre parâmetros.

Figura 12 – Exemplo de gráfico da dimensão da profundidade de campo




Fonte: Autor, 2015

Notas: Exemplo de gráfico da dimensão da profundidade de campo em função da distância focal da lente e da distância do plano em foco. Os demais parâmetros, abertura da lente e diâmetro do círculo de confusão, foram fixados respectivamente em $f/2.8$ e 5 µm.

Como o intuito do trabalho é usar a baixa profundidade de campo, os fatores que devem ser investigados são aqueles que minimizam tal valor, assim, de forma resumida e baseando-se nos conceitos e resultados da ótica apresentados (especialmente os resultados apresentados na Figura 12), deve-se buscar:

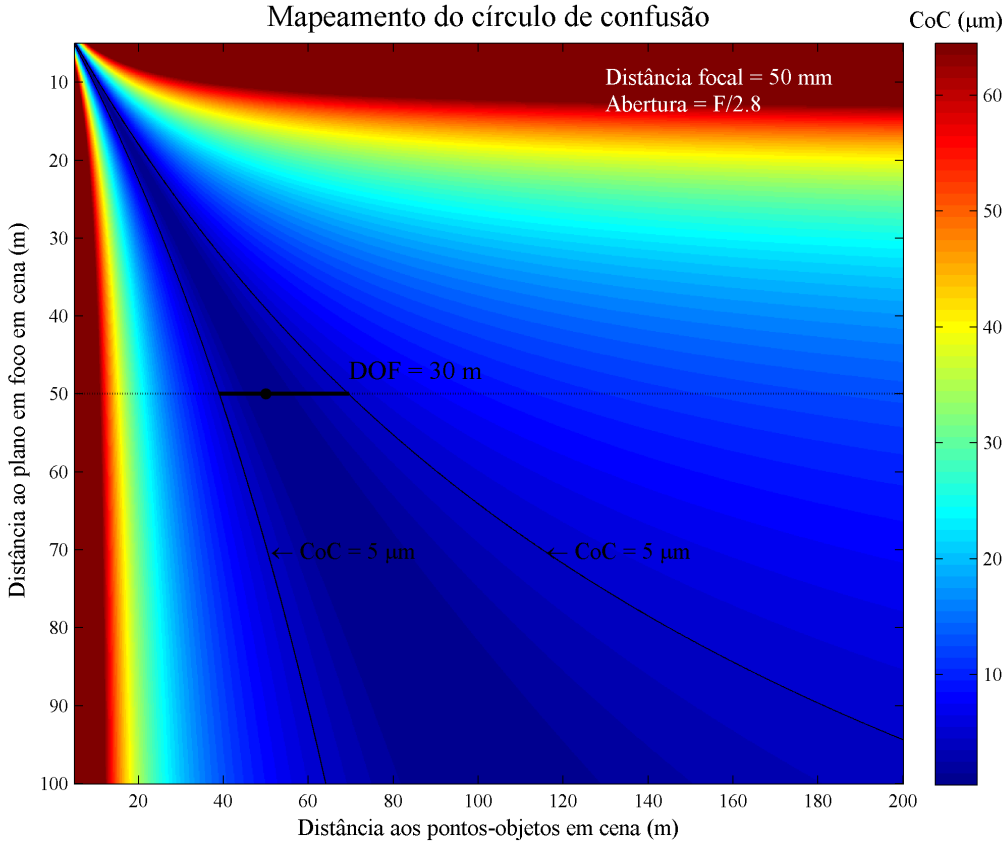
- Abertura máxima. É necessário alertar que ao usar uma abertura máxima, maior quantidade de luz chega ao sensor ou filme numa câmera, o que é interessante para aplicações em que a luz ambiente é limitada. Excesso de luz é facilmente controlado via redução do tempo de captura da imagem ou pela redução da sensibilidade do sensor (ISO);
- Menor distância ao plano de foco possível, considerando que há a limitação da aplicação em que a técnica será usada;
- Maior distância focal possível. Deve-se levar em conta que há a questão da distância do plano em foco, para que não entre na região hiperfocal, conforme mostrado na Figura 11.

A questão da dimensão do círculo de confusão não é influenciada pela ótica, mas sim pela capacidade do algoritmo que analisa a imagem e, portanto, não é listada acima.

## 2.7 ANÁLISE VIA DIMENSÃO DO CÍRCULO DE CONFUSÃO

Uma forma de análise pode ser feita usando a Equação (25). O resultado gráfico é dado pela Figura 13. Os parâmetros comuns são os mesmos da análise exibida na Figura 12, ou seja, abertura de $f/2.8$. Para a distância focal, que passa a ser fixa, o valor empregado é de 50 mm. O círculo de confusão de diâmetro de 5 µm foi destacado no gráfico, tanto na parte longe quanto próxima, e uma das faixas de DOF (em 30 metros para o plano focado) marcada.

Figura 13 – Exemplo de mapa da dimensão do círculo de confusão




Fonte: Autor, 2015

Notas: Exemplo de mapa da dimensão do círculo de confusão em função da distância do objeto ao sistema e da distância focal do mesmo. Demais parâmetros fixados: abertura da lente em $f/2.8$ e distância focal em 50 mm. No gráfico estão destacadas as linhas de diâmetro de círculo de confusão constante, igual a 5 µm, além da faixa de DOF considerando um plano em foco a 50 m do sistema.

Avaliando a Equação (25), pode-se aplicar um caso especial, com conclusões interessantes. Igualando o valor do círculo de confusão a 0 ($c = 0$), considera-se o caso de foco pleno. Assim, para a Equação (25), têm-se:

$$\frac{f^2}{N.(S_{1O}-f)}.\left|1-\frac{S_{1O}}{S_{1NF}}\right|=0\,.$$

Essa equação só se mantém válida no caso em que:

$$S_{1NF}=S_{1O}\,,$$

visto que o valor de $f$ é estritamente positivo, e que os denominadores $N$ e $S_{1O}-f$ não podem ser nulos (e nem negativos). Portanto, numa cena em que são conhecidos os parâmetros fotográficos da sua captura, é possível determinar a distância do objeto em foco visto que $S_{1N}$ e $S_{1F}$ são igualmente conhecidos. Logicamente, valores de $N$ muito grandes implicam numa tendência a zero e, portanto, o ideal é trabalhar com $N$ pequenos, na prática o que significa grandes aberturas óticas.

O resultado pode ser estendido aos casos em que há desfoque, mas com uma maior complexidade, pois a relação demonstrada acima deixa de ser nula, e passa a acompanhar o grau de desfoque (circulo de confusão $c$). Como há uma subjetividade nessa medição (diferente do caso extremo em que é nula), tal cálculo fica sujeito a erros decorrentes da determinação de $c$.

## 2.8 CONSIDERAÇÕES COMPLEMENTARES

Para as equações obtidas, e consequentemente também para seus gráficos, pode-se vislumbrar a possibilidade do uso de sistemas óticos com baixa profundidade de campo a fim de isolar (segmentar) regiões e objetos de uma cena usando critério de foco e desfoco, respectivamente, *foreground* e *background*.

Outra consideração interessante extraída dos cálculos anteriores é a quantidade de desfoque em função da abertura da lente. Mesmo havendo outros critérios que podem ser considerados, a abertura será o principal por se tratar de um parâmetro do equipamento empregado, não dependendo da cena em si.

A consideração acerca da abertura ótica é que apesar de uma profundidade de campo pequena ser muito interessante por ampliar as diferenças entre foco e desfoque na cena, reduzindo os esforços dos algoritmos que avaliam tal imagem, a situação de um valor de DOF extremamente baixo impacta que os próprios objetos em cena terão partes de si em foco e outras não, o que passa a ter efeito negativo. A determinação da dimensão da região em foco, assim como dos parâmetros que levam a isso, são pontos da aplicação a que a técnica proposta nessa dissertação será usada.

Adicionalmente, uma consideração final sobre a uso da abertura ótica como parâmetro principal, é que quanto maior a abertura desejada numa lente objetiva, maior é sua sofisticação de construção, acarretando em maiores custos de fabricação. Deve-se assim ter em mente para cada aplicação possível essa restrição.

# 3 SEGMENTAÇÃO DE IMAGENS

Conforme visto anteriormente, através da profundidade de campo é possível obter mais informações do que simplesmente um mapa de luminância e cores, como geralmente são tratadas as imagens. Apesar da abordagem proposta ser o uso de duas imagens para segmentação, ainda assim é possível fazer uma avaliação de outros métodos que tenham igualmente o objetivo de agrupar foco e desfoco. Obviamente os mesmos usarão apenas uma imagem neste processo.

Desta forma, é possível discutir algoritmos já conhecidos e divulgados na literatura que possuem o propósito de segmentar dentro de uma imagem aquela região que está em foco, separando-a do restante da imagem. Paralelamente, como fruto do uso de duas imagens com diferentes profundidades de campo simultaneamente, é aberta a possibilidade da criação de novos procedimentos. Quando possível, análise experimentais são realizadas, afim de avaliar a capacidade dos métodos, seja do ponto de vista de tempo de execução ou da qualidade da segmentação obtida.

## 3.1 CONCEITO GERAL

Em geral, os algoritmos de segmentação, independentemente de usarem uma ou duas imagens, trabalham classificando os pixels em dois grupos: com foco ou sem foco. É evidente que um pixel sozinho não possui informação suficiente para classificar se há presença de foco ou não, e a análise, portanto, é realizada num nível de conjunto de pixels, onde a informação necessária está presente.

Todas as aplicações de algoritmos, por serem métodos computacionais digitais, ficam limitadas a tal domínio, bem como suas entradas: as imagens. Em outras palavras, as imagens em estudo serão sempre digitais, o que implica em discretização espacial das intensidades das grandezas que representam a imagem. A discretização espacial significa que há uma dimensão mínima para os elementos de captura e reprodução da imagem, denominados de pixels[1]. Pela prática atual em computadores, as imagens são usualmente definidas no domínio de cores RGB com 24-bits (8-bits por cor).

O resultado da segmentação das imagens pelos algoritmos deve ser uma matriz binária, o que é consequência da segmentação em dois níveis apenas, com dimensão igual à imagem original. Portanto, sendo uma imagem de entrada de dimensões (em número de pixels) $M$ na direção vertical por $N$ na direção horizontal (uma matriz $M$-por-$N$), o resultado deverá ser uma outra matriz também $M \times N$. Por características ou limitação de análise de algoritmos, é possível que haja uma redução do resultado nas bordas. O tamanho de tal redução varia conforme o método, mas em todos os casos essa limitação é desconsiderada, por ter pouco impacto prático, visto que é possível compensar o problema aumentando o tamanho da imagem, ou mesmo pela

[1] Não é possível dizer sobre uma quantidade máxima de pixels numa imagem, pois a expansão lateral (ou superior/inferior) da imagem é sempre possível, agregando mais dados. A limitação reside na capacidade do computador em armazenar e processar tal volume de dados.

característica em si da aplicação, na qual as bordas possuem pouca ou nenhuma informação revelante.

Os métodos de segmentação de objetos em foco possuem fundamentação em três tipos de abordagens (AHN; CHONG, 2015):

a) Segmentação a partir das bordas - Baseia-se no fato das bordas da região com foco possuírem bordas salientes em relação ao restante da imagem (sem foco);

b) Segmentação a partir do conteúdo - Baseia-se no fato de parte da imagem que está em foco possuir maior detalhamento, e, por isso, componentes de alta frequência no domínio espectral;

c) Segmentação por características (*feature*) - Baseia-se no fato de que os objetos em foco devem possuir uma uniformidade ou que sejam concisos, assim existem características semelhantes entre os pixels contidos por ele, ao passo que elementos fora dessa região devem ter características distintas. Exemplos de características são cor ou textura. No entanto, ainda é necessário que aja em conjunto com técnicas de segmentação por conteúdo para localização dos detalhes.

Adicionalmente, uma metodologia conjunta entre as técnicas descritas acima pode ser também adotada.

## 3.2 TRABALHOS ANTERIORES

Historicamente, as primeiras propostas de algoritmos para segmentação de regiões de foco em imagens datam de 1994, com as publicações (KAO et al., 1994) e (YANG; YANG; YANG, 1994). Ao longo dos anos novas abordagens foram propostas, buscando melhorar a qualidade do resultado ou do desempenho (tempo de execução). As técnicas mais recentemente publicadas (NEVEROVA; KONIK, 2012) e (AHN; CHONG, 2015), além de proporem novas metodologias, realizam comparações entre algoritmos com objetivos semelhantes, usando banco de dados comum entre si. Os resultados dos autores comprovam uma melhora na qualidade da segmentação final frente aos demais métodos existentes.

Um número de algoritmos divulgados na literatura foram analisados e estão descritos a seguir. A escolha dos mesmos é baseada em fatores históricos (primeiros), por abordagem empregada (por exemplo, o uso de *wavelets*) ou pelos resultados apresentados pelos autores serem mais abrangentes. Para cada análise de algoritmo, foi adotado o princípio de manter o mais semelhante possível os nomes de variáveis conforme usado pelos autores do artigo.

Todas as técnicas possuem em comum o fato de usarem apenas uma imagem como entrada de informação para fazerem a classificação dos pixels em foco ou desfocado.

### 3.2.1 Filtro de rampa linear e aplicação de CNN

Os autores do artigo (KAO et al., 1994) fazem uma análise da ótica, de como a falta de foco causa, num caso extremo de uma alteração abrupta de luminância/cor, uma transição

dada pela função matemática descrita por eles. Essa função é nitidamente muito próxima de uma reta e, portanto, assume que o efeito da falta de foco está relacionado com uma função linear. Assim, criam um filtro (muito similar a um passa-alta) que denominam de "filtro de rampa" e aplicam o mesmo na imagem. O filtro detecta apenas as bordas e, portanto, fazem um processamento de fechamento de bordas, onde fecham pontos abertos nas bordas e depois aplicam um preenchimento para segmentar a região com foco. De forma conjunta (pois possuem conteúdo similar) os autores do artigo (YANG; YANG; YANG, 1994) sugerem que o algoritmo descrito em (KAO et al., 1994) possui um tempo de execução alto, e assim aplicam *Cellular Neural Network* (CNN) para implementação e execução do mesmo algoritmo.

### 3.2.2 Filtro Sobel compensado

No artigo (SWAIN; CHEN, 1995), visa-se a redução de quantidade de informação na transmissão de vídeo. Os autores aliam a detecção de regiões sem foco com análise de movimentos em imagens. Fazem uma união dos resultados (mas não de forma interativa) para resolverem diversas situações, assim, uma técnica compensa a outra quando uma delas possui limitação. Para a questão das partes sem foco, empregam um filtro de Sobel, mas compensado (dividido) pela largura do contorno. Assim, a aplicação desse cálculo resulta num valor que detecta bordas e que resultará num valor maior ou menor pela largura dessa borda. Os autores alegam que uma transição (que seria detectável pelo Sobel) em baixo foco deve ter uma largura maior. Assumem que uma pessoa está definindo a imagem, colocando em foco o objeto de destaque (OOI = *Object of Importance*). Também requer uma operação morfológica de preenchimento da imagem, visto que só extrai o contorno.

### 3.2.3 Wavelet de Haar

O algoritmo criado em (WANG et al., 2001) não emprega a análise de bordas, mas faz uma abordagem do conteúdo a partir do princípio já citado de que a região com foco numa imagem possui componentes de alta frequência e, para tanto, possuem o objetivo de medir a energia de tais elementos. Para fazer tal localização dentro da imagem, é empregada uma análise de multirresolução de wavelet. Por não depender das bordas, os autores acreditam ter uma solução mais robusta para imagens naturais, visto que as bordas do objeto de interesse numa imagem podem não ser sempre discerníveis (por se misturarem com o fundo, ou por terem uma transição muito suave).

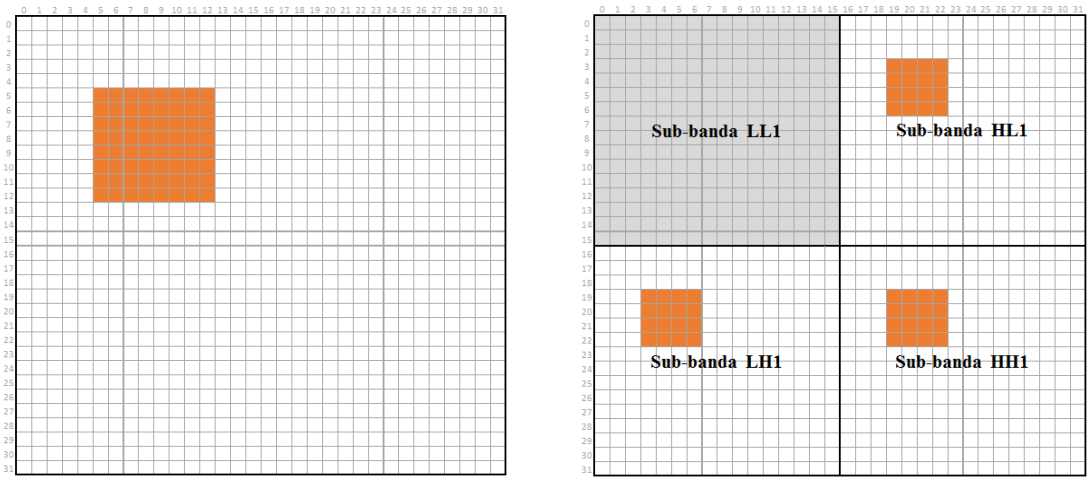
A transformada discreta de wavelet (TDW) realiza a decomposição iterativa da imagem em partes de alta frequência e baixa frequência, sendo ambas subamostradas na sequência, resultando nos coeficientes da transformada. Como imagens são sinais em duas dimensões, o resultado da transformada em cada escala será composta de 3 componentes com alta frequência (LH, HL, HH) e uma componente de baixa frequência pura (LL). Os autores definem a variável $v$ como sendo a variância entre os pixels das 3 sub-bandas com altas frequências, lembrando que cada sub-banda de alta frequência possui a mesma informação espacial, assim, os pixels de cada

uma possui a informação de um mesmo ponto (ou regiões, visto que ocorre a subamostragem). A variância possui uma relação direta com a energia do sinal. A Figura 14 ilustra esse resultado. Os autores optam por usar a wavelet de Haar pelo baixo custo computacional (simplicidade) e pelo seu tamanho diminuto, que apresenta uma boa localização dos pontos em análise.

Figura 14 – Iteração da transformada de wavelet

(a) Imagem exemplo com $32 \times 32$ pixels.

(b) Aplicação da transformada discreta de wavelet com as 4 resultantes.




Fonte: Autor, 2015

Notas: Ilustração de uma iteração da transformada discreta de wavelet. Em (a) uma imagem qualquer de 32 por 32 pixels é representada, sendo que um ponto qualquer da mesma é destacado. Em (b) é ilustrada a primeira etapa da transformada de wavelet, nas quais o mesmo ponto da imagem original é localizado para representação da mudança em posição.

A energia de um sinal (ou imagem, num caso especial de sinal em duas dimensões) pode ser calculada conforme abaixo, e, obviamente, podendo ser sub-definida, em partes do sinal:

$$\varepsilon = \sum_{i,j=1}^{N,M} \left| x_{i,j} \right|^2 .$$

E a relação de energia para um sinal é expressa por:

$$\begin{aligned} v^2 &= \left(\frac{\varepsilon}{N.M}\right) - \mu^2 = \left(\frac{\varepsilon}{N.M}\right) - \left(\frac{1}{N.M} . \sum_{i,j=1}^{N,M} x_{i,j}\right)^2 \\ &= \frac{1}{N.M} . \left[ \left( \sum_{i,j=1}^{N,M} \left| x_{i,j} \right|^2 \right) - \left( \sum_{i,j=1}^{N,M} x_{i,j} \right)^2 \right] . \end{aligned}$$

A variância de um ponto numa imagem subamostrada (sub-bandas) de coordenadas $i$ e $j$ pode ser expressa conforme abaixo, onde tem-se sempre que $A = 3$, e $x_{i,j}^{(k)}$ significa o valor do coeficiente da transformada de wavelet no ponto $i$ e $j$, na sub-banda $k$, e $\mu$ é o valor médio dos pontos, lembrando que todas as sub-bandas possuem o mesmo tamanho $I$ e $J$, ou seja:

$$v = \frac{1}{I.J.A} \sum_{i,j,k=1}^{I,J,A} \left( x_{i,j}^{(k)} - \mu \right)^2 .$$

No algoritmo proposto pelos autores, primeiro é feita uma classificação inicial, na qual é definido um bloco quadrado (mas que poderia ser retangular, por não haver restrição quanto ao formato e tamanho) inicial, preferencialmente de tamanho dado por uma potência de 2. A argumentação é que o tamanho ideal por eles encontrado é 32 por 32 pixels, empregado em imagens de 768 por 512 pixels, pois resulta em valores mais discerníveis entre OOI e o background. Para a imagem dividida em blocos de tal tamanho, é feito o cálculo da variância conforme descrito, chegando a um valor por bloco. Também é feito o cálculo da média de intensidades para cada bloco. Ao resultado dos $v$ é então aplicado uma clusterização de 2 níveis por K-Means no próprio espaço das variâncias (ou energia), sendo que dessa forma haverá um *cluster* para altas energias (foco) e outro para baixas energias (sem foco), respectivamente, com centros $v^{(1)}$ e $v^{(0)}$. Pequenas regiões de background são apagadas, visto que elas podem ser na verdade regiões de interesse, mas que tenham pouca informação (por exemplo, um plano monocromático e com pouca textura que faça parte do objeto em foco).

Na sequência, cada bloco inicial é dividido em 4 partes de tamanhos iguais, sendo que tal divisão será repetida até que atinja-se o valor mínimo de tamanho de regiões (definido pelo usuário, conforme sua necessidade de resolução) ou até que o tamanho seja de um pixel. Para cada nova divisão, os valores de $v$ e $\mu$ são recalculados. Com cada divisão, a classificação dos blocos gerados é uma cópia dos seus "pais" (blocos que deram origem aos atuais, via divisão), ou seja, se o bloco é elemento do OOI ou do *background*.

Em cada *loop* de divisão por 4, os novos valores de $v$ e $\mu$ são analisados, empregando uma lógica de comparação dos valores novos com os dos "pais" e/ou com os 4-vizinhos (superior, inferior, laterais esquerdo e direito), e dessa forma, remarcando a classificação dos blocos entre OOI ou do *background*. A medida que há uma diminuição do tamanho dos blocos em análise, o valor de $v$ deixa de ser empregado, por possuir pouca representatividade. Nesses casos, o valor de $\mu$ predomina.

Ao final, o resultado é submetido a correções morfológicas, removendo pequenas áreas e fazendo o alisamento de bordas.

Apesar de excelentes resultados, onde a sensitividade e a sensibilidade atingiram valores acima de 95%, o algoritmo apresenta limitações, como no caso em que o OOI é muito "liso" ou "suave", onde ele possuirá características iguais ao do *background* ou quando a imagem possui uma resolução muito baixa. Outras limitações citadas são comuns ao princípio da segmentação de imagens com baixo DOF (imagens com DOF muito limitado, onde o próprio OOI não possui foco completo).

### 3.2.4 Filtros e análise morfológica no espaço de cores CIE L*a*b*

Os autores do artigo (GRAF; KRIEGEL; WEILER, 2011) fazem um algoritmo dividido em 5 etapas. Todo trabalho é executado no espaço de cores CIE L*a*b*. Primeiro, dada a imagem $I$ é feito o cálculo da média de cada componente das cores numa vizinhança quadrada de lado $l = 2.r + 1$, onde $r$ é a dimensão especificada do tamanho da vizinhança, centrado em cada pixel da imagem. A título de exemplo, para a componente L, o cálculo é definido como

$$L_{\eta^r_{I(x,y)}} = \frac{\sum_{p \in \eta^r_{I(x,y)}} \left(L^*_p\right)}{\left|\eta^r_{I(x,y)}\right|} ,$$

que trata-se da média aritmética (soma dos valores dos elementos dentro da região $\eta^r_{I(x,y)}$ centrada no pixel $(x, y)$, dividida pela quantidade de pixels na mesma região, ou seja, $\left|\eta^r_{I(x,y)}\right|$ ). Para os demais componentes o método é totalmente análogo.

O resultado da média da vizinhança de cada pixel é denominado $Lab_{\eta^r_{I(x,y)}}$, sendo que o cálculo repetido para todos os pixels da imagem irá compor uma nova imagem (ou 3 novas imagens, uma para cada componente de cor). Os componentes de cores são individualmente denominados:

$$\left(\bar{L*}, \bar{a*}, \bar{b*}\right) = \left(L_{\eta^r_{I(x,y)}}, a_{\eta^r_{I(x,y)}}, b_{\eta^r_{I(x,y)}}\right) .$$

A partir de então chega-se à formação de uma nova imagem, a partir da diferença (euclidiana) entre a imagem original (seus 3 componentes) e a imagem formada das médias (3 componentes). Esse resultado é normalizado entre 0 e 255, isto é,

$$\Delta\eta^r_{I(x,y)} = \min\left(255.\frac{\Delta E * \left(Lab_{\eta^r_{I(x,y)}}, p\right)}{\Delta E^{\max}}, 255\right) ,$$

onde $\Delta E * (u, v)$ é a distância euclidiana entre os pontos $u$ e $v$. Na prática o efeito é de um filtro passa-alta, visto que a aplicação do valor médio numa vizinhança tem o mesmo efeito da aplicação de um filtro passa-baixa, a exemplo de um filtro $F$

$$F = \frac{1}{9}.\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} ,$$

no qual vê-se que a operação de convolução resultará para cada pixel na média aritmética de todos os elementos da região. Já a diferença entre o espectro da imagem original e resultado da filtragem passa-baixa leva a uma outra imagem formada dos componentes restantes, ou seja, de alta frequência. Assim, o resultado final é equivalente à aplicação de um filtro passa-alta.

Em seguida é feito o seguinte cálculo:

$$\mu\left(x, y\right) = \left(\Delta\eta^r_{I(x,y)} - \Delta\eta^r_{I_\sigma(x,y)}\right)^2 ,$$

onde $\Delta\eta^r_{I_\sigma(x,y)}$ é o cálculo análogo para a imagem $I$ original aplicada após uma convolução gaussiana de raio $\sigma$. Assim, repete-se a conclusão anterior, de que tal operação resulta numa filtragem passa-alta. No entanto, dessa vez é feito o cálculo do quadrado do resultado, implicando em um valor relacionado com a energia da imagem. O valor é comparado com um limiar $\Theta_{score}$, e o resultado é a criação de uma nova imagem, tal que

$$I_{score}(x,y) = \begin{cases} 0 & \mu(x,y) < \Theta_{score} \\ \mu(x,y) & \text{caso contrário.} \end{cases},$$

sendo que $I_{score}$ é dividido por 2 para melhorar a velocidade de processamento, sem impactar na precisão. O resultado dessa etapa é a localização dos pixels de alta variação (foco). Em seguida, a imagem $I_{score}$ é clusterizada pelo algoritmo DBSCAN, empregando como seus dois parâmetros $\varepsilon$ e $minPts$, respectivamente:

$$\varepsilon = \sqrt{|I|}.\Theta_\varepsilon$$

$$minPts = \left\lfloor \frac{|\{(x,y)\,|I_{score}(x,y) > 0\}|}{|I|}.(\varepsilon+1)^2 \right\rfloor . \tag{30}$$

Na primeira equação observa-se que o raio de análise do algoritmo DBSCAN é variável com o tamanho da imagem (pois $|I|$ representa a quantidade de pixels na imagem) e de um fator $\Theta_\varepsilon$ definido entre 0 e 1. Isso proporciona maior dinamismo ao algoritmo para trabalhar com diversos tipos de imagens. Já a Equação (30) determina o número mínimo de pontos dentro do círculo de análise para considerar como cluster ou não. Igualmente, esse valor é aplicado de forma dinâmica pelos autores. Com os *clusters* calculados, lembrando que o algoritmo DBSCAN irá formar um número não paramétrico de elementos, é feita a análise dos tamanhos dos clusters. Pequenos clusters são removidos por ser assumido que tratam-se de ruído. O tamanho de corte é definido com a metade do tamanho (número de pixels) do maior cluster encontrado, compondo assim um novo conjunto de clusters $\hat{C}$, tal que

$$\hat{C} = \left\{ c \in C \;\mid\; |c_i| \geq \frac{\max\{|c_1|, |c_2|, \ldots, |c_n|\}}{2} \right\}.$$

Tais clusters reconhecidamente não formarão a imagem que está em foco, mas regiões dessas. A etapa seguinte é então realizar a operação morfológica de preenchimento convexo (*convex hull*) das vizinhanças $N_{Eps}(p)$, ou seja, realiza-se o preenchimento de cada cluster individualmente, dando origem a polígonos $H = \{H_1, H_2, \ldots, H_k\}$. E então varre-se a imagem original $I$, formando uma imagem binária, onde o valor 1 é atribuído aos pixels que estão dentro de um dos polígonos formados pelos preenchimentos convexos anteriores, ou 0 no caso contrário:

$$I_{app}(x,y) = \begin{cases} 1 & \text{se } \exists H_i : p(x,y) \in H_i \\ 0 & \text{caso contrário.} \end{cases},$$

Em seguida, filtros morfológicos são aplicado em $I_{app}$, sequencialmente às operações de fechamento e de reconstrução por erosão, o que possui o efeito de fechar buracos na região segmentada. Segundo os autores, o resultado analisado em diversos casos consegue cobrir o OOI com relativo sucesso, no entanto com um efeito indesejado de criar uma borda larga, sobressalente, o que é tratado nos passos subsequentes.

Todos os pixels de $I_{app}$ são visitados e os vizinhos de cada um analisado em termos de diferença de cor, sendo que no caso de $\Delta E*$ ser menor do que um limiar pré-definido. Sempre que os pixels se enquadrem na condição do limiar, então são marcados como visitados e dentro de um grupo $R$ em análise. A repetição de tal análise (iterações), até que todos os pixels da imagem sejam visitados, irá gerar um conjunto de grupos $R_{color}$, sendo os elementos tratados por $r$.

Para cada região $r$ é feita a construção de um contorno via a dilatação da mesma, menos ela própria, o que é denominado de região $B_r$. Em cada região $B_r$ é analisada a interseção dessa com as demais regiões de $R_{color}$, e feita a contagem de pixels da mesma ($BO_r^R$). O resultado dá origem ao cálculo a seguir:

$$MBO_r = \frac{BO_r^{R_{color}}}{|B_r|} \, .$$

Por um método análogo, é calculado também o valor abaixo para cada região:

$$SBO_r = \frac{BO_r^{\hat{C}}}{|B_r|} \, .$$

A diferença desse segundo valor de mérito para o primeiro é que o cálculo agora é baseado na intersecção da região em análise com os clusters calculados antes das etapas morfológicas ($\hat{C}$). Por fim, o valor de relevância é calculado como:

$$MR_r = SBO_r.MBO_r \, .$$

O cálculo dos valores de $MR_r$ são feitos iterativamente. Os valores são calculados para todas as regiões $r$ numa etapa inicial. As regiões $r$ com valores abaixo de $MR_r$ são apagadas. A perda de regiões afeta o cálculo de $MBO_r$ (mas não o de $SBO_r$), o que leva a mais uma etapa de cálculos. O critério de determinação se uma região $r$ será apagada ou não é dinâmico, variando com o número de iterações. No entanto, a regra geral é que apenas regiões pequenas sejam apagadas. A ciclo de cálculos tem fim quando não há mais nenhuma região a ser deletada. O resultado da segmentação são as regiões restantes.

### 3.2.5 Desvio padrão do perfil (secção) de bordas

O algoritmo criado por Konik e Neverona em (NEVEROVA; KONIK, 2012) aproveita tanto a questão das bordas, quanto do conteúdo para a segmentação de objetos de interesse na imagem. Esse algoritmo se baseia no fato de que bordas naturais são relativamente suaves e a magnitude do gradiente das mesmas tende a seguir um comportamento de uma distribuição

Gaussiana. É um dos método mais recentes e com melhores resultados apresentado na literatura cientifica atualmente. Como será estudado mais profundamente que os demais, pela sua posição de destaque, será referido como algoritmo NK (iniciais dos seus autores) por simplicidade.

Primeiro é aplicado um detector de bordas de Canny modificado para que trabalhe em cada cor separadamente e então combinados via cálculo RMS. Apenas como exemplo, a equação abaixo representa o resultado $F_{R,x}$ das etapas de filtro Gaussiano (passa-baixa) e filtro de detalhes (passa-alta) que são as 2 primeiras etapas do algoritmo clássico, aplicadas no componente vermelho (*red*) da imagem $I_R$ . A notação está ligeiramente alterada para representar a prática mais comum:

$$F_{R,x} = \left[I_R * K_{\sigma 0}\left(x\right) * K_{\sigma 0}\left(y\right)\right] * \frac{\partial K_{\sigma 0}\left(x\right)}{\partial x} ,$$

$$F_{R,y} = \left[I_R * K_{\sigma 0}\left(x\right) * K_{\sigma 0}\left(y\right)\right] * \frac{\partial K_{\sigma 0}\left(y\right)}{\partial y} .$$

É válido esclarecer alguns pontos da equação acima. A primeira etapa do detector Canny é feito com um filtro Gaussiano em duas etapas lineares (em oposição a um filtro bidimensional) para ganhos de desempenho computacional (FISHER; PRICE, 1996). Na sequência (segunda etapa do algoritmo) faz-se uso da derivada da Gaussiana como detector de borda, em contraste com diversas outras implementações do mesmo algoritmo, que empregam um filtro Sobel ou outro equivalente. As demais etapas do detector são aplicadas da forma convencional.

Em seguida, em cada ponto da borda detectada é feito o cálculo do desvio padrão, empregando a seguinte equação:

$$\sigma\left(p\left(x_0, y_0\right)\right) = \sqrt{\frac{\sum_{(x,y)\in p(x_0,y_0)} \left\|G\left(x,y\right)\right\| .l^2\left(x,y\right)}{\sum_{(x,y)\in p(x_0,y_0)} \left\|G\left(x,y\right)\right\|}} .$$

O conceito empregado ao aplicar tal equação é que, após realizado o detector de Canny e convertida a imagem resultante para valores absolutos, formam-se "cristas" nas bordas originais, sendo que o perfil (corte da secção) ao longo da direção de gradiente máximo forma uma curva Gaussiana, e denominada por $p\left(x_0, y_0\right)$. O gradiente máximo de cada ponto é aquele de maior transição da borda original, e normalmente será ortogonal à borda. Para cada ponto $(x_0, y_0)$ das bordas, o perfil $p\left(x_0, y_0\right)$ é percorrido através dos pontos $(x, y)$ somando as magnitudes $G\left(x, y\right)$ ponderadas pela distância ao quadrado entre o ponto $(x, y)$ e o ponto $(x_0, y_0)$. Uma notação mais clara poderia ser empregada para expressar a curva de gradiente, como exemplo: $p\left(x, y\right)|_{(x_0,y_0)}$. Essa somatória é então dividida pela integral da curva (denominador da expressão) para normalização. O cálculo acima é condizente com a expressão do desvio padrão geral de uma variável continua real $x$ dada por uma função de distribuição de probabilidade $f\left(x\right)$ qualquer:

$$\sigma = \sqrt{\int_x \left(x-\mu\right)^2 .f\left(x\right) .dx} .$$

Na prática, no entanto, os cálculos de largura de borda e do desvio padrão são feitos ao longo dos eixos $x$ e $y$ da imagem. Isso visa redução de cálculos e ganho de desempenho computacional. Normalmente, seria necessário que os cálculos fossem feitos no ângulo $\theta$ de variação máxima.

No cálculo do desvio padrão, pixels que contenham um valor abaixo de determinado limite (10%) ou falsos picos duplos (ou múltiplos), que não passem de oscilações (limiar de 0,05) são desconsiderados. Por fim, os valores dos desvios padrões são recalculados em média, a fim de minimizar oscilações entre pontos vizinhos.

Paralelamente ao processo descrito anteriormente, é feita a segmentação da imagem no espaço de textura. As técnicas empregadas são clusterização por *mean shift* (FUKUNAGA; HOSTETLER, 1975) e o espaço de textura é analisado via o algoritmo de energia de textura de Laws (LAWS, 1980). A análise toda é feita em cada espaço de cor RGB separadamente. A proposta é obter como resultado uma segmentação excessiva, de modo a garantir que não haja união entre áreas sem relação.

As regiões obtidas pela segmentação são então rotuladas por "*sharp*" ou "*blurred*", conforme a existência de uma borda com tal característica ("*sharp*" ou "*blurred*" no seu interior), além de outras características estatísticas analisadas (centro de massa, tamanho, características da textura e etc.).

Os autores fazem uma comparação direta com outros métodos divulgados na literatura científica (ACHANTA; SUSSTRUNK, 2010; ACHANTA et al., 2009; GRAF; KRIEGEL; WEILER, 2011; KIM, 2005; LI; NGAN, 2007; ZHANG et al., 2006) que objetivam segmentar objetos salientes em imagens. Essa busca nem sempre se trata de localização de foco ou não foco em imagens específicas com baixa profundidade de campo, mas podendo ser por uma questão de cores, luminância ou contrastes, por exemplo (ACHANTA; SUSSTRUNK, 2010; ACHANTA et al., 2009). A comparação entre tais algoritmos na segmentação de imagens por saliência visual ou para baixo DOF foi feita empregando 112 imagens, otimizando os algoritmos, mas sem intervir nas imagens em si. Os resultados para cada método foram calculados em média, usando como parâmetros de medição *recall* (POWERS, 2011), precisão (*precision*) (POWERS, 2011) e medida-F com $\beta = 1$, $F_1$, o que resulta na seguinte fórmula, conforme (GUILLET; HAMILTON, 2007):

$$F_1 = 2.\frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} .$$

Os resultados dos autores de fato mostram desempenhos superiores de $F_1$ aos demais algoritmos e métodos comparados, por ser o único a atingir um valor superior a 80% de medida-F, assim como para *recall* e precisão simultaneamente.

### 3.2.6 Estatística de ordem superior (HOS)

Sendo uma das publicações mais recentes na área, o artigo de Ahn e Chong (AHN; CHONG, 2015) emprega o princípio de segmentação da região em foco via transformada de

características junto do principio de conteúdo, sendo a técnica centrada no uso de estatística de ordem superior, denominando-a de *adaptative-second-order-statistics* (ASOS). Segundo os próprios autores, a proposta possui vantagens quando aplicada a imagens com ruido, algo que pode ser um empecilho para outros métodos.

Dada uma imagem com baixa profundidade de campo, os autores separam-na em 3 componentes de cor. Para cada um deles são calculados 3 conjuntos de pesos: peso do tom $W_{tonal}$, peso da distância $W_{spatial}$ e peso da derivada segunda $W_{SOD}$. O primeiro deles é calculado entre um ponto $(s,t)$ da imagem $I$ para um outro ponto qualquer $(x,y)$:

$$W_{tonal}\left(s,t\right) = \frac{1}{\sqrt{2.\pi.\sigma_t^2}}.\exp\left[-\frac{1}{2}\left(\frac{\left|I\left(s,t\right)-I\left(x,y\right)\right|}{\sigma_t}\right)^2\right] . \tag{31}$$

O cálculo do peso, como pode ser visto, é feito por uma função gaussiana, regida por um parâmetro de espalhamento $\sigma_t$ e pela diferença de intensidade (de modo absoluto) dos pixels em $(s,t)$ e $(x,y)$. Esse cálculo tem por base o conceito de que pontos com cores próximas possuem uma maior probabilidade (maior peso) de pertencerem ao mesmo objeto representado na região. Como o cálculo não restringe a localização dos pontos em analise, locais distantes e desconexos dentro da imagem, mas que possuem cores similares, terão um grande peso. Por isso é necessário considerar o peso da distância entre pontos também, que é calculado de forma totalmente análoga, dado por:

$$W_{spatial}\left(s,t\right) = \frac{1}{\sqrt{2.\pi.\sigma_s^2}}.\exp\left[-\frac{1}{2}\left(\frac{\sqrt{\left(s-x\right)^2+\left(t-y\right)^2}}{\sigma_s}\right)^2\right] . \tag{32}$$

onde a gaussiana tem o parâmetro $\sigma_s$ de espalhamento e, como seu segundo fator de controle, a distância euclidiana entre os pontos $(s,t)$ e $(x,y)$ na imagem. Nota-se que tanto a função $W_{tonal}$ quanto $W_{spatial}$ possuem como resultados um valor escalar.

O terceiro peso é dado por:

$$W_{SOD}\left(x,y\right) = \frac{\sqrt{\left(\frac{d^2I(x,y)}{dx^2}\right)^2+\left(\frac{d^2I(x,y)}{dy^2}\right)^2}}{\max\left(\sqrt{\left(\frac{d^2I}{dx^2}\right)^2+\left(\frac{d^2I}{dy^2}\right)^2}\right)} . \tag{33}$$

onde a base é a derivada de segunda ordem (*second order derivative*, ou SOD), e, portanto, recebendo o nome de $W_{SOD}$ .

Nota-se uma diferença na forma do cálculo dos pesos para a derivada segunda. No numerador há o cálculo da derivada segunda em cada ponto da imagem, enquanto no denominador há o valor máximo para toda a imagem. Como o cálculo é feito por ponto, as operações de potenciação e radiciação são aplicadas ponto a ponto, resultando numa matriz de mesmo tamanho da imagem. O objetivo desse cálculo é a remoção de regiões sem foco e também de ruido da imagem. A operação de derivada é feita aplicando a convolução das matrizes:

$$d_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} ,$$

$$d_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} .$$

Como os operadores são característicos de derivada de primeira ordem, os autores aplicam a convolução duas vezes para calcular a derivada segunda.

O passo seguinte é o cálculo do segundo momento (HOS) para cada camada de cor, conforme abaixo para o caso da cor verde $G$ (da inicial em inglês):

$$\hat{m}_G^{(2)}(x,y) = \frac{1}{N_\eta}. \sum_{(s,t)\in\eta(x,y)} \left(I_G(s,t) - m_G(x,y)\right)^2 , \tag{34}$$

onde $\eta(x,y)$ define um conjunto na vizinhança em torno do centro $(x,y)$, sendo que $(s,t)$, como indicado na equação, deve pertencer a tal vizinhança. $N_\eta$ é o número de pixels dentro da vizinhança definida e $m_G(x,y)$ a média amostral, que por sua vez é dada pela equação

$$m_G(x,y) = \frac{1}{N_n}. \sum_{(s,t)\in\eta(x,y)} I_G(s,t) , \tag{35}$$

na qual $N_n$ é a contagem de pixels dentro da região $\eta$. Na Equação (35), o valor de $m_G$ pode ser entendido como o valor médio dos pixels contidos na vizinhança $\eta$ centrada em $(x,y)$ e logicamente a Equação (34) faz o cálculo da variância para cada ponto da imagem em torno da região $\eta$.

Os resultados apresentados anteriormente na Equação (34) são reescritos considerando os pesos calculados nas Equações (31), (32) e (33) conforme uma proposta dos autores, assim, novamente considerando apenas uma das camadas de cor, então:

$$s_G(x,y) = W_{SOD}(x,y) .\frac{1}{N_\eta^3}. \sum_{(s,t)\in\eta(x,y)} W_{spatial}(s,t) .W_{tonal}. \left(I_G(x,y) - I_G(s,t)\right)^2 , \tag{36}$$

onde tem-se os pesos $W$ que são os mesmos definidos anteriormente. Como os valores devem ser limitados entre 0 e 255 e alguns dos resultados da Equação (36) podem atingir valores muito mais altos, então aplica-se um limite de 255 como máximo, gerando a seguinte forma de cálculo:

$$ASOS(x,y) = \min\left\{255, \frac{s_G(x,y)}{c}, \frac{s_R(x,y)}{c}, \frac{s_B(x,y)}{c}\right\} , \tag{37}$$

em que $ASOS(x,y)$ é o resultado final para o método *adaptative-second-order-statistics* para um ponto $(x,y)$, e $c$ é um fator de escala. Para uma série de medições, os autores alegam que o valor mais adequado de $c$ é igual a 150. Os resultado do $ASOS$ é tanto maior (tendendo a 255)

quando há informações de alta frequência e tende a 0 no caso oposto. Similar a muitos outros métodos, algumas pequenas regiões, normalmente no interior das partes de interesse, podem não ser detectadas, e para isso são feitas operações morfológicas similares às aplicadas em (YE; LU, 2002) e (WON; PYUN; GRAY, 2002).

Os autores realizaram testes com os bancos de imagens disponíveis em (KIM et al., 2007), entre outros. Para a vizinhança, foi definida a matriz quadrada de tamanho $3 \times 3$ e os valores de $\sigma_t$ e $\sigma_s$ foram definidos, respectivamente, como 1,5 e 1,4.

A métrica de avaliação do desempenho do método é o *Structural Similarity Index* (SSIM). Num primeiro momento são feitas comparações com o método HOS apresentados em outro artigo, ambos frente ao incremento do ruido, e avaliando quanto o resultado se degrada com a maior presença de ruido em cena, o que é completamente esperado. Numa etapa seguinte são feitas avaliações dos parâmetros do algoritmo, mas que não são apresentadas. Por fim o novo método é comparado com outros algoritmos que possuem objetivo comum ou mesmo com métodos de segmentação gerais, e mostram que via medida $F$ que a nova proposta se sai melhor do que os demais.

Infelizmente no artigo não são divulgados os valores dos tempos de execução necessários.

## 3.3 MÉTRICAS DE AVALIAÇÃO E *GROUND TRUTH*

Ao tratar da comparação de algoritmos é essencial que num primeiro momento sejam definidos os critérios de avaliação quantitativos, tanto de desempenho quanto de assertividade. Apesar de uma comparação visual ser válida num primeiro momento, há uma subjetividade inerente ao processo, podendo portanto ser uma classificação apenas qualitativa e não suficiente. O tempo de execução é uma medida direta do desempenho, e dada uma mesma imagem de análise que será submetida a todos os métodos, o tempo absoluto de processamento, dado que os parâmetros de execução (capacidade computacional) sejam constantes, é um critério de medição.

A qualidade de resultados da execução de um algoritmo exige, por outro lado, critérios mais complexos de avaliação. Algo importante nessa função é que uma imagem de referência esteja presente para cada imagem de teste. Essa referência, normalmente denominada de *ground truth*, é o resultado da segmentação realizada por seres humanos. É a imagem que espera-se que o algoritmo obtenha ao final do processamento, portanto, será composta de uma matriz de mesma dimensão da imagem original, e limitada a valores 0 ou 1, que sinalizam respectivamente a falta de foco ou a presença dele. Assim, sendo uma imagem de entrada $I_E$ digital de 24-*bits*[2] de dimensões $M \times N$ pixels, definida no espaço de cores RGB (cartesiano), terá representação de cada vetor de cor conforme abaixo:

$$I_E = \left( I_E^{(R)}, I_E^{(G)}, I_E^{(B)} \right) ,$$

[2] A forma mais usual de definição de imagens de 24-*bits* é num espaço tridimensional, dado pelas cores vermelha, verde e azul, onde cada dimensão possui 8 bits.

$$I_E^{(R)} = \begin{pmatrix} i_{11}^{(R)} & i_{12}^{(R)} & i_{13}^{(R)} & \cdots & i_{1N}^{(R)} \\ i_{21}^{(R)} & i_{22}^{(R)} & i_{23}^{(R)} & \cdots & i_{2N}^{(R)} \\ i_{31}^{(R)} & i_{32}^{(R)} & i_{33}^{(R)} & \cdots & i_{3N}^{(R)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i_{M1}^{(R)} & i_{M2}^{(R)} & i_{M3}^{(R)} & \cdots & i_{MN}^{(R)} \end{pmatrix} \in \mathbb{N}^{M\times N} \mid 0 \leq i_{xy} \leq 255\,,$$

$$I_E^{(G)} = \begin{pmatrix} i_{11}^{(G)} & i_{12}^{(G)} & i_{13}^{(G)} & \cdots & i_{1N}^{(G)} \\ i_{21}^{(G)} & i_{22}^{(G)} & i_{23}^{(G)} & \cdots & i_{2N}^{(G)} \\ i_{31}^{(G)} & i_{32}^{(G)} & i_{33}^{(G)} & \cdots & i_{3N}^{(G)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i_{M1}^{(G)} & i_{M2}^{(G)} & i_{M3}^{(G)} & \cdots & i_{MN}^{(G)} \end{pmatrix} \in \mathbb{N}^{M\times N} \mid 0 \leq i_{xy} \leq 255\,,$$

$$I_E^{(B)} = \begin{pmatrix} i_{11}^{(B)} & i_{12}^{(B)} & i_{13}^{(B)} & \cdots & i_{1N}^{(B)} \\ i_{21}^{(B)} & i_{22}^{(B)} & i_{23}^{(B)} & \cdots & i_{2N}^{(B)} \\ i_{31}^{(B)} & i_{32}^{(B)} & i_{33}^{(B)} & \cdots & i_{3N}^{(B)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i_{M1}^{(B)} & i_{M2}^{(B)} & i_{M3}^{(B)} & \cdots & i_{MN}^{(B)} \end{pmatrix} \in \mathbb{N}^{M\times N} \mid 0 \leq i_{xy} \leq 255\,.$$

Essa representação varia ligeiramente para outros espaços de definições de imagens, como o CIELAB ou o HSV, por exemplo. Para a imagem de referência teremos:

$$I_{GT} = \begin{pmatrix} i_{11} & i_{12} & i_{13} & \cdots & i_{1N} \\ i_{21} & i_{22} & i_{23} & \cdots & i_{2N} \\ i_{31} & i_{32} & i_{33} & \cdots & i_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i_{M1} & i_{M2} & i_{M3} & \cdots & i_{MN} \end{pmatrix} \in \mathbb{N}^{M\times N} \mid 0 \leq i_{xy} \leq 1\,.$$

A título de exemplo, a Figura 15 mostra o *ground truth* da segmentação por foco para uma dada imagem. Uma forma visual de avaliação do resultado da segmentação (tanto do algoritmo quanto do *ground truth*) é a aplicação do mesmo sobre a imagem original, o que também é representado na Figura 15. Essa operação se dá pelo produto de Hadamard[3] para cada vetor, então sendo $I_B$ uma imagem binária da segmentação de $I_E$, o resultado da aplicação da máscara é $I_M$ :

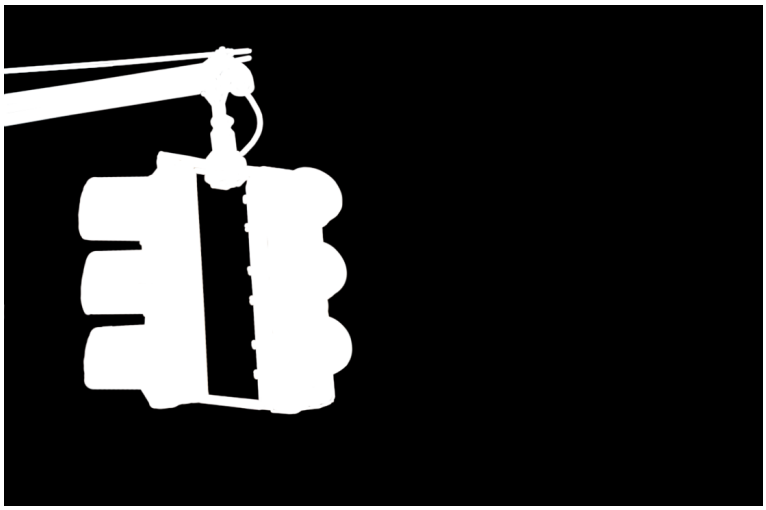
$$I_M = \left(I_E^{(R)} \circ I_B, I_E^{(G)} \circ I_B, I_E^{(B)} \circ I_B\right)\,.$$

[3]O produto de Hadamard é a multiplicação elemento a elemento de duas matrizes de mesmo tamanho.

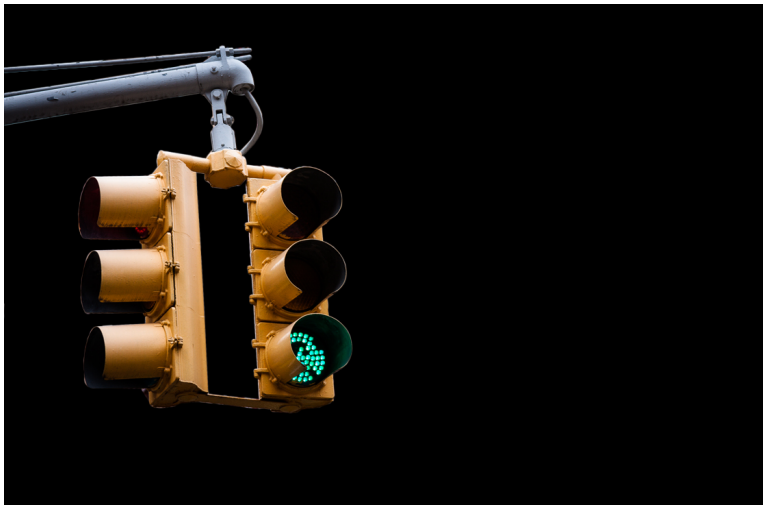
Figura 15 – Exemplo de uma imagem de entrada e da segmentação

(a) Exemplo de imagem de entrada.

(b) Exemplo de *ground truth*.



(c) Resultado da aplicação do *ground truth* binário sobre a imagem original.



Fonte: Autor, 2015

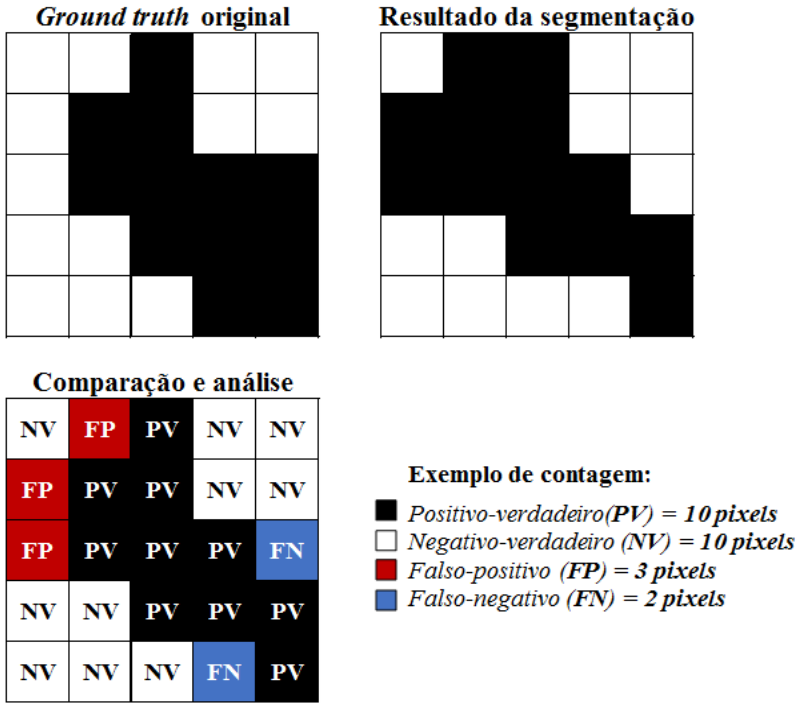
Notas: Exemplo de uma imagem de entrada (A), a segmentação da região em foco realizado por uma pessoa (B) e resultado da aplicação da segmentação em (B) na imagem original (C).

Como o objetivo do trabalho se trata de uma segmentação binária (dois níveis ou grupos apenas), a matriz resultante $I_B$ da segmentação pode ter cada um de seus elementos classificados dentro de 4 possibilidades apenas, conforme a matriz de referência $I_{GT}$:

- Positivo-verdadeiro: Quando o elemento $(m, n)$ de $I_B$ é igual ao de $I_{GT}$ e iguais a $1$;
- Negativo-verdadeiro: Quando o elemento $(m, n)$ de $I_B$ é igual ao de $I_{GT}$ e iguais a $0$;
- Falso-positivo: Quando o elemento $(m, n)$ de $I_B$ é igual a $1$ e o de $I_{GT}$ igual a $0$;
- Falso-negativo: Quando o elemento $(m, n)$ de $I_B$ é igual a $0$ e o de $I_{GT}$ igual a $1$.

Por ser de interesse que haja uma métrica para classificação geral do resultado de cada imagem, pode-se adotar alguns dos mecanismos já difundidos na literatura (OLSON; DELEN, 2008), e que são adequados a sistemas de classificação binários. Nesse caso, o cálculo se baseia exatamente na contabilização de pixels classificados em cada grupo. Alternativamente, seriam possíveis outros métodos, como a distância de Hausdorff (PUJOL; VILLANUEVA; ALBA, 2002) ou por descritores de borda (GONZALEZ; WOODS, 2002). Como o objeto desse trabalho é reduzir o volume de dados de imagem (remoção de pixels do *background*), espera-se que o resultado ótimo seja um algoritmo que delineie exatamente o objeto de interesse em cena. Sendo essa situação ideal não atingida, há duas possibilidades de erros do método: corte excessivo do objeto de interesse (falso-negativo) ou sobra de regiões sem interesse (falso-positivo). Assim, fica evidente uma assimetria nos tipos de erros pois a perda de informação (falso-negativo) é irrecuperável para o sistema, podendo gerar prejuízos à capacidade de reconhecimento/decisão. Por outro lado, falhas do tipo falso-positivo incorrem em diminuição da eficiência geral (capacidade de redução de pixels desnecessários), mas não na capacidade de reconhecimento/decisão, visto que o objeto de interesse dentro da segmentação está ainda integro. Para destacar essa classificação, a Figura 16 demonstra um exemplo de como a comparação acima é feita, além de quantizada.

Figura 16 – Exemplo de análise de resultados de segmentação




Fonte: Autor, 2015

Notas: Exemplo de análise do resultado de segmentação com a referência (*ground truth*) para um caso de 25 pixels no total (5 por 5 pixels). No exemplo, cada tipo de resultado, dentro dos 4 possíveis, são caracterizados por cores diferentes: falso-positivo em vermelho, falso-negativo em azul, positivo-verdadeiro em preto e negativo-verdadeiro em branco.

### 3.3.1 Quantização de valores

Com base nas condições e critérios descritos previamente, há dois valores numéricos como figuras de méritos:

- Acurácia $A$;
- Proporção de falsos negativos ($PFN$).

A primeira medida descreve numericamente a quantidade de pixels certos dentro da imagem quando comparado com o *ground truth*. Adotando as nomenclaturas:

- $Q_{FP}$ como a quantidade (soma) de pixels classificados como falsos-positivos;
- $Q_{FN}$ como a quantidade (soma) de pixels classificados como falsos-negativos;
- $Q_{PV}$ como a quantidade (soma) de pixels classificados como positivos-verdadeiros;
- $Q_{NV}$ como a quantidade (soma) de pixels classificados como negativos-verdadeiros.

Neste contexto, torna-se evidente que todos os pixels da imagem devem se enquadrar necessariamente em alguma das categorias acima. Assim, para uma imagem $M \times N$, tem-se:

$$Q_{FP} + Q_{FN} + Q_{PV} + Q_{NV} = M.N\,.$$

A acurácia $A$ será então:

$$A = \frac{Q_{PV} + Q_{NV}}{M.N}\,. \tag{38}$$

O valor $A$ estará limitado entre 0 e 1 (ou 100%, caso empregado um padrão porcentual). Ainda, quanto maior o número de acertos, maior será o valor de $A$, o que conclui-se que na comparação entre resultados de dois algoritmos, aquele que apresentar um resultado de $A$ maior será considerado como de melhor acurácia.

Somente a acurácia não é suficiente para classificação dos resultados. Como há uma assimetria dos resultados dos erros, como descrito anteriormente, é necessário que haja também outra figura de mérito de forma a contabilizar isso. Uma das formas possíveis é através da proporcionalidade de falsos-negativos frente à quantidade total de erros. Assim:

$$PFN = \frac{Q_{FN}}{Q_{FN} + Q_{FP}}\,. \tag{39}$$

Analogamente a $A$, os valores de $PFN$ estão também limitados entre 0 e 1, ou 100%. Esse valor de $PFN$ será melhor quanto menor, implicando que dentre os erros cometidos na classificação, houve uma taxa maior de falsos-positivos.

Além das medições apresentadas acima, existem outras possíveis. Duas delas são usadas por alguns autores (ACHANTA; SUSSTRUNK, 2010; NEVEROVA; KONIK, 2012), e valem sua explicitação. Trata-se de *recall* $R$ e precisão $P$ (*precision*) (OLSON; DELEN, 2008)

$$R = \frac{Q_{PV}}{Q_{PV} + Q_{FP}}\,, \tag{40}$$

$$P = \frac{Q_{PV}}{Q_{PV} + Q_{NV}}\,. \tag{41}$$

Essas grandezas apresentam uma limitação no contexto de classificação da segmentação, visto que observam apenas os pixels positivos, podendo levar a falhas de avaliação, e por isso não serão usadas, sendo mais apropriadas em classificadores binários (POWERS, 2011). Um exemplo hipotético de mal funcionamento da grandeza é uma imagem que não possua objetos em foco. A segmentação correta seria 100% dos pixels classificados como negativos (verdadeiros). Um algoritmo que atinja tal resultado ideal, no entanto, será classificado com valores de precisão nula e *recall* inválido pela divisão por zero. Opostamente, a acurácia desse caso é 100%, como esperado.

Num sistema de classificação binária, é possível a aplicação da análise via curva ROC (acrônimo do inglês *receiver operating characteristic*, ou característica de operação do receptor). Seu uso é muito comum por ser aceito largamente como o método mais completo de

classificação de resultados (METTER; BEUTEL; KUNDEL, 2009). Nele é feito uma avaliação do valor da sensibilidade pelo *fall-out* de um algoritmo, em função de um parâmetro interno do mesmo, como exemplo, um limiar de classificação. Sensibilidade e *fall-out* são igualmente denominados de razão de positivos-verdadeiros $TPR$ e razão de falsos-positivos $FPR$, respectivamente. Expressando matematicamente, tem-se:

$$TPR = \frac{Q_{PV}}{Q_{PV} + Q_{FN}} ,$$

$$FPR = \frac{Q_{FP}}{Q_{FP} + Q_{NV}} .$$

Os resultados de $TPR$ e $FPR$ são plotados num gráfico bidimensional. Por não haver uma especificidade das mesmas disponíveis (cada uma possui um conteúdo aleatório), aplicar a análise ROC dentro de cada imagem para otimizar os parâmetros do algoritmo deverá resultar em diferentes valores para cada uma (posição do pico de máximo), o que não permitirá conclusões acerca de um parâmetro ótimo. A junção de resultados de várias curvas dessas, igualmente sofrerá pela falta de especificidade do conjunto, resultando numa dispersão grande dos dados.

### 3.3.2 Tempo de execução

Como já descrito anteriormente, a medição do tempo de execução total do algoritmo é um dos critérios de avaliação do mesmo. Como o ambiente de execução dos algoritmos foi o software Matlab (versão R2013a e R2015a), o próprio software foi empregado como mecanismo de medição através das rotinas *tic* e *toc*. O primeiro comando registra a hora interna do computador quando executado, ao passo que o segundo realiza a atividade semelhante, e fornecendo como saída a diferença de tempo entre ambos (*tic-toc*). Apesar dessa forma de medição ser impactada pela capacidade de processamento do computador que a executa, é ainda o método mais recomendado (KNAPP-CORDES; MCKEEMAN, 2011).

## 3.4 EXPERIMENTOS

As imagens de testes foram separadas em 2 tipos e seus resultados avaliados separadamente por grupos:

a) Bancos de imagens gerais e públicas;

b) Imagens controladas.

### 3.4.1 Imagens gerais e públicas

São imagens de tamanho (dimensão $M$ e $N$) e conteúdo variados, mas com o ponto em comum de todas terem sido feitas com baixa profundidade de campo, e assim, um objeto em cena está em foco, ao passo que os demais não. Para cada imagem há também uma imagem paralela de referência (*ground truth*), com os padrões já descritos anteriormente.

O banco de imagens totaliza 269 imagens, sendo que 63 são oriundas do banco de imagens do trabalho (GRAF; KRIEGEL; WEILER, 2011) e as demais 206 imagens são oriundas da referência (LI; NGAN, 2011). O intuito desse banco é poder fazer uma avaliação estatística dos resultados. Com o alto número de imagens, existem possibilidades de avaliação de algoritmos para diversas situações possíveis. Alguns exemplos de imagens desse conjunto estão exibidas na Figura 17, juntamente do seu respectivo *ground truth*. Ainda assim, o banco de imagens não está livre de viés pois todos seus elementos foram feitos por fotógrafos, assim há uma preocupação estética comum em cada imagem, o que restringe seu poder de abrangência, de tal forma que há:

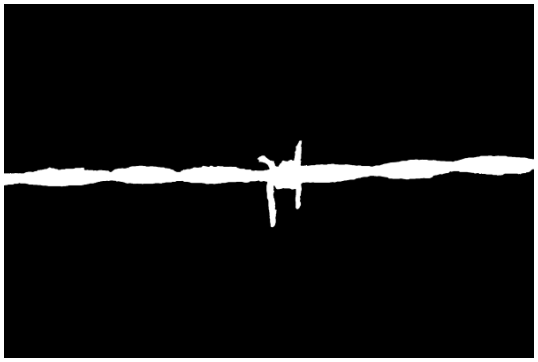
- Tendência nas fotos a terem seus objetos focados no centro ou dentro do $1/3$ central da imagem, conforme os padrões estáticos da fotografia (WOLFE; SHEPPARD, 2013). Essa limitação na composição das fotos faz com que não haja testes dos algoritmos para casos de imagens feitas autonomamente, sem intervenção humana. Nesse segundo caso de imagem, espera-se uma maior aleatoriedade da composição, pois não existem critérios a priori (salvo algumas aplicações) para sua formação;

- Alto grau de desfoque e de uniformidade do mesmo, conforme também padrões estáticos da fotografia (BATT; DOBRO; STEEN, 2014). É esperado que um alto grau de desfoque no *background* frente a um *foreground* em foco será mais fácil de avaliar do que casos em que essa diferença não é tão grande. Além disso, fotos feitas sem o mesmo critério artístico poderão apresentar desfoque em partes do *foreground*.

Figura 17 – Exemplo de elementos do banco de imagens

(a) Exemplo 1.

(b) Referência do Exemplo 1.



(c) Exemplo 2.

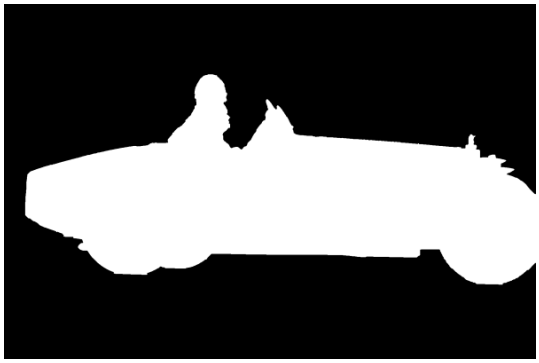
(d) Referência do Exemplo 2.

(e) Exemplo 3.

(f) Referência do Exemplo 3.

(g) Exemplo 4.

(h) Referência do Exemplo 4.



Fonte: Autor, 2015

Notas: Exemplo de elementos do banco de imagens, exibidas lado a lado com sua respectiva referência (*ground truth*).

### 3.4.2 Imagens controladas

São imagens feitas pelo autor com intuito de avaliar o comportamento da segmentação com o aumento ou diminuição do desfoque no *background*. Estão representadas na Figura 18. Para isso, seguem o critério de *ceteris paribus*[4], o que resulta em uma variação mínima entre imagens. Foram mantidos constantes nas imagens a posição dos objetos em cena e a posição da câmera e incidência luminosa na cena real. A variação de abertura do diafragma foi a única variação permitida deliberadamente. Consequentemente, para que a foto registrada tenha mesma intensidade luminosa (exposição), obrigatoriamente o tempo de exposição deve ser ajustado de acordo. Isso foi feito de maneira automática pela própria câmera, usando medição de luminosidade interna da câmera.

[4]Termo em latim que significa “todo os demais são constantes”, usual em economia, mas aplicável em diversos campos de estudo.

Figura 18 – Imagens controladas para avaliação

(a) Abertura $f/1.8$.

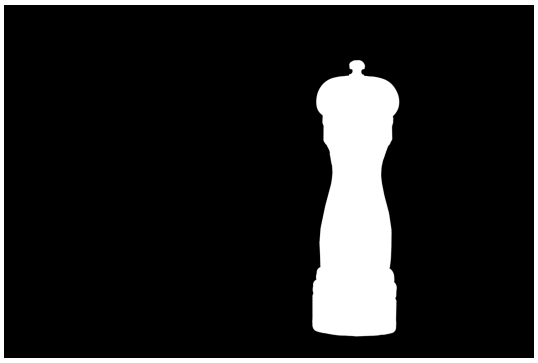
(b) Abertura $f/2.5$.

(c) Abertura $f/5$.

(d) Abertura $f/10$.

(e) Abertura $f/22$.

(f) *Ground truth*.



Fonte: Autor, 2015

Notas: Imagens controladas para avaliação, sendo o conjunto composto de 5 imagens com diferentes valores de profundidade de campo, e demais parâmetros constantes, além de uma imagem de *ground truth*, que é válida para todas.

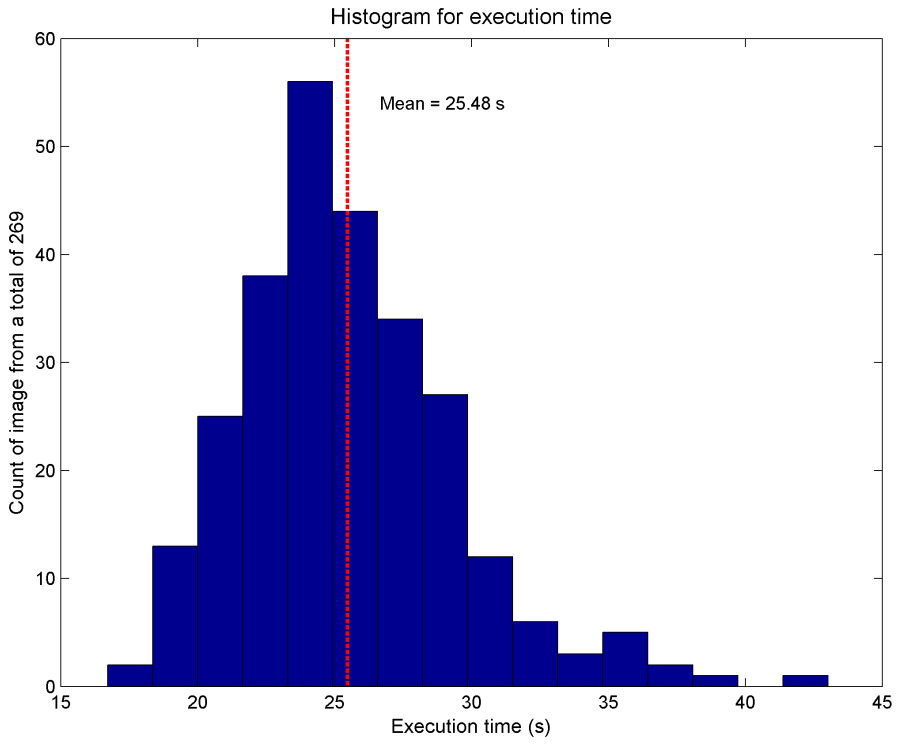
## 3.5 RESULTADOS

Conforme os critérios já apresentados, avaliou-se o resultado da execução do algoritmo NK (apresentado e analisado já em 3.2.5) para o banco de imagens e para as imagens controladas, medindo as três figuras de mérito disponíveis (tempo de execução, acurácia e *PFN*). Os resultados estão representados conforme a seguir. O código do algoritmo foi fornecido pelos próprios autores do mesmo (NEVEROVA; KONIK, 2012), e foram executados com sua calibração padrão.

### 3.5.1 Tempo de execução

O tempo de execução[5] medido para as 269 imagens foi representado via histograma, exibido na Figura 19. Os valores limites são 16,71 e 43,02 segundos, com média igual a 25,48 segundos. É nítido pelo histograma dois pontos críticos: a grande distribuição de resultados e um valor médio elevado. O primeiro tem relação com o tamanho das imagens do banco não serem normalizadas (quantidade de pixels total por imagem diferentes para cada imagem), ainda assim, isso não explica toda a variação observada. Como também será visto nos casos críticos apresentados adiante, o algoritmo em si, por trabalhar numa varredura de contornos com imagens que apresentem muitas bordas (destacadas pelo detector Canny), terá uma maior quantidade de informação para processamento. Assim fica caracterizado que o conteúdo das imagens (assim como seu tamanho) influem diretamente no tempo de processamento.

Figura 19 – Histograma resultante do tempo de execução do método NK




Fonte: Autor, 2015

Notas: Histograma resultante pra a distribuição das medidas de tempo de execução para o método NK usando 269 imagens com baixa profundidade de campo. O valor médio está destacado no mesmo com a reta vermelha pontilhada.

### 3.5.2 Acurácia e PFN

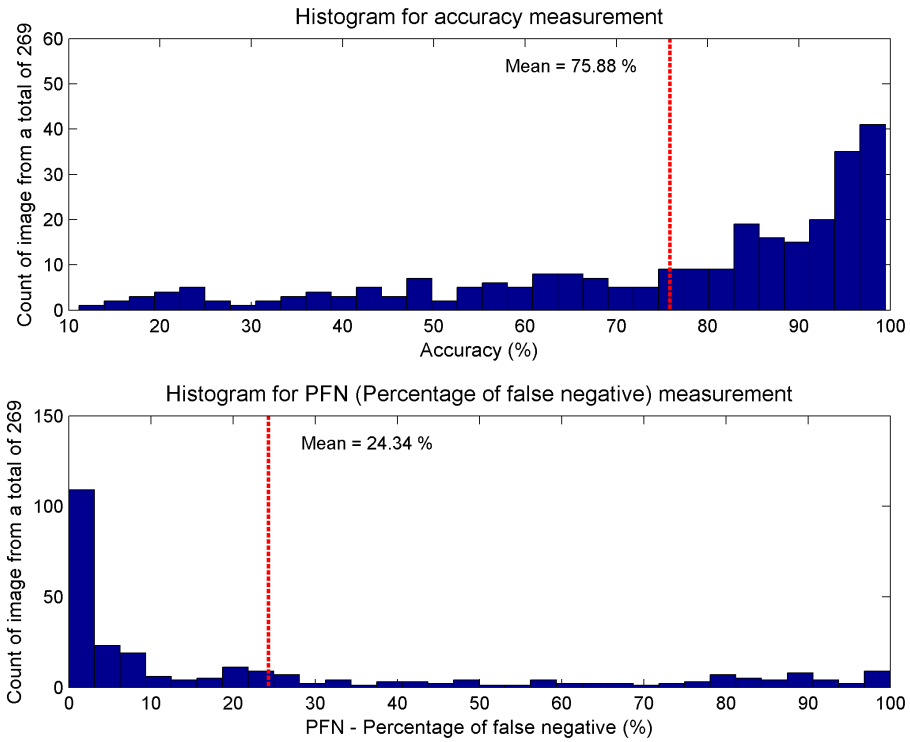
Para a acurácia e para o PFN foram medidos os resultados de cada uma das 269 imagens do banco. Tais resultados estão apresentados no formato de histogramas na Figura 20.

[5] Tempo medido empregando um computador PC com sistema operacional Windows 64 bits, processador Intel® Core™2 Quad Processor Q8200 e 8 gb de memória RAM DDR2. A versão de Matlab usada foi a R2013a.

Novamente há uma grande dispersão dos dados, especialmente na acurácia. Os valores médios obtidos, para acurácia e PFN são respectivamente 75,88% e 24,34%, que também estão sinalizados na Figura 20. Com base nas discussões já estabelecidas sobre perfil do banco de imagens disponível e empregado, o valor de acurácia obtido pode ser considerado baixo. Para qualquer aplicação, uma taxa de sucesso de apenas aproximadamente 75% é baixa. Quando se considera o fator de viés das imagens, o resultado apresentado é melhor do que haveria numa aplicação geral, o que torna o desempenho do método pior ainda numa análise mais vasta.

Para os valores de PFN, como também explicitados anteriormente, quanto menor esse valor, melhor o resultado. Nesse ponto apenas analisar o histograma e sua média indicam que os resultados do método são bons, por estarem bem abaixo dos 50% de valor médio.

Figura 20 – Histogramas resultantes para a acurácia e PFN para o método NK



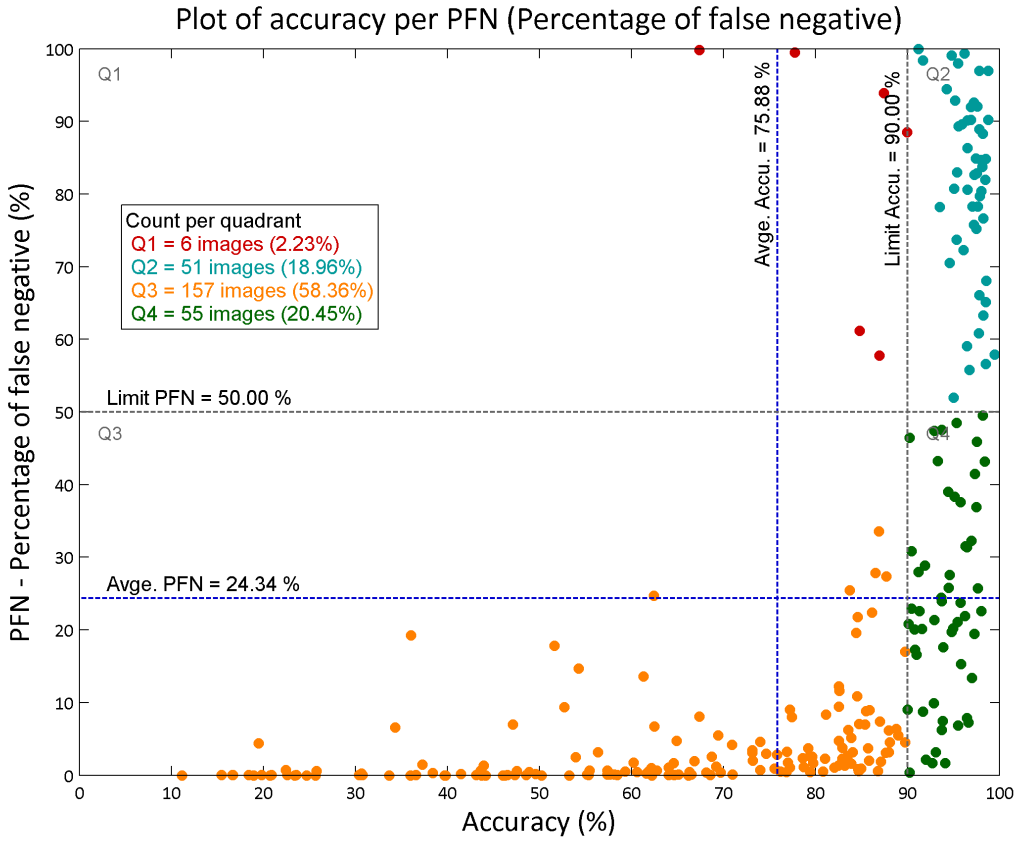

Fonte: Autor, 2015

Notas: Histogramas resultantes para a distribuição das medidas de acurácia e PFN para o método NK usando 269 imagens com baixa profundidade de campo. O valor médio de cada parâmetro está destacado sobre os mesmos com uma reta vermelha pontilhada.

A análise dos dados via histogramas independentes, no entanto, não é completa. Isso pela razão de situações de baixa acurácia com baixo PFN não serem tão ruins quanto resultados de baixa acurácia com alto PFN. Assim, uma segunda maneira de análise é através de uma nuvem de distribuição dos resultados por acurácia e PFN, conforme mostrado na Figura 21. Nessa segunda forma de representação, para cada ponto (imagem analisada) é possível ter os valores dos dois parâmetros medidos simultaneamente, perfazendo para o banco de imagens em estudo 269 pontos. Por essa análise, destaca-se que os pontos de acurácia menor do que o limite

de 80% estão majoritariamente abaixo do limite de 50% de PFN, ao passo que os pontos de acurácia elevada (acima de 80%) possuem o PFN distribuídos mais ou menos uniformemente em torno de 50%. Isso proporciona um resultado melhor do que o visto analisando apenas os histogramas.

Figura 21 – Distribuição em nuvem de acurácia e PFN para o método NK



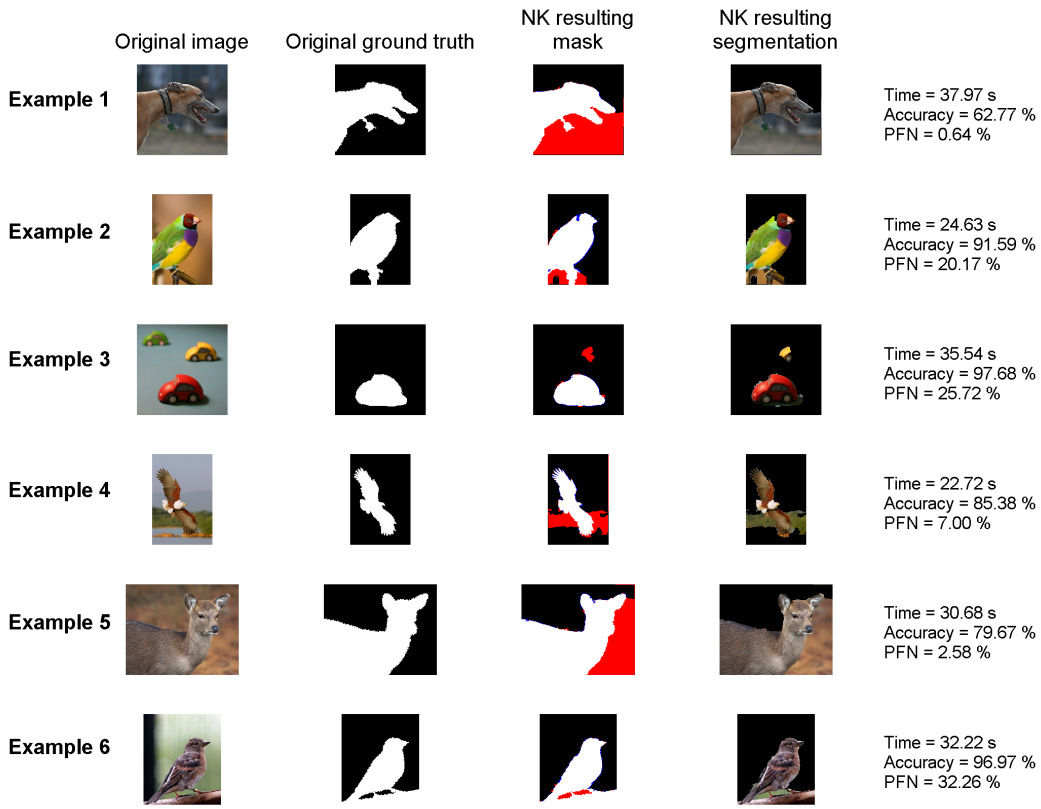

Fonte: Autor, 2015

Notas: Distribuição em nuvem dos valores de acurácia e PFN para o método NK usando 269 imagens com baixa profundidade de campo. O valor médio de cada parâmetro está destacado sobre o gráfico com linhas azuis pontilhada. Um valor de referência limite está destacado com linhas cinzas pontilhadas.

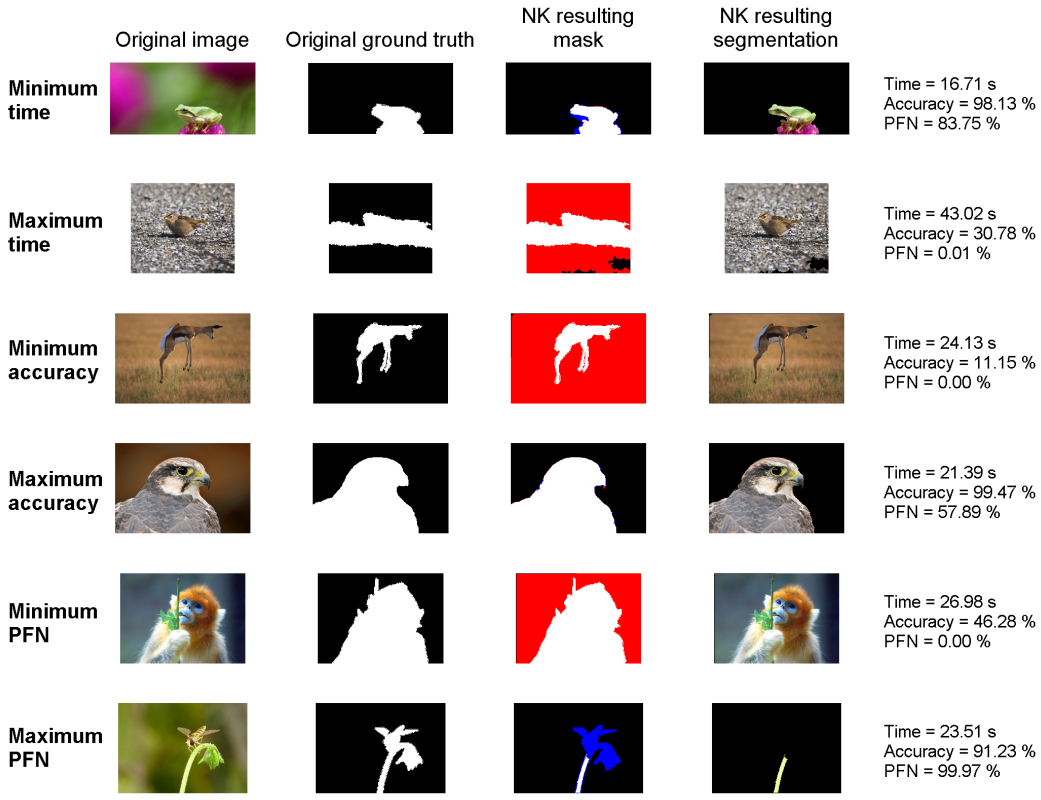
A Figura 22 ilustra exemplos de resultados de seis imagens aleatórias dentro do banco, e também exibe resultados de imagens que tiveram os melhores e piores desempenhos para cada um dos parâmetros avaliados (acurácia, PFN e tempo).

Figura 22 – Exemplo de resultados da execução do método NK

(a) Exemplo de resultados aleatórios.




(b) Destaques de melhores e piores desempenhos para cada grandeza avaliada).



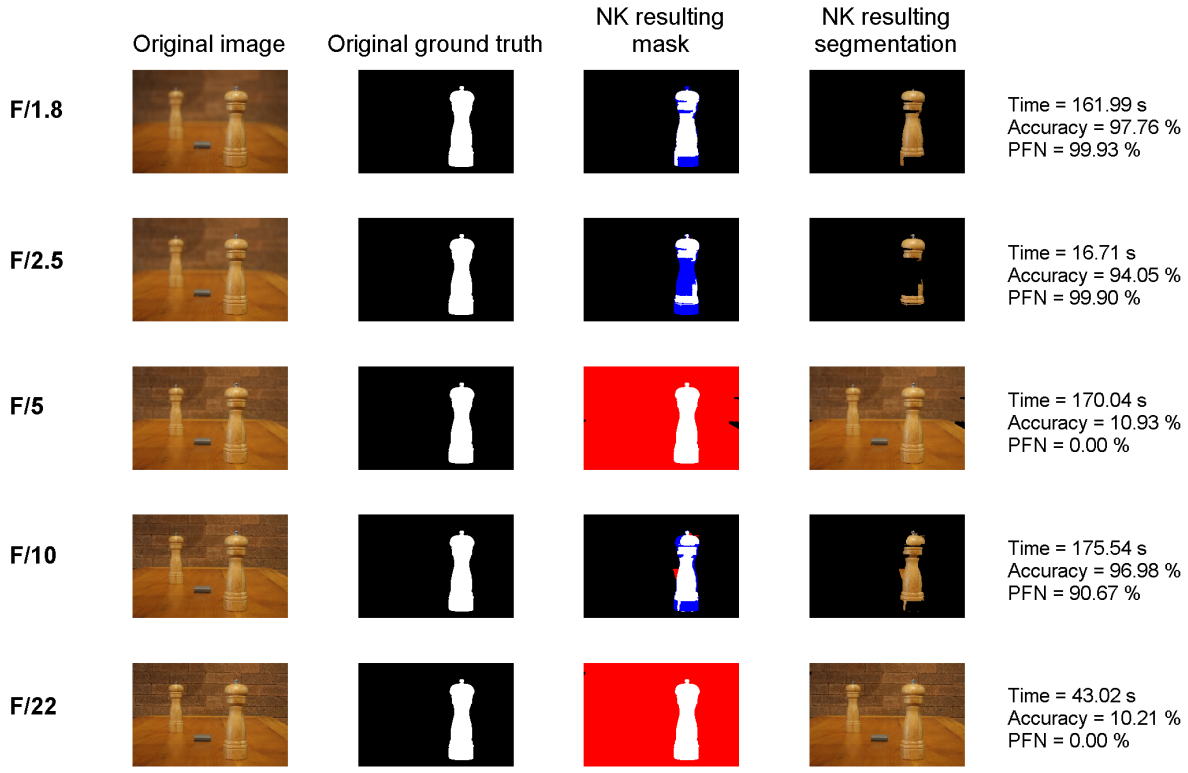

Fonte: Autor, 2015

Notas: Exemplo de resultados da execução do método NK para imagens presentes no banco com baixa profundidade de campo. Cada imagem possui seus resultados quantitativamente expressos ao seu lado.

### 3.5.3 Resultados com as imagens controladas

Além da aplicação do banco de imagens de alto volume, o método foi testado para um grupo de imagens menor, apresentado anteriormente na Figura 18. A razão principal desse teste é determinar uma configuração que torne o algoritmo mais ou menos eficiente, sendo que o fator avaliado é o aumento da relação entre foco e desfoque numa imagem, conforme elucidado já ao longo do trabalho e mais claramente demonstrado como exemplo na Figura 13. A submissão das imagens de teste ao algoritmo está apresentada na Figura 23. Nesse caso, diferente da situação anterior para o banco de imagens, valores estatísticos não são apresentados, visto que o tratamento de cada imagem é individualizado. O que pode ser observado é que o resultado do algoritmo oscila entre acurácias muito altas, na faixa de 94% a 98%, mas ainda assim, sendo resultado ruins, visto que para todos esses o PFN foi muito elevado (acima de 90%). Visualmente, é simples ver que os resultados são de baixa qualidade, pois é difícil mesmo para um ser humano identificar o resultado final do método. Complementando o total de imagens ainda, é visto que algumas das imagens tiveram resultados completamente opostos: baixa acurácia e baixo PFN. São igualmente ruins, pois apesar de preservarem o conteúdo da imagem para identificação dos objetos, não apresentam qualquer melhoria de desempenho no sistema (redução de quantidade de pixels úteis).

Figura 23 – Resultados da execução do método NK para imagens controladas




Fonte: Autor, 2015

Notas: Resultados da execução do método NK para imagens controladas com baixa profundidade de campo. Cada imagem possui seus resultados quantitativamente expressos ao seu lado.

## 3.6 NOVA METODOLOGIA PROPOSTA

Como visto, em virtude dos resultados da aplicação dos métodos disponíveis estarem aquém dos necessários para um caso de uso do algoritmo em tempo real, uma nova metodologia foi proposta a partir da observação de imagens de testes feitas para avaliação dos algoritmos, particularmente as fotos apresentadas na Figura 24.

As fotos representam dois casos opostos de profundidade de campo de uma mesma situação (cenário), dadas as demais condições constantes, ou seja, posição da câmera e dos objetos em cena, intensidade de luz e cores. Lado a lado, analisando com uma mera observação visual primeiramente o objeto em foco (*foreground*), nota-se que há extrema proximidade do conteúdo da informação em ambos os casos, ao passo que no restante da imagem (*background*), a discrepância entre foco e desfoque é grande. Assim, a proposta é um algoritmo que avalie as diferenças entre imagens com diferentes profundidades de campo, como as da Figura 24, no qual espera-se que o resultado dessa operação seja de alta semelhança para o *foreground* e o oposto para o *background*.

Figura 24 – Exemplo de imagens com variação apenas de profundidade de campo

(a) Imagem com menor profundidade de campo (Abertura $f/1.8$).

(b) Imagem com maior profundidade de campo (Abertura $f/22$).

Fonte: Autor, 2015

Notas: Exemplo de imagens com variação apenas de profundidade de campo. Todos os demais parâmetros foram ajustados de forma a proporcionar resultados de imagens semelhantes.

A ideia básica da abordagem proposta é realizar a busca da região em foco por dois princípios distintos e então combinar os resultados:

a) Análise da imagem através das regiões em que ocorre a alta frequência, e consequentemente uma possibilidade de haver foco;

b) Análise da semelhança entre partes da imagem, pois partes em foco deverão ser muito

parecidas entre si, ao passo que partes sem foco numa imagem, e com foco na outra, deverão apresentar uma diferença maior.

As etapas principais do algoritmo são descritas detalhadamente abaixo. As entradas do mesmo devem ser duas imagens de mesmo tamanho $M \times N$, em que os respectivos conteúdos sejam semelhantes, apenas com variação do DOF entre as imagens, conforme o exemplo da Figura 24:

a) Conversão de ambas as imagens para escala de cinza, de forma que cada imagem seja representada por apenas uma matriz. Originalmente, imagens coloridas são tratadas como compostas de 3 matrizes, uma para cada componente de cor (caso seja RGB). A conversão para cinza pode ser qualquer uma, mas foi adotado o uso da luminosidade apenas;

b) Alinhamento das imagens através de correção das matrizes. Após localizar a correlação máxima, então desloca-se uma das imagens para que fique semelhante à outra. Essa etapa é necessária para que as partes comuns entre ambas as imagens estejam o mais alinhada possível, visto que a imagem pode estar alguns pixels deslocadas da outra quando sobrepostas. Essa pequena diferença pode causar erros de análise e dos resultados. A operação visa apenas a operaçãod e translação, sem considerar eventuais rotações ou distorções esféricas que podem existir caso o aparato de captura não esteja nas suas condições ideais;

c) Normalização das imagens correlacionadas com o intuito de compensar divergência de luminosidade entre ambas. É especialmente necessária pois a diferença de abertura ótica entre as imagens pode causar um erro de luminosidade caso os medidores de cada câmera estejam descalibrados entre si, ou caso as imagens sejam feitas por uma única câmera com diferença temporal entre capturas (medições de luminosidade distintas). O processo de normalização da imagem $I\left(x,y\right)$ é feito via o cálculo a seguir, no qual $\min\left(I\left(x,y\right)\right)$ é a operação que determina o menor valor dentre todos os pixels de $I$, ao passo que a operação $\max\left(I\left(x,y\right)\right)$ determina o máximo:

$$I_{Norm}\left(x,y\right) = \frac{I\left(x,y\right) - \min\left(I\left(x,y\right)\right)}{\max\left(I\left(x,y\right)\right)}\,; \tag{42}$$

d) Cálculo do módulo do gradiente das imagens em termos de suas magnitudes apenas. O gradiente é uma medida que será maior para regiões de foco na imagem ao passo que diminui para as demais, pela razão das regiões focadas terem um maior grau de detalhes. Regiões sem foco são mais “lisas” ou “suaves”, tendo um gradiente nessas regiões menor. Existe uma limitação nessa questão pois regiões muitos lisas e em foco terão um comportamento de regiões sem foco. Claramente, a presença de altos picos é maior na região em que há foco, o que no fundo será tratado como um fator de detecção de foco por meio do conteúdo (presença de detalhes);

e) Aplicação de um filtro passa-baixa sobre a etapa anterior com o propósito de obter um valor médio por regiões e eliminar oscilações grandes na detecção. Uma vez que

o que a média da amplitude por região é o que interessa, um filtro passa-baixa é ideal para chegar a tal resultado. Do ponto de vista de processamento de sinais, o filtro passa-baixa aplicado como um núcleo (ou *kernel*) via convolução matemática tem o mesmo efeito da aplicação de uma média móvel ao longo da imagem, fazendo com que os picos sejam distribuídos e suavizados, ao passo que as regiões de constância sofrem pouco efeito, visto que os valores são próximos da própria média;

f) Cálculo da diferença em módulo entre ambas as imagens, pixel a pixel, subsequente normalização conforme Equação (42) e cálculo do complemento de um do resultado. Matematicamente, tem-se uma das matrizes menos a outra normalizadas e então o valor é subtraído de uma matriz de mesma dimensão de valores uns. O objetivo dessa etapa é que pontos ou regiões de grande semelhança entre as duas imagens tendam a 1, enquanto pontos discrepantes tendam a 0. Essa etapa somente é possível dada a presença de duas imagens, sendo uma forma de detecção por comparação de imagens;

g) Cálculo do produto de Hadamard entre a imagem resultante da diferença e o gradiente da imagem com baixo DOF e normalização conforme Equação (42). Essa etapa realiza a junção de dois resultados anteriores. Até então, cada uma das metologias usadas possui valores próximos à um para pontos em que há grande chance de serem pertencentes a regiões de foco, e valores tendendo a 0 para os casos contrários. A operação do produto, como sabido da matemática elementar, manterá um valor próximo à 1 apenas caso ambos os resultados anteriores concordem e sejam próximos à unidade. No demais casos o resultado tenderá a zero. Na prática, a operação, em termos lógicos ou booleanos, estendendo, para um domínio contínuo, é a operação lógica E;

h) Conversão da imagem resultante para uma máscara binária através do seu histograma pelo método de Otsu (OTSU, 1979). Esse método é indicado para casos em que há uma distribuição bimodal do histograma e se aplica de forma interessante ao caso, visto que o algoritmo, até então, tem justamente uma tendencia natural de ter valores próximos aos extremos 0 e 1;

i) Operações básicas de morfologia realizadas numa sequencia, conforme abaixo. São operações válidas pela forte tendência de acerto da área de interesse já nas etapas anteriores, servindo como um refinamento dos resultados, salientando que cada operação morfológica empregada é sempre limitada a pequenos ajustes.

   – Operação de fechamento (dilatação e erosão) para eliminação de pequenos buracos na região de foreground:

   – Operação de remoção de pequenas regiões detectadas (limitadas a 1

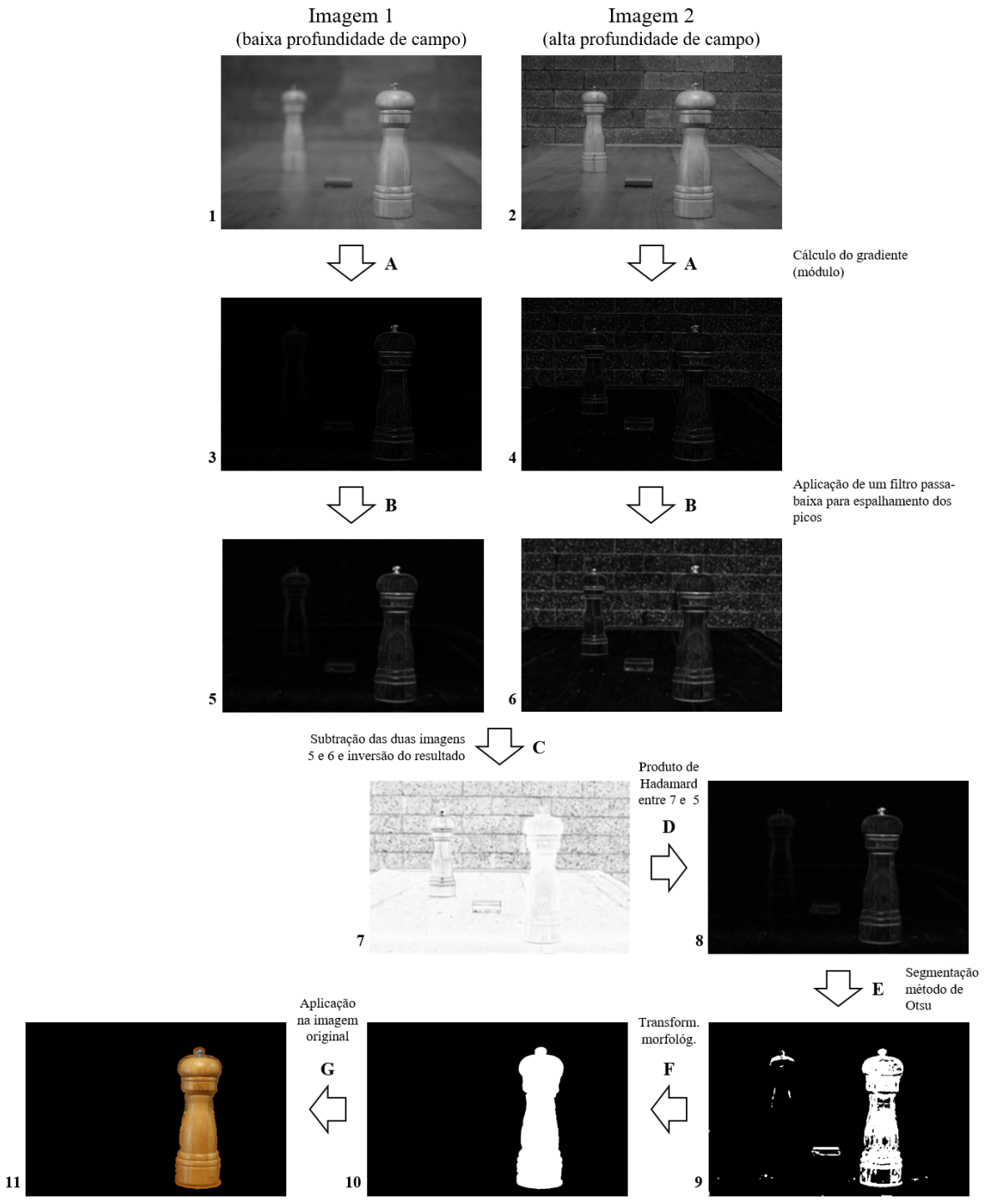
   – Operação de fechamento de pequenos buracos (limitados a 1

– Operação de dilatação para propiciar que haja um halo em torno da área segmentada e que englobe toda o objeto de interesse. Essa ação é baseada no fato de resultados que sejam falsos-positivos são menos prejudiciais que falsos-negativos, como já argumentado, e com tal processo, tende-se a garantir tal resultado.

O código para o algoritmo em Matlab está documentado no Apêndice A.

Uma visão geral dos resultados por etapa de operação do algoritmo está esquematizado na Figura 25, onde apresenta-se passo a passo para o caso onde uma imagem com abertura $f/1.8$ (baixa profundidade de campo) e uma imagem semelhante com abertura ótica $f/22$ (alta profundidade de campo). São as imagens apresentadas anteriormente na Figura 24.

Figura 25 – Representação do algoritmo proposto de forma esquematizada.




Fonte: Autor, 2015

Notas: Representação do algoritmo proposto de forma esquematizada e com exemplo por cada etapa de execução. As etapas após o cálculo do gradiente e após a aplicação do filtro passa-baixa (respectivamente, A e B) estão representados na imagem de forma normalizada pois sua escala não está no padrão de imagens (de 0 a 255).

### 3.6.1 Prova de conceito (sinais unidimensionais)

Uma outra perspectiva pode ser dada à avaliação e entendimento do algoritmo. Trata-se do uso de conceitos de processamento de sinais. Sob tal prisma, imagens digitais são sinais em duas dimensões, discretizados tanto no espaço quando nos valores de suas amostras.

É interessante destacar uma relação adotada entre as nomenclaturas estabelecidas entre as áreas de estudo. A menor unidade possível numa imagem, o pixel, para a área de processamento de sinais passa a ser chamada de amostra[6]. Assim, ambos os termos serão tratados como equivalentes.

Uma maneira de organizar uma matriz bidimensional $M \times N$ de amostras é através de um conjunto de $N$ vetores de dimensão $M \times 1$. Supondo que cada novo vetor não possui relação com os seus vizinhos nessa nova organização, as amostras em geral deixam de ter 8 vizinhos (desconsiderando bordas), como no caso original, para terem apenas 2 vizinhos (desconsiderando os casos nas bordas). Obviamente trata-se de uma perda de informação em tal sinal. Assim, no caso da proposta de segmentar a imagem, haverá uma consequente penalidade na capacidade dos métodos pois passarão a lidar com uma quantidade menor de informações. Obviamente essa perda poderia ser mitigada (mas não ao todo sanada) caso repetisse-se o processo nas demais direções e então congregando os resultados de maneira especifica. No entanto tal abordagem não nos interessa aqui.

O ponto de interesse na questão é que um método que tenha sucesso na segmentação das diversas secções unifilares da imagem, como descritas acima, sem recorrer às vizinhanças perdidas, poder-se-ia dizer ser robusto suficiente, pois ao passo em que novos dados se tornariam disponíveis pela aplicação do caso bidimensional, espera-se no mínimo um aumento do desempenho pela abundância de dados. Já um método que seja válido para todo o conjunto de imagem por depender de toda a informação simultaneamente, não deverá ter sucesso quando aplicado a apenas um dos vetores destacados.

Com base em tal proposta, o algoritmo descrito no trabalho foi adaptado para uma versão unidimensional, capaz de operar com os dados de apenas um vector, e aplicado em uma secção das imagens de teste, para assim analisar etapa a etapa o comportamento dos respectivos resultados intermediários. A Figura 26 mostra a secção de corte empregada na avaliação (posição 800). Novamente, as imagens de testes são aquelas de abertura ótica $f/1.8$ e $f/22$.

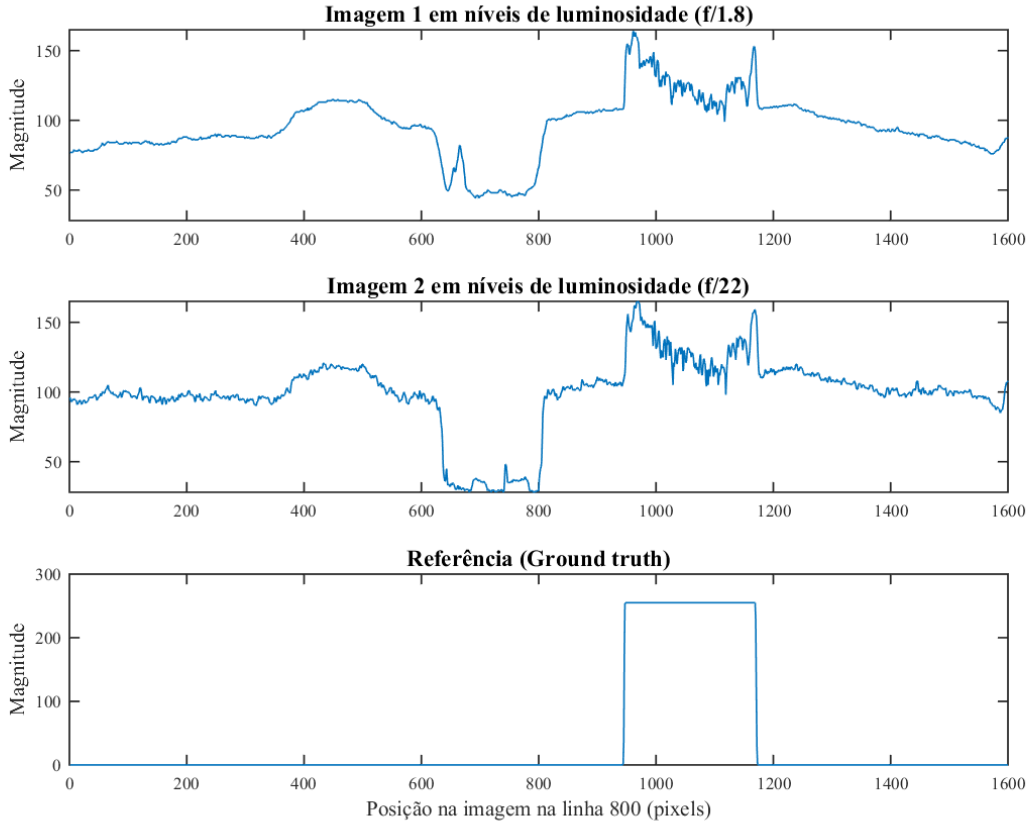
[6] Em processamento de sinais, um sinal discreto em termos do intervalo de captação de sua grandeza, é expresso por um sequenciamento de amostras (ou *samples*, do termo em inglês) (DINIZ; SILVA; NETTO, 2014). Esses intervalos mínimos, são usualmente o tempo de amostragem, mas podem ser estendidos para representar espaço ou demais grandezas necessárias.

Figura 26 – Imagem de teste para a aplicação do algoritmo unidimensional

Fonte: Autor, 2016
Notas: Imagem de teste com abertura ótica $f/1.8$ para a aplicação do algoritmo unidimensional, destacando com uma linha vermelha a secção adotada (posição 800).

O vetor de dados unidimensional para cada uma das imagens ($f/1.8$ e $f/22$), além da imagem de referência (*ground truth*), estão apresentados na Figura 27. Nessa imagem fica evidente uma das motivações dessa análise e até da ideia do algoritmo: a presença de sinais de alta frequência na imagem de alta profundidade de campo, e a falta dos mesmos sinais em regiões de desfoque para a imagem com baixa profundidade de campo.

Figura 27 – Demonstração da secção das imagens

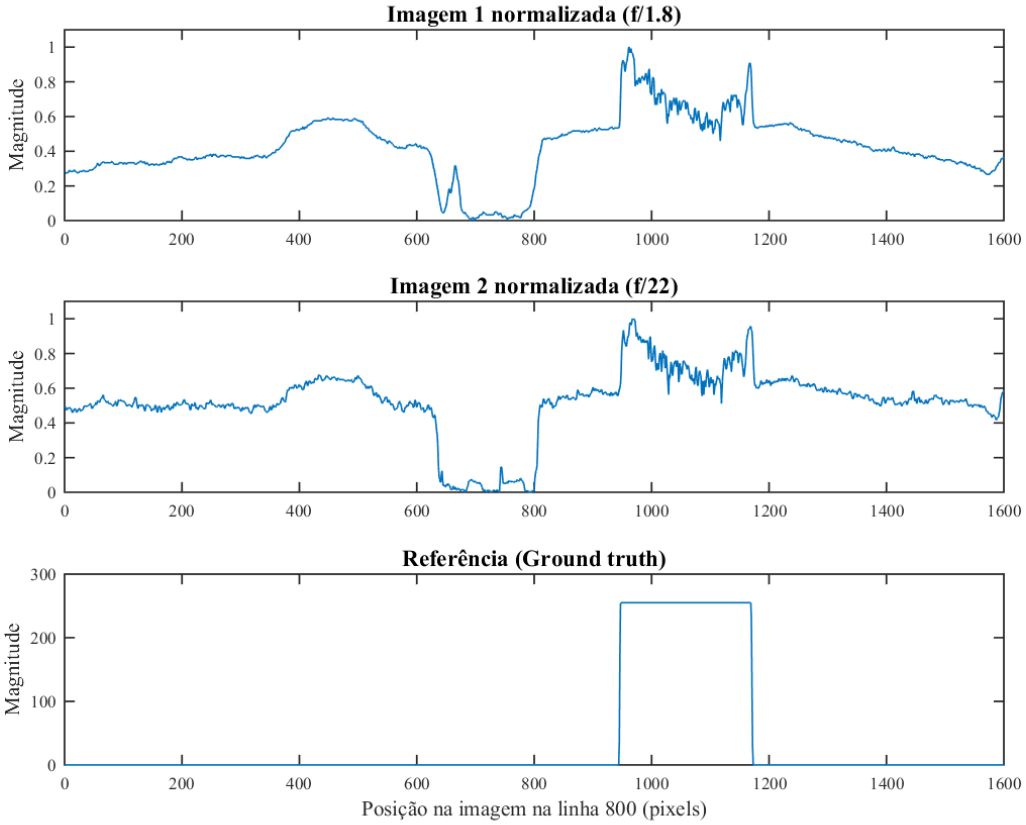



Fonte: Autor, 2016

Notas: Demonstração da mesma secção de três imagens como sinais unidimensionais. No caso, o primeiro gráfico é da secção da imagem de abertura ótica $f/1.8$ e o segundo gráfico da $f/22$. O terceiro gráfico representa a secção da referência (*ground truth*). As três secções foram feitas na mesma posição.

A etapa seguinte no processo é a normalização dos dados de entrada apenas, apresentada na Figura 28, para equiparar distorções na captura de cada uma das imagens.

Figura 28 – Normalização da secção das imagens



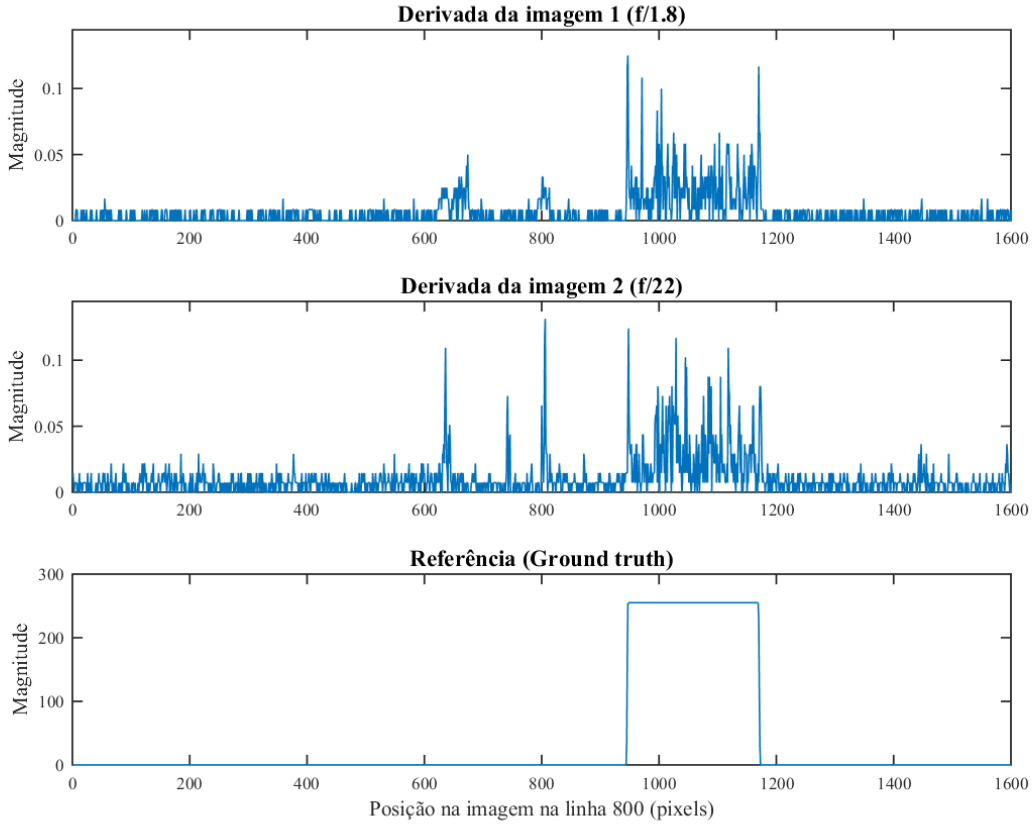

Fonte: Autor, 2016

Notas: Demonstração da operação de normalização dos sinais unidimensionais feitos com a secção de imagens. O primeiro gráfico diz respeito à imagem de abertura ótica $f/1.8$ e o segundo da $f/22$. O terceiro gráfico representa a imagem de referência (*ground truth*). As três secções foram feitas na mesma posição.

No passo seguinte é feito o cálculo da derivada em módulo do vetor, ilustrado na Figura 29. Essa operação é equivalente à operação de gradiente, que só pode ser definida na sua forma convencional para duas ou mais dimensões. Nessa etapa é evidenciado na imagem de baixa profundidade de campo que os altos picos na derivada ocorrem para a região em foco, ao passo que na região sem foco a derivada é muito pequena. Isso novamente caracteriza a presença de detalhes ou não na imagem.

Apesar da semelhança visual dos sinais vistos na Figura 28, o gradiente, por ser uma operação derivativa, introduz mudanças, destacando pequenas diferenças entre os sinais. Assim, a região em que as imagens são muito similares (onde ambas possuem foco, e destacado pelo gráfico de referência) passam a ter sinais muito diversos. Além disso, uma segunda característica do gradiente ou da derivada é que há uma remoção do nível constante (também conhecido como nível DC) e até frequências mais baixas dos sinais, pelo seu próprio preceito, colocando os sinais de maneira mais equiparável.

Figura 29 – Aplicação de operador gradiente na secção das imagens

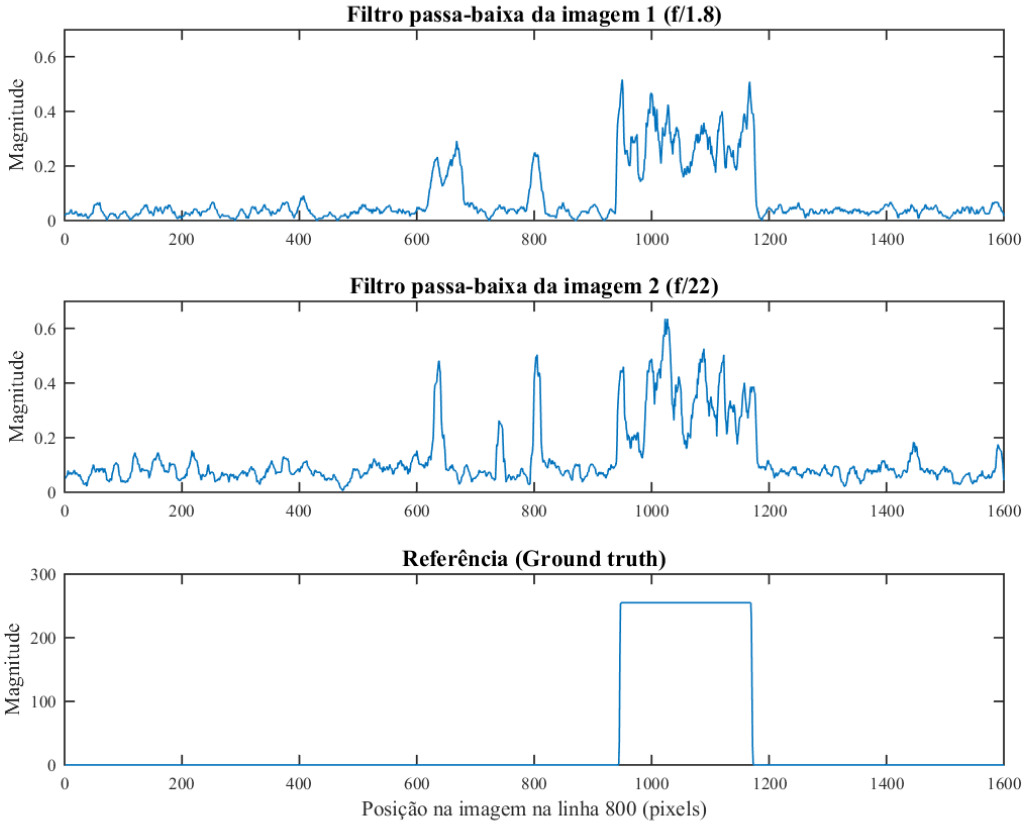



Fonte: Autor, 2016

Notas: Demonstração da aplicação do operador gradiente (ou derivada) dos sinais unidimensionais feitos com a secção de imagens. O primeiro gráfico diz respeito à imagem de abertura ótica $f/1.8$ e o segundo gráfico da $f/22$. O terceiro gráfico representa a imagem de referência (*ground truth*). As três secções foram feitas na mesma posição.

Adiante é aplicado um filtro passa-baixa do tipo média móvel (com *kernel* de tamanho 12). O resultado é demonstrado na Figura 30. A média móvel tem o intuito de mitigar as oscilações causadas pela derivada, chegando a um valor médio por região. Essa medida é interessante pois como pode ser observado pelos gráficos, havia um casamento entre ambos os sinais na região de interesse (com foco na imagem de baixa profundidade) e que foi perdido em parte quando aplicou-se a derivada dos mesmos.

Figura 30 – Aplicação de um filtro passa-baixa na secção das imagens

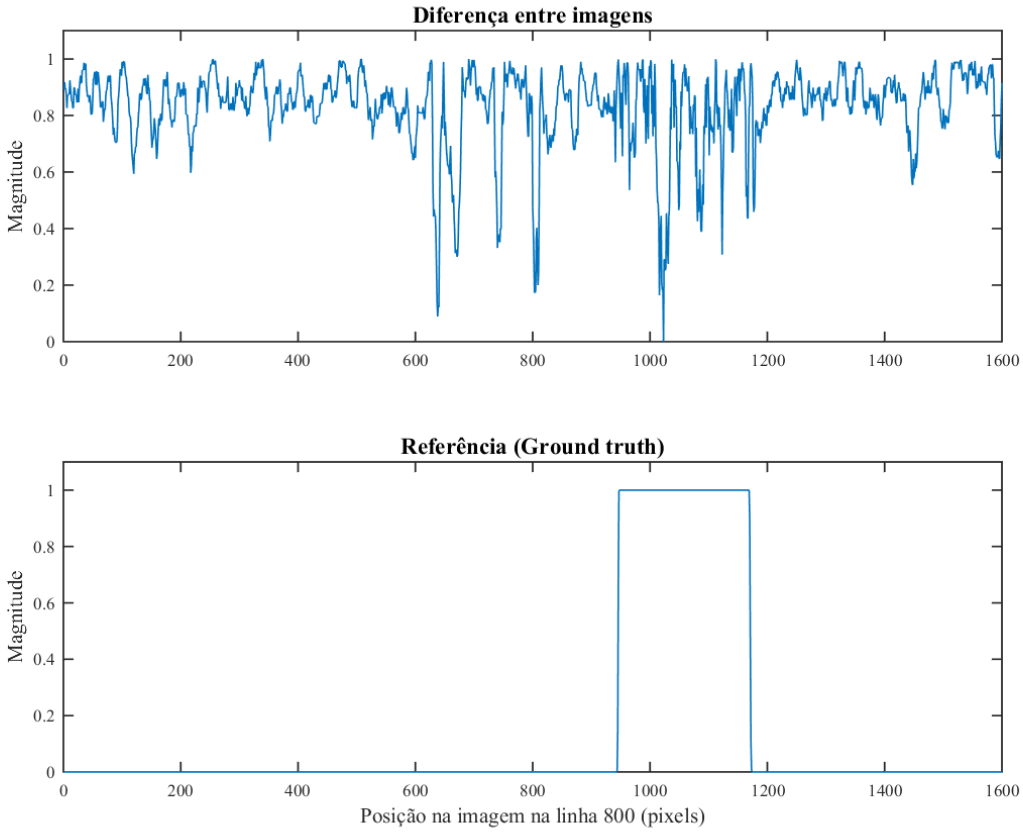



Fonte: Autor, 2016

Notas: Demonstração da aplicação de um filtro passa-vaixa nos sinais unidimensionais feitos com a secção de imagens. O primeiro gráfico diz respeito à imagem de abertura ótica $f/1.8$ e o segundo gráfico para $f/22$. O terceiro gráfico representa a imagem de referência (*ground truth*). As três secções foram feitas na mesma posição.

Como o intuito é localizar as semelhanças entre os sinais, a forma mais objetiva para isso é pelo complemento do módulo da diferença entre os mesmos. Nessa operação regiões do sinal que são iguais ou semelhantes tenderão à um, ao passo que as regiões discrepantes terão um valor menor. Assim, quanto maior o valor obtido numa amostra, mais similares são os valores para os sinais na mesma posição. O resultado está apresentado na Figura 31.

Figura 31 – Operação de complemento da diferenciação em módulo e normalizada dos sinais da secção das imagens

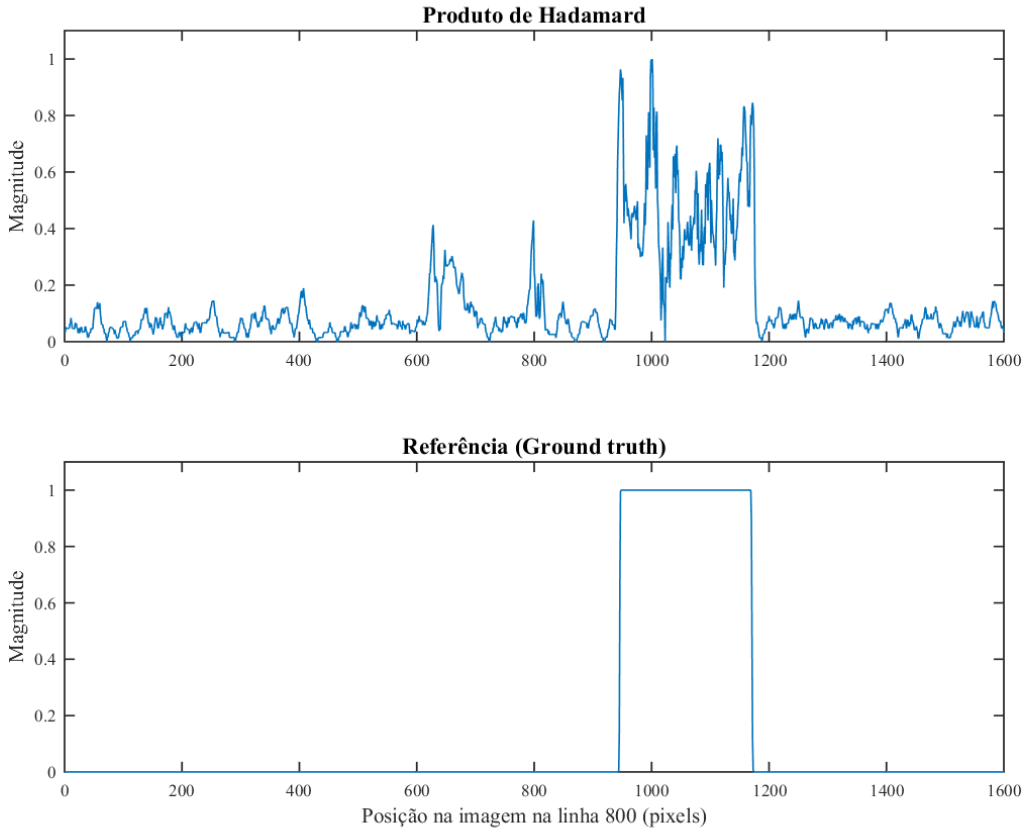



Fonte: Autor, 2016

Notas: Demonstração da operação de complemento da diferenciação dos sinais unidimensionais em módulo e normalizados, feitos com a secção de imagens. O primeiro gráfico é o resultado do processo. O segundo gráfico representa a imagem de referência (*ground truth*).

Retomando agora os resultados da etapa anterior (Figura 31) junto do resultado da aplicação do filtro passa-baixa sobre o gradiente da imagem, demonstrado na Figura 30, aplica-se o produto de Hadamard entre ambos os sinais, o que reforça no sinal original as regiões de interesse. Essa operação visa claramente a união dos resultados de duas abordagens distintas. O produto irá tender a zero para regiões em que apenas o método de análise de altas frequências ou apenas a diferença entre imagens foram relevantes. Em contraste, o produto será alto quando simultaneamente as duas abordagens obtiverem um fator alto de detecção. A Figura 32 demonstra visualmente o resultado.

Figura 32 – Aplicação do produto de Hadamard entre sinais oriundos da secção das imagens

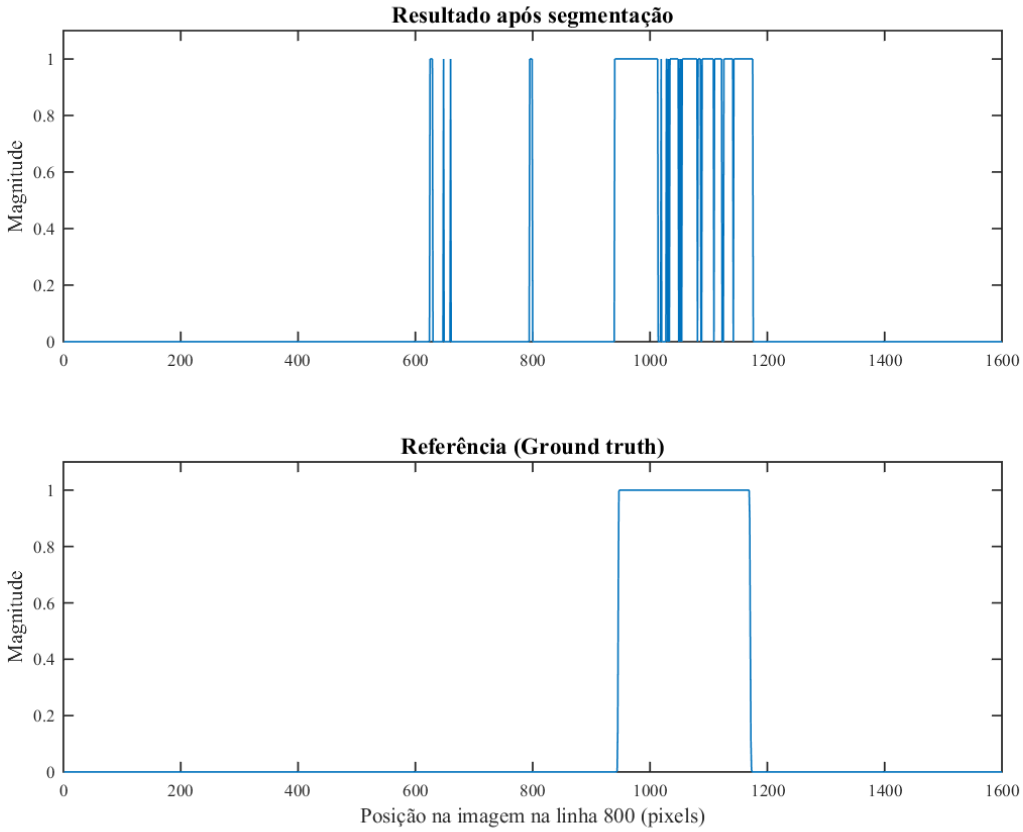



Fonte: Autor, 2016

Notas: Demonstração da operação do produto de Hadamard entre o resultado da diferenciação dos sinais e o resultado da aplicação do filtro passa-baixa para a imagem de baixa profundidade de campo. O primeiro gráfico é o resultado do processo. O segundo gráfico representa a imagem de referência (*ground truth*).

Quando passa-se a ter em evidencia as regiões de interesse, a próxima etapa é discretizar os sinais, indicando como uma amostra válida ou não. Para isso é feito a segmentação binária via o método de Otsu. O resultado está apresentado na Figura 33.

Figura 33 – Representação da binarização por Otsu dos sinais oriundos da secção das imagens

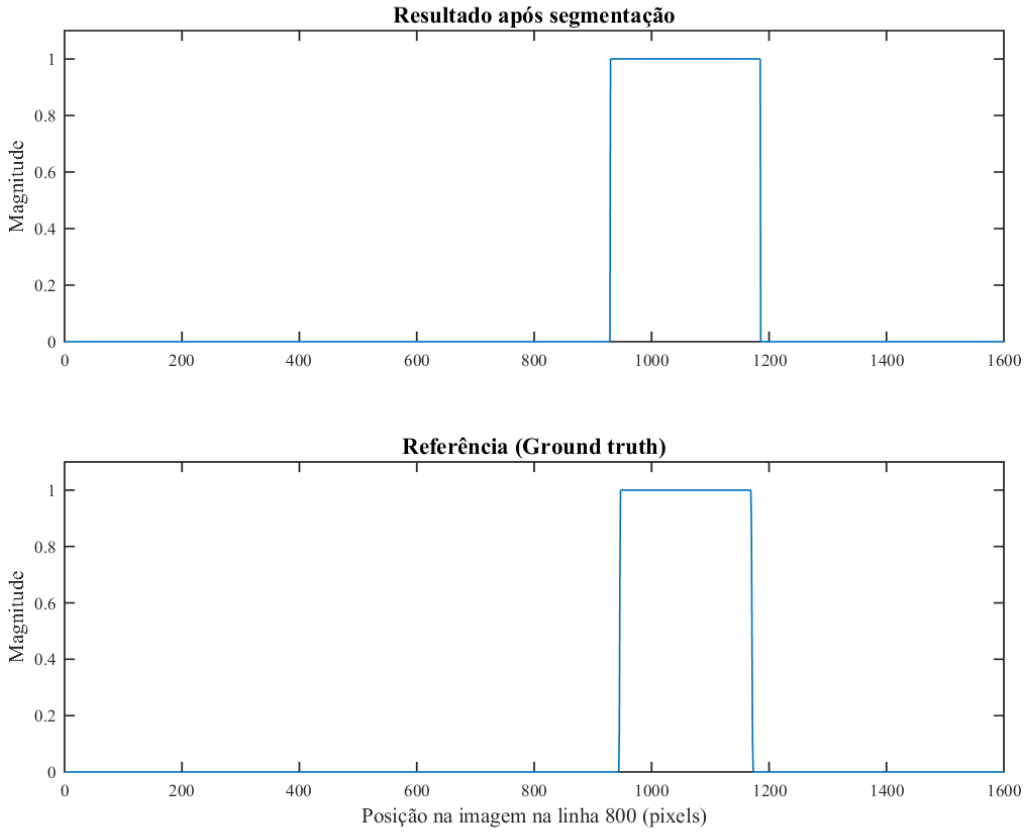



Fonte: Autor, 2016

Notas: Demonstração da binarização dos resultados via método de Otsu para os sinais obtidos da secção das imagens. O primeiro gráfico é o resultado do processo. O segundo gráfico representa a imagem de referência (*ground truth*).

Observando os resultado anteriores, existem ainda diferenças entre os pontos detectados e aqueles da referência. No entanto é clara a aglomeração dos resultados nas regiões de interesse. Ainda assim, existem aberturas nessas regiões, assim como pontos detectados em regiões esparsas, o que implica em falsas detecções ou pelo menos objetos muito pequenos, possivelmente fora do interesse. Para corrigir isso então são feitas operações morfológicas. Elas fecham totalmente os buracos nas aglomerações, assim como removem completamente detecções esparsas. Esse resultado é apresentado na Figura 34.

Figura 34 – Representação das operações morfológicas nos sinais oriundos da secção das imagens




Fonte: Autor, 2016

Notas: Demonstração da aplicação de operações morfológicas nos sinais obtidos da secção das imagens. O primeiro gráfico é o resultado do processo. O segundo gráfico representa a imagem de referência (*ground truth*).

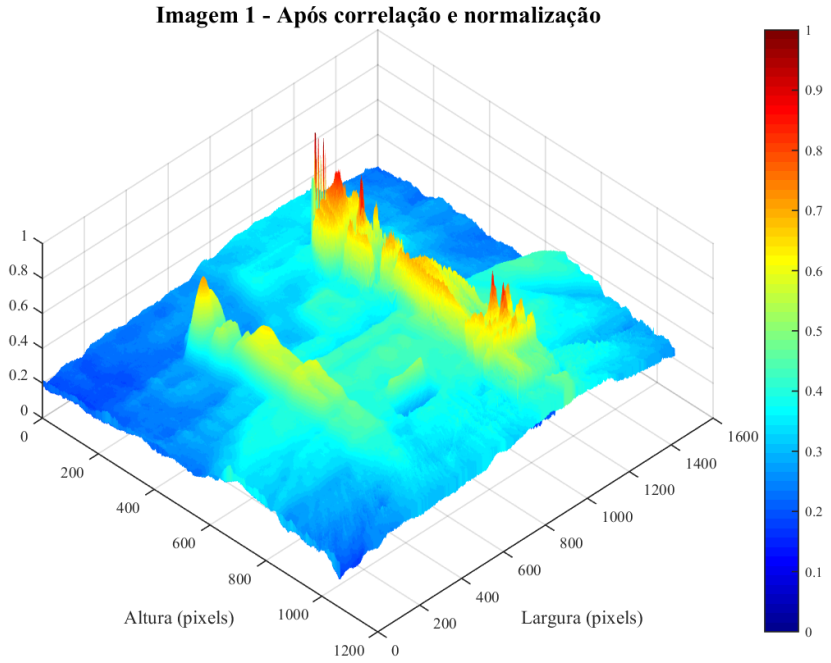
Apesar de estarmos apenas demonstrando a operação do método num caso unidimensional é visível a grande semelhança entre o que foi obtido e o que era esperado. A titulo quantitativo, a acurácia dessa segmentação chegou a 98,38%.

### 3.6.2 Prova de conceito (sinais bidimensionais)

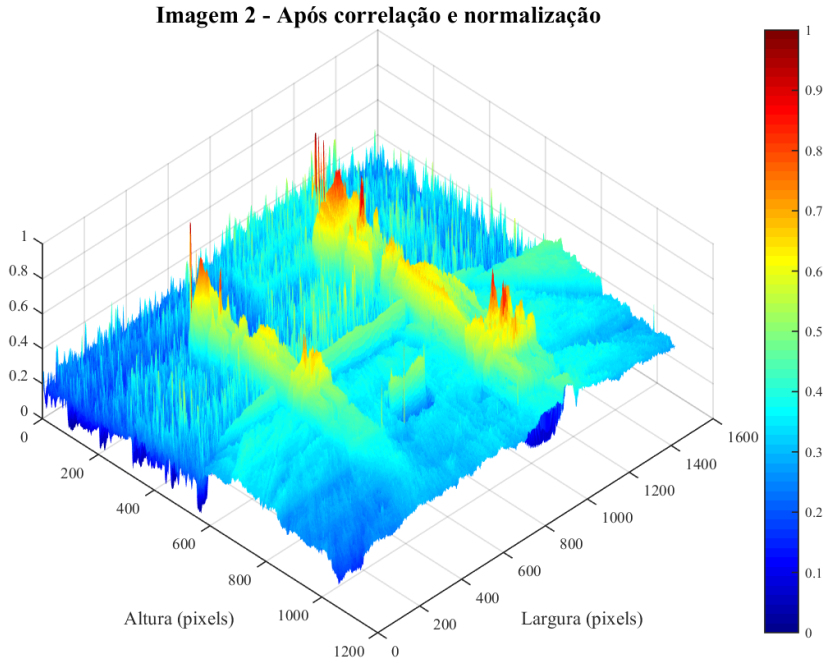
Os mesmos resultados da Figura 25 estão apresentados na forma de gráficos nas Figuras 35, 36, 37, 38 e 39. Isso facilita a observação de algumas diferenças entre as imagens, especialmente dos sinais de alta frequência na imagem de alta profundidade de campo. Além disso, abre uma perspectiva de avaliação mais próxima às práticas de processamento de sinais, como já discutido. Assim como no exemplo anterior de etapas de operação, as imagens empregadas são de abertura ótica $f/1.8$ e $f/22$.

Figura 35 – Demonstração em gráfico das imagens correlacionadas e normalizadas

(a) Imagem em escala de cinza e normalizada com menor profundidade de campo (Abertura $f/1.8$).




(b) Imagem em escala de cinza e normalizada com maior profundidade de campo (Abertura $f/22$).

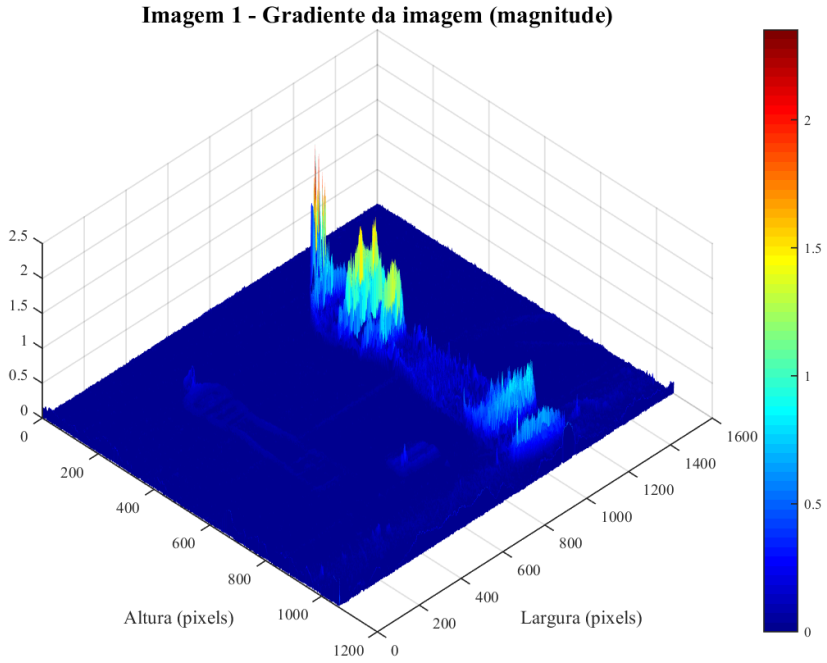



Fonte: Autor, 2016

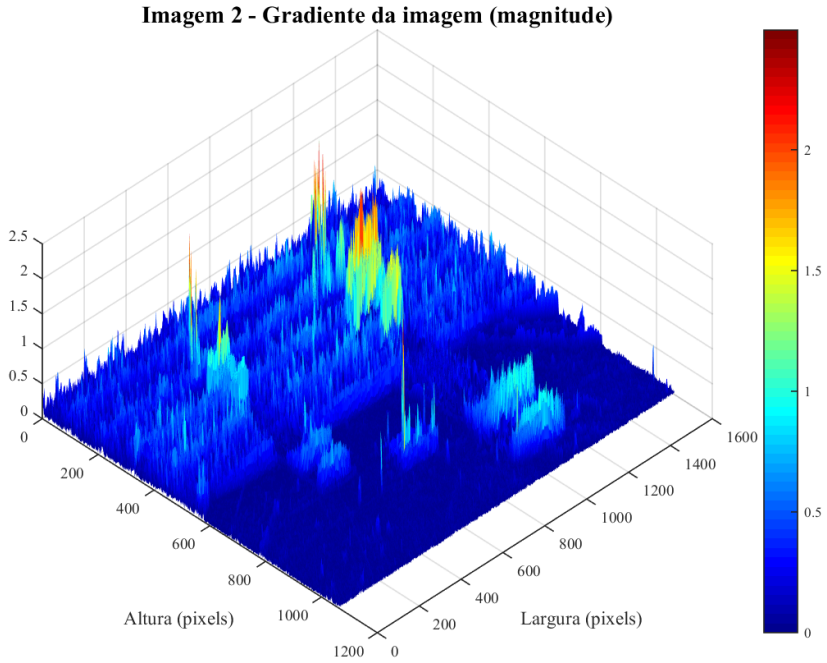
Notas: Demonstração em gráfico das imagens de baixa e alta profundidade de campo ($f/1.8$ e $f/22$ respectivamente) correlacionadas e normalizadas, conforme apresentado na Figura 25.

Figura 36 – Demonstração em gráfico das imagens após aplicação do operador gradiente

(a) Resultado do cálculo da magnitude do gradiente para imagem de menor profundidade de campo.




(b) Resultado do cálculo da magnitude do gradiente para imagem de maior profundidade de campo.

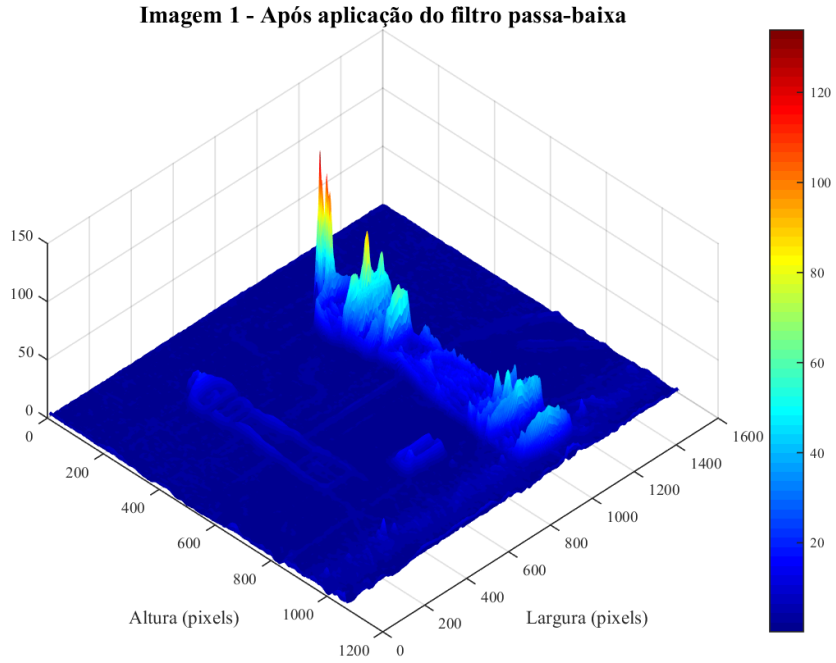



Fonte: Autor, 2016

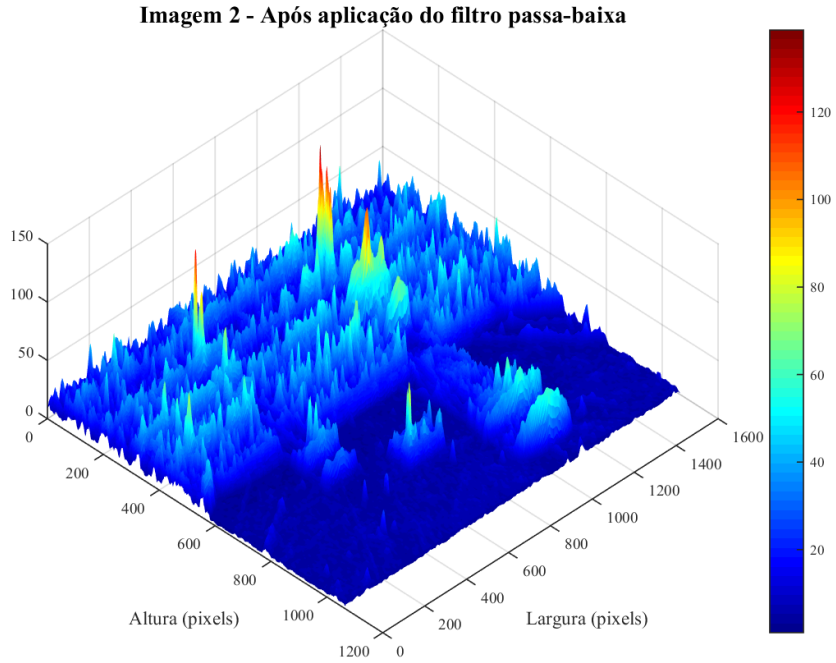
Notas: Demonstração em gráfico das imagens de baixa e alta profundidade de campo ($f/1.8$ e $f/22$ respectivamente) após aplicação do operador gradiente em módulo, conforme apresentado na Figura 25.

Figura 37 – Demonstração em gráfico das imagens após aplicação do filtro passa-baixa

(a) Resultado da aplicação do filtro passa-baixa para a imagem de menor profundidade de campo.




(b) Resultado da aplicação do filtro passa-baixa para a imagem de maior profundidade de campo.

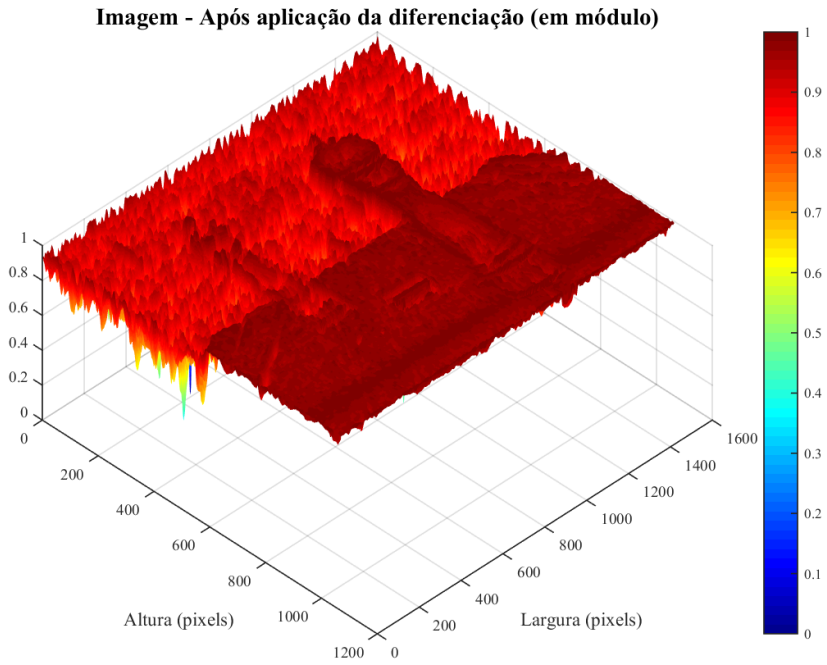



Fonte: Autor, 2016

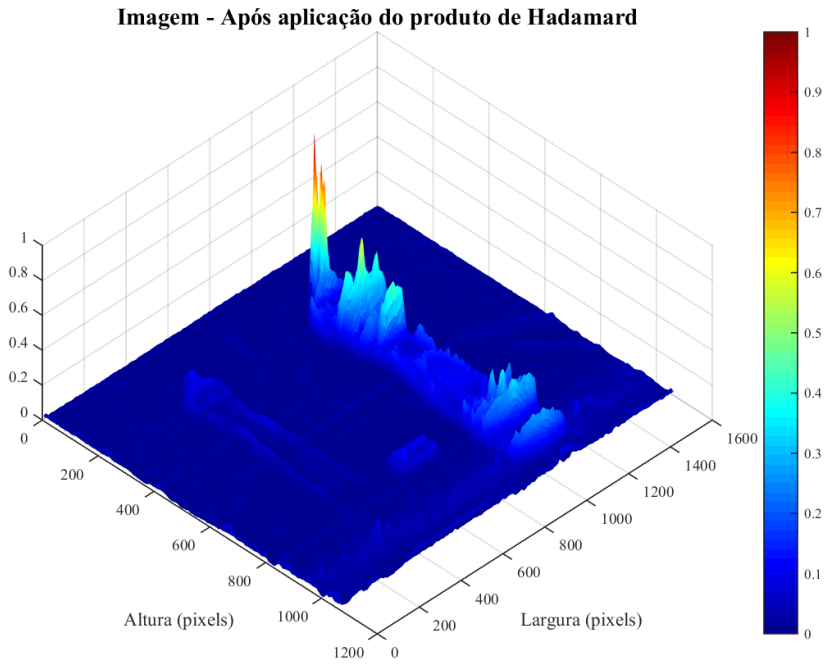
Notas: Demonstração em gráfico das imagens de baixa e alta profundidade de campo ($f/1.8$ e $f/22$ respectivamente) após aplicação do filtro passa-baixa, conforme apresentado na Figura 25.

Figura 38 – Demonstração em gráfico das imagens após operação de diferenciação e do produto de Hadamard

(a) Resultado da diferenciação das duas imagens em módulo.




(b) Resultado da aplicação do produto de Hadamard.

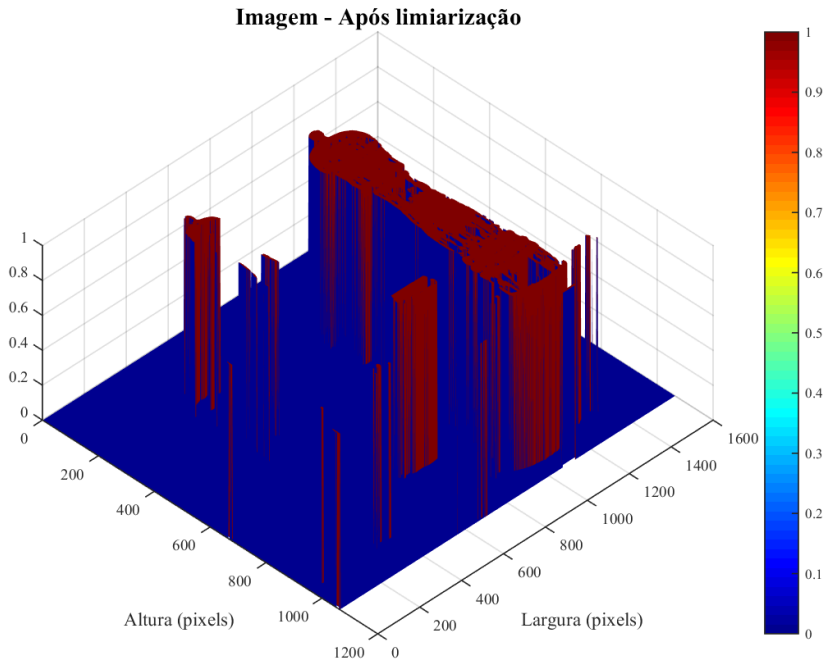



Fonte: Autor, 2015

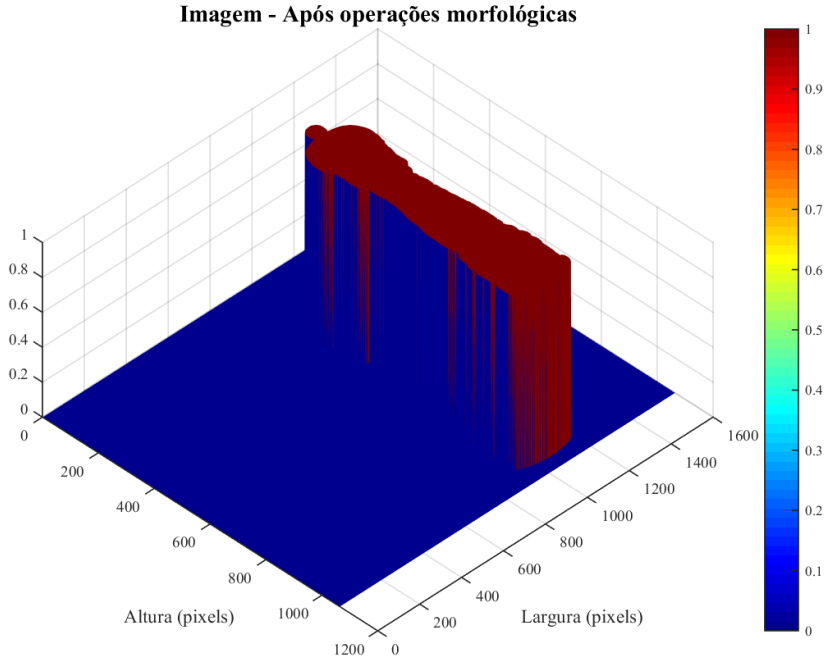
Notas: Demonstração em gráfico das operações de complemento da diferença em módulo e normalizados das duas imagens, e da aplicação do produto de Hadamard desse resultado com o resultado do filtro passa-baixa da imagem de baixa profundidade de campo ($f/1.8$ e $f/22$ respectivamente) após aplicação do filtro passa-baixa, conforme apresentado na Figura 25.

Figura 39 – Demonstração em gráfico das imagens após binarização via Otsu e das operações morfológicas

(a) Resultado da limiarização pelo método de Otsu.




(b) Resultado da aplicação de morfologia matermática.




Fonte: Autor, 2015

Notas: Demonstração em gráfico das operações de binarização via método de Otsu e das operações morfológicas, conforme apresentado na Figura 25.

### 3.6.3 Análise de complexidade

Empregando a análise de complexidade (*Big-Oh*) do algoritmo, dado que o tamanho de cada imagem de entrada é $M$ por $N$, então para cada bloco de operação, tem-se:

a) Conversão para cinza:

$$f_1(x) \propto O\left(M.N\right) \ .$$

b) Alinhamento por correção matemática máxima, ou também correção cruzada. Esse termo é mais complexo e o resultado não é tão direto, sendo assim, é oportuno um detalhamento. Primeiramente as duas matrizes analisados são multiplicadas termo a termo e então somados, apresentando um valor numérico que é o fator de correlação. Essa operação é repetida para cada deslocamento possível de uma matriz sobre a outra, e assim, o valor máximo de correlação é escolhido, por representar aquele em que ocorreu a maior semelhança, assim:

$$f_2(x) \propto O\left(M^2.N^2\right) \ .$$

Na prática esse resultado pode ser bem diferente. Sabendo das características *a priori* da captura das imagens, é possível limitar a operação de busca de correlação máxima para um limite muito pequeno. Nesse caso, sendo o limite dado por um fator $l_C$ entre 0 e 1 em função do tamanho da imagem total, também podendo ser descrito como $L_C = l_C.M.N$, então:

$$f_2(x) \propto O\left(l_C.M^2.N^2\right) \ ,$$
$$f_2(x) \propto O\left(L_C.M.N\right) \ .$$

c) Normalização das imagens:

$$f_3(x) \propto O\left(M.N\right) \ .$$

d) Gradiente da imagem. Como se trata de convolução do operador de Sobel em dois sentidos e depois é feito o cálculo da magnitude, a sua complexidade é proporcional apenas ao operador de convolução, visto que as demais operações possuem complexidade inferior. A operação de convolução para um *kernel* de dimensão $k_{SA}$ por $k_{SL}$ no caso do operador de Sobel é simplificada pelo fato de $k_A = k_L = 3$, assim:

$$f_4(x) \propto O\left(k_{SA}.k_{SL}.M.N\right) \ ,$$
$$f_4(x) \propto O\left(3.3.M.N\right) \ ,$$
$$f_4(x) \propto O\left(M.N\right) \ .$$

e) Aplicação de um filtro passa-baixa constante. Também por se tratar de uma convolução de matrizes, há uma analogia aos resultados anteriores, salvo o fato de nesse caso, o operador não ser prefixado, mas um parâmetro do método. Denominando então a dimensão do *kernel* do filtro passa-baixa por $k_{PBA}$ por $k_{PBL}$, então:

$$f_5(x) \propto O\left(k_{PBA}.k_{PBL}.M.N\right) \ .$$

f) Diferenciação das imagens:

$$f_6(x) \propto O\left(M.N\right) \ .$$

g) Produto de Hadamard:

$$f_7(x) \propto O\left(M.N\right) \ .$$

h) Segmentação pelo método Otsu. Esse método faz a segmentação pelo histograma da imagem. Sendo apenas 2 níveis de segmentação e supondo que a imagem tenha componentes de 0 a 255 (que presenta o pior caso), poder-se-ia dizer que a operação seria proporcional a $O\left(256\right)$, mas como existe a necessidade de contabilização dos níveis da imagem na conversão pra um histograma, além da necessidade posterior de reajustar os níveis de cada pixel:

$$f_8(x) \propto O\left(M.N\right) \ .$$

i) Operações morfológicas. Como se trata de um sequenciamento de operações morfológicas, vale separá-las em grupos por semelhança de operação. No primeiro caso estão as operações de dilatação e erosão. Para elas, a operação num pior caso seria o algoritmo ter de visitar cada pixel da imagem e então seus vizinhos (8 no pior caso) e fazer a respectiva operação. Claro que essa hipótese é absurda, visto que qualquer implementação real desses métodos poderia facilmente otimizar isso. No entanto, ainda considerando o pior caso, temos que as operações estão dentro das mesmas características das demais operações do método geral:

$$f_9(x) \propto O\left(8.M.N\right) \ ,$$

$$f_9(x) \propto O\left(M.N\right) \ .$$

Já para a operação de fechamento de buracos e remoção de pequenos objetos, pode ser avaliada no pior caso, como existindo a necessidade do algoritmo visitar todos os pixels da imagem, e então, a partir de cada, visitar cada um demais pixels para ver a continuidade da região. Novamente, é uma situação absurda pois qualquer implementação prática terá desempenho melhor do que essa, mas que descreve uma possibilidade de análise no pior caso. Há a ressalva no entanto que há uma limitação do número máximo de pixels que devem ser visitados a partir de cada origem. Denominando tal limite como um fato do tamanho total da imagem, tem-se $L_M = l_M.M.N$, onde $l_M$ varia entre 0 e 1 (sendo que no caso igual a 1, toda a imagem volta a ser percorrida). A complexidade do algoritmo é então dada por:

$$f_{10}(x) \propto O\left(l_M.M.N.M.N\right) \ ,$$

$$f_{10}(x) \propto O\left(l_M.M^2.N^2\right) \ ,$$

$$f_{10}(x) \propto O\left(L_M.M.N\right) \ .$$

Somando os resultados, e já considerando a simplificação por termos repetidos, a complexidade total é dada por:

$$f(x) \propto O\left(L_C.M.N\right) + O\left(k_{PBA}.k_{PBL}.M.N\right) + O\left(L_M.M.N\right) + O\left(M.N\right) \ ,$$

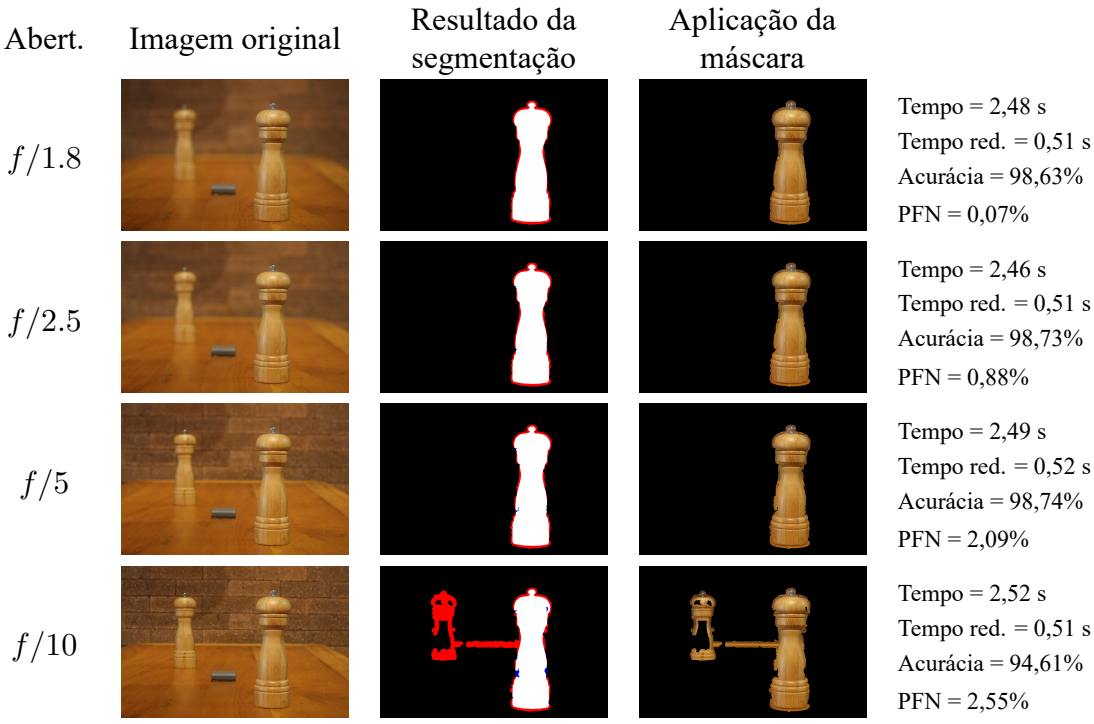
Como os valores usuais de $k_{PBA}$, $k_{PBL}$, $L_M$ e $L_C$ são usualmente muito pequenos, ou seja, muito menores do que $M$ ou $N$:

$$f(x) \propto O\left(M.N\right) \ . \tag{43}$$

## 3.7 RESULTADOS

A avaliação do novo método proposto não possui resultados extensivos para imagens reais em virtude da falta de um banco de imagens no formato necessário: par de imagens e mais a referência. Os bancos de imagens disponíveis possuem uma única imagem e mais a referência. Independentemente disso, é ao menos possível fazer as medições de desempenho entre pares das imagens controladas . A Figura 40 apresenta os resultados para os quatro pares disponíveis, sendo que sempre foi empregada a imagem de maior profundidade de campo como base ($f/22$).

Figura 40 – Resultado da segmentação pelo método proposto para as imagens controladas




Fonte: Autor, 2015

Notas: Resultado da segmentação pelo método proposto para as imagens controladas para aberturas óticas $f/1.8$, $f/2.5$, $f/5$ e $f/10$, todas analisadas com a imagem de abertura $f/22$ juntamente dos seus respectivos resultados quantitativos.

Mesmo que limitados, os resultados indicam um bom potencial para o método: alta acurácia simultânea a baixa PFN, além de tempos de execução muito abaixo do método NK, por exemplo. Observando os resultados, vê-se uma constância nos tempos de execução, pois o algoritmo se baseia numa análise pixel a pixel, e depende muito pouco do conteúdo da imagem. As únicas etapas que avaliam o conteúdo são as operações morfológicas de fechamento (eliminação de buracos e exclusão de pequenas áreas). Ainda, um segundo resultado promissor é que a média das quatro avaliações de 2,48 segundos, reduz-se para 0,51 segundos quando desconsidera-se das medições a operação inicial de correlação entre as imagens, o que mostra que ela representa aproximadamente 79% de toda a carga de processamento para o caso de imagens dessa dimensão (1600 por 1064 pixels).

Os resultados de acurácia medidos para cada abertura ótica são dados por 98,63%, 98,73%, 98,74% e 94,61% respectivamente para $f/1.8$, $f/2.5$, $f/5$ e $f/10$. Os valores das maiores aberturas são muito próximos entre si, o que indica que em relação à acurácia há uma robustez do processo, e que na prática então permite a sua aplicação a partir de aparatos óticos mais simples e baratos. Com a maior diminuição da abertura ótica, a acurácia cai para 94,61%, o que é esperado pois há uma maior profundidade de campo, e assim, mais próximas (em termos de conteúdo) se tornam as imagens analisadas com aquela de referência $f/22$.

Paralelamente, os valores de PFN crescem monotonicamente com a diminuição da abertura ótica, implicando na degradação do desempenho da segmentação do método. Ainda assim esse comportamento qualitativo é esperado, sendo que implica numa limitação das aberturas óticas usadas. Vale ressalvar que mesmo o pior dos valores atingidos, de 2,55%, é extremamente baixo, estando longe de ser uma limitação real.

# 4 IMAGENS ARTIFICIAIS

Pela inovação do uso de duas imagens com profundidades de campos desiguais, há uma falta de bancos de imagens disponíveis na literatura. A criação de fotografias reais aos pares pode ser trabalhosa para um número elevado de imagens, e assim, pouco eficiente, ainda mais quando se almeja uma avaliação inicial. Ainda, como se trata de uma investigação comparativa, há necessidade de que mais versões da imagem estejam disponíveis para avaliação do desempenho do método frente à abertura ótica. Adicionalmente, pode ser uma tarefa árdua criar variações possíveis ou estatisticamente relevantes de composições de cenas possíveis. E, também, imagens reais precisam ser segmentadas por seres humanos para criação do *ground truth*, o que é uma tarefa demorada, e que se multiplica quando diversas versões de profundidade de campo existem.

Com base nessas limitações pelo uso de imagens reais, este capítulo descreve e implementa um método de criação de imagens artificiais em pares (ou múltiplos). A vantagem dessa metodologia é que algumas limitações supramencionadas são anuladas, tais como: rápida criação de um número grande de imagens, representatividade de diversas variações de composição e criação automática de um *ground truth* preciso.

## 4.1 PRINCÍPIOS ÓTICOS

A medida do desfoque já foi demonstrada e é dada pela Equação 25 anterior. Para o tamanho do objeto numa cena, o cálculo é feito analisando a semelhança de triângulos, seja na Figura 5 ou na Figura 10, mostradas anteriormente. A segunda imagem, apesar de ser destinada ao caso hiperfocal, possui total validade para essa análise, e por possuir já a denominação nos vértices dos triângulos, é mais indicada para a análise. Sendo então os triângulos semelhantes:

$$\triangle OBC \sim \triangle CDI \, .$$

Deriva-se a expressão:

$$\frac{o}{S_1} = \frac{i}{S_2} \, ,$$

onde $o$ e $i$ são os tamanhos do objeto e da imagem, respectivamente. Esse resultado é exatamente o mesmo já apresentado na equação 10. Utilizando a partir dai a Equação (9):

$$\frac{o}{i.S_1} = \frac{1}{S_2} = \frac{1}{f} - \frac{1}{S_1} \, .$$

Rearranjando os termos, chega-se a equação para o tamanho da imagem $i$ numa cena:

$$f.o = i\,(S_1 - f) \ ,$$

$$i = \frac{f.o}{S_1 - f} \, . \tag{44}$$

Como, na prática, sempre serão casos de imagens digitais, o valor da distância focal de uma lente é sempre em referência a um sensor chamado *full-frame* ou de 35 milímetros (TARRANT, 2007). Tais sensores possuem tamanho de 35 x 24 mm. Como todos os cálculos são feitos em referência à altura do objeto, deve-se manter a mesma grandeza como referência, assim, expressando a altura da imagem relativamente ao tamanho da imagem, $i_{Rel} = {}^{i}/_{h_S}$, onde $h_s$ é a altura do sensor, têm-se:

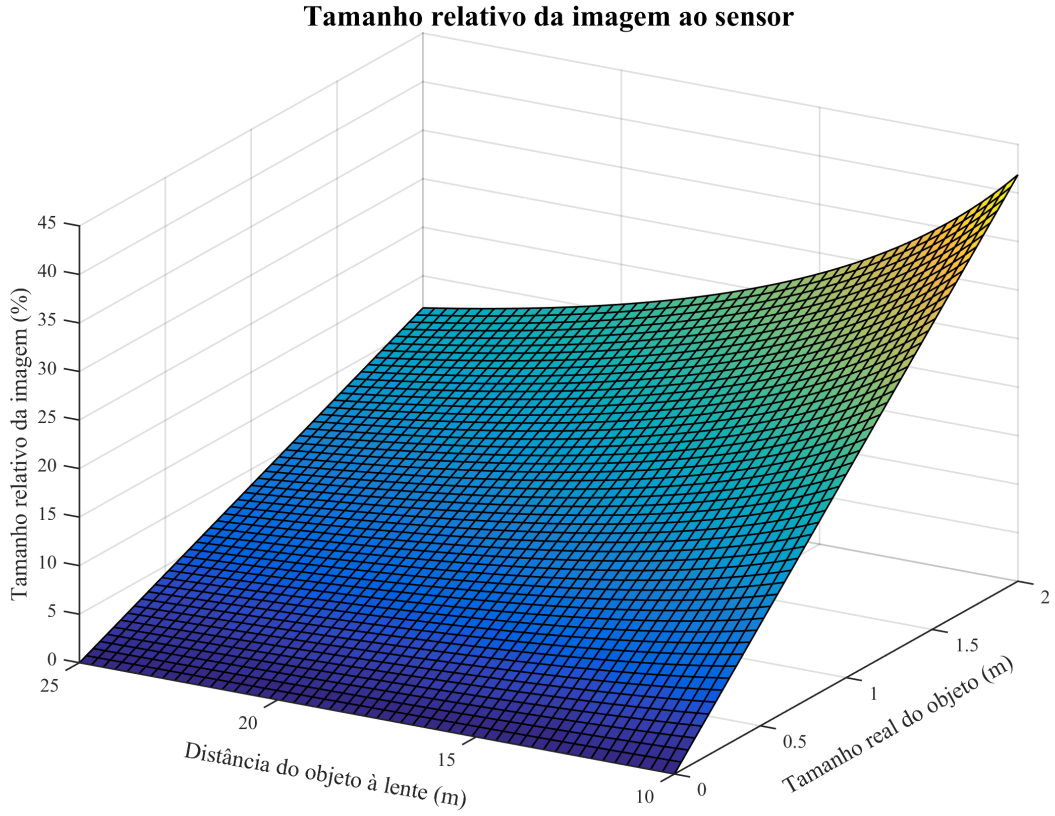
$$i_{Rel} = \frac{o.f}{h_s}.\frac{1}{(S_1 - f)} \, . \tag{45}$$

O interesse nessa representação relativa se deve ao fato do sensor ser o limite prático da imagem. Uma imagem que tenha uma determinada proporção à altura do sensor, terá a mesma proporção à altura da imagem, independentemente de amplificações realizadas no meio do processo.

Essa relação é válida para as premissas da ótica geométrica e paraxial. Assim, deve-se obedecer uma distância mínima para validade da equação. Uma possibilidade de determinação desse limite é analisando a própria equação. Como o resultado prático esperado é que haja uma relação linear (inversa) entre distância e tamanho do objeto na cena, e o que é corroborado pelo estudo da ótica anteriormente, então apenas a parte (domínio) em que a equação se torna aproximadamente linear é empregada, assim, o que ocorre para um objeto a uma distância a partir de aproximadamente 20 vezes a distância focal que, no exemplo de uma lente de 50 mm, se situa a 10 metros da câmera. Vale ressalvar que com o aumento dessa distância, as aberrações óticas tornam-se menos evidentes.

Como exemplo, para uma lente de distancia focal de 50 mm montada numa câmera com sensor de 35 mm, a equação gera os valores representados na Figura 41.

Figura 41 – Exemplo de tamanho relativo de imagem numa cena




Fonte: Autor, 2016

Notas: Exemplo de mapa do tamanho relativo ao sensor (e imagem final) de um objeto em cena. Parâmetros fixados: distância focal de 50 mm e altura do sensor de 24 mm (sensor de 25 mm).

### 4.1.1 Formação do desfoque

Apesar do cálculo do tamanho do círculo de confusão já ter sido demonstrado, ainda é necessário entender como o mesmo é composto, ou mais precisamente, de como a energia é distribuída dentro dele.

A realidade é evidentemente bem complexa, visto que o desfoque é valido apenas para um ponto, como o entendimento da questão feito até o momento. Sendo assim, a aplicação real do conceito é de grande complexidade, pois cada ponto do objeto na cena possui uma distância diferente com relação a lente e, assim, um círculo de confusão próprio. Ainda contribui para essa complexidade a sobreposição de resultados, pois as distribuições da luz em cada ponto do objeto quando projetada na imagem está espalhada. Também, contribuições adicionais são feitas por difração da luz ou aberrações da ótica.

Essa análise é de grande importância pois evidencia que a convolução de matrizes, comum em questões ligadas a processamento de imagens, pode ser tratada apenas como uma aproximação. Todas essas dificuldades já são conhecidas de longa data de pesquisadores, como é visto em (POTMESIL; CHAKRAVARTY, 1981).

Muitas soluções já existem para a questão, pois é de grande importância hoje para a industria cinematográfica e afins, quando o efeito da profundidade de campo deva ser criado virtualmente, como no caso de animações gráficas. As técnicas existentes apresentam resultados melhores ou piores em termos de verossimilhança dos resultados ou desempenho computacional. Esses algoritmos são descritos e analisados em (BARSKY; KOSLOFF, 2008). O que há em comum, no entanto, é que todas as técnicas, por desejarem uma fidelidade do resultado, são válidas para casos em que há grande informação da cena em questão por serem normalmente aplicados em renderização de imagens de esquemas em 3 dimensões (o número de dimensões na realidade é maior por haver informações de cores por exemplo). Assim, caso alguma das técnicas seja aplicada, é necessária a informação espacial de toda a cena.

Uma aproximação possível nas cenas é discretizar os planos da cena. Empregando limitação dos tamanhos dos objetos e da distribuição dos elementos em cena, essa aproximação torna-se melhor em comparação com o resultado real. Com isso, cada elemento da cena é tratado como dentro de um único plano. Consequentemente, cada ponto do objeto terá a mesma contribuição na formação da imagem final.

A distribuição da luz de um ponto sobre um plano (plano imagem) é denominada por função de espalhamento de ponto, normalmente usando a sigla PSF do termo em inglês *point spread function*. Como é feito em (KUTHIRUMMAL et al., 2010) e (CLAXTON; STAUNTON, 2008), a função PSF pode ser aproximada por uma distribuição gaussiana conforme abaixo:

$$PSF = \frac{2}{\pi \left(g.b\right)^2} . \exp\left( -\frac{2.r^2}{\left(g.b\right)^2} \right) , \tag{46}$$

onde $g$ é uma constante, $r$ é o raio do ponto dentro do círculo de confusão e $b$ é dado por:

$$b = \frac{n}{f} . \left| f - S_1 \right| . \tag{47}$$

A partir da PSF, passa a ser válida a operação de uma convolução da função sobre os elementos no seu próprio plano.

## 4.2 IMAGENS ARTIFICIAIS

Uma maneira adicional de avaliar os métodos é com o uso de imagens criadas artificialmente. A proposta nesse caso é o uso de uma imagem fundo e a de elementos independentes sobrepostos. Cada elemento possui características de cenas únicas, incluindo o fundo, a distância à câmera, ou posição em cena, além de informações próprias específicas, como sua dimensão. Com tais dados, e a partir dos resultados demonstrados anteriormente, em especial pelas equações (25) e (45) e também adotando uma PSF e limitações das condições espaciais e ótica, é possível simular uma imagem real aproximadamente.

As vantagens dessa técnica é que o controle sobre a imagem é muito maior, podendo explorar situações mais abrangentes do que apenas pelo uso de imagens reais, em virtude do maior custo (necessidade de equipamento específico e maior tempo necessário) no último caso.

O emprego de um algoritmo de criação de imagens artificiais pode gerar uma enorme quantidade de imagens de testes, explorando diversas situações num curto espaço de tempo.

A técnica ainda apresenta a vantagem de facilitar a criação do *ground truth*, visto que o processo passa a ser o oposto: a imagem é gerada a partir do *ground truth*, e não o oposto, como em imagens reais. Nesse caso, há maior precisão dos mesmos, pois elimina a subjetividade na sua especificação. Por último, de maneira análoga, os parâmetros óticos podem ser variados para uma mesma cena, permitindo que valores de abertura, por exemplo, possam ter seu desempenho analisado junto ao algoritmo.

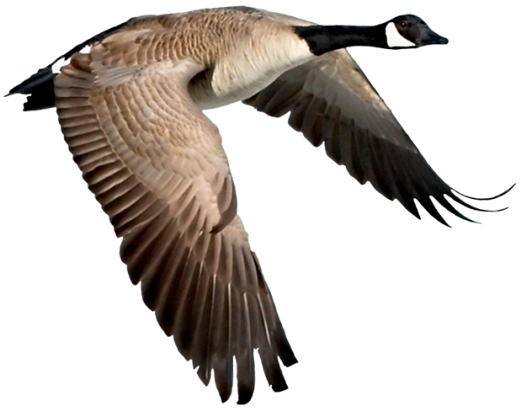
A técnica obviamente possui também suas desvantagens com relação a imagens reais: como já discutido, a incerteza nos cálculos pode gerar imagens que não são exatamente iguais àquelas reais, o que implica numa necessidade de limitação dos casos ou na melhoria dos modelos matemáticos em que deseja-se criar as imagens. Ainda, há uma limitação (também pela razão do modelo matemático) em que objetos extensos não podem ser representados. Tais objetos são aqueles em que o tamanho do círculo de confusão difere muito ao longo de sua superfície, e assim, podem ter partes do objeto em foco e outras partes sem foco numa cena. É possível tratar tais casos com o uso de representações mais detalhadas da cena, construindo a cena toda numa estrutura de 6 dimensões (3 espaciais e mais 3 para cor) e então renderizando uma cena apenas. No entanto essa técnica apresenta grande complexidade para criação das cenas, e pouco aplicável nesse momento, pois muitos casos de estudo não se aplicam a tal critério.

Para a análise, objetivou-se a criação de um algoritmo que a partir da formulação matemática já apresentada, permitisse a criação aleatória de imagens de teste, aplicando as limitações já explicitas anteriormente. A composição da cena foi feita baseada em dois grupos: fundo e elementos. Por a técnica ser semelhante a de uso em muitas ferramentas de desenvolvimento gráficos, empregou-se a mesma convenção de nomes, ou seja, respectivamente, *stage* e *sprite*. Algumas das *sprites* empregadas estão exemplificadas na Figura 42. Por serem já imagens sem fundo, com um canal alfa de transparência (imagens de 32 bits, sendo 24 bits para RGB e 8 bits para alfa), a segmentação é automática. Os objetos foram selecionados de forma a não caírem no caso de grandes extensões e invalidar a técnica.

Figura 42 – Exemplos de *sprites*

(a) Exemplo 1.

(b) Exemplo 2.



(c) Exemplo 3.

(d) Exemplo 4.

(e) Exemplo 5.

(f) Exemplo 6.

Fonte: Autor, 2015

Notas: Exemplo de *sprites* (elementos) usadas na composição das cenas.

Para cada *sprite,* um tamanho real do objeto estimado foi anotado e gravado, a fim de manter a proporcionalidade entre eles em cena. Analogamente, foram definidas imagens para os fundos, como mostra a Figura 43. No caso dos *stages*, para cada imagem há uma distância preestabelecida.

Figura 43 – Exemplos de *stages*

(a) Exemplo 1.

(b) Exemplo 2.

(c) Exemplo 3.

(d) Exemplo 4.

Fonte: Autor, 2015

Notas: Exemplos de *stages* (fundos) das composições de cenas.

O algoritmo gera imagens aleatórias, realizando os seguintes sorteios:

- Imagem de *stage* que será usada;
- Número de *sprites* que será usado (sendo o número mínimo igual a um);
- *Sprites* que serão empregados conforme o número de elementos sorteados anteriormente;
- Posição de cada *sprite* em cena, sendo possível que o elemento se localize parcialmente em cena apenas;
- Distância do *sprite* à câmera.

Para cada composição, a mesma cena é gerada variando-se a abertura da lente, afim de ter-se ao menos um par de imagens com diferentes valores de profundidade de campo. Além disso, o *ground truth* é criado, podendo ser definido um valor de tamanho de círculo de confusão $c$ limite como critério. Apenas para controle e validação, cada imagem é criada duas vezes. Na primeira há a imagem composta pura. Na segunda, são feitas anotações com os valores

calculados dos parâmetros óticos e físicos de cada objeto. Exemplo de imagens geradas estão representados na Figura 44.

Figura 44 – Exemplos de composição

(a) Exemplo com abertura $f/1.8$.

(b) Exemplo com abertura $f/5$.

(c) Exemplo com abertura $f/22$.

(d) Exemplo *Ground truth*.

Fonte: Autor, 2015

Notas: Exemplo de composição de uma cena com diversos parâmetros de profundidade de campo e também de geração de um *ground truth* automático.

O algoritmo desenvolvido está apresentado em sua integra no Apêndice B. Apesar de haver diversos detalhes em sua operação, as partes mais relevantes na composição, dado já o sorteio de elementos, fundo e parâmetros físicos (posição e distância de cada elemento), são:

a) Ordenação dos elementos para que o trabalho seja feito do elemento mais afastado em direção ao mais próximo;

b) Ajuste do tamanho de cada elemento conforme Equação (45);

c) Realização do deslocamento do objeto em cena para cada elemento;

d) Cálculo do desfoque e aplicação do mesmo conforme a Equação (25), e empregando uma convolução de uma função gaussiana. Antes de ser feita a aplicação da convolução no entanto, parte do fundo da imagem que circundaria a mesma é agregado à borda, de modo a ter sua participação no desfoque;

e) Mescla das camadas sobrepostas para formação de uma única imagem;

f) Criação do *ground truth*, dada um limite de CoC, para cada possibilidade de parâmetro ótico usado.

### 4.2.1 Comparação entre imagem (artificial e real)

Sendo o resultado da simulação oriundo de diversas aproximações, é interessante que haja um avaliação da qualidade desse processo. A proposta então é fazer uma foto real e paralelamente usar os elementos da mesma (elementos frontais e fundo) para a criação de uma versão artificial e então compará-las. As imagens reais para comparação, tanto completas, como os *sprites* destacados isoladamente estão apresentados exemplarmente na Figura 45. A separação dos elementos frontais foi realizada a partir da imagem de maior profundidade de campo (abertura igual a $f/22$) de maneira manual, isto é, através de ferramentas de edição de imagens. O operador fez o recorte do contorno. A imagem de fundo por sua vez foi feita realizando uma nova foto do cenário, mas sem os elementos frontais. A necessidade da imagem de fundo isolada é devido à recuperação de informação que ficam no plano atrás dos elementos frontais.

As imagens artificiais similares às reais foram criadas empregando o mesmo algoritmo do Apêndice B, salvo a questão de aleatoriedade. Como as imagens que deseja-se criar possuem elementos em posições bem definidos, então os sorteios dos parâmetros físicos foram suprimidos e substituídos pelos valores reais, medidos no experimento. Um esquema da cena construída para o experimento está demonstrado na Figura 46. Na cena, o foco foi feito a uma distância de 630 milímetros e os objetos nomeados como $A$, $B$ e $C$ no esquema, são respectivamente uma lente objetiva, um temporizador e um sofá. Esses elementos destacados são os ilustrados anteriormente na Figura 45.

Figura 45 – Elementos da composição para criação de cena artificial e a própria cena real de baixa profundidade de campo

(a) Cenário fotografado com abertura $f/1.8$ para minimização do DOF.

(b) Cenário fotografado com abertura $f/22$ para maximização do DOF.

(c) Cenário sem os elementos frontais fotografado com abertura $f/22$ para minimização do DOF.

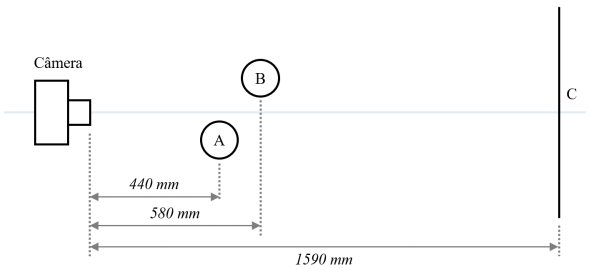
(d) Primeiro *Sprite* removido da imagem (b).

(e) Segundo *Sprite* removido da imagem (b).

Fonte: Autor, 2015

Notas: Apresentação das cenas reais usadas na comparação do método de geração de imagens artificiais. Primeiramente há a cena de baixa profundidade de campo real. Em seguida a mesma cena com alta profundidade, além do fundo feito com uma imagem real. Por último, os elementos destacados, extraídos a partir da cena de alta profundidade de campo.

Figura 46 – Esquema do aparato usado para compor as cenas de comparação




Fonte: Autor, 2015

Notas: Esquema do aparato experimental usado para compor as cenas de comparação com as medidas entre cada elemento (objetos $A$ e $B$, além do fundo da cena) e a câmera.

Uma visão geral dos resultados está descrita na Figura 47. Na mesma, para quatro opções de abertura da lente, estão representadas lado a lado as imagens reais e artificiais. Para uma análise qualitativa dos resultados, foram empregadas as medidas de correlação matemática e SSIM (*structural similarity index*, ou índice de similaridade estrutural) (WANG et al., 2004). Como sugere (WANG; BOVIK, 2009), não adotou-se uma comparação via MSE (*mean square error*, ou erro médio quadrático) ou PSNR (*Peak Signal-to-Noise Ratio*, ou relação sinal ruído de pico) por poderem apresentar resultados equivocados para a fidelidade das imagens.

O cálculo do $SSIM$ entre duas imagens $A$ e $B$ é realizado por janelas $Im_A$ e $Im_B$ da própria imagem e dado conforme a seguir, lembrando que é aplicado apenas à luminância da imagem:

$$SSIM\left(Im_A, Im_B\right) = \left[l\left(Im_A, Im_B\right)\right]^{\alpha} + \left[c\left(Im_A, Im_B\right)\right]^{\beta} + \left[s\left(Im_A, Im_B\right)\right]^{\gamma} ,$$

sendo:

$$l\left(Im_A, Im_B\right) = \frac{2.\mu_{Im_A}.\mu_{Im_B} + C_1}{\mu_{Im_A}^2 + \mu_{Im_B}^2 + C_1} ,$$

$$c\left(Im_A, Im_B\right) = \frac{2.\sigma_{Im_A}.\sigma_{Im_B} + C_2}{\sigma_{Im_A}^2 + \sigma_{Im_B}^2 + C_2} ,$$

$$s\left(Im_A, Im_B\right) = \frac{\sigma_{Im_A Im_B} + C_3}{\sigma_{Im_A}.\sigma_{Im_B} + C_3} .$$

No caso aplicado, fazendo os três expoentes iguais a 1 ($\alpha = \beta = \gamma = 1$), e fazendo $C_3 = {}^{C_2}/_2$, que é o uso padrão, então a equação se simplifica da da seguinte maneira:

$$SSIM\left(Im_A, Im_B\right) = \frac{\left(2.\mu_{Im_A}.\mu_{Im_B} + c_1\right).\left(2.\sigma_{Im_A Im_B} + c_2\right)}{\left(\mu_{Im_A}^2 + \mu_{Im_B}^2 + c_1\right).\left(\sigma_{Im_A}^2 + \sigma_{Im_B}^2 + c_2\right)} ,$$

onde $\mu_{Im_A}$ é a média de $Im_A$ , $\mu_{Im_B}$ é a média de $Im_B$ , $\sigma_{Im_A}$ é a variância de $Im_A$ , $\sigma_{Im_B}$ é a variância de $Im_B$ , $\sigma_{Im_A Im_B}$ é a covariância de $Im_A$ e $Im_B$. Os valores $c_1$ e $c_2$ são

variáveis que controlam a fórmula para que não haja uma explosão do valor de $SSIM$ quando o denominador é muito baixo, ou seja, quando a intensidade da imagem é próxima de zero na região em análise. São valores ligados à luminosidade.

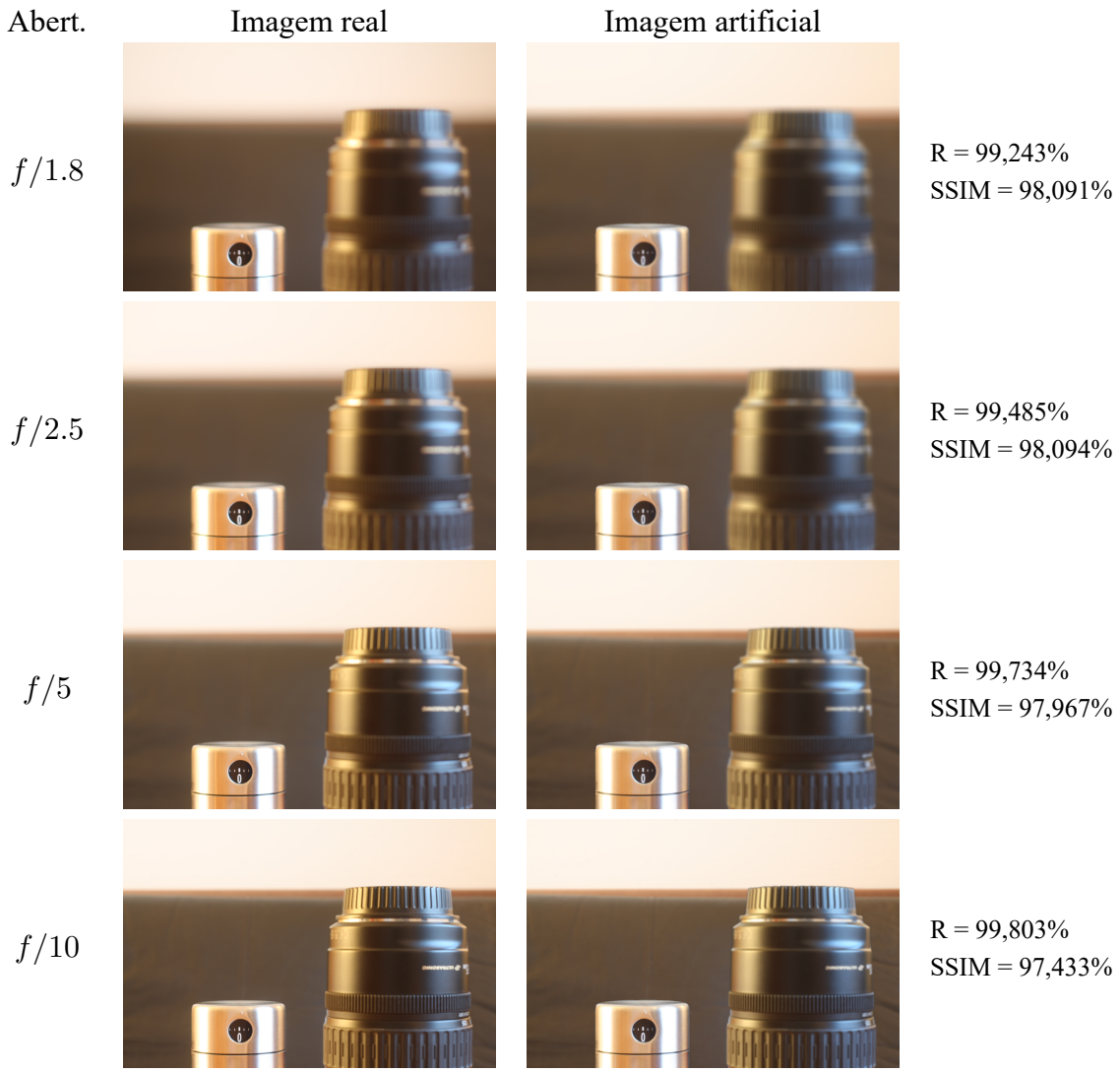
A segunda métrica qualitativa analisada, a correlação matemática $R$, é aplicada conforme equação abaixo para duas imagens $A$ e $B$. Nesse caso, como a aplicação é feita no total da imagem, as janelas $Im_A$ e $Im_B$ são as próprias imagens. Emprega-se aqui a nomenclatura de variáveis de forma a manter o mais próximo possível da apresentada para o $SSIM$. Assim:

$$R\left(Im_A, Im_B\right) = \frac{\sum_i^N \sum_j^M \left(a_{ij} - \mu_{Im_A}\right) . \left(b_{ij} - \mu_{Im_B}\right)}{\sqrt{\left(\sum_i^N \sum_j^M \left(a_{ij} - \mu_{Im_A}\right)^2\right) . \left(\sum_i^N \sum_j^M \left(b_{ij} - \mu_{Im_B}\right)^2\right)}},$$

onde $i$ e $j$ são índices de posicionamento do pixel em análise, e no caso das imagens, limitados entre 1 e o limite superior (tamanho da imagem) dados por $N$ e $M$ . Por fim, $a_{ij}$ e $b_{ij}$ são os pixels das imagens $A$ e $B$ na posição $i$ e $j$ .

A maior profundidade de campo (obtida com abertura igual a $f/22$) não foi repetida nesse experimento por se tratar da própria base dos elementos das imagens artificiais. Aplicar a técnica nesse caso seria uma questão apenas de reposicionar os objetos em cena, sem causar impactos na simulação do desfoque.

Figura 47 – Comparação entre imagens reais e imagens geradas artificialmente




Fonte: Autor, 2015

Notas: Comparação entre imagens reais (lado esquerdo) e imagens geradas artificialmente (lado direito), assim como os resultados individuais de correlação matemática $R$ e índice de similaridade estrutural (SSIM).

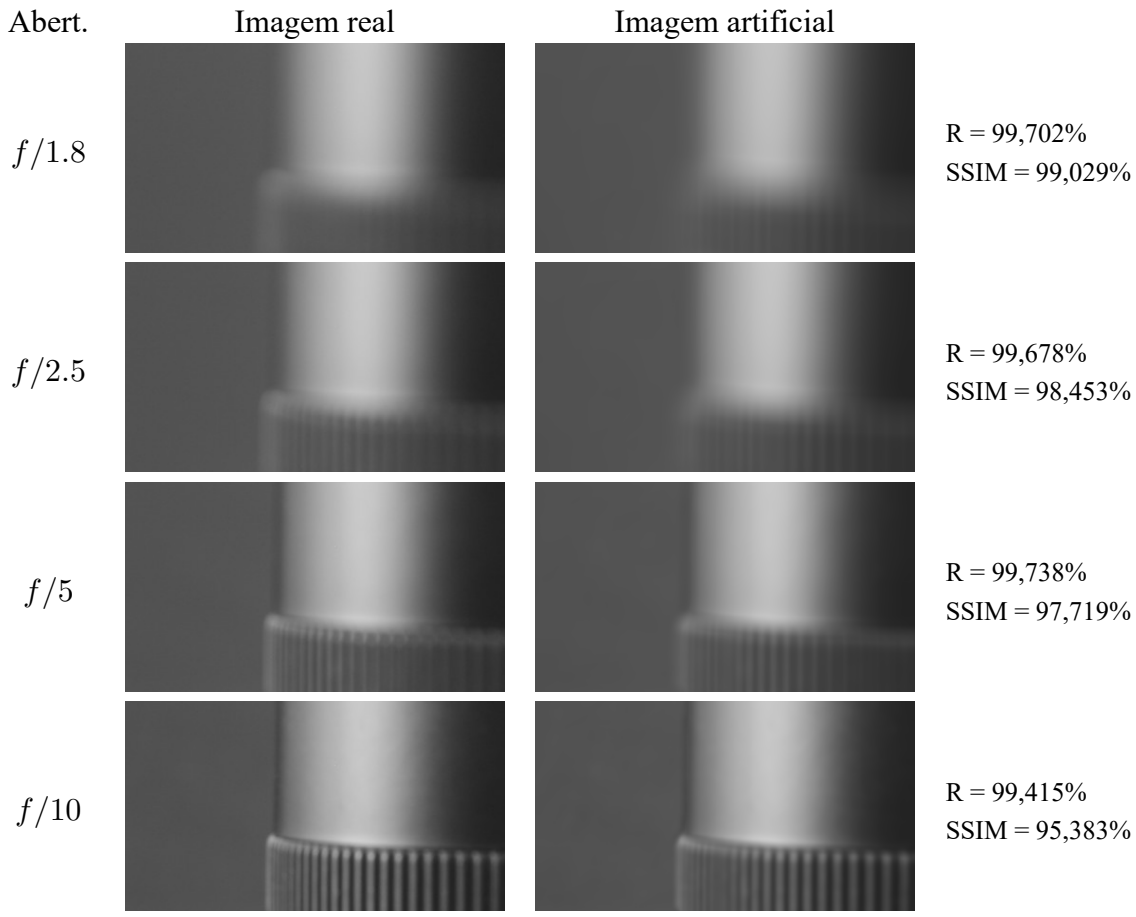
Como é apresentado na Figura 47, os valores de $R$ e $SSIM$ são altos para todas as imagens, sendo acima de 99% para a correlação e acima de 97% para o $SSIM$. Esses resultados são satisfatórios, mas poderiam ser melhorados caso ainda houvesse uma preocupação em equalizar a luminosidade das fotos. Há uma variação da luminosidade geral e não uniforme[1] da foto e que não foi o mesmo caso na foto de $f/22$ usada de referência e aplicada a todas. A medição $SSIM$ é mais sensível a tal variação do que $R$, já que a forma de cálculo descarta o nível médio do sinal, o que tem relação local com a luminosidade. A forma mais adequada de equilibrar as luminosidades, por não ser uniforme, é pela equalização de regiões através da avaliação dos histogramas.

Além da avaliação global, um segundo grupo de comparações foi realizado em janelas da imagem, destacando apenas pedaços em que a transição do foco é evidente. Isso faz com que

[1] A não uniformidade é devido à iluminação da cena. Por ser uma foto real e que usa iluminação natural (solar), não é possível controlar totalmente a incidência de luz para cada objeto ao longo da sequência de fotos.

dentro da imagem, o total de pixels relevantes nessa análise seja proporcionalmente maior, se tornando mais relevante na avaliação. A escolha de cada janela leva em conta a presença de um elemento diferente da composição da imagem, de modo a avaliar a relação do desfoque (círculo de confusão) com a distância do objeto. Os resultados da primeira janela estão na Figura 48 e da segunda na Figura 49.

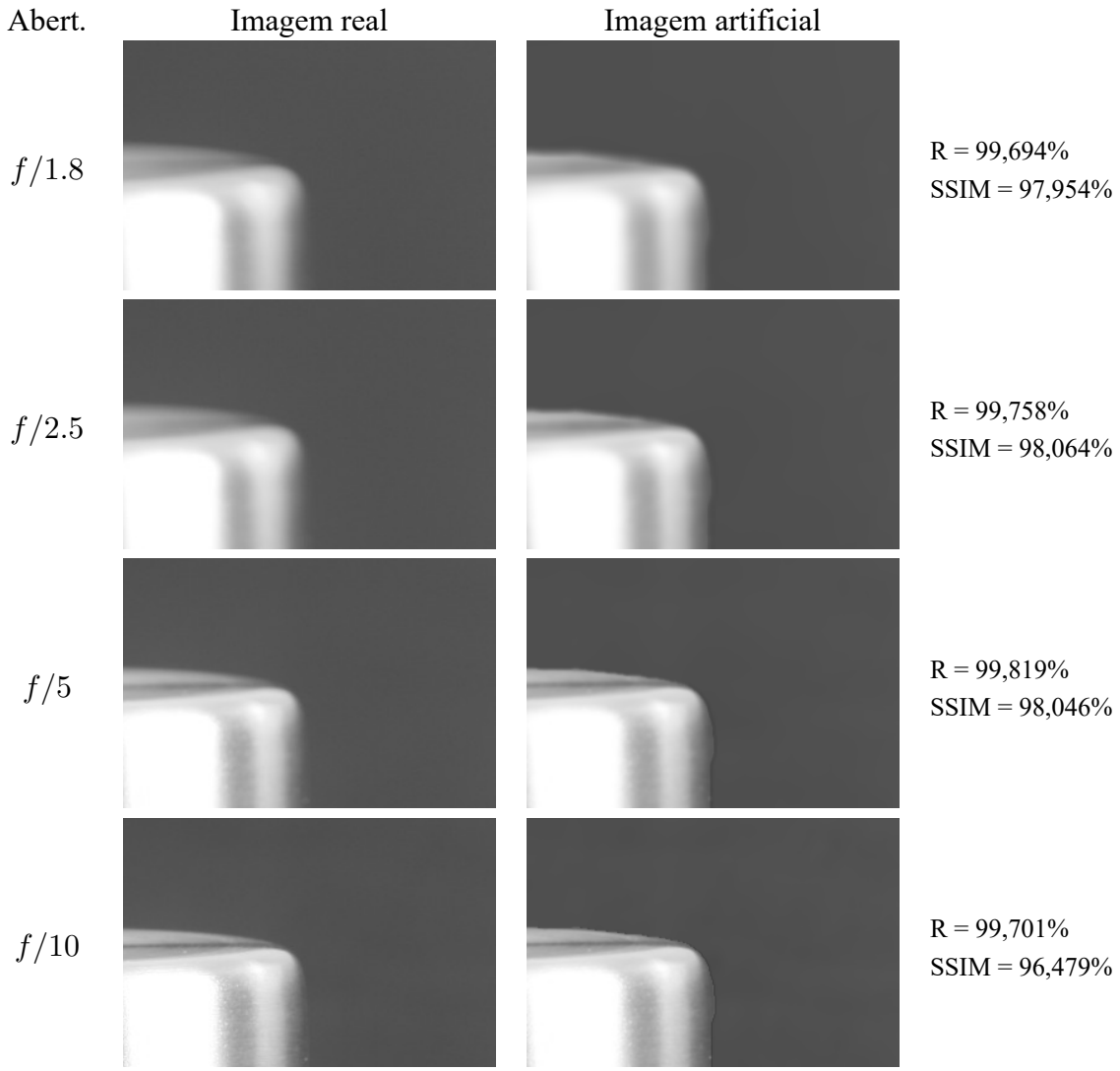
Figura 48 – Comparação entre imagens reais e imagens geradas artificialmente para um pedaço da imagem total




Fonte: Autor, 2015

Notas: Comparação entre imagens reais (lado esquerdo) e imagens geradas artificialmente (lado direito) para um pequeno pedaço da imagem total, assim como os resultados individuais de correlação matemática $R$ e índice de similaridade estrutural (SSIM).

Figura 49 – Comparação entre imagens reais e imagens geradas artificialmente para um segundo pedaço da imagem total




Fonte: Autor, 2015

Notas: Comparação entre imagens reais (lado esquerdo) e imagens geradas artificialmente (lado direito) para um segundo pequeno pedaço da imagem total, assim como os resultados individuais de correlação matemática $R$ e índice de similaridade estrutural (SSIM).

Como pode ser visto pelos valores resultantes, novamente há uma relação satisfatória entre as imagens, obtendo, localmente, valores acima de 99% para $R$ e valores acima de 95% para $SSIM$. Mesmo para uma análise subjetiva (visual) é possível determinar que há uma grande similaridade entre as imagens reais e artificiais. No entanto, é válido destacar novamente que o intuito não é a criação de imagens perfeitamente similares por si só, mas a construção de imagens para avaliação de um algoritmo de segmentação que não possui dependência direta com a estrutura individual do desfoque de uma imagem. Em outras palavras, o algoritmo a que se destina o uso dessas imagens não depende da distribuição de energia do desfoque nas bordas, como existem casos disponíveis na literatura, mas comparação entre foco e desfoco de duas imagens, diminuindo a necessidade de maiores graus de confidencialidade das simulações. Assim, a conclusão é que as imagens obtidas via geração, na forma atual, são válidos para análise

do algoritmo de segmentação proposto. Ressalvas são válidas, porém, caso deseje-se estender o uso da técnica de criação para os demais métodos disponíveis na literatura para segmentação, caso não sejam propostas melhorias na forma de criação.

## 4.3 AVALIAÇÃO DO NOVO MÉTODO DE SEGMENTAÇÃO USANDO IMAGENS ARTIFICIAIS

Adicionalmente aos resultados descritos com o uso de imagens reais, e conforme já elucidado, foram executados processos de segmentação com o algoritmo em imagens criadas artificialmente. A busca nas análises nesse ponto passa a ter um objetivo adicional de otimização do algoritmo, em virtude da nova capacidade de emprego de processos estatísticos, consequência direta da volume maior de imagens.

O algoritmo em questão possui diversas variáveis internas que permitem a otimização, mas três aspectos são de maior interesse e podem, a princípio, serem de maior relevância ou terem contribuição significativa no desempenho do mesmo. A saber:

- Dimensão do núcleo (ou *kernel*) do filtro passa baixa empregado no espalhamento dos picos;
- Dimensão do DOF para cada uma das imagens do par de cenas, que dentro das possibilidades óticas, é representado na prática como a abertura ótica empregada em cada foto;
- Aplicação de operações morfológicas realizadas no final do algoritmo. As operações englobam, na proposta inicial, as operações de dilatação, erosão, preenchimento de buracos, remoção de pequenos fragmentos e, por fim, novamente uma dilatação. Cada operação possui possibilidades de ajustes, tais como a dimensão do padrão de operação (dilatação e erosão) e dimensão limite para consideração de buracos ou de pequenos objetos para descarte.

Além dos itens listados acima, pode-se avaliar a dimensão da imagem em si. Ao reduzir o tamanho da imagem analisada, apesar da redução de capacidade de informação representada na imagem (perda de resolução), há simultaneamente o declínio de pixels a serem analisados, o que resultado numa melhoria do desempenho de tempo de execução do algoritmo. Apesar dessa melhoria, a prioridade da análise é primeiramente garantir uma qualidade de segmentação mínima, para apenas num segundo momento, garantida a primeira premissa, então buscar a otimização do tempo de execução.

Da listagem de possibilidades previstas, a de impacto mais incerto são ajustes do resultado via operações morfológicas, visto que, diferente das demais, englobam mais de um processo, e consequentemente multiplicando as possibilidades de análise. A própria variação do sequenciamento das operações morfológicas, por não serem lineares no estrito senso matemá-

tico, conduzem a diferentes resultados. Ainda assim, vale a pena discorrer sobre as motivações de cada operação:

- Dilatação seguida de erosão: Como é justificado a seguir, o algoritmo possui uma maior tendencia a identificar as bordas dos objetos. No entanto, muitas vezes essas bordas, ou contornos, ficam abertos, impedindo uma operação de fechamento de buracos. Para possibilitar essa etapa então é feita a sequência das duas operações morfológicas, que é capaz, em muitos casos, de fechar pequenas aberturas nos cantos, sem alterar por demais o restante do conteúdo segmentado;

- Fechamento de buracos: Existe uma tendência do algoritmo de segmentar as bordas dos objetos em virtude dela apresentar a maior variação de conteúdo entre detalhe e não-detalhe. As regiões internas de objetos podem ser lisas ou com pouquíssima informação de alta frequência, o que possui menor diferença entre as imagens de alta e baixa profundidade de campo, sendo incapaz, portanto, o algoritmo de acrescentá-la naturalmente a segmentação. A operação morfológica de fechamento busca cobrir essa deficiência;

- Remoção de pequenos objetos: Essa operação pode ser a mais controversa, visto que não possui relação direta com o algoritmo. Ela busca uma melhoria dos resultados, partindo do pressuposto de que os objetos segmentados serão sempre maiores do que um tamanho limite, no qual se enquadram demais objetos fora de interesse ou, mais comumente, ruídos da segmentação. Esse conceito é válido, desde que haja cuidados com a aplicação do método. Como as características óticas das lentes são bem conhecidas, dada uma distância especifica de operação, é possível estimar o tamanho que tais objetos são representados sobre o sensor fotográfico e, portanto, na cena;

- Dilatação final: A razão dessa operação já foi explicada anteriormente. O objeto é criar um halo ou auréola de valores falso-positivos em torno da área segmentada efetiva, visto que um resultado assim é vantajoso sobre o caso oposto (falso-negativos), em que haveria a perda de informação dos contornos do objeto.

Das operações listadas acima, pode-se enumerar ao menos cinco parâmetros (um para cada operação) que afetam seus desempenhos, o que gera uma grande complexidade de combinações. Aliando tal complexidade à menor importância ao cerne do algoritmo, a avaliação dos processos morfológicos foi descartada em detrimento de parâmetros mais centrais do método.

Em face das razões apresentadas, a busca de parâmetros que melhorem a qualidade da segmentação, ou seja, aumentem os valores de acurácia e concomitantemente mantenham os valores de proporção de falso negativos em taxas mínimas, será realizada no domínio da abertura ótica e do tamanho do *kernel* do filtro passa-baixa. Adicionalmente, as operações morfológicas que representam a última etapa do algoritmo foram descartadas por completo, para que as

medições quantitativas resultantes sejam uma função direta dos elementos do domínio. Assim, o algoritmo de segmentação apresentado no Apêndice A é aplicado sem o seu último bloco de operações.

## 4.4 EXPERIMENTOS E RESULTADOS

A segmentação foi aplicada, para cada imagem, em 240 combinações de aberturas óticas e de dimensão de *kernel* do filtro passa-baixa, fruto da combinação dos seguintes espectros de valores de cada variável:

$$A = \left\{ f/1.4, f/2, f/2.8, f/4, f/5.6, f/8, f/11, f/16 \right\} ,$$
$$K = \left\{ k \mid k, n \in \mathbb{N} \wedge k = 2.n \wedge 1 \leq n \leq 30 \right\} .$$

Para todas as imagens, o par de contraste de alta profundidade de campo foi a imagem abertura ótica $f/22$. Todas as imagens criadas possuem dimensão de 1024 por 768 pixels.

No total foram analisadas 10 imagens geradas artificialmente, realizando a segmentação do método de pares de imagens e então mensurando as grandezas já descritas e padronizadas no trabalho: acurácia, proporção de falsos negativos e tempo de execução. Para as medidas de tempo, além do padrão, em que todo o algoritmo de segmentação é contemplado dentro dos limites que definem o intervalo de tempo que mede a execução, adicionalmente foi realizada a medida de tempo reduzida, em que o alinhamento de imagens via correlação máxima de matrizes (imagens) é desconsiderado. A motivação está no fato dessa operação exigir um alto tempo de execução e ainda poder ser desnecessária, conforme a aplicação do método.

Para cada uma das configurações possíveis foi realizada a medição da média e desvio padrão dos resultados de cada imagem. Os resultados médios estão apresentados na Figura 50. Como pode ser visto, o pico de acurácia média $\mu$ foi de 96,36%, com valor de erro padrão da média de 0,0173, calculado por

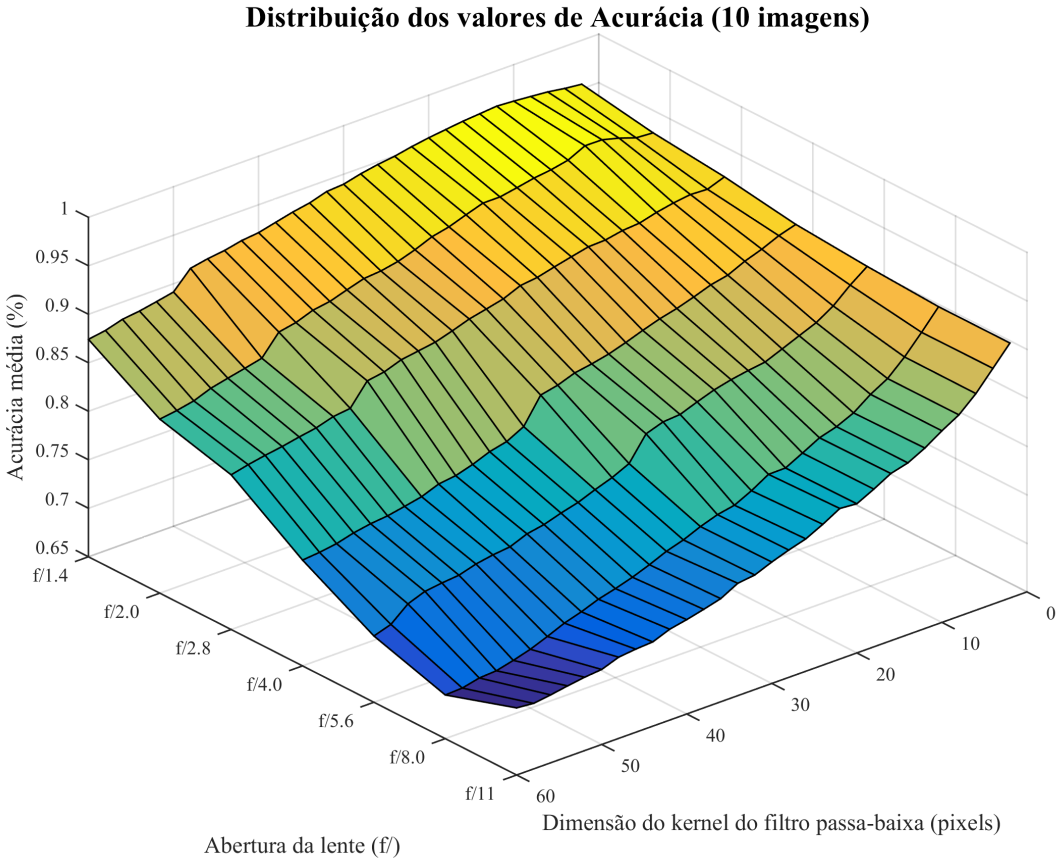
$$E = \frac{\sigma}{\sqrt{\mu}} ,$$

onde $\mu$ é a média das medições de acurácia e $\sigma$ é o desvio padrão das mesma medições. Dentro do domínio pré-estabelecido, esse valor é obtido para uma abertura ótica de $f/1.4$, obviamente junto à imagem feita com abertura de $f/22$, que foi usada para todas como o par de alta profundidade campo. Esse resultado era esperado por tal par representar o conjunto de imagens que possui maior disparidade de profundidade de campo. No entanto, ao considerar um limite de acurácia mínima de 90%, imagens com $f/11$ ainda se enquadram nesse grupo, o que corrobora com um resultado previamente já enunciado, de que a técnica não precisa de imagens com baixíssima profundidade de campo para obter resultados expressivos, o que permite que o aparato de captação das imagens empregado seja, em principio, menos oneroso, pelas suas aberturas óticas de trabalho serem menores.

O segundo parâmetro de controle, a dimensão do *kernel* do filtro passa-baixa, possui uma relação com a acurácia que depende da abertura usada. É possível notar pelo gráfico que a

curva que descreve o desempenho da média da acurácia cai de forma mais rápida para pequenas aberturas óticas e para os casos de grandes aberturas ocorre um pico intermediário com uma queda mais suave à medida que o tamanho do *kernel* cresce. A primeira conclusão é que com valores de profundidade de campo menores, o método é mais robusto, tendo menor influencia o valor do *kernel*. A segunda é que o pico de desempenho da acurácia de 96,36% ocorre para uma tamanho de filtro de 12 pixels. Esse resultado indica uma aparente relação com as medidas dos objetos em cena. Enquanto a dimensão do filtro é próxima ao próprio tamanho dos objetos e compatível com o traçado do contorno dos mesmos, os resultados são otimizados. A medida que crescem, e consequentemente passam a ter uma dimensão maior do que os próprios objetos, a capacidade de segmentação do processo passa a ser reduzida.

Figura 50 – Medição da acurácia média pelo algoritmo proposto aplicado a imagens geradas artificialmente



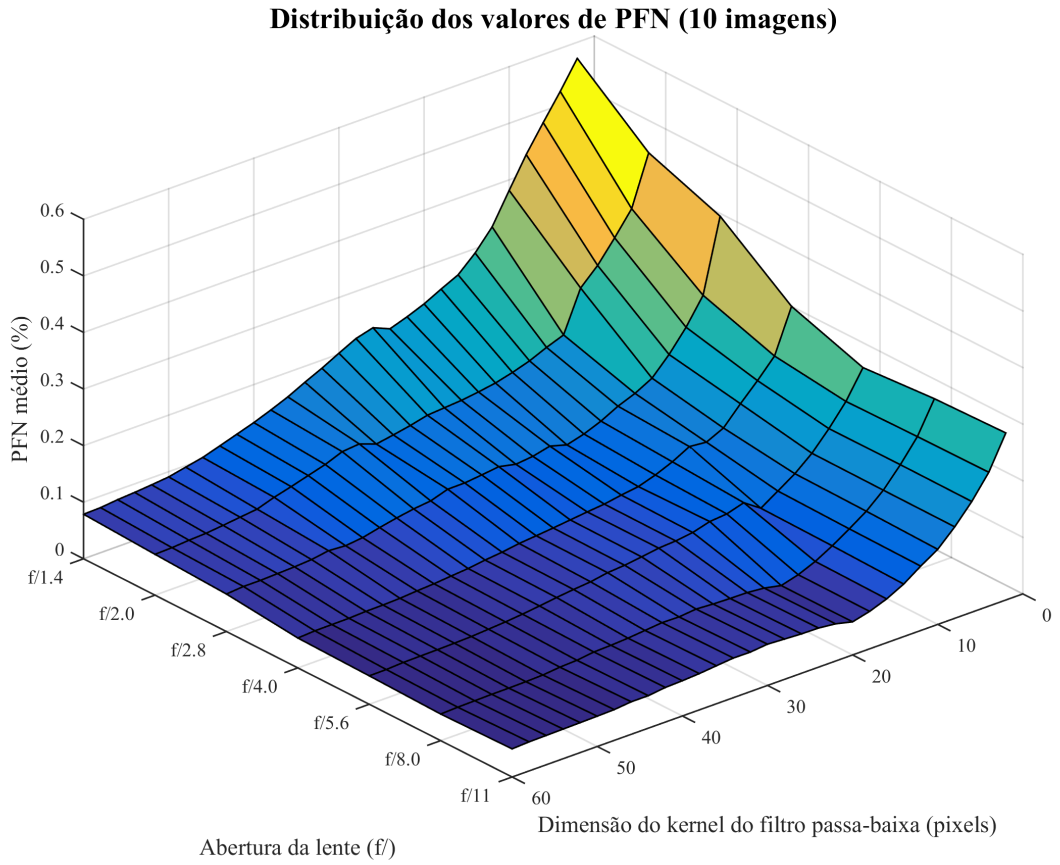

Fonte: Autor, 2015

Notas: Medição da acurácia média para variações de parâmetros do algoritmo proposto aplicado a 10 imagens geradas artificialmente.

Analogamente, para a medição da média da proporção de falsos negativos, os resultados são apresentados na Figura 51. Analisando de forma objetiva, ou seja, pelos pontos em que a medida paralela da acurácia média possui altas taxas, nota-se que os pontos de piores PFN médio (valores mais altos) ocorrem justamente na região que tem a acurácia média maximizada. A razão para tal é que quanto maior o valor da acurácia, mais próxima a segmentação obtida fica

do *ground truth*. A segmentação passa a ter um contorno muito próximo do real, permitindo mais facilmente que o traçado limite da segmentação fique dentro da região de *ground truth*, incrementando a contagem de falsos-negativos.

Figura 51 – Medição de PFN média pelo algoritmo proposto aplicado a imagens geradas artificialmente




Fonte: Autor, 2015

Notas: Medição da PFN (proporção de falsos negativos) média para variações de parâmetros do algoritmo proposto aplicado a 10 imagens geradas artificialmente.

O pico medido de PFN médio possui valor igual a 57,16% (e erro padrão da média igual a 14,37%). Já o valor correspondente ao melhor desempenho de acurácia é de 32,77$\pm$8,28 %. O valor, mesmo ainda considerando o limite do erro padrão, está abaixo dos 50%, e aliado à alta taxa de acurácia, implica num resultado geral de segmentação ótimo. É possível reforçar o resultado fazendo o cálculo da taxa de pixels nos resultados que são falsos negativos e também dos que são falsos positivos: 1,19% e 2,45%, respectivamente.

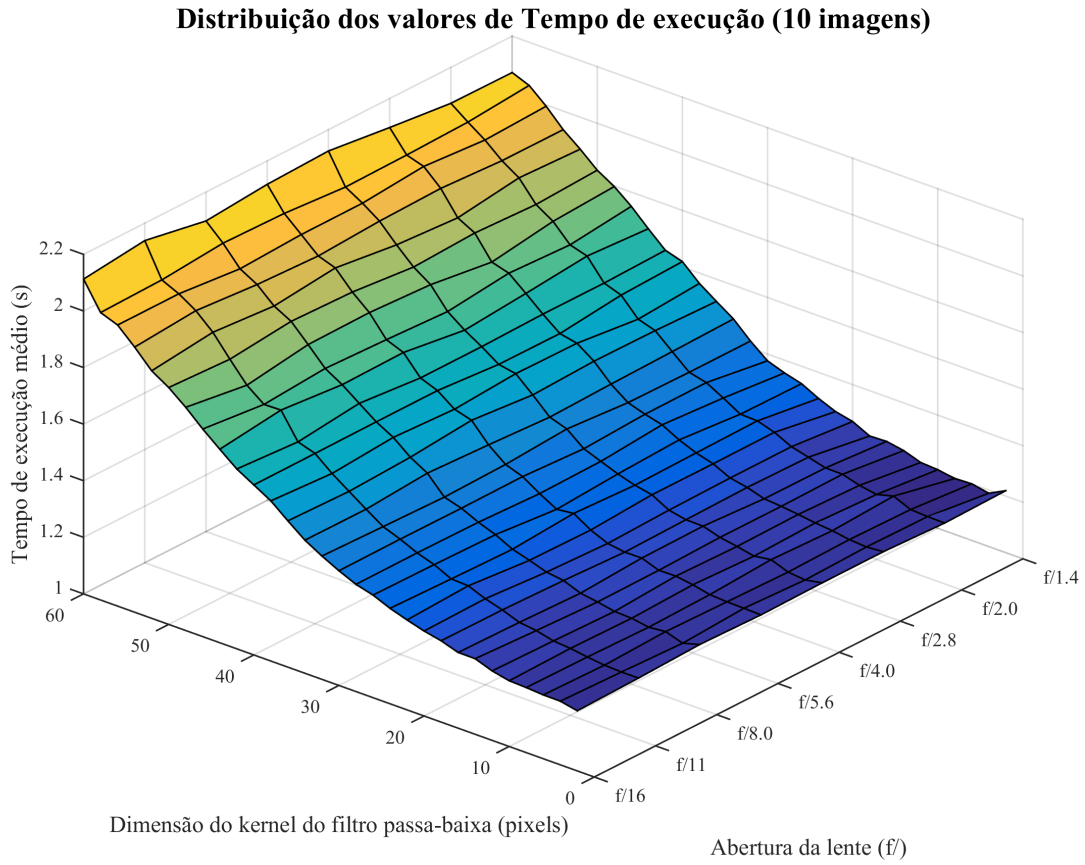
Apesar da região de maior importância segundo a medição de acurácia possuir os valores máximos medidos de PFN média, os menos interessantes, ainda assim, como pode ser visto pelo exemplo citado acima, os valores não são de todo ruins, e nota-se uma queda abrupta nos valores de PFN, o que indica que o mesmo se estabiliza rapidamente para taxas próximas à zero. O sentido de queda de PFN está para os valores do tamanho do *kernel*. O motivo é o mesmo já apresentado: a medida que o tamanho do *kernel* do filtro passa-baixa cresce, perde

a capacidade de resolução, tendendo a extrapolar as bordas da segmentação, ou seja, englobar conteúdo periférico do objeto de interesse. A quantidade de informação englobada é maior para filtros passa-baixa maiores.

Como o objeto desse método de segmentação é aplicado ao processamento de tempo real, ainda é necessário avaliar o tempo de execução do mesmo. Analogamente, o tempo de execução foi calculado em média, mas com duas marcações diferentes, como feito anteriormente, sendo uma para o processo completo e outra em que a etapa de alinhamento das imagens por meio de maximização de correlação matemática foi desconsiderado. Os resultados estão apresentados respectivamente nas Figuras 52 e 53. Existem duas diferenças entre os resultados agora obtidos e os demais apresentados anteriormente. A primeira é que as operações morfológicas foram suprimidas pelas razões já destacadas e, portanto, tais operações não estão englobadas no tempo medido total. A segunda questão é o uso de um *hardware* com maior capacidade[2] de processamento para execução do método. Vale destacar que todas as imagens usadas na análise possuem dimensões de largura e altura respectivamente de 1024 e 768 pixels, o que é considerada uma imagem de grande porte e que precisa ser levada em consideração na avaliação do tempo de processamento.

[2]Computador PC com sistema operacional Windows 64 bits, processador Intel® Core™ i7-3632QM @ 2.20 GHz e 8 gb de memória RAM DDR2. A versão de Matlab usada foi a R2015a.

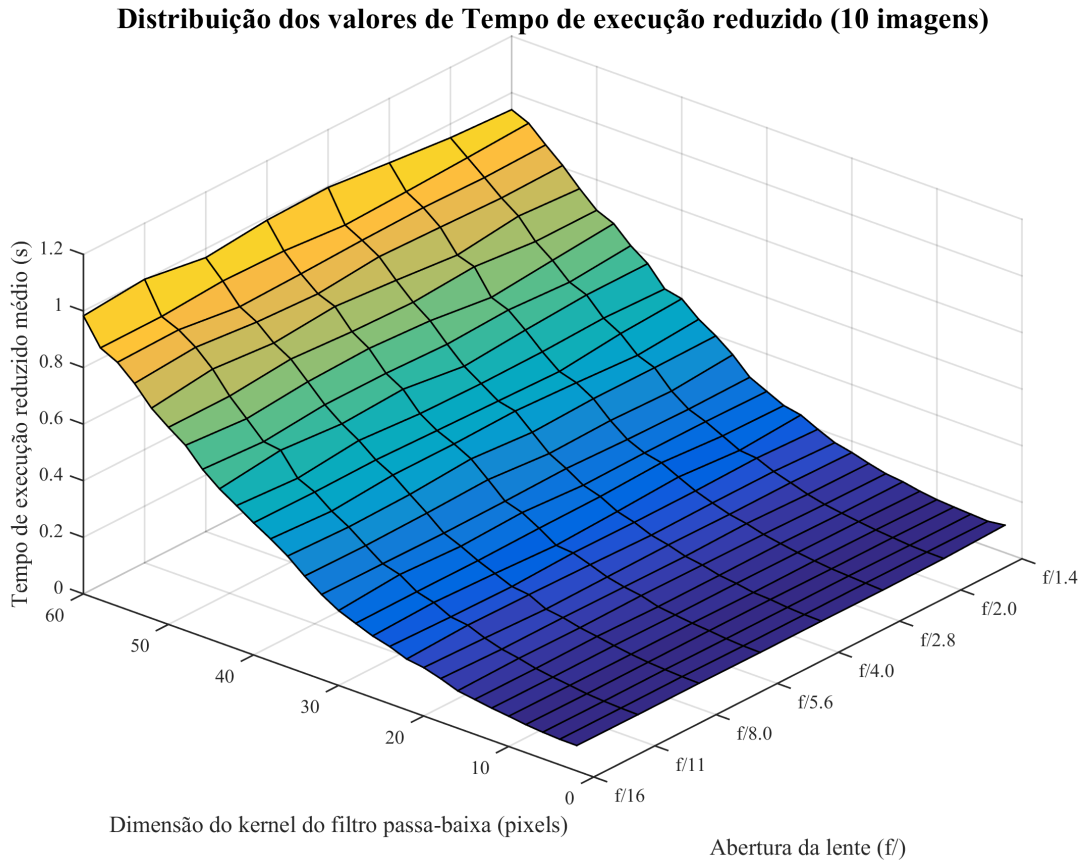
Figura 52 – Medição do tempo de execução médio pelo algoritmo proposto aplicado a imagens geradas artificialmente




Fonte: Autor, 2015

Medição do tempo de execução médio para variações de parâmetros do algoritmo proposto aplicado a 10 imagens geradas artificialmente.

Figura 53 – Medição do tempo de execução reduzido médio pelo algoritmo proposto aplicado a imagens geradas artificialmente




Fonte: Autor, 2015

Medição do tempo de execução reduzidos (sem operação de correlação cruzada) médio para variações de parâmetros do algoritmo proposto aplicado a 10 imagens geradas artificialmente.

A análise dos gráficos indica claramente que possuem padrões semelhantes, e que a remoção da correção foi responsável apenas por causar um deslocamento ao longo do eixo da cota. A média dessa redução de tempo foi de $1{,}1298 \pm 0{,}0005$ segundos, o que implica numa grande constância do desempenho do processo de correlação, em face do baixíssimo erro padrão da média. Para o caso estudado de imagens geradas artificialmente, não existe um efeito de paralaxe por deslocamento da posição da lente, então a etapa de correlação das imagens passa a ser totalmente desnecessária. Não obstante a isso, a análise deve ser apresentada aos pares (com e sem correlação) por não sabermos de antemão se a operação será ou não necessária.

O pior desempenho de tempo de execução ocorre para valores de *kernel* altos, enquanto a abertura ótica representa um parâmetro praticamente indiferente para a velocidade de execução do algoritmo. A razão desse incremento de tempo de execução em função do tamanho do *kernel* é que as operações de convolução entre as matrizes da imagem e do filtro tornam-se mais complexas com o aumento da área do segundo (lembrando que a área do primeiro é constante), pois cresce o número de elementos considerados nas operações básicas. Esse resultado é condizente com a análise de complexidade de tempo do algoritmo já realizada.

Como o comportamental das curvas de tempo de execução indicam um crescimento assintótico em função do tamanho do *kernel*, fica automaticamente descartada a avaliação de valor máximo. O valor mínimo passa a ser também óbvio. O que interessa nesse momento é indicar para aqueles pontos já apresentados anteriormente que tiveram melhor desempenho de acurácia média, os seus tempos de execução. Assim, o pico de desempenho de acurácia média de 96,36$\pm$0,0173%, localizado para abertura ótica $f/1.4$ e dimensão do *kernel* de 12 pixels, teve um tempo de execução 1,2109$\pm$0,0288 segundos para a execução completa e 0,1093$\pm$0,0023 segundos para a execução reduzida (sem alinhamento).

# 5 CONCLUSÃO

A abordagem proposta nesta dissertação sobre o uso de duas imagens para a segmentação de cenas com baixa profundidade de campo apresentou resultados promissores tanto em termos da qualidade da segmentação quanto do tempo computacional necessário para processamento. A crítica negativa a esta abordagem está obviamente na necessidade de uso de duas imagens. Para corroborar com os resultados positivos obtidos, uma análise detalhada do algoritmo sob o ponto de vista de processamento de sinais foi apresentada, o que ao final serviu de confirmação adicional aos resultados práticos descritos. A título de exemplificação da ordem de grandeza, para muitos dos casos testados, a segmentação teve acurácia acima de 95%.

As medições de desempenho empíricas foram feitas em dois momentos separados. Num primeiro caso, imagens reais com poucas amostras, por indisponibilidade de um acervo maior, foram empregadas. Apesar do resultado limitado do ponto de vista estatístico, mas não menos interessante, vislumbrou-se a possibilidade de obtenção das imagens requeridas para complementar tais medições. A solução adotada, em face das dificuldades de criação de fotografias reais com as condições necessárias (cenas com variação de profundidade de campo apenas entre si, capacidade de variações de cenas e criação de imagens *ground truth*), foi desenvolver um algoritmo capaz de gerar imagens semelhantes às reais, usando os conceitos teóricos da ótica descritos ao longo do trabalho. O método de geração de cenas artificias usando componentes reais foi verificado com a comparação qualitativa e quantitativa em relação à cena real equivalente. O sucesso na comparação obteve valores de relação acima de 97% para as métricas adotadas, o que permite assegurar que há grande semelhança dos resultados, e que a prática de criar imagens artificias pode ser válida para aplicar aos métodos de segmentação.

Com a disponibilidade de uma maior quantidade de imagens, o algoritmo proposto de segmentação foi submetido a outros testes, se valendo ainda da pesquisa para otimização das variáveis de entrada: otimização de uso de abertura ótica e otimização de dimensão do *kernel* do filtro passa-baixa usado em uma de suas etapas. Como era esperado, os melhores desempenhos foram para aberturas óticas grandes e dimensão do *kernel* próxima ao tamanho dos objetos em cena.

Ainda com relação aos resultados, para os casos na região ótima, atingiu-se valores altos de acurácia (em alguns casos acima de 95%), aliados a baixos PFN e baixos valores de tempo de execução (abaixo de 1,5 segundos), sendo esse último resultado especialmente interessante para a proposta inicial de aplicação do método para segmentação em tempo real. O uso de uma amostra maior de imagens permite o uso de ferramentas estatísticas que potencialmente devem revelar dados igualmente promissores. O principal ponto a ser visto é que os erros padrões médios observados são muito baixos, descrevendo então uma consistência nos resultados. Vale notar que apesar desse fato, as imagens possuem uma certa semelhança, que não permite ainda generalizar tal conclusão para quaisquer imagens.

No entanto, os resultados obtidos com a abordagem proposta superam o método usado

atualmente como referência na literatura afim (NEVEROVA; KONIK, 2012) tanto no desempenho de segmentação, como também, e principalmente, no tempo de execução. A crítica positiva com relação a essa referência e demais algoritmos disponíveis na literatura é o uso de uma única imagem apenas. Essa vantagem pode deixar de ser relevante conforme a aplicação, exigência de alta assertividade ou requerimento de baixos de tempos computacionais de resposta.

Há margem para uma possível melhoria de capacidade de segmentação e de redução do tempo de execução à medida que o algoritmo seja calibrado ou ajustado para ser especialista em determinadas aplicações e situações. Um dos casos mais diretos, e já medidos, é com a possibilidade de descartar completamente a necessidade de alinhamento entre as imagens, o que faria o tempo de execução reduzirem aproximadamente 1,1298 segundos. Além disso, a proposta em si de uso de duas imagens permite que outros algoritmos sejam desenvolvidos, podendo esses terem melhor desempenho, seja em tempo de execução, acurácia ou ambas métricas.

## 5.1 TRABALHOS FUTUROS

Os seguintes possibilidades de extensão deste trabalho mostram-se importantes futuramente.

### 5.1.1 Uso separado de cores

O algoritmo proposto emprega as imagens em termos de sua luminância apenas. É possível pensar que, paralelamente, poder-se-ia fazer o processo semelhante em diversos espaços, como por cores individualizadas, e então unindo os mesmos, empregando uma métrica pré-definida, que por exemplo, poderia ser o operador mínimo (equivalente ao operador lógico E) ou máximo (equivalente ao operador lógico OU) entre as diferentes camadas. Espera-se disso uma melhoria na acurácia e na robustez da abordagem, com talvez uma possível piora (aumento) no tempo de execução.

### 5.1.2 Uso de ordens maiores de derivadas

Para a abordagem descrita de algoritmo de segmentação, a forma de detecção de altas frequências foi feita via operador gradiente. No entanto, esse operador não é único. É possível usar maiores ordens de derivadas, tal como o operador Laplaciano $L$ para uma imagem $I$:

$$L\left(x,y\right) = \nabla^2 I = \nabla\left(\nabla I\right) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}\,.$$

O operador Laplaciano também pode realizar, numa análise pragmática, um filtro passa-alta, assim como o operador gradiente. A diferença é que se pensarmos no sinal como uma série (seja de Taylor ou Fourier), enquanto o gradiente remove o nível DC (constante), o operador Laplaciano irá remover um nível a mais do sinal, ainda nas baixas frequências. Conforme se eleva o grau da derivada, esse processo de remoção cresce.

### 5.1.3 Combinação com outras técnicas

O uso de duas câmeras para obtenção de imagens não é totalmente novidade quando se pensa em termos de imagens estéreo. No entanto, esse principio, que emprega a paralaxe das imagens para o cálculos, exige que ambas as imagens sejam construídas de forma semelhante, para comparação. A proposta adicional aqui é que se empregue uma terceira câmera com baixa profundidade de campo, ao centro. Isso permitiria uma analise conjunta do efeito estéreo e da baixa profundidade de campo, possivelmente otimizando resultados.

### 5.1.4 Construção de bancos de imagens

A construção de bancos de imagens reais conforme a necessidade do algoritmo pode ser extremamente complexa, inviabilizando a tarefa em si, ou tornando-a pouco eficiente. Uma maneira diferente de abordar a questão é com a construção de um aparato de 2 câmeras, capaz de capturar a cena e processá-la, já usando o algoritmo, em tempo real ou próximo a isso (pequenos atrasos). O resultado da segmentação deve então ser apresentado a um ser humano para que julgue os acertos ou erros. A avaliação logicamente não é mais na acurácia em termos de pixel em cada cena, mas na capacidade de segmentação de um objeto dentro de um *frame* do vídeo. A métrica então é o número de *frames* em que houve acerto frente ao total de *frames* medidos.

# REFERÊNCIAS BIBLIOGRÁFICAS


ACHANTA, R. et al. Frequency-tuned salient region detection. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2009, Miami. **Proceedings...** Miami: IEEE, 2009. p. 1597–1604.

ACHANTA, R.; SUSSTRUNK, S. Saliency detection using maximum symmetric surround. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 17., 2010, Hong Kong. **Proceedings...** Hong Kong: IEEE, 2010. p. 2653–2656.

AHN, S.; CHONG, J. Segmenting a noisy low-depth-of-field image using adaptive second-order statistics. **IEEE Signal Processing Letters**, v. 22, n. 3, p. 275–278, mar. 2015.

ANWER, R. M.; VÁZQUEZ, D.; LÓPEZ, A. M. Opponent colors for human detection. In: VITRIÀ, J.; SANCHES, J. M.; HERNÁNDEZ, M. (Ed.). **Pattern Recognition and Image Analysis: 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings**. Berlin: Springer, 2011. p. 363–370.

BARSKY, B. A.; KOSLOFF, T. J. Algorithms for rendering depth of field effects in computer graphics. In: WSEAS INTERNATIONAL CONFERENCE ON COMPUTERS, 12., 2008, Heraklion, Greece. **Proceedings...** Stevens Point, WI: World Scientific and Engineering Academy and Society (WSEAS), 2008. (ICCOMP'08), p. 999–1010.

BASS, M. et al. **Handbook of Optics: Geometrical and Physical Optics, Polarized Light, Components and Instruments**. 3rd. ed. New York: McGraw-Hill Education, 2009.

BASS, M. et al. **Handbook of Optics: Devices, Measurements, and Properties**. 2nd. ed. New York: McGraw-Hill Professional, 1994.

BATT, A.; DOBRO, C.; STEEN, J. **Camera & Craft: Learning the Technical Art of Digital Photography: (The Digital Imaging Masters Series)**. Massachusetts: Focal, 2014.

Camera and Imaging Products Association (CIPA). **CIPA Statistical results & Outlook on shipment**. 2014. Online. Disponível em: <http://www.cipa.jp/stats/dc_e.html>. Acesso em: 3 nov. 2014.

CLAXTON, C. D.; STAUNTON, R. C. Measurement of the point-spread function of a noisy imaging system. **J. Opt. Soc. Am. A**, v. 25, n. 1, p. 159–170, jan. 2008.

DINIZ, P.; SILVA, E. da; NETTO, S. **Processamento Digital de Sinais: Projeto e Análise de Sistemas**. 2. ed. Porto Alegre: Bookman, 2014.

European New Car Assessment Programme (Euro NCAP). **2020 Roadmap**. 2014. Disponível em: <http://www.euroncap.com/files/Euro-NCAP-2020-Roadmap---June-2014-2---0-e11c0984-af94-420e-9d63-63edc8538745.pdf>. Acesso em: 24 nov. 2014.

FISHER, B.; PRICE, S. **Edges: The Canny Edge Detector**. 1996. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm>. Acesso em: 20 nov. 2014.

FUKUNAGA, K.; HOSTETLER, L. The estimation of the gradient of a density function, with applications in pattern recognition. **IEEE Transactions on Information Theory**, v. 21, n. 1, p. 32–40, jan. 1975.

GOKTURK, S.; YALCIN, H.; BAMJI, C. A time-of-flight depth sensor - system description, issues and solutions. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOP (CVPRW), 2004, Washington. **Proceedings...** Washington: IEEE, 2004. p. 35–35.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 2nd. ed. New Jersey: Prentice Hall, 2002.

GRAF, F.; KRIEGEL, H.-P.; WEILER, M. Robust segmentation of relevant regions in low depth of field images. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 18., 2011, Brussels. **Proceedings...** Brussels: IEEE, 2011. p. 2861–2864.

GUILLET, F. J.; HAMILTON, H. J. **Quality Measures in Data Mining**. Berlin: Springer, 2007.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentals of Physics**. 9th. ed. New Jersey: Wiley, 2010.

HECHT, E. **Optics**. 4th. ed. San Francisco: Addison Wesley, 2002.

HESS, A.; MCLERNON, B. **Photography Techniques: Digital Field Guide**. Indiana: Wiley, 2012.

HIRSCH, R. **Light and Lens: Photography in the Digital Age**. 2nd. ed. Burlington: Focal, 2012.

JACOBSON, R. et al. **Manual of Photography (Media Manual)**. 9th. ed. Oxford: Focal, 2000.

JAVARAN, T. A.; HASSANPOUR, H.; ABOLGHASEMI, V. Automatic estimation and segmentation of partial blur in natural images. **The Visual Computer**, Springer, p. 1–11, out. 2015.

KAO, S.-M. et al. Extraction of the focused object in an image by filtering out the defocused background. In: INTERNATIONAL SYMPOSIUM ON SPEECH, IMAGE PROCESSING AND NEURAL NETWORKS (ISSIPNN), 1994, Hong Kong. **Proceedings...** Hong Kong: IEEE, 1994. v. 1, p. 53–56.

KHAN, R.; DINET, E.; KONIK, H. Visual attention: Effects of blur. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 18., 2011, Brussels. **Proceedings...** Brussels: IEEE, 2011. p. 3289–3292.

KIM, C. Segmenting a low-depth-of-field image using morphological filters and region merging. **IEEE Transactions on Image Processing**, v. 14, n. 10, p. 1503–1511, out. 2005.

KIM, C. et al. Fast extraction of objects of interest from images with low depth of field. **ETRI journal**, Electronics and Telecommunications Research Institute, v. 29, n. 3, p. 353–362, jun. 2007.

KIM, G.; EOM, J.; PARK, Y. Investigation on the occurrence of mutual interference between pulsed terrestrial LIDAR scanners. In: IEEE INTELLIGENT VEHICLES SYMPOSIUM, 4., 2015, Seoul. **Proceedings...** Seoul: IEEE, 2015. p. 437–442.

KINGSLAKE, R. **Optical System Design**. New York: Academic, 1983.

KNAPP-CORDES, M.; MCKEEMAN, B. **Improvements to tic and toc Functions for Measuring Absolute Elapsed Time Performance in MATLAB**. 2011. Disponível em: <http://www.mathworks.com/tagteam/68600_91934v00_TicToc.pdf>. Acesso em: 30 set. 2015.

KUPERMAN, V. **Magnetic Resonance Imaging: Physical Principles and Applications**. California: Academic, 2000.

KUTHIRUMMAL, S. et al. Flexible depth of field photography. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 33, n. 1, p. 58–71, jan. 2010.

LAWS, K. **Textured image segmentation**. jan. 1980. 178 f. Tese (Doutorado) — Dept. Electrical Engineering, University of Southern California, Los Angeles, jan. 1980.

LEVIN, A. et al. Image and depth from a conventional camera with a coded aperture. In: ACM SIGGRAPH, 2007, San Diego. **Proceedings...** New York: ACM, 2007. v. 26, n. 3, p. 70.

LI, H.; NGAN, K. Unsupervized video segmentation with low depth of field. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 17, n. 12, p. 1742–1751, dez. 2007.

LI, H.; NGAN, K. Learning to extract focused objects from low DOF images. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 21, n. 11, p. 1571–1580, nov. 2011.

LIPSON, A.; LIPSON, S. G.; LIPSON, H. **Optical Physics**. 4th. ed. Cambridge: Cambridge University, 2010.

LONG, B. **Complete Digital Photography**. 7th. ed. Boston: Cengage Learning PTR, 2012.

LUCA, F. P. de; THOMAZ, C. E. Analysis of methods for extraction of information on images with low-depth of field. In: CONGRESSO SAE BRASIL, 2015, São Paulo. **Anais...** São Paulo: SAE, 2015.

MARCOS, S.; MORENO, E.; NAVARRO, R. The depth-of-field of the human eye from objective and subjective measurements. **Vision Research**, v. 39, n. 12, p. 2039–2049, mar. 1999.

METTER, R. L. V.; BEUTEL, J.; KUNDEL, H. L. **Handbook of Medical Imaging: Physics and Psychophysics**. Washington: SPIE, 2009.

NEVEROVA, N.; KONIK, H. Edge-based method for sharp region extraction from low depth of field images. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2012, San Diego. **Proceedings...** San Diego: IEEE, 2012. p. 1–6.

OLSON, D. L.; DELEN, D. **Advanced Data Mining Techniques**. Berlin: Springer, 2008.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man and Cybernetics**, v. 9, n. 1, p. 62–66, jan. 1979.

PESSOA, F. O Monstrengo. In: ____. **Mensagem**. São Paulo: Centaur, 2013. p. 296–297.

PETERSON, J. et al. The effects of blur on selective visual attention. **Journal of Vision**, v. 15, n. 12, p. 1071–1071, set. 2015.

POTMESIL, M.; CHAKRAVARTY, I. A lens and aperture camera model for synthetic image generation. In: SIGGRAPH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 8., 1981, New York. **Proceedings...** New York: ACM, 1981. p. 297–305.

POWERS, D. M. W. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, dez. 2011.

PRÄKEL, D. **Basics Photography 07: Exposure**. Lausanne: AVA, 2009.

PUJOL, A.; VILLANUEVA, J. J.; ALBA, J. L. A supervised modification of the Hausdorff distance for visual shape classification. **IJPRAI**, v. 16, n. 3, p. 349–359, maio 2002.

SALDAÑA, E. et al. Review: computer vision applied to the inspection and quality control of fruits and vegetables. **Brazilian Journal of Food Technology**, v. 16, n. 4, p. 254–272, out. 2013.

SAVAZZI, E. **Digital photography for science**. Raleigh: Lulu, 2010.

SERWAY, R. A.; JEWETT, J. J. W. **Physics for Scientists and Engineers with Modern Physics**. 9th. ed. Boston: Brooks Cole, 2013.

SWAIN, C.; CHEN, T. Defocus-based image segmentation. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING (ICASSP), 1995, Detroit. **Proceedings...** Detroit: IEEE, 1995. v. 4, p. 2403–2406.

TARRANT, J. **Understanding Digital Cameras: Getting the Best Image from Capture to Output**. Oxford: Focal, 2007.

THE ECONOMIST. O carro autônomo pode dominar o mundo? **O Estado de São Paulo**, 06 Agosto 2015 2015. Disponível em: <http://economia.estadao.com.br/noticias/geral, o-carro-autonomo-pode-dominar-o-mundo-,1739102>. Acesso em: 11 ago. 2015.

WANG, J. et al. Unsupervised multiresolution segmentation for images with low depth of field. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 1, p. 85–90, jan. 2001.

WANG, Z.; BOVIK, A. Mean squared error: Love it or leave it? A new look at signal fidelity measures. **IEEE Signal Processing Magazine**, v. 26, n. 1, p. 98–117, jan. 2009.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, abr. 2004.

WHITAKER, J. C. **The Electronics Handbook**. Boca Raton: CRC, 1996.

WIKIMEDIA COMMONS. **Lentilles**. 2005. Altura: 324 pixels. Largura: 924 pixels. 5 kB. Formato PNG. Disponível em: <https://commons.wikimedia.org/wiki/File:Lentilles.png>. Acesso em: 20 set. 2015.

WIKIMEDIA COMMONS. **Spherical aberration**. 2011. Altura: 1122 pixels. Largura: 2362 pixels. 10 kB. Formato SVG. Disponível em: <https://commons.wikimedia.org/wiki/File: Spherical_aberration_3.svg>. Acesso em: 20 set. 2015.

WOLFE, A.; SHEPPARD, R. **The Art of the Photograph: Essential Habits for Stronger Compositions**. New York: Amphoto, 2013.

WON, C. S.; PYUN, K.; GRAY, R. Automatic object segmentation in images with low depth of field. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2002, Rochester. **Proceedings...** Rochester: IEEE, 2002. v. 3, p. 805–808.

YANG, X.-P.; YANG, T.; YANG, L.-B. Extracting focused object from defocused background using cellular neural networks. In: IEEE INTERNATIONAL WORKSHOP ON CELLULAR NEURAL NETWORKS AND THEIR APPLICATIONS (CNNA), 3., 1994, Rome. **Proceedings...** Rome: IEEE, 1994. p. 451–455.

YE, Z.; LU, C.-C. Unsupervised multiscale focused objects detection using hidden Markov tree. In: JOINT CONFERENCE ON INFORMATION SCIENCE (JCIS), 2002, Durham. **Proceedings...** Durham, 2002. p. 812–815.

ZHANG, K. et al. A fuzzy segmentation of salient region of interest in low depth of field image. In: CHAM, T.-J. et al. (Ed.). **Advances in Multimedia Modeling: 13th International Multimedia Modeling Conference, MMM 2007, Singapore, January 9-12, 2007. Proceedings, Part I**. Berlin: Springer, 2006. p. 782–791.

**APÊNDICE A** – Código fonte do método proposto

# APÊNDICE A – CÓDIGO FONTE DO MÉTODO PROPOSTOAPÊNDICES

Código da função em Matlab para o algoritmo de segmentação de imagens a partir de duas cenas com diferenças de profundidade de campo:

```
1
2  function [ imgDeltaFinal, ...
3             tempo, ...
4             tempoReduz ] = SegmentacaoLowDOF(img1, img2, ....
5                                              aplicarMorf, ...
6                                              salvarEtapa, ...
7                                              caminhoSaidaEtapa,...
8                                              nomeSaida, ...
9                                              dimFiltPassBaixa,...
10                                             exibirPlot,...
11                                             salvarPlot)
12
13 %SegmentacaoLowDOF Segmentação de imagens com baixo DOF
14 %=================================================================
15 %   Função para segmentação de imagens com baixa profundidade de
16 %   campo (DOF).
17 %
18 %   Variáveis de entrada:
19 %       img1        = Primeira imagem (baixa profundidade de
20 %                     campo)
21 %       img2        = Segunda imagem (alta profundidade de
22 %                     campo)
23 %       aplicarMorf = Valor booleano, true se deseja aplicar os
24 %                     filtros morfológicos no algoritmo ou false
25 %                     caso não queira aplicação
26 %       salvarEtapa = Salva como imagens cada etapa de execução
27 %                     do método para controle de operação e
28 %                     avaliações intermediárias
29 %       caminhoSaidaEtapa = Caminho onde deverão ser salvos as
30 %                     imagens no caso da variável 'salvarEtapa'
31 %                     ser true.
32 %       nomeSaida   = Nome dos arquivos salvos caso a variável
33 %                     'salvarEtapa' ser true.
34 %       dimFiltPassBaixa = Dimensão do kernel do filtro
35 %                     passa-baixa aplicado.
36 %       exibirPlot  = Exibe um plot 3d (surf) das etapas
37 %       salvarPlot  = Salva o plot 3d (surf) das etapas
38 %=================================================================
```

```
39
40 %-------------------------------------------------------------------
41 % Verifica se as imagens possuem mesmo tamanho
42 %-------------------------------------------------------------------
43
44 [img1M,img1N,img1L] = size(img1);
45 [img2M,img2N,img2L] = size(img2);
46
47 if ((img1M ~= img2M) || (img1N ~= img2N) || (img1L ~= img2L))
48     msgbox(['As imagens possuem tamanhos diferentes!' ...
49             'Terminando o processo...']);
50     return
51 end
52
53 %-------------------------------------------------------------------
54 % Começa o cálculo do tempo
55 %-------------------------------------------------------------------
56
57 T1 = tic;
58
59 %-------------------------------------------------------------------
60 % Converte para cinza as duas imagens de análise
61 %-------------------------------------------------------------------
62
63 img1gray = uint8(rgb2gray(img1));
64 img2gray = uint8(rgb2gray(img2));
65
66 if (salvarEtapa)
67
68     Salvar_Imagem_Etapa(img1gray, caminhoSaidaEtapa, ...
69         nomeSaida, dimFiltPassBaixa, ...
70         '_01_1_Grayscale', 'png', ...
71         'Imagem 1', ...
72         -45,60, exibirPlot, salvarPlot);
73
74     Salvar_Imagem_Etapa(img2gray, caminhoSaidaEtapa, ...
75         nomeSaida, dimFiltPassBaixa, ...
76         '_01_2_Grayscale', 'png', ...
77         'Imagem 2', ...
78         -45,60, exibirPlot, salvarPlot);
79 end
80
```

```
81 %-------------------------------------------------------------------
82 % Procura uma correlação entre as imagens, de modo a maximizar as
83 % semelhanças na comparação. A primeira imagem será deslocada
84 % para atender a maximização
85 %-------------------------------------------------------------------
86
87 matCorr = normxcorr2(uint8(img1gray-mean(mean(img1gray))), ...
88                      uint8(img2gray-mean(mean(img2gray))));
89
90 [ypeak, xpeak] = find(matCorr==max(matCorr(:)));
91 yoffSet = ypeak-size(img1gray,1);
92 xoffSet = xpeak-size(img1gray,2);
93
94 img1corr = circshift(img1gray,[yoffSet xoffSet]);
95 img2corr = img2gray;
96
97 if (salvarEtapa)
98
99     Salvar_Imagem_Etapa(img1corr, caminhoSaidaEtapa, ...
100         nomeSaida, dimFiltPassBaixa, ...
101         '_02_1_Correlated', 'png', ...
102         'Imagem 1 - Após correlação máxima', ...
103         -45,60, exibirPlot, salvarPlot);
104
105     Salvar_Imagem_Etapa(img2corr, caminhoSaidaEtapa, ...
106         nomeSaida, dimFiltPassBaixa, ...
107         '_02_2_Correlated', 'png', ...
108         'Imagem 2 - Após correlação máxima', ...
109         -45,60, exibirPlot, salvarPlot);
110 end
111
112 %-------------------------------------------------------------------
113 % Tempo sem correlação
114 %-------------------------------------------------------------------
115
116 T2 = tic;
117
118 %-------------------------------------------------------------------
119 % Normaliza as imagens correlacionadas
120 %-------------------------------------------------------------------
121
122 img1corr = double(img1corr);
```

```
123 img1corr = normalizarMatriz(img1corr);
124
125 img2corr = double(img2corr);
126 img2corr = normalizarMatriz(img2corr);
127
128 if (salvarEtapa)
129
130     Salvar_Imagem_Etapa(img1corr, caminhoSaidaEtapa, ...
131         nomeSaida, dimFiltPassBaixa, ...
132         '_03_1_Normalized', 'png', ...
133         'Imagem 1 - Após correlação e normalização', ...
134         -45,60, exibirPlot, salvarPlot);
135
136     Salvar_Imagem_Etapa(img2corr, caminhoSaidaEtapa, ...
137         nomeSaida, dimFiltPassBaixa, ...
138         '_03_2_Normalized', 'png', ...
139         'Imagem 2 - Após correlação e normalização', ...
140         -45,60, exibirPlot, salvarPlot);
141 end
142
143 %-----------------------------------------------------------------
144 % Calcula os gradientes. Para o salvamento num formato de imagem
145 % é feita uma normalização que é válida apenas para a representa-
146 % ção. Não deve ser aplicada ao algoritmo em si.
147 %-----------------------------------------------------------------
148
149 img1grad = abs(imgradient(img1corr));
150 img2grad = abs(imgradient(img2corr));
151
152 if (salvarEtapa)
153
154     Salvar_Imagem_Etapa(img1grad, caminhoSaidaEtapa, ...
155         nomeSaida, dimFiltPassBaixa, ...
156         '_04_1_Grad', 'png', ...
157         'Imagem 1 - Gradiente da imagem (magnitude)', ...
158         -45,60, exibirPlot, salvarPlot);
159
160     Salvar_Imagem_Etapa(img2grad, caminhoSaidaEtapa, ...
161         nomeSaida, dimFiltPassBaixa, ...
162         '_04_2_Grad', 'png', ...
163         'Imagem 2 - Gradiente da imagem (magnitude)', ...
164         -45,60, exibirPlot, salvarPlot);
```

```
165
166     img1grad_Norm = normalizarMatriz(img1grad);
167     img2grad_Norm = normalizarMatriz(img2grad);
168
169     Salvar_Imagem_Etapa(img1grad_Norm, caminhoSaidaEtapa, ...
170         nomeSaida, dimFiltPassBaixa, ...
171         '_04_1_Grad_Norm', 'png', ...
172         'Imagem 1 - Gradiente da imagem', ...
173         -45,60, false, false);
174
175     Salvar_Imagem_Etapa(img2grad_Norm, caminhoSaidaEtapa, ...
176         nomeSaida, dimFiltPassBaixa, ...
177         '_04_2_Grad_Norm', 'png', ...
178         'Imagem 2 - Gradiente da imagem', ...
179         -45,60, false, false);
180
181 end
182
183 %-----------------------------------------------------------------
184 % Aplica um filtro passa-baixa. A razão é que o gradiente não é
185 % 100% igual, por diferenças de luzes e intensidades e então o
186 % filtro espalha os picos, sendo a compração possível
187 %-----------------------------------------------------------------
188
189 filtKernel = ones(dimFiltPassBaixa,dimFiltPassBaixa);
190 img1gradF = conv2(img1grad, filtKernel, 'same');
191 img2gradF = conv2(img2grad, filtKernel, 'same');
192
193 if (salvarEtapa)
194
195     Salvar_Imagem_Etapa(img1gradF, caminhoSaidaEtapa, ...
196         nomeSaida, dimFiltPassBaixa, ...
197         '_05_1_LowPass', 'png', ...
198         'Imagem 1 - Após aplicação do filtro passa-baixa', ...
199         -45,60, exibirPlot, salvarPlot);
200
201     Salvar_Imagem_Etapa(img2gradF, caminhoSaidaEtapa, ...
202         nomeSaida, dimFiltPassBaixa, ...
203         '_05_2_LowPass', 'png', ...
204         'Imagem 2 - Após aplicação do filtro passa-baixa', ...
205         -45,60, exibirPlot, salvarPlot);
206
```

```
207     img1gradF_Norm = normalizarMatriz(img1gradF);
208         img2gradF_Norm = normalizarMatriz(img2gradF);
209 
210     Salvar_Imagem_Etapa(img1gradF_Norm, caminhoSaidaEtapa, ...
211         nomeSaida, dimFiltPassBaixa, ...
212         '_05_1_LowPass_Norm', 'png', ...
213         'Imagem 1 - Após aplicação do filtro passa-baixa', ...
214         -45,60, false, false);
215 
216     Salvar_Imagem_Etapa(img2gradF_Norm, caminhoSaidaEtapa, ...
217         nomeSaida, dimFiltPassBaixa, ...
218         '_05_2_LowPass_Norm', 'png', ...
219         'Imagem 2 - Após aplicação do filtro passa-baixa', ...
220         -45,60, false, false);
221 
222 end
223 
224 %-----------------------------------------------------------------
225 % Calcula a diferença entre ambas as imagens
226 %-----------------------------------------------------------------
227 
228 imgDeltaDif = abs(img1gradF - img2gradF);
229 imgDeltaDif = normalizarMatriz(imgDeltaDif);
230 imgDeltaDif = ones(size(imgDeltaDif)) - imgDeltaDif;
231 
232 if (salvarEtapa)
233 
234     Salvar_Imagem_Etapa(imgDeltaDif, caminhoSaidaEtapa, ...
235         nomeSaida, dimFiltPassBaixa, ...
236         '_06___Differ', 'png', ...
237         'Imagem - Após aplicação da diferenciação (em módulo)', ...
238         -45,60, exibirPlot, salvarPlot);
239 
240 end
241 
242 %-----------------------------------------------------------------
243 % Calcula o produto de Hadamard entre a imagem resultante da
244 % diferença e a imagem com baixo DOF
245 %-----------------------------------------------------------------
246 
247 imgDeltaDif2 = abs(img1gradF .* imgDeltaDif);
248 imgDeltaDif2 = normalizarMatriz(imgDeltaDif2);
```

```
249
250 if (salvarEtapa)
251
252     Salvar_Imagem_Etapa(imgDeltaDif2, caminhoSaidaEtapa, ...
253         nomeSaida, dimFiltPassBaixa, ...
254         '_07___Hadamard', 'png', ...
255         'Imagem - Após aplicação do produto de Hadamard', ...
256         -45,60, exibirPlot, salvarPlot);
257
258 end
259
260 %-----------------------------------------------------------------
261 % Converte para ser uma imagem binária e outra intermediária
262 %-----------------------------------------------------------------
263
264 imgDeltaDifBin = imgDeltaDif2;
265
266 level = graythresh(imgDeltaDifBin);
267 imgDeltaDifBin = im2bw(imgDeltaDifBin,level);
268
269 if (salvarEtapa)
270
271     Salvar_Imagem_Etapa(imgDeltaDifBin, caminhoSaidaEtapa, ...
272         nomeSaida, dimFiltPassBaixa, ...
273         '_08___Threshold', 'png', ...
274         'Imagem - Após limiarização', ...
275         -45,60, exibirPlot, salvarPlot);
276
277 end
278
279 %-----------------------------------------------------------------
280 % Operações básica de morfologia para remover pontos pequenos e
281 % preencher alguns buracos, além de realizar um crescimento
282 % das bordas (dilatação)
283 %-----------------------------------------------------------------
284
285 if (aplicarMorf)
286     SE = strel('disk',11);
287     imgDeltaFinal = imdilate(imgDeltaDifBin,SE);
288     imgDeltaFinal = imerode(imgDeltaFinal,SE);
289     imgDeltaFinal = bwareaopen(imgDeltaFinal, ...
290                                round(img1M*img1N*0.01));
```

```
291     imgDeltaFinal = imcomplement(imgDeltaFinal);
292     imgDeltaFinal = bwareaopen(imgDeltaFinal, ...
293                                round(img1M*img1N*0.01));
294     imgDeltaFinal = imcomplement(imgDeltaFinal);
295     imgDeltaFinal = imdilate(imgDeltaFinal,SE);
296 else
297     imgDeltaFinal = imgDeltaDifBin;
298 end
299
300 if (salvarEtapa)
301
302     Salvar_Imagem_Etapa(imgDeltaFinal, caminhoSaidaEtapa, ...
303         nomeSaida, dimFiltPassBaixa, ...
304         '_09___Morphological', 'png', ...
305         'Imagem - Após operações morfológicas', ...
306         -45,60, exibirPlot, salvarPlot);
307 end
308
309 %-----------------------------------------------------------------
310 % Medição do tempo do algoritmo
311 %-----------------------------------------------------------------
312
313 tempo = toc(T1);
314 tempoReduz = toc(T2);
315
316 end
```

**APÊNDICE B** – Código fonte para geração de imagens

# APÊNDICE B – CÓDIGO FONTE PARA GERAÇÃO DE IMAGENS

Código em Matlab para o algoritmo de geração de imagens artificias que simulam a baixa profundidade de campo junto dos comentários pertinentes ao mesmo:

```
1
2  %================================================================
3  %
4  %   TRABALHO DE MESTRADO. ESSE ALGORITMO CRIA UMA IMAGEM
5  %   COMBINANDO ELEMENTOS DE DOIS TIPOS: SPRITES E STAGES.
6  %   AS SPRITES SÃO CADA UM DOS ELEMENTOS QUE COMPÕE A CENA
7  %   PODENDO SER MAIS DE UM POR IMAGEM RESULTANTE. JÁ O
8  %   STAGE É O BACKGROUND, E HÁ SOMENTE UM POR GERAÇÃO.
9  %
10 %================================================================
11
12 %----------------------------------------------------------------
13 %   Limpa a execução anterior do Matlab
14 %----------------------------------------------------------------
15
16 clc;
17 clear all;
18 close all;
19
20 %----------------------------------------------------------------
21 %   Configuração de parâmetros de operação do algoritmo
22 %----------------------------------------------------------------
23
24 numeroImagens = 10;          % Seta o número de imagens que serão
25                              % criadas pelo processo.
26
27 probabSpriteFoco = 0.8;      % Determina a probabilidade de haver
28                              % um objeto em foco (sprite). Será
29                              % sempre o primeiro sprite da lista
30                              % sorteada
31
32 probabSpriteFoco2 = 0.5;     % Determina a probabilidade de haver
33                              % um segundo objeto em foco
34
35 probabSpriteFoco3 = 0.2;     % Determina a probabilidade de haver
36                              % um terceiro objeto em foco
37
38 distanciaMin = 10;           % Determina a distância mínima
```

```
39                              % perimitida para uma sprite em
40                              % metros. Esse limite é necessário
41                              % para que as formulações da ótica
42                              % permaneçam válidas
43
44  distanciaMax = 100;         % Determina a distância máxima
45                              % perimitida para uma sprite em
46                              % metros. Esse limite é necessário
47                              % para que o objeto não seja tão
48                              % pequeno e acabe desaparecendo
49                              % na cena
50  tamanhoMaxLarg = 1024;      % Determina o tamanho máximo para
51                              % a largura de uma imagem de fundo
52
53  tamanhoMaxAlt = 768;        % Determina o tamanho máximo para
54                              % a altura de uma imagem de fundo
55
56  numeroMaxSprites = 5;       % Determina o número máximo de sprit
57                              % a serem usados numa cena. Caso
58                              % não seja definido (comentado), o
59                              % número máximo será o número de
60                              % sprites definidos (imagens
61                              % declaradas)
62
63  salvarEtapas = false;       % Habilita ou não o salvamento em
64                              % arquivos de imagens, as etapas
65                              % intermediárias do processo
66
67  dispSimples = true;         % Escreve na saída do Matlab apenas
68                              % informações de controle de execução
69                              % sem os valores calculados nas
70                              % etapas
71
72  f = 70;                     % Distância focal da lente (mm)
73                              % usada
74
75  F = distanciaMin;           % Distância do ponto em foco (m).
76                              % Será considerada a mais próxima
77                              % possível. Esse valor pode ser
78                              % alterado para qualquer outro,
79                              % mas o mais interessante é deixá-lo
80                              % no valor mais próximo possível
```

```
81                              % (perto da lente) e que haja objetos
82
83  hs = 24;                    % Altura do sensor (mm). Sensor
84                              % full-frame de 35mm tem altura de
85                              % 24mm.
86
87  cropFact = 1.54;            % Fator de corte do sensor da camera
88
89  cocLimite = 0.015;          % Limite para entendimento de que é
90                              % DoF em milimetros (mm)
91
92  %-------------------------------------------------------------------
93  %   Configuração de caminhos do algoritmo
94  %-------------------------------------------------------------------
95
96  caminhoRoot    = pwd;
97  caminhoSprites = [caminhoRoot '\01 - Sprites\'];
98  caminhoFundos  = [caminhoRoot '\02 - Backgrounds\'];
99  caminhoSaida   = [caminhoRoot '\03 - Resultados\'];
100
101 %-------------------------------------------------------------------
102 %   Limpa os arquivos de saída e recria os diretórios
103 %-------------------------------------------------------------------
104
105 if isequal(exist(caminhoSaida, 'dir'),7)
106     rmdir(caminhoSaida,'s')
107 end
108
109 mkdir(caminhoSaida);
110 if (salvarEtapas)
111     mkdir([caminhoSaida 'Intermediario_00_Fundo\']);
112     mkdir([caminhoSaida 'Intermediario_01_Resize\']);
113     mkdir([caminhoSaida 'Intermediario_02_Crop\']);
114     mkdir([caminhoSaida 'Intermediario_03_Padding\']);
115     mkdir([caminhoSaida 'Intermediario_04_Shift\']);
116     mkdir([caminhoSaida 'Intermediario_05_Expand\']);
117     mkdir([caminhoSaida 'Intermediario_06_Blur\']);
118 end
119 mkdir([caminhoSaida 'Suporte\']);
120
121 %-------------------------------------------------------------------
122 %   Configuração de caminhos das imagens (nomes dos arquivos)
```

```
123 %-------------------------------------------------------------
124
125 arquivosSprites = {'Ave_Arara_01.png';
126                     'Ave_Arara_02.png';
127                     'Ave_Aguia_01.png';
128                     'Ave_Aguia_02.png';
129                     'Ave_Ganso_01.png';
130                     'Ave_Gaivota_01.png';
131                     'Ave_Gaivota_02.png';
132                     'Ave_Beijaflor_01.png';
133                     'Ave_Pombo_01.png';
134                     'Mamifero_Morcego_01.png';
135                     'Objeto_Bola_01.png';
136                     'Objeto_Bola_02.png';
137                     'Objeto_Bola_03.png';
138                     'Objeto_Bola_04.png'};
139
140 %-------------------------------------------------------------
141 %   Definição do tamanho real de cada objeto representado
142 %   na imagem. Valores em milimetros.
143 %-------------------------------------------------------------
144
145 tamanhoRealSprite = [ 1100 ;    % Arara 01
146                       1100 ;    % Arara 02
147                       778 ;     % Aguia 01
148                       1300 ;    % Aguia 02
149                       900 ;     % Ganso 01
150                       920 ;     % Gaivota 01
151                       750 ;     % Gaivota 02
152                       120 ;     % Beija-flor 01
153                       300 ;     % Pombo 01
154                       300 ;     % Morcego 01
155                       222 ;     % Bola 01 = Bola de futebol
156                       68 ;      % Bola 02 = Bola de tênis
157                       248 ;     % Bola 03 = Bola de basquete
158                       500 ];    % Bola 04 = Bola de praia
159
160 %-------------------------------------------------------------
161 %   Configuração de caminhos das imagens de fundo (nomes dos
162 %   arquivos)
163 %-------------------------------------------------------------
164
```

```
165 arquivosFundos  = {'Ceu_01.png';
166                    'Ceu_02.png';
167                    'Ceu_03.png';
168                    'Ceu_04.png';
169                    'Ceu_05.png';
170                    'Ceu_06.png';
171                    'Ceu_07.png';
172                    'Floresta_01.png';
173                    'Floresta_01.png';
174                    'Urbano_01.png'};
175
176 %-------------------------------------------------------------------
177 %   Definição da distância do objeto em cena à camera em metros
178 %-------------------------------------------------------------------
179
180 distanciaFundo = [ 5000 ;   % Céu 01
181                    5000 ;   % Céu 02
182                    5000 ;   % Céu 03
183                    5000 ;   % Céu 04
184                    5000 ;   % Céu 05
185                    5000 ;   % Céu 06
186                    5000 ;   % Céu 07
187                    400 ;    % Floresta 01
188                    400 ;    % Floresta 02
189                    300 ];   % Urbano 01
190
191 %-------------------------------------------------------------------
192 %   Criação de um vetor de índice para cada elemento
193 %-------------------------------------------------------------------
194
195 arquivosSpritesInt = 1:size(arquivosSprites,1);
196 arquivosSpritesInt = arquivosSpritesInt';
197
198 %-------------------------------------------------------------------
199 %   Define o vetor de configuração de parâmetros óticos do ground
200 %   truth, além de outros parâmetros da ótica do sistema simulado
201 %-------------------------------------------------------------------
202
203 confOtica = {'f1.4';
204              'f2.0';
205              'f2.8';
206              'f4.0';
```

```
207                 'f5.6';
208                 'f8.0';
209                 'f11';
210                 'f16';
211                 'f22'};
212
213 N = [1.4;
214      2.0;
215      2.8;
216      4.0;
217      5.6;
218      8.0;
219      11;
220      16;
221      22];           % Vetor com as aberturas que deve seguir o
222                     % vetor de texto definido anteriormente.
223
224 %-------------------------------------------------------------------
225 %   Determina o número máximo de sprites possíveis, conforme
226 %   o tamanho do vetor definido. Faz o mesmo para o fundo da
227 %   imagem. Caso o usuário já tenha pré-definido um valor
228 %   então a escolha do usuário é predominante
229 %-------------------------------------------------------------------
230
231 if exist('numeroMaxSprites','var') == 0
232     numeroMaxSprites = size(arquivosSprites,1);
233 end
234 numeroMaxFundos  = size(arquivosFundos,1);
235
236 %-------------------------------------------------------------------
237 %   Definições aleatórias. Para cada imagem (cena) que será
238 %   criada, já são definidos os parâmetros de quantos sprite
239 %   haverá na cena e de qual o fundo será usado.
240 %-------------------------------------------------------------------
241
242 numeroSprites = randi([1 numeroMaxSprites],numeroImagens,1);
243 listaFundos = randi([1 numeroMaxFundos],numeroImagens,1);
244
245 %-------------------------------------------------------------------
246 %   Realização de um loop para definir os parâmetros físicos
247 %   de cada cena
248 %-------------------------------------------------------------------
```

```
249
250 for i=1:numeroImagens
251
252     if (numeroSprites(i)>0)
253
254         %-------------------------------------------------------
255         %   Monta a lista de quais sprites serão usados na
256         %   cena e qual a posição de cada um deles (x,y)
257         %-------------------------------------------------------
258
259         listaSprites{i} = randperm(size(arquivosSprites,...
260                           1),numeroMaxSprites);
261
262         posYSprite{i}   = randi([0 tamanhoMaxAlt],1,...
263                           numeroSprites(i));
264         posXSprite{i}   = randi([0 tamanhoMaxLarg],1,...
265                           numeroSprites(i));
266
267         %-------------------------------------------------------
268         %   Monta a lista de distância de cada sprite na cena. O
269         %   primeiro elemento é substituido por disância zero, ou
270         %   seja, em foco com uma probabilidade pre-definida de
271         %   acontecer.
272         %-------------------------------------------------------
273
274         distanciaSpriteElem = randi([distanciaMin ...
275                               distanciaMax],1,numeroSprites(i));
276         if (rand <= probabSpriteFoco)
277             distanciaSpriteElem(1) = F;
278         end
279         if (rand <= probabSpriteFoco2)
280             distanciaSpriteElem(2) = F;
281         end
282         if (rand <= probabSpriteFoco3)
283             distanciaSpriteElem(3) = F;
284         end
285         distanciaSprite{i} = distanciaSpriteElem;
286     end
287 end
288
289 %-----------------------------------------------------------------
290 %   Ajusta as matrizes de cell para o formato desejado
```

```
291 %-------------------------------------------------------------
292
293 if exist('listaSprites','var') == 1
294     listaSprites = listaSprites';
295 end
296 if exist('posXSprite','var') == 1
297     posXSprite = posXSprite';
298 end
299 if exist('posYSprite','var') == 1
300     posYSprite = posYSprite';
301 end
302 if exist('distanciaSprite','var') == 1
303     distanciaSprite = distanciaSprite';
304 end
305
306 %-------------------------------------------------------------
307 %   Loop pelo total de imagens de saída
308 %-------------------------------------------------------------
309
310 for i=1:numeroImagens
311
312     for m=1:size(confOtica,1)
313
314         %-----------------------------------------------------
315         %   Mensagem na saída do Matlab para controle de fluxo
316         %-----------------------------------------------------
317
318         clc;
319         disp(' ');
320         disp(['-----------------------------------------------'...
321               '-----']);
322         disp(['Nova imagem -- ' num2str(i) ' de ' ...
323               num2str(numeroImagens)]);
324         disp(['-----------------------------------------------'...
325               '-----']);
326         disp(' ');
327         disp(['   Config ' confOtica{m} ] );
328         disp(['   Etapa ' num2str(m+(i-1)*(size(confOtica,1)))...
329               ' de ' num2str(numeroImagens*size(confOtica,1))...
330               ' (' num2str((m+(i-1)*(size(confOtica,1)))/...
331               (numeroImagens*size(confOtica,1)),2) '%)']);
332
```

```
333         %-------------------------------------------------------
334         %   Limpa as variáveis carregadas. Por serem matrizes que
335         %   podem alterar de tamanho ao longo da execução, é
336         %   recomendado que sejam limpas e evitar conflitos
337         %-------------------------------------------------------
338
339         clear imagemFundo;
340         clear imagemComposta;
341
342         clear imagemFundoGT;
343         clear imagemCompostaGT;
344
345         %-------------------------------------------------------
346         %   Carrega a imagem de fundo
347         %-------------------------------------------------------
348
349         imagemFundo = imread([caminhoFundos ...
350                       arquivosFundos{listaFundos(i)}]);
351         imagemComposta = double(imagemFundo);
352
353         imagemCompostaGT = zeros(size(imagemComposta,1),...
354                                  size(imagemComposta,2),1);
355
356         %-------------------------------------------------------
357         %   Cálculo do círculo de confusão no fundo. Primeiro
358         %   determina a distância do fundo e em seguida cálcula o
359         %   diametro do CoC e corrige para que o resultado seja o
360         %   diametro e não o raio do círculo
361         %-------------------------------------------------------
362
363         dFundo = distanciaFundo(listaFundos(i));
364
365         spriteCoCFundo = ((f/1000)^2)/(N(m)*(F-f/1000))*...
366                          abs(1-(F/dFundo))*1000;
367         blurFundo = ceil(spriteCoCFundo/hs*size(imagemFundo,1));
368         blurDesvFundo = blurFundo*2;
369
370         %-------------------------------------------------------
371         %   Aplica o blur no fundo conforme parâmetro calculado
372         %   anteriormente. Aplica um filtro gaussiano
373         %-------------------------------------------------------
374
```

```
375         H = fspecial('gaussian',blurFundo,blurDesvFundo);
376         imagemComposta = imfilter(imagemComposta,H,'replicate');
377 
378         if (salvarEtapas)
379                 imwrite(uint8(imagemComposta),[caminhoSaida ...
380                 'Intermediario_00_Fundo\Imagem_' num2str(i) ...
381                 '_' confOtica{m} '_01_Composta.png']);
382             imwrite(uint8(imagemCompostaGT),[caminhoSaida ...
383                 'Intermediario_00_Fundo\Imagem_' num2str(i) ...
384                 '_' confOtica{m} '_01_Composta_GT.png']);
385         end
386 
387         %-------------------------------------------------------
388         %   Determina o tamanho da imagem de fundo
389         %-------------------------------------------------------
390 
391         tamanhoAlt = size(imagemFundo,1);
392         tamanhoLarg = size(imagemFundo,2);
393 
394         %-------------------------------------------------------
395         %   Ordena os elementos para que se monte os mais
396         %   distantes primeiro. Ordenação descrescente de
397         %   distância
398         %-------------------------------------------------------
399 
400         [listaDistanciasDesc,listaDistanciasIndex] = ...
401                         sort(distanciaSprite{i},'descend');
402 
403         %-------------------------------------------------------
404         %   Monta a sobreposição dos sprites
405         %-------------------------------------------------------
406 
407         if numeroSprites(i) > 0
408             for k=1:numeroSprites(i)
409 
410                 %-----------------------------------------------
411                 %   Organiza a varedura de forma a começar pelos
412                 %   objetos mais distantes
413                 %-----------------------------------------------
414 
415                 j = listaDistanciasIndex(k);
416 
```

```
417                 %-----------------------------------------------
418                 %   Limpa variáveis do loop
419                 %-----------------------------------------------
420
421                  clear imagemSprite;
422                  clear imagemSpriteAlpha;
423
424                 %-----------------------------------------------
425                 %   Carrega a imagem do sprite
426                 %-----------------------------------------------
427
428                 [imagemSprite,map,imagemSpriteAlpha] = imread(...
429                     [caminhoSprites arquivosSprites{...
430                     listaSprites{i}(j)}]);
431
432                 if (~dispSimples)
433                     disp(' ');
434                     disp(['Nome = ' arquivosSprites{...
435                         listaSprites{i}(j)}]);
436                 end
437
438                 %-----------------------------------------------
439                 %   Determina a variável de identificação de cada
440                 %   sprite, para busca de informação fixa delas
441                 %-----------------------------------------------
442
443                                 ID = listaSprites{i}(j);
444                 if (~dispSimples)
445                     disp(['ID = ' int2str(ID)]);
446                 end
447
448                 %-----------------------------------------------
449                 %   Caso seja cinza a imagem, converte para
450                 %   dimensão 3 para ficar compatível com as
451                 %   demais
452                 %-----------------------------------------------
453
454                 if size(imagemSprite,3) == 1
455                     imagemSprite = cat(3,imagemSprite,...
456                         imagemSprite,imagemSprite);
457                 end
458
```

```
459                 %-----------------------------------------------
460                 %   Determina o tamanho do sprite conforme a
461                 %   distância do mesmo na cena. A fórmula correta
462                 %   para esse cálculo é:
463                 %
464                 %   i = f*o/(d-f)
465                 %
466                 %   Onde: h é o tamanho do objeto no sensor
467                 %         f é a distância focal
468                 %         H é o tamanho real do objeto
469                 %         d é a distancia do objeto à lente
470                 %   Como ainda depende da representação da cena,
471                 %   então considerando um sensor full frame de
472                 %   35mm, o mesmo possui dimensão 36x24mm. Em
473                 %   termos de proporção, a fórmula fica:
474                 %
475                 %   i/hs = (f*o/hs)*(1/(d-f))
476                 %
477                 %   Onde: hs é a altura do sensor (24mm)
478                 %-----------------------------------------------
479
480                 distancia = distanciaSprite{i}(j);
481                 tamanhoReal = tamanhoRealSprite(ID);
482
483                 if (~dispSimples)
484                     disp(['Distancia = ' int2str(distancia) ...
485                         ' (m)']);
486                     disp(['Tamanho real = ' ...
487                         int2str(tamanhoReal) ...
488                         ' (mm)']);
489                 end
490
491                 escalaSprite = (f/hs)*(tamanhoReal/(distancia...
492                     *1000-f));
493                 tamanhoSensor = f*(tamanhoReal/(distancia*...
494                     1000-f));
495
496                 if (~dispSimples)
497                     disp(['Tamanho no sensor = ' num2str(...
498                         tamanhoSensor) ' (mm)']);
499                 end
500
```

```
501                 escalaSpriteLista(j) = escalaSprite;
502                 tamanhoSensorLista(j) = tamanhoSensor;
503
504                 %-------------------------------------------------
505                 %   ETAPA 01
506                 %   Faz o resize da imagem para o tamanho
507                 %   adequado
508                 %-------------------------------------------------
509
510                 clear imagemSpriteResize;
511                 clear imagemSpriteResizeAlpha;
512
513                 imagemSpriteResize = imresize(imagemSprite,...
514                     escalaSprite);
515                 imagemSpriteResizeAlpha = imresize(...
516                     imagemSpriteAlpha,escalaSprite);
517
518                 %-------------------------------------------------
519                 %   Salva o resultado parcial para controle
520                 %-------------------------------------------------
521
522                 if (salvarEtapas)
523                     imwrite(imagemSpriteResize,[caminhoSaida ...
524                         'Intermediario_01_Resize\Imagem_' ...
525                         num2str(i) '_' confOtica{m} '_' ...
526                         num2str(k) '_01_Resize.png']);
527                 end
528
529                 %-------------------------------------------------
530                 %   Determina informação de deslocamento
531                 %-------------------------------------------------
532
533                 deslocY = posYSprite{i}(j);
534                 deslocX = posXSprite{i}(j);
535
536                 %-------------------------------------------------
537                 %   ETAPA 02
538                 %   Realiza o crop do sprite caso esteja
539                 %   fora de visão
540                 %-------------------------------------------------
541
542                 clear imagemSpriteCrop;
```

```
543                 clear imagemSpriteCropAlpha;
544
545                 altSprite = size(imagemSpriteResize,1);
546                 largSprite = size(imagemSpriteResize,2);
547
548                 imagemSpriteCrop = imagemSpriteResize;
549                 imagemSpriteCropAlpha = imagemSpriteResizeAlpha;
550
551                 if ~((deslocY + ceil(altSprite/2) <= ...
552                         tamanhoAlt) && ...
553                     (deslocX + ceil(largSprite/2) <= ...
554                         tamanhoLarg) && ...
555                     (deslocY > ceil(altSprite/2)) && ...
556                     (deslocX > ceil(largSprite/2)))
557
558                     if ~(deslocY + ceil(altSprite/2) <= ...
559                             tamanhoAlt)
560                         imagemSpriteCrop = imagemSpriteCrop(...
561                             1:(floor(altSprite/2)+tamanhoAlt-...
562                             deslocY),:,:);
563                         imagemSpriteCropAlpha = ...
564                             imagemSpriteCropAlpha(...
565                             1:(floor(altSprite/2)+tamanhoAlt-...
566                             deslocY),:,:);
567                     elseif (deslocY == floor(altSprite/2))
568                         imagemSpriteCrop = imagemSpriteCrop(...
569                             1:altSprite,:,:);
570                         imagemSpriteCropAlpha = ...
571                             imagemSpriteCropAlpha(...
572                             1:altSprite,:,:);
573                     elseif ~(deslocY > ceil(altSprite/2))
574                         imagemSpriteCrop = imagemSpriteCrop(...
575                             abs(deslocY-floor(altSprite/2)):...
576                             altSprite,:,:);
577                         imagemSpriteCropAlpha = ...
578                             imagemSpriteCropAlpha(abs(deslocY-...
579                             floor(altSprite/2)):altSprite,:,:);
580                     end
581
582                     if ~(deslocX + ceil(largSprite/2) <= ...
583                             tamanhoLarg)
584                         imagemSpriteCrop = imagemSpriteCrop(...
```

```
585                     :,1:(floor(largSprite/2)+...
586                     tamanhoLarg-deslocX),:);
587                 imagemSpriteCropAlpha = ...
588                     imagemSpriteCropAlpha(:,1:(floor(...
589                     largSprite/2)+tamanhoLarg-...
590                     deslocX),:);
591             elseif (deslocX == floor(largSprite/2))
592                 imagemSpriteCrop = imagemSpriteCrop(...
593                     1:largSprite,:,:);
594                 imagemSpriteCropAlpha = ...
595                     imagemSpriteCropAlpha(...
596                     1:largSprite,:,:);
597             elseif ~(deslocX > ceil(largSprite/2))
598                 imagemSpriteCrop = imagemSpriteCrop(:,...
599                     abs(deslocX-floor(largSprite/2)):...
600                     largSprite,:);
601                 imagemSpriteCropAlpha = ...
602                     imagemSpriteCropAlpha(:,abs(...
603                     deslocX-floor(largSprite/2)):...
604                     largSprite,:);
605             end
606         end
607 
608         %------------------------------------------------
609         %   Exibe o resultado parcial para controle
610         %------------------------------------------------
611 
612         if (salvarEtapas)
613             imwrite(imagemSpriteCrop,[caminhoSaida ...
614                     'Intermediario_02_Crop\Imagem_' ...
615                     num2str(i) '_' confOtica{m} '_' ...
616                     num2str(k) '_01_Crop.png']);
617         end
618 
619         %------------------------------------------------
620         %   ETAPA 03
621         %   Executa o padding para que o sprite
622         %   tenha um tamanho igual ao do fundo
623         %------------------------------------------------
624 
625         clear imagemSpritePad;
626         clear imagemSpritePadAlpha;
```

```
627
628             altSpriteCrop = size(imagemSpriteCrop,1);
629             largSpriteCrop = size(imagemSpriteCrop,2);
630
631             paddingInf = abs(tamanhoAlt - altSpriteCrop);
632             paddingDir = abs(tamanhoLarg - largSpriteCrop);
633
634             imagemPadInf = zeros(paddingInf,...
635                 largSpriteCrop,3)*100;
636             imagemPadDir = zeros(tamanhoAlt,...
637                 paddingDir,3)*100;
638
639             imagemPadInfAlpha = zeros(paddingInf,...
640                 largSpriteCrop);
641             imagemPadDirAlpha = zeros(tamanhoAlt,...
642                 paddingDir);
643
644             imagemSpritePad = cat(1,imagemSpriteCrop,...
645                 imagemPadInf);
646             imagemSpritePad = cat(2,imagemSpritePad,...
647                 imagemPadDir);
648
649             imagemSpritePadAlpha = cat(1,...
650                 imagemSpriteCropAlpha,imagemPadInfAlpha);
651             imagemSpritePadAlpha = cat(2,...
652                 imagemSpritePadAlpha,imagemPadDirAlpha);
653
654             %-------------------------------------------------
655             %   Exibe o resultado parcial para controle
656             %-------------------------------------------------
657
658             if (salvarEtapas)
659                 imwrite(imagemSpritePad,[caminhoSaida ...
660                         'Intermediario_03_Padding\Imagem_'...
661                         num2str(i) '_' confOtica{m} '_' ...
662                         num2str(k) '_01_Padding.png']);
663             end
664
665             %-------------------------------------------------
666             %   ETAPA 04
667             %   Faz o shift para a posição final
668             %-------------------------------------------------
```

```
669
670                 clear imagemSpriteShift;
671                 clear imagemSpriteShiftAlpha;
672
673                 %**** Fora da região de crop (standard) ****
674
675                 shiftSup = deslocY - floor(altSpriteCrop/2);
676                 shiftDir = deslocX - floor(largSpriteCrop/2);
677
678                 if ~((deslocY + ceil(altSprite/2) <= ...
679                         tamanhoAlt) && ...
680                      (deslocX + ceil(largSprite/2) <= ...
681                         tamanhoLarg) && ...
682                      (deslocY > ceil(altSprite/2)) && ...
683                      (deslocX > ceil(largSprite/2)))
684
685                     %**** Dentro da região de crop ****
686
687                     if (deslocY + ceil(altSpriteCrop/2) > ...
688                             tamanhoAlt)
689                         shiftSup = tamanhoAlt - altSpriteCrop;
690                     elseif (deslocY < ceil(altSprite/2))
691                         shiftSup = 0;
692                     end
693                     if (deslocX + ceil(largSpriteCrop/2) > ...
694                             tamanhoLarg)
695                         shiftDir = tamanhoLarg - largSpriteCrop;
696                     elseif (deslocX < ceil(largSprite/2))
697                         shiftDir = 0;
698                     end
699                 end
700
701                 imagemSpriteShift = circshift(imagemSpritePad,...
702                     [shiftSup shiftDir]);
703                 imagemSpriteShiftAlpha = circshift(...
704                     imagemSpritePadAlpha,[shiftSup shiftDir 0]);
705
706                 %-------------------------------------------------
707                 %   Exibe o resultado parcial para controle
708                 %-------------------------------------------------
709
710                 if (salvarEtapas)
```

```
711                     imwrite(imagemSpriteShift,[caminhoSaida ...
712                             'Intermediario_04_Shift\Imagem_' ...
713                             num2str(i) '_' confOtica{m} '_' ...
714                             num2str(k) '_01_Shift.png']);
715                 end
716 
717                 %-------------------------------------------------
718                 %   Determina o tamanho do blur
719                 %-------------------------------------------------
720 
721                 spriteCoC = ((f/1000)^2)/(N(m)*...
722                     (F-f/1000))*abs(1-(F/distancia))*1000;
723                 spriteCoCLista(j) = spriteCoC;
724 
725                 if (~dispSimples)
726                     disp(['Diametro do CoC = ' ...
727                           num2str(spriteCoC) ...
728                           ' (mm)']);
729                 end
730 
731                 blur = ceil(spriteCoC/hs*tamanhoAlt);
732                 blurDesv = blur/2;
733                 blurLista(j) = blur;
734                 blurDesvLista(j) = blurDesv;
735 
736                 %-------------------------------------------------
737                 %   ETAPA 05
738                 %   Gera uma copia da borda da região de blur
739                 %   para uma dilatação igual ao filtro gaussiano
740                 %-------------------------------------------------
741 
742                 if (blur<0);
743                     SE = strel('disk',1);
744                 else
745                     SE = strel('disk',blur*3);
746                 end
747 
748                 imagemSpriteShiftContAlpha = ...
749                     imagemSpriteShiftAlpha;
750                 imagemSpriteShiftGrowAlpha = ...
751                     imdilate(imagemSpriteShiftAlpha, SE);
752 
```

```
753                 imagemSpriteShiftContAlpha = ...
754                     double(imagemSpriteShiftContAlpha);
755                 imagemSpriteShiftGrowAlpha = ...
756                     double(imagemSpriteShiftGrowAlpha);
757                 imagemSpriteShiftGrowAuxAlpha = ...
758                     double(imagemSpriteShiftAlpha);
759
760                 imagemSpriteShiftContAlpha = ...
761                     imagemSpriteShiftContAlpha/...
762                     max(max(imagemSpriteShiftContAlpha));
763                 imagemSpriteShiftGrowAlpha = ...
764                     imagemSpriteShiftGrowAlpha/...
765                     max(max(imagemSpriteShiftGrowAlpha));
766                 imagemSpriteShiftGrowAlphaInv = ...
767                     ones(size(imagemSpriteShiftGrowAlpha))-...
768                     imagemSpriteShiftGrowAlpha;
769
770                 imagemSpriteShiftGrowAuxAlphaInv = ...
771                     ones(size(imagemSpriteShiftGrowAuxAlpha))...
772                     -imagemSpriteShiftGrowAuxAlpha;
773
774                 % Aumenta a borda
775
776                 if size(imagemComposta,3)==1
777                     imagemSpriteShiftGrowBorda = cat(3, ...
778                         imagemComposta, ...
779                         imagemComposta, ...
780                         imagemComposta);
781                 else
782                     imagemSpriteShiftGrowBorda = ...
783                         double(imagemComposta);
784                 end
785
786                 imagemSpriteShiftGrowBorda = cat(3, ...
787                     imagemSpriteShiftGrowBorda(:,:,1) .* ...
788                     imagemSpriteShiftGrowAlpha, ...
789                     imagemSpriteShiftGrowBorda(:,:,2) .* ...
790                     imagemSpriteShiftGrowAlpha, ...
791                     imagemSpriteShiftGrowBorda(:,:,3) .* ...
792                     imagemSpriteShiftGrowAlpha);
793
794                 if (salvarEtapas)
```

```
795                         imwrite(uint8(imagemSpriteShiftGrowBorda),...
796                         [caminhoSaida ...
797                         'Intermediario_05_Expand\Imagem_' ...
798                         num2str(i) '_' confOtica{m} '_' ...
799                         num2str(k) '_01_FundoExpandido.png']);
800                 end
801
802                 % Remove o conteudo real
803
804                 imagemSpriteShiftGrowBorda = cat(3, ...
805                     imagemSpriteShiftGrowBorda(:,:,1) .* ...
806                     imagemSpriteShiftGrowAuxAlphaInv, ...
807                     imagemSpriteShiftGrowBorda(:,:,2) .* ...
808                     imagemSpriteShiftGrowAuxAlphaInv, ...
809                     imagemSpriteShiftGrowBorda(:,:,3) .* ...
810                     imagemSpriteShiftGrowAuxAlphaInv);
811
812                 if (salvarEtapas)
813                         imwrite(uint8(imagemSpriteShiftGrowBorda),...
814                         [caminhoSaida ...
815                         'Intermediario_05_Expand\Imagem_' ...
816                         num2str(i) '_' confOtica{m} '_' ...
817                         num2str(k) '_02_Border.png']);
818                 end
819
820                 imagemSpriteShiftConteudo = double(...
821                     imagemSpriteShift);
822                 imagemSpriteShiftConteudo = cat(3, ...
823                     imagemSpriteShiftConteudo(:,:,1) .* ...
824                     imagemSpriteShiftContAlpha, ...
825                     imagemSpriteShiftConteudo(:,:,2) .* ...
826                     imagemSpriteShiftContAlpha, ...
827                     imagemSpriteShiftConteudo(:,:,3) .* ...
828                     imagemSpriteShiftContAlpha);
829
830                 if (salvarEtapas)
831                     imwrite(uint8(imagemSpriteShiftConteudo),...
832                         [caminhoSaida ...
833                         'Intermediario_05_Expand\Imagem_' ...
834                         num2str(i) '_' confOtica{m} '_' ...
835                         num2str(k) '_03_Content.png']);
836                 end
```

```
837
838                 % Combina os resultados
839                 imagemSpriteShiftConteudo  = ...
840                     uint8(imagemSpriteShiftConteudo);
841                 imagemSpriteShiftGrowBorda = ...
842                     uint8(imagemSpriteShiftGrowBorda);
843
844                 imagemSpriteShiftGrow = ...
845                     imagemSpriteShiftConteudo + ...
846                     imagemSpriteShiftGrowBorda;
847
848                 if (salvarEtapas)
849                     imwrite(imagemSpriteShiftGrow,...
850                         [caminhoSaida ...
851                         'Intermediario_05_Expand\Imagem_' ...
852                         num2str(i) '_' confOtica{m} '_' ...
853                         num2str(k) '_04_Total.png']);
854                 end
855
856                 %------------------------------------------------
857                 %   ETAPA 06
858                 %   Aplica o blur gaussiano
859                 %------------------------------------------------
860
861                 clear imagemSpriteBlur;
862                 clear imagemSpriteBlurAlpha;
863
864                 if (blur > 0)
865                     H = fspecial('gaussian',blur,blurDesv);
866                     imagemSpriteBlur = imfilter(...
867                         imagemSpriteShiftGrow,H,'replicate');
868                     imagemSpriteBlurAlpha = imfilter(...
869                         imagemSpriteShiftAlpha,H,'replicate');
870                 else
871                     imagemSpriteBlur = imagemSpriteShiftGrow;
872                     imagemSpriteBlurAlpha = ...
873                         imagemSpriteShiftAlpha;
874                 end
875
876                 %------------------------------------------------
877                 %   Exibe o resultado parcial para controle
878                 %------------------------------------------------
```

```
879
880                 if (salvarEtapas)
881                     imwrite(imagemSpriteBlur,[caminhoSaida ...
882                             'Intermediario_06_Blur\Imagem_' ...
883                             num2str(i) '_' confOtica{m} '_' ...
884                             num2str(k) '_01_Blur.png']);
885                 end
886
887                 %-------------------------------------------------
888                 %   Normaliza os canais alpha
889                 %-------------------------------------------------
890
891                 imagemSpriteBlurAlpha = double(...
892                     imagemSpriteBlurAlpha);
893                 imagemSpriteBlurAlpha = imagemSpriteBlurAlpha/...
894                     max(max(imagemSpriteBlurAlpha));
895                 imagemSpriteBlurAlphaInv = ones(size(...
896                     imagemSpriteBlurAlpha))-...
897                     imagemSpriteBlurAlpha;
898
899                 %-------------------------------------------------
900                 %   Conversão para double para as operações
901                 %-------------------------------------------------
902
903                 imagemSpriteBlur = double(imagemSpriteBlur);
904
905                 %-------------------------------------------------
906                 %   Sobrepõe as imagens novas
907                 %-------------------------------------------------
908
909                 %**** Imagem ground truth ****
910                 if (spriteCoC <= cocLimite)
911                     imagemComponenteGT  = 255*...
912                         imagemSpriteBlurAlpha;
913                     imagemCompostaGT = imagemCompostaGT  + ...
914                         imagemComponenteGT ;
915                 end
916
917                 %**** Imagem convencional ****
918                 imagemComponente = cat(3, ...
919                     imagemSpriteBlur(:,:,1) .* ...
920                     imagemSpriteBlurAlpha, ...
```

```
921                     imagemSpriteBlur(:,:,2) .* ...
922                     imagemSpriteBlurAlpha, ...
923                     imagemSpriteBlur(:,:,3) .* ...
924                     imagemSpriteBlurAlpha);
925
926                 imagemComposta = cat(3, ...
927                     imagemComposta(:,:,1) .* ...
928                     imagemSpriteBlurAlphaInv, ...
929                     imagemComposta(:,:,2) .* ...
930                     imagemSpriteBlurAlphaInv, ...
931                     imagemComposta(:,:,3) .* ...
932                     imagemSpriteBlurAlphaInv);
933
934                 imagemComposta = imagemComposta + ...
935                     imagemComponente;
936
937             end
938         end
939
940         %-----------------------------------------------------------
941         %   Salva a imagem
942         %-----------------------------------------------------------
943
944         imagemComposta = uint8(imagemComposta);
945         imwrite(imagemComposta,[caminhoSaida 'Imagem_' ...
946             num2str(i) '_' confOtica{m} '.png']);
947
948         imagemCompostaGT = uint8(imagemCompostaGT);
949         imwrite(imagemCompostaGT,[caminhoSaida 'Imagem_' ...
950             num2str(i) '_' confOtica{m} '_GroundTruth.png']);
951
952         %-----------------------------------------------------------
953         %   Salva uma cópia da imagem com as informações de
954         %   cada sprite, além de salvar um arquivo de texto
955         %   separado
956         %-----------------------------------------------------------
957
958         if (numeroSprites(i) > 0 && N(m) >= 0)
959
960             % Cria a imagem com comentários
961             imagemCompostaTexto = imagemComposta;
962
```

```
963             % Cria o arquivo de texto
964             arquivoInfo = fopen([caminhoSaida ...
965                                  'Suporte\Imagem_' num2str(i) '_' ...
966                                  confOtica{m} '_Info.txt'], 'w', ...
967                                  'n', 'UTF-8');
968             for k=1:numeroSprites(i)
969
970                 j = listaDistanciasIndex(k);
971                 ID = listaSprites{i}(j);
972
973                stringInfoShort = sprintf(['Nome = %s\n' ...
974                                           'Coord. = (x=%d,y=%d)\n' ...
975                                           'Tam. real = %d (mm)\n' ...
976                                           'Dist. = %d (m)\n' ...
977                                           'CoC = %.4f (um)'], ...
978                                           arquivosSprites{...
979                                           listaSprites{i}(j)}, ...
980                                           posXSprite{i}(j), ...
981                                           posYSprite{i}(j), ...
982                                           tamanhoRealSprite(...
983                                           ID), ...
984                                           distanciaSprite{i}...
985                                           (j), ...
986                                           spriteCoCLista(j)*1000);
987
988                 % Cria o texto de conteúdo completo
989                 stringInfo = sprintf(['Nome = %s\r\n' ...
990                             'Codigo = %d\r\n' ...
991                             'Coord. = (x=%d,y=%d)\r\n'...
992                             'Tam. real = %d (mm)\r\n' ...
993                             'Tam. no sensor = %0.3f (mm)\r\n' ...
994                             'Dist. = %d (m)\r\n' ...
995                             'Escala = %.2f\r\n' ...
996                             'CoC = %.4f (um)\r\n' ...
997                             'Gauss. Tam. = %d\r\n' ...
998                             'Gauss. Desv. = %.2f'], ...
999                             arquivosSprites{...
1000                            listaSprites{i}(j)}, ...
1001                            j, ...
1002                            posXSprite{i}(j), ...
1003                            posYSprite{i}(j), ...
1004                            tamanhoRealSprite(ID), ...
```

```
1005                                tamanhoSensorLista(j), ...
1006                                distanciaSprite{i}(j), ...
1007                                escalaSpriteLista(j), ...
1008                                spriteCoCLista(j)*1000, ...
1009                                blurLista(j), ...
1010                                blurDesvLista(j));
1011
1012                    % Adiciona informação à imagem
1013                    imagemCompostaTexto = insertText(...
1014                        imagemCompostaTexto,[posXSprite{i}(j) ...
1015                        posYSprite{i}(j)],stringInfoShort,...
1016                        'FontSize',12,'BoxColor','red',...
1017                        'BoxOpacity',0.7,'TextColor','white');
1018                    imwrite(imagemCompostaTexto,[caminhoSaida ...
1019                                        'Suporte\Imagem_' num2str(i)
                                               '_' ...
1020                                        confOtica{m} '_Info.png']);
1021
1022                    % Adiciona ao arquivo de texto
1023                    fprintf(arquivoInfo,['----------------------'...
1024                        '--------------------\r\n']);
1025                    fprintf(arquivoInfo,'   SPRITE %d\r\n',k);
1026                    fprintf(arquivoInfo,['----------------------'...
1027                        '--------------------\r\n']);
1028                    fprintf(arquivoInfo,stringInfo);
1029                    fprintf(arquivoInfo,'\r\n \r\n');
1030
1031                end
1032
1033                % Fecha o arquivo de texto
1034                fclose(arquivoInfo);
1035            end
1036        end
1037    end
1038
1039    %-----------------------------------------------------------------
1040    % Finaliza a execução do algoritmo
1041    %-----------------------------------------------------------------
1042
1043    disp('-------------------------------------------------');
1044    disp('   EXECUÇÃO TERMINADA!   ');
1045    disp('-------------------------------------------------');
```