Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

ABMAEL VILAR BARROS

UAEA – UNIDADE ACADÊMICA DE ENGENHARIA AGRÍCOLA

Campina Grande, Paraíba
Abril de 2022

ABMAEL VILAR BARROS

CÁLCULO AUTOMATIZADO DO CONFORTO DE CAPRINOS UTILIZANDO PROCESSAMENTO DE IMAGENS

Relatório de Estágio Supervisionado submetido à Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Imagens

Orientadora:
Luciana Ribeiro Veloso, Dra.

Campina Grande, Paraíba
Abril de 2022

ABMAEL VILAR BARROS

CÁLCULO AUTOMATIZADO DO CONFORTO DE CAPRINOS UTILIZANDO PROCESSAMENTO DE IMAGENS

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Imagens

Aprovado em ____ / ____ / _____

Profa. Luciana Ribeiro Veloso, Dra.
Universidade Federal de Campina Grande
Orientadora.

Profa. Raquel Aline Araújo Rodrigues Felix, Dra.
Universidade Federal de Campina Grande
Avaliadora.

Dedico este trabalho a meus pais e todos que acreditaram no meu potencial.

Agradecimentos

Agradeço à minha orientadora Professora Luciana Ribeiro Veloso pela orientação e apoio.
Agradeço ao professor José Pinheiro Lopes Neto pela oportunidade do estágio na UAEA.
Agradeço a Tiago Lira por ter me indicado para esse estágio.

Epígrafe

“Às vezes encontramos nosso destino pelo caminho que tomamos para evitá-lo.”

- Hugo, Kung Fu Panda.

Resumo

O presente relatório descreve as principais atividades realizadas pelo estagiário Abmael Vilar Barros, graduando em Engenharia Elétrica pela Universidade Federal de Campina Grande, em sua Unidade de Engenharia Agrícola, no período entre 30 de novembro de 2021 a 07 de abril de 2022, totalizando 663 horas. Durante esse período foram implementados algoritmos que viabilizam a automação do procedimento de cálculo no nível de estresse de caprinos através da análise da dilatação da pupila nos animais.

Palavras-chave: Processamento de Imagens, Classificadores, Transformada de *Hough*

Abstract

This report describes the main activities carried out by intern Abmael Vilar Barros, graduating in Electrical Engineering from the Federal University of Campina Grande, in his Agricultural Engineering Unit, from November 30, 2021 to April 7, 2022, totaling 663 hours. During this period, algorithms were implemented that enable the automation of the calculation procedure on the stress level of goats through the analysis of pupil dilation in the animals.

Keywords: *Image Processing, Classifiers, Hough Transform*

LISTA DE ABREVIATURAS E SIGLAS

UAEA	Unidade Acadêmica de Engenharia Agrícola
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
OPENCV	*Open Source Computer Vision*
PDI	Processamento Digital de Imagens
XML	*Extensible Markup Language*

LISTA DE ILUSTRAÇÕES

LISTA DE EQUAÇÕES

LISTA DE TABELAS

Sumário

1 INTRODUÇÃO

Na Unidade de Engenharia Agrícola são desenvolvidas pesquisas relativas a caprinos usados como fonte de alimentação e reprodução. O estresse calórico resulta em um decréscimo na produção de carne, distúrbios reprodutivos e distúrbios alimentares. Esses processos decorrem em função dos efeitos da temperatura e umidade relativa do ar, radiação solar, vento e intensidade/duração do agente estressor (MARQUES, 2017).

A pesquisa teve como objetivo medir através de imagens digitais a dilatação pupilar de caprinos mestiços Boer + SPRD (sem padrão racial definido) submetidos à diferentes temperaturas em câmara climática, para correlacionar esta variável com as respostas fisiológicas e estabelecer padrões de variação com o estresse térmico sofrido (MARQUES, 2017). Até então, na unidade de engenharia Agrícola da UFCG, todo o processo de tratamento de imagens até a medição das dimensões da pupila para o cálculo do conforto térmico era feito manualmente.

1.1 OBJETIVOS

O estagiário tem como objetivo automatizar o processo do cálculo de conforto térmico de caprinos através das dimensões pupilares em ambiente controlado utilizando processamento de imagens.

2 UNIDADE ACADÊMICA DE ENGENHARIA AGRÍCOLA (UAEA)

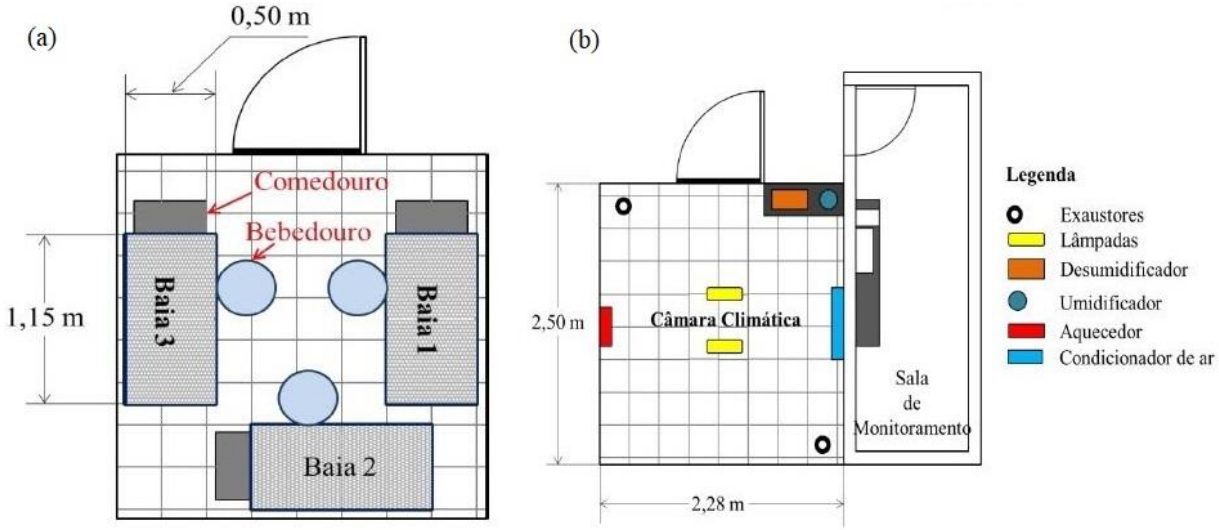
O Departamento de Engenharia Agrícola, hoje Unidade Acadêmica de Engenharia Agrícola (UAEA), foi criado e instalado em 22 de fevereiro de 1976. Oferece atualmente o curso de graduação em Engenharia Agrícola e o Programa de Pós-Graduação em Engenharia Agrícola, com Mestrado e Doutorado (Conceito CAPES: 5).

Dentre os diversos campos de pesquisa realizados na UAEA, a que tem relevância para este trabalho é desenvolvida pelo professor Dr. José Pinheiro Lopes Neto e outros pesquisadores, e consiste no cálculo do nível de conforto de caprinos em relação ao ambiente onde ele se encontra, e a relação com as variáveis ambientais, como temperatura, umidade, luminosidade.

O experimento foi realizado em um laboratório criado na UAEA com condições controladas de luminosidade, temperatura e umidade, de acordo com o comitê de ética em pesquisa protocolo N 284-2015 (MARQUES, 2017).

Seis caprinos mestiços machos foram selecionados para rebanhos reprodutores. Os animais foram dispostos em baias individuais com dimensões de 1,15 m de comprimento, 0,50 m de largura e 0,84 m de altura, como ilustrado na Figura 1 (a), dispostas dentro de uma câmara climática de área igual a 5,70 m^2, ilustrado na Figura 1 (b). Internamente na câmara, foram instalados lâmpadas e umidificadores controláveis externamente, que possibilitam o ajuste de condições de ambiente para o experimento (MARQUES, 2017).

Figura 1 - (a) Baia metálica para o alojamento dos animais; (b) Ilustração da câmara climática para simulação das condições ambientais.



Fonte: Marques 2017.

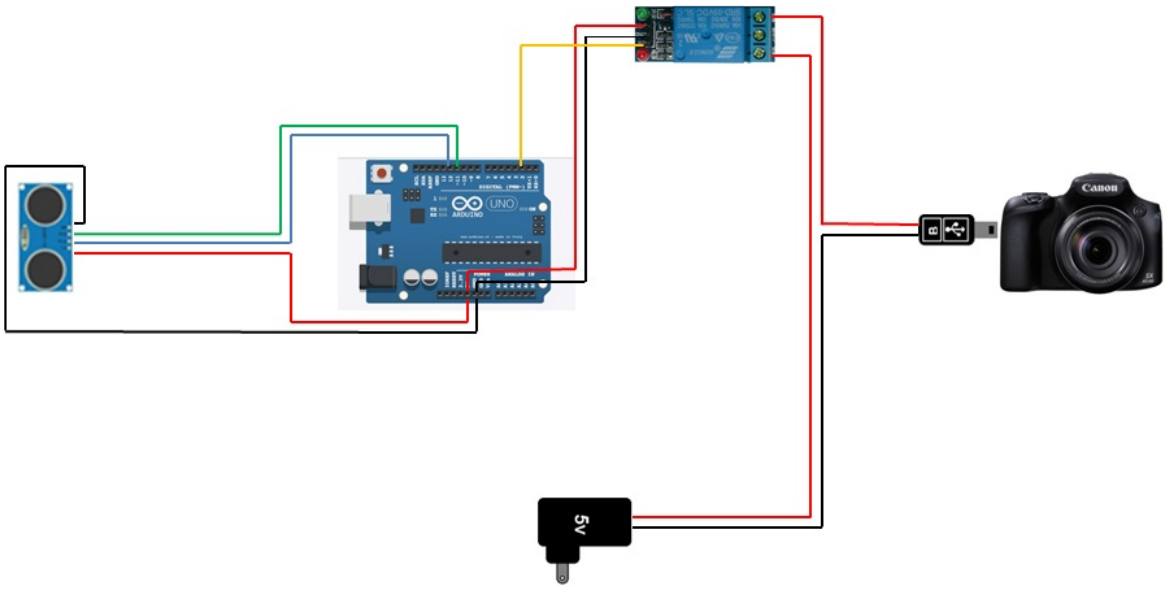
As temperaturas do ar utilizadas na experimentação foram definidas com base na zona de conforto térmico (ZCT) para caprinos mencionada por Souza et al. (2008) que se situa entre 20 e 30 °C com umidade relativa do ar podendo variar entre 50 e 70% (MARQUES, 2017). Com isso, os animais foram submetidos a 3 diferentes temperaturas controladas médias, sendo elas: T1 = 26±0,07 ºC (zona de conforto térmico), T2 = 29,44±0,06 ºC (temperatura limite entre as zonas de conforto e estresse térmico) e T3 = 33,38±0,31ºC (acima da ZCT) com a umidade relativa do ar e velocidade do vento médias de 68±3,89% e 1 m/s, respectivamente (MARQUES, 2017).

Para cada temperatura, foi adotado um período de cinco dias de adaptação ao ambiente controlado, manejo e alimentação e dez dias de coleta de dados (MARQUES, 2017). No início

de cada etapa experimental e entre o final de cada tratamento e início do próximo, os animais ficaram durante cinco dias expostos às condições de temperatura e umidade relativa do ar ambiente (com a câmara aberta) para a recomposição de suas funções fisiológicas (MARQUES, 2017).

A captura das imagens para o monitoramento da dilatação pupilar dos animais ocorreu durante os 10 dias de cada tratamento, no período em que a câmara permanecia fechada. As imagens foram capturadas sem que houvesse o contato direto com os animais e para isso foram montadas estruturas de *hardware*, como ilustrado na Figura 2, sendo esta composta por uma câmera de alta resolução (16 MP), um microcontrolador específico – Arduino UNO, um módulo relé de 5 V, um sensor ultrassônico modelo HC-SR04 e uma fonte/carregador de 5 V de tensão (MARQUES, 2017). As câmeras foram acopladas em tripés e posicionadas de maneira a captar as imagens das pupilas dos animais (MARQUES, 2017).

Figura 2 - Estrutura de *hardware* para aquisição das imagens da pupila dos animais



Fonte: (MARQUES, 2017).

Foi fixada uma escala de referência no chifre dos animais, com intervalos de 5 mm para permitir a conversão das dimensões da pupila que foram coletadas nas imagens em escala de *pixel* e posteriormente convertidas para milímetros em suas reais dimensões (MARQUES, 2017). Antes do procedimento descrito neste documento, todo o processo era feito manualmente. A seguir, serão descritos a fundamentação teórica necessária para a implementação dos algoritmos de processamento de imagem para análises requerida.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 PROCESSAMENTO DIGITAL DE IMAGENS (PDI)

O Processamento Digital de Imagens ou PDI, corresponde ao processamento de imagens digitais por um computador digital. Existem três níveis de processamento de imagens: o nível baixo, no qual a entrada e a saída do sistema são imagens; O nível médio, cuja entrada é uma imagem e a saída é um destaque de características de objetos pertencentes a entrada; O nível alto, onde a entrada é uma imagem e a saída são informações cognitivas relacionadas à visão (GONZALEZ; WOODS, 2010).

3.2 BINARIZAÇÃO DE IMAGENS

A binarização é uma técnica de limiarização em PDI que transforma uma imagem em níveis de cinza, em uma imagem com apenas dois tons, comumente preto e branco, como mostrado na Figura 3. O valor que delimita qual nível de cinza será convertido em preto ou branco, pode ser definido manualmente ou pode ser feita de forma automática.

A limiarização automática pode ser realizada usando um procedimento baseado na função densidade de probabilidade dos níveis de intensidade de cada classe e a probabilidade que cada uma ocorra em uma determinada aplicação (GONZALEZ; WOODS, 2010).

O método de Otsu, é um método que tem por ideia básica de que as classes com limiares bem estabelecidos devem ser distintos em relação aos valores de intensidade de seus *pixels*, e inversamente que um limiar ótimo (GONZALEZ; WOODS, 2010).

Figura 3 - Binarização de imagens.

Fonte: Disponível em https://maalencar.files.wordpress.com/2012/06/014.png

3.3 DETECTOR DE BORDAS DE *CANNY*

O detector de bordas de *Canny* é um algoritmo que destaca pontos de borda em uma imagem. A essência do trabalho de Canny foi expressar matematicamente 3 critérios para uma detecção ótima de bordas, são eles: baixa taxa de erro, pontos de borda bem localizados e que a resposta do procedimento seja um único ponto de borda. Em geral é difícil encontrar uma solução que satisfaça os três critérios simultaneamente (GONZALEZ; WOODS, 2010). O algoritmo de detecção de *Canny* possui 4 passos para a obtenção do resultado.

1. Suavização da imagem
2. Cálculo do gradiente
3. Supressão de não máximos
4. Limiarização por histerese

A suavização da imagem propicia ao algoritmo uma melhor análise de variações de tonalidade relevantes para detecção de bordas. Pode ser obtida com a aplicação de um filtro gaussiano na imagem original, Figura 4 (a), e como resultado obter a Figura 4 (b).

O cálculo do gradiente é realizado por um processo de convolução da imagem filtrada com uma máscara que destaca uma direção preferencial de borda. A supressão de não máximos, descarta os pontos que não são máximos em relação a sua vizinhança na direção do gradiente. Uma vez feita a supressão de não máximos, os pontos que sobram são os candidatos finais a pontos de borda. Nesta etapa o algoritmo percorrerá todos os pontos não nulos da imagem, e manterá o caminho de pontos que estiverem dentro do limiar superior e inferior pré-definidos no algoritmo. O resultado dessas operações está ilustrado na Figura 4 (c).

Figura 4 – Imagem original (a), imagem filtrada (b), detecção de bordas de *Canny* (c).

Fonte: Autoria Própria.

3.4 TRANSFORMADA DE *HOUGH*

A Transformada de *Hough* é uma técnica de PDI para detecção de toda e qualquer forma geométrica, porém se mostra bastante eficiente em detecção de linhas (FELTRIN, 2020) e círculos.

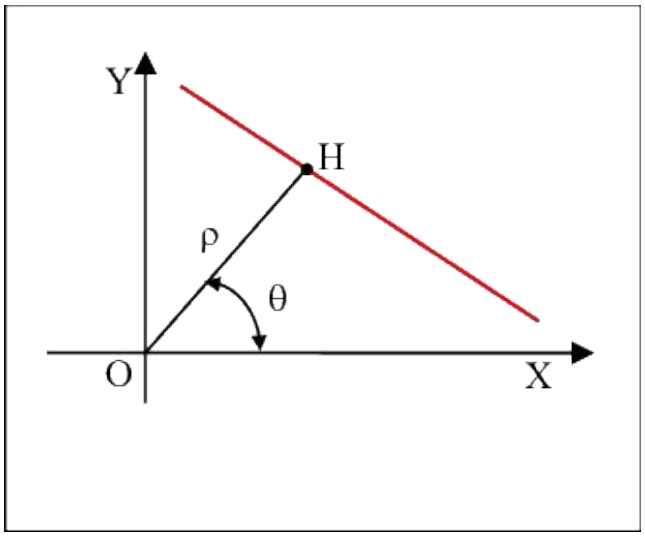
Para a aplicação da Transformada de *Hough* é necessário a extração de bordas na imagem, e a partir desses pontos de borda detectados que se inicia o processo. Para cada ponto de borda, aplica-se suas coordenadas na equação paramétrica da forma geométrica que se deseja detectar. Caso o ponto satisfaça a equação para algum conjunto de parâmetros da forma em questão, um acumulador, chamado de matriz de votação para aqueles parâmetros especificados é incrementado (GONZALEZ; WOODS, 2010).

Ao fim do processo, busca-se na matriz de votação por máximos locais, que indicam quais os parâmetros das formas detectadas devem ser destacados na imagem. Por exemplo, para a detecção de retas em uma imagem, elas são parametrizadas com a Equação (1).

$$x \cos\theta + y \sin\theta = \rho \tag{1}$$

Onde, x e y são as coordenadas dos pontos de borda detectados na imagem, θ e ρ são os parâmetros que caracterizam cada reta que pode ser detectada, como mostrado na Figura 5, definida pelo projetista. Cada vez que um ponto de borda pertencer a alguma combinação θ e ρ, um voto é acumulado na matriz de votação, como mostrado na Figura 6.

Figura 5 – Reta parametrizada.



Fonte: Disponivel em https://www.researchgate.net/figure/Variables-empleadas-en-la-transformada-de-*Hough*_fig4_271248788

Figura 6 – Matriz de votação para detecção de retas.

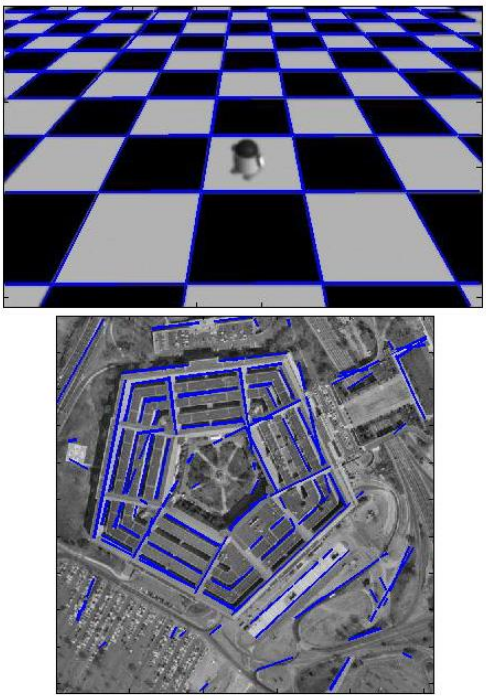
MATRIZ DE VOTAÇÃO

	ρ_1	ρ_2	ρ_3	$\cdots$	ρ_n
θ_1					
θ_2		1			
θ_3	8				
$\vdots$			3		
θ_n					

Fonte: Autoria Própria.

Em seguida, o algoritmo deve procurar na matriz por máximos locais, e então destacar na imagem as retas correspondentes. O resultado do processo pode ser visto na Figura 7.

Figura 7 – Detecção pela transformada de *Hough*.

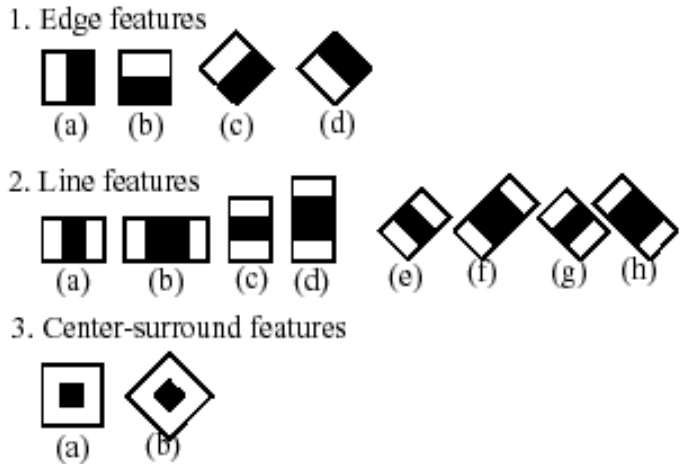


Fonte: Disponivel em https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/9226/versions/1/screenshot.jpg

3.5 CLASSIFICADOR *HAAR CASCADE*

Esse algoritmo de detecção de objetos é muito usado na detecção de faces em imagens ou vídeos em tempo real. O algoritmo usa um conjunto de filtros, alguns ilustrados na Figura 8 para identificação de padrões na imagem durante o processamento, de modo a tentar categorizar características de cada tipo de objeto na imagem.

Figura 8 - Conjunto de filtros do *Haar Cascade*.



Fonte: Disponível em https://www.researchgate.net/figure/Types-of-Haar-features_fig1_326181724

A extração de características é realizada a partir de um cálculo, que computa separadamente os *pixels* da imagem questão sob a região branca do filtro e os *pixels* da imagem que estão sob a região preta do filtro. Esse resultado é utilizado para definir o quão próximo aquela região de *pixels* na imagem está similar ao filtro de características. Na Figura 9 (a) temos um filtro detector de borda e na Figura 9 (b) uma região de *pixels* da imagem.

Figura 9 – Matriz de *pixels* do filtro (a), matriz de *pixels* da imagem (b).

(a)

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

(b)

0,1	0,2	0,6	0,8
0,2	0,3	0,8	0,6
0,2	0,1	0,6	0,8
0,2	0,1	0,8	0,9

Fonte: Autoria Própria.

Para calcular a similaridade, primeiro calcula-se a média dos valores dos *pixels* da imagem correspondentes a região de *pixels* brancos do filtro, pela Equação (2).

$$w = \frac{1}{n} \sum_{white}^{n} I(x) \tag{2}$$

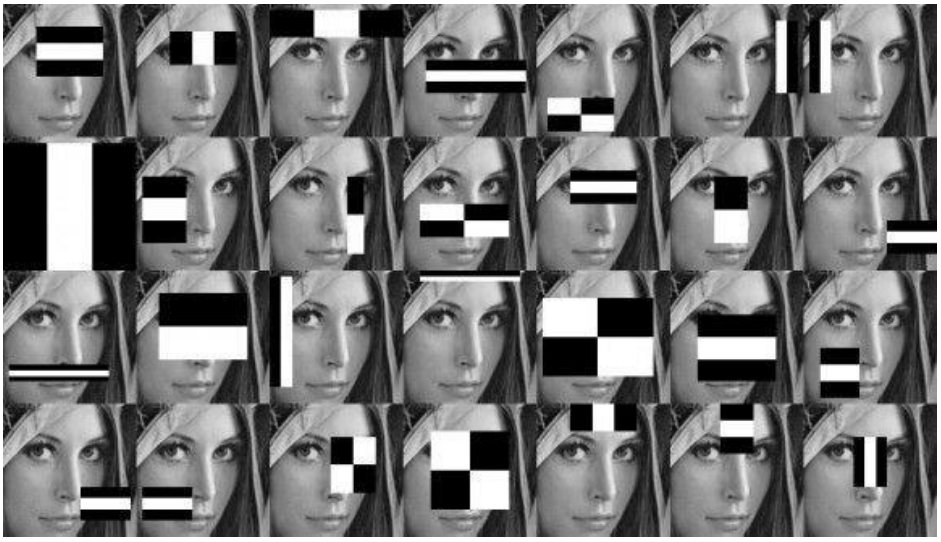
Em seguida, calcula-se a média dos valores dos *pixels* da imagem correspondentes a região de *pixels* pretos do filtro, pela Equação (3), e por fim calcula-se a diferença entre as médias pela Equação (4).

$$b = \frac{1}{n} \sum_{black}^{n} I(x) \tag{3}$$

$$\Delta = b - w \tag{4}$$

Quanto mais próximo de 1 for o resultado Δ, significa que foi encontrado um padrão mais próximo a característica. O procedimento anteriormente demonstrado é realizado para todos os filtros de detecção que varrem a imagem completa, como ilustrado na Figura 10.

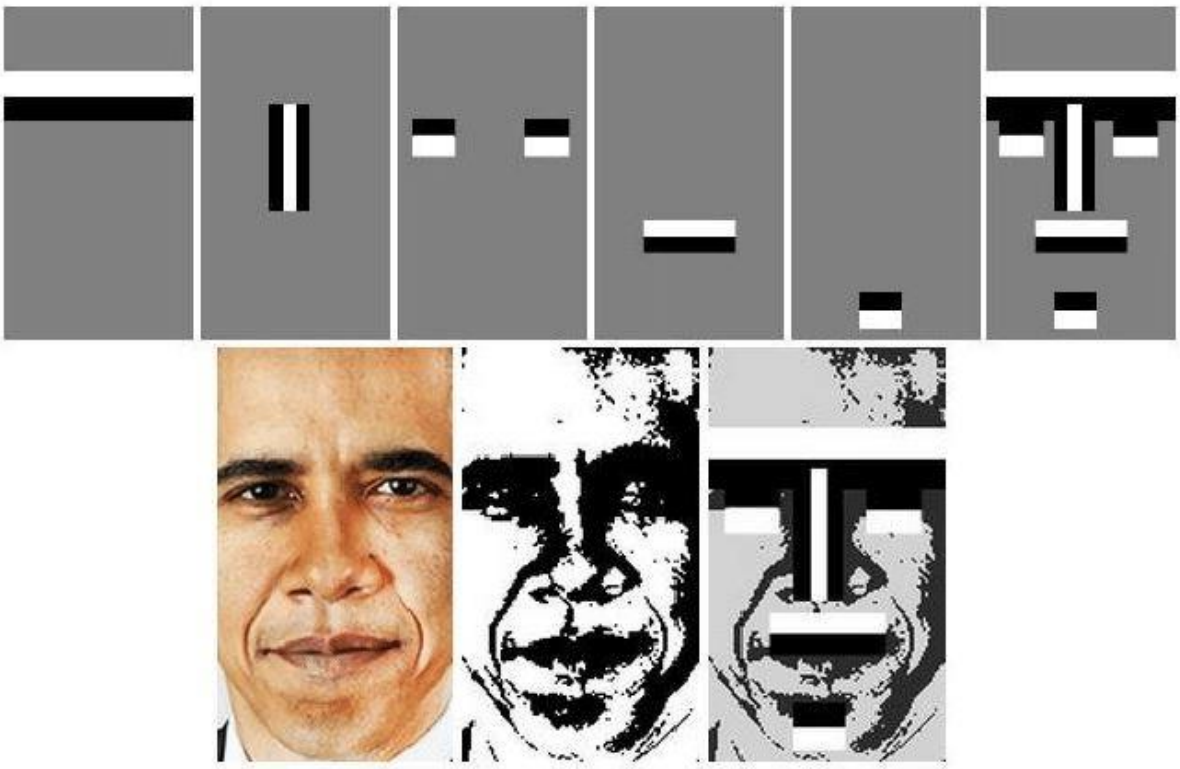
Figura 10 - Uma representação de um classificador *Haar Cascade* em treinamento.



Fonte: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

Após a passagem de todos os filtros, pode ser detectado um padrão similar ao padrão da característica identificada no treinamento, caso ocorra o *match* a região em questão é destacada pelo classificador, como mostrado na Figura 11.

Figura 11 – Detecção do classificador.



Fonte: Disponível em https://levelup.gitconnected.com/haar-like-features-seeing-in-black-and-white-1a240caaf1e3

3.6 LINGUAGEM *PYTHON*

Python é uma linguagem de programação interpretada e de alto nível, ou seja, roda executando uma instrução por vez (WES MCKINNEY, 2018). Para seu uso, é necessário o uso de alguma ferramenta de interpretação como o Anaconda, que foi o interpretador utilizado neste trabalho. Essa é uma das linguagens mais utilizadas em todo mundo e caracteriza-se por ser uma linguagem de baixa complexidade de programação (HALTERMAN, 2011).

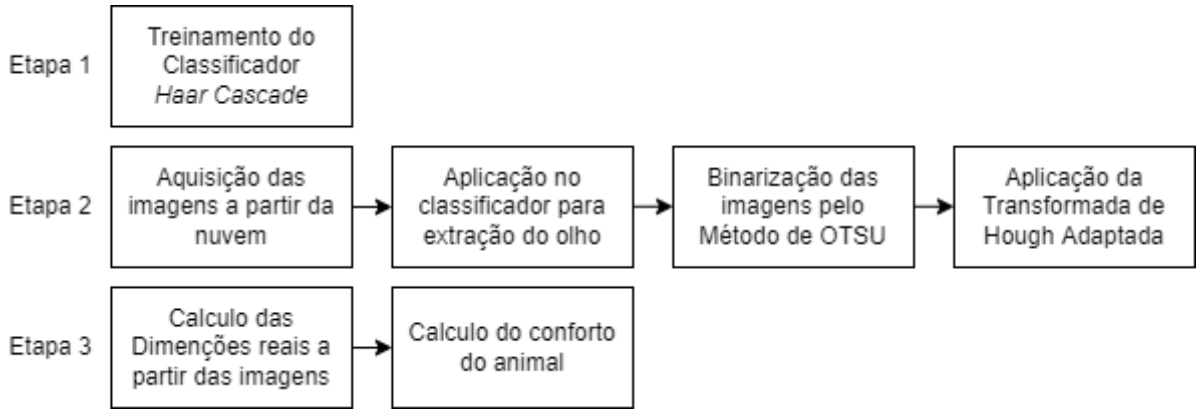
3.7 *OPENCV*

A *Open Source Computer Vision* ou *OpenCV* é uma biblioteca de código aberto criada e disponível para várias linguagens de programação, que permite a manipulação de imagens de forma matricial, além de disponibilizar a implementação de diversas técnicas de PDI, e o uso de redes neurais.

4 ATIVIDADES DESENVOLVIDAS

Durante o período vigente do estágio, foram desempenhadas atividades relacionadas à implantação dos algoritmos da automatização do sistema automático em linguagem *Python*. As atividades foram divididas de acordo com o fluxograma da Figura 12.

Figura 12 - Fluxograma de Atividades



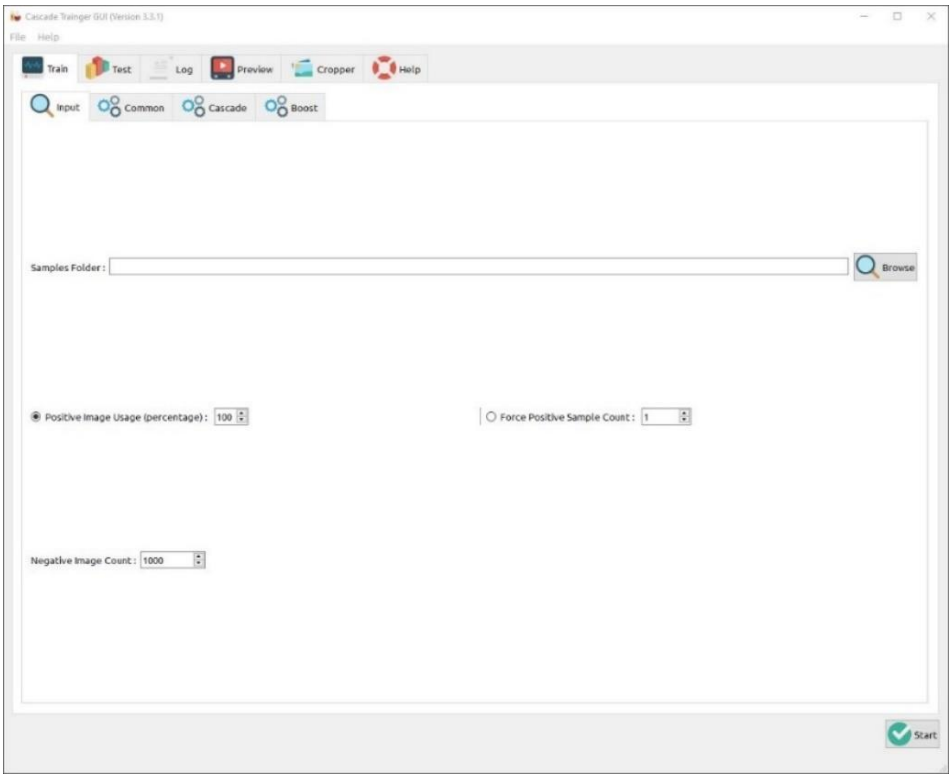
Fonte: Autoria própria.

4.1 ETAPA 1

O primeiro passo desenvolvido, foi o treinamento do classificador *Haar Cascade*, para que identificasse o olho do animal. Esse algoritmo foi treinado utilizando dois bancos de imagens, um contendo a característica que se quer detectar, e outro conjunto que não tenha a referida característica. Foi utilizado o software *Cascade Trainger GUI (Version 3.3.1)* para realizar o treinamento sem a necessidade de desenvolver um código para tal função.

Na Figura 13 pode-se observar a tela inicial de configuração do software, onde define-se a pasta com as imagens para treinamento no campo *Samples Folder*. Na parte inferior, há dois indicadores onde deve-se indicar o percentual de imagens a ser utilizado que contém a característica (*Positive Image Usage*), e o número de imagens que não contém a característica *(Negative Image Count)*.

Figura 13 - *Cascade Traigner GUI* – Página de Configuração *input*

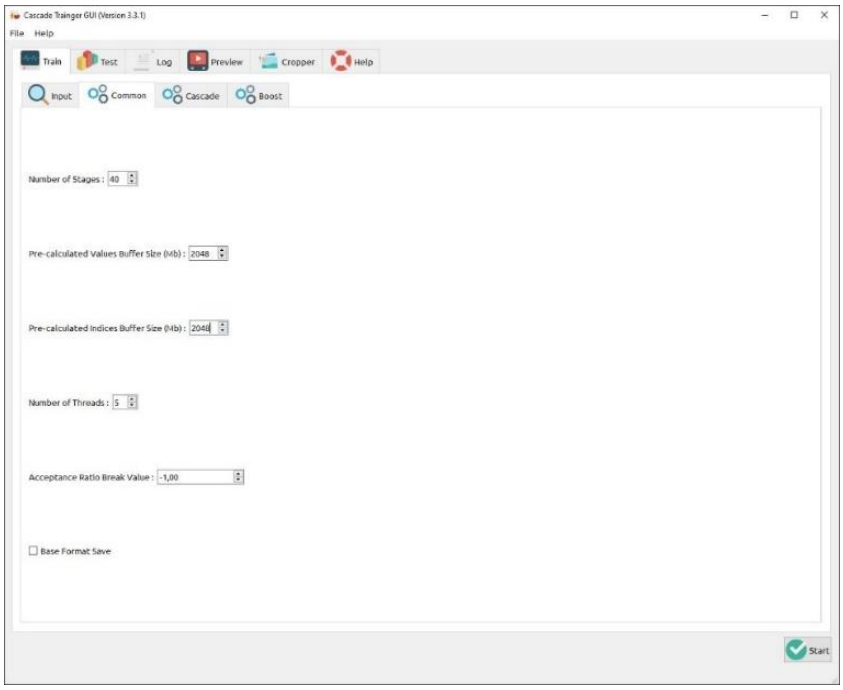


Fonte: Autoria Própria.

A pasta indicada no *Sample Folder*, precisa conter duas subpastas com os nomes “p” onde se coloca as imagens positivas, e “n” onde se coloca as imagens negativas. Foram utilizadas 50 imagens aleatórias de olhos de caprinos, retiradas manualmente das 377 fotos disponibilizadas. Essas imagens foram recortadas em um tamanho 256 x 256 *pixels* e com o centro da pupila coincidindo com o centro da imagem, para propiciar um melhor treinamento e consequentemente uma melhor detecção. Para imagens negativas, foram utilizadas 516 imagens retiradas da internet que não continham olhos de nenhuma espécie. Para as configurações, foi necessário o uso de 96% das fotos positivas, pois um valor maior gerava um erro não explicável no treinamento. Após uma pesquisa no site do desenvolvedor do aplicativo, ele dizia que não sabia por que o erro acontecia, mas indicava a solução de reduzir o percentual de imagens positivas até funcionar o treinamento.

Na Figura 14, pode-se observar a segunda página de configurações, onde se define as mais relevantes sendo o número de épocas de treinamento *(Number of Stages)*, configurado para 40 épocas e a quantidade de memória disponibilizada configurado para 2048 MB.

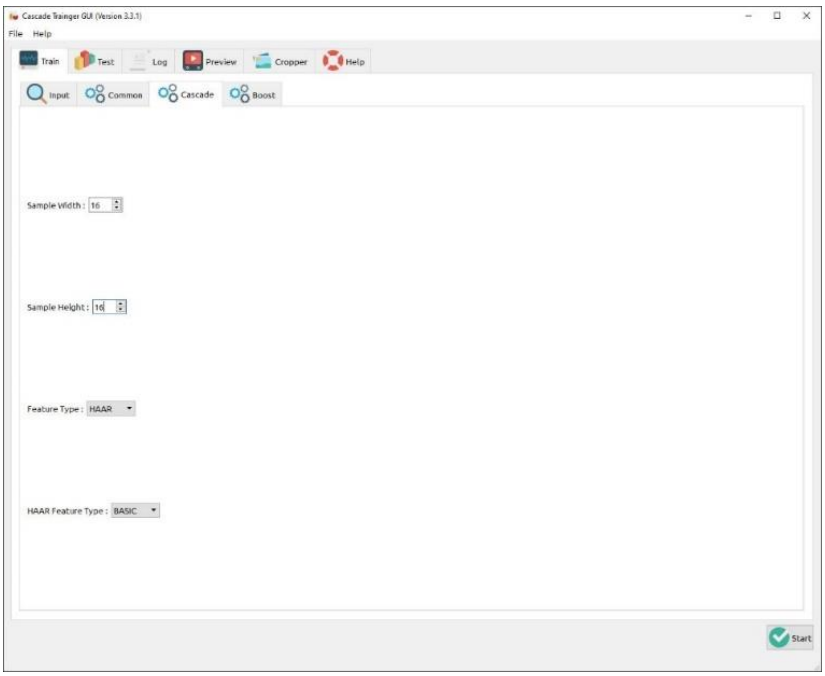
Figura 14 - *Cascade Traigner GUI* – Página de Configuração *commom*



Fonte: Autoria Própria.

Na Figura 15, pode-se observar a terceira página de configurações, onde a mais relevante é a determinação do tamanho dos filtros *(Sample Width e Sample Height)*, já que o *software* por padrão, vem definido para treinar um classificador *Haar cascade*. Foi escolhido os filtros de tamanho 16 x 16 *pixels*.

Figura 15 - *Cascade Traigner GUI* – Página de Configuração *cascade*



Fonte: Autoria Própria.

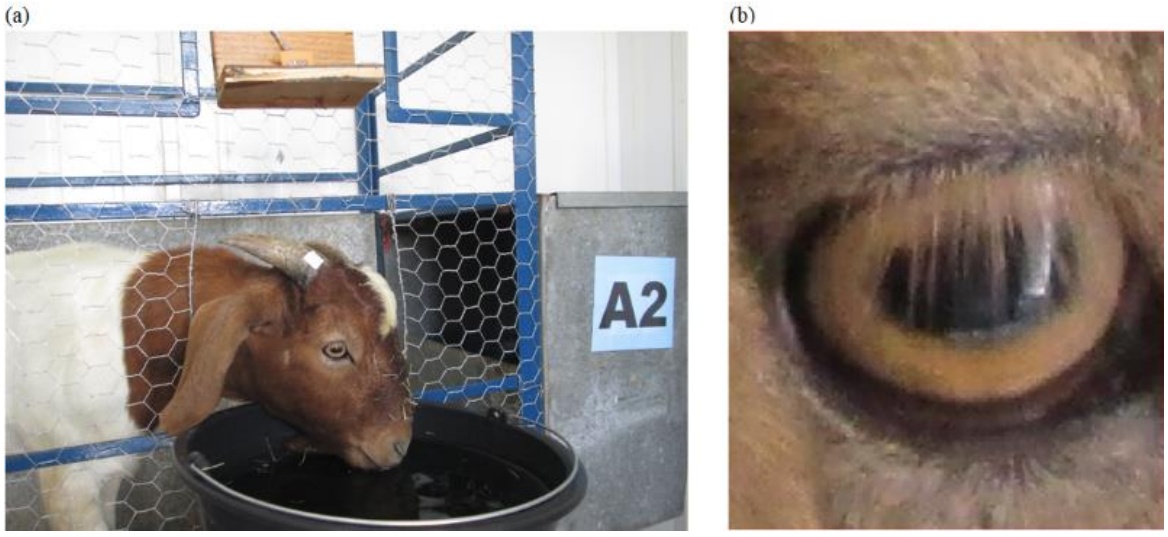
O resultado do treinamento é um arquivo de extensão XML que é o classificador treinado, e pode ser chamado a partir de funções do *OpenCV* na etapa de código.

4.2 ETAPA 2

4.2.1 Aplicação do Classificador para Extração do Olho

As imagens originais, Figura 16 (a), são enviadas para o classificador para a detecção e localização da característica desejada, resultando na imagem ilustrada na Figura 16 (b).

Figura 16 – Imagem de entrada do classificador (a), imagem de saída do classificador (b).



Fonte: Autoria Própria.

4.2.2 Binarização pelo Método de Otsu

Com a posse das imagens extraídas pelo classificador, foi desenvolvido um trecho de código para aplicar uma binarização em cada extração, e gerar um novo conjunto de imagens de trabalho para a próxima etapa. A decisão pela binarização foi tomada para melhorar o resultado do detector de bordas, que é uma das etapas necessárias para aplicação da Transformada de *Hough*, anteriormente mencionada.

A binarização pelo método de OTSU foi escolhida pelo fato de abranger um resultado melhor para as diversas fotos, que tinham níveis de iluminação muito distintos.

4.2.3 Aplicação da Transformada de *Hough* Adaptada

Na última fase desta etapa, foi desenvolvido uma forma adaptada para a aplicação da transformada de *Hough*, uma vez que o formato geométrico a ser buscado na imagem é o elíptico, que é o formato mais próximo da pupila do animal e a biblioteca do *OpenCV* não tem uma implementação pronta para este formato.

Foram enfrentadas algumas dificuldades para desenvolver de maneira fiel ao procedimento teórico. O primeiro desafio foi conseguir pontos de contorno da pupila a partir da imagem binarizada. Para a transformada de *Hough* teórica é necessário aplicar um algoritmo de detecção de bordas inicialmente. Porém, ao aplicar o algoritmo de *Canny*, o resultado esperado para a transformada não foi satisfatório, devido a representação de pontos relacionados aos cílios do animal por cima do olho do animal.

Outra dificuldade encontrada, foi a implementação da equação paramétrica relativa à elipse. A Equação (5), representa uma elipse em qualquer posição no plano.

$$\frac{\big((x - x_c)\cos\alpha + (y - y_c)\sin\alpha\big)^2}{a^2} + \frac{\big((x - x_c)\sin\alpha - (y - y_c)\cos\alpha\big)^2}{b^2} = 1 \qquad (5)$$

Onde, x_c e y_c são as coordenadas do centro da elipse, a e b são respectivamente os semieixos maior e menor, e α é o ângulo de rotação da elipse, totalizando cinco parâmetros. Ao implementar a Equação (5) e aplicar na imagem, as elipses desenhadas detinham características não condizentes com o esperado.

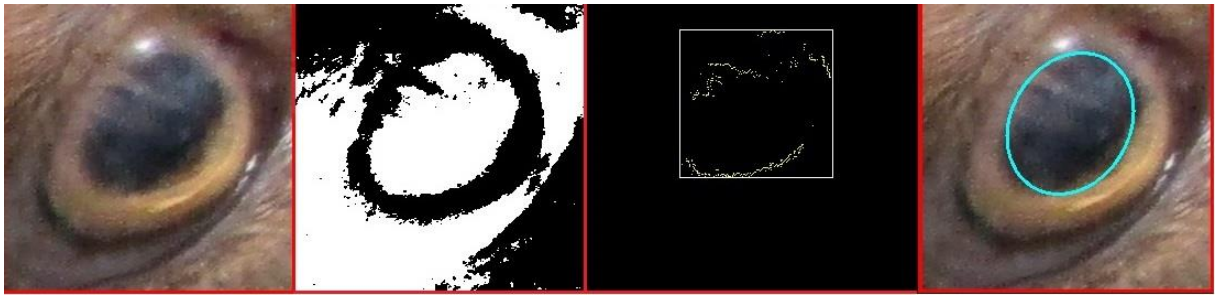
Então para solucionar o problema, optou-se por usar a função de desenho de elipse já implementada pelo *OpenCV*, que geravam desenhos mais condizentes com os parâmetros da forma. No entanto, foi necessário adaptar o procedimento de aplicação da transformada de *Hough*.

A transformada de *Hough* descreve que, uma vez escolhido o conjunto de todos os parâmetros que definem a forma que se deseja detectar, cria-se uma matriz de votação, que no caso da elipse é penta dimensional, e seria necessário inserir cada ponto de borda detectado na Equação (5). Porém, foi optado o uso de uma binarização em detrimento de uma detecção de borda, era necessário um algoritmo para definir os pontos que seriam aplicados. Sendo assim, foi desenvolvido um algoritmo próprio para detectar os pontos que delimitam a pupila a partir da imagem binarizada.

O algoritmo desenvolvido, a partir do pixel central da imagem, varre todos os pixels em 4 direções até encontrar uma borda. Após a detecção, esses pontos são marcados de amarelo e o *pixel* central é deslocado para o lado repetindo o processo. Ao término do processamento do algoritmo, uma imagem com os pontos relativos a pupila é gerada.

Após a detecção dos pontos pertencentes à pupila, a matriz de votação é inicializada com zero em todas as posições, e o algoritmo contabiliza a intersecção entre os pontos pertencentes à pupila e os pontos de cada elipse desenhada. O número de interseções entre esses pontos, corresponde ao número de votos que cada combinação de parâmetros recebeu. Finalizado o processo, é feita uma busca pelo conjunto de parâmetros que recebeu a maior quantidade de votos, indicando que é a elipse que melhor representa o formato da pupila, como mostrado na Figura 17.

Figura 17 - Processo de binarização e aplicação da transformada de *Hough* Adaptada



Fonte: Autoria Própria.

4.3 ETAPA 3

Nesta etapa, utilizou-se a imagem de saída da etapa anterior, que corresponde ao olho do animal com a marcação da pupila que foi detectada.

4.3.1 Cálculo das Dimensões Reais a partir da Imagem

Para o cálculo de comprimentos em uma imagem, é necessário que ela tenha uma referência fixa de tamanho pré-determinado. Em relação à forma original que era feita, um papel de 15 x 15 mm subdividido em 3 partes foi fixado no chifre de cada animal, como visto na Figura 18.

Figura 18 – Imagem do papel de referência.

Fonte: (MARQUES, 2017).

Para uma abordagem de sistema automatizado, é necessária uma referência que se mantenha fixa ao chifre, que seja fortemente fixa impedindo movimentos de rotação ou translação, e que possua um formato que permita uma boa detecção de contorno. Como o experimento com os animais das fotos tratadas já tinha sido encerrado, foi realizado uma simulação para demonstrar o funcionamento dessa parte do algoritmo.

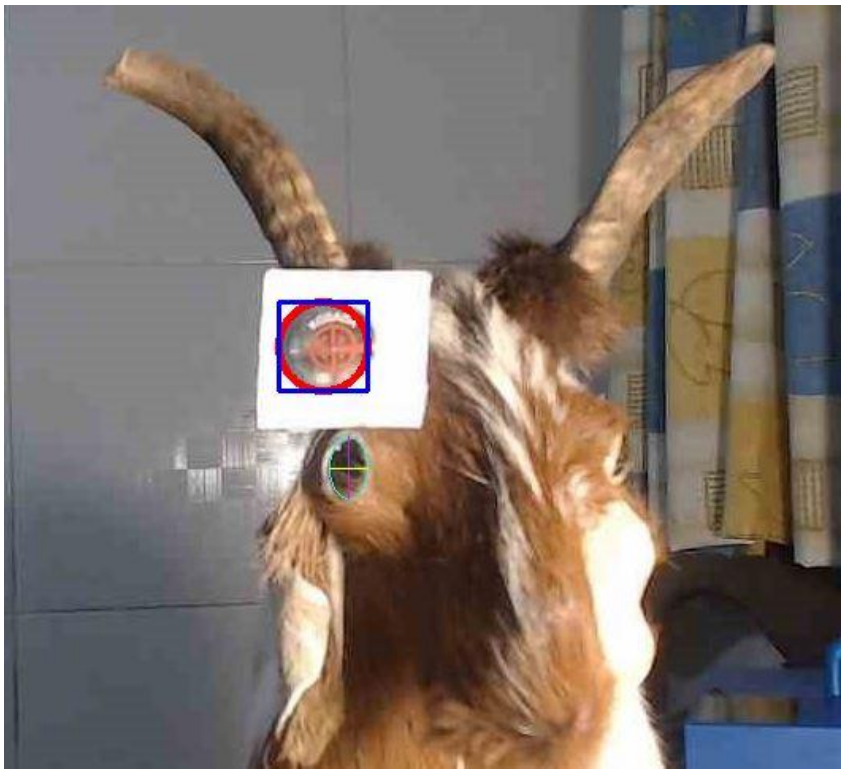
O experimento foi realizado utilizando uma estátua de cabra e uma *webcam* para simular a situação do experimento original. A câmera foi fixada por um tripé e posicionada à 30 cm da cabeça do animal. Para a referência fixa foi utilizado uma peça circular de tabuleiro com um diâmetro de 19,6 mm colado por cima de um pedaço de papelão rígido branco, como ilustrado na Figura 19.

Figura 19 – Experimento simulado

Fonte: Autoria Própria.

Partindo do pressuposto que a referência está no mesmo plano do olho, a próxima etapa desenvolvida foi realizar a detecção da forma de referência, que neste caso é o círculo. Para isso foi utilizado a transformada de *Hough* para círculos, que já possui implementação nativa na biblioteca do *OpenCV*. Após a detecção do círculo, é necessário desenhar uma caixa circunscrita sobre ele, como ilustrado na Figura 20, que será usado para a medição de *pixels*.

Figura 20 – Detecção de formas



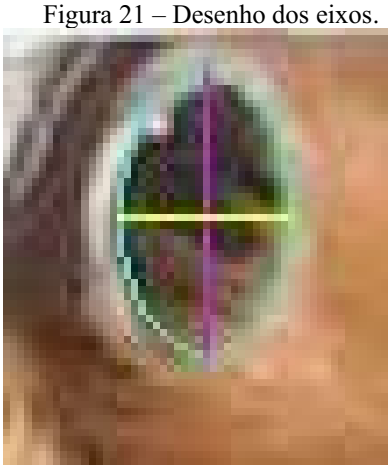
Fonte: Autoria própria.

Agora o algoritmo contou quantos *pixels* a caixa delimitadora tem na horizontal e na vertical e de posse desse valor, pode-se para calcular a relação *pixel*/mm para ambas as direções de acordo com a Equação (6).

$$pixel/mm = \frac{comprimento\ do\ objeto\ em\ pixel}{comprimento\ do\ objeto\ em\ mm} \quad (6)$$

De posse do valor de $pixel/mm$, deve-se calcular o comprimento dos eixos maior e menor da elipse que marca a pupila, pela distância euclidiana. Para tal é necessário desenhar os semieixos na como ilustrado na Figura 21.

Figura 21 – Desenho dos eixos.



Fonte: Autoria Própria.

De posse das dimensões euclidianas dos eixos, os valores das dimensões da imagem em milímetros foram calculados pela Equação (7).

$$dim = \frac{distância\ euclidiana\ em\ pixel}{pixel/mm} \tag{7}$$

Para o experimento simulado, os valores de eixo maior e menor medem 40 e 25 u.d., respectivamente. Utilizando a Equação (7), as dimensões maior e menor resultaram nos valores 13,9 mm e 8,7 mm, respectivamente, que estão compatíveis com as medidas realizadas com o paquímetro, medindo 13,8 mm e 8,5 mm, como ilustrada na Figura 22 (a) e (b).

Figura 22 – Medição com paquímetro da dimensão menor (a), e dimensão maior (b).

(a) (b)

Fonte: Autoria Própria.

4.3.2 Cálculo do Conforto do Animal

Uma vez calculadas as dimensões dos eixos em milímetros, foi implementado o cálculo do conforto do animal no ambiente em relação a variação de temperatura, de acordo com a pesquisa. Primeiramente foi calculado a área pupilar AP, como descrito na Equação (8).

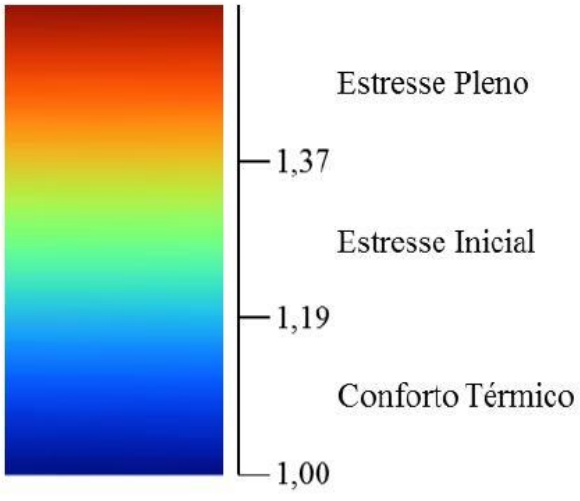
$$Ap = a \cdot b \cdot \pi \; mm^2 \tag{8}$$

Com o valor obtido de 94,98 mm^2 de área pupilar, o algoritmo desenvolvido calcula a Razão de Estresse Pupilar ou REP, de acordo com a Equação (9).

$$REP = \frac{Ap}{71{,}88} \tag{9}$$

Onde, 71,88 mm^2 corresponde a área pupilar do animal em condição de conforto térmico (MARQUES, 2017), o valor obtido no experimento simulado foi de 1,32. De acordo com a escala de estresse indicado na Figura 23, o animal estaria em uma situação de estresse inicial.

Figura 23 – Escala de estresse



Fonte: (MARQUES, 2017)

5 RESULTADOS E DISCUSSÕES

Para avaliar o desempenho do algoritmo desenvolvido, foram utilizadas duas métricas de classificação, a acurácia e o *F1-Score*. Foram utilizadas 377 imagens de caprinos no *dataset*, gerando 355 imagens de detecção após a classificação. Isso indica que o classificador não obteve detecção em todas as imagens do *dataset*. As imagens finais foram divididas em quatro grupos definidos de arquivos:

- **Verdadeiros Positivos (VP):** Número de olhos classificados como olho.
- **Falsos Positivos (FP):** Número de não olhos classificados como olho.
- **Verdadeiros Negativos (VN):** Número de não olhos classificados como não olho.
- **Falsos Negativos (FN):** Número de olhos classificado como não olho.

A partir dessas definições é possível montar uma matriz com todas as combinações denominada de matriz de confusão, que é comumente utilizada para melhor visualização nos algoritmos de classificação, como ilustrado na Figura 24.

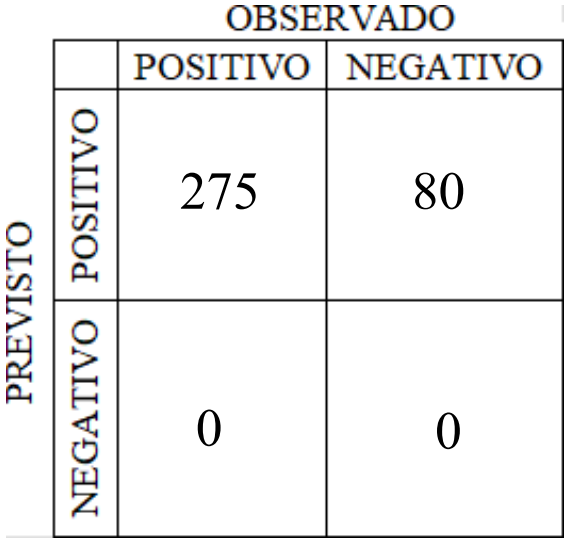
Figura 24 – Matriz de confusão.

		OBSERVADO	
		POSITIVO	NEGATIVO
PREVISTO	POSITIVO	VP	FP
	NEGATIVO	FN	VN

Fonte: Autoria Própria.

Com base nos dados obtidos com o experimento, é possível montar a matriz de confusão, como ilustrado na Figura 25, e calcular a acurácia e o *F1-Score*.

Figura 25 – Matriz de confusão.



Fonte: Autoria Própria.

Com base nos valores da matriz, a acurácia é obtida pela Equação (10). Os dados e resultados relativos a este experimento são demonstrados na Tabela 1.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{10}$$

Tabela 1 - Resultados de acurácia.

Classe	VP+VN	VP+VN+FP+FN	Resultado (%)
Olho Detectado	275	355	77

Fonte: Autoria Própria.

Foi realizado uma segunda métrica de avaliação chamada de *F1-Score*. Essa métrica consiste no cálculo de uma média harmônica entre fatores de métricas mais precisas definidas a seguir:

- **Precisão (P):** É a proporção de verdadeiros positivos classificados como positivos. Ou seja, dentre todas classificadas como positivas, essa métrica avalia quantas são realmente positivas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$P = \frac{VP}{VP+FP}, 0 \leq P \leq 1 \tag{11}$$

- **Revocação (R):** É o número de positivos classificados corretamente em relação ao total de positivos. Dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$R = \frac{VP}{VP+FN}, 0 \leq R \leq 1 \quad (12)$$

O *F1-Score* é calculado combinando os resultados das Equações (11) e (12) e inserindo-os na Equação (13).

$$F1\ Score = \frac{2 \cdot P \cdot R}{P + R}, 0 \leq F1\ Score \leq 1 \quad (13)$$

Com base nos valores das matrizes, os valores de *F1-Score* para cada classe e compará-los com o método de acurácia, indicados na Tabela 2.

Tabela 2 – Resultados comparativos entre Acurácia e *F1-Score*

Classe	Acurácia	*F1-Score*
Olho Detectado	77%	87%

Fonte: Autoria Própria.

Em relação ao número de pupilas corretamente marcadas tem-se os seguintes valores:

Tabela 3 – Resultados de marcações corretas das pupilas

N° Pupilas Marcadas Corretamente	N° Pupilas Marcadas	Acurácia
60	275	21,8%

Fonte: Autoria Própria.

O resultado do processamento automático conseguiu detectar algumas pupilas perfeitamente, porém, devido a presença dos cílios na detecção tornou o processo marcação defeituoso. Foi acordado com o professor supervisor a possibilidade de aparar os cílios dos animais sem nenhum prejuízo aos mesmos, e, portanto, melhoraria a marcação das pupilas.

Outro problema é a demasiada demora no processamento do algoritmo, uma vez que a detecção de elipses requer muitos parâmetros variáveis.

Uma proposta para trabalhos futuros é o treinamento de uma *CNN* para detecção dos olhos, uma vez que o classificador *haar cascade* também detecta padrões que não são olhos com uma certa frequência, e portanto, também são destacados na imagem, indicando um falso positivo.

6 CONCLUSÃO

Foi apresentado neste trabalho uma sequência de procedimentos realizados durante a vigência do estágio, que realiza o cálculo automatizado da Razão de Estresse Pupilar, que indica o nível de estresse em caprinos utilizando processamento de imagens.

O resultado apresentou uma baixa taxa de acerto nas marcações de pupila devido à presença de cílios na maioria das imagens. Uma solução possível para esta situação seria o corte prévio dos cílios dos animais, sem prejuízo aos mesmos.

As maiores dificuldades enfrentadas foram o treinamento do classificador mediante a vários testes de configurações, a implementação de um código eficiente da transformada de *Hough* para elipses, que torna o processamento lento e a criação de um algoritmo próprio para detecção de bordas na imagem da pupila binarizada.

REFERÊNCIAS BIBLIOGRÁFICAS

MARQUES, Jordânio Inácio, **Dilatação Pupilar como Indicador de Estresse térmico em Caprinos Mestiços Boer Mantidos em Câmara Frigorífica.** 2017. Dissertação (Mestrado) – Curso de Engenharia Agrícola. Universidade Federal de Campina Grande, Campina Grande, 2017. Disponível em: Link do Trabalho. Acesso em: 21 mar. 2022.

GONZALEZ, C.; WOODS, E. ***Digital Image Processing*** (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

FELTRIN, F. **Visão Computacional em *Python*.** Ed. Uniorg, 2020.

VARGAS, C.; CARVALHO, M.; VASCONCELOS, N. **Um estudo sobre Redes Neurais Convolucionais e sua aplicação em detecção de pedestres**, 2016.

MCKINNEY, W. ***Python*** **para Análise de Dados tratamento de dados com pandas, *numpy* e *iPython*.** 2 ed. Novatec Editora Limitada, 2018.

CHOLLET, François. ***Deep Learning with Python*.** 2 ed. Hanning, 2020.

HALTERMAN, R. ***Learning to program with Python*.** 2011.

Et al. LENCIONIL, G.; SOUZA, R. **Avaliação da dor em cavalos usando reconhecimento automático de expressão facial por meio de modelagem baseada em aprendizagem profunda**. 2021. Curso de Medicina Veterinária e Zootecnia. Universidade de São Paulo. São Paulo. 2021. Disponível em: https://jornal.usp.br/ciencias/algoritmo-usa-imagens-de-expressoes-faciais-para-detectar-nivel-de-dor-em-cavalos/