



CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS

RODRIGO DA COSTA GYORFI

CONTAGEM DE AVES DE PEQUENO PORTE UTILIZANDO
TÉCNICAS DE VISÃO COMPUTACIONAL



CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS

RODRIGO DA COSTA GYORFI

**CONTAGEM DE AVES DE PEQUENO PORTE UTILIZANDO
TÉCNICAS DE VISÃO COMPUTACIONAL**

Monografia apresentada ao Curso de Ciência da Computação da Faculdade de Ciências Exatas e da Terra do Centro Universitário da Grande Dourados, como pré-requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof.Me Felipe Pereira Perez.

Dourados
2019

Resumo

A contagem de aves na avicultura é um problema presente na comunidade produtora brasileira e ele pode ser resolvido com a ajuda de técnicas de Visão Computacional. No presente projeto é criado um protótipo de um sistema de Visão Computacional para a contagem de aves de pequeno porte no interior de imagens obtidas a partir de dispositivos móveis. O sistema recebe como entrada as imagens, processa essas imagens, detecta e efetua a contagem das aves contidas nas mesmas, atingindo uma margem de erro aceitável.

Palavras-chaves: *Contagem, Segmentação de Imagens, Processamento Digital de Imagens, Visão Computacional, Avicultura.*

Lista de ilustrações

Figura 1 – Exemplo de imagem em escala preto e cinza	32
Figura 2 – Exemplo de imagem binarizada	33
Figura 3 – Exemplo de imagem aplicada a função distanceTransform	34
Figura 4 – Exemplo de imagem binarizada	34
Figura 5 – Resultado detecção das aves	35
Figura 6 – Imagens originais capturadas	36
Figura 7 – Imagens originais capturadas	36
Figura 8 – Interface do sistema	38
Figura 9 – Imagens de resultados obtidos	42
Figura 10 – Imagens de resultados obtidos	42

Lista de tabelas

Tabela 1 – Tabela com resultados dos testes em imagens	40
Tabela 2 – Tabela com resultados dos testes em imagens	41
Tabela 3 – Tabela com resultados dos testes em imagens	41

Lista de siglas

PDI	<i>Processamento Digital de Imagens</i>
MM	<i>Morfologia Matemática</i>
RNA	<i>Redes Neurais Artificiais</i>
PC	<i>Computador Pessoal</i>
DLL	<i>Dynamic Linked Library</i>
IPL	<i>Image Processing Library</i>
USB	<i>Universal Serial Bus</i>

Listas de Códigos

4.1	Função cvtColor	32
4.2	Função threshold	33
4.3	Função distanceTransform	33
4.4	Função normalize	34
4.5	laço para desenho de retângulos nas áreas detectadas	35
4.6	laço para divisão das áreas com mais de uma ave	36
7.1	Código do protótipo	46

Sumário

Lista de Códigos	7
1 INTRODUÇÃO	10
1.1 Objetivo Geral	12
1.1.1 Objetivos Específicos	12
1.2 Justificativa/Motivação	12
1.3 Trabalhos relacionados	13
1.4 Metodologia	14
1.5 Cronograma	16
1.6 Organização do Texto	17
2 PROCESSAMENTO DIGITAL DE IMAGENS	18
2.1 Introdução	18
2.2 Aquisição da imagem	19
2.3 Pré-processamento	19
2.4 Representação em escala de cinza	20
2.5 Operadores morfológicos	20
2.6 Filtros	21
2.6.1 Filtro de Canny	21
2.6.2 Filtro Passa-Baixa e Filtro Passa-Alta	21
2.7 Histogramas	21
2.8 Segmentação de imagens	22
2.8.1 Limiarização (thresholding)	22
2.9 Detecção de bordas	22
2.10 Segmentação orientada a regiões	23
2.10.1 Crescimento de regiões por agregação de pixels	23
2.10.2 Difusão e fusão de regiões	23
2.11 OpenCV	24
2.12 Conclusão	24
3 DETECÇÃO E RASTREAMENTO DE OBJETOS	26
3.1 Introdução	26
3.1.1 Extração de atributos	27
3.2 Técnicas para detecção de objetos	27
3.2.1 Reconhecimento de padrões	28
3.2.2 Redes Neurais Artificiais	28

3.2.3	Métodos exatos	29
3.3	Técnicas para rastreamento de objetos	29
3.3.1	Filtro de Kalman	30
3.3.2	CONDENSATION (Conditional Density Propagation)	30
3.3.3	Corner Detection	30
3.3.4	Subpixel Corners	31
3.4	Conclusão	31
4	DESENVOLVIMENTO	32
4.1	O sistema desenvolvido	32
4.2	Implementações e Imagens	32
4.2.1	Imagens originais	36
4.2.2	Protótipo final	37
4.2.2.1	Botão CÂMERA	37
4.2.2.2	Botão GALERIA	37
4.2.2.3	Interface do sistema	37
5	RESULTADOS	39
5.0.1	Introdução	39
5.0.2	Considerações iniciais	39
5.0.3	Resultados obtidos	39
5.0.4	Imagens de resultados obtidos	42
5.0.5	Considerações finais	43
6	CONCLUSÃO	44
6.0.1	Limitações	44
6.0.2	Contribuições	44
7	DIFICULDADES ENCONTRADAS	45
7.0.1	Trabalhos futuros	45
	Anexos	46
	REFERÊNCIAS BIBLIOGRÁFICAS	54

1 Introdução

O campo de Visão Computacional busca integrar as áreas de PDI (Processamento Digital de Imagens) e IA (Inteligência Artificial). Segundo Neves, Neto e Gonzaga (2012), ela visa a obtenção de algoritmos capazes de interpretar o conteúdo visual e a detecção de elementos em imagens. Suas aplicações estão presentes em diversos segmentos tecnológicos que envolvem a observação, análise de imagens e reconhecimento de padrões, abrangendo diversas áreas do conhecimento, tais como Agronomia, Astronomia, Biologia, Biometria, Medicina e muitas outras. Constitui, portanto, uma área multidisciplinar com várias aplicações práticas.

Como auxílio para a implementação de algoritmos e o desenvolvimento de sistemas de Visão Computacional relacionados a detecção ou rastreamento de objetos em imagens, tem-se adotado a lógica de inicialmente realizar um pre-processamento nessas imagens afim de remover ruídos na imagens, reduzir o custo computacional, ou deixar tal imagem de maneira mais explícita para o detector.

Tendo isso em mente, observa-se que o presente projeto exige a utilização de técnicas de PDI, que em resumo é qualquer forma de processamento ou transformação dos dados presente em imagens digitais, gerando também como saída imagens, porém de uma forma que satisfaça sua aplicação final.

A maneira que as máquinas enxergam é um assunto que vem sendo observado e estudado profundamente nas ultimas décadas, na maioria das aplicações dessa área os computadores são programados para resolver determinados problemas específicos. O campo de Visão Computacional trata de manipular dados dentro de uma imagem, com a ideia de diferenciar os elementos presente nessa imagem.

Existem várias abordagens que procuram resolver problemas relacionados a detecção e rastreamento de objetos em imagens, tais como ImageJ, TensorFlow e OpenCV. O ImageJ é um domínio público desenvolvido em java com base para o processamento de imagens, pode executar varias funções, tais como calcular a área de valor de pixels em uma imagem e suporta funções de processamento de imagem padrão, sendo algumas delas operações lógicas e aritméticas entre imagens e a detecção de bordas. O TensorFlow é uma biblioteca de software de código aberto desenvolvida pelo Google para computação numérica usando gráficos de fluxo de dados, que pode ser usada na realização de aprendizado de máquina e pesquisa de redes neurais profundas. E a OpenCV é uma biblioteca de visão computacional de código aberto desenvolvida em C/C++ para o desenvolvimento de aplicações, possui módulos de processamento de imagens além de mais de 350 algoritmos de Visão Computacional, sendo a ferramenta utilizada até o presente momento neste

projeto.

A detecção e a contagem de um volume grande de objetos presentes em imagens é um problema amplamente estudado dentro do contexto de Visão Computacional. Que devido a natureza do problema não existe uma solução ótima. Problema como a detecção de bordas, que envolve mudanças repentinas no brilho de uma imagem e dependem da complexidade da mesma não atingem uma boa taxa de acerto. Geralmente estas soluções só resolvem problemas específicos do domínio de aplicação.

Dentro deste contexto, este projeto apresenta uma possível solução para os produtores de avicultura, com o desenvolvimento de um protótipo de um sistema para contagem de aves a partir de imagens capturadas por um dispositivo móvel, fazendo com que a contagem seja feita de forma rápida, dessa forma facilitando a simples conferência na quantidade dessas aves.

A avicultura envolve a criação de aves para produção de alimentos, em especial carne e ovos. Entre as espécies criadas na avicultura destaca-se o frango. Em muito menor escala, também são criadas aves como perus, patos, gansos, codornas e avestruzes. A avicultura é uma atividade econômica cada vez mais relevante mundialmente, e o Brasil está entre os principais exportadores de frango do mundo.

A entrada das aves nos aviários é realizada através de caixas e dentro dessas caixas são colocadas a princípio 50 aves, porém na maioria das vezes essa quantidade é variada podendo ter menos ou até mesmo passar deste valor. A empresa distribuidora sugere que sejam conferidas ao menos 10 dessas caixas antes de serem descarregadas dentro do aviário, no entanto a maioria dos produtores não conferem por não ter uma maneira eficiente de realizar esta tarefa, ficando a dúvida quanto à quantidade de aves estar correta.

A divergência na quantidade de aves nas caixas ocorrem devido a problemas ao inserir as aves nas caixas, ou no período de deslocamento das aves, feito da distribuidora até os aviários. O deslocamento é realizado através de caminhões podendo abalancar e fazendo com que as aves venham a cair fora das caixas, interferindo na quantidade posteriormente.

O problema de não ter uma ferramenta eficiente para conferência das aves em caixas antes de distribuí-las no aviário afeta os produtores fazendo com que eles deixem de realizar a tarefa e assim restando a incerteza sobre a quantidade estar correta. Sem um método ágil os produtores muitas vezes precisam fazer a contagem ave a ave, demandando um tempo totalmente maior afetando o processo de descarregamento.

Este projeto consiste no desenvolvimento de um protótipo de sistema que será capaz de realizar a contagem dessas aves. Isto se dá através da captura de imagens em caixas por meio de um dispositivo móvel, e com o desenvolvimento de tal sistema apresentar uma boa taxa de acerto na contagem das aves, ou seja, apresentar um resultado

que seja satisfatório.

A proposta é capturar algumas imagens das caixas com as aves e a partir dessas imagens desenvolver um algoritmo para a detecção das aves no interior das mesmas. As imagens serão submetidas no sistema desenvolvido em java que neste projeto foi vinculado com a biblioteca OpenCV com o propósito de realizar um processamento na imagem original, deixando a imagem pronta a detecção e contagem das aves. Assim, utilizando a linguagem de programação Java e a biblioteca OpenCV desenvolvendo o sistema com foco em Visão Computacional que apresente resultados satisfatórios para o usuário final.

1.1 Objetivo Geral

O objetivo geral deste projeto consiste no desenvolvimento de um protótipo funcional de sistema para plataforma android capaz de detectar e contar aves em imagens obtidas a partir de dispositivos móveis. O sistema desenvolvido será capaz de contar a quantidade de aves presentes em uma imagem com ângulo e local definido, dentro de uma margem de erro aceitável.

1.1.1 Objetivos Específicos

Os objetivos específicos deste projeto são:

- Explorar o problema de contagem de aves no momento que as mesmas chegam nos aviários.
- Analisar as técnicas de PDI para conclusão de quais são mais adequadas para este projeto.
- Aplicar técnicas de PDI nas imagens das caixas com aves, objetivando prepará-las para detecção e contagem.
- Analisar o índice de acerto na detecção e contagem de aves verificando os respectivos resultados.
- Desenvolver um protótipo funcional capaz de realizar a contagem de aves de pequeno porte em caixas, dentro de uma margem de erro aceitável.

1.2 Justificativa/Motivação

Visto que a contagem de elementos em imagens é um problema antigo do campo de Visão Computacional, que resolver determinado problema tem grande relevância não só para a computação mas para diversas áreas variando desde a Robótica até aplicações na

área da saúde, portanto optou-se por estudar algumas abordagens que procuram resolver este problema, para utilizar neste projeto.

Existem várias abordagens que possibilitam a resolução do problema, como ImageJ, TensorFlow e OpenCV. Porém não existe uma solução absoluta, por isso o encorajamento para apresentar uma abordagem eficiente que possilita a resolução do problema.

Após entender o problema que uma grande quantidade de produtores de avicultura estão vivenciando na contagem de aves no interior das caixas no processo de descarregamento nos aviários, foi observado que há a necessidade da criação de uma ferramenta que de certa maneira seja eficiente na conferência da quantidade dessas aves.

1.3 Trabalhos relacionados

A seguir são apresentados alguns trabalhos relacionados ao tema proposto, objetivando entender e/ou levantar problemas da área de aplicação, e analisar técnicas utilizadas por pesquisadores da área ou problemas semelhantes.

Em Botelho (2014) propõe uma nova abordagem para detecção de aves utilizando combinações de segmentações, tal abordagem foi capaz de segmentar imagens de alta resolução mantendo compromisso entre precisão e desempenho no processamento. O projeto avalia quantitativamente a qualidade das segmentações obtidas.

Em Quinta (2009) aborda-se o problema em realizar o controle de leveduras e bactérias presentes no processo de produção de etanol, função que é analisada manualmente por um especialista, sendo uma tarefa árdua e exaustiva. Propõe um sistema com o objetivo automatizar a contagem de leveduras viáveis e inviáveis em imagens microscópicas utilizando técnicas de Visão Computacional. O sistema utiliza técnicas de pré-processamento, segmentação de imagens, extração de atributos e reconhecimento de padrões para realização dos experimentos.

O trabalho apresentado por SILVA (2008) propõe o desenvolvimento de um sistema para contagem automática de pessoas, em ambientes internos ou externos, baseado em análise de sequências de imagens utilizando o filtro preditivo de Kalman. As pessoas são rastreadas e contadas ao deixarem a cena. O problema de pesquisa abordado no trabalho consiste em estimar fluxos multidirecionais de pessoas em ambientes internos ou externos

O trabalho apresentado por SILVA (2014) propõe aplicar técnicas de detecção e rastreamento de objetos para o desenvolvimento de um sistema de Visão Computacional utilizando a biblioteca OpenCV, assim objetivando resolver o problema de contagem de telhas de construção civil. O sistema recebe como entrada imagens gravadas por um dispositivo móvel e calcula a quantidade total de telhas presente no vídeo.

Em Morais (2005) refere-se ao problema de se estimar o número de peixes que

nadam através de mecanismos de transposição, também conhecidos como escadas de peixes. O sistema de contagem de peixes proposto no trabalho tem o objetivo de realizar o monitoramento e a avaliação da eficácia de escadas de peixe. Utiliza uma abordagem baseada em análise de sequência de imagens e no uso de um rastreador Bayesiano de múltiplos objetos. A direção de migração dos peixes é estimada dividindo a cena em regiões de interesse que são constantemente monitoradas pelo sistema.

1.4 Metodologia

A coleta de dados foi realizada através do celular Asus Zenfone 4 Selfie que possui uma câmera de 16 megapixels capaz de capturar fotos com resolução de 4608 x 3456 pixels. As fotos foram capturadas na chegada das aves no aviário, após ser retiradas as caixas do caminhão serão escolhidas algumas caixas e serão despejadas as aves em uma outra caixa já pronta que tem o fundo pintado de preto para assim capturar a imagem.

A posição da câmera para a captura das imagens foi um fator muito relevante para melhores resultados na contagem. A câmera deve estar posicionada de maneira estática e de forma que obtê-se uma imagem acima da caixa tendo visão de todas as aves no interior dela. Fatores como sombra e estremecimento na imagem podem influenciar no resultado.

Após a captura das imagens, as mesmas foram armazenadas no mesmo dispositivo móvel e submetidas ao sistema que foi instalado no mesmo dispositivo para realizar os testes de detecções das aves e obter um resultado médio de cada imagem.

Utilizou-se o programa Android Studio e a linguagem de programação Java para o desenvolvimento do protótipo, juntamente com a importação da biblioteca OpenCV. Foram inseridas as imagens no Android Studio, e nessas imagens foram aplicados métodos de PDI. A seguir é explicado algumas das técnicas que foram utilizadas neste projeto, todas essas técnicas que foram abordadas são funções nativas da biblioteca OpenCV.

Inicialmente a imagem foi carregada para o programa, após isso a imagem foi transformada em escala de preto e cinza, para esse processo foi utilizado a função cvtColor, esta função tem o papel de converter uma imagem de um espaço de cor para outro, transformando então a imagem original para de escala preto e cinza. Um exemplo desse tipo de técnica quando aplicada é apresentado no Capítulo 4 na Figura 1.

Após isso foi utilizado a técnica de limiarização (thresholding), técnica pertencente ao processo de segmentação de imagens. Para este procedimento foi utilizada a função threshold, esta função faz com que a imagem seja binarizada, com objetivo de tentar separar o fundo e as aves. Um exemplo desse tipo de técnica quando aplicada é apresentado no Capítulo 4 na Figura 2.

Utilizou-se outra técnica de segmentação de imagens, mais especificamente a fun-

ção distanceTransform. Esta função tem o objetivo de possibilitar a separação de aves que estejam muito juntas, para realização de uma contagem mais precisa das aves. Após aplicar a função distanceTransform é aplicada novamente a função threshold para binarização da imagem. Um exemplo da função distanceTransform quando aplicada é apresentado no Capítulo 4 na Figura 3.

O algoritmo desenvolvido utiliza uma função chamada findContours. Esta função permite encontrar um objeto branco a partir do fundo preto da imagem. Por isso a mesma deve estar binarizada, tornando-se uma imagem de cor preta e branca. As aves contornadas a serem encontradas devem ser brancas e fundo deve ser preto de forma a ser ignorado pelo contador.

A seguir é a etapa final do algoritmo, que objetiva realizar a contagem das aves presente na imagem. Na função findContours é retornado um array com os contornos detectados, utilizando esse array foi efetuada a contagem das quantidades de pixels presentes para cada contorno. Baseado na quantidade de pixels encontrados podemos estipular a quantidade de aves na área obtida.

A fim de melhorar o resultado na contagem das aves, quando o sistema não consegue separar aves que estão muito juntas e assim fazendo com que 2 ou mais aves sejam contadas como apenas uma, foi criada uma rotina no sistema que calcula a área média de uma ave baseado em pixels, assim é possível saber que se tal área é maior que a média de uma ave sabe-se que há mais aves na área detectada. O sistema divide essa área total pela a área média de uma ave, fazendo com que tenha mais precisão nos resultados.

A análise dos resultados da contagem de aves foram obtidos de forma que, foram capturas 50 imagens onde cada imagem possui exatas 50 aves, todas as imagens foram submetidas ao protótipo para contagem. O resultado da contagem de cada imagem foi adicionado em uma tabela que será apresentada neste trabalho.

1.5 Cronograma

1.6 Organização do Texto

A organização do texto foi dividida em seis capítulos, descritos das seguintes formas:

No capítulo 1 foi apresentado uma contextualização inicial do tema, apresentação do problema e uma breve apresentação da possível solução a ser desenvolvida, objetivo geral e específicos, justificativa/motivação, trabalhos relacionados e metodologia.

No capítulo 2, foi apresentado um referencial teórico, com uma breve contextualização sobre Processamento de Imagens Imagens, falando sobre suas respectivas técnicas.

No capítulo 3, foi apresentado um referencial teórico, apresentando algumas técnicas para detecção e rastreamento de objetos em imagens e/ou vídeos.

No capítulo 4, foi apresentado a etapa de desenvolvimento do projeto na detecção e contagem de aves.

No capítulo 5, foi apresentado resultados obtidos no presente trabalho.

No capítulo 6, foi apresentado a conclusão, limitações e contribuições deste trabalho.

No capítulo 7, foi apresentado algumas dificuldades encontradas ao longo deste trabalho.

2 Processamento digital de imagens

2.1 Introdução

Neste capítulo é apresentado algumas técnicas de PDI que possivelmente serão utilizadas nesse trabalho. Serão abordados filtros e operações básicas necessárias para o pré-processamento de imagens e logo após são apresentadas algumas técnicas de segmentação de imagens, ou seja, a subdivisão de imagens em múltiplas regiões ou objetos, etapas fundamentais do processamento.

O PDI tem por finalidade dar suporte para o desenvolvimento de algoritmos de Visão Computacional. Segundo Marengoni e Stringhini (2009), os processos de Visão Computacional, muitas vezes, necessitam de uma etapa de pré-processamento envolvendo o PDI. As imagens de onde queremos extrair alguma informação em alguns casos precisam ser convertidas para um determinado formato ou tamanho e precisam ainda ser filtradas para remover ruídos provenientes do processo de aquisição da imagem.

O PDI é qualquer forma de processamento de dados dentro de uma imagem, no qual se busca fazer transformações em uma imagem de entrada gerando uma nova imagem processada como saída, segundo SILVA (2014), o Processamento Digital de Imagens (PDI) é uma importante área da computação e possui aplicações em diversos setores da indústria, comércio e outras áreas, como a saúde por exemplo. Desde a aplicação de um simples filtro até operações complexas como a restauração de imagens, o PDI possui ferramentas robustas que permitem o desenvolvimento de aplicações que ajudam a solucionar problemas do mundo real.

Para processar uma imagem digital e extrair algum dado significativo é de certa maneira fundamental um breve conhecimento computacional ou matemático, Para Filho e Neto (1999), o Processamento Digital de Imagens envolve procedimentos normalmente expressos sob forma algorítmica. A maioria das funções de processamento de imagens pode ser implementada via software. O uso de hardware especializado para processamento de imagens somente será necessário em situações nas quais certas limitações do computador principal (por exemplo, velocidade de transferência dos dados através do barramento) forem intoleráveis.

Segundo Filho e Neto (1999, p.01):

A área de Processamento Digital de Imagens vem sendo objeto de crescente interesse por permitir viabilizar um grande número de aplicações em duas categorias distintas: 1- o aprimoramento de informações visuais para interpretação humana; e 2- a análise automática por computador, de informações extraídas de uma cena. O presente trabalho está inserido

na segunda categoria mencionada, já que um dos objetivos definidos é identificar objetos em uma cena e extrair informações relativas a eles, para então processá-las.

Existem etapas fundamentais no PDI com o objetivo de produzir o resultado esperado, fazendo com que o resultado de cada uma dessas etapas interfira de forma positiva diretamente nas etapas subsequentes. Segundo Gonzalez e Woods (2007), as etapas fundamentais para o processamento digital de imagens se dividem em: aquisição da imagem, pré-processamento, segmentação, descrição, reconhecimento e interpretação.

2.2 Aquisição da imagem

A aquisição de imagem é o processo que conversão de uma imagem de uma cena real em uma imagem eletrônica estática, com o objetivo de levá-la ao meio digital dos computadores, geralmente essa aquisição é realizada por meio de uma câmera. Para realizar esse processo é necessário um sensor e um digitalizador. O sensor tem o papel de converter a informação óptica em sinal elétrico e o digitalizador é responsável pela transformação da imagem analógica em uma imagem digital.

Dentre os aspectos importantes envolvidos nesta etapa, Filho e Neto (1999) mencionam a escolha do tipo de sensor, o conjunto de lentes a utilizar, as condições de iluminação da cena, os requisitos de velocidade de aquisição, a resolução e o número de níveis de cinza da imagem digitalizada. Esta etapa produz à saída uma imagem digitalizada da cena real.

2.3 Pré-processamento

Visto que a imagem resultante do passo anterior que é a aquisição da imagem na maioria das vezes pode apresentar inúmeras imperfeições, apresentamos o pré-processamento que é o processamento inicial dos dados brutos em uma imagem, como a correção de distorções geométricas e remoção de ruído, tendo como função melhorar a qualidade da imagem.

A função da etapa de pré-processamento é aperfeiçoar a qualidade da imagem para melhor eficiência das etapas subsequentes, Segundo Filho e Neto (1999), as operações efetuadas nesta etapa são ditas de baixo nível porque trabalham diretamente com os valores de intensidade dos pixels, sendo eles a presença de pixels ruidosos, contraste e/ou brilho inadequado etc. A imagem resultante desta etapa é uma imagem digitalizada de melhor qualidade que a original.

2.4 Representação em escala de cinza

Imagens coloridas são representadas pela sigla RGB que é um padrão e representa as três cores primárias Vermelho (Red), Verde (Green) e Azul (Blue). Através dessas três cores é possível formar qualquer outra cor.

As imagens em tons de cinza podem ser representadas por matrizes, onde cada elemento da matriz indica a intensidade do pixel correspondente. A maioria dos arquivos digitais atuais usam o número 0 para indicar a cor preta e o número 255 para indicar a cor branca que é a intensidade máxima, totalizando 256 tons de cinza diferentes entre o preto e o branco.

A representação de uma imagem em escala de cinza tem grande importância no pré-processamento, pois ela diminui o peso de processamento computacional. Para Gonzalez e Woods (2007), para diminuir o peso do Processamento Digital de Imagens, antes de qualquer operação, as imagens devem ser convertidas do formato RGB para Escala de Cinza. Computacionalmente essas imagens são armazenadas utilizando-se 8 bits (um byte) por pixel, o que possibilita 256 intensidades possíveis. A conversão de uma imagem colorida RGB em escala de cinza é uma operação simples, aplicada a todos os pontos da imagem.

2.5 Operadores morfológicos

A Morfologia Matemática (MM) tem importante papel dentro do PDI, podendo ser aplicada em diversas áreas como, segmentação de imagens, filtragem, detecção de bordas etc. Dois operadores morfológicos básicos são utilizados na maior parte dessas técnicas, a erosão e a dilatação.

A técnica de erosão já está sendo testada no projeto com o objetivo de tentar separar as aves que estão muito juntas uma as outras, assim trazendo melhor eficiência na contagem.

De acordo com Esquef, ALBUQUERQUE e ALBUQUERQUE (2003, p.09), "a erosão tem o objetivo de separar os objetos que se tocam em uma imagem, e a dilatação tem o objetivo de preencher furos ou falhas no interior de um objeto ou mesmo ligá-los".

Baseia-se em analisar a estrutura geométrica de uma imagem, tem vantagem por ser de fácil implementação, Para Filho e Neto (1999), o princípio básico da Morfologia Matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação através de outro conjunto completamente definido, chamado elemento estruturante. Portanto, a base da morfologia matemática é a teoria de conjuntos. Por exemplo, o conjunto de todos os pixels pretos em uma imagem binária descreve completamente a imagem (uma vez que os demais pontos só podem ser brancos).

2.6 Filtros

As técnicas de filtragem são transformações feita na imagem pixel a pixel. São ferramentas que objetivam a remoção de ruídos desagradáveis na imagem. Esses ruídos geralmente são fatores como, tremores e sombra, referentes ao momento de aquisição das imagens, sendo assim, fatores que prejudicam na luminosidade da imagem.

Segundo Marengoni e Stringhini (2009, p.128):

Os filtros podem ser espaciais (filtros que atuam diretamente na imagem) ou de frequência, onde a imagem é inicialmente transformada para o domínio de frequência usando da transformada de Fourier (geralmente através da transformada de Fourier discreta) e então é filtrada neste domínio e em seguida a imagem filtrada é transformada de volta para o domínio de espaço.

2.6.1 Filtro de Canny

É uma técnica para detecção de bordas, utiliza uma metodologia de multi-estágios para suavizar os ruídos e detectar as bordas . Baseiam-se em três critérios para um bom desempenho, que é uma boa detecção, uma boa localização e uma boa resposta.

De acordo com Pedroni (2011, p.28):

detectores de borda são algoritmos que visam extrair de uma imagem digital as bordas dos objetos nela presentes, sejam essas imagens coloridas ou em tons de cinza. o algoritmo criado por Canny serve como um robusto detector de bordas que funciona relativamente bem mesmo com dados ruidosos, como é o caso de imagens obtidas com câmera de custo reduzido.

2.6.2 Filtro Passa-Baixa e Filtro Passa-Alta

O Filtro Passa-Baixa tem como objetivo a diminuição de ruídos na imagem principal, aumenta de certa forma a suavidade da imagem e faz que ela tenha uma redução significativa nos detalhes.

O filtro Passa-Alta é o inverso do Filtro Passa-Baixa, pois este tem o objetivo de aumentar o detalhes na imagem, aumentando o ruido e atenuação de detalhes na mesma.

2.7 Histogramas

Técnicas de histogramas contribuem para o ajuste dos valores de intensidade de forma a melhorar o contraste em uma imagem. De acordo com Marengoni e Stringhini (2009) os histogramas são ferramentas de Processamento Digital de Imagens que possuem grande aplicação prática. Os histogramas são determinados a partir de valores de intensidade dos pixels. Entre as principais aplicações dos histogramas estão a melhora da

definição de uma imagem, a compressão de imagens, a segmentação de imagens ou ainda a descrição de uma imagem.

2.8 Segmentação de imagens

A segmentação de imagens baseia-se em sub-dividir as imagens em seus elementos de significância sendo eles fundo e objetos, resultando em um conjunto de regiões ou contornos extraídos da imagem. É uma das técnicas mais simples do processamento de imagens, de fácil implementação e pode ser feita de forma automática.

A segmentação facilita na detecção de formas e objetos na imagem, de acordo com SILVA (2014), na etapa de segmentação é feita a separação entre os objetos e o fundo da imagem, e armazenadas as coordenadas dos pontos de cada objeto em estruturas de dados. Como a imagem limiarizada não apresenta variação nas cores de preenchimento dos objetos, pois todos são representados pela cor branca, somente as coordenadas das bordas dos objetos são armazenadas.

2.8.1 Limiarização (thresholding)

A limiarização é um processo da segmentação de imagens e em outras palavras funciona como a binarização de uma imagem, assim convertendo uma imagem em forma binária com objetivo de diminuir o peso do processamento, varre-se a imagem pixel a pixel classificando cada pixel como sendo objeto ou fundo, dependendo do nível de cinza do pixel. Ou seja é necessário analisar a semelhança dos níveis de cinza da imagem para extrair os objetos de interesse.

Segundo Filho e Neto (1999, p.71):

O princípio da limiarização consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto). Devido ao fato da limiarização produzir uma imagem binária à saída, o processo também é denominado, muitas vezes, binarização. A forma mais simples de limiarização consiste na bipartição do histograma, convertendo os pixels cujo tom de cinza é maior ou igual a um certo valor de limiar (T) em brancos e os demais em pretos.

2.9 Detecção de bordas

É uma técnica de segmentação de imagens com a finalidade de detectar mudanças repentinas na luminosidade da imagem, podendo diferenciar o objeto e o fundo. Fatores na imagem como o ruído e sombras podem criar bordas falsas e prejudicar a detecção, trazendo resultados incorretos.

Borda é uma característica muito importante dentro de uma imagem, em termos computacionais pode ser de fácil detecção, segundo Queiroz e Gomes (2006), a detecção de bordas possibilita a análise de descontinuidades nos níveis de cinza de uma imagem. As bordas na imagem de interesse caracterizam os contornos dos objetos nela presentes, sendo bastante úteis para a segmentação e identificação de objetos na cena. Pontos de borda podem ser entendidos como as posições dos pixels com variações abruptas de níveis de cinza. Os pontos de borda caracterizam as transições entre objetos diferentes.

2.10 Segmentação orientada a regiões

A segmentação orientada a regiões se constitui na similaridade dos níveis de cinza de uma imagem, nela aplica-se o procedimento de crescimento de regiões, caracterizado pelo agrupamento de pixels, esse procedimento agrupa os pixels de uma imagem em regiões maiores, com o objetivo de deixá-los juntos com seus similares de cores cinza.

2.10.1 Crescimento de regiões por agregação de pixels

A técnica de agregação de pixels consiste em buscar características semelhantes nos pixels adjacentes, é uma técnica semelhante a de limiarização que separa as regiões de uma imagem dependendo de sua escala de cinza, porém a diferença é que esta técnica inicia a partir de determinados pixels específicos, denominados de sementes.

É uma técnica simples da segmentação orientada a regiões, de acordo com Queiroz e Gomes (2006), se fundamenta na definição de uma semente, que é um conjunto de pontos similares em valor de cinza, a partir do qual as regiões crescem com a agregação de cada pixel à semente à qual estes apresentem propriedades similares. A técnica apresenta algumas dificuldades fundamentais, se afigurando como problemas imediatos a seleção de sementes que representem adequadamente as regiões de interesse, e a seleção de propriedades apropriadas para a inclusão de pontos nas diferentes regiões, durante o processo de crescimento.

2.10.2 Difusão e fusão de regiões

Esta técnica consiste em agrupar ou separar regiões de uma imagem, caso a mesmas tenham propriedades similares, por exemplo, se a escala de cinza dessas regiões forem semelhantes. As técnicas de fusão e difusão podem ser combinadas de modo a trazer vantagens na solução de problemas em segmentação de imagens.

Segundo Gonzalez e Woods (1992 apud SILVA, 2014, p.18):

Segmentar uma imagem por meio dessa técnica consiste em subdividir a imagem de entrada sucessivamente em quadrantes que satisfaçam certa

propriedade. Basicamente consiste em agrupar regiões adjacentes caso sejam similares ou dividir a região caso ela seja homogênea. Durante o processo, deve-se verificar para cada quadrante se a propriedade estabelecida é satisfeita, quando isso não ocorre, o quadrante é subdividido e o processo continua. O processamento termina quando não é mais possível agrupar ou dividir as regiões.

Segundo SILVA (2014, p.18), "Uma desvantagem desta técnica é que a segmentação final tende a ter regiões com formas quadradas, devido à segmentação ser feita a partir da produção de divisões quadradas da imagem original".

2.11 OpenCV

A biblioteca OpenCV é de livre e total acesso acadêmico e comercial, sua estrutura foi desenvolvida nas linguagens de programação C/C++, porém ela da suporte a desenvolvedores que utilizem Java, Python e Visual Basic, podendo a biblioteca ser incorporada em seus aplicativos.

Algumas de suas áreas de aplicação são, identificação de objetos, sistema de reconhecimento facial, reconhecimento de movimentos, realidade virtual, realidade aumentada. A estrutura da biblioteca está dividida em funções como: Processamento de imagens; Análise estrutura de dados; Controle de Interface e dispositivos de entrada; Análise de movimento e rastreamento de objetos; Reconhecimento de padrões e Machine Learning (Aprendizagem de Maquina).

É uma biblioteca multiplataforma que possibilita o desenvolvimento de aplicativos na área de Visão Computacional. Segundo Marengoni e Stringhini (2009) foi idealizada com o objetivo de tornar a visão computacional acessível a usuários e programadores em áreas tais como a interação humano-computador em tempo real e a robótica. A biblioteca está disponível com o código fonte e os executáveis (binários) otimizados para os processadores Intel. Um programa OpenCV, ao ser executado, invoca automaticamente uma DLL (Dynamic Linked Library) que detecta o tipo de processador e carrega, por sua vez, a DLL otimizada para este. Juntamente com o pacote OpenCV é oferecida a biblioteca IPL (Image Processing Library), da qual a OpenCV depende parcialmente, além de documentação e um conjunto de códigos exemplos.

2.12 Conclusão

Enfim entende-se que o PDI é uma forma de tornar a imagem mais simples para extrair suas características, trazendo grandes vantagens para detecção de elementos. Faz-se o uso desse processamento para tornar a imagens mais pequenas ou leves fazendo com que o processamento computacional seja mais eficiente.

Suas técnicas trazem possibilidades de separar informações de uma imagem, facilitando a diferenciação entre o fundo e o objeto presente na mesma. Assim após utilizar algumas dessas técnicas é possível desenvolver algoritmos de visão computacional capazes de detectar objetos de interesse em uma imagem, podendo então determinar a quantidade dos mesmos.

Ou seja neste projeto está sendo feito o processamento das imagens para que elas estejam de uma maneira que satisfaçam o algoritmo para detecção das aves, como transformadas para escala preta e cinza, fazendo com que essas imagens sejam exploradas pelo algoritmo com um menor custo computacional. O algoritmo de Visão Computacional em desenvolvimento trata de manipular os dados presente nas imagens já processadas.

3 Detecção e rastreamento de objetos

3.1 Introdução

Na seção relacionada ao PDI foi abordado alguns estágios do processo de nível mais baixo, ou seja, processos utilizados para eliminação de ruídos, melhorias no contraste da imagem. Nesta seção é abordado métodos mais relacionados a Visão Computacional considerados processos de nível mais alto, como o rastreamento de objetos em imagens.

A detecção e o rastreamento de objetos em imagens tem grande importância em sistemas de Visão Computacional, no qual existem diversas áreas que adotam sistemas dessa especialidade. Sistemas como os de monitoramento de tráfego de veículos, monitoramento automático de ações humanas, e muitos outros.

Sendo o campo de Visão Computacional o estudo da extração de informação de uma imagem, mais especificamente, é a construção de descrições explícitas e claras dos objetos em uma imagem, Ballard e Brown (1982), diferencia do Processamento de Imagens porque, enquanto ele se trata apenas da transformação de imagens em outras imagens, a visão computacional trata explicitamente da obtenção e manipulação dos dados de uma imagem e do uso deles para diferentes propósitos.

Um sistema de Visão Computacional pode aplicar várias transformações em uma imagem e descobrir bordas, os objetos que são apresentados, perspectiva e movimento quando apresentado com várias imagens. De acordo com Quinta (2009), na área da Visão Computacional, são desenvolvidos algoritmos para obtenção de informações a partir de imagens, como detecção e rastreamento, e algumas vezes, buscando a automatização de tarefas geralmente associadas à visão humana. Na visão humana, os olhos capturam as imagens e posteriormente o cérebro realiza a análise e identificação de seu conteúdo. A Visão Computacional possui uma série de etapas para reproduzir essa tarefa realizada pelos seres humanos.

Além desse campo, o campo de IA, principalmente as áreas de aprendizado de máquina e de reconhecimento de padrões, é muito empregado para o entendimento da informação gerada a partir da imagem. Outro campo importante é a própria Física, que explica como radiação numa certa frequência sensibiliza sensores óticos e como sua trajetória se comporta ao atingir um anteparo. Outros campos também relacionados são o processamento de sinais, PDI, Visão de Máquina, entre outros.

O problema de contagem é amplamente estudado no campo de Visão Computacional, para Lempitsky e Zisserman (2010), o problema de contagem é uma estimativa do número de objetos em uma imagem estática ou quadro de vídeo. Surge em muitas aplica-

ções do mundo real, incluindo a contagem de células em imagens microscópicas, sistemas de vigilância e realização de censo de vida selvagem ou contagem do número de árvores em uma imagem aérea de uma floresta.

3.1.1 Extração de atributos

Extrair dados presente em imagens é de fato o objetivo principal da Visão Computacional, e a extração de atributos é a parte propriamente dita de analisar o conteúdo da imagem. De acordo com Quinta (2009), a extração de atributos visa obter informações sobre um determinado objeto. Essas informações serão utilizadas para caracterizar objetos de uma mesma classe. Em uma imagem, são apresentadas diversas regiões. A etapa de segmentação irá evidenciar apenas a região de interesse, nesse caso, os micro organismos. Nesse momento, a extração de atributos irá ser aplicada. Necessita-se discriminar os objetos e caracterizá-los de acordo com as classes que eles constituem.

Quando a dimensão do espaço dos atributos é grande faz com que o custo computacional para detecção desse atributo seja maior. Segundo Santos (2007), o melhor critério de qualidade para avaliação e detecção dos subconjuntos de atributos é o mínimo erro, onde as imagens são classificadas segundo os subconjuntos selecionados e considera-se como o melhor subconjunto de imagens, aquele que apresenta a maior exatidão na classificação dos atributos.

3.2 Técnicas para detecção de objetos

Existem diversas técnicas de detecção de objetos em imagens, porém nenhuma delas traz uma solução definitiva para tal detecção. Geralmente são soluções que resolvem problemas específicos do domínio de aplicação, não sendo soluções genéricas.

As técnicas de detecção de objetos buscam determinar a posição de certo objeto em uma imagem, segundo SILVA (2014), existem diferentes maneiras de se detectar objetos em imagens. Geralmente, a detecção é baseada na identificação de características dos objetos de interesse que estejam presentes na imagem. A seleção das características a serem utilizadas depende diretamente do problema e é uma escolha fundamental para o sucesso da identificação.

O objetivo dessas técnicas é um índice de detecção de objetos próximo aos 100%, para Morais (2005, apud SILVA, 2014, p.25), a ideia fundamental é associar características como cor ou segmentos, quadro-a-quadro na sequência de imagens. Técnicas mais recentes utilizam filtros preditivos capazes de estimar o estado de um alvo no próximo quadro da sequência. A estimativa permite determinar qual objeto detectado no próximo quadro melhor corresponde ao objeto rastreado. Com a correspondência quadro-a-quadro, é possível extrair parâmetros do movimento realizado como trajetória e velocidade.

3.2.1 Reconhecimento de padrões

Reconhecimento de padrões é uma área da ciência que pode ser utilizada no processo de classificação de objetos presentes em uma sequencia de imagens. Várias aplicações implementam algoritmos baseado em reconhecimento de padrões, como reconhecimento de escrita, reconhecimento de faces etc. É bastante utilizado no sub-campo de aprendizagem de maquina, objetivando treinar maquinas que possam reconhecer padrões em uma categoria de imagens, sinais e outros.

Muitas das técnicas de reconhecimento de padrões baseiam-se em Redes Neurais, técnicas matemáticas e/ou estatísticas, Para Souza et al. (1999), o objetivo principal do reconhecimento de padrões a distinção entre os diferentes tipos de padrões, a capacidade de discriminação é colocada em questão. Diferentes padrões são compostos de características diferentes, com valores numéricos diferentes ou relação entre as próprias características diferentes. A maioria dos classificadores (Estatísticos, Sintáticos ou Neurais) são baseados no conceito de similaridade.

3.2.2 Redes Neurais Artificiais

Uma forma de detectar objetos em imagens é por meio de RNA (Redes Neurais Artificiais), que são inspiradas na funcionalidade dos neurônios biológicos. É possível treina-las para de detecção ou reconhecimento de objetos em imagens, com isso o campo de Visão Computacional determina as redes neurais como forma de soluções promissoras para problemas de reconhecimento. Algumas das aplicações que mais utilizam essa tecnologia são aplicações de Robótica, Reconhecimento de Caracteres, Reconhecimento de Voz etc.

Segundo Osório, Bittencourt e Osório (2000, p.07):

as Redes Neurais Artificiais (RNA), também conhecidas como métodos conexionistas, são inspiradas nos estudos da maneira como se organiza e como funciona o cérebro humano. Este tipo de método possui características peculiares de representação e de aquisição de conhecimentos, sendo considerado um método de nível sub-simbólico.

Através das RNAs é possível desenvolver aplicações para aprendizado de maquina, bem como reconhecimento de padrões em imagens, segundo Perelmuter et al. (1995), a Visão Artificial é efetuada através de um conjunto de transformações, que permitem a extração dos aspectos invariantes das imagens. Tais invariâncias possibilitam o reconhecimento ou a caracterização da imagem, permitindo a interação do sistema de visão com as mesmas. Pode-se dizer então que a visão artificial procurar perceber a informação presente em uma imagem com o objetivo de classificação e/ou caracterização dessa imagem.

3.2.3 Métodos exatos

São métodos usados para detecção de objetos em imagens, segundo SILVA (2014), as características utilizadas nos métodos exatos são relativas e variam de acordo com alguns aspectos: distância, posicionamento e ângulo do(s) objeto(s) em relação à câmera. Por isso, outro requisito fundamental para o uso desse método é que o posicionamento da câmera em relação à cena seja fixo e previamente conhecido, o que geralmente ocorre em ambientes controlados.

Os métodos exatos se baseiam em definir alguma característica do objeto presente na imagem como, distância, raio ou altura para que o objeto seja detectado, de acordo com SILVA (2014), esse método é recomendado quando os objetos de interesse possuem características bem definidas e constantes. Pode ainda servir de auxílio para outros métodos mais complexos, pois como sua aplicação se baseia em operações simples sobre imagens, apresenta um baixo custo computacional. Logo, pode ser utilizado como etapa complementar para evitar o uso desnecessário desses métodos em determinadas situações.

3.3 Técnicas para rastreamento de objetos

As técnicas para rastreamento de objetos em imagens, são determinadas pela maneira como determinam a trajetória de um objeto em uma imagem ou vídeo, porém não é uma tarefa fácil devido a fatores como a quantidade e movimentação repentina dos objetos nas imagens. E essas mudanças ou variações dos objetos certamente influenciará no resultado desejado.

O rastreamento é considerado como o pré-processamento dos dados para aplicações de reconhecimento de objetos, ou seja, primeiramente é feito o rastreamento, após isso será feito o reconhecimento desse objeto podendo o mesmo ser de interesse ou irrelevante para o sistema. De acordo com SILVA (2014), o rastreamento de objetos é uma etapa fundamental no processo de contagem. A contagem automática de objetos é feita a partir de uma sequência de imagens (vídeo) como entrada e o processamento dessas imagens é realizado frame a frame. Logo, é necessária uma forma de armazenar os estados dos objetos ao longo do tempo e isso é feito através do rastreamento. Dessa forma, é possível correlacionar objetos entre imagens e evitar que um mesmo objeto seja contado mais de uma vez.

De acordo com Marengoni e Stringhini (2009, p.146):

O processo de rastreamento é um processo de reconhecer um padrão em uma sequência de imagens. O rastreamento poderia ser feito desta forma, porém, a busca em cada imagem de uma sequência sem o uso de qualquer conhecimento específico é relativamente lenta. Os processos de rastreamento atrelam um conhecimento sobre o movimento do objeto

que está sendo rastreado para minimizar a busca entre as imagens em uma sequência.

3.3.1 Filtro de Kalman

É tipo de solução probabilística para rastreamento de objetos em uma sequência de imagens ou vídeos, tem sido amplamente estudado em campos de engenharia, robótica e outras. Seu objetivo é ajudar o computador a seguir ou monitorar objetos em vídeos e/ou imagens com eficiência e gerar resultados que se aproximam ao máximo dos valores reais das características dessa fonte. É desejável que seja realizado um pre-processamento antes, processos como a segmentação para redução de custo computacional e consequentemente obter maior performance.

Os filtros de Kalman são baseados em álgebra linear, e são soluções do tipo recursivas, segundo SILVA (2008, p.76), o filtro de Kalman é amplamente utilizado em aplicações da engenharia, controle modernos, navegação e robótica móvel tendo como foco o rastreamento de objetos em movimento. Se um sistema pode ser descrito através de um modelo linear e as incertezas dos sensores e do sistema podem ser modelados como ruídos gaussianos brancos, então o filtro de Kalman possibilita uma predição estatisticamente ótima para os dados estimados.

3.3.2 CONDENSATION (Conditional Density Propagation)

É um algoritmo de visão computacional, e uma abordagem denominada simples comparado as abordagens baseadas no Filtro de Kalman. Seu objetivo principal é rastrear e detectar contornos em constante movimento em uma sequência de cenas e/ou vídeos. Tem sido utilizado em aplicações para reconhecimento de gestos humanos em uma sequência de imagens, reconhecimento facial em uma sequência de vídeos, entre outras.

Rastrear objetos é uma das tarefas mais difíceis da visão computacional, e essa abordagem dedica-se a explorar a possível solução desse problema. Segundo Morais (2005), o algoritmo Condensation é uma abordagem probabilística para rastrear objetos em cenas ambíguas, como por exemplo objetos camuflados em que o objeto de interesse imita características do plano de fundo. Algoritmos anteriores possuem limitações na representação de distribuições de probabilidades como o Filtro de Kalman que assume distribuição Gaussiana para os eventos.

3.3.3 Corner Detection

Corner Detection (Detecção de canto) é uma técnica utilizada em sistemas de Visão Computacional para rastreamento de objetos. É frequentemente utilizada em aplicações

de detecção de movimento, modelagem 3D, monitoramento de video, reconhecimento de objetos e outras.

Um canto pode ser definido como o ponto de intersecção entre duas arestas, ou um ponto ao encontro de duas bordas. Técnicas desse tipo geralmente são robustas e exigem muitas vezes varias repetições com o objetivo de dominar a tarefa de reconhecimento com o mínimo de erros.

A maioria dos métodos de detecção de canto detecta pontos de interesse em geral. Segundo Marengoni e Stringhini (2009) este tipo de técnica pressupõe a busca por um mesmo objeto de interesse em sequências de frames num stream de vídeo. A ideia básica é buscar pontos diferenciados em uma imagem, passíveis de serem novamente encontrados em frames subsequentes.

3.3.4 Subpixel Corners

É uma técnica que traz maior precisão na extração de pontos de interesse em uma sequencia de imagens, pois esta técnica pode trabalha com equações matemáticas para obter pontos entre os pixels. De acordo com Marengoni e Stringhini (2009), as técnicas de localização de subpixels são utilizadas para que se obtenha com maior precisão a localização de detalhes de uma imagem. Entre as aplicações estão o rastreamento em reconstruções tri-dimensionais, calibragem de câmera, reconstrução de imagens repartidas, localização precisa de elementos em uma imagem de satélite, entre outras.

3.4 Conclusão

As técnicas de detecção e rastreamento de objetos possibilitam-nos o desenvolvimento de aplicações de Visão Computacional capazes de detectar e contar elementos em imagens e/ou vídeos. É indispensável que essas técnicas sejam estudadas e analisadas para que sejam utilizadas de maneira eficiente para obtenção do resultado desejável na contagem.

Neste trabalho até o momento como forma de detecção das aves está sendo utilizada a técnica de segmentação de imagens, fazendo com que a imagem seja sub-dividida em fundo e objeto, dependendo do valor de cada pixel ele será classificado como branco, assim sendo parte do objeto detectado, caso contrario esse pixel será classificado como preto, sendo parte de fundo da imagem. E em termos de detecção de bordas está sendo testado o Filtro de Canny.

4 Desenvolvimento

4.1 O sistema desenvolvido

O sistema desenvolvido neste presente trabalho foi desenvolvido em linguagem Java para plataforma Android utilizando a IDE Android Studio integrada com a biblioteca de recursos de Visão Computacional OpenCV. O Sistema possibilita que o usuário escolha a maneira que deseja fazer a contagem das aves, de forma que possa capturar a imagem pelo sistema e realizar a contagem, ou selecionar a imagem a partir da galeria no dispositivo móvel.

4.2 Implementações e Imagens

Logo abaixo (Código 4.1) é apresentado a implementação no algoritmo da função cvtColor que tem como objetivo transformar a imagem original definida como "src"em uma imagem de escala preto e cinza. A imagem convertida é armazenada na variável "cannyEdges".

Código 4.1 – Função cvtColor

```
Imgproc.cvtColor(src, cannyEdges, Imgproc.COLOR_BGR2GRAY);
```

Figura 1 – Exemplo de imagem em escala preto e cinza



A Figura 1 representa o modelo de imagem quando é realizado o processo de transformação da imagem original em escala preto e cinza. Realizado através da função cvtColor da biblioteca OpenCV.

Abaixo (Código 4.2) é apresentado a implementação da função threshold objetivando binarizar a imagem que estava em escala de preto e cinza estando definida como "cannyEdges", assim a imagem binarizada é armazenada em "bin". A função threshold recebe 1 parâmetro de extrema importância para binarização da imagem, que é o parâmetro de valor 53.

Este parâmetro define uma limiar significando que os pixels de valores abaixo de 53 serão transformados em cor branca passando a ter valor 1 e os pixels de valores acima de 53 serão transformados em preto obtendo valor 0, fazendo com que a imagem possua apenas 2 cores/valores se tornando binaria. Ou seja, o que for definido como 1 o algoritmo detecta como o objeto presente na imagem, e 0 como fundo da imagem.

Código 4.2 – Função threshold

```
Imgproc.threshold(cannyEdges, bin, 53, 255,  
Imgproc.CV_THRESH_BINARY);
```

Figura 2 – Exemplo de imagem binarizada



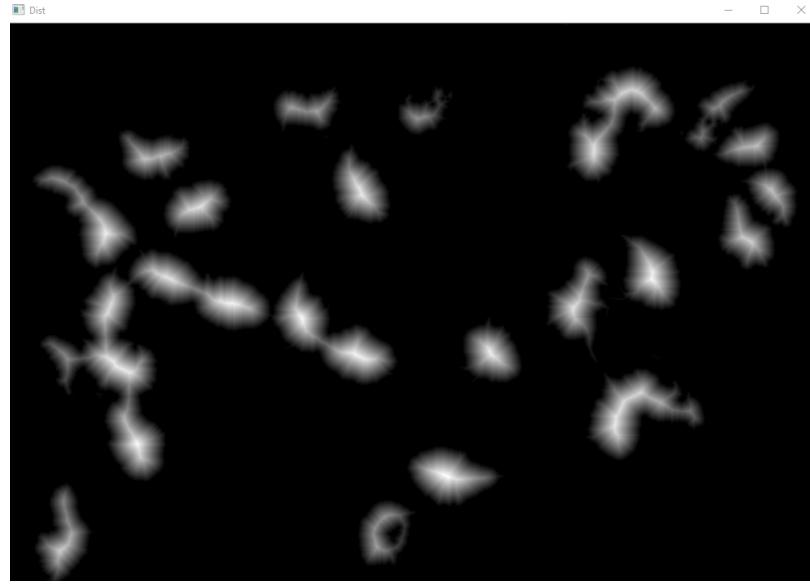
A Figura 2 representa o modelo de imagem após ser realizado o primeiro processo de limiarização (thresholding). Processo realizado através da função threshold da biblioteca OpenCV.

Abaixo (Código 4.3) é apresentado a implementação da função distanceTransform com o objetivo de separar as aves que estão juntas, para melhor precisão na contagem. Após a execução dessa função é gerada um novo tipo de imagem, por este motivo a imagem deve ser normalizada novamente para prosseguir.

Código 4.3 – Função distanceTransform

```
Imgproc.distanceTransform(bin, dist, Imgproc.CV_DIST_L2, 3);
```

Figura 3 – Exemplo de imagem aplicada a função distanceTransform



A Figura 3 representa o modelo de imagem após ser aplicada a função distanceTransform com o objetivo de obter a separação das aves que estão juntas.

Abaixo (Código 4.4) é apresentado a implementação da função normalize que tem como objetivo normalizar a imagem após a execução da função distanceTransform, e após isso a imagem é binarizada novamente para prosseguir com a detecção.

Código 4.4 – Função normalize

```
Core.normalize(dist, dist, 0, 1, Core.NORM_MINMAX);
Imgproc.threshold(dist, dist, .5, 1, Imgproc.CV_THRESH_BINARY);
```

Figura 4 – Exemplo de imagem binarizada



A Figura 4 representa o modelo de imagem após ser realizado um segundo pro-

cesso de limiarização (threshold). Esta segunda chamada da função threshold faz com que algumas aves que ficaram coladas na primeira chamada sejam separadas. Processo realizado após aplicar a função distanceTransform.

Abaixo (Código 4.5) é apresentado a implementação no algoritmo utilizada para desenhar retângulos nas áreas detectadas, foi bastante utilizada no desenvolvimento do sistema para visualização das áreas detectadas.

Código 4.5 – laço para desenho de retângulos nas áreas detectadas

```
for (int i = 0; i < contornos.size(); i++){
    if (Imgproc.contourArea(contornos.get(i)) > 0){
        Rect rect = Imgproc.boundingRect(contornos.get(i));
        Imgproc.rectangle(src, new Point(rect.x, rect.y),
                          new Point(rect.x + rect.width, rect.y +
                          rect.height), new Scalar(0, 0, 255), 2);
    }
}
```

Figura 5 – Resultado detecção das aves



A Figura 5 representa um exemplo na detecção das aves na caixa, observa-se que nesta imagem foi desenhado retângulos na áreas detectadas, como foi dito no capítulo 1 as áreas detectadas possuem valores em pixels, então a área dos retângulos desenhados quando são maiores que a área média que uma ave possui são divididas para obter a quantidade correta de aves presente naquela área. Neste exemplo havia no total 30 aves, porém foram detectadas 27 delas.

Abaixo (Código 4.6) é apresentada a implementação no algoritmo definida para divisão e a contagem das áreas detectadas quando elas são maiores que a área de apenas

uma ave. Nota-se que valor médio da área de uma ave foi definido como 30000 pixels.

Código 4.6 – laço para divisão das áreas com mais de uma ave

```

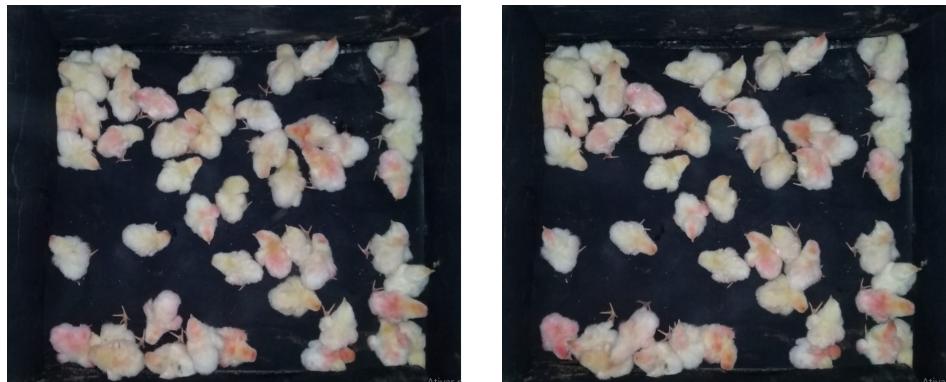
for (Double area : areas){
    double areaValida = area/30000;
    if(areaValida > 1) {
        areas_validas = areas_validas + (int) areaValida;
    } else{
        areas_validas++;
    }
}

```

4.2.1 Imagens originais

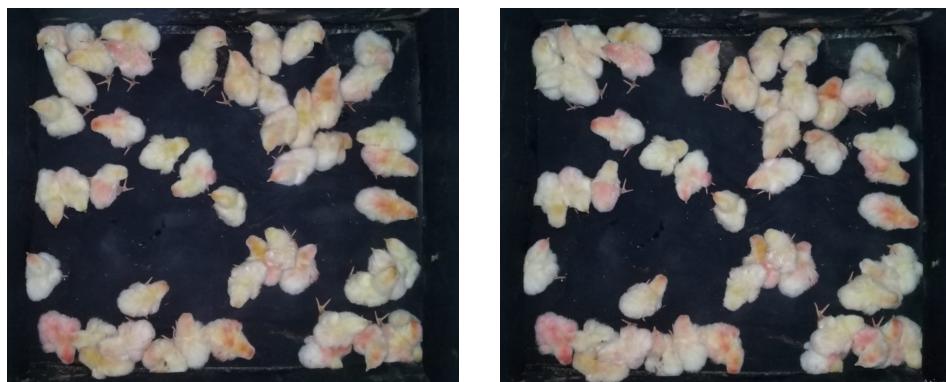
Abaixo nas Figuras 6 e 7 foram apresentadas imagens capturadas para os testes do projeto, tais imagens foram submetidas ao sistema a partir da galeria do dispositivo móvel obtendo os respectivos resultados que serão apresentados logo mais a frente.

Figura 6 – Imagens originais capturadas



A Figura 6 representa as imagens capturadas no aviário para contagem

Figura 7 – Imagens originais capturadas



A Figura 7 representa as imagens capturadas no aviário para contagem das aves através do sistema

Foram capturadas uma sequencia de 50 imagens para a realização de testes no sistema, acima foram apresentadas algumas dessas imagens como exemplo, a imagens mostram a aparência de serem iguais, porem se observar atentamente pode-se ver a diferença nas posições das aves, devido as aves sempre estarem em constante movimento tem-se que a cada intervalo de captura as aves já se deslocaram tornando as imagens que foram capturadas sequencialmente distintas.

4.2.2 Protótipo final

O protótipo final deste projeto consiste em aplicativo de interface simples, fazendo com que o usuário tenha facilidade na utilização do sistema. Mas a frente será apresentada a interface do sistema em funcionamento, como pode-se observar ao iniciar o sistema o mesmo é composto por 2 botões de navegação com funções distintas, sendo eles:

4.2.2.1 Botão CÂMERA

Através desse botão o usuário é capaz de capturar a imagem para realização da contagem, ao acionar o botão câmera o usuário captura a imagem podendo também descartar a imagem para captura de uma nova, ou prosseguir para contagem das aves, como pode-se observar na Figura 8.

4.2.2.2 Botão GALERIA

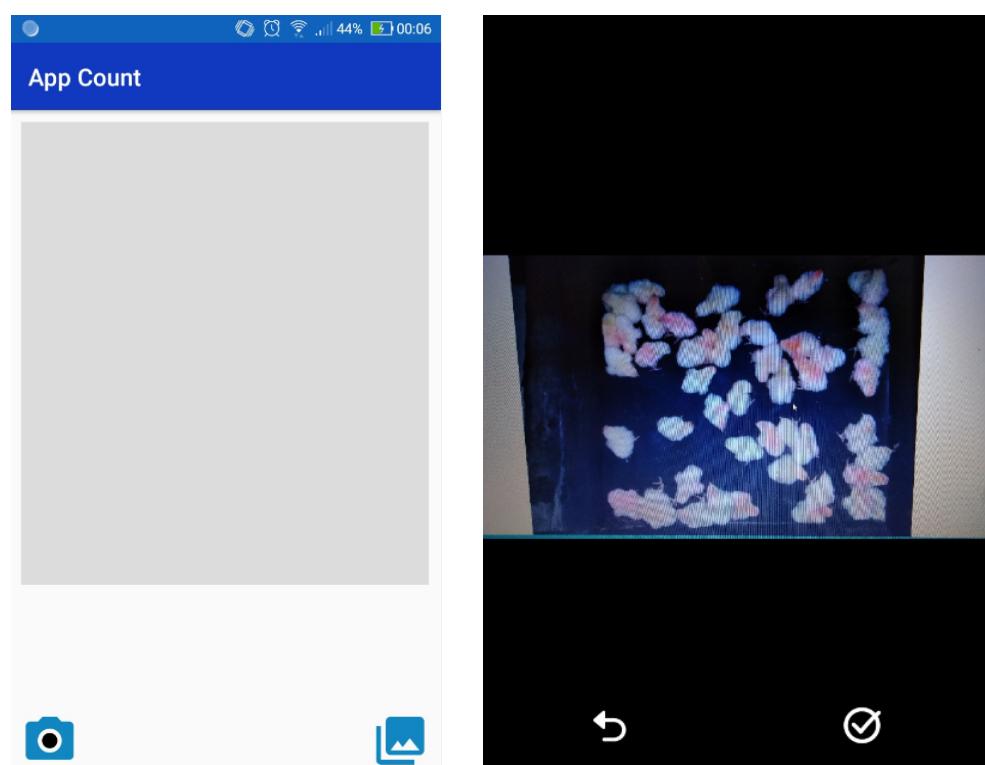
Através desse botão o sistema permite que o usuário escolha a foto a partir da galeria do dispositivo móvel, ou seja o usuário pode optar por capturar varias imagens e armazena-las no dispositivo sem a obrigatoriedade de realizar a contagem no exato no momento da captura das mesmas, assim o mesmo pode executar o aplicativo posteriormente e submeter as imagens armazenadas na galeria.

4.2.2.3 Interface do sistema

Abaixo na Figura 8 é apresentada a interface gráfica do sistema desenvolvido neste presente trabalho, sendo ela responsável pela comunicação entre o usuário e o sistema. Interface desenvolvida em linguagem Java utilizando Android Studio.

Como pode-se observar na Figura 8 o protótipo desenvolvido foi renomado de App Count (Aplicativo de contagem), e sua interface é representada pela cor azul.

Figura 8 – Interface do sistema



5 Resultados

5.0.1 Introdução

Este capítulo tem como objetivo apresentar os principais resultados obtidos a partir dos testes realizados com o protótipo de contagem de aves de pequeno porte utilizando técnicas de Visão Computacional desenvolvido. O conjunto de Imagens é composto por uma série de fotografias capturadas por um dispositivo móvel no interior de um aviário na localidade de Gloria de Dourados MS, a aquisição dessas imagens ocorreu durante a chegada e distribuição das aves no aviário.

5.0.2 Considerações iniciais

A análise de desempenho do sistema levou em consideração a qualidade dos resultados obtidos na contagem de aves, sendo que foram executados testes de contagem em 50 imagens capturadas e em todos os testes obteve-se resultados aproximados da quantidade exata das aves na imagem.

5.0.3 Resultados obtidos

Durante a fase dos testes observa-se que existe uma variação na detecção e contagem das aves. Como pode-se observar na Tabela 1 e 2 em 48 imagens obtém-se valores aproximados de 50 aves e que em 2 imagens restantes obtém-se o valor exato de aves presente na imagem. Para todos os testes realizados foi utilizado o valor total de 50 aves em cada imagem.

A Tabela 1 e 2 descreve uma sequência de 50 imagens capturadas para a realização dos testes de contagem. As tabelas apresentam a quantidade de imagens, quantidade de aves presente na imagem, quantidade de aves detectada na imagem e a diferença entre contagem detectada e quantidade total.

A Tabela 3 apresenta a quantidade total de aves presente em 50 imagens, assim como o total detectado dessas 50 imagens e diferença na quantidade de aves detectadas em relação ao total.

Os resultados apresentados demonstram a capacidade do protótipo de separar as regiões sendo elas o fundo da imagem e as aves, fazendo com que tenham valores diferentes para detecção e contagem de aves. Ou seja o algoritmo faz com que só as regiões contornadas como mostra a Figura 5 do Capítulo 4 tenham um valor significante para contagem, não tendo relevância as demais regiões.

Tabela 1 – Tabela com resultados dos testes em imagens

Imagen	Total Detectado	Total de Aves	Diferença
1	40	50	-10
2	34	50	-16
3	36	50	-14
4	39	50	-11
5	41	50	-9
6	39	50	-11
7	47	50	-3
8	52	50	+2
9	47	50	-3
10	45	50	-5
11	40	50	-10
12	41	50	-9
13	42	50	-8
14	37	50	-13
15	40	50	-10
16	33	50	-17
17	36	50	-14
18	48	50	-2
19	30	50	-20
20	48	50	-2
21	45	50	-5
22	52	50	+2
23	46	50	-4
24	50	50	0
25	43	50	-7

Como pode-se observar na Tabela 3 foram contadas uma quantidade de 2213 de 2500 aves presentes em 50 imagens, obtendo em geral uma margem de acerto de 88,52 por cento e uma margem de erro de 11,48 por cento.

Na equação (5.1) apresentada abaixo esta indicada a forma que foi realizado o cálculo da porcentagem de acerto acumulado do protótipo desenvolvido. Onde 2213 é a quantidade de aves contadas e 2500 é a quantidade total de aves nas 50 imagens.

$$Resultado = \left(\frac{2213}{2500} \right) * 100 = 88,52\% \quad (5.1)$$

Nota-se que os erros cometidos pelo sistema que ocasionam uma contagem de aves diferente da real são gerados basicamente pelas seguintes ocasiões:

- A ocorrência de mais de uma aves nas regiões detectadas, isso ocorre devido a algumas aves estarem juntas nas fotografias capturadas. Ou seja, quanto mais separadas as aves antes da captura da imagem, mais preciso o resultado.

- O estremecimento da imagem no momento da captura da mesma, devido o dispositivo móvel estar em mãos, não estando em um local fixo para captura da imagem esta sujeito a essa imprecisão.
- A luminosidade do ambiente no qual a caixa de aves foi posicionada para captura das imagens e fatores como sombra afetam o resultado na contagem.

Tais erros cometidos pelo protótipo não constituem de fato particularidade do algoritmo desenvolvido, mas sim uma limitação determinada pelas características das próprias imagens capturadas. Sendo assim é comum que na ocorrência de situações citadas acima o protótipo não realize a contagem de forma correta.

Tabela 2 – Tabela com resultados dos testes em imagens

Imagen	Total Detectado	Total de Aves	Diferença
26	40	50	-10
27	37	50	-13
28	44	50	-6
29	47	50	-3
30	43	50	-7
31	46	50	-4
32	51	50	+1
33	52	50	+2
34	52	50	+2
35	45	50	-5
36	51	50	+1
37	51	50	+1
38	51	50	+1
39	52	50	+2
40	52	50	+2
41	49	50	-1
42	51	50	+1
43	27	50	-23
44	40	50	-10
45	46	50	-4
46	45	50	-5
47	51	50	+1
48	45	50	-5
49	50	50	-1
50	44	50	-6

Tabela 3 – Tabela com resultados dos testes em imagens

Numero de Imagens	Total Detectado	Total de Aves	Diferença
50	2213	2500	287

5.0.4 Imagens de resultados obtidos

Figura 9 – Imagens de resultados obtidos

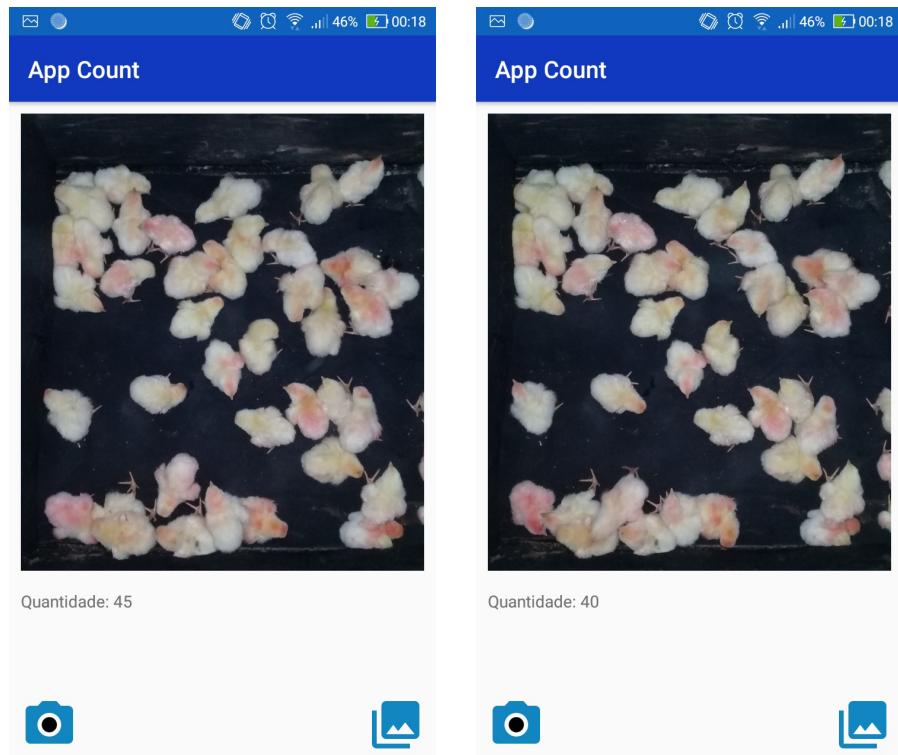
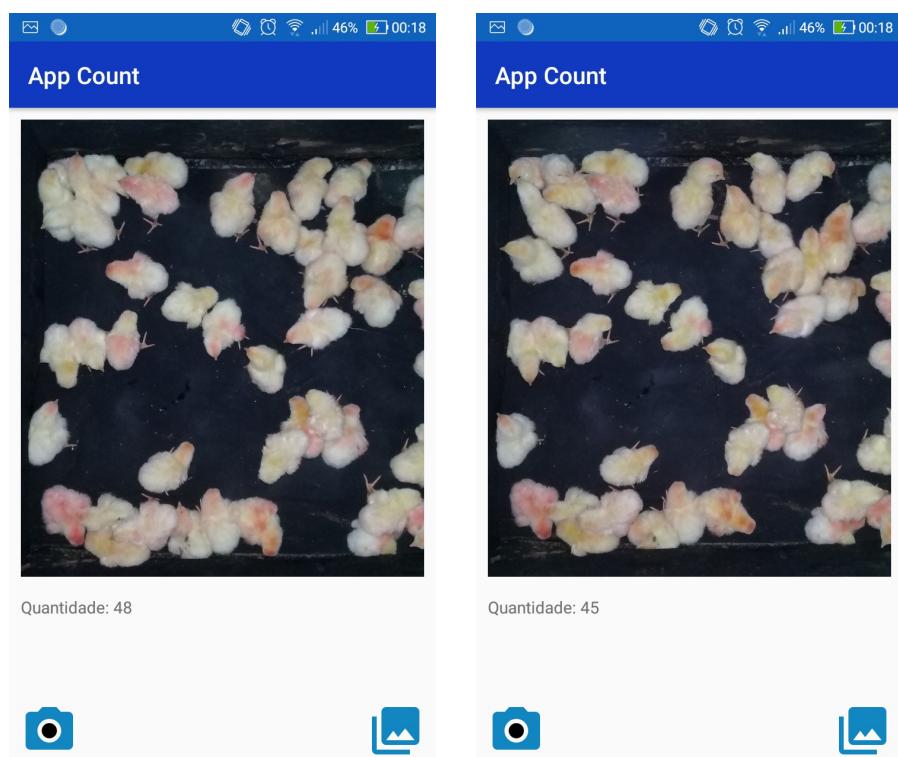


Figura 10 – Imagens de resultados obtidos



Acima nas Figuras 9 e 10 foram apresentadas imagens da tela principal do sistema, nestas imagens pode-se ver os respectivos resultados na contagem de cada imagem com aves, pode-se observar no canto esquerdo inferior o resultado obtido em cada teste apresentado nas figuras.

5.0.5 Considerações finais

Neste capítulo foram apresentados testes realizados com o protótipo assim como resultados obtidos no presente trabalho utilizando recursos de Visão Computacional.

O protótipo desenvolvido apresenta resultados aproximados da quantidade exata de aves presente na imagem, sabendo que por não ter uma ferramenta eficiente para realização desse tipo de contagem neste ambiente o protótipo passa a ser uma possível solução para tal problema, sabendo também que a contagem manual também esta sujeita a erros.

A tarefa de contar aves a partir do protótipo não se torna tão precisa, pois envolve a subjetividade humana, sendo que depende da qualidade da imagem que esta sendo capturada, ou seja o usuário pode posicionar a câmera de maneira incorreta prejudicando a tarefa de detecção do protótipo e o resultado da contagem das aves.

Seria inviável contar esta quantidade de aves de forma manual, sendo ela também sujeita a erros. Contar manualmente necessita de um tempo necessariamente longo e é uma prática exaustiva. Devido a esses fatores os produtores não praticam o tipo de contagem manual das aves.

6 Conclusão

Este trabalho desenvolveu uma técnica para contagem de aves de pequeno porte em avicultura utilizando técnicas de Visão Computacional. Para isso foram utilizados no algoritmo desenvolvido técnicas como: transformações na imagem, segmentação de imagens, transformação de distância e etc.

Os resultados obtidos na contagem das aves foram de 88,52%, obtidos a partir da captura de 50 imagens, sendo submetidas todas essas imagens no sistema para contagem e obteve-se uma margem de erro de 11,48%.

Observou-se a dificuldade de obter 100% de acerto nas detecções, devido a problemas como, estremecimento da câmera e sombra. Outro fator de grande relevância que dificulta para que a contagem seja precisa são as aves estarem juntas na imagens capturadas dificultando na separação dessas aves pelo algoritmo.

6.0.1 Limitações

A tarefa de segmentação de imagens tem grande importância neste trabalho pois objetiva-se separar as aves do fundo da imagem, sendo ela uma tarefa de certa forma complexa devida à diferença de cada imagem capturada. Fatores como estremecimento e reflexos na imagem confundem a segmentação da imagem.

Houve dificuldade em posicionar a câmera por cima da caixa com aves para captura da imagem acabou atrapalhando o resultado da contagem. Devido ao dispositivo móvel estar em mãos acaba estremecendo a imagem no momento da captura. Uma alternativa para este problema seria a construção de um suporte para fixação do dispositivo móvel por cima da caixa, fazendo com que a captura da imagem obtenha maior qualidade.

6.0.2 Contribuições

A principal contribuição deste trabalho é a proposta de uma abordagem automática para contagem de aves em aviários, obtendo uma contagem de até 88,52% das aves em caixas. É uma abordagem que realiza a contagem de forma eficiente e rápida, não precisando executar a contagem de forma manual. E também a contribuição para o estudo de técnicas de detecção de objetos em imagens a partir de técnicas de Visão Computacional, Processamento de Imagens Digitais.

7 Dificuldades encontradas

Devido o fundo das caixas originais serem de cor semelhante a das aves, houve a necessidade de criar uma caixa nas mesmas medições das caixas originais e pinta-la de preto para que a diferença entre o fundo e as aves seja maior, fazendo com que se tenha melhor precisão na detecção e contagem das aves.

Ao longo dos testes foi constatado que quanto mais a aves estiverem espalhadas no momento da captura da imagem, melhor será a eficiência e taxa de acerto na detecção e contagem dessas aves.

Inicialmente foi optado por utilizar Redes Neurais artificiais (RNA) com o objetivo de treiná-la para contagem dessas aves, utilizando o programa NeuroPh Studio, que recebe uma grande quantidade de imagens capturadas dessas caixas com aves, para assim efetuar o treinamento das RNAs.

Porém devido a dificuldade de encontrar conteúdo referente a esse tipo de método, e ao programa NeuroPh Studio optou-se por mudar a metodologia para dar sequencia ao projeto, passando a se utilizar a biblioteca OpenCV juntamento com a IDE Android Studio, utilizando a linguagem Java.

Houve dificuldade em configurar a biblioteca OpenCV dentro do Android Studio, para implementação do algoritmo, e desenvolvimento do protótipo.

Outra dificuldade que afetou os resultados obtidos foi por não ter uma base fixa para o dispositivo móvel na hora de capturar as imagens, fazendo com que algumas imagens saíssem estremecidas afetando o resultado.

7.0.1 Trabalhos futuros

Em relação a abordagem desenvolvida neste trabalho pretende-se estudar formas de integrar o protótipo desenvolvido com Inteligência Artificial, de forma que a cada contagem de aves realizada com o aplicativo, o mesmo aprenda progressivamente. Além do mais, é considerável realizar testes utilizando vídeos substituindo as fotos.

Pretende-se confeccionar algum tipo de suporte para fixação do aparelho móvel por cima da caixa com aves para melhorar na precisão da captura das imagens, sendo esta captura fator de grande relevância para que o protótipo obtenha melhores resultados. Além disso, é importante estudar desenvolver uma caixa de maior tamanho para que as aves possam estar mais espalhadas.

Anexos

A. Código completo do protótipo

Código 7.1 – Código do protótipo

```
public class MainActivity extends AppCompatActivity {  
    ImageView imagem;  
    ImageView btnGaleria;  
    ImageView btnCamera;  
    TextView resultado;  
    Uri uri;  
    private final int GALERIA_IMAGES = 1;  
    private final int PERMISSAO_REQUEST = 2;  
    private final int TIRAR_FOTO = 3;  
    private static final String TAG = "MYAPP::OPENCV";  
  
    private CameraBridgeViewBase mOpenCvCameraView;  
  
    BaseLoaderCallback mCallBack = new BaseLoaderCallback(this){  
        @Override  
        public void onManagerConnected(int status) {  
            super.onManagerConnected(status);  
            switch (status){  
                case BaseLoaderCallback.SUCCESS:  
                {  
                    Log.i(TAG, "OpenCV loaded successfully");  
                    break;  
                }  
                default:  
                {  
                    super.onManagerConnected(status);  
                    break;  
                }  
            }  
        }  
    };  
    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    if(ContextCompat.checkSelfPermission(this,
        Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
        if(ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.READ_EXTERNAL_STORAGE)){
            }else{
                ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                    PERMISSAO_REQUEST);
            }
        }

        if(ContextCompat.checkSelfPermission(this,
            Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED){
            if(ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.CAMERA)){
                    }else{
                        ActivityCompat.requestPermissions(this, new String[]
                            {Manifest.permission.CAMERA}, PERMISSAO_REQUEST);
                    }
        }

        imagem = (ImageView) findViewById(R.id.ivImagem);
        btnGaleria = (ImageView) findViewById(R.id.btnGaleria);
        btnCamera = (ImageView) findViewById(R.id.btnCamera);

        btnCamera.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {

                File diretorio = Environment.
                    getExternalStoragePublicDirectory(
```

```
        Environment.DIRECTORY_PICTURES);  
    File img = new File(diretorio.getPath() +  
        "/" + System.currentTimeMillis() + ".jpg");  
    uri = Uri.fromFile(img);  
  
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    if(intent.resolveActivity(getApplicationContext()) != null) {  
        startActivityForResult(intent, TIRAR_FOTO);  
    }  
});  
  
btnGaleria.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent(Intent.ACTION_PICK,  
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI);  
        startActivityForResult(intent, GALERIA_IMAGES);  
    }  
});  
}  
  
@Override  
protected void onResume() {  
    super.onResume();  
    OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0,  
        this, mCallBack);  
}  
  
@Override  
protected void onActivityResult(int requestCode,  
    int resultCode, @Nullable Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    if(resultCode == RESULT_OK && requestCode == GALERIA_IMAGES){  
        Uri selected = data.getData();  
        String[] filePath = {MediaStore.Images.Media.DATA};  
        Cursor c = getContentResolver().query(selected,  
            filePath, null, null, null);
```

```
    c.moveToFirst();
    int columnIndex = c.getColumnIndex(filePath[0]);
    String picturePath = c.getString(columnIndex);
    c.close();

Bitmap imagemGaleria = (BitmapFactory.decodeFile(picturePath));

    Bitmap imagemResult = bordas(imagemGaleria);

    imagem.setImageBitmap(imagemResult);
} else if (resultCode == RESULT_OK && requestCode == TIRAR_FOTO){

    Bitmap bitmap = (Bitmap) extras.get("data");
    imagem.setImageBitmap(bordas(bitmap));
}

@Override
public void onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode,
            permissions, grantResults);
    if (requestCode == GALERIA_IMAGES){
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){

    } else{

    }
    return;
}
}

public Bitmap bordas(Bitmap img){
    Mat src = new Mat();
    Utils.bitmapToMat(img, src);

    Mat hierarchy = new Mat();
```

```
List<MatOfPoint> contornos = new ArrayList<MatOfPoint>();  
  
Mat cannyEdges = new Mat();  
Mat bin = new Mat();  
Imgproc.cvtColor(src, cannyEdges, Imgproc.COLOR_BGR2GRAY);  
  
Imgproc.threshold(cannyEdges, bin, 53, 255,  
Imgproc.CV_THRESH_BINARY);  
  
Mat dist = new Mat();  
Imgproc.distanceTransform(bin, dist, Imgproc.CV_DIST_L2, 3);  
  
Core.normalize(dist, dist, 0, 1, Core.NORM_MINMAX);  
  
Imgproc.threshold(dist, dist, .5, 1,  
Imgproc.CV_THRESH_BINARY);  
  
int sizeErosao = 2;  
  
Mat kernel = Imgproc.getStructuringElement(  
Imgproc.MORPH_CROSS,  
new Size(2 * sizeErosao + 1, 2 * sizeErosao + 1),  
new Point(sizeErosao, sizeErosao));  
  
Imgproc.erode(dist, dist, kernel);  
  
Mat dist_8u = new Mat();  
  
dist.convertTo(dist_8u, CvType.CV_8U);  
  
Mat canny_outPut = new Mat();  
  
Imgproc.Canny(dist_8u, canny_outPut, 0, 0);  
  
Imgproc.findContours(dist_8u, contornos, hierarchy,  
Imgproc.RETR_TREE, Imgproc.CHAIN_APPROX_NONE);  
  
int areas_validas = 0;
```

```
ArrayList<Double> areas = new ArrayList<>();
for (MatOfPoint contorno : contornos) {
    if (Imgproc.contourArea(contorno) > 0) {
        Rect c = Imgproc.boundingRect(contorno);
        areas.add(c.area());
    }
}

for (Double area : areas) {
    double areaValida = area / 30000;
    System.out.println("rea □ Retangulo" + areaValida);
    if (areaValida > 1) {
        areas_validas = areas_validas + (int) areaValida;
    } else {
        areas_validas++;
    }
}

resultado = (TextView) findViewById(R.id.qtdId);

if (areas_validas != 0) {
    resultado.setText("Quantidade:" + areas_validas);
}

Utils.matToBitmap(src, img);

return img;
}

public static Bitmap rotateBitmap(Bitmap bitmap,
    int orientation){
```

```
Matrix matrix = new Matrix();
switch (orientation) {
    case ExifInterface.ORIENTATION_NORMAL:
        return bitmap;
    case ExifInterface.ORIENTATION_FLIP_HORIZONTAL:
        matrix.setScale(-1, 1);
        break;
    case ExifInterface.ORIENTATION_ROTATE_180:
        matrix.setRotate(180);
        break;
    case ExifInterface.ORIENTATION_FLIP_VERTICAL:
        matrix.setRotate(180);
        matrix.postScale(-1, 1);
        break;
    case ExifInterface.ORIENTATION_TRANSPOSE:
        matrix.setRotate(90);
        matrix.postScale(-1, 1);
        break;
    case ExifInterface.ORIENTATION_ROTATE_90:
        matrix.setRotate(90);
        break;
    case ExifInterface.ORIENTATION_TRANSVERSE:
        matrix.setRotate(-90);
        matrix.postScale(-1, 1);
        break;
    case ExifInterface.ORIENTATION_ROTATE_270:
        matrix.setRotate(-90);
        break;
    default:
        return bitmap;
}
try {
    Bitmap bmRotated = Bitmap.createBitmap(bitmap, 0,
    0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);
    bitmap.recycle();
    return bmRotated;
}
catch (OutOfMemoryError e) {
    e.printStackTrace();
```

```
    return null;  
}  
}  
}
```

Referências Bibliográficas

- BALAN, A. G. R. *Técnicas de segmentação de imagens aéreas para contagem de população de aves*. Tese (Doutorado) — Universidade de São Paulo, 2003.
- BALLARD, D. H. D. H.; BROWN, C. M. *Computer vision*. [S.l.], 1982.
- BARRETO, J. M. Introduçaoas redes neurais artificiais. In: *V Escola Regional de Informática. Sociedade Brasileira de Computação, Regional Sul, Santa Maria, Florianópolis, Maringá*, p. 5–10, 2002.
- BOTELHO, G. M. *Segmentação de imagens baseada em redes complexas e superpixels: uma aplicação ao censo de aves*. Tese (Doutorado) — Universidade de São Paulo, 2014.
- ESQUEF, I. A.; ALBUQUERQUE, M. P. d.; ALBUQUERQUE, M. P. d. Processamento digital de imagens. *CENTRO BRASILEIRO DE PESQUISAS FÍSICAS-CBPF*, 2003.
- FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999.
- FLORENCIO, R. de B. et al. Processamento de imagens subaquáticas. 2009.
- GONZALEZ, R. C.; WOODS, R. E. Image processing. *Digital image processing*, v. 2, 2007.
- LEMPITSKY, V.; ZISSERMAN, A. Learning to count objects in images. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2010. p. 1324–1332.
- MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, v. 16, n. 1, p. 125–160, 2009.
- MORAIS, E. *Rastreamento e contagem de peixes utilizando filtro preditivo*. Tese (Doutorado) — Dissertação apresentada ao curso de Pós-Graduação da Universidade Federal de Minas Gerais. Belo Horizonte-BH, 2005.
- NEVES, L. A. P.; NETO, H. V.; GONZAGA, A. Avanços em visão computacional. *Omnipax, Curitiba, PR, Brasil*, 2012.
- OSÓRIO, F. S.; BITTENCOURT, J. R.; OSÓRIO, F. S. Sistemas inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens. In: *I WORKSHOP DE INTELIGÊNCIA ARTIFICIAL UNISC-Universidade de Santa Cruz do Sul Departamento de Informática-Junho*. [S.l.: s.n.], 2000.
- PEDRONI, R. U. *Sistema autônomo em FPGA para captura e processamento em tempo real de imagens da pupila*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2011.
- PERELMUTER, G. et al. Reconhecimento de imagens bidimensionais utilizando redes neurais artificiais. *Anais do VIII SIBGRAPI*, p. 197–203, 1995.
- QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *RITA*, v. 13, n. 2, p. 11–42, 2006.

- QUINTA, L. N. B. *Desenvolvimento de um sistema de visão computacional para o controle microbiano em processos de produção de etanol*. Tese (Doutorado) — Universidade Católica Dom Bosco, 2009.
- RIOS, L. R. S. Visão computacional. *Departamento de Ciência da computação-Universidade Federal da Bahia (UFBA) Salvador, Bahia, Brasil*, 2010.
- SANTOS, J. C. Extração de atributos de forma e seleção de atributos usando algoritmos genéticos para classificação de regiões. *Instituto Nacional de Pesquisas Espaciais (INPE)*, 2007.
- SILVA, B. R. D. A. E. Sistema de contagem automática de objetos utilizando processamento digital de imagens em dispositivos móveis. *UERN, Mossoró, RN, Brasil*, 2014.
- SILVA, L. S. D. *Sistema Computacional para Contagem Automática de Pessoas Baseado em Análise de Seqüências de Imagens*. Tese (Doutorado) — CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS, 2008.
- SOUZA, J. A. d. et al. Reconhecimento de padrões usando indexação recursiva. Florianópolis, SC, 1999.
- STAIR, R. M.; REYNOLDS, G. W.; SILVA, F. S. C. da. *Princípios de sistemas de informação: uma abordagem gerencial*. [S.l.: s.n.], 1998.