UNIVERSIDADE FEDERAL DO AMAZONAS – UFAM
INSTITUTO DE COMPUTAÇÃO – ICOMP
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

ANDERSON GADELHA FONTOURA

TÉCNICA APRIMORADA DE SEGMENTAÇÃO NÃO-SUPERVISIONADA EM IMAGENS COM FELINOS DOMÉSTICOS

MANAUS – AM

2016

ANDERSON GADELHA FONTOURA

TÉCNICA APRIMORADA DE SEGMENTAÇÃO NÃO-SUPERVISIONADA EM IMAGENS COM FELINOS DOMÉSTICOS

Dissertação desenvolvida pelo discente Anderson Gadelha Fontoura, com o intuito de obtenção da aprovação final e obtenção do título de Mestre em Informática, oferecido pelo Instituto de Computação (ICOMP/UFAM).

Orientador: Prof. Dr. José Reginaldo H. Carvalho

MANAUS – AM

2016

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Fontoura, Anderson Gadelha
F684t Técnica aprimorada de segmentação não-supervisionada em imagens com felinos domésticos / Anderson Gadelha Fontoura. 2016
132 f.: il. color; 31 cm.

Orientador: José Reginaldo Hughes Carvalho
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Felinos Domésticos. 2. Watershed. 3. Filtro Color Boost. 4. Segmentação Não-supervisionada. 5. Filtro Homomórfico. I. Carvalho, José Reginaldo Hughes II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"TÉCNICA APRIMORADA DE SEGMENTAÇÃO NÃO-SUPERVISIONADA EM IMAGENS COM FELINOS DOMÉSTICOS"

ANDERSON GADELHA FONTOURA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. José Reginaldo Hughes Carvalho - PRESIDENTE

Profa. Eulanda Miranda dos Santos - MEMBRO INTERNO

Prof. José Pinheiro de Queiroz Neto - MEMBRO EXTERNO

Manaus, 19 de Junho de 2016

DEDICATÓRIA

A minha esposa Priscilla, que apesar de todos os sacrifícios, sempre me apoiou nessa conquista e dedicou sua vida pelo meu sucesso. A ela, meu eterno amor.

AGRADECIMENTO

Agradeço a ajuda prestimosa de meu orientador, José Reginaldo Hughes, pela paciência, confiança e carinho com que sempre me acolheu;

Agradeço aos meus professores, Eulanda, Pio, Souto, Altigran, Ricardo e Marlene, por saberem como me apoiar e ajudar no caminho correto de meus estudos;

Agradeço aos meus amigos, Adria, Janaina, Marcus, Daily Daleno, Caio, Delano, Bernardo e Michel. Pelo incentivo e as palavras amigas (bem como os espaireceres) nos momentos difíceis.

Obrigado a todos vocês.

"Lembre-se ... não custa nada encorajar um artista. Os benefícios disso são incríveis e os resultados, bons ou ruins, sempre serão alcançados.

Desencoraje-o e você não receberá nada em troca, nunca."

Kevin Smith

Cineasta e roteirista

1970 -

RESUMO

Muitos trabalhos atuais têm como foco principal a preservação da fauna e flora através do monitoramento e de pesquisas centradas em regiões com ecossistemas bem diversos, como é o caso da Amazônia. Pesquisas sobre monitoramento de animais sempre são realizadas em diversas partes do mundo. O problema principal deste tipo de monitoramento é que sua catalogação ainda é realizada de forma manual, consumindo o tempo dos pesquisadores que poderia ser melhor utilizado no alcance dos objetivos das pesquisas propriamente ditas. Na Austrália por exemplo, a falta de monitoramento em diversas espécies de felinos, principalmente gatos domésticos, preocupa cientistas devido a tomada de decisões errôneas por parte dos governos, que deseja combatê-los como se realmente fossem pragas. No Brasil, pesquisas similares são realizadas para prover a melhor conservação das espécies de felinos selvagens. Nesse contexto, o objetivo deste trabalho é de colaborar nessa área com o estudo de reconhecimento de padrões e processamento digital de imagens para a construção de um método mais eficaz de segmentação de um animal, em especial: o felino doméstico. O método consiste na criação de um processo combinado de um filtro de aumento de contraste *Color Boost*, filtro homomórfico, filtro *Mean-Shift* e do Mapa de distância para conseguir de forma não-supervisionada, segmentar o felino em uma cena. Além de conter uma regra para diminuir o processo de sobre segmentação em imagens, que é muito comum em segmentadores do tipo *Watershed*. Os resultados conseguem alcançar até 84% em média de exatidão na extração do felino, tendo a possibilidade de no futuro ser extrapolado para outros objetos ou espécies.

Palavras-chaves: Felinos Domésticos. *Watershed.* Filtro *Color Boost.* Segmentação Não-supervisionada. Filtro Homomórfico.

ABSTRACT

A great number of recent projects have, as their focus, the preservation of fauna and flora through monitoring and research centered on regions with very heterogenic ecosystems, such as the Amazon Rain Forest. In fact, research projects based on animal monitoring are carried out in various parts of the world. The main problem of this type of monitoring lies on the cataloguing aspect that is still completed manually, consuming precious time of the researchers which could be better used in truly achieving the objectives of the research. As an example, in Australia, the lack of monitoring in several species of felines, especially domestic cats, is a concern of scientists because of questionable decisions made by governments that consider these animals as pests and treat them as a menace for environment balance. In Brazil, similar researches are conducted in order to maintain conservation of wild cats, such as jaguars. In this context, the objective of this work is to collaborate in this area with the study of pattern recognition and digital image processing in order to build a more effective method for animal segmentation in pictures, particularly the domestic cat. This method consists in creating a combined process that integrates a contrast enhance Color Boost filter, homomorphic filter, Mean- Shift filter and Distance Map in order to achieve an unsupervised way for segmenting cats on picture scenes. In addition to this method, a merge rule for decreasing the process of over-segmentation in images is applied, avoiding this common issue in many Watershed algorithms. The results can reach up to 84% on average accuracy in feline segmentation, with the possibility, in the future, to be extrapolated to others objects or species.

Keywords: Domestic Felines. Watershed. Color Boost Filter. Unsupervised Segmentation. Homomorphic Filter.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE PSEUDOCÓDIGOS

LISTA DE SIGLAS E EXPRESSÕES

ABNT	Associação Brasileira de Normas Técnicas
AMBE	*Absolute Mean Brightness Error* ou Erro Médio Absoluto de Brilho
aMSRM	*Advanced Maximal Similarity-based Region Merging*
BoW	*Bag-of-Words*
CLAHE	*Contrast-Limited Adaptive Histogram Enhancement*
DCT	*Discrete Cosine Transform* ou Transformada de Cossenos Discretos
DFT	*Discrete Fourier Transform*
DSLR	*Digital Single-Lens Reflex Camera*
EME	*Enhancement Measure*
FFT	*Fast Fourier Transform*
FPR	*False Positive Rate*
GB	GigaBytes
GHz	GigaHertz
GMM	*Gaussian Mixture Models* ou Modelo de Mistura Gaussiana
GT	*Ground Truth*
H	*Entropy*
HD	*Hard Drive* ou Disco Rígido
HDR	*High Dynamic Range* ou Grande Alcance Dinâmico
HSV	*Hue, Saturation, Value* ou Matiz, Saturação, Valor
ICMBio	Instituto Chico Mendes de Conservação da Biodiversidade
IComp	Instituto de Computação
IDCT	*Inverse Discrete Cosine Transform* ou Transformada Inversa de Cossenos Discretos

IDE	*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado
IDSM	Instituto de Desenvolvimento Sustentável Mamirauá
INAN	Instituto de Apoio ao Desenvolvimento e à Preservação da Natureza
IOP	Instituto Onça Pintada
ISO	*International Organization of Standardization*
JPEG	*Joint Photographic Experts Group*
MSMR	*Maximal Similarity-based Region Merging*
OC-SVM	*One-Class Support Vector Machines*
p.e.	Por exemplo
PDF	*Probably Density Function*
PDI	Processamento Digital de Imagens
PNG	*Portable Network Graphics*
PSNR	*Peak Signal-Noise Ratio* ou Relação Sinal-Ruído de Pico
px	*Pixel*
RAM	*Random Access Memory* ou Memória de Acesso Aleatório
RGB	*Red, Green, Blue* ou Vermelho, Verde, Azul
RW	*Random Walker*
RWR	*Random Walker with Restart*
s.ed.	Sem Editora
s.l.	Sem Local
SIFT	*Scale-Invariant Feature Transform*
sRGB	*standard* RGB
SSD	*Solid State Drive* ou Disco de Estado Sólido
SURF	*Speed Up Robust Features*
SVM	*Support Vector Machines*

TPR	*True Positive Rate*
VC	Visão Computacional

SUMÁRIO

INTRODUÇÃO

Inúmeros problemas do estado da arte em visão computacional derivam de situações que necessitam de aplicações inteligentes para o reconhecimento e detecção de formas e padrões. Diferentemente do sistema de visão humano que consegue detectar estas características, os sistemas de visão computacionais ainda carecem de ferramentas que eficientemente lidem com diferentes contextos, iluminação, oclusão, movimento, dentre outros desafios, fazendo deste um dos campos de pesquisa mais estudados da computação. A questão central desta abordagem é: como um computador pode processar informação visual com a mesma eficiência do ser humano?

Uma das abordagens usuais é a de combinar o processamento de imagem e vídeo com técnicas de inteligência artificial. Dentre as subáreas da Inteligência Artificial, um dos campos que tem sido alvo de interesse é a de **aprendizado de máquina** (THEODORIDIS e KOUTROUMBAS, 2009). Quando combinadas, as áreas de processamento digital de sinais e imagens, produzem aplicações interessantes tanto no campo de **visão computacional** como no **reconhecimento/classificação de padrões**.

De acordo com os trabalhos de He, Kim e Kuo (2014), o reconhecimento de padrões cresceu exponencialmente nos últimos 16 anos. Vertentes e aproximações novas sobre esses processos, como os vistos nos trabalhos de Crall *et al.* (2013) e Grady (2006), por exemplo, são bem recentes e contém fortes aplicações práticas e teóricas no mercado atual, principalmente nas áreas de medicina e vigilância. Sendo comum encontrar nestes e em outros trabalhos aplicados a **segmentação** e o **reconhecimento de objetos** em imagens (onde imagem neste trabalho é a projeção em duas dimensões da cena).

Por isso, aplicações médicas (como a Tomografia Computadorizada) e aplicações de vigilância (como imagens aéreas), por exemplo, vêm utilizando técnicas diversas

para a identificação mais rápida e precisa de formas, objetos, áreas e outros tipos de classificações. Sabe-se que o ser humano é capaz de fazê-lo apenas observando, porém, com a ajuda do computador, o processo acelera e pode prover detalhes adicionais que podem estar "ocultos" aos olhos humanos (Gonzalez e Woods, 2010). O *range* de observação de sensores visuais hoje em dia supera, e muito, a visão humana em termos de detalhes (onde detalhe é um conceito que generaliza a resolução).

Contudo, é óbvio que o processo de classificação e identificação de objetos em cenas não é um processo simples e dependerá de muitos passos anteriores. Conforme Duda *et al.* (2001), um dos processos importantes para a aceleração da classificação de elementos na cena é o processo de segmentação de imagens. É nesta etapa que o algoritmo de classificação poderá ganhar tempo em execução pela simples "eliminação" de partes da cena que não devem significar muito ao conteúdo que se deseja extrair.

A classificação de objetos é importante e deve ser mencionada. Simplesmente porque a ideia de classificar objetos em cena pode ser facilmente aplicada para problemas de reconhecimento de animais e outros elementos que necessitam de uma minuciosa observação, o que acarreta na realização de um trabalho puramente manual. Além disso, este trabalho é desenvolvido numa das áreas geográficas que mais recebe atenção do ponto de vista do planeta sustentável – a Amazônia. Sua proteção e preservação estão no centro de interesse de pesquisadores, biólogos, geógrafos e engenheiros de todo mundo e depende da monitoração e policiamento. A ciência da computação tem investido o seu conhecimento para prover ferramentas que ajudem a alcançar esse objetivo. Neste contexto o processamento de imagens digitais provê várias ferramentas importantes para a vigilância aérea, o reconhecimento de características, e várias outras técnicas em que a automatização acelerou o processo de reconhecimento. Essas aplicações têm resultado em avanços importantes para a compreensão desse ecossistema, bem como uma melhor preservação da fauna e da flora das mãos de predadores humanos (IOP, 2015).

Atualmente, na Amazônia existem vários institutos de conservação ambiental, tais como: o Instituto Chico Mendes de Conservação da Biodiversidade (ICMBio), o Instituto de Apoio ao Desenvolvimento e à Preservação da Natureza (INAN), o Instituto de Desenvolvimento Sustentável Mamirauá (IDSM), etc. Este último, tem como um de seus propósitos, em conjunto com biólogos e outros profissionais, o de compreender o comportamento de espécies de animais nas áreas de várzea da região amazônica (próximo a Tefé-AM), que estão ameaçadas pela fragmentação de seu *habitat* (IOP, 2015). Por isso, a classificação de objetos, que neste caso seriam elementos da fauna e da flora amazônica, ajudam a acelerar o processo de análise desses institutos em ordem de prover um ambiente natural de proteção para a proliferação e sucesso dessas espécies.

Apesar de um estudo sobre essas espécies selvagens ser prioritário, o estado do Amazonas também possui problemas relacionados a espécies de animais urbanas abandonados aos milhares. Cerca de 200 mil cães e gatos residem hoje nas ruas de Manaus (VASCONCELOS; G1, 2014; 2015). Não somente o Amazonas, mas em todo o Brasil, em torno de 1% dentre aproximadamente 60 milhões de cães e gatos serão abandonados até 2016, somando-se aos já 30 milhões de "animais de rua" (JUSBRASIL, 2014). Estes animais ocasionalmente podem carregar doenças, porém, não é comum serem os causadores de catástrofes ambientais (HAIDAR, 2015).

No Brasil, esse problema urbano aparenta ser tratado com desinteresse pelos líderes de governo e população em geral, porém na Austrália em 2015, o ministro de meio ambiente, Greg Hunt, afirmou que o governo australiano investirá em torno de 5 milhões de dólares para erradicar 2 milhões de "gatos de rua" até 2020 (LINKANIMAL, 2015). Os motivos demonstram ser plausíveis, pois pesquisas apontam que os gatos já exterminaram cerca de 27 espécies de pequenos mamíferos e de pássaros nativas da Austrália. Entretanto, diversos biólogos e veterinários especialistas no assunto, afirmam que isso é errôneo, pois o governo australiano baseia-se apenas em informação dadas

por evidências anedóticas fornecidas pela população, que considera todos os gatos como prejudiciais ao ambiente local (HAIDAR, 2015). Segundo o biólogo John Woinarski (FRANK *et al.*, 2014), os principais causadores desses problemas são os gatos "ferais", felinos que não convivem com seres humanos, e que o governo australiano estaria baseando-se em conclusões equivocadas para tornar os "gatos de rua" culpados, de maneira a justificar seu extermínio. O referido biólogo ainda afirma que o esforço será inútil a longo prazo e que o correto seria investir em formas de estudo e monitoração destes animais para entender se estes são realmente os causadores destes impactos ou se o desequilíbrio é maior do que pensam (HAIDAR, 2015). O problema torna-se ainda mais grave, quando outras nações, como Estados Unidos e Alemanha, veem estes atos como exemplos a serem seguidos e discutem a utilização de procedimentos semelhantes em suas áreas urbanas (VEJA, 2015). Contudo, os Estados Unidos realmente possuem um problema sério a respeito de gatos selvagens (LOSS *et al.*, 2013), o que mostra que os mesmos têm exterminado cerca de 4 bilhões de aves e 23 bilhões de pequenos mamíferos ao ano.

Cientistas entendem que o extermínio poderá diminuir a competição natural, fazendo com que o problema aumente e, por isso, buscam conhecer esses animais de forma a observá-los à distância (devido ao seu comportamento), utilizando armadilhas com comida e usando câmeras especiais que detectam movimentos e tiram fotos de forma automática (FRANK *et al*, 2014). Com esses dados, muito indivíduos são catalogados e analisados pela sua fisiologia e sua extensão territorial, além de terem seu comportamento analisado para futuras referências. O processo é parecido com o trabalho do IDSM para o monitoramento de onças pintadas, porém, assim como de outros monitores, essas aplicações e experimentos tomam um longo tempo para se catalogar os resultados e, quase em sua totalidade, são feitos de forma manual.

De acordo com as afirmações expostas anteriormente, este trabalho propõe um processo de detecção de animais, começando por gatos e restringindo-se à fase de

segmentação, contudo, de forma não-supervisionada. O objetivo é de colaborar com outros processos que identifiquem e melhorem a rapidez e eficácia na catalogação de imagens com gatos ou outros animais. A escolha dos gatos segue dois motivos iniciais: 1) o IDSM e outros institutos poderão usar os resultados desse experimento para extrapolar os benefícios e classificar animais de interesse como, por exemplo, a onça pintada, o maior felino do Brasil; e 2) estas aplicações poderão ajudar no reconhecimento de animais, como gatos, para o ajudar os governos a tratar problemas de zoonoses e evitar extermínio de outros animais, com análises críticas baseadas em evidências científicas. Fora isso, o processo de segmentação não supervisionada é importante para melhorar a compreensão de algoritmos para uma segmentação quase perfeita no futuro (HE, KIM e KUO, 2014).

Nesse contexto, será levantado um banco de dados de pelo menos cinco indivíduos e felinos domésticos (Figura I.1) para conceber o algoritmo de melhoria na segmentação desses animais.

Figura I.1 – Imagens de felinos domésticos para análise.

Fonte: Próprio Autor, 2016.

O processo de segmentação dos gatos terá a possibilidade de ajudar a catalogar esses indivíduos no futuro, visto que a etapa da segmentação irá diminuir as chances de erros na hora da extração de características desses animais, além de reduzir o tempo de análise pelo algoritmo (HE, KIM e KUO, 2014). Existe também a hipótese apresentada por Merritt (2009), onde partes do corpo do indivíduo caracterizam quais

espécies de felinos poderão estar presentes em uma cena (Figura I.2), evidenciando assim qual espécie poderá estar associada ao monitoramento e, consecutivamente, aos impactos ambientais já citados.

Figura I.2 – Regiões de possíveis características (*features*) usadas para reconhecer felinos domésticos.

Fonte: Próprio Autor, 2016.

Apesar deste trabalho propor uma solução para o problema de segmentação, a capacidade atual de compreensão dos sistemas de segmentação não-supervisionada está longe da perfeição. Neste trabalho, o processo apenas trata de uma técnica de melhoria da segmentação automática para a análise posterior dos indivíduos em uma cena. Dessa forma, gerando uma aceleração do processo de catalogação desses animais que poderá prover uma nova abordagem dentro da segmentação de animais que sejam tão expressivos como das onças pintadas ou mesmo outros animais.

1 CONCEITUAÇÃO E MOTIVAÇÃO DO TRABALHO

Este trabalho apresenta o problema de segmentação de áreas e extração de características automáticas de imagens em cenas que contém gatos domésticos. Um processo complicado de ser aplicado sem a supervisão do usuário, mesmo em áreas bem estudadas. A aplicação torna-se interessante tanto pelo ineditismo quanto pelo desafio de prover um processo cientificamente embasado, onde a análise de gatos possa realmente ser próxima ao processo manual e humano. Por isso, a ideia de prover um algoritmo que possa ajudar na segmentação não-supervisionada, tem mais peso do que a segmentação desses felinos em si e dará no futuro a oportunidade de estudar como generalizar o processo a fim de expandir a caracterização da pesquisa a outros tipos de animais e objetos.

1.1 JUSTIFICATIVA

Tendo em vista o tema e a delimitação deste trabalho, o conceito de reconhecimento automático de felinos é um dos inúmeros problemas que os biólogos do IDSM vêm enfrentando ao longo de vários anos. No mundo, o processo de análise de gatos selvagens também gera um trabalho exaustivo na coleta de dados e estudo desses animais. Os biólogos possuem o trabalho, geralmente manual, de realizar a seleção e classificação desses animais capturados por câmeras em armadilhas. Além do estudo do comportamento de cada indivíduo animal, os biólogos devem realizar o trabalho de separar (e catalogar) todas as imagens que podem ser usadas em suas pesquisas.

Soma-se ao contexto o problema dos “gatos de rua”, que tem potencial de causar danos ao ambiente e serem fontes de importantes zoonoses. Por este motivo, o

monitoramento e a classificação dessas espécies são de grande importância para tomadas de decisões mais elaboradas do que a eutanásia ou a simples castração.

Levando para o lado computacional, a proposta de construir um processo não-supervisionado não é nova, porém, de acordo com Gonzalez e Woods (2010), He, Kim e Kuo (2014), Ning *et al.* (2010), Duda *et al.* (2001), e praticamente todos os autores e suas obras pesquisadas, informam que o campo desses processos é altamente interessante e demanda com certa urgência a criação de novas técnicas que ajudem a compreensão e surgimento de novos segmentadores para atingir no futuro, uma segmentação automática próxima a 100% de eficácia.

Por isso, como uma forma de contribuir com a área, este trabalho apresenta uma ferramenta computacional de segmentação válida capaz de segmentar de forma não-supervisionada o animal na cena, com um desempenho comparável com os métodos supervisionados estudados.

1.2 PROBLEMAS ABORDADOS

Os problemas mais observados na concepção da futura aplicação são descritos a seguir:

1. Como o processo de segmentação deve ser aplicado de forma automática e porquê?
2. A segmentação deve ser aplicada para melhorar uma futura classificação?
3. Que porção do corpo do objeto precisa ser analisada ou segmentada para o posterior casamento de padrões satisfatório?
4. As imagens do banco de dados a ser analisado neste trabalho conterão somente felinos ou não?

5. Essas imagens estão em boa qualidade (para a aplicação dos conceitos posteriores de classificação)?
6. Devem ser comparados vários segmentadores ou será necessário desenvolver um segmentador próprio para extrair o objeto da cena?

Diante destes pressupostos, pode-se levantar hipóteses para as possíveis soluções na próxima seção.

1.3 HIPOTESES LEVANTADAS

Com base nos problemas analisados e o banco de dados criado, pode-se propor as seguintes hipóteses:

1. É importante principalmente devido ao fato da ausência de métodos não-supervisionados e a necessidade da criação de novas ferramentas que ajudem os pesquisados em campo a melhorar o monitoramento automático de objetos (fauna e flora), o qual ainda apresenta muito trabalho manual. A proposta de segmentação automática deverá ainda ter duas vertentes: usando um casamento de aplicações com a transformada da distância e/ou filtro Gabor (para análise de texturas), baseado em um conjunto de imagens e/ou, via segmentadores adaptados, como filtro de Kalmann ou segmentadores Markovianos;
2. Sempre que o animal estiver com boa parte de seu corpo na imagem, a segmentação pode ser feita com alta confiabilidade e reduzido esforço computacional, visto que nas imagens obtidas do banco de dados, o corpo do felino é bem contrastante em relação ao plano de fundo (*background*). Adicionalmente, é possível aplicar técnicas com a subtração do fundo em

relação ao animal que poderão ajudar bastante a parte de extração de características;

3. Torso, orelhas e cauda do felino (objeto) geram os principais descritores na hora de identificar a raça do animal. Esta hipótese é fundamentada no trabalho de Merritt (2009);
4. Para definir um escopo mais coerente, todas as imagens analisadas neste trabalho conterão gatos domésticos. Contudo, vários cenários diferentes serão preparados para testar se a técnica de segmentação automática consegue ser capaz de "extrair" a maior porção do gato na cena, tanto em cenário *indoor* quanto *outdoor*;
5. Para resolver problemas de qualidade, serão testadas várias aplicações de realce de imagens e depois comparadas aos modelos comuns de segmentação. Se houver significância (melhores resultados), será desenvolvida uma etapa anterior ao processo de segmentação puramente dito, que será o realce da imagem. Visto que muitas das imagens são retiradas sem flash ou com péssima iluminação ambiente, procedimentos de uniformização de luz, como a filtragem homomórfica, serão sempre utilizadas;
6. A *priori*, serão usados os segmentadores *Watershed* e *GrabCut*. O método que será desenvolvido proverá apenas um processo inicial de pré e pós segmentação, afim de aumentar a eficiência do segmentador quando utilizado em um método automático.

Com base nestas hipóteses, pode-se propor os objetivos para este trabalho.

1.4 OBJETIVOS

1.4.1 GERAL

- Propor uma técnica de segmentação automática adaptada a felinos domésticos, usando um banco de dados já existente, de forma que o procedimento consiga separar a maior parte do indivíduo da cena sem a supervisão do usuário, aplicando-se principalmente técnicas de Processamento Digital de Imagens (PDI) e Visão Computacional (VC).

1.4.2 ESPECÍFICOS

A. Definir um conjunto de métodos de melhoria de imagens para melhorar a confiabilidade de criação automática de marcadores (ou *seeds*);

B. Produzir uma análise entre os segmentadores ***Watershed*** e/ou ***GrabCut*** para determinar a capacidade de segmentação dos objetos na imagem.

C. Prover uma técnica de forma genérica e com baixa variância de resultados positivos, a ponto de permitir que o método usado seja eficiente na maior parte das imagens utilizadas.

1.5 ESTRUTURA DO TRABALHO

A formatação deste trabalho segue as normas da ABNT NBR 14724 (ABNT, 2011) e os conselhos de texto científico explanados por Furasté (2013) e Prodanov e Freitas (2013).

Este trabalho será dividido nos seguintes cinco capítulos mais um apêndice, além da introdução e deste primeiro capítulo, já apresentados:

- O capitulo dois apresenta a fundamentação teórica, explicando de forma clara os conceitos fundamentais do Processamento Digital de Imagens (PDI) e a Visão Computacional (VC), e dos processos que levam a prover melhorias na segmentação de objetos em imagens digitais;
- O capitulo três descreve os trabalhos relacionados e correlatos com as características que poderão ser usadas para aplicar o método proposto e auxiliar o leitor a compreender a metodologia da proposta e seu fundamenta na pesquisa cientifica;
- O capitulo quatro demonstra o método de resolução do problema exposto nos capítulos anteriores, em forma definitiva, para gerar a técnica de segmentação não-supervisionada, bem como os métodos de aquisição das principais fontes pesquisadas;
- O capitulo cinco discute os resultados obtidos, bem como as comparações com outros trabalhos correlatos, de forma analítica, com o intuito de prover respaldo cientifico sobre o tema proposto;
- O capítulo final conclui o trabalho, mostrando os pontos finais, objetivos específicos concluídos e as limitações do método. O capítulo de conclusão sugere também projetos futuros e as extrapolações dos resultados para os felinos selvagens.
- Apêndice A contém o código fonte do filtro *Color Boost* em duas versões: a versão para intensificação da energia (*Color Boost*) e a versão para melhoria de cor (*Active Color Boost*).

2 FUNDAMENTOS SOBRE A SEGMENTAÇÃO E EXTRAÇÃO DE CARACTERÍSTICAS EM IMAGENS

Neste capitulo será demonstrado um resumo das principais informações do estudo sobre segmentação e processamento digital de imagens ao leitor. Com o simples intuito de facilitar o entendimento de termos e tópicos dos próximos capítulos.

2.1 O QUE É A SEGMENTAÇÃO?

De acordo com Solomon e Breckon (2013), a segmentação é o processo de **subdividir** uma imagem em regiões ou objetos constituintes bem delimitados. Os autores ainda afirmam que o processo de segmentação autônoma é uma das tarefas mais difíceis dentro do campo de visão computacional e que ainda se encontra muito distante de ser perfeita (não por falta de processamento e sim, pela falta de compreensão que os cientistas têm sobre o assunto). Por isso, esse campo é muito ativo para a melhoria de técnicas e outros processos de segmentação.

Segundo Duda *et al.* (2001), a segmentação é um **processo intermediário** da classificação de padrões e desempenha um papel importante no processamento digital de imagens, constituindo-se no **primeiro passo vital** (SOLOMON e BRECKON, 2013) a ser aplicado anteriormente a uma série de processos de reconhecimento de padrões (como por exemplo extração de características, classificação, descrição, etc.).

De certa forma, a aplicação da segmentação não é "inicial", mas sim **intermediária** (PEDRINI e SCHWARTZ, 2008), pois sempre é necessário passar a entrada de dados por métodos de filtragem espacial e de frequência de forma anterior ao estágio de segmentação. Com o objetivo claro de remover ruídos ou até mesmo correção/estimação de certos valores, como cor ou textura. Ruídos que por sua vez, podem levar os métodos de segmentação à distorção das formas dos objetos, que são

valores desejados à extração de características, que prejudicarão o trabalho dos descritores.

Por isso, como outra definição (HE, KIM e KUO, 2014), a segmentação serve para **particionar um conjunto de dados** (pixels ou voxels) em estruturas com conteúdo semântico relevantes, que já foram submetidas a aplicações de filtragem e realce para validar suas aplicações (e obviamente, sua importância).

Em suma, os métodos de segmentação podem ser separados em duas vertentes: **supervisionados** - quando há regiões da imagem que dispõem de informações que permitam a identificação de uma ou mais classes ou objetos; e **não-supervisionados** - quando o processo de segmentação depende de uma associação de classes que são aprendidas por meio de um banco de dados e são comparadas por similaridade de características (*features*) usadas em um processo de treinamento para poder classificar objetos presentes na imagem. Geralmente este último processo depende de técnicas de reconhecimento de padrões, como *Support Vector Machines* (SVM) e classificadores Bayes, e podem ser automáticos ou semiautomáticos (DUDA *et al.*, 2001).

Ainda conforme Solomon e Breckon (2013) e Pedrini e Schwartz (2008), os processos de segmentação podem ainda ser colocados em outras subcategorias: **os métodos de contorno, fronteira ou descontinuidade** - que usam a abordagem da detecção de bordas como meio de identificação de fronteiras entre regiões. Essa abordagem busca, portanto, grandes diferenças entre grupos de pixels e ainda pode, em alguns casos, necessitar de uma análise de aglomerados de pixels (*clusterização*). Aplicações comuns aqui são o **Sobel** (1ª ordem) e **Operadores Laplacianos** (2ª ordem).

Os métodos baseados em regiões ou por similaridade buscam alocar um conjunto de pixels a uma determinada região com base no grau de similaridades entre eles, ou de forma análoga, determinar a relação que um pixel tem com o seu vizinho e

com outros pixels da imagem (local e/ou global). Provavelmente o representante mais conhecido deste tipo de método é a **Transformada *Watershed***.

De acordo com He, Kim e Kuo (2014), ainda existem dois outros métodos: os por ***Graph-Cut***, que se baseiam em um processo interativo onde um usuário fornece dicas sobre objetos que devem ser segmentados a partir de uma imagem de entrada. Em outras palavras, um usuário fornece as informações para cumprir os objetivos de segmentação. Por exemplo, um usuário marca alguns pixels como "objeto" ou o "background", que são referidos como marcadores (ou *seeds*) para fornecer certas restrições à tarefa de segmentação posterior. Então, um procedimento de optimização é realizado, onde cada pixel se torna um vértice num grafo de árvore igual ao tamanho da imagem, mas não interconectado. A partir desse ponto, cortes são feitos no grafo para se obter uma solução global entre todos os segmentos possíveis que satisfaçam às *seeds* implantadas pelo usuário. Ao mesmo tempo, as propriedades de contorno e região são incorporadas na função de custo do problema de optimização, e estas propriedades são vistas como limitações suaves para a segmentação. Aplicações famosas aqui são o ***GrabCut*** e o ***Lazy Snapping***.

Por fim, os **métodos por *Random-Walk*** (RW) descritos por Grady (2006), são semelhantes ao método anterior, onde a imagem é convertida em um grafo que se encaixa a um caso especial da cadeia de Markov (KOLLER, 2009). O algoritmo RW oferece um processo de multi-etiqueta de uso geral (multi-objeto) que permite que o usuário possa inicializar as *seeds* de pano de fundo (*background*) e de primeiro plano (*foreground*) na imagem.

Assim como no método anterior, percebe-se que o problema de segmentação pode ser tratado como um problema de etiquetação, onde os pixels são etiquetados como *backgrounds*, objetos do *foreground* ou até mesmo outros objetos. O RW inverte o processo, atribuindo a etiqueta aos pixels que não contém a *seed*, apenas fazendo com que o algoritmo pergunte: dado o RW de um ponto de início (pixel sem a *seed*), qual

a probabilidade desse pixel atingir o pixel marcado (com a *seed*)? O resultado da segmentação é obtido através da aglomeração de pixels que forma o caminho mais próximo ao *seed*. Infelizmente estes métodos são muito robustos e demorados a se obterem resultados, além de apresentar falhas com imagens com bordas fracas e texturas bem padronizadas. Aplicações famosas aqui são o ***Random Walk*** e o ***Random Walk with Restart* (RWR),** este último resolve os problemas descritos acima com um processo de reinicio de probabilidades usando regras Bayesianas.

Vale lembrar ainda que certas características das imagens como cor, forma, textura e movimento, não são técnicas de segmentação, mas sim propriedades de imagens obtidas através do PDI e que podem ser usadas para ajudar na segmentação (SOLOMON e BRECKON, 2013). Em resumo, a maioria dos procedimentos de segmentação usa e combina informações sobre uma ou mais propriedades encontradas ou "dadas" pela imagem.

De forma bem simples, exemplos de segmentações e objetos que se deseja obter/extrair da cena podem ser vistos na Tabela 2.1.

Tabela 2.1 – Exemplos conceituais simples e possíveis abordagens de segmentação.

Objeto(s)	Imagem	Propósito da segmentação	Abordagem mais apropriada
Avião	Avião no céu	Rastreamento	Movimento, cor
Rostos humanos	Multidão em *shopping center*	Sistema de segurança com identificação facial	Cor, forma
Estruturas feitas pelo homem	Fotografia aérea de satélite	Aquisição de inteligência via aeronave	Textura, cor
Regiões agricultadas	Regiões não cultivadas	Levantamento feito por LandSAT	Textura, forma
Maçãs Granny Smith e peras	Várias frutas em uma esteira rolante	Seleção automática de frutas	Cor, forma

Fonte: Solomon e Breckon, 2013.

Os próximos tópicos apresentam com mais detalhes dois dos principais segmentadores utilizados neste trabalho, o ***GrabCut*** e o ***Watershed***. O trabalho omite

outras técnicas para concentrar-se no processo de segmentação automática implementado a partir de pelo menos um desses dois segmentadores.

2.2 FILTROS DE REALCE E DE UNIFORMIZAÇÃO DE CARACTERÍSTICAS

Antes da segmentação, a imagem geralmente é submetida a processos de filtragem para poder aumentar a credibilidade do resultado e remover ruídos. Por isso, será mostrado uma abordagem simples com alguns filtros que podem ser utilizados dentro do trabalho. Entre eles: o **filtro gaussiano, *Mean-Shift*, Bilateral e Homomórfico**. Note-se que nem todos esses filtros são usados especificamente para remover ruídos, mas são importantes para, por exemplo, uniformizar características importantes, como a cor, luz e texturas.

2.2.1 Filtro Homomórfico

Conforme Padmavathi *et al.* (2010) já havia dito, a formação das imagens pode ser modelada como um produto entre duas matrizes: uma representando a intensidade da iluminação e uma outra representando a refletância dos objetos na cena (refletância representa a capacidade de um objeto devolver a irradiação eletromagnética da luz, se for zero, o objeto é puramente escuro, ser for um, o objeto é um espelho perfeito). O processo matemático que descreve isso está em (2.1).

$$f(x,y) = i(x,y) \cdot r(x,y) \tag{2.1}$$

Onde $f(x,y)$ representa a imagem capturada, $i(x,y)$ a matriz de iluminação e $r(x,y)$ a matriz de refletância dos objetos na cena. Conforme a Figura 2.1, considera-

se que a iluminação tende a variar lentamente (baixa frequência) sobre a imagem, enquanto a refletância, que pode ser caracterizada pelas bruscas mudanças na "formação" dos objetos na cena (alta frequência).

Figura 2.1 - Formação da imagem a partir do modelo de refletância e iluminação. Em (a) a refletância em uma imagem, em (b) o padrão de iluminação e em (c) a imagem visualizada coma multiplicação de (a) e (b).

Fonte: Próprio Autor, 2015.

Dessa forma é possível suprimir as componentes de baixa frequência, enquanto reforça as de media à alta frequência, fazendo com que objetos que estavam "ocultos" pela má distribuição da iluminação, possam "aparecer" na imagem. Contudo, para realizar essa filtragem, é necessário obter o espectro da imagem no domínio da frequência. Neste caso, transformada como a Fourier pode ser usada, mas de acordo com (2.2), ela não é suficiente para se obter as matrizes de forma linearizada.

$$\Im\{f(x,y)\} \neq \Im\{i(x,y)\}\Im\{r(x,y)\} \tag{2.2}$$

Segundo Rodrigues *et al.* (2015), Gonzalez e Woods (2010) e Padmavathi *et al.* (2010), existem cinco passos para obter-se a imagem com a iluminação corrigida.

Passo 1: aplicar o operador de logaritmo para linearizar o processo de obtenção do espectro da imagem no espaço de cores HSV (se a imagem for colorida) ou em tons de cinza, conforme demonstrado em (2.3).

$$z(x,y) = \log[f(x,y)] = \log[i(x,y)] + \log[r(x,y)] \tag{2.3}$$

Passo 2: aplicar a transformada de Fourier (representada pelo símbolo $\Im$), conforme (2.4).

$$\Im\{z(x,y)\} = \Im\{\log[i(x,y)]\} + \Im\{\log[r(x,y)]\} \rightarrow Z(u,v) = I(u,v) + R(u,v) \tag{2.4}$$

Passo 3: filtrar o espectro com um filtro passa-alta. $S(u,v)$ é o resultado do processo entre o espectro Z e o filtro passa-alta H, demonstrado em (2.5).

$$S(u,v) = H(u,v) \cdot Z(u,v) = H(u,v) \cdot I(u,v) + H(u,v) \cdot R(u,v) \tag{2.5}$$

Passo 4: aplicar a transformada inversa de Fourier para voltar a imagem no domínio espacial, visto em (2.6).

$$\Im^{-1}\{S(u,v)\} = s(x,y) = i'(x,y) + r'(x,y) \tag{2.6}$$

Assumindo em (2.7).

$$i'(x,y) = \Im^{-1}\{H(u,v)I(u,v)\} \ e \ r'(x,y) = \Im^{-1}\{H(u,v)R(u,v)\} \tag{2.7}$$

Passo 5: aplicar o operador exponencial em toda a imagem representada por (6), revertendo os efeitos do operador logaritmo do passo 1. Como $z(x,y)$ foi criado a partir do $\log[f(x,y)]$, a inversa de $s(x,y)$ resulta em (2.8).

$$g(x,y) = e^{[s(u,v)]} \rightarrow e^{[i'(x,y)+r'(x,y)]} \therefore e^{i'(x,y)}e^{r'(x,y)} \quad (2.8)$$

Onde $g(x,y)$ é a imagem final filtrada homomórficamente, e derivada da imagem original $f(x,y)$.

Alguns pontos finais na filtragem podem ser examinados mais claramente, como por exemplo: Diniz *et al.* (2014) informa que é comumente usada a implementação da Transformada de Fourier de Cooley-Tukey, que representam uma das melhores implementações da conhecida *Fast Fourier Transform* (FFT ou Transformada Rápida de Fourier) utilizada no passo 2. O pseudocódigo que descreve essa aplicação da FFT pode ser visto no Pseudocódigo 2.1.

Pseudocódigo 2.1: *Fast Fourier Transform* (fft) – Cooley-Tukey

Entrada: Imagem $X_0,...,X_{N-1} \leftarrow$ fft(x, N, s): DFT de (x_0, x_S, x_{2S}, ..., $x_{(N-1)S}$)

Saída: espectro $X_k(m,n)$

```
1:  Se N = 1 faça
2:      X0 ← x0
3:  Senão mapeia
4:      X0, ..., XN/2-1 ← fft(x, N/2, 2s)        %DFT de (x0, x2s, x4s, ...)
5:      XN/2, ..., XN-1 ← fft(x+s, N/2, 2s)      %DFT de (xs, xs+2s, xs+4s, ...)
6:  Para k = 0 até (N/2)-1                       %Combina as DFT's
7:      t ← Xk
8:      Xk ← t + exp(-2πjk/N)*Xk+N/2
9:  Xk+N/2 ← t – exp(-2πjk/N)*Xk+N/2
```

Aqui, a $fft(x,N,1)$ é um algoritmo recursivo do tipo ***divide and conquer***, que calcula $X = DFT(x)$ por uma FFT de grau 2 DET (Decimação em Tempo), onde N é uma potência inteira de 2 e $s = 1$ é o passo da matriz x de entrada. $x + s$ indica a matriz começando com xs.

Já Weeks (2012) informa que um ótimo filtro adaptativo é o filtro Butterworth, ao qual pode ser tanto de passa-alta quanto passa-baixa. A versão que pode ser usada no passo 3 é descrita por (2.9).

$$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}} \tag{2.9}$$

Onde n é o grau da ordem do filtro, D_0 é um valor de limiar (corte) do filtro, e $D(u,v)$ é o coeficiente de análise de distância entre o ponto (u,v) até o centro do espectro de frequência, que é dado por (2.10).

$$D(u,v) = \sqrt{[(u - P/2)^2 + (v - Q/2)^2]} \tag{2.10}$$

Onde P e Q é o tamanho da imagem de entrada no domínio da frequência (nesse caso $Z(u,v)$). O resultado final de uma imagem filtrada homomórficamente pode ser visto na Figura 2.2. Perceba que em todos os casos mostrados, houve uma certa "aparição" de objetos na cena que antes não podiam ser vistas, mas já estavam lá.

Figura 2.2 – Exemplos de filtragem homomórfica. Na esquerda as imagens originais e na direita as imagens filtradas.

Imagem Original *Imagem c/ Filtro Homomórfico*

Fonte: Base de Imagens do Autor, 2015.

O pseudocódigo 2.2 descreve todos os passos da filtragem homomórfica vista nessa seção.

Pseudocódigo 2.2: *Filtro Homomórfico*

Entrada: Imagem original I_o

Saída: Imagem filtrada I_f

1: $I_o \leftarrow$ Leitura da imagem original

2: Tamanho(x,y,z) ← tamanho (NxMx3) da imagem I_o

3: Se tamanho z for diferente de 1 faça

4: $I_o \leftarrow (R_{I_o} + G_{I_o} + B_{I_o})/3$

5: $I_{log} \leftarrow \log(I_o)$

6: $I_{fft} \leftarrow \text{fft}(I_{log})$

7: Para a matriz de frequência gerada faça

8: ButterworthFilter(I_{fft}, tamanhoX, tamanhoY, ordem-n, range D_0)

9: Controle de frequência com $\alpha_L - \alpha_H$ %utilizado na filtragem para adaptação do filtro H(u,v)

10: $I_{ef} \leftarrow$ fft

11: $I_{ifft} \leftarrow \text{ifft}(|I_{ef}|)$

12: $I_f \leftarrow \exp(I_{ifft})$

13: Retorna I_f

Onde I_o é a imagem original de tamanho (x, y, z), z é o espaço de cores, se $z > 1$ a imagem será convertida em tons de cinza, n é a ordem em frequência do filtro, geralmente dado por $1000/300$ Hz e D_0 é o limiar de corte do filtro, que por sua vez é obtido por $\alpha_L - \alpha_H$ que define que um filtro passa-alta é sempre obtido por uma subtração por um passa-baixa (α_L).

2.2.2 Filtro Gaussiano

Segundo Gonzalez e Woods (2010), o filtro Gaussiano é um operador de convolução 1D (uma dimensão) ou 2D (duas dimensões) usado para suavizar (também chamado de ***blurring***) imagens. O filtro Gaussiano é utilizado para remover ruídos e alguns detalhes indesejados na imagem, como artefatos causados por padrões em

texturas ou mesmo inconsistências de pixels (por falhas de sensores ou por *dead pixels*). Trata-se de um filtro passa-baixa.

De acordo com Duda *et al.* (2001), o operador gaussiano pode ser obtido por (2.11).

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{2.11}$$

O mesmo representa o coeficiente univariante gaussiano ou o coeficiente 1D. E μ representa a média do valor da variável randômica x, e pode ser obtida por (2.12).

$$\mu = E[x] = \int_{-\infty}^{+\infty} xp(x)dx \tag{2.12}$$

O $E[\cdot]$ representa esse valor médio ou aproximado de x. Já σ representa a variância de x e pode ser obtida por (2.13).

$$\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{+\infty} (x-\mu)^2 p(x)dx \tag{2.13}$$

Como a maioria das imagens são em 2D, o filtro gaussiano precisa se adaptar a aplicação em duas dimensões ou mais. Para isso, se uma imagem de tamanho $n \times m$ precisa passar pelo filtro, basta multiplicar (2.11) duas vezes e obter (2.14) aonde o filtro atenderá cada uma das dimensões da imagem bidimensional.

$$p(x,y) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right] \tag{2.14}$$

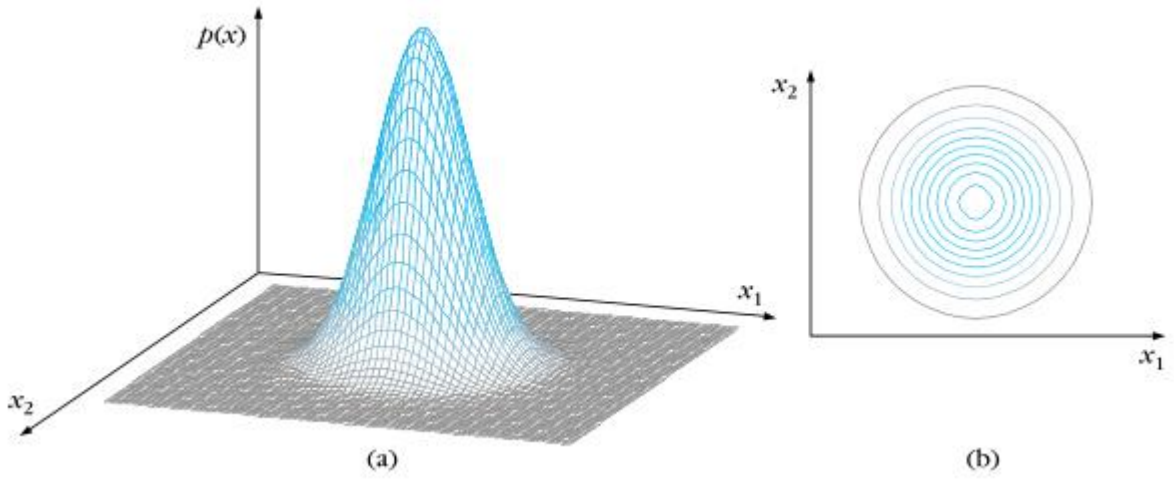
Assim, o operador gaussiano criará a distribuição estatística vista na Figura 2.3. Se perceber, a PDF cria uma matriz de covariância que é chamada de Kernel. Essa

matriz pode ser aplicada na imagem em convolução com cada pixel da mesma. Obtendo assim os resultados de ***smoothing*** desejados. Um Kernel de 5×5 pode ser visto em (2.15).

$$\sum = \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{2.15}$$

O resultado do filtro sobre imagens, pode ser visto na Figura 2.4. Aplicando vários valores de σ ou tamanho do Kernel, percebe-se que a imagem vai ficando cada vez mais borrada.

Figura 2.3 – Gráfico da distribuição de uma Gaussiana 2D. (a) mostra a PDF (*Probably Density Function* ou Função Provável de Densidade); (b) vista superior de (a) que mostra os círculos correspondentes aos isovalores de uma matriz diagonal de covariância $\sum$ de $\sigma_1^1 = \sigma_2^2$. O gráfico é simétrico é não apresenta preferência em direções.



Fonte: Theodoridis e Koutroumbas, 2009. p.22.

Figura 2.4 – Imagem de Van Gogh sendo borrada pelo filtro gaussiano. (a) imagem original; (b) tamanho do Kernel = 9; (c) tamanho do Kernel = 31.

(a) (b) (c)

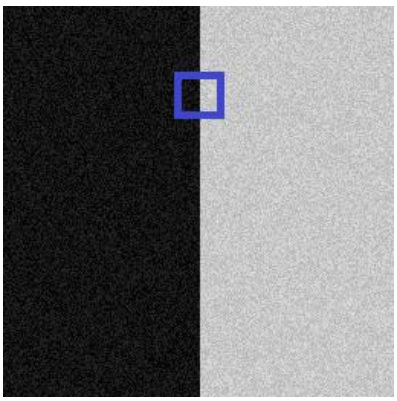
Fonte: Yuan, 2013.

2.2.3 Filtro Bilateral

Como visto anteriormente, o filtro gaussiano remove ruídos, mas não preserva as bordas, podendo constituir-se em um grave problema para algumas aplicações. Para resolver esse problema, Tomasi e Manduchi (1998) propuseram o filtro não-linear Bilateral, que é uma melhoria do filtro gaussiano para evitar o problema de suavização (*smoothing*) de bordas. Nesse caso, o processo de filtragem envolve outro peso que representa como dois pixels podem ter intensidades similares. Considerando assim o peso desses dois pixels, o filtro consegue manter as bordas dos objetos na cena. Vejamos a imagem mostrada na Figura 2.5.

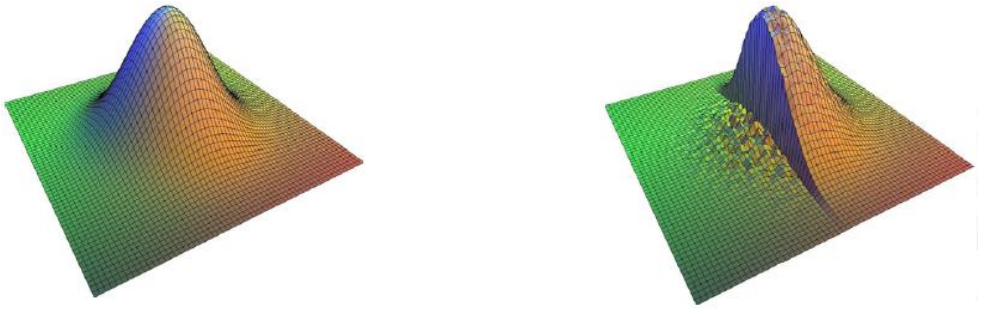
Percebe-se na Figura 2.5 que a imagem está cheia de ruído e por isso precisa ser filtrada. Irá ser aplicado tanto o filtro gaussiano quanto o bilateral, com enfoque nos pixels que estão no centro do quadrado azul. O processo de distribuição dos pesos entre os pixels é visto na Figura 2.6.

Figura 2.5 – Imagem de teste para o filtro bilateral



Fonte: Yuan, 2013.

Figura 2.6 – Kernels. (a) gaussiano; (b) bilateral. Percebe-se que o Bilateral considera ambos os pesos dos pixels na borda.

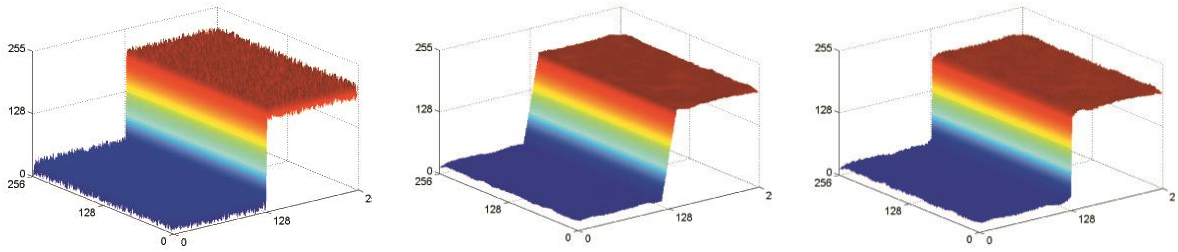


(a) (b)

Fonte: Yuan, 2013.

Digamos que se tem a imagem original com ruídos representada na Figura 2.7a. Se aplicarmos o filtro gaussiano (Fig. 2.7b) a imagem fica com menos ruído, porém a borda não fica mais nítida (uma descida aparece entre os pixels brancos e pretos). Já na Fig. 2.7c, o filtro bilateral consegue deixar a imagem com menos ruído e preserva a borda.

Figura 2.7 – Processos de filtragem na borda. (a) imagem original; (b) filtro gaussiano; (c) filtro bilateral.



(a) (b) (c)

Fonte: Yuan, 2013.

Assim, conforme Tomasi e Manduchi (1998), o filtro bilateral pode ser obtido por (2.16).

$$I_f(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) g_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \tag{2.16}$$

Onde I_f é a imagem filtrada, I é a imagem original, x são as coordenadas do pixel a ser filtrado, Ω é a janela de operação do kernel centrada em x, g_r é o *range* do kernel que serve para verificar o valor mínimo de intensidade de uma borda (função gaussiana) e g_s é o kernel espacial que serve para verificar a extensão espacial do kernel, que é o tamanho da vizinhança que deve ser considerada na coordenada x (função gaussiana). W_p é um termo de normalização, que é dado por (2.17).

$$W_p = \sum_{x_i \in \Omega} g_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \tag{2.17}$$

O resultado da aplicação do filtro bilateral pode ser melhor observado na Figura 2.8.

Figura 2.8 – Imagem de Van Gogh sendo borrada pelo filtro bilateral. (a) imagem original; (b) tamanho do Kernel = 9; (c) tamanho do Kernel = 31.

(a) (b) (c)

Fonte: Yuan, 2013.

2.2.4 Filtro Deslocamento de Média (*Mean-Shift*)

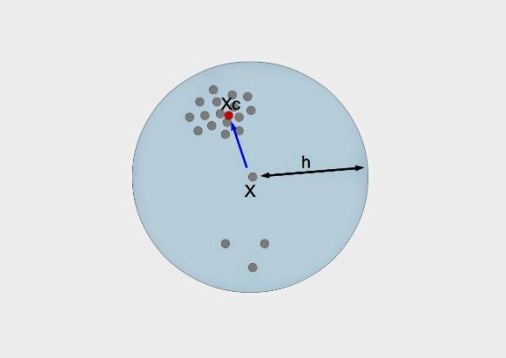
O *Mean Shift* é um método não supervisionado e não paramétrico para estimar o gradiente de uma função de probabilidade. Criado primeiramente no trabalho de Fukunaga e Hostetler (1975), tinha como função principal o reconhecimento de padrões e filtragem de dados e foi muito usado para aplicações de rastreamento de objetos em vídeos. No trabalho de Comaniciu e Meer (1999), o *Mean Shift* teve sua aplicação na visão computacional, mais especificamente usado na segmentação e na suavização de imagens.

Para entender a teoria por trás desse método, Comaniciu e Meer (2002) propõem o seguinte: suponha a imagem RGB que consiste n pixels que tem três valores com base nos canais R, G e B. Assumindo que a imagem está cheia de distribuições aleatórias dentro do espaço de cores (a maior parte das imagens é assim), pode-se definir a função da densidade $f(\vec{x})$ pela (2.18).

$$f(\vec{x}) = \frac{1}{nV_d} \sum_{i=1}^{n} K(\vec{x}_i; \vec{x}, h) \tag{2.18}$$

Onde V_d denota o volume da hiperesfera d-dimensional com raio h e centro $\vec{x}$ (Figura 2.9). Assim (2.19) pode ser satisfeita e a função de densidade converge.

Figura 2.9 – Hiperesfera usada para a detecção de pontos de máxima densidade local.



Fonte: Comaniciu e Meer, 2002.

$$\int d\vec{x}\, f(\vec{x}) = 1 \tag{2.19}$$

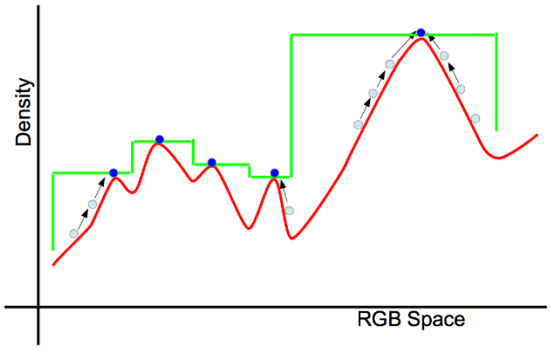
A função $K(\vec{x}_i; \vec{x}, h)$ definida por (2.20) é chamada de kernel. Geralmente o kernel adotado na prática é o **kernel de Fukunaga** (COMANCIU e MEER, 2002), representando o círculo unitário centrado na origem, dado por (2.21).

$$K(\vec{x}_i; \vec{x}, h) = \begin{cases} 1 & \text{se} \quad \|\vec{x} - \vec{x}_i\| \leq h \\ 0 & \text{se} \quad \|\vec{x} - \vec{x}_i\| > h \end{cases} \tag{2.20}$$

$$G(\vec{x}_i; \vec{x}, h) = \begin{cases} 1 & \text{se} \quad x \leq 1 \\ 0 & \text{se} \quad x > 1 \end{cases} \tag{2.21}$$

Esse procedimento é chamado de **análise de *Mean-Shift***, permitindo a detecção de pontos de máxima densidade local como descrito na Fig. 2.10.

Figura 2.10 – Análise de *Mean-Shift* em uma imagem. Os pontos azuis representam os valores $\vec{x}_i$ que representam os valores de máxima densidade e linha verde denota os valores de *smoothing* da imagem final.



Fonte: Comaniciu e Meer, 2002

Qualquer ponto $\vec{x}$ dentro do espaço de cores tridimensional pode convergir em torno de um ponto de máxima densidade próxima a $\vec{x}$. Se todos os pontos $\vec{x}_i$ forem substituídos pelo valor $\vec{x}$ de máxima densidade adjacente, obtém-se a imagem borrada (Fig. 2.10).

Deve-se, contudo, notar que a imagem não tem apenas informações de cores, mas também de localidade desses pontos. Sendo assim, o procedimento explicado a pouco para borrar a imagem não considera a informação de localidade. Assim, dois objetos que têm cores similares podem, na prática, serem rotulados com a mesma cor, produzindo inconsistências. Para contornar este problema, se faz necessário construir um novo vetor de informações, levando em conta a localização (COMANICIU e MEER, 2002), dado por (x, y, r, g, b). Nesse caso, o kernel deve ser reajustado para (2.22).

$$K(\vec{x}_i; \vec{x}, h) = K_{spartial}(\vec{x}_i^s; \vec{x}^s, h_s) K_{range}(\vec{x}_i^r; \vec{x}^r, h_r) \tag{2.22}$$

Onde $\vec{x} = (\vec{x}^s, \vec{x}^r)$, $\vec{x}^s = (x, y)$ e $\vec{x}^r = (r, g, b)$, h_s é o raio da hiperesfera de localização espacial do ponto $\vec{x}$ e h_r é o raio da hiperesfera de espaço de cores. Para aplicar esse kernel, também é necessário normalizar o pixel de máxima densidade, através da expressão (2.23).

$$\vec{m}(\vec{x}) = \frac{\sum_{i=1}^{n} K(\vec{x}_i; \vec{x}, h)\vec{x}_i}{\sum_{i=1}^{n} K(\vec{x}_i; \vec{x}, h)} \tag{2.23}$$

Apenas quando o kernel é calculado que se tem os valores de $\vec{x}^s$ e $\vec{x}^r$. Assim, se aplicarmos a análise de *Mean-Shift* em todos os pontos da imagem $\{\vec{x}_i | i = 1, \dots, n\}$ obtemos uma série de pontos convergentes $\{\vec{z}_i | i = 1, \dots, n\}$. Se substituirmos $\vec{x}_i^{\,r}$ (componente de cor do ponto original) por $\vec{z}_i^{\,r}$ (componente de cor do ponto convergente) temos um novo ponto $\vec{y}_i = (\vec{x}_i^{\,s}, \vec{z}_i^{\,r})$. A imagem final e filtrada pelo *Mean-Shift* pode ser obtida pelos pontos $\{\vec{y}_i | i = 1, \dots, n\}$. Uma amostra prática do que o filtro pode fazer é representada pela Fig. 2.11.

Figura 2.11 – Imagem de um Babuíno sendo borrada pelo filtro *Mean-Shift*. (a) imagem original; (b) $(h_s, h_r) = (16, 32)$; (c) $(h_s, h_r) = (16, 64)$.

(a) (b) (c)

Fonte: Comaniciu e Meer, 1999.

2.3 SEGMENTADORES

Por limitação de escopo, este trabalho aborda somente dois tipos de segmentadores: os do tipo ***Graph-Cut*** e os do tipo baseados em região, sendo que para o cada tipo, serão explanados o ***GrabCut*** e o ***Watershed***. Estes foram escolhidos pelo seu potencial de serem utilizados para compor a solução do método proposto.

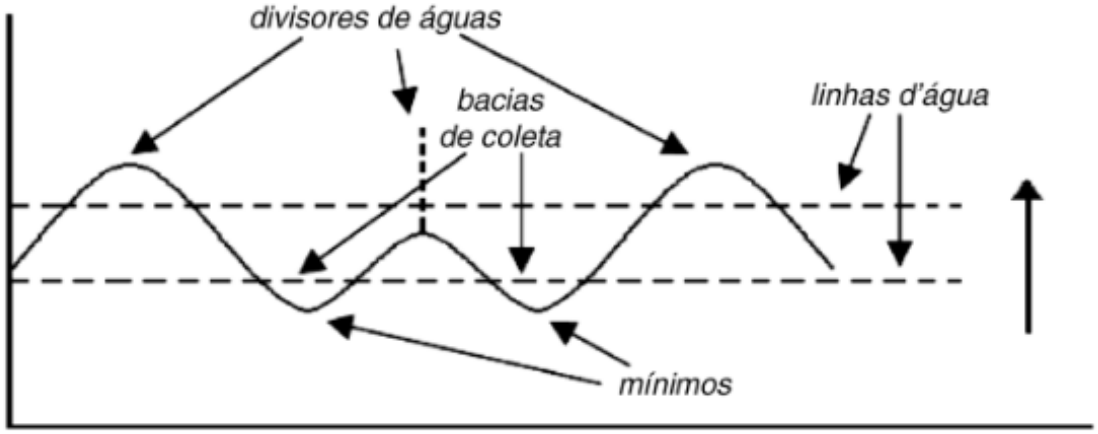
2.3.1 Transformada *Watershed*

Criado por Serge Beucher em 1979, e posteriormente adaptado pelo próprio autor para a visão computacional (BEUCHER, 1992), o algoritmo de segmentação *Watershed* é hoje um dos mais importantes e usados segmentadores no mundo. Conforme Solomon e Breckon (2013) explicam, essa é uma abordagem que se baseia no princípio de que qualquer imagem 2D de tamanho (x, y) em escala de tons de cinza pode ser assemelhada a uma superfície topológica, onde as intensidades de tons de cinza são associadas a uma dimensão z.

Duas são as variações da transformada *Watershed*: a baseada no algoritmo *Waterfall* e a que utiliza marcadores (*seeds*). No caso do ***algoritmo Waterfall***, uma "chuva" cai nesta paisagem, que será naturalmente drenada para as regiões de ponto mínimo mais próximos, tudo através do efeito "gravitacional" (peso). Uma bacia de coleta define a região conectada onde qualquer chuva é drenada para o mesmo ponto de mínimo local. Essa bacia de coleta é nada mais que um grupo de pixels conectados através de semelhanças na sua topologia. Nota-se assim que haverá pontos na paisagem (máximos locais) dos quais a chuva pode ser drenada com igual probabilidade em duas bacias de coleta adjacentes, isso é como se fosse o cume de uma montanha. Linhas vão surgindo gradativamente a medida que a agua das chuvas inunda os vales, até o ponto que essas linhas se tornam divisores entre uma bacia e outra. Essas linhas de divisão

são as áreas ou objetos diferentes que foram separados pela segmentação. Este processo é hierárquico e necessita de múltiplas passagens antes de remover os problemas causados pela sobre segmentação. O processo análogo pode ser visto na Figura 2.12.

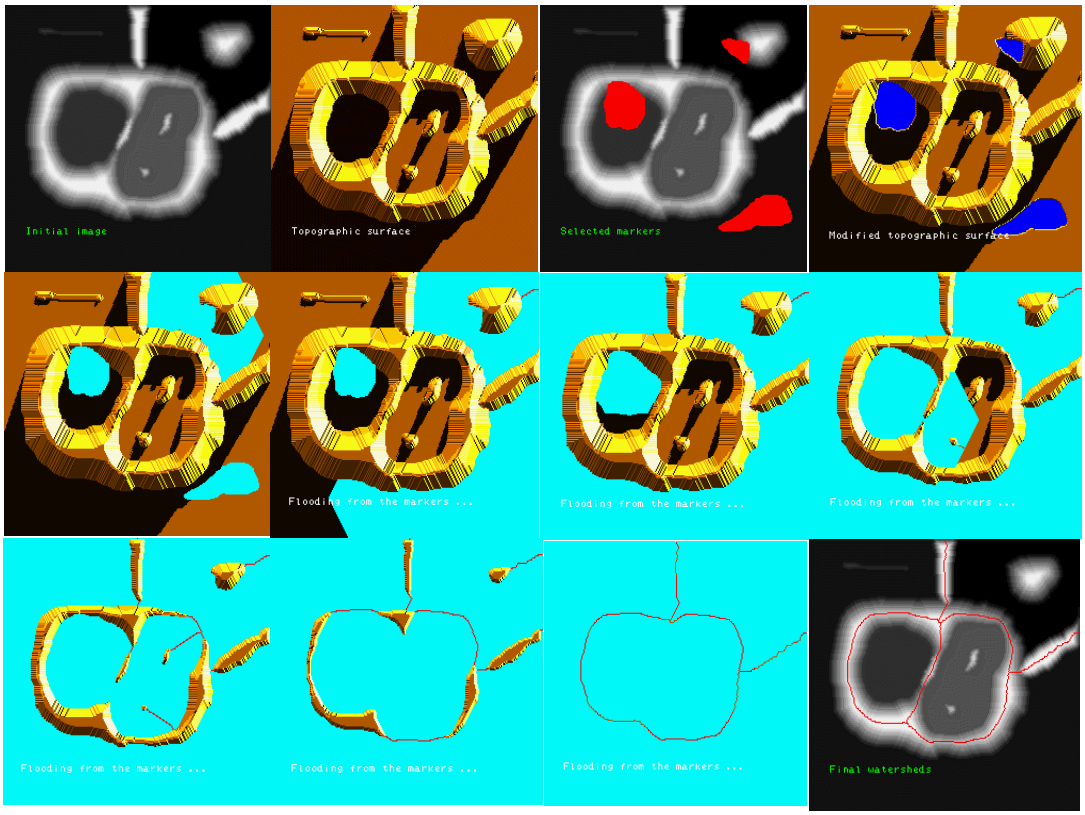
Figura 2.12 – Divisores de água e bacias de coleta em 1D. As bacias são definidas pelos pixels interconectados, é cada uma dessas bacias tem um ponto de mínimo para onde a água da chuva fluirá. Os divisores são como os cumes aonde a água tem a probabilidade de fluir para qualquer uma das bacias adjacentes.



Fonte: Solomon e Breckon, 2013. pp.250.

Considerando agora o caso dos marcadores (*seeds*), proposto para evitar sobre segmentação, considera-se alagar gradativamente a paisagem através de fontes de água que tem origem nos *seeds* ou pelos mínimos locais. À medida que o nível da agua se eleva, o algoritmo vai construindo represas para evitar que a água de uma bacia de coleta transborde para as bacias vizinhas. Quando o nível de água chega ao pico mais alto, o processo de construção das represas é interrompido. As represas construídas (e que não ficaram submersas) são os divisores de águas. O processo passo-a-passo pode ser visto na Figura 2.13.

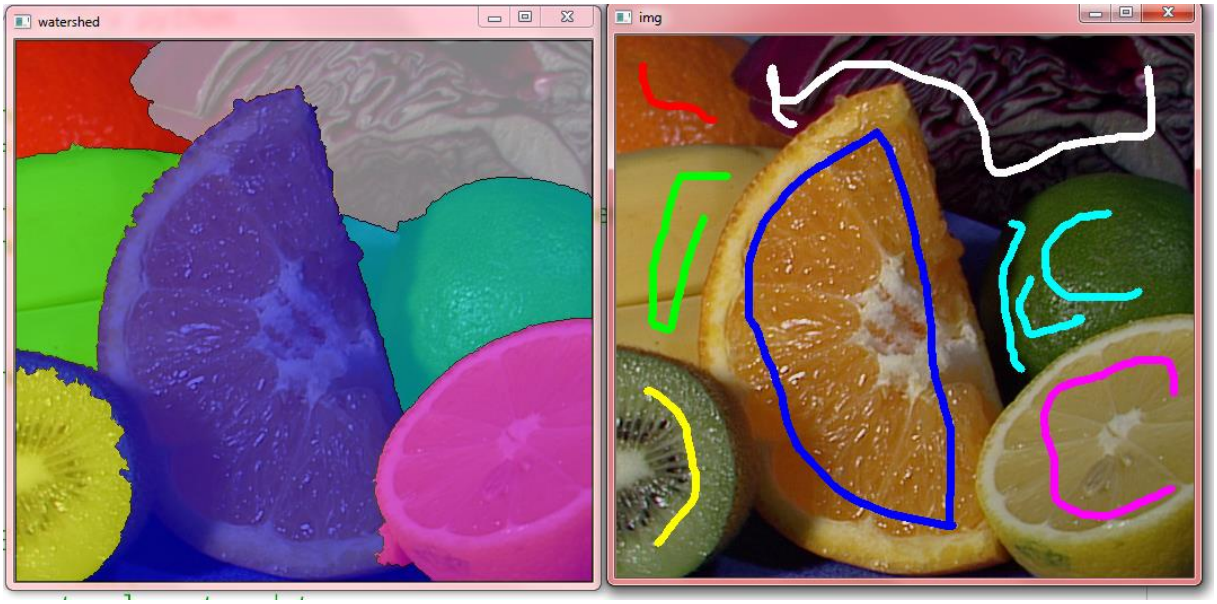
Figura 2.13 – Etapas da segmentação pela inundação começando a partir das marcas. O processo começa na imagem da esquerda no topo (tons de cinza) e prossegue para a direita, de cima para baixo, passando pela topologia, colocação das marcas, mudança de topologia, inundação até atingir as represas no cume, gerando a imagem segmentada no final. Note que isso ajuda a evitar a sobre segmentação.



Fonte: Beucher, 2010.

Uma aplicação mais eficaz do ***Watershed*** pode ser vista na Figura 2.14. A mesma contém as marcas com os rótulos de várias cores (e objetos a serem segmentados). No final o processo tenta diminuir a sobre segmentação, que nada mais é do que o caso quando a aplicação do *watershed* gera uma serie de segmentos dentro do objeto a ser segmentado, ou seja, o objeto a ser segmentado está sobre segmentado.

Figura 2.14 – Segmentação *Watershed*. A esquerda a segmentação final, na direita as marcas aplicadas de forma manual. Perceba que em algumas frutas, ocorre a sobre segmentação.



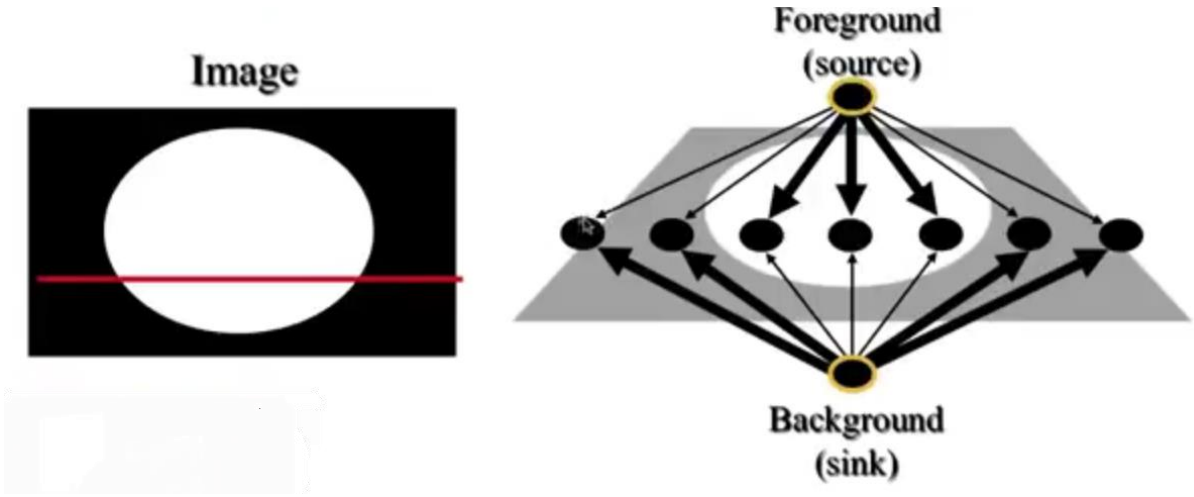
Fonte: Próprio Autor, 2016.

2.3.2 GrabCut

De todas as aplicações, os segmentadores do tipo *Graph-Cut* são os mais recentes. Criado pelo trabalho de Boykov e Jolly (2001) que propuseram a ideia de converter os pixels que fariam parte do "corte" como se fossem partes de um grafo ponderado, aonde a árvore total seria a dimensão da imagem de entrada e a mesma seria cada vez mais descontinua com base em pesos entre as arestas que representariam os divisores da imagem.

A ideia é simples e pode ser vista na Figura 2.15. Observe que uma imagem em tons de cinza é usada como entrada na aplicação. Observando mais de perto (nos pixels da linha vermelha), nota-se que o algoritmo irá precisar que o usuário crie *seeds* que identifiquem quem é o *foreground* e quem é o *background*. A eles, são associados um vértice de *source*/fonte (para o *foreground*) e um vértice de *sink*/sumidouro (para o *background*).

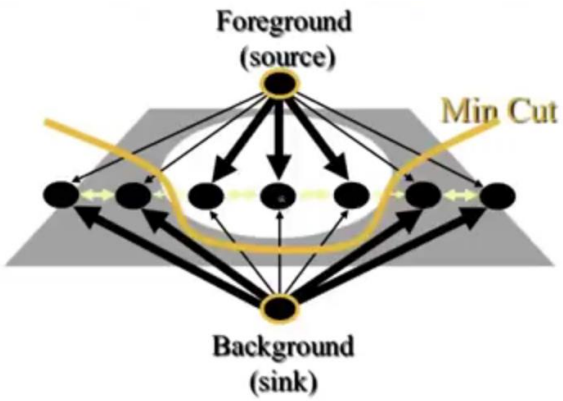
Figura 2.15 – Corte feito pelo *Graph-Cut* em uma imagem. Imagem que será passada pelo *Graph-Cut* (esquerda). Visualização dos pixels na faixa vermelha (direita).



Fonte: Boykov e Jolly, 2001.

Após essa etapa, o método interconecta todos os pixels com arestas ponderadas. O peso de cada aresta é medido por função gaussiana de energia para verificar a mudança na intensidade de um pixel vizinho. Se a intensidade da mudança for alta, o peso tende ao infinito. Logo, a imagem é dividida em apenas duas árvores ou regiões, a que representa a árvore dos pixels ligados a fonte (*foreground*) e a que representa os pixels ligados ao sumidouro (*background*). Assim, um algoritmo de corte é lançado, conforme Figura 2.16.

Figura 2.16 – Aplicação do corte feito por um algoritmo de *Max-Flow*, como o *Ford-Fulkerson*.



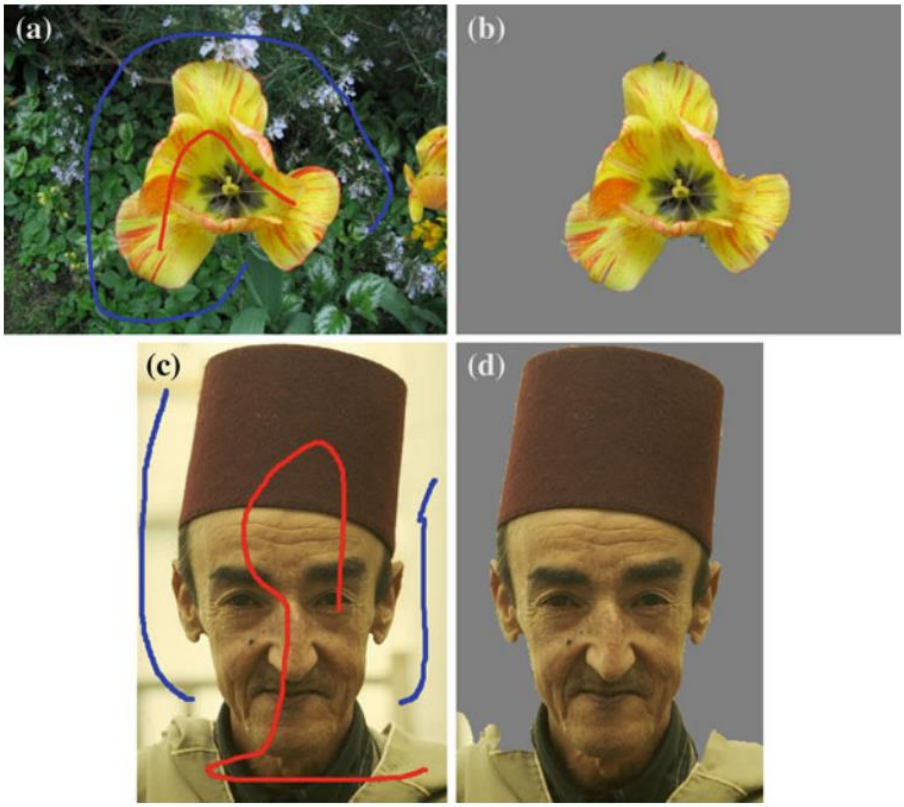
Fonte: Boykov e Joly, 2001.

Quando um algoritmo de corte como um algoritmo de *Max-Flow* é aplicado (por exemplo o algoritmo de Ford-Fulkerson), o fluxo máximo tende a retornar o corte mínimo necessário para segmentar a imagem, que separa apenas os pixels associados a aresta fonte e elimina os que estão associados a aresta sumidouro. O problema aqui é que sempre o processo retorna uma coleção de arestas que causam uma pixelação na imagem ou uma sub segmentação dos objetos na cena. Lembrando ainda que o peso das arestas é medido por uma função de energia, dada por (2.24).

$$E(L) = \sum_{i \in V} D_i(L_i) + \sum_{(i,j) \in E} V_{i,j}(L_i, L_j) \tag{2.24}$$

Onde L_i é um método binário de classificação dos pixels da imagem (0 para *background* e 1 para *foreground*); D_i é uma função pré-especificada que indica a preferência de *labels* dos pixels e seus vizinhos, baseados na cor ou intensidade; $V_{i,j}$ denota o custo do caminho, e $(i,j) \in E$ significa que x_i e x_j são nós adjacentes e conectados pela aresta (x_i, x_j) no grafo G. O princípio aplicado pelo método é que o peso da aresta encoraja o sistema a ter uma coerência espacial em troca de uma penalidade: penas maiores ocorrem quando os pixels são similares e menores quando os pixels são diferentes. Um exemplo do método é mostrado na Figura 2.17.

Figura 2.17 – Aplicação real do corte feito pelo *Graph-Cut*. (a) e (c) são as imagens originais com as *seeds* manuais; e (b) e (d) é o resultado final.



Fonte: He, Kim e Kuo, 2014.

Agora que a ideia principal do *Graph-Cut* foi introduzida, é hora de falar da aplicação melhorada: ***GrabCut***. Criado por Rother *et al.* (2004), usa uma otimização laplaciana (que também é usada no *Graph-Cut*, apenas após o corte) em cada iteração da formação do grafo. É o algoritmo do tipo *Graph-Cut* mais usado e tem três funções especiais que vale destacar:

1. *GrabCut* usa um Modelo de Mistura Gaussiana (GMM) para representar as cores dos pixels (em vez do modelo histograma monocromático usado no *Graph-Cut*).
2. *GrabCut* fica alternando entre as estimativas de objetos e estimativa por parametros do GMM de forma iterativa, enquanto a otimização é feita apenas uma vez no *Graph-Cut*.

3. *GrabCut* exige menos interação do usuário. Basicamente, um utilizador apenas tem de colocar um retângulo ou laço em torno de um objeto (em vez de traços detalhados). Um usuário ainda pode desenhar marcas para posterior refinamento, caso seja necessário.

Um outro detalhe é que a equação de energia muda para se adequar a repetição da optimização, a Eq. (2.25) denota isso (critério de Gibbs).

$$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z) \tag{2.25}$$

Onde α pode ser 0 ou 1, definido o *background* do *foreground*, respectivamente. k é um vetor que associa cada pixel com uma componente única do GMM. z é a imagem de entrada. θ representa o modelo de parâmetros da GMM. $V(\cdot)$ é o custo da aresta e $U(\cdot)$ são os termos de informação obtidas do retângulo inserido pelo usuário. Uma aplicação pode ser vista na Figura 2.18.

Figura 2.18 – Aplicação real do corte feito pelo *GrabCut*. A esquerda é a imagem original com um retângulo inserido pelo usuário. A direita, a segmentação final.

Fonte: Próprio Autor, 2015.

2.4 DISCUSSÃO SOBRE A LITERATURA

Conforme descrito, a literatura apresenta muitas ferramentas capazes de prover os fundamentos necessários para a resolução da problemática deste trabalho. Em suma, a obtenção de informações para a filtragem de ruídos e a segmentação vistas nas referências descritas proporcionou novas ideias e opções de pesquisa aos trabalhos que podem ser correlatados com o problema em si.

Duda *et al.* (2001), Theodoridis e Koutroumbas (2009) e He, Kim e Kuo (2014), além de outros autores pesquisados, demonstraram uma serie de arguições referentes aos termos de segmentação e possíveis processos anteriores a essa etapa, principalmente para as aplicações de caráter inovador, pois a ideia de segmentação envolve uma compreensão intima dos princípios mais fundamentais do comportamento matemático. Com isso, no próximo capitulo apresenta-se uma abordagem baseada em uma diversidade de técnicas de segmentação e extração de características, incluído a parte de melhoria de imagem.

3 TRABALHOS CORRELATOS

3.1 APRESENTAÇÃO

Dada a complexidade do processo e para preparar a imagem do objeto para a sua segmentação (que no caso é o gato doméstico), faz-se necessário compreender as etapas de segmentação, bem como aplicações que envolvam um processo não-supervisionado. Para começar, conforme explanado por Gonzalez e Woods (2010), a caracterização de imagens no meio digital é dada por um processo em etapas distintas, representado pela Figura 3.1.

Figura 3.1 – Etapas da classificação em imagens.

Fonte: Próprio Autor, 2015 (Adaptado de Gonzalez e Woods, 2010).

A etapa de aquisição de imagens está fora do escopo deste trabalho, sendo que somente será mencionado no próximo capitulo o método de captura utilizado. Na etapa de pré-processamento, a melhoria da imagem será usada para aumentar a confiabilidade do processo não-supervisionado do segmentador. Pois de acordo com He, Kim e Kuo (2014), o processo de segmentação deve prover um método que utilize uma imagem de entrada bem definida pelo usuário. Nesse caso, o processo de segmentação deve conter uma imagem (ou uma série delas) que esteja com a melhor qualidade possível para a aplicação.

As etapas de representação e reconhecimento também não serão abordadas neste trabalho, sendo mencionadas somente na conclusão.

Além disso, foi visto no capítulo anterior que problemas de sobre e sub segmentação são comuns em processos de segmentação. Por isso, trabalhos correlatos sobre segmentação automática de animais (principalmente sobre gatos domésticos) são raros na comunidade cientifica, este capitulo irá abordar trabalhos correlatos em partes do processo que se deseja levantar para concluir o objetivo principal deste trabalho, que são:

- **Trabalhos sobre melhoria de imagens (*image enhancement*):** aqui serão abordados trabalhos cujas técnicas apresentadas podem ajudar a aplicação de remoção de ruídos e atenuação de características capazes de prover uma melhoria na aplicação de segmentação posterior.
- **Trabalhos sobre processos de segmentação não-supervisionada (*unsupervised segmentation*):** aqui serão abordados trabalhos que demonstram técnicas de segmentação automática através da inserção automática de *seeds* ou da análise de características da imagem, de forma que possam ajudar na pesquisa. Algumas outras características da segmentação sejam explicadas nesta parte.
- **Trabalhos sobre processos de redução de sobre segmentação (*over-segmentation reduction*):** aqui serão abordados trabalhos que buscam resolver os problemas de sobre segmentação (e até de sub segmentação) que praticamente ocorrem sempre em processos de segmentação (HE, KIM e KUO, 2014).

Para isso, foi feita uma revisão sistemática capaz de prover conteúdo necessário para a pesquisa, como: *journals*, livros e artigos. Todos pesquisados em repositórios como: **Google Scholar, Scopus e IEEE Xplore**. Foram usadas várias *strings* de busca, sendo que as mais utilizadas foram: *animal segmentation, unsupervised segmentation, over/under-segmentation reduction, cats, object detection, animal*

recognition, animal tracking. Como resultado obteve-se 283 artigos, dos quais, após vários refinamentos, foram catalogados 15 a 20 artigos viáveis.

3.2 MELHORIA DE IMAGENS

A melhoria da entrada deve ser feita devido ao fato que os processos de segmentação podem causar ruídos indesejados (NIXON e AGUADO, 2012) ou mesmo a imagem original pode conter informações que gerem artefatos inadequados (que podem contribuir com a sobre segmentação). Pode ser necessário a aplicação dessa mesma técnica durante o processo de segmentação também, afim de aumentar as chances do segmentador de reduzir a sobre segmentação.

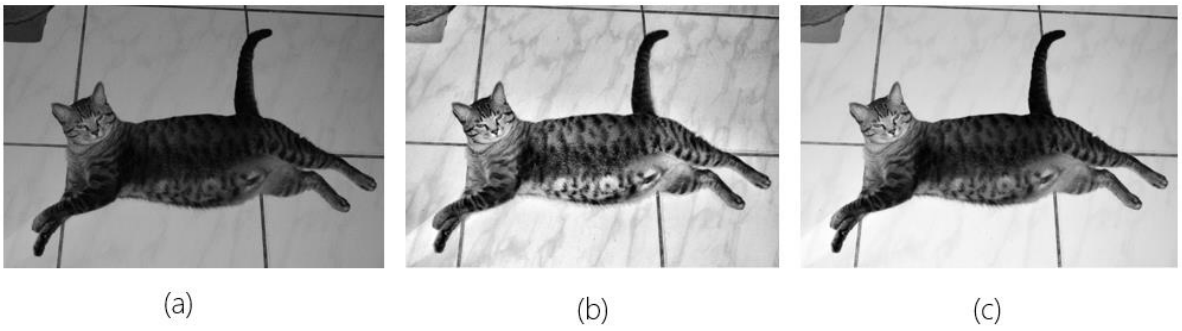
Com isso, pode-se utilizar um processo de melhoria simples e sem muito custo computacional, abordado por Rodrigues *et al.* (2015), onde é aplicado um processo de três passos para a correção de imagens com a combinação de técnicas de Filtragem homomórfica, melhoria por contraste e correção de cores no sistema HSV (*Hue* - Matiz, *Saturation* - Saturação e *Value* - Intensidade).

Um dos processos dessa melhoria, pode ser visto de forma empírica pelo leitor na Figura 3.2.

Aparentemente, o processo da imagem com *Contrast-Limited Adaptive Histogram Enhancement* (ou CLAHE) se torna mais nítida, contudo o processo deixa muito ruído na cena (e "aparições" de objetos inoportunos). Criando assim, muitos pontos que podem confundir o segmentador mais tarde. A técnica de Rodrigues *et al.* (2015), ainda inclui comparações com os trabalhos de Iqbal *et al.* (2007) e Schettini *et al.* (2010). Ambos os trabalhos comparam técnicas similares com os processos de *Cropping* e *Stretching* de imagens em ambos os espaços de cores RGB (*Red*, *Green* e

Blue) e HSV, para depois combiná-los em uma única imagem. O processo de Rodrigues *et al.* (2015) é claramente melhor e mais adequado a essa aplicação, não devido comparação subjetiva, mas sim através de comparação analítica feita por medidores de contraste como o *Enhancement Measure* (EME) e a *entropy* (H).

Figura 3.2 – Demonstração das técnicas correlatas para a melhoria de imagens. (a) foto original com má distribuição de iluminação na cena; (b) método CLAHE; e (c) método Rodrigues *et al.*



Fonte: Próprio Autor, 2015.

Lembrando que o processo ainda precisa ser melhorado para a aplicação final, pois, conforme Meritt (2009) afirma, gatos se destacam em imagens devido ao seu alto contraste gerado pela textura de seus pelos, causando um destaque maior na cena em relação a outros objetos. Por isso, o autor desenvolveu um filtro para melhorar o destaque de cores "quentes" que envolvam os gatos, o filtro desenvolvido é chamado de *Color Boost* e será comentado no Capitulo quatro. Por hora, será continuado para a próxima fase: segmentação não-supervisionada.

3.3 SEGMENTAÇÃO NÃO-SUPERVISIONADA

Nessa etapa, o processo de segmentação irá retirar o gato da imagem. Percebe-se que a ideia de segmentação usada por este trabalho é o de separar o animal

(*foreground*) do fundo (*background*) da imagem. O algoritmo segmentador buscará tentar segmentar o gato, com melhor precisão e de forma automática.

Lembrando que neste trabalho, também se considera que todas as imagens já contenham gatos. O processo de detecção de animais na cena combinada a segmentação automática, foto a foto, será abordada em um projeto futuro.

De acordo com Forsyth e Ponce (2012), a segmentação é considerada uma etapa *mid-level* do processo de visão humana. Caracteriza-se pelo processo de *clusterização* de pixels e de elementos semelhantes (vizinhos) em uma imagem, formando *shapes* e objetos que atraem atenção. De acordo com Grady (2005 e 2006) e He, Kim e Kuo (2014), os pixels vizinhos e semelhantes entre si podem ser considerados grafos interconectados pelas suas similaridades de intensidade de tons de cinza. Para abordagem de Beucher (1992), as imagens podem ser separadas em valores de máximo e mínimos para aplicação de um processo de limiarização capaz de remover objetos com bordas fracas e apenas deixando os objetos de maior contraste nas cenas.

Conforme o trabalho exposto por Kim *et al.* (2005), técnicas como o *Graph-Cut* e o *Random Walker* (RW) são aplicações de segmentação bem inovadoras e recentes para a descrição da problemática, e envolvem aplicações de *tracking* (para retirar e comparar o movimento do gato em várias fotos). As aplicações por Grady (2005) demonstram métodos originais para aplicações de RW em múltiplos objetos, o que pode ser validado em relação as fotos de gatos que estejam em ambientes com ótima iluminação, mas que apresentem muitas informações no *background* (processo não-homogêneo).

Outra abordagem seria a utilização de outros tipos de aplicação vistas nos trabalhos de Couprie *et al.* (2011), que utilizam técnicas parecidas com *Watershed de* preenchimento de regiões, mas levando em conta aplicações combinadas de *Graph-Cuts* e outros segmentadores. E dentre os métodos vistos por He, Kim e Kuo (2014), também

é mencionada uma técnica baseada em região chamada *Maximal Similarity-based Region Merging* (MSRM) ou Mescla de Regiões baseadas em Similaridade Máxima, também proposta por Ning *et al.* (2010), que nesse caso, usa o *Mean-Shift* em um processo de filtragem antes de usar a segmentação para tentar uniformizar as texturas e aumentar a confiabilidade da segmentação.

De todos os trabalhos, no processo do *GrabCut*, o trabalho de Jahangiri e Heesch (2009) apresenta uma forma não-supervisionada para aplicação do *GrabCut*. Essa por sua vez, utiliza o mesmo princípio usado por Rother *et al.* (2004) para criar um método mais refinado de GMM para estimar quais pixels pertencem ao *foreground* e ao *background* através da minimização do custo associado à energia por um processo de aplicação de contornos ativos que são criados a partir da segmentação inicial. Além disso, enquanto o método original de Rother *et al.* termina as iterações através do critério de convergência da energia de Gibbs (Eq. (2.25)) (definido um *Trimap* de imutabilidade de cores de pixels), o processo de Jahangiri e Heesch busca a convergência através de um método de medida de contraste simétrico máximo no pixel analisado, definido por (3.1), que em suma simboliza a parada por maximização de contraste. O método pode ser visto nas Figuras 3.3 e 3.4.

$$\text{Contrast}(P,Q) = D_{KL}(P,Q) + D_{KL}(Q,P) \tag{3.1}$$

Onde $D_{KL}(P,Q) = \sum_{i=1}^{N} p_i \log p_i/q_i$ é a divergência de Kullback-Leibler, e P e Q são os histogramas normalizados de cores no RGB dos pixels do *foreground* e do *background*, respectivamente.

Figura 3.3 – Imagem a ser segmentada pelo *GrabCut* Automático. Esquerda: original; Direita: imagem com o *GrabCut* e os contornos ativos já aplicados.

Fonte: Jahangiri e Heesch, 2009.

Figura 3.4 – Comparação entre o *GrabCut* original e o *GrabCut* Automático de Jahangiri e Heesch. Esquerda: método de Jahangiri e Heesch; Direita: *GrabCut* original por Rother *et al.*

Fonte: Jahangiri e Heesch, 2009.

O trabalho de Park *et al.* (2009), busca algo semelhante, mas em vez de se preocupar com o critério de parada, esta busca primeiro aplicar uma série de filtros gaussianos e um filtro de difusão anistroscópica para, assim, usar um método robusto

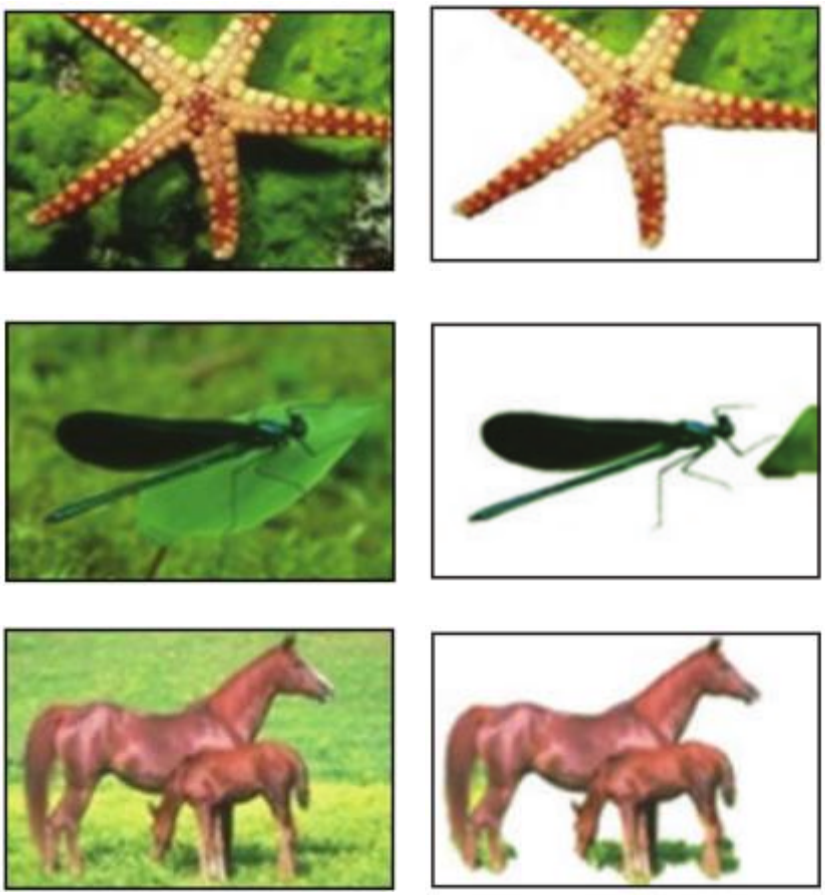
de segmentação inicial que sobre segmenta a imagem, e depois utiliza o *GrabCut* comum (apenas com uma leve mudança na equação de energia, que agora também se preocupa com a energia causada pela uniformização das texturas na imagem) para segmentar a imagem original. O processo pode ser visto na Figura 3.5.

Figura 3.5 – Demonstração do *GrabCut* de Park *et al.* Esquerda: imagens originais; Direita: método Park *et al.*

Fonte: Park *et al.*, 2009.

Por fim, na parte dos trabalhos com o *GrabCut*, temos o trabalho de Khattab *et al.* (2014), que utiliza o método de *clusterização* de Orchard e Bouman (e não o GMM) para atingir melhores resultados de segmentação. Contudo, o processo de segmentação é feito em basicamente todos os espaços de cores (RGB, HSV, etc.). Assim, no final da aplicação, o algoritmo recombina todas as segmentações para prover a melhor segmentação do objeto de maior contraste da cena. O processo pode ser visto na Figura 3.6.

Figura 3.6 – Demonstração do *GrabCut* de Khattab *et al.* Esquerda: imagens originais; Direita: *GrabCut* por Khattab *et al.*



Fonte: Khattab *et al.*, 2014.

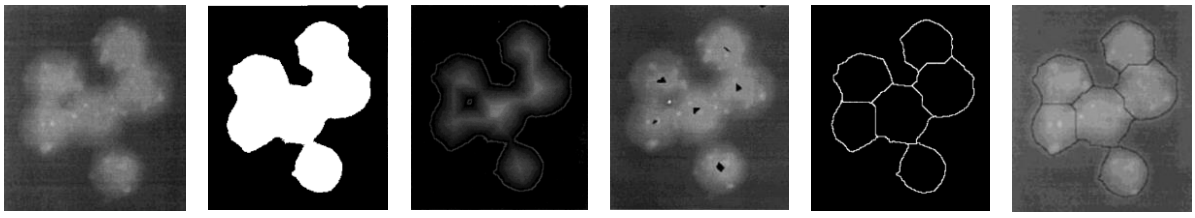
Já na parte do *watershed*, o trabalho de Puzicha *et al.* (1997), que utiliza uma série de filtros Gabor para determinar pelo PDF e a densidade empírica (histogramas) as semelhanças das texturas em ordem de segmenta-las (não o objeto em si) e utiliza o algoritmo de Monte Carlo como critério de parada. Essa aplicação usa uma derivada da *watershed*.

E por fim, Pratikakis *et al.* (2006) e Malpica *et al.* (1997). No trabalho de Pratikakis, temos uma proposta não-supervisionada de segmentação usando um *watershed* hierárquico que cria suas bacias de coleta com uma ponderação de regiões singulares através de uma descriminação por uma melhoria de contraste que é retirada de uma série de filtros *log*-Gabor, os quais conseguem transformar o objeto a ser

segmentado de forma mais contrastante possível para ser segmentado através de processo parecido com as técnicas de *Graph-Cut*, transformando as regiões em vértices de grafos ponderados e aplicando um corte para separar e criar as represas (segmentações). A aplicação de Malpica *et al.* é mais simples e intuitiva, e apesar de ter sido usada primeiro em processos de segmentação em imagens médicas, hoje o processo é bastante difundido devido a sua baixa complexidade computacional e a aceleração do algoritmo de segmentação (SOLOMON e BRECKON, 2013). Para o trabalho de Malpica *et al.* é usada a transformada da distância como processo de encontrar as marcas, que geralmente serão alocadas nos mínimos locais, gerando as bacias de coleta. O processo consiste em se limiarizar a imagem e inverter seus valores, aplicar a posterior transformada da distância e por fim, uma última limiarização. Os processos de limiarização são realizados por vários métodos de *Threshold* conhecido como *Multi-level Threshold*. O resultado dessas operações gera as marcas necessárias para serem usadas no *watershed*. O processo dessa aplicação pode ser visto na Figura 3.7.

Com o processo de segmentação terminado, parte-se para a próxima etapa de redução de sobre segmentação em imagens.

Figura 3.7 – Segmentação não supervisionada de Malpica *et al.* Da esquerda para direita: imagem original; limiarização e inversão de valores; transformada de distância; marcas criadas (pontos pretos); transformada *watershed* aplicada; imagem segmentada.



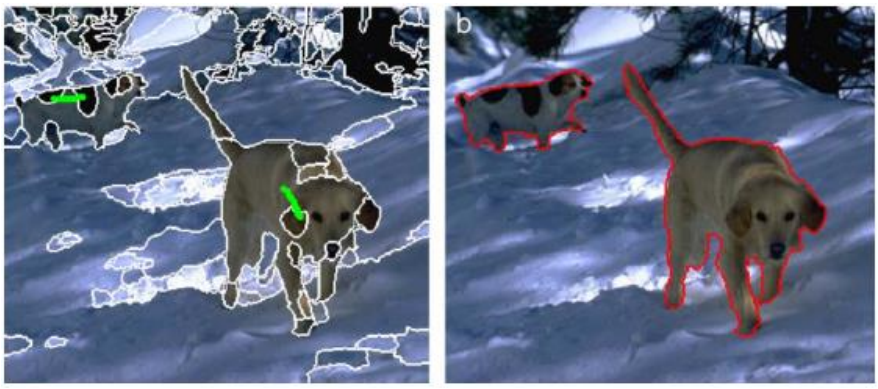
Fonte: Malpica *et al.*, 1997.

3.4 REDUÇÃO DE SOBRE SEGMENTAÇÃO

Conforme foi mencionado ao longo do trabalho, problemas de sobre segmentação geralmente são causados quando se utiliza técnicas de segmentação por *clusterização* de pixels, como por exemplo o *watershed*. Para recapitular, o problema de sobre segmentação ocorre quando a imagem é segmentada em muitas pequenas aglomerações de pixels (*clusters* ou segmentos), logo, o objeto que se deseja segmentar pode (na maioria das vezes) ser formado por vários desses *clusters*. Apesar da união dos *clusters* claramente formar a segmentação do objeto como um todo, o algoritmo não entende dessa forma e segmenta cada *cluster* como se fosse um objeto único, isso é sobre segmentação. O processo da sub segmentação é inverso, as vezes o objeto que se deseja segmentar está contido dentro de um *cluster*, mas o *cluster* também engloba partes de outros objetos e do *background* que se deseja remover da segmentação (THEODORIDIS e KOUTROUMBAS, 2009).

O processo demonstrado pelo MSRM no trabalho de Ning *et al.* (2010) já mencionando na seção anterior, busca aqui criar um método ou regra de mescla (*merge rule*) de regiões com base na comparação de suas similaridades por histogramas, tanto no canal H quanto no S para as regiões analisadas – se forem similares se unem, caso contrário, continuam separadas. O processo consiste em aplicar técnicas como o *Mean-Shift* para uniformizar as texturas antes do processo de segmentação, isso ajuda a evitar a sobre segmentação também. Após a etapa de segmentação, a *merge rule* é usada, aonde o algoritmo de Ning *et al.* (2010) busca usar um coeficiente probabilístico capaz de mensurar os histogramas, nesse caso, o coeficiente de Bhattacharyya é usado (vide Eq. (3.2)), também demonstrado em alta performance por Dubuìsson (2009). O processo de diminuição da sobre segmentação pode ser visto na Fig. 3.8.

Figura 3.8 – Método MSRM. (a) destaque aos dois objetos na cena a serem separados e as regiões de *background* analisadas pela primeira etapa, gerando sobre segmentação; (b) objetos extraídos e regiões de possível *foreground* e *background* mescladas, diminuindo a sobre segmentação.



Fonte: Ning *et al.*, 2010.

$$d(P,Q) = \sqrt{1 - \sum_{i=1}^{N} \sqrt{P(b_i)Q(q_i)}} \tag{3.2}$$

Onde P e Q são os histogramas normalizados dos canais H e S, respectivamente, da mesma imagem de tamanho x, y, e b e q são i-*bins* de cada histograma.

Outras técnicas similares podem ser vistas, como por exemplo no trabalho de Nehaverma (2013), que une a ideia do MSRM com velocidade e otimização do algoritmo, usando o *Multi-Level Threshold*, combinado com o trabalho de Gastal e Oliveira (2011), para aplicar um filtro *edge-aware* mais eficiente do que o Bilateral e o *Mean-Shift* antes da segmentação e também, usando o coeficiente de Chi-Square como método de controle da *merge rule*, que é mais utilizado em aplicações em segmentação de objetos com texturas contrastantes. O processo de redução da sobre segmentação é visto na Figura 3.9.

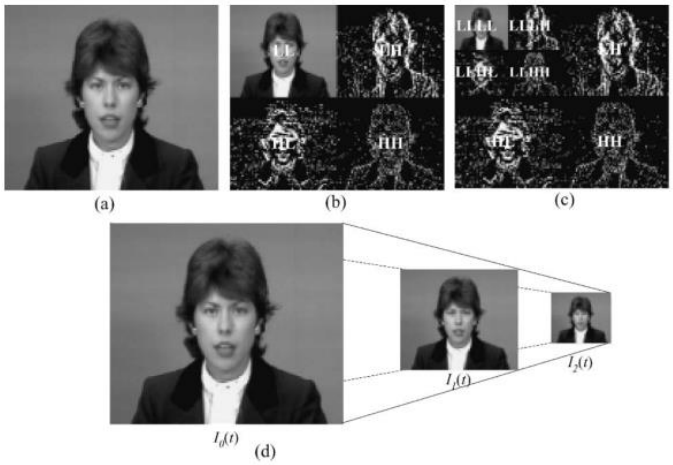
Figura 3.9 – Método aMSRM. (a) imagem original; (b) objetos extraídos e regiões de sobre segmentação criadas; (c) *merge rule* aplicado.

(a) (b) (c)

Fonte: Nehaverma, 2013.

Por fim, o trabalho de Kim e Kim (2003), busca diminuir a sobre segmentação através da aplicação de um *watershed* em multi-escala, que é capaz de uniformizar cores e texturas através da criação de uma pirâmide de resolução construída pela transformada *wavelet* em até duas escalas (também com a preocupação no tempo de execução). A pirâmide pode ser vista na Fig. 3.10.

Figura 3.10 – Representação das escalas criadas pela *wavelet*. (a) imagem original; (b) primeira escala; (c) segunda escala; (d) representação da pirâmide de escalas.



Fonte: Kim e Kim, 2003.

Depois disso, todas as escalas sofrem uma segmentação por *watershed*, e são recombinadas através de uma *wavelet* inversa que recombina os *clusters* criados pela mescla entre as três segmentações. O processo pode ser visto na Fig. 3.11.

Figura 3.11 – Redução da sobre segmentação pelo método de Kim e Kim. 1ª Coluna: segmentação na imagem em escala original; 2ª Coluna: segmentação na primeira escala; 3ª Coluna: segmentação na segunda escala; 4ª Coluna: resultado da mescla das segmentações pelas três colunas anteriores.

Fonte: Kim e Kim, 2003.

Terminada a seção, pode ser visto uma pequena sumarização dos trabalhos pesquisados até o momento.

3.5 SUMARIZAÇÃO

Ao final deste capitulo, foram observados vários trabalhos que procuram, de uma maneira ou outra, buscar os processos de melhoria e segmentação de imagens. Lembrando aqui que não cabe a esses trabalhos ditar o que deve ser feito de agora em diante, mas sim buscar e utilizar parte (ou o conceito) dessas ideias para um melhor direcionamento na pesquisa, a fim de obter um resultado mais fidedigno em relação ao problema dado. Um resumo mais simplificado desses trabalhos pode ser visto nas

tabelas 3.1, 3.2 e 3.3. Cada uma representando uma das seções vistas anteriormente neste capítulo (no caso, melhoria de imagens, segmentação não-supervisionada e redução de sobre segmentação, respectivamente).

Tabela 3.1 – Artigos correlatos para melhoria de imagens.

Autor(es)	**Ano**	**Método Usado**
Rodrigues *et al.*	2015	*Homomorphic Filtering + Color Correction*
Schettini *et al.*	2010	*Local-Dependent Exponencial Correction*
Iqbal *et al.*	2007	*Contrast Stretching in RGB+HSV*

Fonte: Próprio Autor, 2016.

Tabela 3.2 – Artigos correlatos para segmentação.

Autor(es)	**Ano**	**Semi ou Não-Supervisionado**	**Método Usado**
Ning *et al.*	2010	Não	*MSRM*
Grady	2005 e 2006	Não	*Random Walk*
Kim *et al.*	2008	Não	*Random Walk with Restart*
Couprie *et al.*	2011	Sim	*Power Watershed*
Rother *et al.*	2004	Sim	*GrabCut*
Jahangiri e Heesch	2009	Sim	*GrabCut + Refined GMM*
Park *et al.*	2009	Sim	*Graph-Cut + Lazy Snipping*
Khattab *et al.*	2014	Sim	*Automatic GrabCut*
Pratikakis *et al.*	2006	Sim	*Watershed + log-Gabor*
Puzicha *et al.*	1997	Sim	*Watershed + Gabor*
Malpica *et al.*	1997	Sim	*Automatic Watershed*

Fonte: Próprio Autor, 2016.

Tabela 3.3 – Artigos correlatos para redução de sobre segmentação.

Autor(es)	**Ano**	**Segmentador**	**Critério Usado**
Ning *et al.*	2010	*MSRM*	*Merge Rule via Histogram Similarity*
Nehaverma	2013	*Watershed + aMSRM*	*Uniform texture + Histogram Similarity*
Kim e Kim	2003	*Watershed*	*Wavelets segmentation and Merge by recombination*

Fonte: Próprio Autor, 2016.

Algumas técnicas serão testadas para verificar se os métodos podem ser utilizados como partes auxiliares deste trabalho, pois diante da complexidade, muitas adaptações deverão ser feitas e testadas várias vezes. Por hora, será utilizado grande parte do aprendizado visto neste capitulo, combinado com o conhecimento teórico, para realizar os experimentos do modelo proposto.

4 MODELO PROPOSTO PARA SEGMENTAÇÃO NÃO-SUPERVISIONADA

Com base no que foi encontrado nas propostas dos trabalhos correlatos, o autor buscou testar, observar e analisar os métodos estudados em ordem de prover o método proposto descrito neste capitulo, com o intuito justo de obter resultados pretendidos nos objetivos deste trabalho. Nas próximas seções, o leitor irá encontrar como o método proposto foi construído e em quais aplicações este poderá ser utilizado.

4.1 MÉTODO PROPOSTO DE SEGMENTAÇÃO

Nessa seção será discutido o processo de análise proposto pelo autor deste trabalho. Antes de explicar o método, algumas informações precisam ser levadas em conta na hora da aplicação. Entre elas:

- A fase de aquisição de imagens terá pouco detalhamento, apenas será informado como as imagens foram dispostas e capturadas a princípio;
- O banco de imagens (*dataset*) foi criado pelo autor que, posteriormente, fará parte do banco de imagens do Instituto de Computação (IComp) para outras pesquisas. O motivo que levou a criar o banco de dados, e não utilizar o de terceiros, está relacionado ao fato que de todos os trabalhos analisados como correlatos, poucos forneciam o seu *dataset* de imagens, e os mesmos estavam em resoluções muito pequenas (alguns não passavam de 320x240px). Isso levou ao autor buscar a criação de um *dataset* próximo das condições informadas nos artigos;
- Todas as imagens analisadas dentro deste trabalho contêm felinos domésticos em cenário (*background*) ***indoor e outdoor***. O processo de

separar as imagens que contem ou não gatos será foco de trabalhos futuros.

Com base nestes fatos, podemos delimitar o processo de aplicação proposto de forma dedutiva, conforme Prodanov e Freitas (2013) explicam como metodologia básica.

O processo gerará um software que irá realizar o processo de segmentação não-supervisionada do animal selecionado. Esse processo se dará em quatro etapas.

1ª etapa – processo de melhoria da imagem de entrada: o algoritmo proposto deve, antes de partir para etapa de segmentação, aumentar o contraste e melhorar a distribuição de energia da mesma, incluindo a iluminação. Métricas de qualidade de imagens como o EME (*Enhancement Measure* ou Medida de Atenuação) e H (*Entropy* ou Entropia) poderão ser aplicados para analisar se as imagens fonte estão "melhores" em termos de qualidade (métricas como o PSNR - *Peak Signal-Noise Ratio* ou Relação Sinal-Ruído de Pico - ou o AMBE - *Absolute Mean Brightness Error* ou Erro Médio Absoluto de Brilho - não serão usadas nessa etapa, pois essas medidas são mais utilizadas para filtragens onde ocorre a remoção de ruídos, o que não é o foco principal dessa etapa). Em alguns casos, como o processo necessita de uma avaliação subjetiva, a ideia de passar estes filtros iniciais se dá através da validação deste passo e não como um resultado ou critério de escolha das imagens que serão segmentadas.

O tipo de filtragem homomórfica aplicada por Rodrigues *et al.* (2015) será aplicado ao conjunto de todas as imagens antes de serem analisadas pelo segmentador não-supervisionado. Ainda será acrescentado o filtro *Color Boost* (ou Filtro Aumento de Cor), criado pelo autor deste trabalho, com a ideia de aumentar a intensidade de cores mais contrastantes na textura do objeto a ser segmentado sem aumentar tanto a entropia. Essa etapa tem o intuito de prover uma melhor confiabilidade ao sistema. Apesar de que o filtro em alguns casos aumenta a sobresegmentação, a etapa de redução

da sobre segmentação está incorporada a este trabalho justamente para diminuir ou eliminar esse defeito.

Uma aplicação do método proposto por Rodrigues *et al.* pode ser vista na Figura 3.2, no Capítulo três.

2ª etapa – uniformização das texturas e *image sharpen*: nessa etapa, o algoritmo proposto irá tentar buscar a uniformização das texturas na cena, aplicando o método inicial descrito por Ning *et al.* (2010) e Comaniciu e Meer (2002), que é o filtro *Mean-Shift*. Combinado a este último, o autor propõe que o algoritmo também tente preservar as bordas dos objetos, mas não usando o *edge-aware* ou o filtro Bilateral, porém um processo de filtragem gaussiana capaz de ajudar a borrar a uniformização do *Mean-Shift* e aguçar a imagem (*image sharpen*) ao mesmo tempo e sem muito custo computacional.

Essa etapa tem o foco de prover ao processo posterior de segmentação, imagens mais uniformes que ajudem a diminuir a sobre segmentação do objeto. Isso também ajuda a filtrar certos ruídos ou remover conjuntos de pixels "isolados" dentro da textura do gato (o termo "isolados" indica pequenas áreas de tonalidades de cores diferentes dentro do objeto, como manchas e sinais brancos, por exemplo).

3ª etapa – segmentação não-supervisionada: nesta penúltima etapa, o algoritmo proposto busca segmentar a imagem de entrada de maneira automática. Aqui, o modelo original proposto por Malpica *et al.* (1997) será usado e adaptado para aumentar a performance de segmentação e evitar a sobre segmentação em conjunto.

Apesar do autor ter possuído a oportunidade de comentar sobre descritores, como o SIFT e SURF, os mesmos não serão usados por alguns motivos básicos. O primeiro é o fato que o uso desses descritores aumentou muito o tempo de execução, ao qual preocupa-se em ser o mais *real-time* possível; e segundo, as extrações em

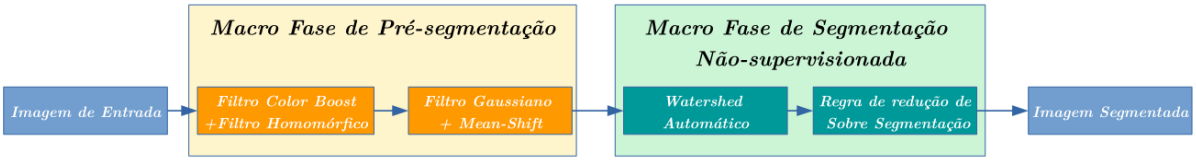
algumas imagens acabam por aumentar a sobre segmentação, gerando muitos *seeds* desnecessários (parcialmente devido aos *backgrounds* das imagens). Medidas de redução foram tomadas (filtros, classificadores, BoW, etc.), mas a aplicação acabou ficando muito longe de ser rápida, fato que invalidaria uma das características do processo não-supervisionado.

4ª etapa – redução de sobre segmentação: sendo essa a etapa final, o algoritmo neste ponto já terá a imagem segmentada e irá prover uma técnica de redução de sobre segmentação. Nesse caso, a redução usada no MSRM (NING *et al.*, 2010) será utilizada. A diferença aqui será a aplicação de um novo coeficiente métrico para a análise de histogramas que irá prover uma melhor comparação em histogramas gerados por texturas – que é o fator que destaca o felino na imagem.

Um outro detalhe que pode chamar a atenção do leitor é a respeito do título do trabalho ser sobre a etapa de segmentação, mas alguns desses processos descritos na etapa 3 em diante, foram parcialmente desenvolvidos por outros trabalhos correlatos. O que leva ao autor dizer que o método sirva como um método aprimorado de segmentação em geral, vem do fato de que a distinção exata das etapas de segmentação varia bastante de autor para autor. Por exemplo, Gonzalez e Woods (2010), consideram a etapa de segmentação somente o processo dos segmentadores como o *Watershed* por exemplo. Theodoridis e Koutroumbas (2009) afirmam que o processo de segmentação para a classificação envolve todas as etapas posteriores a extração de características. E Duda *et al.* (2001) corrobora com as afirmações de Theodoridis e Koutroumbas, mas afirmam que a extração de características faz parte da segmentação também. Por isso, o autor deste trabalho preferiu seguir as afirmações desses outros autores e considera que as etapas 1 e 2 deste trabalho, como parte da segmentação (ao qual é chamada de pré-segmentação).

Visto isso, a Figura 4.1 mostra um fluxograma que resume todas as etapas comentadas (etapas 1 e 2 estão mescladas numa macro etapa chamada de pré-segmentação).

Figura 4.1 – Fluxograma de etapas do método proposto.



Fonte: Próprio Autor, 2016.

Nas próximas seções, será explanado melhor cada etapa com base na aplicação em imagens do banco de dados.

4.2 OBTENÇÃO DO BANCO DE IMAGENS

Inicialmente, é necessário conhecer como é formado o banco de imagens e como este foi adquirido. Primeiramente, o banco de dados foi adquirido com as seguintes características:

- Imagens de gatos domésticos em cenários *indoor* e *outdoor*;
- 664 imagens catalogadas até fevereiro de 2016, mais 356 imagens em março de 2016. Totalizando 1020 imagens;
- 10 indivíduos (gatos diferentes) e pode haver mais de um gato na mesma imagem;
- Câmeras DSLR Canon Rebel EOS T3i (mar/16) e Nikon D5200 (fev/16), resposta em 2ms do obturador e com lente objetiva AF-S NIKKOR 18-55mm, abertura $f/4.5$, ISO variada;

- Resolução de cada imagem: 6000x4000px (fev/16) e 1920x1080px (mar/16) (sendo que as mesmas foram reduzidas para 720x480px antes de serem segmentadas pelo algoritmo);
- Formato JPEG em alta qualidade;
- Sem uso de Flash ou HDR (*High Dynamic Range* ou Grande Alcance Dinâmico). Condições de iluminação natural do ambiente;
- Espaço de Cor sRGB.

Todos os gatos foram fotografados em posições naturais e a câmera foi posta em um tripé para garantir que as fotos não ficassem tremidas ou com distorções em relação ao movimento do operador (em alguns casos, as fotos que foram tiradas próximas do chão, não foi utilizado o tripé).

A demonstração de como foi obtida uma foto pode ser vista na Figura 4.2.

Figura 4.2 – Demonstração da obtenção do banco de imagens.

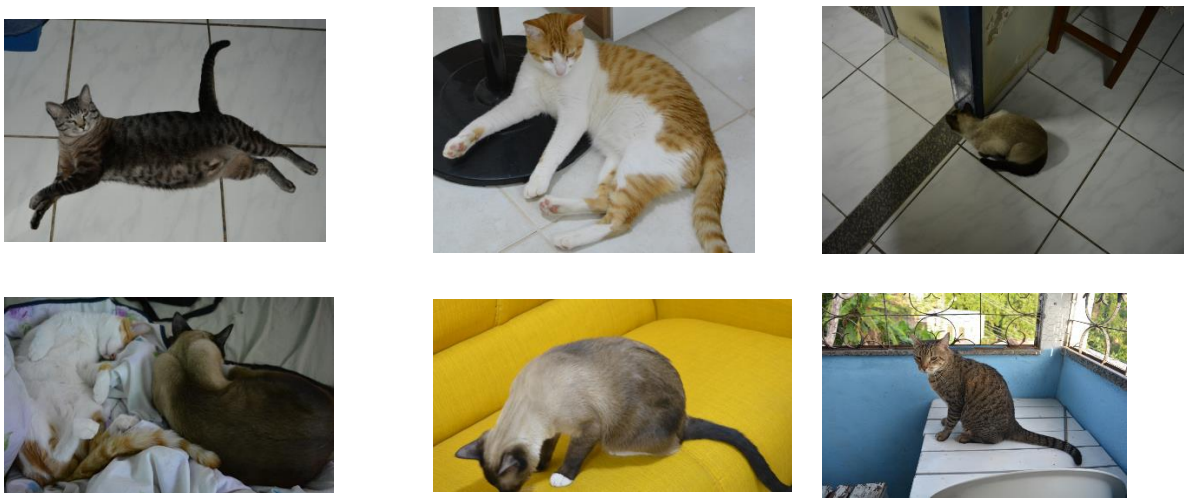
Fonte: Próprio Autor, 2016.

Uma amostra das imagens usadas pode ser vista na Figura 4.3.

Como visto pela Fig. 4.3, a preocupação do autor na criação do banco vem da heterogenia de posições/ângulos e *backgrounds* aonde os objetos estão locados. Isso é importante para PDI e VC (FORSYTH e PONCE, 2012), e não para a classificação em geral.

Após a apresentação deste trabalho, tanto o algoritmo, quanto o banco de imagens, serão dispostos no GitHub: https://github.com/hyperiondeimos.

Figura 4.3 – Exemplos de imagens do banco.



Fonte: Próprio Autor, 2016.

4.3 ETAPA DE MELHORIA DE IMAGENS

Nesta etapa, o método propõe o aumento da intensidade de cores com o filtro *Color Boost*, para depois ser aplicado o filtro homomórfico do trabalho de Rodrigues *et al.* (2015). A diferença aqui é a substituição da *Fast Fourier Transform* pela *Discrete Cosine Transform* (DCT ou Transformada de Cossenos Discretos), aonde a mesma busca aumentar a concentração de iluminação sobre as altas frequências (bordas) com base na aplicação de cossenóide fundamental do filtro *Color Boost*.

4.3.1 Filtro *Color Boost*

Neste ponto, a ideia de prover um melhor destaque para certas cores, moveu o autor a criar um filtro não-linear capaz de intensificar cores frias e quentes ao mesmo tempo, diminuindo o efeito de cores amareladas (que geralmente causam artefatos na

imagem) para poder prover uma eficiência maior na etapa da filtragem homomórfica e também, aumentar o destaque dos gatos nas cenas (ou dos objetos mais contrastantes). Em suma, o filtro criado aqui pelo autor busca aumentar o contraste sem aumentar tanto a entropia da imagem.

Com base nisso, foi aplicado a **teoria de controle** para prover um método de aumento e correção da intensidade do sinal por meio de um processo de **amortecimento de resposta de 2ª ordem**, descrito claramente por Nise (2015). Aonde a passagem do máximo sobressinal retorna o valor máximo de intensidade necessária para obter o valor desejado, nesse caso a cor. Além de amortecer o processo de entropia por um ganho consistente de contraste para cada pixel da imagem, com base na distribuição global de média e variância. A derivação do filtro vem da Eq. (4.1).

$$P_{max} = 1 - e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)}\left(\cos\omega + \frac{\zeta}{\sqrt{1-\zeta^2}}\operatorname{sen}\omega\right) = 1 + e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)} \tag{4.1}$$

Onde se o valor de ω atingir o valor máximo de frequência π, gera a igualdade descrita na Eq. (4.1). Onde o valor de P é proporção da intensidade que o pixel pode alcançar e ζ é o coeficiente de amortecimento que irá variar de 0 a 1 (pois a radiciação não está em módulo e por isso valores maiores que 1 irão gerar senoides exponenciais instáveis). Para se encontrar o filtro, agora é necessário encontrar a relação entre o valor máximo que o pixel pode chegar em relação ao valor final do pixel encontrado na imagem de entrada (P_{final}), assim $F_B = \left(P_{max} - P_{final}\right)/P_{final}$. Se uma entrada *heaviside* for aplicada como sinal básico de convolução ao pixel que se quer atenuar, $P_{final} = 1$ e a Eq. (4.1) fica simplificada, conforme (4.2).

$$F_B = e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)} \tag{4.2}$$

Onde F_B é o filtro *Boost* não normalizado. Em ordem de aumentar a sua validação, uma normalização é feita em conjunto com o filtro. A normalização é feita usando uma análise de média da intensidade do pixel analisado. Por isso, a normalização é feita multiplicando o filtro *Boost* pelo termo de norma, gerando (4.3).

$$F_B = \left(I_{x_i}/M_p\right)e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)} \tag{4.3}$$

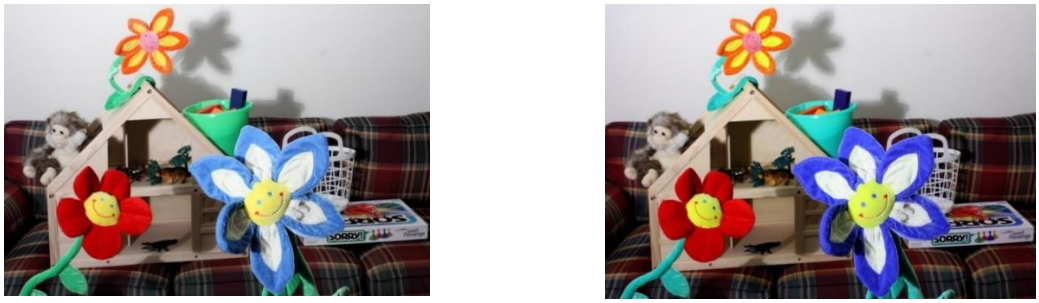
Onde M_p é a média de intensidade de todos os pixels da imagem e I é a imagem de tamanho $x = (n, m)$ bidimensional no exato pixel i a ser intensificado. Conforme o filtro é aplicado, percebe-se que o mesmo varia muito devido à falta de um ajuste fino causado pelo amortecimento sem um controlador do tipo Integrador. Assim, é possível adicionar o ajuste fino a equação do filtro, conforme (4.4). O módulo é aplicado para impor números reais positivos na aplicação do ajuste fino.

$$F_{CB} = \left|\delta \log(\zeta + 1) - \frac{I_{x_i}}{M_p}\right| e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)} \tag{4.4}$$

Onde δ é um coeficiente de ajuste (os exemplos mostrados na Figura 4.4 usam o valor de 20 como ajuste). Gerando assim o filtro F_{CB} chamado de Filtro ***Color Boost***. A aplicação é feita pixel a pixel nos canais H e S da imagem a ser melhorada. Uma amostra subjetiva do efeito do filtro em comparação a uma imagem com melhoria de cores feitas por *stretching* e normalização do histograma em cada canal RGB, pode ser vista na Figura 4.4.

É possível perceber que a figura 4.4b possui uma melhoria subjetiva de destaque das cores azul (fria) e vermelho (quente), e tenta diminuir o amarelo fazendo a saturação se inverter com o filtro, isso irá ajudar a próxima etapa. O resultado foi obtido usando o $\zeta = 0.63$.

Figura 4.4 – Exemplo do filtro *Color Boost* em comparação com *stretching* e normalização de histograma no RGB. (a) imagem com *stretching* e normalização de histograma; (b) Filtro *Color Boost*.



(a) (b)

Fonte: Próprio Autor, 2016.

Com base na aplicabilidade do filtro, randomicamente será selecionado 3 imagens do banco para serem comparadas com as métricas EME e H. Veja que o objetivo desse trabalho não é gerar imagens melhores, mas sim segmentar. Então, para apenas verificar se houve melhora no contraste, esse processo será aplicado em apenas 3 imagens. A Figura 4.5 mostra uma comparação entre as imagens colhidas.

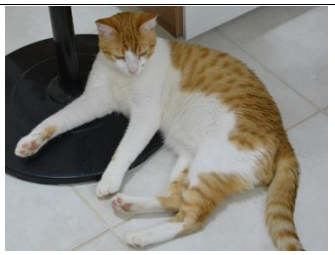
Percebe-se que o resultado do *Color Boost* não é para deixar as imagens melhores ou mais bonitas, e sim para aumentar a intensidade em alguns pixels de cores dominantes. Assim, a Tabela 4.1 mostra os valores de EME e H para as imagens analisadas, quanto maior o resultado de EME e menor a variação de H, melhor.

Contudo, é possível utilizar o filtro para melhorar as cores de forma subjetiva. Para isso, o filtro leva uma mudança, descrita pela na Equação (4.5). Atribuindo a homogeneização do filtro gaussiano na equação.

$$F_{ACB} = \left| \delta \log(\zeta + 1) - \frac{I_{x_i} M_p}{2\pi\sigma} \right| e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)} \tag{4.5}$$

Onde σ é o desvio padrão da imagem I, formando o filtro ***Active Color Boost***. Tanto na Eq. (4.4) quanto na Eq. (4.5), o resultado é mesclado através da proporção dada por (4.6).

Figura 4.5 – Imagens do banco usadas para validação do filtro. Na coluna esquerda: imagens originais. Na coluna direita: imagens com o filtro *Color Boost* com $\zeta = 0.6$ e ajuste do *ColorMap* em todos os canais HSV.

Image ID	*Original*	*Color Boost*
1		
2		
3		

Fonte: Próprio Autor, 2016.

Tabela 4.1 – Resultado da análise objetiva com EME e H.

Image ID	*Métrica*	*Original*	*Color Boost*
1	EME	3.9439	8.1548
	H	7.2397	7.5416
2	EME	5.2058	7.6552
	H	7.2964	7.7892
3	EME	2.8251	5.7882
	H	6.9296	7.5636

Fonte: Próprio Autor, 2016.

$$IF_{x,y} = \alpha \cdot I_{x,y} + \beta \cdot F_{ACB_{x,y}} + \gamma \tag{4.6}$$

Onde α e β são coeficientes de peso e γ é o coeficiente de correção, I é a imagem original de tamanho (x, y) e IF é a imagem filtrada. Na Figura 4.6 pode-se perceber uma melhora das cores de forma bem mais significante. O autor escolheu a imagem com "maior variação" de cores (subjetivamente) da Fig. 4.5 para demonstrar a eficácia do filtro em aplicações futuras.

Figura 4.6 – Exemplo do filtro *Active Color Boost* em comparação com *o* filtro *Color Boost* original. (a) Imagem original; (b) Filtro *Color Boost* + ajuste do *ColorMap* nos canais H, S e V; (c) Filtro *Active Color Boost* com ajuste de iluminação apenas no canal V.

(a) (b) (c)

Fonte: Próprio Autor, 2016.

Os valores de α, β e γ usados foram: 0.13, 0.87 e 0.0 respectivamente. Além do amortecimento $\zeta = 0.63$.

Percebe-se, subjetivamente, que a Fig. 4.6c possui uma melhoria na cor de maior destaque na cena, deixando mais "viva" e quente em relação ao cenário relatado. Ambos os filtros podem ser usados no processo da filtragem homomórfica, sem problemas.

Por isso, o filtro pode atingir um range de operações com base no amortecimento aplicado na Eq. (4.5), isso se for aplicado em ambos os canais H e S. Pode-se ver na Figura 4.7 os efeitos causados pela mudança no amortecimento na imagem.

Na Fig. 4.7, a imagem à esquerda tem o amortecimento nulo, permitindo alta variação de frequência e geração de pixels pretos e brancos, saturando a imagem e

gerando uma resposta oscilatória (valores complexos); na imagem do meio, o amortecimento fica entre 0 e 1 e provê a imagem uma melhoria de contraste com base no fator de amortecimento (mais baixo, mais pálida a imagem fica; mais alto, mais quente a imagem fica); na direita, o amortecimento é crítico e gera um aumento muito alto de cores mais avermelhadas. Se combinado com o canal S, a saturação é amortecida de tal forma que os valores de cor ficam nulos e a imagem fica em tons de cinza.

Vale ressaltar que os algoritmos do *Color Boost* e do *Active Color Boost*, ambos feitos no MATLAB®, estão disponíveis no Apêndice A deste trabalho.

Figura 4.7 – Filtro *Active Color Boost* gerando vários tipos de respostas quando aplicado diferentes valores de amortecimento.

Fonte: Próprio Autor, 2016.

4.3.2 Filtragem Homomórfica com DCT

Nessa parte, buscou-se utilizar da técnica descrita por Rodrigues *et al.* (2015) para tentar suavizar a iluminação da cena. Porém, no trabalho de Stockham Jr. (1972), o mesmo demonstrava que o uso da *Discrete Cosine Transform* ou DCT em imagens, caracteriza uma maior concentração de altas frequências no centro do espectro. Isto levou ao autor, além de criar o filtro anterior para atuar em altas frequências, fazer

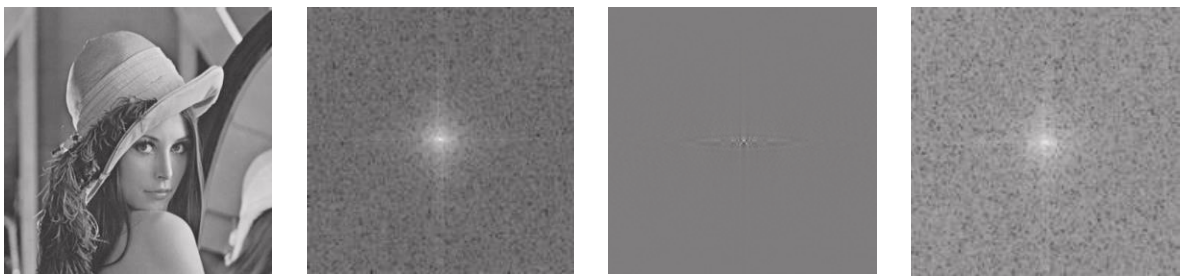
com que a distribuição da aplicação homomórfica fique melhor quando o filtro é utilizado em conjunto com o *Active Color Boost* (devido a aplicação de senoides ao longo do filtro inicial). A Equação (4.7) descreve como o espectro de uma imagem pode ser encontrada com a DCT.

$$DP_{u,v} = \begin{cases} \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} P_{x,y} & \text{se } u = v = 0 \\ \frac{2}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} P_{x,y} \times \cos\left[\frac{(2x+1)u\pi}{2N}\right] \times \cos\left[\frac{(2y+1)v\pi}{2N}\right] & \text{c. c.} \end{cases} \tag{4.7}$$

Onde N são todos os pontos na imagem P de tamanho (x, y), e DP é a magnitude espectral da imagem, com tamanho (u, v).

A Figura 4.8 demonstra uma comparação de espectros de uma imagem, com outras transformadas de frequência.

Figura 4.8 – Demonstração de espectros obtidos da imagem Lena usando transformadas de frequência diferentes.



Lena | Fourier | *Discrete Cosine* | Hartley

Fonte: Nixon e Aguado, 2012.

Com base nesse conhecimento, buscou-se combinar as duas técnicas para prover um filtro homomórfico que concentre a iluminação da imagem em torno das bordas. A diferença aqui, é que com o uso anterior do filtro *Color Boost*, a aplicação desse efeito será claramente intensificada pelo objeto de maior contraste na cena – gerando um efeito de brilho bem peculiar ao redor do possível *foreground*.

Para obter esse efeito, após obter o espectro pela DCT, usa-se um filtro de alta ênfase que será convolucionado com o espectro da imagem em questão. A Eq. (4.8), descrita por Diniz (2014), define o filtro.

$$T_{u,v} = DP_{u,v} * HF_{u,v} \tag{4.8}$$

Onde HF é o filtro de alta ênfase para cores, descrito por (4.9).

$$HF_{u,v} = \left[\left(1 - \frac{1}{1 + e^{r-\tau}}\right)(U - L) + L\right] \tag{4.9}$$

Onde $r^2 = u^2 + v^2$ é a distância Euclidiana entre o ponto (u, v) do espectro analisado e o centro do espectro. O τ é o limiar de corte usado na correção da frequência. E por fim, U e L são, respectivamente, os valores dos limites máximo e mínimo de frequências do espectro obtido da imagem analisada. Depois disso, a transformada inversa (*Inverse Discrete Cosine Transform* ou IDCT) é aplicada para obter a imagem final. A IDCT é definida por (4.10).

$$IF_{x,y} = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} DP_{u,v} \times \cos\left[\frac{(2x+1)u\pi}{2N}\right] \times \cos\left[\frac{(2y+1)v\pi}{2N}\right] \tag{4.10}$$

Para demonstrar o uso dessa filtragem, a Figura 4.9 mostra uma comparação entre os dois métodos analisados, o original de Rodrigues *et al.* (que usa a FFT) e o método de Rodrigues *et al.* com o uso da DCT em vez da FFT. Os resultados obtidos nesta imagem (Fig. 4.9c) usam os parâmetros de $U = 2.0$, $L = 0.5$ e $\tau = 0.65$ como *inputs* básicos.

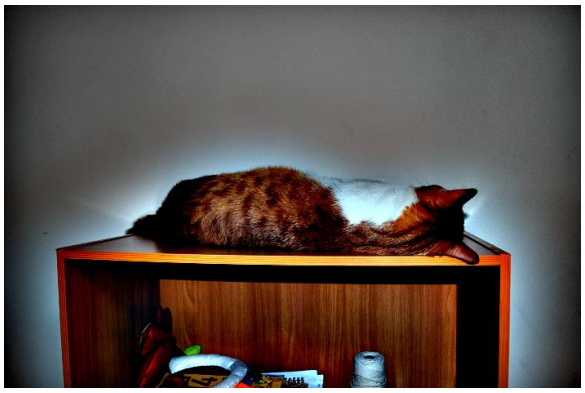
Figura 4.9 – Aplicação da filtragem homomórfica em tons de cinza. (a) Imagem original; (b) Filtragem homomórfica de Rodrigues *et al.* (com FFT); (c) Filtragem homomórfica proposta (com DCT + *Color Boost*).

(a) (b) (c)

Fonte: Próprio Autor, 2016.

Para dar uma boa sensação de intensidade da iluminação, as imagens filtradas foram deixadas em tons de cinza. Na Fig. 4.9c é visto o efeito de brilho em torno dos objetos mais contrastantes na cena. Neste caso em especifico, o gato e a estante possuem uma intensidade de cor similar e, por isso, ambos são "destacados" (vide Fig. 4.9a). A Figura 4.10 demonstra a versão colorida da Fig. 4.9c. Nota-se uma intensificação em cores mais azuladas. Essa diferença na intensificação da cor será removida com a etapa seguinte.

Figura 4.10 – Aplicação da filtragem homomórfica com a DCT – versão colorida.



Fonte: Próprio Autor, 2016.

Na Figura 4.11 demonstra a diferença entre a filtragem proposta com e sem o uso do *Color Boost*. Note que o espectro de intensidade de brilho da imagem com o

Color Boost é amplificado, permitindo que os contornos fiquem mais visíveis a etapa posterior de detecção de bordas externas.

Figura 4.11 – Comparação entre aplicações da filtragem homomórfica com a DCT. (a) Imagem com o *Color Boost*; (b) Imagem sem o *Color Boost*.

(a) (b)

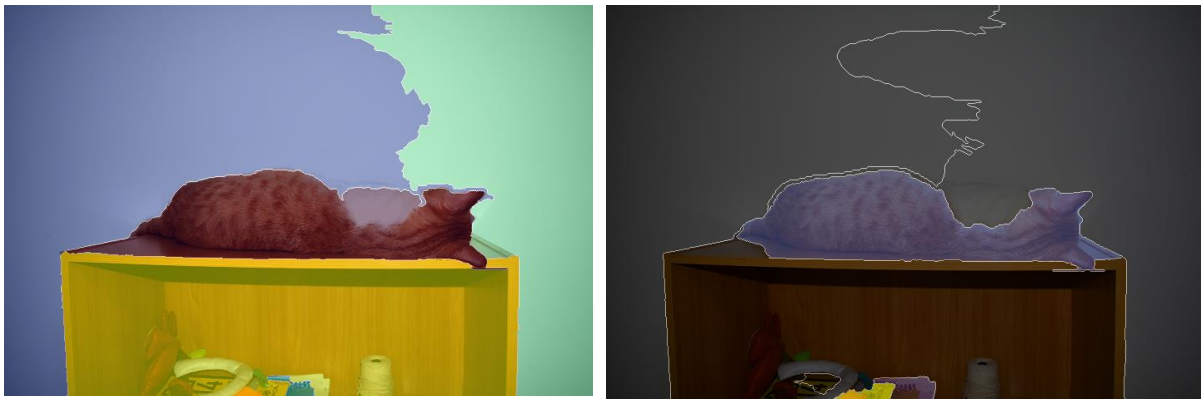
Fonte: Próprio Autor, 2016.

Percebe-se também na Fig. 4.11b que o efeito de brilho em torno gato é menor em espessura do que a Fig. 4.11a. Por isso o uso do *Color Boost* é importante, pois ele ajuda a delimitar melhor o contorno que se deseja extrair, ajudando a evitar a subsegmentarão justamente por utilizar melhor a concentração de senoides fundamentais com o uso da DCT em vez da FFT.

Para que se haja uma ideia da diferença no resultado final, pula-se aqui para conclusões com a Figura 4.12. Aonde pode ser visto a comparação entre o resultado final das etapas de segmentação pelo método proposto com e sem o filtro *Color Boost*. Apesar da Fig. 4.12a conter uma parte maior da estante do que a Fig. 4.12b. A segmentação do gato foi quase "completa", o que em termos de qualidade da segmentação, é o melhor resultado a ser obtido.

Na próxima etapa, o uso do *Mean-Shift* irá prover uma uniformização das texturas em ordem de melhorar a redução da sobre segmentação.

Figura 4.12 – Resultado final da segmentação para comparação. (a) Imagem com o *Color Boost*; (b) Imagem sem o *Color Boost*.



(a) (b)

Fonte: Próprio Autor, 2016.

4.4 ETAPA DE UNIFORMIZAÇÃO DE TEXTURAS

Para resolver o problema da sobre segmentação, a etapa de uniformização das texturas é aplicada, conforme foi recomendado por Ning *et al.* (2010). Nesta última etapa da fase de pré-segmentação, o filtro *Mean-Shift* será usado para remover ruídos e alguns artefatos remanescentes de aplicações anteriores. Além de prover um processo de homogeneização das texturas.

Antes de aplicar o filtro *Mean-Shift*, primeiro é necessário receber um tratamento para as altas frequências, pois a etapa anterior acaba condicionando a criação de novas "bordas" (devido a mudança do espectro de energia) e acaba também criando pontos esbranquiçados ao longo das bordas "reais".

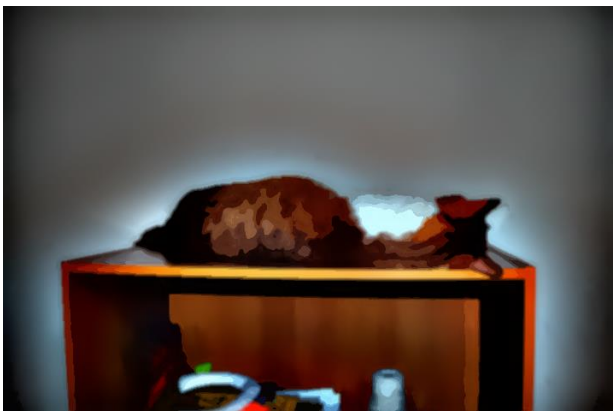
Assim, a aplicação do filtro *Gaussian Blur* é requerida. O filtro irá remover esses ruídos gerados na origem e carregados pela etapa anterior e irá prover também um aguçamento na imagem para manter as bordas enquanto o filtro *Mean-Shift* uniformiza as texturas. O processo é parecido com o filtro Bilateral, mas testes comprovaram que

o resultado desse processo é melhor e mais rápido. Para isso obter o resultado esperado, a Eq. (4.6) é usada novamente, modificada para atender a imagem filtrada e descrita em (4.11).

$$M_{x,y} = \alpha \cdot G_{x,y} + \beta \cdot IF_{x,y} + \gamma \tag{4.11}$$

A imagem M é o resultado da fusão entre G (Imagem com o *Gaussian Blur*) e IF (Imagem com o filtro Homomórfico), todas de tamanho (x, y). Nesse caso, $\alpha = 1.5$, $\beta = -0.5$ e $\gamma = 0.0$. Pode-se observar a imagem filtrada por essa etapa na Figura 4.13.

Figura 4.13 – Aplicação do filtro *Mean-Shift*.



Fonte: Próprio Autor, 2016.

Percebe-se observando a Fig. 4.13 que muitos ruídos foram eliminados, especialmente o detalhe da acentuação da cor azul/verde que ficava nas bordas perto do efeito de brilho gerado anteriormente. Agora os mesmos não são mais considerados um problema para as próximas etapas.

Processos similares poderiam ter sido obtidos através do filtro Bilateral e do filtro de *edge-aware*, ambos citados no Capitulo dois. Porém, além dos mesmos proverem uma demora a mais na execução do resultado, os mesmos atenuaram a sobre segmentação. Isso devido ao fato de que o efeito de brilho iria prover a esses algoritmos uma forma de outro objeto, criando mais bordas do que se deveria existir.

4.5 ETAPA DA SEGMENTAÇÃO NÃO-SUPERVISIONADA

Após sair da macro etapa de pré-segmentação, que envolvia as etapas de melhoria e uniformização da imagem. Agora a primeira fase da macro etapa de segmentação começa com a segmentação não-supervisionada. Aqui, o método de Malpica *et al.* (1997), Beucher (1992) e as aplicação descritas na obra de Solomon e Breckon (2013) são utilizadas para compor os passos de obtenção da segmentação automática requerida. Algumas melhorias foram feitas com base na análise observacional de cada etapa, afim de obter um processo computacional mínimo justamente para otimizar o algoritmo.

Usando a imagem derivada da etapa anterior como entrada, o método do uso da transformada da distância irá gerar automaticamente as *seeds* que colocaram as marcas iniciais para o *watershed* começar a segmentar as regiões. Em suma, para encontrar as marcas, as etapas são essas:

Passo 1: imagem é passada para tons de cinza.

Passo 2: é aplicado um método de limiarização, como o Otsu ou o Canny. Combinado com um processo de limiarização binaria inversa.

Passo 3: aplica-se a abertura morfológica no resultado anterior para eliminar áreas pequenas e isoladas.

Passo 4: aplica-se a Transformada de Distância e normaliza o histograma.

Passo 5: aplica-se a limiarização binaria comum para diminuir o tamanho dos *blobs* que serão usados para gerar as marcas.

Passo 6: normaliza novamente o histograma.

Passo 7: encontra as marcas com base nos contornos extraídos das novas formas criadas.

Passo 8 (opcional): utiliza-se o SURF para extrair características da imagem e posteriormente mesclar os resultados com o passo 7.

Percebe-se que alguns passos são diferentes dos métodos propostos por Malpica *et al.* (1997), em especial nos passos 2, 3, 4 e 5 (e também no 8, mas como esse é opcional, não será comentado). Para resumir, o passo 2 utiliza uma técnica combinada de limiarização binaria inversa com o método de Otsu ou Canny (o usado nos resultados é o de Otsu). A Equação (4.12) demonstra como a limiarização é considerada.

$$I_{x,y} = \begin{cases} 0 & \text{se } P(x,y) > t \\ \text{valor máx.} & \text{caso contrário} \end{cases} \tag{4.12}$$

Onde I é a imagem de tamanho (x, y), P é a posição do pixel e t é o valor de limiarização. E t é automaticamente calculado pelo método de Otsu. A Equação (4.13) demonstra o método de Otsu descrito por Gonzalez e Woods (2010).

$$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{4.13}$$

Onde σ_ω é a soma ponderada das variâncias entre duas classes (0 e 1). $\omega_{0,1}$ são as probabilidades das classes separadas pelo limiar t e $\sigma_{0,1}$ são suas variâncias. As probabilidades podem ser calculadas em função de vários histogramas L vistos em (4.14).

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \;\; e \;\; \omega_1(t) = \sum_{i=t}^{L-1} p(i) \tag{4.14}$$

E por fim, o valor do limiar pode ser calculado por (4.15).

$$\sigma_b^2 = \sigma^2 - \sigma_{\omega}^2 = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \tag{4.15}$$

Onde o valor máximo obtido pela variância binária σ_b será o limiar a ser passado ao algoritmo de limiarização. E $\mu_{0,1}$ são as médias das classes, que podem ser calculadas por (4.16).

$$\mu_0(t) = \sum_{i=0}^{t-1} \frac{i \cdot p(i)}{\omega_0} \quad e \quad \mu_1(t) = \sum_{i=t}^{L-1} \frac{i \cdot p(i)}{\omega_1} \tag{4.16}$$

O pseudocódigo 4.1 demonstra como o limiar é aplicado pelo método de Otsu.

Pseudocódigo 4.1: *Limiarização Otsu + Binary Inverse*

Entrada: imagem em tons de cinza $G_{x,y}$

Saída: imagem binarizada/limiarizada $B_{x,y}$

1: Computa o histograma L_i e as probabilidades p_i para cada nível de intensidade da imagem G.
2: Aplica o valor inicial de $\omega_i(0)$ e $\mu_i(t)$.
3: Para todos os possíveis limiares t de 1 até valor de intensidade máxima, faça
4: Atualize ω_i e μ_i.
5: Compute σ_b^2
6: Encontre o valor máximo de $\sigma_b^2(t)$
7: $t \leftarrow \sigma_{b_{max}}^2(t)$.
8: $B \leftarrow BinaryThresholdInverse(G, t)$
9: Normaliza histograma de B.

O passo 3 foi adotado através da observação de cada etapa do processo, visto que em algumas imagens, o processo de limiarização gera várias "ilhas" de pixels que não significaram em nada na segmentação final e que só aumentarão a sobre segmentação. A abertura morfológica foi tomada como medida mais eficaz para eliminar ao máximo esses *blobs* de pixels.

No passo 4, usa-se a Transformada de Distância ou o Mapa de distância. Em termos simples, a transformada em questão apenas busca encontrar valores internos de formas obtidas pela etapa da limiarização. A partir de uma métrica de distância, como Manhattan ou Euclidiana, uma série de pixels associados ao centroide da forma são

encontrados e posteriormente atribuídos ao maior valor binário que a forma em si contém. A medida que a distância entre pixels aumenta em relação ao conjunto de pixels próximos do centro da forma, os pixels mais distantes vão sendo ponderados a valores menores que os pixels ao centro. Formando uma espécie de mancha ou esqueleto da forma. Um processo similar pode ser visto na Figura 4.14, aonde um quadrado branco no meio de um fundo preto (4.14a), tem a aplicação da transformada da distância realizada (4.14b).

Figura 4.14 – Exemplo do Mapa de distância. (a) Imagem binarizada; (b) Imagem com o Mapa de distância aplicado.

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	2	2	2	1	0
0	1	2	3	2	1	0
0	1	2	2	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

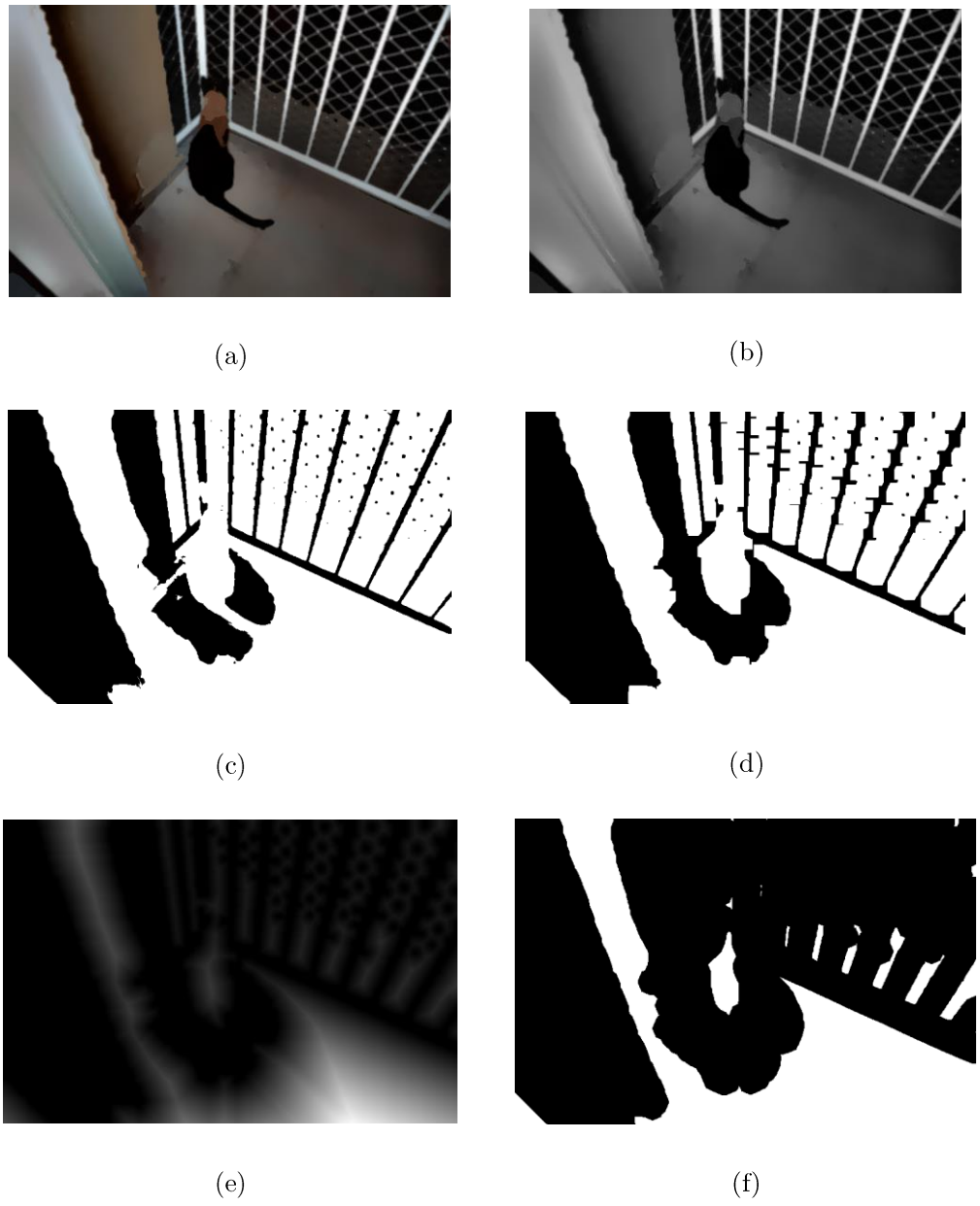
(b)

Fonte: Próprio Autor, 2016.

No passo 5, a aplicação da nova limiarização faz com que o tamanho dos contornos das formas geradas seja diminuído significantemente para evitar a criação de sobre segmentos em torno de outros objetos ou entre objetos. Além de otimizar o segmentador na geração das marcas.

Com todos os passos explicados, a Figura 4.15 demonstra como as operações aparentam ser passo-a-passo nas imagens dos gatos.

Figura 4.15 – Passo-a-passo da etapa de segmentação não-supervisionada. (a) Imagem filtrada; (b) Imagem em tons de cinza; (c) Imagem binarizada por Otsu; (d) Abertura morfológica; (e) mapa de distância e; (f) nova limiarização simples.



(a) (b)

(c) (d)

(e) (f)

Fonte: Próprio Autor, 2016.

Com base nisso, o próximo passo é aplicar a transformada *watershed* usando as marcas obtidas no passo 7, a partir da extração dos centroides de cada forma obtida na Fig. 4.15f. O processo da aplicação não-supervisionada da *watershed*, pode ser vista na Figura 4.16.

Figura 4.16 – Aplicação do segmentador não-supervisionado com a transformada *watershed*.

Fonte: Próprio Autor, 2016.

Percebe-se que a *watershed* cria regiões de segmentação com cores distintas. Cada região mostrada na Fig. 4.16 é considerada uma porção completamente independente das regiões adjacentes ou distantes. Perceba que por causa das etapas anteriores, o processo diminuiu bastante a sobre segmentação (fique atento ao gato, que está quase todo "unido" em uma mesma região, mesmo com partes contrastantes na sua textura, isso foi devido ao efeito de brilho obtido pela fase homomórfica).

Para recombinar as regiões adjacentes ou não, na próxima e última fase, será visto o processo de redução da sobre segmentação.

4.6 ETAPA DA REDUÇÃO DE SOBRE SEGMENTAÇÃO

Por fim, a última etapa do método consiste em aplicar uma regra para reduzir a sobre segmentação nas imagens. O processo leva em conta a regra desenvolvida por Ning *et al.* (2010) no segmentador MSRM, posteriormente melhorada por Nehaverma (2013), ambos já comentados no capitulo 2. A diferença aqui é que o método de comparação dos histogramas nos canais H e S não é feita pelo operador Battacharrya. Em vez disso, o operador usado é uma versão alternativa do operador Chi-Square, que

segundo Puzicha *et al.* (1997) é altamente recomendado para comparações em áreas bem texturizadas. Além do mesmo prover uma aceleração na execução do algoritmo. A Eq. (4.17) mostra como o operador Chi-Square Alternativo pode ser encontrado.

$$d(H_h, H_s) = 2 * \sum_{I_r} \frac{[H_h(I_r) - H_s(I_r)]^2}{H_h(I_r) + H_s(I_r)} \tag{4.17}$$

Onde d é o operador de distância de medida entre os dois histogramas normalizados, H_h e H_s, que representam os histogramas dos canais H e S, respectivamente. I_r é a região segmentada em questão. O pseudocódigo 4.2 descreve o processo de como a regra de redução é feita.

Pseudocódigo 4.2: *Regra de redução da sobre segmentação*

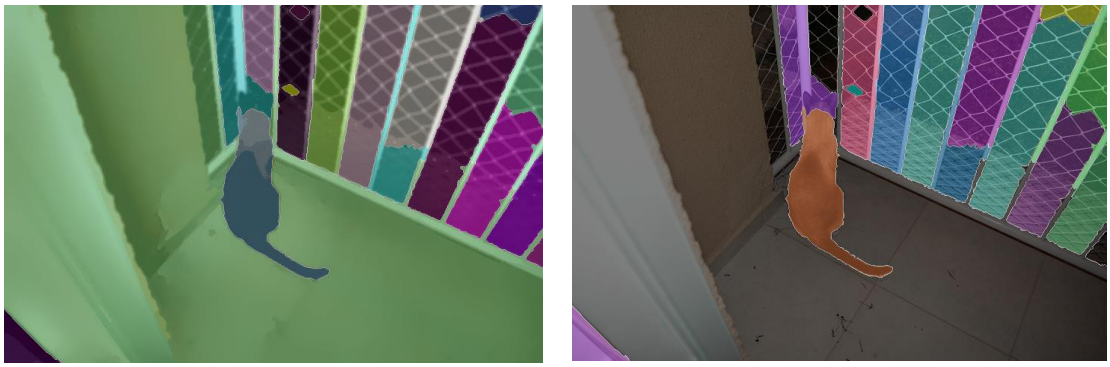
Entrada: número de segmentos n e *cluster* de pixels de cada segmento da etapa anterior

Saída: segmentos mesclados

1: Cria as filas H_c e H_q para armazenar os histogramas
2: Calcula os histogramas de cada segmento nos canais H e S
3: Normaliza todos os histogramas de cada segmento
4: $H_c \leftarrow$ Histogramas normalizados do canal H de cada segmento
5: $H_q \leftarrow$ Histogramas normalizados do canal S de cada segmento
6: Para H_c igual a 1 até $n - 1$, faça
7: Para H_q igual a 1 até $n - 1$, faça
8: Se os *clusters* de segmentos não estão mesclados, então
9: Compare os histogramas de cada segmento analisado usando o método Chi-Square Alternativo
10: Se a similaridade obtida pela comparação for próxima, então
11: Mescle os segmentos
12: $n \leftarrow n - 1$.
13: Retorne os elementos mesclados

Para uma comparação mais visual, a Figura 4.17 mostra um comparativo entre o resultado da última etapa com a regra de redução da sobre segmentação feita.

Figura 4.17 – Resultado final da etapa de redução da sobre segmentação. (a) Imagem segmentada; (b) Aplicação da regra de redução.



(a) (b)

Fonte: Próprio Autor, 2016.

Nota-se que na Fig. 4.17b, existem regiões identificadas com mesmas “cores”. Nesse caso, as regiões com a mesma identificação são consideradas um único segmento, mesmo as regiões que não são conexas entre si. Em comparação com a Fig. 4.17a, muitas regiões se mesclaram com outras, diminuindo a sobre segmentação e com isso, eliminado futuras falhas.

4.7 SUMARIZAÇÃO

Com toda a aplicação demonstrada neste capítulo, percebe-se o potencial da aplicação para o ramo da segmentação não-supervisionada, através de uma explicação bem detalhada, porém simples. Além disso, o filtro *Color Boost* também se mostra com grande potencial para ser aplicado em outras vertentes na área de melhoria de imagens. Talvez num futuro próximo, a aplicação do filtro e deste processo possa ser extrapolada para outros segmentadores.

Por hora, basta aplicar o método ao banco de imagens adquirido e verificar se os resultados realmente condizem com o que é esperado.

5 RESULTADOS OBTIDOS

Nesta última parte do trabalho apresentam-se os resultados obtidos usando o método construído e demonstrado no capítulo anterior. Aqui também serão apresentadas algumas das possíveis limitações que o método encontra e como o ambiente experimental foi construído, além de uma avaliação analítica do resultado final.

5.1 Arquitetura Usada

Antes de apresentar os resultados, é importante conhecer a plataforma de hardware e de software aonde foram concebidos os resultados.

Foi preparado um ambiente virtualizado baseada no VMWARE Workstation v12. Usando o Ubuntu 15.10, 64 bits, com 4GB de RAM, 20GB de HD em modo SSD e um processador Core i7 3820 de 3.6GHz. O algoritmo do segmentador foi construído usando o C++, em conjunto com bibliotecas da distribuição OpenCV v.3.1 e IDE Eclipse CDT Mars.2.

Todas as imagens estão em formato PNG/JPG e em resolução 720x480px usado de forma arbitraria todos os indivíduos (gatos) capturados no banco de imagens.

5.2 Resultado Final

Nessa etapa, e com o banco de dados possuindo 1020 imagens, foram selecionadas 20 imagens que serão mostradas aqui suas segmentações finais (variando entre ambientes *indoor* e *outdoor*), mais 4 imagens finais que serão colocadas de outro *dataset* obtido da Stanford Visual Geometry Group (http://www.robots.ox.ac.uk/~vgg/data/pets/), todas estas imagens em *outdoor*. O

resultado compara na Tabela 5.1 quatro técnicas diferentes: a *Watershed* Manual (fornecida pelo pacote do OpenCV), o método de Malpica *et al.* (1997) para o *Watershed* automático, como apenas a inclusão da regra de redução sobre segmentação (apenas para melhorar a comparação da parte da regra de redução na sobre segmentação, não afetando o método em si), o método de Jahangiri e Heesch (2009) para o *GrabCut* automático e, por fim, o método proposto pelo autor e descrito no Capítulo quatro. Ressaltando ainda que o filtro *Color Boost* será aplicado somente no método proposto, por ser um processo desenvolvido especificamente para este trabalho.

Na Tabela 5.1, pode-se observar que, no geral, o resultado obtido pelo segmentador proposto é mais eficiente, por ser menos variante e obter na maior parte dos casos uma segmentação melhor. Contudo, nota-se que em alguns casos o método apresenta desempenho inferior. Tais situações serão analisadas mais à frente neste capítulo.

Para confirmar a eficiência do método, o mesmo procedimento proposto por Ning *et al.* (2010) é usado e apresentado na Tabela 5.2, que mostra os resultados de forma analítica usando os coeficientes de *True Postive Rate* (TPR) e *False Postive Rate* (FPR) para cada método e imagem utilizados nos experimentos. A imagem segmentada pelos métodos comparados tem suas áreas de segmentação rotuladas manualmente, de forma que essas áreas possuam a maior concentração de pixels do objeto em questão e que se deseja segmentar (de forma subjetiva). O TPR é definido por uma taxa do número de pixels que coincidem corretamente sobre o objeto a ser rotulado como *foreground*, e o FPR consiste por uma taxa do número de pixels que não coincidem com o *foreground,* mas acabam fazendo parte da área segmentada, e por isso são considerados parte do *background* que foi extraído em conjunto com o *foreground.*

Para realizar essa comparação, *ground truth* (GT) de cada *foreground* testado foi retirado manualmente com ajuda do Adobe Photoshop CC 2015. A Figura 5.1 exibe alguns exemplos de GT's retirados manualmente.

Tabela 5.1 – Resultado final da segmentação do método proposto comparado com outros trabalhos. A 1ª coluna tem a identificação da imagem usada; a 2ª coluna tem a imagem original sem o filtro *Color Boost*; a 3ª coluna mostra a transformada *Watershed* com as marcas manualmente inseridas; a 4ª coluna mostra a *Watershed* usada por Malpica *et al.*; a 5ª coluna mostra o método *GrabCut* usado por Jahangiri e Heesch e; a 6ª coluna mostra o resultado alcançado pelo método descrito neste trabalho.

(Continua)

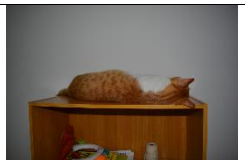
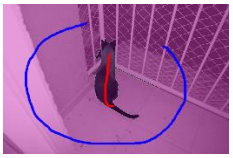
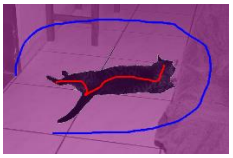
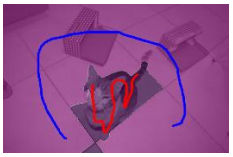
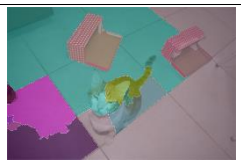
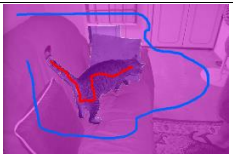
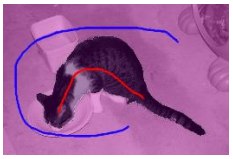
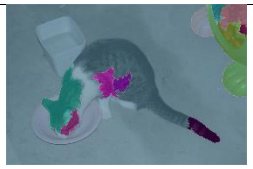
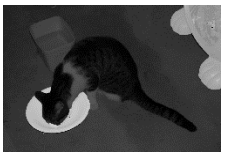
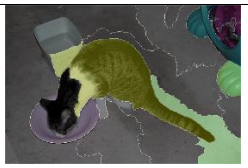
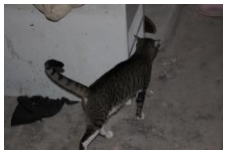
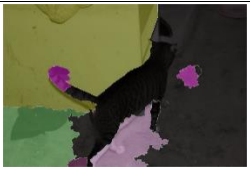
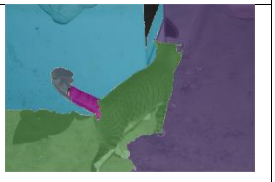
Image ID	*Imagem Original*	*Watershed*	*Malpica et al.*	*Jahangiri e Heesch*	*Método Proposto*
1					
2					
3					
4					
5					
6					
7					
8					

Tabela 5.1 – Resultado final da segmentação do método proposto comparado com outros trabalhos. A 1ª coluna tem a identificação da imagem usada; a 2ª coluna tem a imagem original sem o filtro *Color Boost*; a 3ª coluna mostra a transformada *Watershed* com as marcas manualmente inseridas; a 4ª coluna mostra a *Watershed* usada por Malpica *et al.*; a 5ª coluna mostra o método *GrabCut* usado por Jahangiri e Heesch e; a 6ª coluna mostra o resultado alcançado pelo método descrito neste trabalho.

(Continuação)

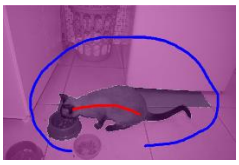
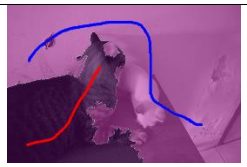
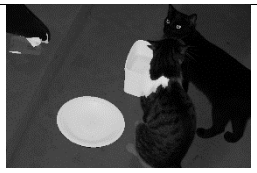
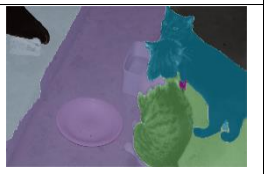
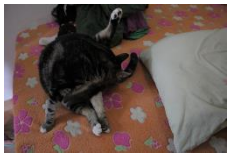
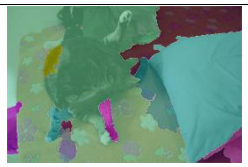
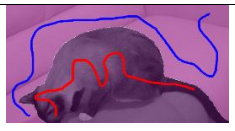
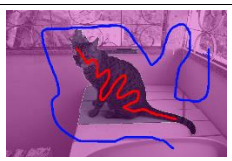
Image ID	*Imagem Original*	*Watershed*	*Malpica et al.*	*Jahangiri e Heesch*	*Método Proposto*
9					
10					
11					
12					
13					
14					
15					
16					

Tabela 5.1 – Resultado final da segmentação do método proposto comparado com outros trabalhos. A 1ª coluna tem a identificação da imagem usada; a 2ª coluna tem a imagem original sem o filtro *Color Boost*; a 3ª coluna mostra a transformada *Watershed* com as marcas manualmente inseridas; a 4ª coluna mostra a *Watershed* usada por Malpica *et al.*; a 5ª coluna mostra o método *GrabCut* usado por Jahangiri e Heesch e; a 6ª coluna mostra o resultado alcançado pelo método descrito neste trabalho.

(Continuação)

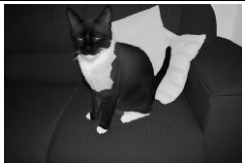
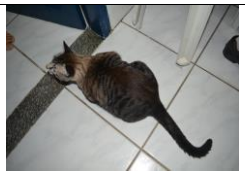
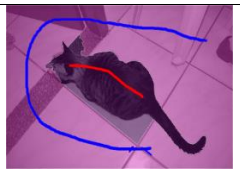
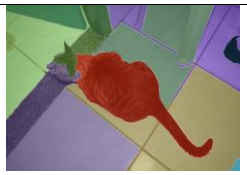
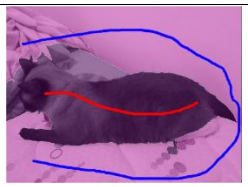
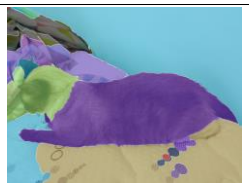
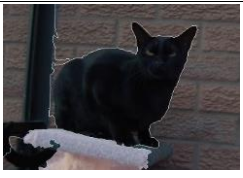
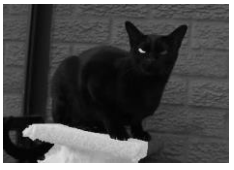
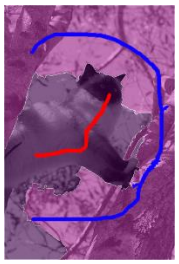
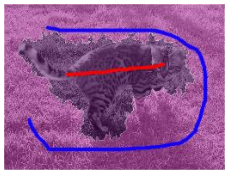
Image ID	*Imagem Original*	*Watershed*	*Malpica et al.*	*Jahangiri e Heesch*	*Método Proposto*
17					
18					
19					
20					
21					
22					

Tabela 5.1 – Resultado final da segmentação do método proposto comparado com outros trabalhos. A 1ª coluna tem a identificação da imagem usada; a 2ª coluna tem a imagem original sem o filtro *Color Boost*; a 3ª coluna mostra a transformada *Watershed* com as marcas manualmente inseridas; a 4ª coluna mostra a *Watershed* usada por Malpica *et al.*; a 5ª coluna mostra o método *GrabCut* usado por Jahangiri e Heesch e; a 6ª coluna mostra o resultado alcançado pelo método descrito neste trabalho.

(Conclusão)

Image ID	*Imagem Original*	*Watershed*	*Malpica et al.*	*Jahangiri e Heesch*	*Método Proposto*
23					
24					

Fonte: Próprio Autor, 2016.

No caso de métodos do tipo de região (*Watershed*), as áreas rotuladas manualmente foram retiradas com o critério de maior concentração do *foreground*. Já o método do tipo *GrabCut*, que somente retorna o *foreground* (o *background* é descartado automaticamente), apresenta-o como a parte mais clara da imagem (vide Tabela 5.1) e será usado na comparação analítica.

Figura 5.1 – Exemplos de *ground truth* retirado manualmente.

Fonte: Próprio Autor, 2016.

Na Tabela 5.2, maiores valores de TPR e menores valores de FPR indicam o método de melhor desempenho (dados em porcentagem). Obviamente, o método manual é o melhor, mas em comparação com os métodos não-supervisionados testados, o método proposto é o que apresenta menor desvio padrão (cerca de 10.75), e obtém uma média muito maior (cerca de 84.83%), em termos do TPR, tomando todos os resultados mostrados. Apesar que em alguns casos, como p.e. as imagens 14, 15 e 16 (vide Tabela 5.1), o método de Jahangiri e Heesch e o de Malpica *et al.* possam superar o método proposto, o mesmo sobressai-se na maioria dos casos. O método de Jahangiri e Heesch sempre se sobressai em imagens que tem o *background* bem homogêneo em cor, como na imagem 15, da Tabela 5.1. Já o método de Malpica *et al.* se sobressai em imagens onde o objeto em destaque tem contornos muito bem definidos, como no caso dos gatos das imagens 14 e 16 (Tabela 5.1), que são bem "escuros" e tem bordas bem destacadas em relação ao plano de fundo.

Tabela 5.2 – Comparação analítica pelos coeficientes TPR e FPR entre os métodos testados.

(Continua)

Imagem ID	*Método*	*TPR%*	*FPR%*
1	*Watershed* Original	81.55	3.17
	Malpica *et al.*	40.4	7.3
	Jahangiri e Heesch	66.4	22.17
	Proposto	95.72	1.48
2	*Watershed* Original	95.67	0.62
	Malpica *et al.*	90.91	1.22
	Jahangiri e Heesch	0.00	4.67
	Proposto	91.08	0.58
3	*Watershed* Original	99.74	1.02
	Malpica *et al.*	89.45	1.43
	Jahangiri e Heesch	0.09	13.64
	Proposto	80.42	1.44

Tabela 5.2 – Comparação analítica pelos coeficientes TPR e FPR entre os métodos testados.

(Continuação)

Imagem ID	*Método*	*TPR%*	*FPR%*
4	*Watershed* Original	96.69	0.67
	Malpica *et al.*	96.15	1.21
	Jahangiri e Heesch	95.59	7.64
	Proposto	91.81	0.95
5	*Watershed* Original	82.39	5.1
	Malpica *et al.*	28.16	6.67
	Jahangiri e Heesch	59.85	12.03
	Proposto	74.62	3.44
6	*Watershed* Original	90.19	5.81
	Malpica *et al.*	76.57	49.66
	Jahangiri e Heesch	87.97	41.26
	Proposto	75.63	2.49
7	*Watershed* Original	91.85	2.16
	Malpica *et al.*	67.12	98.65
	Jahangiri e Heesch	1.25	37.7
	Proposto	73.12	6.87
8	*Watershed* Original	86.53	13.61
	Malpica *et al.*	82.47	45.89
	Jahangiri e Heesch	0.00	18.85
	Proposto	83.86	20.84
9	*Watershed* Original	95.69	6.21
	Malpica *et al.*	63.21	37.18
	Jahangiri e Heesch	14.27	19.38
	Proposto	81.17	2.15
10	*Watershed* Original	86.76	4.46
	Malpica *et al.*	51.22	7.01
	Jahangiri e Heesch	35.37	16.94
	Proposto	97.59	6.16

Tabela 5.2 – Comparação analítica pelos coeficientes TPR e FPR entre os métodos testados.

(Continuação)

Imagem ID	*Método*	*TPR%*	*FPR%*
11	*Watershed* Original	86.55	10.85
	Malpica *et al.*	81.15	98.12
	Jahangiri e Heesch	1.37	46.15
	Proposto	91.78	22.42
12	*Watershed* Original	92.93	22.24
	Malpica *et al.*	67.05	87.15
	Jahangiri e Heesch	3.73	53.11
	Proposto	57.33	17.82
13	*Watershed* Original	99.46	21.18
	Malpica *et al.*	82.5	53.91
	Jahangiri e Heesch	0.06	32.38
	Proposto	98.87	39.86
14	*Watershed* Original	99.01	0.71
	Malpica *et al.*	93.72	0.92
	Jahangiri e Heesch	0.00	5.39
	Proposto	99.86	5.63
15	*Watershed* Original	99.76	19.36
	Malpica *et al.*	94.25	95.78
	Jahangiri e Heesch	99.35	3.39
	Proposto	84.65	12.92
16	*Watershed* Original	96.66	4.92
	Malpica *et al.*	89.15	9.93
	Jahangiri e Heesch	1.55	38.51
	Proposto	86.46	4.92
17	*Watershed* Original	94.35	1.18
	Malpica *et al.*	75.74	28.42
	Jahangiri e Heesch	21.61	31.41
	Proposto	88.23	17.73

Tabela 5.2 – Comparação analítica pelos coeficientes TPR e FPR entre os métodos testados.

(Continuação)

Imagem ID	*Método*	*TPR%*	*FPR%*
18	*Watershed* Original	97.62	4.35
	Malpica *et al.*	83.97	55.57
	Jahangiri e Heesch	47.91	20.17
	Proposto	87.01	2.65
19	*Watershed* Original	41.85	2.61
	Malpica *et al.*	28.36	19.7
	Jahangiri e Heesch	0.02	10.81
	Proposto	61.4	3.01
20	*Watershed* Original	96.74	11.16
	Malpica *et al.*	87.08	92.15
	Jahangiri e Heesch	17.51	50.86
	Proposto	79.5	18.9
21	*Watershed* Original	72.54	5.41
	Malpica *et al.*	88.12	8.11
	Jahangiri e Heesch	4.95	0.12
	Proposto	92.52	5.64
22	*Watershed* Original	85.12	2.78
	Malpica *et al.*	86.42	15.96
	Jahangiri e Heesch	3.56	14.54
	Proposto	86.44	2.11
23	*Watershed* Original	99.87	20.81
	Malpica *et al.*	69.89	62.08
	Jahangiri e Heesch	0.02	12.47
	Proposto	89.97	23.44
24	*Watershed* Original	91.52	18.55
	Malpica *et al.*	80.64	55.62
	Jahangiri e Heesch	14.89	0.45
	Proposto	85.79	7.8

Tabela 5.2 – Comparação analítica pelos coeficientes TPR e FPR entre os métodos testados.

(Conclusão)

	Método	*TPR%*	*FPR%*
Média	*Watershed* Original	90.04	7.87
	Malpica *et al.*	74.74	39.15
	Jahangiri e Heesch	24.06	21.42
	Proposto	**84.83**	**9.64**
Desvio Padrão	*Watershed* Original	12.44	7.40
	Malpica *et al.*	19.74	35.2
	Jahangiri e Heesch	33.4	16.18
	Proposto	**10.75**	**9.97**

Fonte: Próprio Autor, 2016.

Deve-se mencionar que as imagens 2, 7, 8, 12, 14 e de 21 a 24 (vide Tab. 5.1), são todas em ambientes *outdoor* com mudanças significativas de iluminação. Sendo as últimas quatro (21 a 24) obtidas de *dataset* externo. Verifica-se que o método proposto funciona em outras imagens, validando ainda mais o método.

Percebe-se também pela Tabela 5.2 na parte de média e desvio padrão, que o método proposto, além de ser menos variante, possui os melhores resultados em cenários heterogêneos (onde os métodos de Malpica *et al.* e Jahangiri e Heesch falharam).

Contudo o método proposto apresenta limitações em algumas circunstâncias em que o seu desempenho mostra-se claramente fora das expectativas. Cabe ressaltar que isso é esperado, visto que os métodos não-supervisionados são projetados para trabalharem dentro de suas condições de operação (HE, KIM e KUO; JAHANGIRI e HEESCH, 2014; 2009). Na próxima seção é visto algumas das limitações encontradas pelo método proposto.

5.3 Limitações Encontradas

Nesta seção, o método proposto (assim como os outros métodos testados) encontrou dificuldades na segmentação em situações específicas. Entre essas limitações, o autor busca discorrer a melhor compreensão do porque o método não foi conclusivo nestes casos apresentados.

Serão mostrados três exemplos comuns que foram encontrados usando o banco de imagens coletado. Lembrando que a limitação dos resultados que obtiveram problemas na sobre segmentação e sub segmentação não serão explanados aqui, visto que esta é uma característica intrínseca da segmentação não supervisionada (FORSYTH e PONCE; NING *et al.*, 2012; 2010).

Inicialmente, imagens com padrões muito repetidos se impõem como uma grande limitação. Quando o filtro *Color Boost* é aplicado, o contraste é intensificado nos objetos de maior destaque na cena. O problema aqui é que as formas repetidas dão mais ênfase nas cores do que no gato em si, fazendo delas o "objeto" de maior destaque. Além disso, o *Watershed* busca segmentar melhor objetos com formas bem definidas, por isso, o padrão repetido em lençóis, toalhas, panos, etc. são melhores segmentados por esse tipo de segmentador. A Figura 5.2a, b e c demonstra um pouco dessa limitação encontrada.

Em segundo caso, a limitação imposta por dois ou mais gatos na cena. Neste caso, a filtragem homomórfica irá criar o efeito de brilho focado no objeto com maior contraste. A diferença aqui é que quando uma cena possui dois gatos, por exemplo, o algoritmo não tem "inteligência" necessária para perceber isso, e por isso o efeito de brilho é criado em apenas um gato – fazendo do outro gato parte do *background* durante a segmentação. Isso quando os dois gatos estão "separados" na mesma cena. Quando os dois estão bem "juntos", o algoritmo também erra ao dizer quem é quem (que não

faz parte do objetivo aqui) e por vezes, acaba considerando os dois como o mesmo objeto e segmenta-os num *cluster* único. A Figura 5.3a e 5.3b mostram esse caso.

Figura 5.2 – Exemplos da limitação do método proposto à padrões repetidos na cena.

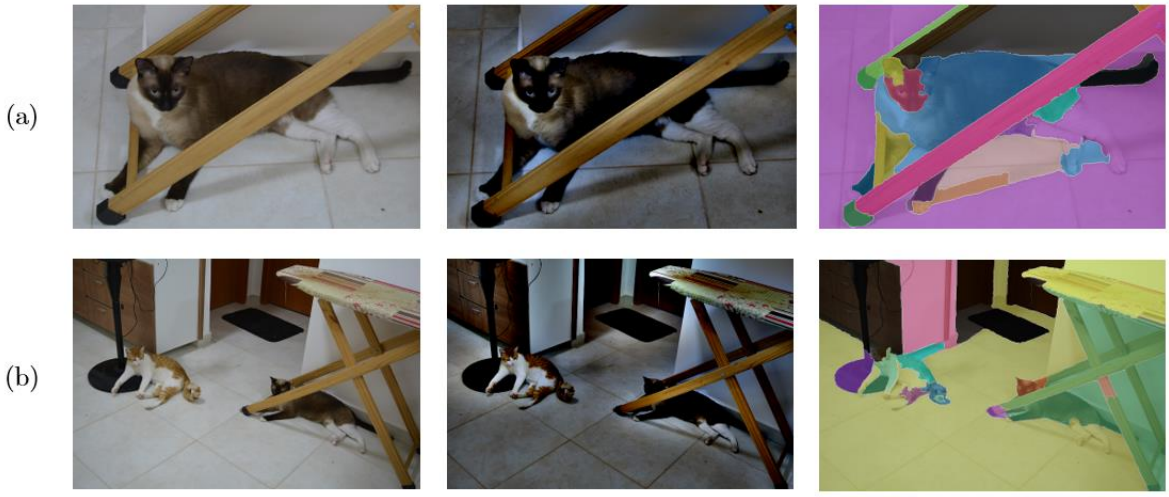
Fonte: Próprio Autor, 2016.

Figura 5.3 – Exemplos da limitação do método proposto à múltiplos gatos na cena.

Fonte: Próprio Autor, 2016.

Por fim, este último caso não é exatamente uma limitação, visto que o método proposto consegue segmentar o gato por "inteiro". O problema consiste na geração de sobre segmentação causada por um outro objeto à frente do gato. A Figura 5.4a e 5.4b exemplificam o caso.

Figura 5.4 – Exemplos da limitação do método proposto devido objetos sobreporem o gato.



Fonte: Próprio Autor, 2016.

Na Fig. 5.4a e 5.4b, o gato consegue ser segmentado sem grandes problemas, mas a ausência de um processo de reconhecimento de padrões faz com que o segmentador "entenda" que cada parte do gato é um segmento completamente diferente, devido principalmente a filtragem homomórfica criar o efeito de brilho ao longo do objeto que sobrepõe o objeto a ser segmentado. Isso afeta até a regra de redução da sobre segmentação, que não ajuda neste caso, devido ao fato que várias partes do animal são diferentes em termos de textura e cor. Lembre-se que a regra compara os histogramas dos canais H e S de cada segmento e tenta mescla-los. O problema é que um outro objeto passa por cima dessa possível vizinhança, confundido a regra e não mesclando o objeto como um todo. Esse caso poderia ser resolvido por um extrator de características como o SURF p.e., para servir como um processo secundário da regra de redução da sobre segmentação e mesclar os resultados através da afinidade de características e não somente de histogramas.

DISCUSSÃO FINAL E PROJETOS FUTUROS

Nesta última parte, será discutido quais são as conclusões que se pôde obter a partir dos resultados vistos no capítulo anterior e quais as futuras pretensões para o projeto em si.

Os resultados obtidos provaram que o método proposto consegue atingir uma confiabilidade grande em termos de segmentar os gatos/objetos, mesmo em comparação aos trabalhos de Malpica *et al.* e de Jahangiri e Heesch (que são muito bons), pois fazem bastante diferença em cenários aonde o *background* é mais uniforme e seus artigos são bem ricos e detalhados – além de serem bastante citados em outros trabalhos. A escolha desses dois métodos não foi ao acaso, mas sim para testar métodos que são estruturais para construção de outros, como os dos trabalhos de Khattab *et al.* (2014) e de Park *et al.* (2009), por exemplo. Estes dois últimos não foram utilizados por apenas uma razão: falta de tempo e espaço dentro deste trabalho para comparar todos os métodos possíveis. Lembre-se que este trabalho não propõe criar um novo segmentador, mas sim buscar um método mais eficiente de prover o início de uma segmentação, podendo ser combinado com qualquer método supervisionado, semi e/ou não-supervisionado.

Sobre os métodos de Khattab *et al.* e Park *et al.*, apesar de não terem sidos usados na comparação dos resultados, os mesmos não são desnecessários, porém ambos possuem o mesmo comportamento em relação aos dois métodos estudados e comparados. Estatisticamente, o resultado seria bem próximo do já obtido no capitulo 5.

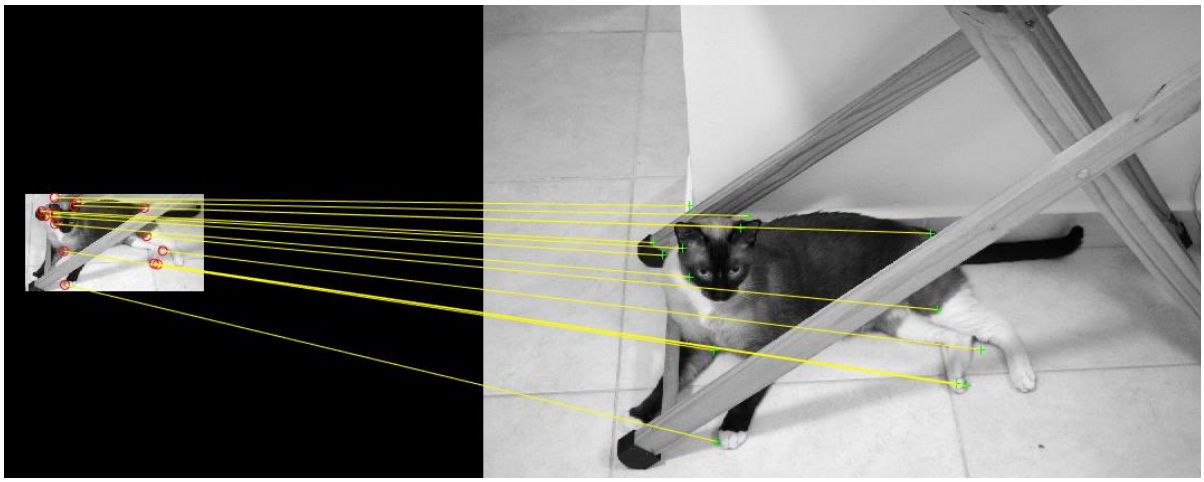
Com base nisso, observa-se também que os objetivos específicos foram obtidos gradativamente ao longo do trabalho (incluindo um adicional da inclusão da regra de redução da sobre segmentação melhorada de Ning *et al.* (2010)), provendo também de

forma bem detalhada, os processos de melhoria de imagem e de pré-segmentação do algoritmo. Em suma, o crédito aqui fica para a abordagem do filtro *Color Boost* e da filtragem homomórfica usando a DCT, que provem a parte mais forte desta etapa.

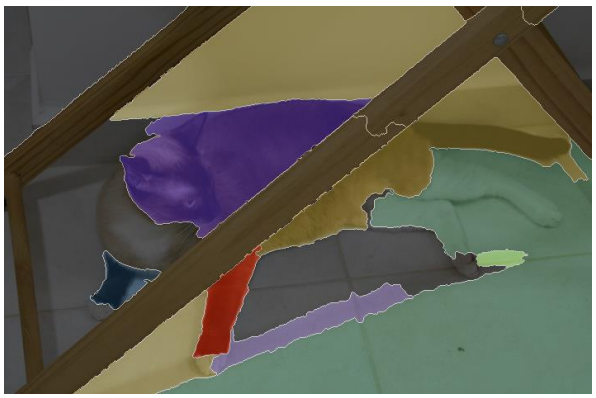
Já com relação a segmentação, a combinação da etapa de pré-segmentação mais o *Watershed* automático, casou perfeitamente num método bem eficiente e rápido. Tão rápido, que mesmo o processo original do *Watershed* ainda é mais lento (e em algumas pequenas vezes, também perde no coeficiente TPR).

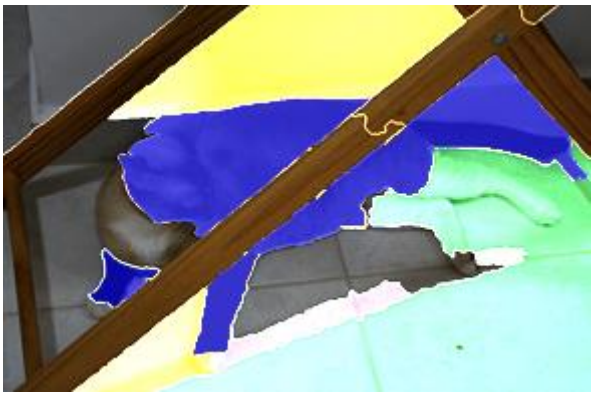
Obstante a tudo isso, percebe-se também que o método pode ser melhorado e estudado para fornecer um método de segmentação universal e com certa inteligência. Neste sentido, projetos futuros para este trabalho podem promover a criação de novos segmentadores que recombinem essas informações de forma a se adaptar a região da imagem. Neste caso, poder-se-ia utilizar características retiradas por algum extrator (como o SURF, p.e.) para permitir que uma nova regra de redução de sobre segmentação seja criada e comparada com a regra atual, provendo assim um mescla das partes do objeto que podem ser recombinadas, sem serem exatamente similares (em termos do histograma). Um exemplo seria a Figura C.1 que tentaria prover esta mescla através de comparação com *features* extraídos de parte do animal e verificados se fazem parte do mesmo objeto a ser segmentado. Os *features* extraídos na Fig. C.1a seriam utilizados como marcas para uma nova segmentação, só que todos fariam parte do possível *foreground*. Após essa segmentação, a nova regra combinaria as áreas segmentadas pela similaridade do objeto e não apenas do histograma.

Figura C.1 – Exemplo da recombinação pela nova regra de redução da sobre segmentação proposta no futuro. (a) extraídos os *features*; (b) segmentação pelo método proposto; (c) possível recombinação com os *features* extraídos e o método proposto.



(a)





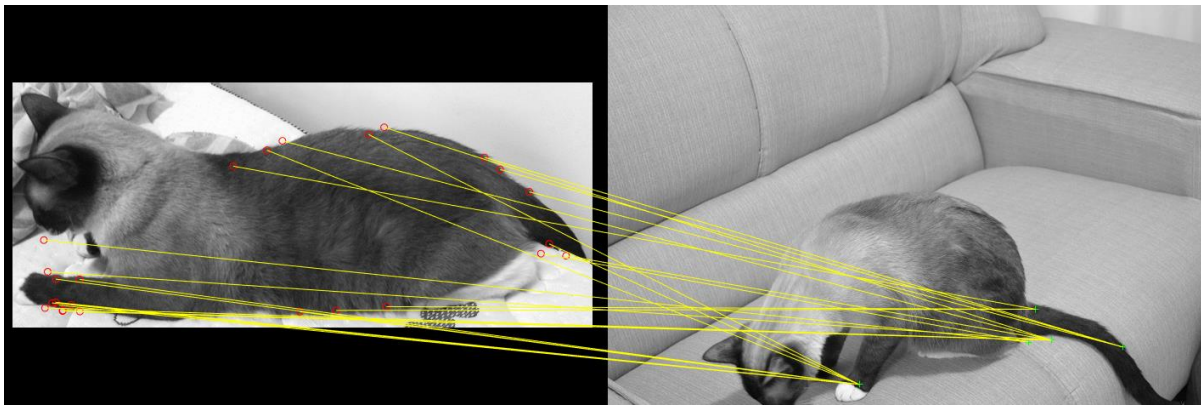
(b) (c)

Fonte: Próprio Autor, 2016.

Outra possível vertente seria utilizar algo bem parecido com o método de Jahangiri e Heesch, para criar o contorno da segmentação inicial sobre o animal, a partir da localização/posição do gato na imagem. Isso pode ser obtido com o SURF ou o DAISY facilmente. Nesse caso bastaria extrair as características de cada cena e tentar, através de homografia e comparação, localizar o objeto numa cena *clusterizada* (que tem muitos outros objetos) através de *Point Feature Matching* (ou casamento de características pontuais). Logo após isso, indicar a separação do objeto a ser

segmentado da maior parte da cena em si, para só depois, iniciar o método proposto por este trabalho. Na Tabela 5.1, a imagem 15 teve esse experimento de forma bem rudimentar, pois, a separação foi manual. Na Figura C.2 pode ser visto como foi o processo de detecção de características e localização do objeto/animal da cena.

Figura C.2 – Exemplo do reconhecimento de objetos usando o SURF. (a) comparação de características, incluindo *features* externos ao objeto; (b) criação de um retângulo que indica a localização do objeto na cena que se deseja segmentar; (c) processo de corte feito manualmente com base na resposta dada em (b).



(a)

(b)

(c)

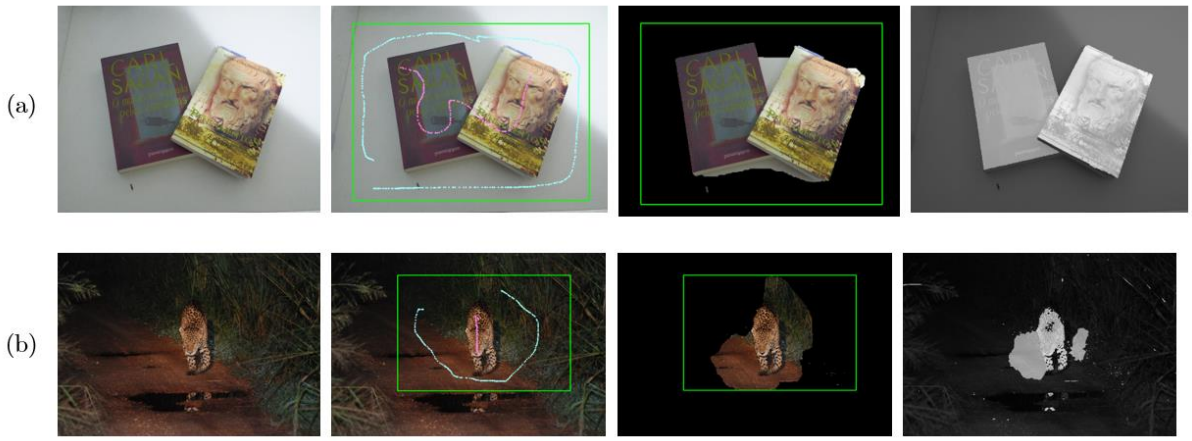
Fonte: Próprio Autor, 2016.

Observa-se que na Fig. C.2b, um retângulo foi desenhado em torno do objeto a ser segmentado. Pode-se, então, criar uma aplicação para segmentar apenas a área que está dentro do retângulo criado e devolver o objeto a qual se queira segmentar de forma mais eficiente e rápida, visto na Fig. C.2c. Apesar da imagem 15 da Tabela 5.1 ter sido

obtida dessa forma, não foi incluído esse passo no método porque carece de muitas modificações e o processo ainda está imperfeito e só funciona (até o presente momento) em poucas imagens dentro do banco. O melhor seria usar um processo combinado com um *One Class* SVM (OC-SVM) para criar primeiramente um dicionário de características que pudessem separar a cena em apenas uma classe: a área que tem gato ou não. Ou seja, o que é *foreground* ou não. Percebe-se que esse procedimento ainda pode prover bastantes ideias de melhoria.

Além disso, o autor busca também prover esse método de forma que possa ser reconfigurado pelo usuário. Neste possível trabalho futuro, o algoritmo poderia juntar as ideias anteriores e prover um processo de segmentação universal. A Figura C.3 mostra como o algoritmo pode segmentar algo quando está adaptado para outros objetos, neste caso, livros (Fig. C.3a) e outros animais (Fig. C.3b). Aqui o resultado foi obtido por uma tentativa de combinar o *GrabCut* do método de Jahangiri e Heesch com a primeira macro etapa do método proposto (vide Cap. 4).

Figura C.3 – Exemplo da reconfiguração do método proposto. A esquerda as imagens originais, no centro, a segmentação e o resultado feito de forma manual com o *GrabCut* e a direita o resultado combinando o método proposto com o de Jahangiri e Heesch (a) segmentação de objetos genéricos; (b) segmentação em onças.



Fonte: Próprio Autor, 2016.

Na Fig. C.3, ambos os objetos que se busca destacar estão imersos em *backgrounds* bem uniformes, por isso a escolha pelo *GrabCut* geraria melhores resultados. Talvez no futuro, o algoritmo possa também identificar a homogeneidade do *background* a ponto de escolher o segmentador a ser usado para obter melhores resultados. Fica claro que o método tem a possibilidade, se adaptado corretamente, para segmentar vários tipos de objetos (não todos, visto uma série de limitações), desde que seja melhorado no futuro.

Com base em tudo isso, o método de segmentação mostrado neste trabalho poderá auxiliar os profissionais da área que buscam processos mais rápidos para diminuir os erros e economizar processamento computacional a ponto de convergir em novos tipos de segmentadores capazes colaborar na classificação de outros objetos. Isso poderá prover o passo inicial para a construção de programas comerciais capazes de ajudar monitores, a separar e classificar objetos e animais de forma automática. Diminuindo o tempo de catalogação e aumentando a eficiência da análise objetiva e parcimoniosa. Além de contribuir para encontrar resultados mais eficazes e capazes de ajudar animais e profissionais na área da monitoração da fauna e da flora em especial.

Verifica-se que apesar de todos os problemas, a área da segmentação não-supervisionada provê um espaço bem amplo para a melhoria e capacidade matemática de tentar inferir (e modelar) como os métodos de segmentação podem ocorrer no mundo real.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 14724**: Informação e documentação - Trabalhos Acadêmicos - Apresentação. Rio de Janeiro, 2011.

BEUCHER, Serge. **The Watershed Transformation Applied to Image Segmentation**. 10th Pfefferkorn Conf. on Signal and Image Processing in Microscopy and Microanalysis, 16-19 sept. 1991, Cambridge, UK, Scanning Microscopy International, suppl. 6. 1992, pp. 299-314.

_____. **The Watershed for Image Segmentation and Mathematical Morphology**. 2010. Disponível em: <http://cmm.ensmp.fr/~beucher/wtshed.html>. Último acesso em: 15 fevereiro 2016.

BOYKOV, Y; JOLLY, M. **Interactive Graph-Cuts for Optimal Boundary and Region Segmentation of Objects in nd images**. In Proceeding of 8th IEEE International Conference on Computer Vision, ICCV, IEEE, vol 1, 2001, pp. 105–112.

COMANICIU, D.; MEER, P. **Mean Shift Analysis and Applications**. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Vol. 2. pp. 1197–1203 vol.2, 1999.

_____. **Mean Shift: A Robust Approach Toward Feature Space Analysis**. Proc. IEEE transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, 2002.

COUPRIE, Camille; GRADY, Leo; NAJMAN, Laurent; TALBOT, Hughes. **Power Watersheds: A New Image Segmentation Framework Extending Graph Cuts, Random Walker and Optimal Spanning Forest**. [s.l.]: [s.ed.], 2011.

CRALL, Jonathan P.; STEWARD, Charles V.; BERGER-WOLF, Tanya Y.; RUBENSTEIN, Daniel I.; SUNDARESAN, Siva R. **HotSpotter - Patterned Species Instance Recognition**. [s.l.]: IEEE Conference on Computer Vision and Pattern Recognition, 2013.

DINIZ, Paulo Sergio R. *et al.* **Processamento Digital de Sinais**. 2.ed. Porto Alegre: Bookman, 2014.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern Classification**. 2.ed. New York: Wiley, 2001.

DUBUISSON, Séverine. **The Computation of the Bhattacharyya Distance Between Histograms Without Histograms**. 2nd International Conference on Image Processing Theory, Tools and Applications, IPTA, 2010, pp. 373-378.

FORSYTH, David A.; PONCE, Jean. **Computer Vision:** A Modern Approach. 2.ed. New Jersey: PEARSON, 2012.

FRANK, *et al.* **Experimental Evidence that Feral Cats Cause Local Extirpation of Small Mammals in Australia's Tropical Savannas**. Journal of Applied Ecology, vol. 51, British Ecological Society, 2014. pp. 1486-1493.

FUKUNAGA, K.; HOSTETLER, L. D. **The Estimation of the Gradient of a Density Function**. In pattern recognition. IEEE Transactions on Information Theory vol. 21, pp. 32–40, 1975.

FURASTÉ, Pedro Augusto. **Normas Técnicas para o Trabalho Científico:** Explicação das Normas da ABNT. 17.ed. Porto Alegre: Dáctilo Plus, 2013.

G1. **Manaus tem mais de 200 mil Animais Abandonados**. 2015. Disponível em: <http://goo.gl/g4mLqB>. Último acesso em: 14 fevereiro 2016.

GASTAL, Eduardo S. L.; OLIVEIRA, Manuel M. **Domain Transform for Edge-Aware Image and Video Processing**. In proceeding (SIGGRAPH), ACM Transactions on Graphics, vol. 30, issue 4, 2011.

GONZALEZ, Rafael C.; WOODS, Richard C. **Processamento Digital de Imagens**. 3.ed. São Paulo: PEARSON, 2010.

GRADY, Leo. **Multilabel Random Walker Image Segmentation Using Prior Models**. [s.l.]: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, p. 763–770, 2005.

_____. **Random Walks for Image Segmentation**. [s.l.]: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 11, 2006.

HAIDAR, Silvia. **Saiba Quem São os Gatos Ferais que a Austrália quer Exterminar**. Folha de São Paulo, 2015. Disponível em: <http://goo.gl/FFguhO>. Último acesso em: 16 fevereiro 2016.

HE, Jia; KIM, Chang-Su; KUO, C.-C Jay. **Interactive Segmentation Techniques: Algorithms and Performance Evaluation**. New York, NY: Springer, 2014.

INSTITUTO ONÇA-PINTADA (IOP). **A Onça-Pintada.** 2015**.** Disponível em: <http://www.jaguar.org.br/iop/pt/a-onca-pintada/>. Último acesso em: 14/02/2015.

IQBAL, K.; SALAM, R. A.; OSMAN, A.; TALIB, A. Z. **Underwater Image Enhancement Using an Integrated Colour Model**. Penang, Malaysia: IEEE International Journal of Computer Science, 2007.

JAHANGIRI, Mohammad; HEESCH, Daniel. **Modified Grabcut for Unsupervised Object Segmentation**. [s.ed], Image Processing ICVPR, 2009.

JUSBRASIL. **Brasil tem 30 Milhões de Animais Abandonados**. 2014. Disponível em: <http://goo.gl/0bXfdL>. Último acesso em: 14 fevereiro 2016.

KHATTAB, Dina; EBIED, Hala Mousher; HUSSEIN, Ashraf Saad; TOLBA, Mohamed Fahmy. **Color Image Segmentation Based on Different Color Space Models Using Automatic GrabCut**. The Scientfic World Journal, vol. 2014, Hindawi Publishing, 2014.

KIM, Jong-Bae; KIM, Hang-Joon. **Multiresolution-based Watersheds for Efficient Image Segmentation**. Pattern Recognition Letters, vol. 24, issue 1-3, 2003, pp. 473-488.

KIM, Kyungnam; CHALIDABHONGSE, Thanarat H.; HARWOOD, David; DAVIS, Larry. **Real-time Foreground–Background Segmentation Using Codebook Model**. [s.l.]: Real-Time Imaging Conference: ELSEVIER, 2005.

KOLLER, D.; FRIEDMAN, N. **Probabilistic Graphical Models: Principles and Techniques**. Cambridge, MIT Press, 2009.

LINKANIMAL. **Mais de dois milhões de gatos serão mortos na Austrália.** 2015. Disponível em: < http://goo.gl/iuox3N>. Último acesso em: 14 fevereiro 2016.

LOSS, Scott R.; WILL, Tom; MARRA, Peter P. **The Impact of Free-ranging Domestic Cats on Wildlife of the United States**. In proceedings of Nature Communications, vol.4, [s.ed], 2013.

MALPICA, Norberto; SOLÓRZANO, Carlos O. de; VAQUERO, Juan J.; SANTOS, Andrés; VALLCORBA, Isabel; GARCÍA-SAGREDO, José M.; POZO, Francisco del, **Applying Watershed Algorithms to the Segmentation of Clustered Nuclei**. Cytometry Journal, vol. 28, issue 4, 1997, pp. 289-297.

MERRITT, Pamela. **What Breeds are in Our Cat? Look at the Ears**. 2009. Disponível em: <http://www.wayofcats.com/blog/whats-in-my-cat-ears/4974>. Último acesso em: 19 novembro 2015.

NEHAVERMA, Deepak Sharma. **Advanced Maximal Similarity Based Region Merging by User Interactions.** International Journal of Engineering Research and Applications (IJERA), vol. 3, issue 4, 2013, pp. 681-688.

NING, Jifeng; ZHANG, Lei; ZHANG, David; WU, Chengke. **Interactive Image Segmentation by Maximal Similarity-based Region Merging**. [s.l.]: Pattern Recognition – ELSEVIER, 2010.

NISE, Norman S. **Control Systems Engineering**. 7.ed. New York: WILEY, 2015.

NIXON, Mark S.; AGUADO, Alberto S. **Feature Extraction and Image Processing for Computer Vision**. 3.ed. United Kingdom: Elsevier, 2012.

PADMAVATHI, Ganapathi; *et al.* **Comparison of Filters used for Underwater Image Pre-Processing**. Department of Computer Science, Avinashilingam University for Women, Coimbatore, TN, India: [s.ed], 2010. 8p.

PARK, K.T.; LEE, J.H.; MOON, Y.S. **Unsupervised foreground segmentation using background elimination and graph cut techniques**. In proceedings IEEE Electronics Letters, 24th September, vol. 45, no. 20, 2009.

PEDRINI, Hélio; SCHWARTZ, William R. **Análise de Imagens Digitais**: Princípios, Algoritmos e Aplicações. São Paulo: Thomson Learning, 2008.

PRATIKAKIS, I.; VANHAMEL, I.; SAHLI, H.; GATOS, B.; PERANTONIS, S. J. **Unsupervised Watershed-driven Region-based Image Retrieval**. IEEE Proc.-Vis. Image Signal Process., Vol. 153, No. 3, 2006.

PRODANOV, Cleber C.; FREITAS, Ernani C. **Metodologia do Trabalho**: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. 2.ed. Novo Hamburgo: Feevale, 2013.

PUZICHA, Jan; HOFMANN, Thomas; BUHMANN, Joachim M. **Non–parametric Similarity Measures for Unsupervised Texture Segmentation and Image**

Retrieval. In proceedings (CVPR), IEEE computer society conference on computer vision and pattern recognition. 1997.

RODRIGUES, Daily Daleno de O.; FONTOURA, Anderson G.; CARVALHO, José Reginaldo H.; NETO, José P. de Queiroz; VIEIRA, Renato P. **Enhancement of Degraded Images by Natural Phenomena**. In Proceedings of 10th International Conference on Computer Vision Theory and Applications (VISAPP), Berlim, Germany, Elsevier, 2015.

ROTHER, Carsten; KOLMOGOROV, Vladimir; VLAKE, Andrew. **"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts**. In proceedings (SIGGRAPH), ACM Transactions on Graphics, vol. 3, issue 3, 2004, pp. 309.

SCHETTINI, R.; GASPARINI, F.; CORCHS, S.; MARINI, F.; CAPRA, A.; CASTORINA, A. **Contrast Image Correction Method**. Milano, Italy: [s.ed.], 2010.

SOLOMON, Chris; BRECKON, Toby. **Fundamentos de Processamento Digital de Imagens**: uma abordagem prática com exemplos em MATLAB. Rio de Janeiro: LTC, 2013.

STOCKHAM Jr., Thomas G. **Image Processing in the Context of a Visual Model**. In proceedings of the IEEE, vol. 60, issue 7, 1972, pp. 828-842.

THEODORIDIS, Sergios; KOUTROUMBAS, Konstantinos. **Pattern Recognition**. 4.ed. San Diego: ELSEVIER, 2009.

TOMASI, Carlo; MANDUCHI, Roberto. **Bilateral Filtering for Gray and Color Images**. In proceedings of Computer Vision ICPR, 1998. Sixth International Conference on. IEEE, 1998, pp. 839– 846.

VASCONCELOS, Jéssica. **Drama e Riscos: Cerca de 200 mil Cães e Gatos Estão Abandonados pelas Ruas de Manaus**. Á Critica. 2014. Disponível em: <http://goo.gl/Z0t7EY>. Último Acesso em: 14 fevereiro 2016.

VEJA. **Austrália Declara Guerra aos Gatos.** 2015. Disponível em: <http://goo.gl/QZ8TFv>. Último acesso em: 14 fevereiro 2016.

WEEKS, Michael. **Processamento Digital de Sinais Utilizando Matlab e Wavelets**, 2.ed. Rio de Janeiro: LTC, 2012.

YUAN, Eric. **Bilateral Filtering**. Eric Yuan Blog. 2013. Disponível em: <http://eric-yuan.me/bilateral-filtering/>. Último acesso em: 13 março 2016.

APÊNDICE A – Filtro *Color Boost*

Nesse apêndice, será distribuído o código-fonte do filtro *Color Boost* desenvolvido pelo autor e feito no MATLAB®.

A.1 – Código Fonte para o Filtro *Active Color Boost*.

```
function acbf(filename)

%% Função Active Color Boost Filter
%
%  Essa função tem o objetivo de aplicar o filtro Active Color Boost na Imagem
%  Inserida.
%
%  Input: imagem de entrada
%
%  Output: imagem com o active color boost aplicado
%
%  Exemplo: acbf('image3.png');

%% Leitura da imagem
    img = imread(filename);
    %img = imresize(img, 0.5);
    [r, c, s] = size(img);

    imgHSV = rgb2hsv(img);
    H = imgHSV(:,:,1);
    S = imgHSV(:,:,2);
    V = imgHSV(:,:,3);

%% Retira os valores da média e variancia dos canais H e S
    meanH = mean2(H);
    meanS = mean2(S);
    devH = std(H(:));
    devS = std(S(:));
    sigma = 0.63;
    t = abs(20*log(sigma+1));

    %% Aplicação do filtro no canal H
    for i = 1:r
        for j = 1:c
            x = abs((H(i,j)*meanH)/(2*pi*devH^2));
            es = exp((-sigma*pi)/((1-sigma^2)^0.5));
            imgnH(i,j) = 0.5*H(i,j) + 0.5*H(i,j).*((t-x)*es);
        end
    end

    % for i = 1:r
    %     for j = 1:c
    %         x = abs((S(i,j)*meanS)/(2*pi*devS^2));
    %         es = exp((-sigma*pi)/((1-sigma^2)^0.5));
    %         imgnS(i,j) = 0.17*S(i,j) + 0.83*S(i,j).*((t-x)*es);
    %     end
```

```
    % end

    %% Concatenação e melhoria da imagem
    eHSVcommon = cat(3, H, S, imadjust(V));
    eHSVacb = cat(3, imgnH, S, imadjust(V));
    imgRGBcomm = hsv2rgb(eHSVcommon);
    imgRGBacb = hsv2rgb(eHSVacb);
    % eme(imgRGBcomm)
    % eme(imgRGBacb)
    % entropy(imgRGBcomm)
    % entropy(imgRGBacb)

    %% Exibe e grava a imagem com o Color Boost
    figure, imshow(img), figure, imshow(imgRGBcomm), figure, imshow(imgRGBacb)
    imwrite(imgRGBacb,'imgCBF.png');
    %imwrite(imgRGB1,'withoutColorBoost.jpg');
end

%EOF
```

A.2 – Código Fonte para o filtro *Color Boost*

```
function cbf(filename)

%% Função Color Boost Filter
%
%  Essa função tem o objetivo de aplicar o filtro Color Boost na Imagem
%  inserida.
%
%  Input: imagem de entrada
%
%  Output: imagem com o color boost aplicado
%
%  Exemplo: cbf('image3.png');

%% Leitura da imagem
    img = imread(filename);
    %img = imresize(img, 0.5);
    [r, c, s] = size(img);

imgHSV = rgb2hsv(img);
H = imgHSV(:,:,1);
S = imgHSV(:,:,2);
V = imgHSV(:,:,3);

%% Retira os valores da média e variancia dos canais H e S
meanH = mean2(H);
meanS = mean2(S);
sigma = 0.63;
t = abs(20*log(sigma+1));

%% Aplicação do filtro no canal H
for i = 1:r
    for j = 1:c
        x = abs(H(i,j)/meanH);
        es = exp((-sigma*pi)/((1-sigma^2)^0.5));
        imgnH(i,j) = H(i,j).*((t-x).*es);
```

```
    end
end

%% Aplicação do filtro no canal S
for i = 1:r
    for j = 1:c
        x = abs(S(i,j)/meanS);
        es = exp((-sigma*pi)/((1-sigma^2)^0.5));
        imgnS(i,j) = S(i,j).*((t-x).*es);
    end
end

%% Concatenação e melhoria da imagem
enhancedHSV1 = cat(3, imadjust(H), imadjust(S), imadjust(V));
enhancedHSV2 = cat(3, imadjust(imgnH), imadjust(imgnS), imadjust(V));
imgRGB1 = hsv2rgb(enhancedHSV1);
imgRGB2 = hsv2rgb(enhancedHSV2);

%% Exibe e grava a imagem com o Color Boost
figure, imshow(img), figure, imshow(imgnH), figure, imshow(imgRGB1), figure,
imshow(imgRGB2)
imwrite(imgRGB2,'imgColorBoost.png');

end

%EOF
```