UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS AGRÁRIAS
DEPARTAMENTO DE ENGENHARIA AGRÍCOLA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA AGRÍCOLA

FLÁVIO ROBERTO DE FREITAS GONÇALVES

VISÃO COMPUTACIONAL APLICADA A AUTOMAÇÃO DE COLHEDORA MULTIFUNCIONAL DE HORTÍCOLAS - ALFACE

FORTALEZA
2019

FLÁVIO ROBERTO DE FREITAS GONÇALVES

VISÃO COMPUTACIONAL APLICADA A AUTOMAÇÃO DE COLHEDORA MULTIFUNCIONAL DE HORTÍCOLAS - ALFACE

Tese apresentada ao Programa de Pós-Graduação em Engenharia Agrícola da Universidade Federal do Ceará, como parte dos requisitos à obtenção do título de doutor em Engenharia Agrícola. Área de concentração: Engenharia de Sistemas Agrícolas.

Orientador: Prof. Dr. Daniel Albiero.

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G625v Gonçalves, Flávio Roberto de Freitas.
Visão Computacional Aplicada a Automação de Colhedora Multifuncional de Hortículas: Alface / Flávio Roberto de Freitas Gonçalves. – 2019.
92 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências Agrárias, Programa de Pós-Graduação em Engenharia Agrícola, Fortaleza, 2019.
Orientação: Prof. Dr. Daniel Albiero.

1. Arducam. 2. Visão Estereo. 3. Redes Convolucionais. 4. Colhedora Multifuncional. 5. Yolo. I. Título.
CDD 630

FLÁVIO ROBERTO DE FREITAS GONÇALVES

VISÃO COMPUTACIONAL APLICADA A AUTOMAÇÃO DE COLHEDORA MULTIFUNCIONAL DE HORTÍCOLAS: ALFACE

Tese apresentada ao Programa de Pós-Graduação em Engenharia Agrícola da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia Agrícolas. Área de concentração: Engenharia de Sistemas Agrícolas.

Aprovada em: 25/07/2019.

BANCA EXAMINADORA

_____

Prof. Dr. Daniel Albiero (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Alexsandro Oliveira da Silva
Universidade Federal do Ceará (UFC)

_____

Prof. Ph.D. Adunias dos Santos Teixeira
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Auzuir Ripardo de Alexandria
Instituto Federal do Ceará (IFCE)

_____

Prof. Dr. Angel Pontin Garcia
Universidade Estadual de Campinas (UNICAMP)

A Deus.

Aos meus pais, Maria Eni de Freita Oliveira e José Gonçalves de Oliveira.

A minha companheira Marcia Batista Torres.

Ao meu amigo e orientador Daniel Albiero e a todos que acreditaram neste trabalho.

AGRADECIMENTOS

Um trabalho como este que apresentamos, apesar de ter um autor, tem a participação de várias mãos que contribuíram direta ou indiretamente para sua conclusão. Espero não ser injusto com todas estas mãos.

Em especial, meu enorme agradecimento ao Prof. Dr. Daniel Albiero, grande entusiasta e excelente orientação pela fé neste trabalho e em seu executor.

Aos professores participantes da banca examinadora Prof. Dr. Alexsandro Oliveira da Silva, Prof. Dr. Adunias dos Santos Teixeira, Prof. Dr. Auzuir Ripardo de Alexandria e Prof. Dr. Angel Pontin Garcia pelo tempo e valiosas colaborações e sugestões.

Aos professores do Programa de Pós-Graduação em Engenharia Agrícola da UFC, pelo nível de excelência que empregam na formação profissional dos seus estudantes.

Aos colegas da turma de doutorado, pelas reflexões, críticas e sugestões recebidas.

Um especial agradecimento a Leonardo Lima, Babalorisá Leo Ty Osun, - Igbasé Ominolá, pelo aconselhamento espiritual e apoio nos momentos de indecisão, pois "a fé é aquilo que brota quando a razão e o sentimento não possuem forças para te impulsionar".

Aos meus amigos irmãos Paulo Fernando, Péricles Araújo, Haroldo Abreu, Fábio Montenegro, Achiles Chaves, Jósé Gledson, Matheus Costa, Rafael Utta, Rafael Constantino, Ciáxares Cipriano, Abraaõ Facó, Leonardo, Nunes Maria Batista, Allan, Thiago, |Erick e todos que contribuíram direta ou indiretamente e que não estão nominalmente citados. Suas colaborações e incentivo foram de enorme ajuda.

A minha família, em especial, ao meus pais, Maria Eni de Freitas Oliveira e Jose Gonçalves de Oliveira, que um dia tomaram a decisão de propiciar a seus filhos melhores oportunidade de estudo e migraram do interior para a capital, sem nenhuma certeza de sucesso e hoje posso agradece-los dizendo que seus esforços não foram em vão.

Nos momentos da jornada, quando pensamos em jogar tudo para o alto, abandonar a batalha também é o momento onde encontramos a voz, a mão, o porto seguro, o apoio que falta. Foi quando pensava em desistir que a metade complementar apareceu. Sem você não teria continuado, concluído, chegado onde cheguei. A você dedico este trabalho, pois sem você ele não existiria. A você Marcia Batista Torres a quem devo a realização deste trabalho.

Por fim, um especial agradecimento aos que não acreditaram na conclusão deste trabalho, por servirem de motivação para que este fosse concluído.

Vanitas vanitatum et omnia vanitas
O quam cito transit gloria mundi
Quod tibi non vis alteri ne facias
Amor vincit omnia
Labor improbus omnia vincit
Vincit omnia veritas
Omnia mecum porto
Fortis fortuna adiuvat
Memento mori
Si vis pacem, para bellum

(Eclesiastes XII, 8), Thomas Kempes, Écloga X- 69; Geórgicas, 144 e 145, Bias, Públio Terêncio Afro, saudação trapistas)

RESUMO

A automação agrícola tornou-se significativa em nosso país devido à necessidade das empresas nacionais de competirem de modo adequado com empresas estrangeiras e do aumento da produtividade e redução de perdas. Com o uso de recursos de avançada tecnologia como dispositivos eletrônicos embarcados e técnicas de agricultura de precisão e controle de processos, já é possível o aumento da produção. As culturas hortícolas são de grande importância econômica, social e alimentar para a população mundial. O grande desafio atual no setor de mecanização agrícola de hortícolas é a colheita. Para isso o desenvolvimento de um sistema de visão computacional com emprego de processamento de imagem e redes neurais artificiais que permita a identificação de hortícolas e forneça parâmetros de posicionamento a um conjunto atuador robótico, para operações de colheita, atenderia a este desafio. O objetivo do trabalho foi desenvolver um sistema de detecção, que através destes recursos possibilite a detecção de hortícolas e o posicionamento da mesma. Para esta finalidade definiu-se um aparato de captura utilizando câmeras seriais modelo OV2640 com interface de controle ArduCam /Arduino ajustadas em configuração de visão estéreo. A programação do sistema foi feita em C# utilizando a biblioteca EMGU com o uso de algoritmos de aprendizagem profunda (YOLO) e algoritmo de métrica de profundidade por meio de mapa de disparidades para estimação de distância. Como resultado temos a definição do aparato onde foram estabelecidas condições de operação e analisadas o desempenho dos algoritmos de captura de imagens, coletou-se banco de imagens, definiu-se a rede neural artificial e efetuado treinamento com estas imagens para identificação da hortícola da alface, gerou-se o sistema de detecção de objetos, calibrado o módulo de detecção de distâncias e realização ensaios para detecção de alfaces. Avaliando os resultados concluímos ser possível a detecção de hortícolas com o uso do sistema implementado.

Palavras-chave: Arducam. Visão Estéreo. Redes Convolucionais. Colhedora multifuncional. YOLO.

ABSTRACT

Agricultural automation has become significant in our country due to the need for domestic companies to compete properly with foreign companies and increased productivity and reduced losses. Using advanced technology features such as embedded electronic devices and precision farming techniques and process control, increased production is now possible. Horticultural crops are of great economic, social and food importance to the world's population. The current major challenge in the horticultural agricultural mechanization sector is harvesting. To this end, the development of a computer vision system employing image processing and artificial neural networks that allow the identification of vegetables and provide positioning parameters for a robotic actuator set for harvesting operations would meet this challenge. The objective of this work was to develop a detection system that through these resources enables the detection of vegetables and their positioning. For this purpose a capture apparatus was defined using model OV2640 serial cameras with ArduCam / Arduino control interface set in stereo vision configuration. The programming of the system was done in C # using the EMGU library using deep learning algorithms (YOLO) and depth metric algorithm by means of distance estimation map. As a result we have the definition of the apparatus where operating conditions were established and the performance of the image capture algorithms were analyzed, an image bank was collected, an artificial neural network was defined and training was performed with these images to identify the lettuce horticulture. , the object detection system was generated, the distance detection module was calibrated and lettuce detection tests were performed. Evaluating the results we concluded that it is possible to detect vegetables using the implemented system.

Keywords: Arducam. Stereo Vision. Convolutional Networks. Multifunctional Harvester. YOLO.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

ADALINE	Adaptive Linear Element
BSD	Berkeley Software Distribution
CCD	Charged-coupled device
CNN	Convolutional Neural Network
COLAB	Colaboratory Jupyter notebook environment
CMOS	Complementary Metal Oxide Semiconductor
CUDA	Compute Unified Device Architecture
EMGU	Emulator Graphic Unit
HSV	Sistema de cor Hue, Saturation, Value
IBM	International Business Machines Corporation
JPEG	Joint Photographic Experts Group
MADALINE	Multiple Adaline
NTSC	National Television System(s) Committee
OPENCV	Open Source Computer Vision Library
RGB	Sistema de Cor Red, Green, Blue
RNA	Redes Neurais Artificiais
SCCB	Serial Camera Control Bus
YOLO	You Only Look Once

SUMÁRIO

1 INTRODUÇÃO

A automação agrícola está a cada dia tornando-se mais significativa em nosso país principalmente devido à grande competitividade de empresas estrangeiras e da necessidade de aumento da produção e redução de perdas, consequentemente redução de custos no cultivo.

Com o auxílio de máquinas agrícolas já é possível o aumento da produção, porém, as máquinas atuais, mesmo com grande número de dispositivos eletrônicos embarcados e recursos de agricultura de precisão, ainda não contribuem para uma agricultura sustentável ao meio ambiente.

Culturas hortícolas são de grande importância econômica, social e alimentar para a população mundial, sendo o grande desafio atual no setor de mecanização agrícola de hortícolas é a colheita dotar uma máquina agrícola, da capacidade de identificar e localizar uma hortícola em campo utilizando-se de uma interface artificial para enfim colhe-la poderia atender a esse desafio.

Uma das interfaces de interação entre o meio e os sistemas robóticos é sem dúvida o sistema de Visão Computacional. Para tanto, acredita-se que a estereoscopia seja a melhor estratégia utilizada para detecção de distancias com uso de processamento de imagens.

Já a identificação, necessita de técnicas que permitam ao mecanismo reconhecer a hortícola com adequado grau de assertividade e em tempo hábil para operação de coleta.

O desenvolvimento de um sistema de identificação e localização de culturas com emprego de técnicas de processamento de imagem e inteligência artificial de modo a permitir que um dispositivo automático possa efetuar sua colheita, é de grande utilidade na colheita mecanizada de hortícolas e poderá reduzir as perdas quantitativas e qualitativas.

Espera-se com isto elaborar o sistema de modo a definir as distâncias entre cultura e o aparato, permitindo que um elemento atuador possa efetivar a coleta do alvo.

1.1 Hipótese

O desenvolvimento de sistema de visão computacional com emprego de técnicas de processamento de imagem e inteligência artificial permite a detecção da cultura e fornecer parâmetros de posicionamento na colheita mecanizada de hortícolas.

1.2. Objetivo

Este trabalho tem como objetivo desenvolver um sistema de identificação e localização da hortícola da alface utilizando recursos de processamento de imagem e redes neurais artificiais Visão.

1.2.1 Objetivos Específicos

- Especificar componentes constituintes do sistema de e elaboração do aparato.
- Codificar algoritmos para captura e processamento de imagem e de reconhecimento da cultura da alface.
- Estabelecer estratégias para posicionamento do dispositivo nos pontos selecionados.
- Avaliar desempenho dos algoritmos.

1.3. Justificativa

As aplicações das tecnologias de automação em sistemas agrícolas já são aplicadas em controle de pós colheita, beneficiamento de produtos, condução de máquinas agrícolas e monitoramento da produção.

A automação de processos agrícolas vem transformando o panorama da produção de alimentos, seja no plantio assessorado por sistemas SIG (Sistema de Informações Geográficas) ou no manejo com automação da irrigação e o uso de agricultura de precisão embarcado em máquinas agrícolas.

A utilização desta tecnologia embarcada em máquinas agrícolas e a velocidade de resposta deste sistema para o aumento da produtividade se apresentam como desafios a serem superados no desenvolvimento destes sistemas.

Por se tratar de um sistema inovador, a sua aplicação promoverá o aumento da produtividade, redução de perdas no cultivo de hortícolas, além de promover aporte de tecnologia visto que a maior parte das tecnologias de automação agrícolas e agricultura de precisão serem importadas.

1.4. Delimitação do Problema

Para atender a demanda em equipamento para horticultores, foram realizadas pesquisas pelo Grupo de Pesquisa em Energia e Máquinas para a Agricultura do Semiárido -

GEMASA que resultaram em uma colhedora multifuncional de hortícolas.

Esta colhedora é composta por sistema propulsor e cabeçote colhedor acionado automaticamente. (Patente número de depósito BR1020130197173).

Para o controle do cabeçote propõe-se um sistema de Visão Artificial, com algoritmos de detecção e estimação de distancias.

Este trabalho trata do desenvolvimento do sistema de visão a ser utilizado no controle de uma colhedora multifuncional de hortícolas.

2 REVISÃO BIBLIOGRÁFICA

2.1 O agronegócio

No panorama da produção agrícola mundial, o Brasil se apresenta como um grande competidor e forte candidato na disputa pelo mercado exterior. O agronegócio nacional vem sendo uma atividade segura e sustentável promovendo a economia, não somente pela participação no PIB, como se destacando no comércio internacional, sendo um dos líderes mundiais na produção e exportação de vários produtos agropecuários.

Conforme a CNA - Confederação da Agricultura e Pecuária do Brasil (2016) "O setor agropecuário liderou a economia brasileira em 2016 ao superar dificuldades conjunturais e a crise política. Aumentou de 21,5% para 23% sua participação no Produto Interno Bruto (PIB) e, hoje, representa 48% das exportações totais do país."

> O agronegócio é responsável por 33% do Produto Interno Bruto (PIB), 42% das exportações totais e 37% dos empregos brasileiros. Estima-se que o PIB do setor chegue a US$ 180,2 bilhões em 2004, contra US$ 165,5 bilhões alcançados no ano passado. Entre 1998 e 2003, a taxa de crescimento do PIB agropecuário foi de 4,67% ao ano. No ano passado, as vendas externas de produtos agropecuários renderam ao Brasil US$ 36 bilhões, com superávit de US$ 25,8 bilhões. (Portal do Agronegócio, 2017).

Segundo Ministério da Agricultura, Pecuária e Abastecimento – MAPA(2019), o valor bruto de produção, correspondente ao faturamento bruto dentro da propriedade rural, na safra de 2018 foi de R$570,31 bilhões, enquanto o de 2019 neste primeiro semestre foi de R$564,32 bilhões. No setor da Agricultura R$ 383,97 (safra 2018) / R$ 372,07 (safra 2019).

Segundo O Estadão (2019), os dados divulgados pelo Instituto Brasileiro de Geografia e Estatística - IBGE**,** O Produto Interno Bruto - PIB da agropecuária subiu 0,1% em 2018 . O PIB do país avançou 1,1% no mesmo resultado de 2017, e 0,1% no último trimestre**.**

Diante deste contexto um grave gargalo desta cadeia produtiva é a falta de equipamentos capazes de atender aos produtores de hortaliças, tanto em termos de exigências agronômicas das diversas espécies oleícolas, como em relação ao tempo gasto na execução de diferentes operações agrícolas ainda é elevado (MASCARENHAS e ROCHA, 1991).

Para Sarti, Sabbatini e Vian (2009) as tendências atuais de mercado são: maior potência e a automação das máquinas, permitindo melhor eficiência, maiores ganhos e redução de custos, com as empresas do setor, agora muito mais concentrada que nas décadas anteriores, buscando cada vez mais a diferenciação pela qualidade e por potência dos tratores e colheitadeiras.

Na área de pós-colheita já existem tecnologias e procedimentos uniformes que desde que adotados asseguram a qualidade das hortaliças produzidas nos diferentes sistemas de produção empregados no País (MORETTI, 2003).

Segundo Day (2011) citado por Sanches (2012), com o advento dos robôs agrícolas, um novo nicho no mercado, com a utilização de unidades que operem em tarefas específicas de forma autônoma e com maior precisão, de forma contínua, controlada e com ganhos de produção e redução de desperdícios.

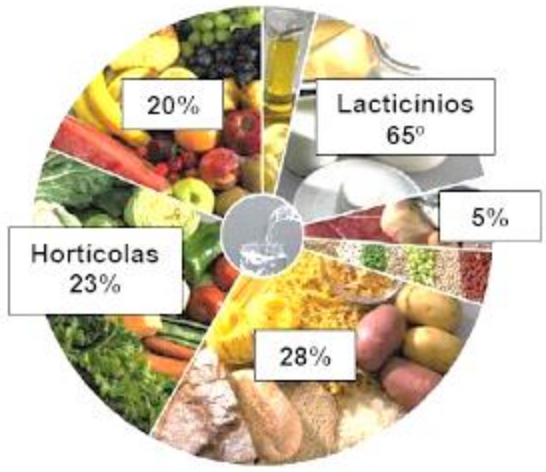
Conceitualmente, um robô é definido como um dispositivo reprogramável que realiza de forma autônoma a obtenção de informações do meio, tomando decisão de interação e atuando para atualização do meio ou de seu próprio estado. (RIA, 2014)

Segundo Cruvinel (2000), o panorama mundial aponta claramente para um futuro em que a agricultura dependerá inevitavelmente da automação. A automação contribui de forma preponderante tanto para gerar sustentabilidade no processo produtivo quanto para fomentar o desenvolvimento econômico e social. A aplicação da automação é ampla e existe potencial de contribuição em várias áreas.

2.1.1 Horticulturas

Dentre os alimentos utilizados pelo ser humano para a manutenção da vida e da saúde, as hortícolas têm importante papel. Para o perfeito equilíbrio alimentar os especialistas desenvolveram ferramentas para educação alimentar como por exemplo a Roda dos Alimentos como mostra a Figura 1.

Figura 1- Roda dos Alimentos.



Fonte: APHORT – Associação Portuguesa de Hotelaria, Restauração e Bebidas, (2010).

Com a função de promover hábitos alimentares saudáveis e de consumo, a roda dos alimentos sugere o percentual adequado de consumo de cada grupo alimentar. Destes

grupos, o segundo maior grupo corresponde às hortícolas com 23% de importância na alimentação. Este grupo é composto por vegetais variados tais como cenouras, nabos, brócolos, couve flor, couves, alface, agrião, salsa, abóbora, ervilhas, feijão verde, pepino, tomate, alho, cebola entre outros.

São ricos em água, fibras alimentares, carotenos, vitaminas do complexo B, vitamina C e sais minerais. Possuem baixo teor de gordura e açúcar não tendo valor energético, porem responsáveis pela regulação de outras funções orgânicas(APHORT, 2010)

O panorama da horticultura brasileira é muito favorável ao desenvolvimento de novas tecnologias para esta cadeia produtiva. Segundo Melo (2013), o valor da produção de hortícolas em 2006 foi de R$ 11,5 bilhões de reais em uma área cultivada de 771 mil hectares com uma produção total de 17,5 milhões de toneladas, gerando 10 milhões de empregos diretos. Estes números apresentam a pujança do setor e indicam um forte mercado para a indústria de máquinas agrícolas. É um mercado altamente diversificado com mais de 100 espécies cultivadas comercialmente que apresentam ciclo curto de produção, o que leva a diversas safras por ano.

No Brasil, apesar do aumento da população brasileira que já ultrapassou a marca de 200 milhões de habitantes e do acréscimo da renda média, o consumo no país tem decrescido, em contraste com o fato do país ser conhecido internacionalmente como super produtor agrícola.

As razões que levam a esta realidade vão desde aspectos culturais ao alto custo das hortaliças, aliado a vulnerabilidade do país devido a dependência em importações.

De acordo com Guedes (2016), "O consumo médio doméstico de hortaliças no Brasil está em torno de 27 kg/Hab/ano onde países como a Coreia do Sul tem consumo médio anual de 170 kg/Hab/ano justificado em grande parte pelo elevado custo deste produto".

Ainda segundo ele,

> O acréscimo da renda média do trabalhador e a busca por melhoria da qualidade de vida está estimulando o interesse na produção de hortaliças em contraposição ao aspecto econômico social que gerando uma competição com outras setores produtivos que vem tornado a mão de obra para a agricultura escassa e cara, tem gerado desafios para a produção agrícola conduzindo ao uso mais constante de mecanização em áreas de cultivo e cultivo protegido de hortaliças (GUEDES, 2016).

Esta realidade faz com que a SNA - Sociedade Nacional de Agricultura (2017) declare que:

> A busca no mercado de horticultura hoje é por novidades em tecnologia e inovação que elevem a produtividade no campo e assegurem qualidade cada vez maior dos produtos, nas diversas formas de cultivo. O crescimento acelerado de produção

> estimado em mais de 120% na última década e o faturamento superior a R$ 45 bilhões por ano, de acordo com a Associação Brasileira de Sementes e Mudas (ABCSEM), revelam ainda que as áreas ocupadas por hortaliças representam 500 mil hectares no país.

A Companhia Nacional de Abastecimento – Conab (2017), analisou as cotações das Centrais de Abastecimento – Ceasas, neste início de semestre de 2019 e constatou que a alface teve queda de preços em algumas centrais em Recife (34,76%) e Goiânia (11,11%) e com menor percentual em Belo Horizonte e Fortaleza.

2.1.1.1 Alface

A alface (*Lactuca sativa L.*) é uma hortalíça do tipo monoica, herbácea pertecente a Ordem *Asterales*, Família *Asteraceae*, Gênero *Lactuca*. Seu caule é curto, não ramificado, ao qual se prendem as folhas, estas por sua vez podem ser lisas ou crespas. A sua coloração irá depender da cultivar, pode ser verde ou roxa.

Segundo Filgueiras(2003), o sistema radicular do tipo pivotante, com ramificações muito finas, delicadas e curtas atigindo os primeiros 25 cm de profundidade do solo, quando a cultura é transplantada.

Já conforme Nunes(2019), em semeadura direta, pode atingir 60 cm de profundidade..

Originou-se de espécies silvestres das regiões sul da Europa e da Asia Ocidental. Segundo Ryder, (2012, p. 01), foi introduzida na América em 1494, por Cristóvão Colombo em uma de suas expedições para o novo mundo.

Segundo Putti (2014, p. 09), a cultura apresenta melhor adaptação ao solo de textura média, e com boa capacidade de retenção de água. A faixa de pH em que se deve conduzir a cultura deve variar entre 6,0 a 6,8. A calagem deve ser realizada para manter a saturação por base entre 80% a 90%. A adubação orgânica, principalmente com esterco animal, eleva o rendimento da cultura, por ser uma fonte de nutrientes, melhorar a qualidade física do solo e aumentar a atividade microrgânica.

Segundo Costa (2005, p. 01), no Brasil, as principais cultivares de alface americana disponíveis apresentam limitações de cultivo em determinadas regiões e épocas de plantio. O pendoamento precoce, devido a temperaturas elevadas, afeta a formação de cabeça e a alta pluviosidade tem limitado seu cultivo no período de verão, devido a perdas ocasionadas por doenças fúngicas e bacterianas.

O fator que afeta a planta é o fotoperíodo, pois a alface exige dias curtos durante a fase vegetativa e dias longos para que ocorra o pendoamento. Dias longos associados a

temperaturas elevadas aceleram o processo, o qual é também dependente da cultivar (RYDER, 1986 apud, SANTOS, 2013).

Segundo Lima (2007, p. 03), o período de cultivo varia de 40 a 70 dias dependendo do sistema (semeadura direta ou transplante de mudas), período de plantio (verão ou inverno), cultivar utilizada e sistema de condução, no campo ou protegido.

A alface é a mais popular entre as hortaliças folhosas, sendo produzida em quase todas as regiões do mundo. Tem importância econômica para o Brasil, já que ocupa o primeiro lugar entre as folhosas cultivadas e comercializadas. A alface está entre as dez hortaliças mais consumidas *in natura* (YURI *et al.*, 2004ª, p. 322).

Em uma planta de alface de 350 g, encontra-se aproximadamente 56 kcal; 95,8% de água; 2,3% de hidratos de carbono; 1,2% de proteínas; 0,2% de gordura; 0,5% de sais minerais (potássio – 13,3 mg, fósforo – 147 mg, cálcio – 133 mg, sódio, magnésio e ferro – 3,85 mg). Contém ainda provitamina A (245 UI), vitaminas do complexo B (B1 – 0,3 mg e B2 – 0,66 mg) e C (35,00 mg). (FILGUEIRA, 2003, p. 67).

Segundo Santos (2013, p. 14), no Nordeste brasileiro a produção desta cultura restringe-se a pequenas áreas, com a utilização de cultivares pouco adaptados às condições climáticas da região, o que favorece o pendoamento precoce (QUEIROGA *et al.*, 2001, p. 324). Entre os principais fatores associados às baixas produtividades obtidas na região tem-se o baixo nível tecnológico, falta de cultivares adaptadas às altas temperaturas e escassez de informações técnicas sobre a implantação e o manejo da cultura (OLIVEIRA *et al.*, 2006, p. 113). Destaque pode ser feito ao espaçamento a ser adotado para a cultura o quê, segundo Silva *et al.* (2000, p. 134), afeta significativamente a alface, alterando sua arquitetura, seu peso e, consequentemente sua qualidade final.

2.2 Tecnologias dos Sensores de Imagem

2.2.1 Sensores de Imagem

Segundo a Axis Communication (2017) um sensor de imagem é formado por um conjunto de fotopontos, correspondendo a um elemento de imagem conhecido como “pixel”. Cada pixel de um sensor de imagem registra a quantidade de luz à qual ele é exposto. Quanto maior a intensidade da luz, mais elétrons são gerados.

Duas tecnologias principais podem ser usadas no sensor de imagem de uma câmera: *Charged-coupled device* (CCD) e *Complementary Metal Oxide Semiconductor* (CMOS.

2.2.1.1 - Tecnologia CCD

Nos anos 70, o CCD foi o referencial de qualidade de imagem. O sensor CCD estava disponível com maiores resoluções, pixels de menor dimensão e um menor nível de ruído. Oferecem uma sensibilidade à luz ligeiramente melhor e geram menor quantidade de ruído que os sensores CMOS, entretanto, sensores CCD são mais caros e mais complexos de incorporar a uma câmera e podem consumir até 100 vezes mais energia que um sensor CMOS equivalente. (Security Camera King, 2017).

2.2.1.2 Tecnologia CMOS

Os sensores CMOS oferecem mais possibilidades de integração e mais funções, reduzem o custo total das câmeras, pois contêm toda a lógica necessária para montar as câmeras com base neles. Apresentam uma saída mais rápida (o que é uma vantagem quando são necessárias imagens com resolução temporaal mais alta), menor dissipação de energia no chip, além de reduzir as dimensões do sistema. Avanços recentes nos sensores CMOS estão aproximando dos sensores CCD em termos de qualidade de imagem, de forma a oferecer uma resolução mais alta e mais detalhes. Ela é menos sensível à luz que uma câmera VGA, pois seus pixels são menores e a luz refletida por um objeto se dispersa por um número maior de pixels. (Security Camera King, 2017).

2.2.2 Câmera serial

Uma câmera serial possui a característica de capturar imagens, codificá-la em um formato específico e transmiti-la por meio de um canal de comunicação (em sua maioria um canal serial). Possuem recursos de configuração podendo ser utilizada na captura de vídeos *National Television System(s) Committee* (NTSC) ou fotografia de 640x480, 320x240 ou 160x120 no formato compresso *Joint Photographic Experts Group* - JPEG ou outro para armazenamento e ou transmissão.

Com a utilização de um módulo serial, pode transmitir por apenas dois pinos com taxa de até 115200 *baunds,* como mostra a Figura 2.

Figura 2 – Câmera Serial.

Fonte: Saelig (2017).

A grande vantagem das câmeras seriais está na possibilidade de manipulação de parâmetros de captura, como resolução, formato e sistema de cores que permite adequar a câmera a uma operação mais específica.

2.2.3 - Sistemas de Cor

Para Scuri, (2002) a escolha do sistema ou espetro de cores para o processo de identificação de objetos em imagens é da maior importância para o sucesso do processamento de imagens. Existem diferentes sistemas de cores das quais podem-se destacar como principais o sistema de Cor *Red, Green, Blue* (RGB) e o sistema de cor *Hue, Saturation, Value* (HSV).

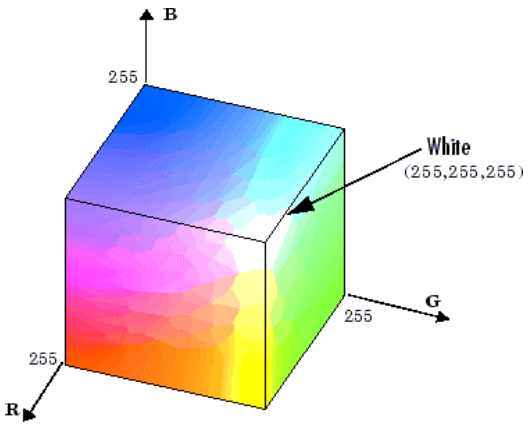
2.2.3.1 Sistema de Cor Red, Green, Blue - RGB

O RGB é o sistema de cores mais comumente utilizado em monitores ou câmaras baseia-se na teoria de visão colorida tricromática de *Young-Helmholtz*. Sua operação é equivalente ao dos receptores do olho humano. É um sistema de cores aditivo, ou seja, cada cor obtida corresponde ao somatório das intensidades de vermelha, verde e azul. Computacionalmente, são utilizados 8 bits de codificação de cada canal, representando 256 (de 0 e 255) níveis de intensidade luminosa.

Neste sistema, o branco resulta da soma dos valores máximos de cada componente, o preto da soma dos mínimos e um nível de cinza dos outros casos em que as três componentes tomam o mesmo valor. No RGB, as informações de cor (*chroma*) e intensidade (luma) encontram-se misturadas. Desta forma, se for desejado mais brilho numa

cor, todas as componentes devem ser escaladas do mesmo valor; se desejarmos uma cor com saturação máxima, sabemos que ela se vai encontrar numa das faces visíveis do cubo RGB, conforme a Figura 3.

Figura 3 – Cubo RGB.



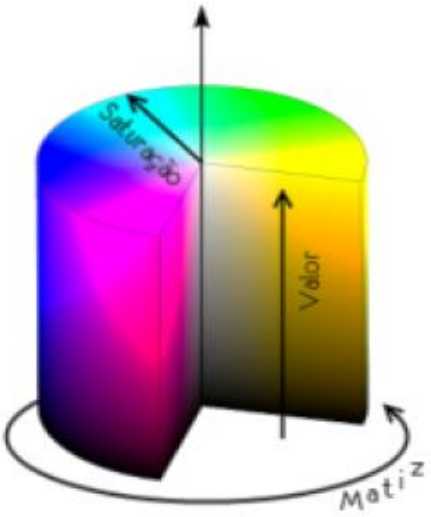
Fonte: Matwoks (2017).

Em aplicações em que se pretende fazer identificação de objetos e zonas de interesse em imagens, o sistema RGB revela-se ineficaz na presença de regiões de sombra provocadas por variações na iluminação. A alta correlação entre as três componentes faz com que uma variação no brilho (também chamado de luminância ou intensidade) provoque variação idêntica nos três canais.

2.2.3.2 Sistema de cor Hue, Saturation, Value - HSV

Com efeito, Paula(2012) apresenta os canais de cores HSV (*Hue*, *Saturation* e *Value*), HLS (*Hue*, *Luminance* e *Saturation*) e HSI (*Hue*, *Saturation* e *Intensity*) que utilizam três componentes presentes em diversas técnicas de pintura: Matiz, Saturação e Iluminação(Intensidade ou Valor adquirido através do brilho na imagem). A matiz é definida de modo angular e os outros componentes são lineares. O sistema HSV separa cada cor em matiz (H - *Hue*), saturação (S - *Saturation*) e brilho (V - *Value*). A informação de cor e intensidade estão separadas: a de cor é representada pelos valores de H e S e a de intensidade pelo canal V. A matiz representa as cores elementares e é determinada pelo comprimento de onda dominante, como mostra a Figura 4.

Figura 4 – Sistema de Cores HSV.



Fonte: Neves (2013).

A saturação é uma medida da pureza da cor e contém informação da quantidade de luz branca misturada com a matiz. O brilho refere-se à quantidade de luz. A matiz varia entre 0 e 360º enquanto a saturação o e brilho variam entre 0 e 100%. No caso particular do OpenCV, a matiz varia entre 0 e 180 para que seja possível utilizar uma variável de 8 bits, e saturação e brilho estão escaladas para 0-255.

2.3 Interface de Prototipagem ARDUINO

O Arduino é uma pequena placa de circuito impresso, sendo indicado para criação de protótipos de eletrônica, baseado nas filosofias de *software* e *hardware* livres. Ele pode interagir com o ambiente recebendo em suas entradas sinais dos mais variados tipos de sensores e pode inferir em um processo por meio do acionamento de luzes, motores ou outros atuadores (LIMA, 2013).

Sua estrutura composta por placa composta por um microcontrolador ATmega328P, 6 entradas analógicas, 14 entradas e saídas digitais, conversor serial para USB, fonte de alimentação externa e os pinos de energia com 3,3 V, 5 V e Terra (GND).

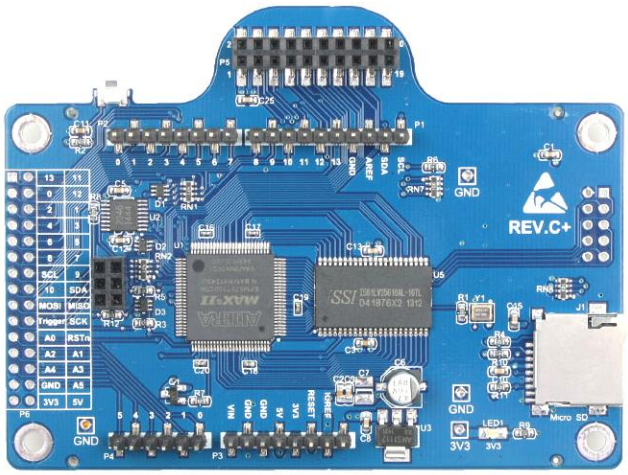
Para expandir a interfaces Arduino, são desenvolvidos módulos com funcionalidades específicas chamadas *Shields* que permitem adicionar capacidades a mesma. Um módulo importante para o nosso trabalho é a placa de expansão ArduCam.

2.4 Módulo controlador ARDUCAM

ArduCAM é uma controladora de vídeo de uso geral para Arduino. Esta interface mascara a complexidade da câmera permitindo facilmente controle e acesso a características do hardware da câmera. O ArduCAM suporta vários módulos de câmera de 0.3 MP a 5 MP com formatos RAW, RGB, YUV, JPEG e pode ser acoplado com placas padrão Arduino.

O modelo atual de interface disponível é o ArduCAM Rev.C + que oferece melhor desempenho e funções atualizadas. É projetada para Arduino, mas pode ser utilizado com outras arquiteturas que utilizam a interface SPI e I2C.

Figura 5 – Interface controladora ArduCam



Fonte: ArduCam ,2017

Seguem abaixo as características do módulo ArduCam, assim como estão listados na Tabela 01 os módulos de câmera suportados por esta interface.

- Suporta módulos de câmera de 0.3MP a 5MP
- Conexão com LCD TFT LCD de 3,2" com *touch screen*
- Card socket para SD/TF
- Suporta modo de compressão JPEG e captura de vídeo
- Opera com múltiplas plataformas microprocessadas
- Possui código e bibliotecas open source

Tabela 1: Câmeras suportadas pela interface ArduCam

Resolução	Sensor	Módulo de câmera
0.3MP	Omnivision	OV7660 / OV7670 / OV7675 / OV7725
0.3MP	Aptina	MT9V111
1.3MP	Aptina	MT9M112 / MT9M001
2MP	Omnivision	OV2640

2MP	Aptina	MT9D111 / MT9D112
3MP	Omnivision	OV3640
3MP	Aptina	MT9T112
5MP	Omnivision	OV5640 / OV5642

Fonte: ArduCam

2.5. Visão Computacional e Processamento de Imagem

2.5.1 - Visão Computacional

Pode-se definir Visão Computacional como sendo um conjunto de algoritmos através dos quais sistemas baseados em computadores podem extrair informações dos pixels que compõem a imagem (Gardiman,).

Segundo Crowley e Christensen (1995 *apud* Silva, 2008), "Visão Computacional é a área da ciência que estuda e procura desenvolver teorias e métodos voltados à extração automática de informações úteis contidas em imagens. Tais imagens são capturadas por dispositivos como câmera de vídeo e scanner".

Segundo Marr (1982), visão é o método ou processo de descobrir por meio de imagens o que está presente no mundo e onde se está localizado, sendo assim, uma tarefa de processamento de informação por meio de imagens.

Segundo Faugeras (1993 *apud* Paula, 2012), nos dias de hoje a crescente utilização destes sistemas se dá pelo desenvolvimento de novas tecnologias que promoveram a redução dos custos e a aplicações destes sistemas em diferentes áreas como por exemplo, análise biomédicas automáticas, medição dimensional de peças, monitoramento de animais , análise morfológica ,reconhecimento e identificação de faces humanas, entre outras.

Os processos de visão computacional, que é o conjunto de métodos e técnicas através dos quais sistemas computacionais podem ser capazes de interpretar imagens, muitas vezes, necessitam de etapas de processamento de imagens.

Para Morris (2004) um sistema de Visão Computacional possui as seguintes etapas:

- Aquisição de imagem;
- Pré-processamento;
- Extração de características;
- Detecção e segmentação;

• Processamento de alto nível;

2.5.2 Processamento Digital de Imagens

Processamento de imagem é qualquer forma de manipulação de dados no qual a entrada e saída são imagens, tais como fotografias ou quadros de vídeo.

O Processamento de Imagens tem como foco a melhoria dos aspectos visuais de uma imagem para a análise humana.

Umas das principais técnicas de processamento digital de imagens tinha como objetivo melhorar ilustrações de jornais que eram enviadas por cabo submarino entre Londres e Nova York por volta de 1920 (Gonzalez and Woods, 1992), evidenciando assim o principal interesse da utilização dessa técnica: a necessidade de melhorar a qualidade da informação para interpretação humana.

Já nos anos 60, o processamento digital de imagens evoluiu com o advento de computadores digitais e com o programa espacial americano.

2.5.2.1 Técnicas de Processamento de Imagens

Processamento em visão computacional necessitam de uma etapa de pré-processamento onde as imagens de onde extrairemos a informação sofrem operações para melhorar ou destacar uma característica específica. Destas operações é a filtragem que visa remover ruídos provenientes do processo de aquisição da imagem (Marengoni e Stringhini, 2009).

Os ruídos podem aparecer de diversas fontes, como por exemplo, o tipo de sensor utilizado, a iluminação do ambiente, as condições climáticas no momento da aquisição da imagem, a posição relativa entre o objeto de interesse e a câmera. Note que ruído não e apenas interferência no sinal de captura da imagem, mas também interferências que possam atrapalhar a interpretação ou o reconhecimento de objetos na imagem (SANTOS, 2011).

Os filtros são as ferramentas básicas para remover ruídos de imagens. A Figura 6 apresenta um exemplo de uma imagem com ruído (à esquerda) e da imagem filtrada (à direita).

Figura 6 - Exemplo de ruído em imagem.

Fonte: Luiz (2008)

Os filtros podem ser espaciais (atuam diretamente na imagem) ou de frequência (a imagem é transformada para o domínio de frequência através da transformada de Fourier é filtrada neste domínio).

2.5.2.2 Domínio de Espaço

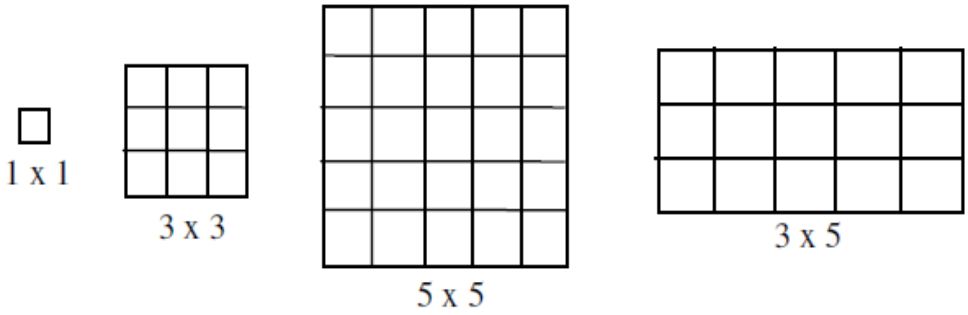
As operações no domínio do espaço referem à imagem em si, e tem como base a manipulação direta dos *pixels* da imagem. Os processos no domínio espacial são caracterizados pela equação 1.

$$g(x,y) = T(f(x,y)) \quad (1)$$

onde: f(x,y) é a imagem original, T(.) é uma transformação na imagem e g(x,y) é a imagem transformada.

A transformação T é realizada em um *pixel* considerando a sua vizinhança de influência. A vizinhança de influência considera os *pixels* ao redor da posição x, y. Esta vizinhança é delimitada por uma região (quadrada ou retangular) e de tamanho ímpar. A Figura 7 apresenta exemplos de vizinhança com tamanhos que definem matrizes nas operações de transformação chamadas de máscaras.

Figura 07 - Exemplo de regiões de vizinhança (máscaras).



Fonte: Luiz (2008).

As transformações comumente realizadas são as de intensidade, binarização e realce de contraste definidas nas equações 2, 3 e 4 e apresentadas na Figura 8.

$$g(x,y) = \begin{cases} 1, se f(x,y) \geqslant k \\ 0, caso\ contrario \end{cases} \quad (2)$$

$$g(x,y) = \frac{1}{1 + (\frac{m}{f(x,y)})^2} \quad (3)$$

$$g(x,y) = C.\log(1 + f(x,y)) \quad (4)$$

Figura 8 - Transformações no dominio do espaço: a -intensidade, b – binarização, c – contraste.

Fonte: Autor, 2019.

2.5.2.3 Histograma

Os histogramas são diagramas definidos através da distribuição de frequência de uma determinada variável. Em processamento de imagens são determinados a partir de valores de intensidade dos pixels. Dentre as aplicações dos histogramas estão a melhora da definição de uma imagem, a compressão, a segmentação e a descrição de uma imagem.

Em operações com imagens o ajuste dos valores de intensidade de forma a melhorar a normalização de uma intensidade na imagem chama-se equalização. Com a

equalização e possível entre outras tarefas a correção de contraste. A distribuição de um histograma permite o realce de elementos da imagem. A equalização é definida pela expressão 5.

$$h_{eq}(k) = \frac{(L-1)}{MN} \sum_{j=0}^{k} \quad n_j \tag{5}$$

onde: k é a intensidade no histograma equalizado;

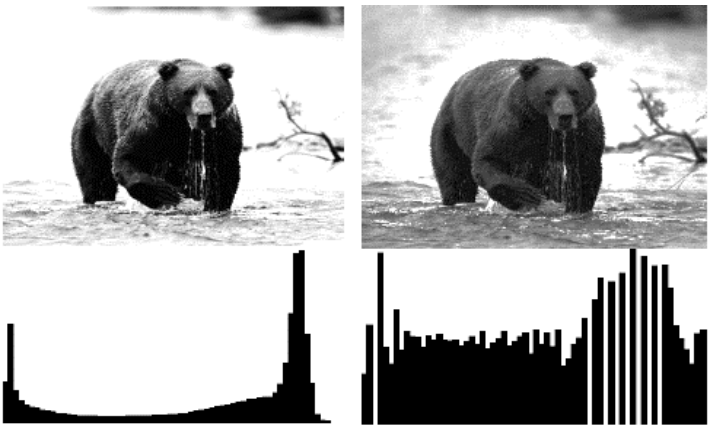
L é o valor máximo de intensidade na imagem;

M e N são as dimensões da imagem;

n_j é o número de pixel na imagem com valor de intensidade igual a j.

|Na Figura 9 a aplicação de um ajuste por equalização de histograma e seus respectivos histogramas.

Figura 9 – Ajuste de histograma por equalização.



Fonte: QUEIROS, 2001.

As operações com filtros podem ser realizadas em duas diferentes estratégias. As transformações no domínio do espaço por dependerem da vizinhança de influência (máscara). As operações de correlação e convolução, correspondem a criação de uma máscara com dimensões *d* com valores distintos e para cada posição (x,y) é efetuado a somatória dos produtos de cada valor da máscara pelo valor do pixel, conforme as equações 6 e 7 respectivamente.

$$g(x,y) = \sum_{i=m/2}^{m/2} \sum_{j=n/2}^{n/2} f(x+i, y+j) * w(i,j) \quad (6)$$

$$g(x,y) = \sum_{i=m/2}^{m/2} \sum_{j=n/2}^{n/2} f(x-i, y-j) * w(i,j) \quad (7)$$

A principal diferença da correlação e da convolução é a rotação da máscara.

Ainda para o domínio do espaço podemos utilizar filtros estatísticos tais como filtro de média, mediana, moda, mínimo e máximo.

Outros filtros possuem sua máscara determinada por funções como por exemplo os a função Gaussiana discreta conforme a equação 8.

$$Gauss(x,y) = \frac{1}{2\pi\sigma} \exp\left(\frac{-(x^2+y^2)}{(2\sigma^2)}\right) \quad (8)$$

onde x e y são as posições na máscara e Gauss(x,y) dá o valor a ser colocado na posição (x,y) da máscara.

Além dos filtros citados, filtros do tipo passa-alta também podem ser utilizados para realçar bordas ou regiões com transições abruptas de intensidade.

2.5.2.4 Domínio de Frequência

Uma outra forma de abordar uma imagem é representá-la por uma soma ponderada de senoidais onde mudanças na imagem são vistas como frequências altas ou baixas. Esta análise é realizada por meio da transformada de Fourier. No domínio da frequência, definir um filtro é encontrar uma máscara aplicável a uma Imagem transformada obtendo a imagem filtrada neste domínio.

As funções que expressam estas transformações de base são apresentadas nas equações 9 e 10, no modo contínuo e discreto respectivamente.

Contínuo:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{(-j2\pi(ux+vy))} \quad (9)$$

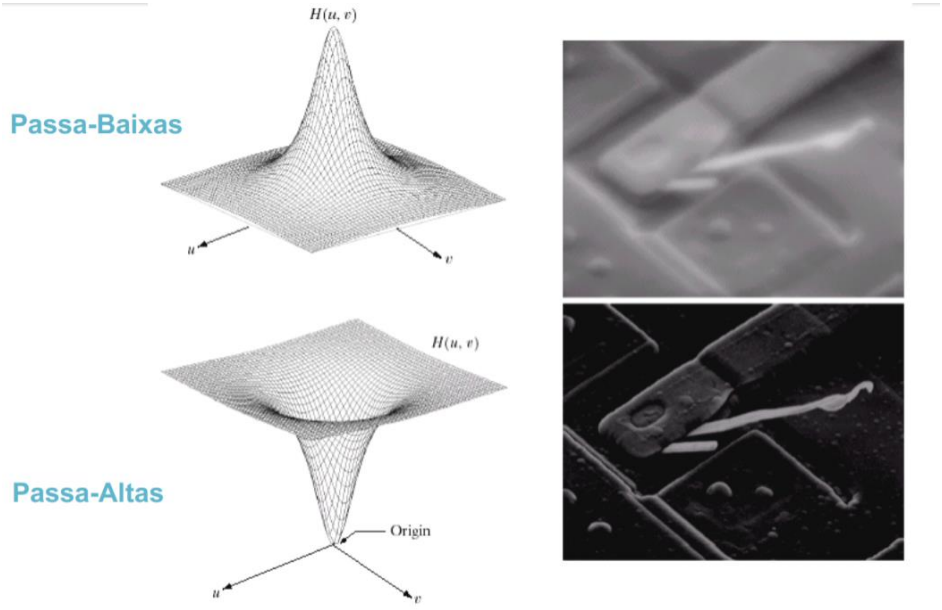
Discreto:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{n-1} f(x,y) e^{(-j2\pi(ux/M+vy/N))} \quad (10)$$

Assim como no domínio do espaço a operação de convolução no domínio da frequência corresponde matematicamente a multiplicação de expressões.

Os principais filtros aplicados neste domínio são os filtros passa-baixa e passa-alta, como mostra a Figura 10.

Figura 10- Filtros no domínio da Frequência.



Fonte: Iris.sel.eesc.usp, 2019.

2.5.2.5 Segmentação

A segmentação consiste em dividir uma imagem em regiões, ou objetos com características distintas como cor ou proximidade. A profundidade da segmentação depende da resolução da imagem e do que se deseja segmentar.

O objetivo das técnicas de segmentação é dividir a imagem em suas diversas partes constituintes ou segmentos (objetos e regiões). O nível ou quantidade de divisões aplicadas na imagem varia conforme a aplicação, e em geral é realizada até atingir um nível de separação suficiente entre os objetos de interesse da cena analisada (GONZALEZ and WOODS, 1992).

Existem diferentes técnicas de segmentação que podem ser classificadas em três grupos, a saber:

- Segmentação por detecção de borda.
- Segmentação por corte.

• Segmentação por crescimento de região.

2.5.2.5.1 Segmentação por detecção de borda

Detecção de borda é uma técnica de processamento de imagem e visão computacional utilizada para detectar pontos de uma imagem em que a intensidade luminosa muda bruscamente. Estas mudanças, em geral, refletem uma descontinuidade ou transição de elementos da imagem.

Os operadores mais comuns de uma imagem são os operadores de Prewitt e de Sobel. A Figura 11 mostra as máscaras de cada um destes operadores para obtenção de bordas horizontais e verticais. Na Figura 12 observa-se no caso de bordas diagonais deve-se aplicar à máscara o referido ângulo de rotação diagonal.

Figura 11 – Operadores Prewitt e Soble.

Prewitt

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Sobel

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Fonte: QUEIROS, 2001.

Figura 12 – Operador Soble rotacionado a 45º.

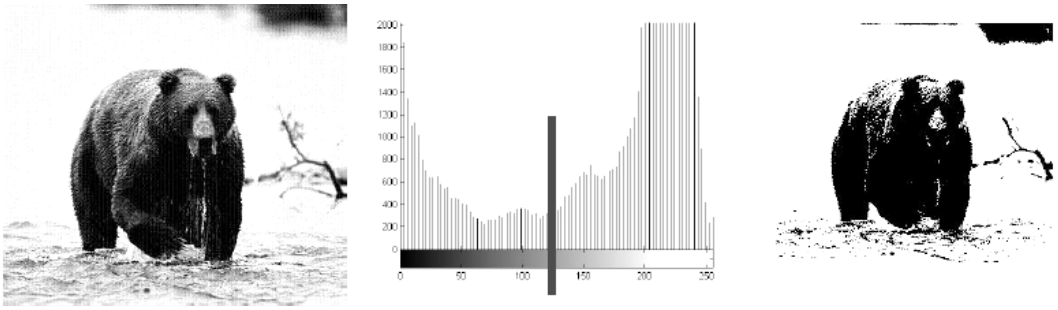
-2	-1	0	0	1	2
-1	0	1	-1	0	1
0	1	2	-2	-1	0

Fonte: QUEIROS, 2001.

2.5.2.5.2 Segmentação por Corte

A segmentação por corte utiliza de propriedades intuitivas para criar a imagem segmentada. Ela particiona a imagem em regiões tendo por base os valores de intensidade ou propriedades. A segmentação por corte busca no histograma da imagem regiões de picos e vales que servirão de base para a segmentação. A Figura 13 mostra uma imagem, o seu histograma e destaca onde a imagem poderá ser particionada e a imagem após o corte.

Figura 13 – Segmentação por corte.



Fonte: QUEIROS, 2001

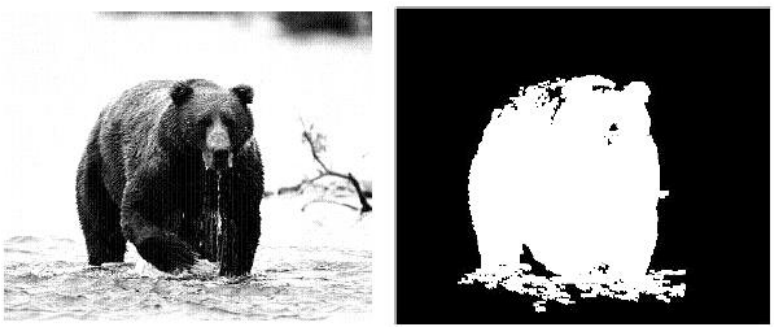
2.5.2.5.3 Segmentação por Crescimento de Região

A técnica de segmentação por crescimento de região encontra regiões diretamente na imagem agrupando os pixels em regiões baseadas em critérios de crescimento e parada. Partindo de um conjunto de pontos (geralmente chamados de sementes), segue agrupando pontos utilizando uma vizinhança de influência, formando regiões.

Nesta vizinhança são analisadas propriedades e são medidas similaridades para determinar se o pixel faz parte ou não da região sendo considerada. As propriedades normalmente analisadas são: cor, intensidade de nível de cinza, textura.

A definição das sementes e de um critério de parada para o crescimento de regiões são pontos importantes nesta técnica, conforme mostrado na Figura 14.

Figura 14 – Segmentação por crescimento de região.



Fonte: QUEIROS, 2001.

2.6 Reconhecimento de Padrões

O reconhecimento de padrões está engajado no campo da visão computacional com atuações e perspectivas importantes para alcançar e realizar a máquina inteligente (Jähne, 2002 *apud* Silva, 2008).

Para Marengoni e Stringhini (2009), citado por Marques e Scardua (2018), "Reconhecer significa conhecer de novo, e isto implica num processo onde existe algum conhecimento prévio armazenado do conhecimento sobre o objeto a ser reconhecido".

Um sistema de visão necessita uma base de conhecimento dos objetos a serem reconhecidos, esta base de conhecimento pode ser implementada no código, ou pode ser aprendida a partir de um conjunto de amostras.

O reconhecimento de objetos em visão computacional está relacionado diretamente com o reconhecimento de padrões.

Um objeto pode ser definido por mais de um padrão (textura, forma, cor, dimensões, etc) e o reconhecimento individual de cada um destes padrões pode facilitar o reconhecimento do objeto como um todo. (MARENGONI e STRINGHINI , 2009.)

As técnicas de reconhecimento de padrões podem ser divididas em dois grandes grupos:

- Estruturais: onde os padrões são descritos de forma simbólica;
- Baseado em teoria de decisão: neste grupo os padrões são descritos por propriedades quantitativas - o objeto possui ou não estas propriedades.

As técnicas estruturais são mais complexas devido a dependência da estrutura em questão. Já os classificadores baseados em teoria da decisão possuem pacotes definições já prontos como os encontrados na biblioteca OpenCV.

2.7 Open Source Computer Vision Library - OPENCV

O Open Source Computer Vision Library (OpenCV) é uma biblioteca de software de visão computacional e aprendizado de máquina em código aberto sob a licença *Berkeley Software Distribution* (BSD) que oferece funções e métodos para sistemas de visão computacionais e processamento de imagens. (OPENCV, 2013).

O OpenCV foi construído para fornecer uma infra-estrutura comum para aplicativos de visão computacional e para acelerar o uso da percepção da máquina nos produtos comerciais.

A biblioteca tem mais de 2500 algoritmos otimizados, o que inclui um conjunto abrangente de algoritmos de visão computacional e de aprendizado de máquina clássicos e de última geração que podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmera, objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos a partir de câmeras estéreo, unir imagens para produzir uma resolução alta, acompanhar movimentos dos olhos, reconhecer paisagens e estabelecer marcadores para sobreposição com realidade aumentada, etc.

OpenCV tem mais de 47 mil pessoas de usuário comunidade e número estimado de downloads superiores a 18 milhões. A biblioteca é amplamente utilizada em empresas, grupos de pesquisa e órgãos governamentais, (OpenCV, 2019).

Possui interfaces C ++, Python, Java e MATLAB e suporta Windows, Linux, Android e Mac OS. O OpenCV inclina-se principalmente para aplicativos de visão em tempo real e aproveita as instruções da MMX e da SSE quando disponíveis.

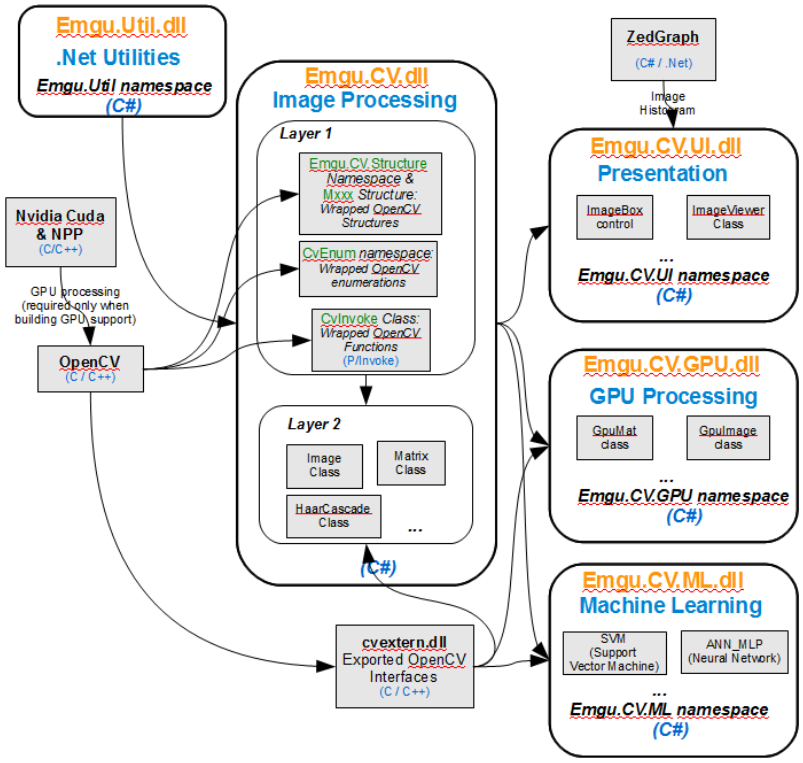
O uso desta biblioteca facilita não apenas na utilização de vários métodos sem a necessidade de implementação, que poderia levar a problemas, mas também mantém um padrão de estruturas de dados e de bom desempenho dos algoritmos que tratam a imagem em memória.

2.8 EMGUCV

O EmguCV é um *wrapper* .Net de plataforma cruzada para a biblioteca de processamento de imagens OpenCV permitindo que seus métodos sejam chamados a partir de linguagens compatíveis com .NET tais como C #, VB, VC ++, IronPython etc. O wrapper pode ser compilado pelo Visual Studio, Xamarin Studio e Unity, pode ser executado em Windows, Linux, Mac OS X, iOS, Android e Windows Phone.

As Vantagem de se utilizar EmguCV é que por ser escrita inteiramente em C # é capaz de rodar em qualquer plataforma suportada, incluindo iOS, Android, Windows Phone, Mac OS X e Linux. Outras vantagens são: Classes de imagem com cor e profundidade genérica, documentação XML e suporte, entre outras. A Figura 15 apresenta a arquitetura da EmguCV.

Figura 15 – Arquitetura da EMGUCV.



Fonte: Emgu.org, 2019.

2.9 - Redes Neurais Artificiais

Define-se uma Rede Neural Artificial como sendo um processador paralelamente distribuído e constituído de unidades de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Ela se assemelha ao cérebro pelo fato de que o conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizagem, onde forças de conexões entre neurônios são utilizadas para armazenar o conhecimento adquiridos, estes são conhecidos como pesos sinápticos (HAYKIN, 2009).

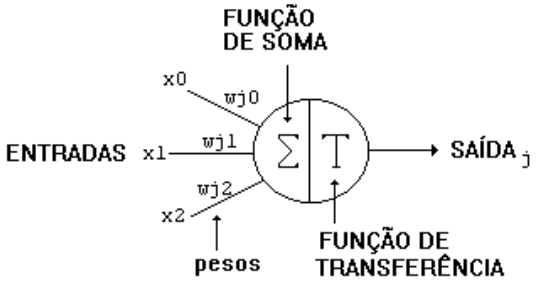
As redes neurais artificiais, (RNA) possuem hoje uma ampla aplicação, no agronegócio, sendo utilizados na detecção de ervas daninhas, estimativa da produtividade agrícola da cana-de-açúcar, suporte a decisão de colheita. determinação da cobertura do solo por análise de imagens entre outras.

O princípio de uma rede neural é a soma das entradas ponderadas nos neurônios, assim enviando os resultados para uma função de transferência para produzir uma saída. (Vadivambal, 2016).

2.9.1. Breve histórico

Proposto por McCulloch e Pitts em 1943 este modelo matemático simplificava a concepção a respeito do neurônio biológico da época em um dispositivo com entradas com ganhos arbitrários e uma saída binária. A descrição matemática proposta por eles resultou no modelo descrito abaixo, como mostra a Figura 16.

Figura 16 – Neurónio artificial McCulloch e Pitts.



Fonte: Redes Neurais Artificiais, 2019.

Onde: X1, X2, ...Xn são os sinais de entrada;

Wk são os pesos sinápticos do neurônio k;

Uk é o integrador linear de saída;

b é o biais e y o sinal de saída do neurônio.

Matematicamente o Neurônio Artificial pode ser expresso como nas Equações 11 e 12:

$$U_k = \sum_{j=1}^{p} W_{kj} X_j + b_k \qquad (11)$$

$$Yk = \varphi(uk) \qquad (12)$$

Com o avanço dos computadores digitais em 1950, pôde-se simular uma hipotética rede neural feito por Nathanial Rochester dos laboratórios de pesquisa da IBM. Em 1956, o Projeto de Pesquisa de Verão de Dartmouth sobre Inteligência Artificial proporcionou um impulso tanto à Inteligência Artificial como às Redes Neurais. O Perceptron, que resultou dessa pesquisa, foi construído em hardware e é a mais antiga rede neural ainda em uso hoje.

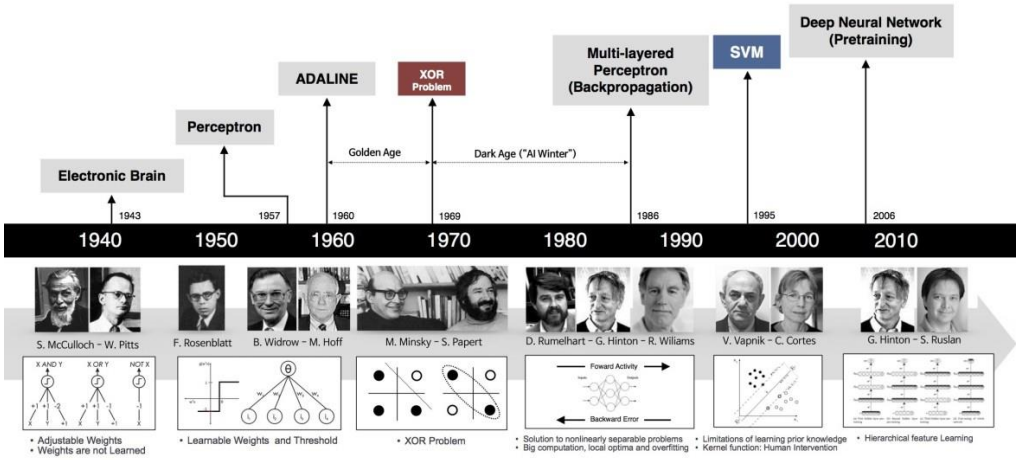
Em 1959, Bernard Widrow e Marcian Hoff, de Stanford, desenvolveram modelos denominados “ADALINE” e “MADALINE”. Em 1985, o Instituto Americano de Física começou o que se tornou uma reunião anual – Redes Neurais para Computação. Em 1987, a

primeira Conferência Internacional sobre Redes Neurais do Institute of Electrical and Electronic Engineer's (IEEE) atraiu mais de 1.800 participantes.

Em 1986, com redes neurais de várias camadas nas notícias, o problema era como estender a regra Widrow-Hoff para várias camadas.

Segundo, Data Science Academy,1992: Juyang Weng publica o Cresceptron, um método para realizar o reconhecimento de objetos 3-D automaticamente a partir de cenas desordenadas. Já em meados dos anos 2000: o termo "aprendizagem profunda" começa a ganhar popularidade após um artigo de Geoffrey Hinton e Ruslan Salakhutdinov mostrar como uma rede neural de várias camadas poderia ser pré-treinada uma camada por vez. Em2009: acontece o NIPS Workshop sobre Aprendizagem Profunda para Reconhecimento de Voz e descobre-se que com um conjunto de dados suficientemente grande, as redes neurais não precisam de pré-treinamento e as taxas de erro caem significativamente. 2012: algoritmos de reconhecimento de padrões artificiais alcançam desempenho em nível humano em determinadas tarefas. E o algoritmo de aprendizagem profunda do Google é capaz de identificar gatos. 2014: o Google compra a Startup de Inteligência Artificial chamada DeepMind, do Reino Unido, por £ 400m. 2015: Facebook coloca a tecnologia de aprendizado profundo – chamada DeepFace – em operação para marcar e identificar automaticamente usuários do Facebook em fotografias. Algoritmos executam tarefas superiores de reconhecimento facial usando redes profundas que levam em conta 120 milhões de parâmetros. 2016: o algoritmo do Google DeepMind, AlphaGo, mapeia a arte do complexo jogo de tabuleiro Go e vence o campeão mundial de Go, Lee Sedol, em um torneio altamente divulgado em Seul. 2017: adoção em massa do Deep Learning em diversas aplicações corporativas e mobile, além do avanço em pesquisas. Todos os eventos de tecnologia ligados a Data Science, IA e Big Data, apontam Deep Learning como a principal tecnologia para criação de sistemas inteligentes. A Figura 17 mostra o marcos no desenvolvimento das redes neurais.

Figura 17 - Marcos no desenvolvimento das redes neurais.



Fonte: Deep Learning Book, 2019.

2.9.2 Arquitetura de uma rede

Segundo Marques (2016),

> A arquitetura de uma rede se refere a organização da estrutura da rede, ou seja, quantidade de camadas, conexões, parâmetros e unidades de aprendizado. As redes são organizadas em camadas, que ganham um formato de cadeia. Em cada camada é aplicada a função de ativação. Normalmente, os valores de saída das camadas servem de entrada para a camada seguinte e cada camada recebe uma função de ativação, por exemplo, para a primeira camada:
> h(1) = g(1) (W(1)T x0 + b(1))
> Para a i-_esima camada tem-se:
> h(i+1) = g(i+1) (W(i+1)T xi + b(i+1))

Hoje muitas arquiteturas foram elaboradas para diferentes tarefas. Os principais tipos de arquitetura de rede neural são as redes recorrentes e *feedfoward*, más já existem arquiteturas específicas como a das redes convolucionais (derivadas da arquitetura *feedfoward*) que são muito eficientes na aplicação de reconhecimento de imagem e visão computacional.

2.9.3 Treinamento da rede

O treinamento de uma rede neural consiste na estimação de um modelo que minimize o erro entre a saída do sistema e a resposta desejada. Para a estimação desse modelo, precisa-se ajustar uma função de custo e o termo de regularização, escolher a função de ativação para calcular os pesos e, para atualização desses pesos, usa-se um método de aprendizagem baseado no gradiente. Basicamente, o treinamento da rede neural é feito utilizando otimizadores iterativos baseados em gradientes que conduzem a função de custo

para um valor muito baixo (MARQUES, 2016)

Segundo Bishop (2006), a função de custo ´e “uma medida global de perdas ocorridas para se tomar qualquer decisão ou ação. Sendo assim, o objetivo ´e minimizar a perda total do evento ocorrido”.

A função de custo aplicada para o treinamento de redes neurais profundas em geral combina o risco com técnicas para redução de erro denominada de regularização. Define-se regularização como qualquer modificação realizada no algoritmo de aprendizagem com o objetivo de reduzir o erro de generalização, mas não o erro de treinamento”. (GOODFELLOW; BENGIO; COURVILLE, 2016).

As funções de ativação de uma rede dependem do tipo de situação a ser tratada. Normalmente é difícil determinar quando usar cada função de ativação, sendo que o melhor modelo é definido a partir de tentativa e erro. Algumas funções de ativação mais comumente utilizadas são: sigmoide, tanh, ReLU.

2.9.4 Redes Neurais Convolucionais

Segundo Goodfellow, Yoshua, Courville (2016), as redes convolucionais, ou Convolutional Neural Network (CNN), são “redes especializadas em processamento de dados com topologia semelhante a grades. Por exemplo, dados de imagens possuem sua representação em uma grade 2D de pixels ou dados de séries temporais que podem ser tomados como amostras de grade 1D de um período regular de tempo. Além disso, as CNN’s podem ser aplicadas em diversos tipos de dados com diferentes dimensões”. O nome convolucional se refere a compilação matemática realizada neste tipo de rede, ou seja, redes convolucionais são simplesmente redes neurais que usam a convolução no lugar da multiplicação de matrizes em pelo menos uma de suas camadas.

O reconhecimento de imagem é um clássico problema de classificação, e as Redes Neurais Convolucionais possuem um histórico de alta acurácia para esse problema. A primeira aplicação com sucesso de uma CNN foi desenvolvida por Yann LeCun em 1998, com sete camadas entre convoluções e fully connected. Desde então as CNNs ficaram cada vez mais profundas e complexas, como AlexNet em 2012, que, apesar de ter apenas oito camadas (cinco convoluções e três fully connected), apresenta sessenta milhões de parâmetros, e a GoogleNet com vinte e duas camadas e quatro milhões de parâmetros.

No contexto de inteligência artificial e aprendizagem de máquina, uma rede neural convolucional (CNN do inglês *Convolutional Neural network* ou ConvNet) é uma classe de

rede neural artificial do tipo *feed-forward*, que vem sendo aplicada com sucesso no processamento e análise de imagens digitais (LISA Lab, 2013).

2.9.4.1 Visão geral da Arquitetura

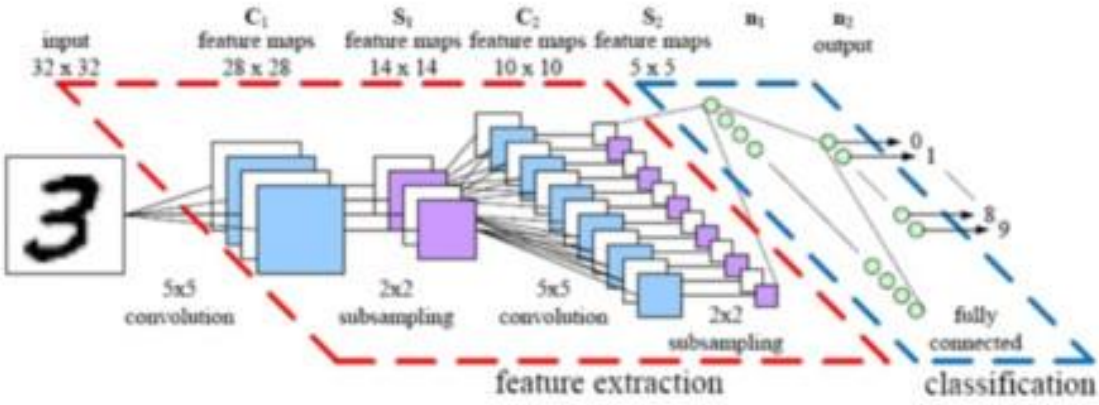
Uma ConvNet é composta de camadas. Cada camada transforma um volume 3D de entrada em um volume 3D de saída com alguma função diferençável que pode ou não ter parâmetros. Usam-se três tipos principais de camadas para construir as arquiteturas ConvNet: Camada Convolucional, Camada de Pooling e Camada Completamente Conectada (Fully-Connected Layer).

Como exemplo de arquitetura uma simples ConvNet para classificação poderá ter a arquitetura [INPUT - CONV - RELU - POOL - FC]:

- INPUT [32x32x3] - valores brutos de pixels da imagem (neste caso uma imagem de largura 32, altura 32 e com três canais de cores R, G, B.);
- CONV - calculará a saída de neurônios conectados a regiões locais na entrada, cada um computando um produto de ponto entre seus pesos e uma pequena região à qual estão conectados no volume de entrada. Isso pode resultar em um volume como [32x32xF], onde F define o número de filtros;
- A camada RELU aplicará uma função de ativação elementar, como o limiar máximo (0, x) em zero. Isso deixa o tamanho do volume inalterado ([32x32x12])
- A camada POOL executará uma operação de redução de resolução ao longo das dimensões espaciais (largura, altura), resultando em um volume como [16x16x12];
- A camada FC (ou seja, totalmente conectada) calculará as pontuações resultando em um volume de tamanho [1x1xM], em que cada um dos M números corresponderá a uma pontuação da turma, como entre as M categorias.

Desta forma, os ConvNets transformam a imagem original camada por camada dos valores de pixel originais para as pontuações finais da classe.

Figura 18 - Modelo arquitetural de uma rede neural convolucional



Fonte: KDnuggets™, 2019

2.9.4.1.1 - Camada Convolucional

A camada Conv é o núcleo do bloco de construção de uma Rede Convolucional responsável pela maior parte do trabalho computacional. Os parâmetros da camada CONV consistem em um conjunto de filtros que podem ser aprendidos. Por exemplo, um filtro típico em uma primeira camada de uma ConvNet pode ter tamanho 5x5x3 (5 pixels de largura e altura e 3, os canais de cores). À medida que deslizamos o filtro pela largura e altura do volume de entrada, produzimos um mapa de ativação bidimensional que fornece as respostas desse filtro. O resultado deve ser finalmente remodelado de volta à sua dimensão de saída adequada.

2.9.4.1.2 - Camada de agrupamento (*pooling*)

É comum inserir periodicamente uma camada *Pooling* entre as camadas Conv sucessivas em uma arquitetura ConvNet. Sua função é reduzir progressivamente o tamanho espacial da representação para reduzir a quantidade de parâmetros e computação na rede e, portanto, também controlar o *overfitting* e a redimensiona espacialmente.

2.9.4.1.3 - Camada totalmente conectada (*Fully-connected layer*)

Neurônios em uma camada totalmente conectada têm conexões completas com

todas as ativações na camada anterior, como visto em Redes Neurais regulares. Suas ativações podem, portanto, ser computadas com uma multiplicação de matrizes seguida por um deslocamento de polarização.

2.10 Darknet You Only Look Once - YOLO

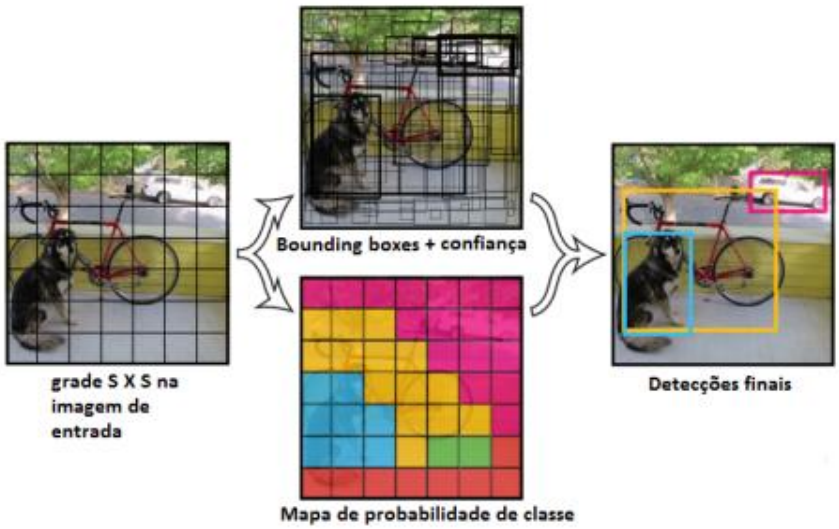
A rede neural profunda, cuja arquitetura é chamada de *darknet* é um *framework* de rede neural de código aberto escrito em C e *Compute Unified Device Architectur* (CUDA) e suporta computação de CPU e GPU. Esta arquitetura é composta de camadas convolucionais, filtros e funções de ativação descritas por um arquivo de configuração de extensão .cfg. (na rede YOLO, este arquivo é denominado yolov3.cfg). Além da arquitetura da rede neural, são necessários também os parâmetros de treinamento desta rede (*weights* e *biases*).

Segundo Veltroni (2018, p. 15) a rede convolucional YOLO detecta objetos em tempo real, além da detecção de quais objetos estão presentes em uma imagem. A YOLO também indica a localização de cada objeto através de *bounding boxes*. A YOLO é formada por uma rede neural convolucional inspirada na arquitetura da rede GoogleLeNet e tem as seguintes funções, todas realizadas de forma concorrente, o que agiliza a detecção:

- Extração de características da imagem;
- Predição de bounding box para cada objeto;
- Classificação e rotulação de cada bounding box com a classe mais provável.

Segundo Veltroni, (2018, p. 15) primeiramente a YOLO divide a imagem de entrada em uma grade de tamanho S x S. Cada célula da grade prediz no máximo duas bounding boxes em conjunto com as probabilidades de cada classe para cada *bounding box*, e também o grau de confiança. Em seguida, é realizado um procedimento que unifica as *bounding boxes* encontradas de forma que não haja dupla detecção de um mesmo objeto. A Figura 23 mostra o procedimento da YOLO em uma imagem de entrada.

Figura 19 - Funcionamento da rede YOLO.



Fonte: Redmon, 2016.

Sistemas de detecção prévia redirecionam classificadores ou localizadores para realizar a detecção. Eles aplicam o modelo a uma imagem em vários locais e escalas. Regiões de alta pontuação da imagem são consideradas detecções.

A Darknet YOLO utiliza uma abordagem diferente. Aplicamos uma única rede neural à imagem completa. Essa rede divide a imagem em regiões e prevê caixas delimitadoras ponderadas pelas probabilidades previstas.

2.11 Estéreo Visão

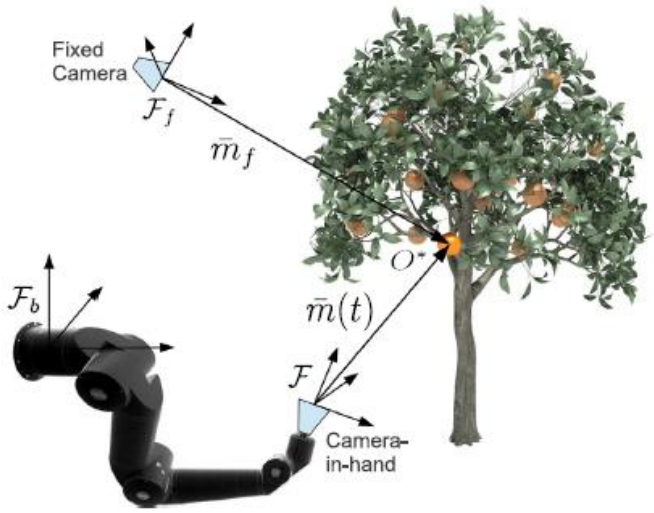
Estereoscopia ou Visão Estéreo consiste em registrar duas vistas de uma cena emulando as características vasculares do olho humano.

Segundo Fernandes (2010), a Estereoscopia é uma técnica utilizada, tanto em sistemas artificiais como biológicos, para a determinação de posicionamento tridimensional de objetos em uma cena. Esta técnica utiliza duas câmeras em posições diferentes, para a obtenção de um par de imagens da cena. Através da Figura 24 podemos perceber que um dado ponto da superfície observada vai se projetar em posições distintas nas duas imagens. Fazendo o "casamento" entre as imagens, é possível determinar, para cada ponto, o deslocamento acarretado pela mudança na posição de observação, ou seja, é possível obter o mapa de disparidades estereoscópicas.

Muitas pesquisas já possuem resultados significativos na extração de atributos

aplicáveis ao posicionamento de dispositivos para operação de colheita em cultivos de árvores frutíferas como apresentado na Figura 20.

Figura 20 - Posicionamento de braço por visão computacional.



Fonte: Mehta & Burks 2014.

Porém, nestas aplicações, o objeto de destaque possui característica que se destaca do conjunto observado, já entre as horticulturas há uma maior dificuldade devido a maior homogeneidade do conjunto, dificultando a segmentação de seus elementos. Podemos observar nas Figuras 21 e 22 estas características.

Figura. 21 - Classificação de frutos por segmentação e normalização.

Fonte: Jun Lu, Nong Sang. 2014.

Figura 22 - Classificação de hortícolas por segmentação e normalização.

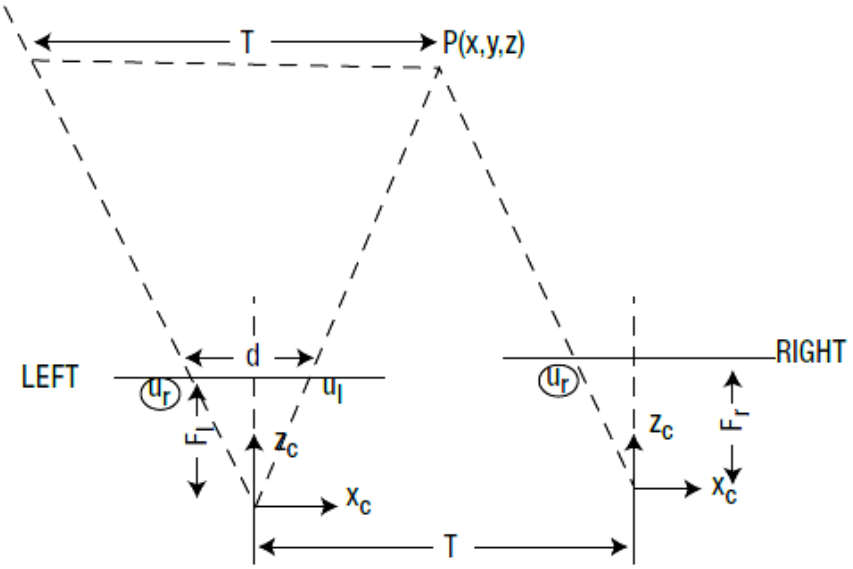
Fonte: Autor, 2019.

O algoritmo de extração dos atributos será elaborado de modo a identificar e classificar de forma aperfeiçoada as diferentes variedades de hortícolas, a fim de extrair os atributos adequados a cada umas delas como na classificação de frutos cítricos da Figura 21.

Em um sistema de Visão Computacional, o olho humano é substituído por dispositivo ótico de captura (câmeras) com a função desta emulação.

Em um sistema de Visão Estéreo duas câmeras são colocadas lado a lado com eixos ópticos paralelos. Os planos das imagens coincidem, mas possuem seus próprios sistemas de coordenadas. Na Figura 23 mostra o sistema de visão binocular.

Figura 23 - Sistema de Visão Binocular para obtenção de coordenadas espaciais.



Fonte: Brahmnhatt 2013.

Com o alinhamento dos planos de imagem, e assumindo que foram eliminadas as

distorções intrínsecas à câmera e estando os planos de imagem alinhados, os pontos *ul* e *ur* deverão estar sobre a mesma linha, permitindo uma avaliação direta e unidimensional da disparidade. Depois de encontradas as projeções de P nos dois planos de imagem, as distâncias *ul* e *ud* são medidas. Por norma, na visão computacional a origem do referencial está no canto superior esquerdo da imagem. Observando as posições do ponto P nas imagens, a disparidade é calculada como sendo:

$$d = ul - ur \tag{13}$$

Usando a concepção de similaridades entre triângulos, a profundidade Z do ponto é deduzida a partir da expressão:

$$\frac{d}{T} = \frac{f}{Z} \tag{14}$$

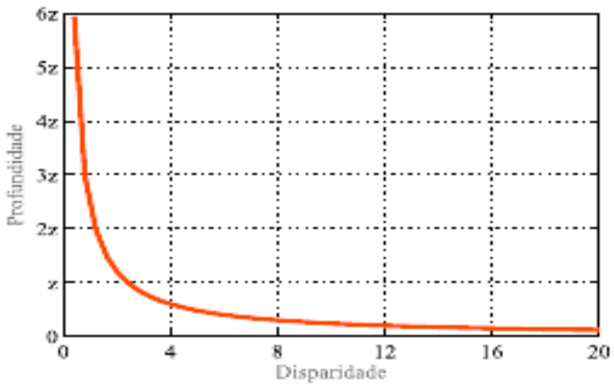
Logo:

$$Z = T * (\frac{f}{d}) \tag{15}$$

Na Figura 28 esboça-se a relação entre a disparidade e a profundidade. Quando temos um objeto mais próximo, temos uma disparidade maior. Isto resulta também em um menor erro no cálculo da profundidade.

A disparidade é um valor inteiro, porque a imagem estar discretizada e a sua menor unidade é 1 pixel. Assim sendo, quanto mais longe estiver o objeto maior será o erro na distância obtida, como é visível pela derivada acentuada quando a disparidade se aproxima de zero.

Figura 24 - Relação entre disparidade e profundidade.



Fonte:Neves, 2013

2.11.1 Captura Estéreo

A captura de imagens estéreo é realizada através da captura simultânea de duas câmeras. Uma característica de extrema importância é a distância entre as câmeras.

Para aplicações estéreo, as imagens a esquerda e a direita devem ser capturadas no mesmo instante. Isso é possível com sincronização de hardware, mas se a câmera estéreo não tem esse recurso, você pode pegar os quadros brutos usando o método *grab()* da classe *VideoCapture* e, em seguida, decodifica-se e armazenam-se os *frames* usando o método *retrieve ()*. Isso garante que ambos os quadros são capturados quase ao mesmo tempo.

2.11.2 Calibração

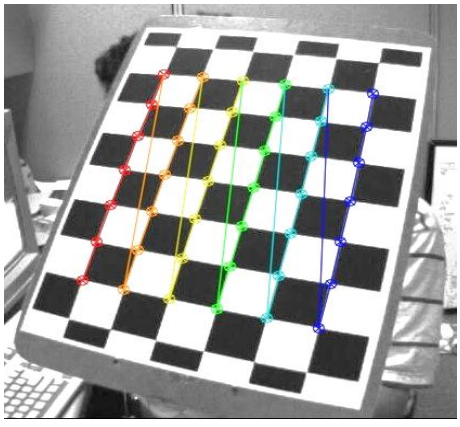
O processo de calibração é fundamental para se efetuar a medição de profundidade dos elementos em imagens.

Algumas estruturas estéreas são fornecidas já calibradas de fábrica, mas quando recorremos a estruturas estéreo com câmaras de mais baixo custo, a calibração deve ser realizada via software.

Nesta configuração, as câmeras devem estar posicionadas de forma a que os eixos principais sejam paralelos e os planos de imagem horizontalmente alinhados.

Existem diversas métodos e ferramentas de calibração de câmeras. O mais utilizado é a de tabuleiro de xadrez (*ChessBorder)*. Esta ferramenta é utilizada devido a facilidade de detecção dos cantos (*cornes*) dos quadrados do tabuleiro de xadrez. Na biblioteca a função *findCameraCorners()* é responsável por esta tarefa. A Figura 25 mostra um exemplo de tabuleiro de xadrez (*ChessBorder).*

Figura 25 - Tabuleiro de Xadrez.



Fonte: OpenCV

.

Para se obter a calibração de um sistema estéreo deve-se calibrar as câmeras individualmente. Esta operação tem como resultado a uma matriz de rotação **R**, um vetor de translação **T** e as matrizes essencial **E** e fundamental **F**. A função *stereoCalibrate()* da biblioteca OpenCV permite o cálculo destes parâmetros.

Para se calibrar individualmente as câmeras devemos capturar um conjunto de imagens das câmeras a esquerda e a direita e calibra-las individualmente gerando os arquivos de dados de calibração das câmeras.

Calibrações que envolvam poucas imagens, imagens de perspectivas pouco variadas ou em que o padrão de calibração se encontra demasiado longe da câmera são de pouca utilidade.

Assumindo que cada par de imagens direita-esquerda foi adquirido com o tabuleiro visível por ambas as câmeras, o final desse processo devolve o vetor de rotação e translação da câmera direita em relação à esquerda e otimiza os parâmetros de forma a minimizar o erro de reprojeção.

As etapas de calibração passam pela captura de instantâneos (snapshoot) das câmeras que serão utilizados para calibração do sistema. A calibração é feita individualmente (cada câmera) e após esta etapa, os dados de calibração individuais são consolidados em um arquivo de calibração para o sistema estéreo.

2.11.3 Retificação e disparidade por correspondência

Com a utilização de câmeras estéreo pode-se determinar a disparidade de um pixel em um par de imagens, descobrindo qual *pixel* na imagem da direita corresponde a um pixel na imagem da esquerda. Isto pressupõe um perfeito alinhamento vertical entre as imagens. Porém, em um sistema estéreo, mesmo com câmeras montadas em estrutura única, pequenos desalinhamentos verticais podem existir, quanto mais em sistemas que utilizam câmeras individuais.

O uso da calibração também minimiza esse problema calculando-se as matrizes de rotação **R** e de translação **T** da câmera esquerda em relação a câmera direita. Utilizando-se estas matrizes de transformação as duas imagens podem ser exatamente alinhadas. Na biblioteca OpenCV, este processo de alinhamento chama-se retificação estéreo, e é realizado pelas funções *StereoRectify()*, *initUndistortRectifyMap()* e *remap()*. *StereoRectify()* recebe as informações de calibração individual das câmeras, bem como da plataforma estéreo e fornece as saídas :

• R1 - Matriz de rotação aplicável à câmera esquerda para alinhá-la

• R2 - Matriz de rotação aplicável à câmera direita para alinhá-la

• P1 - Matriz aparente de alinhamento da câmera esquerda unida a uma translação para origem do sistema de coordenadas calibrado (sistema de coordenadas da câmera esquerda). Para a câmera esquerda [0 0 0]T

• P2 - Matriz aparente de alinhamento da câmera direita unida a uma translação para origem do sistema de coordenadas calibrado (sistema de coordenadas da câmera esquerda). Para a câmera direita [-T 0 0] T onde T é a distância entre as origens das câmeras

• Q matriz de mapeamento de disparidade-para-profundidade (função *reprojectImageTo3d ()*)

Depois de obter essas transformações, você deve aplicá-las à imagem correspondente. A função OpenCV *initUndistortRectifyMap()* calcula mapas de pixels tomando as matrizes de saída da função *stereoRectify()* como entrada. A função *remap()* aplica o mapa de pixels a uma Imagem para corrigi-la.

2.11.4 Cálculo de Disparidades

Após estes procedimentos obtém-se pixels correspondentes nas duas imagens e calcular a disparidade. Na biblioteca OpenCV o algoritmo *Semi-Global Block Matching* (SGBM) da classe *StereoSGBM* utiliza um técnica de programação dinâmica para tornar a correspondência mais robusta. O algoritmo utiliza parâmetros associados tais como:

- *SADWindowSize*: O tamanho do bloco para corresponder (deve ser um número ímpar)
- P1 e P2: parâmetros que controlam a suavidade da disparidade calculada.
- *speckleWindowSize* e *speckleRange*: Esses parâmetros Controlam o filtro de *speckle*. *SpeckleWindowSize* indica o tamanho máximo de uma região de disparidade para que seja considerada como um *speckle*, enquanto *speckleRange* indica a máxima variação.
- *minDisparity* e *numberOfDisparities:* Juntos controlam a configuração de disparidade estéreo.

2.11.5 Detecção de Distância.

Para a mensuração da distância com sistema de estéreo visão a função *reprojectImageTo3d()* faz o calcula das coordenadas 3D tomando a disparidade e a matriz de mapeamento disparidade-profundidade **Q** gerada por *stereoRectify* () .

No processo de medição de distâncias por estéreovisão, uma série de tarefas de calibração do sistema devem ser efetuadas.

3. METODOLOGIA

O presente trabalho propõe o desenvolvimento de um sistema de visão computacional para máquina agrícola, capaz de identificar e localizar a hortícola da alface. Este sistema está sendo denominado de SVAM – Sistema de Visão Artificial para Máquinas.

Este sistema, registrado sob o protocolo BR 10 2018 010359 8, é parte integrando da patente número de depósito BR1020130197173 - Colhedora Multifuncional de Hortícolas.

Para isto faz-se necessário a construção de um aparato de captura de imagens utilizando câmeras e do desenvolvimento de um sistema *software* composto de conjunto de algoritmos e responsável pelo ajuste de parâmetros de captura de imagens, decodificação de dados, detecção de objetos e estimação de distância.

Para melhor apresentação do sistema, este será dividido em três etapas denominadas Captura, Detecção e Estimação de Distâncias.

3.1 Etapa de Captura

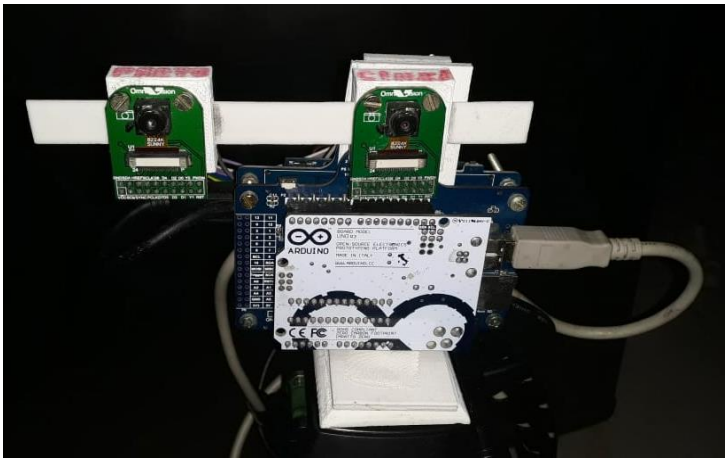
Para este trabalho define-se módulo de Captura como sendo o elemento primário de obtenção de imagens. Este é formado pelo seu *hardware* de captura (câmeras, dispositivo de interface e placa controladora) e por algoritmos de controle (configurador de interface, transmissão de dados, decodificação de imagens e armazenamento, captura continua).

3.1.1 Hardware de Captura

Constituído por um par de câmeras, interface de conexão e placa controladora conforme apresentado na Figura 26, é o elemento primário de recepção de imagens.

Possuem base para encaixe/desencaixe de fácil manuseio e estrutura deslizante que permite a regulagem de posicionamento das câmeras para ajuste da distância de base entre as câmeras.

Figura 26– Hardware de Captura.



Fonte: Autor (2019).

A câmera utilizada foi a OV2640 conforme apresentado na Figura 27. Conforme dados técnicos do dispositivo (Anexo 1), esta câmera CMOS com total funcionalidade em UXGA 1632x1232 de resolução possui dispositivo *Serial Camera Control Bus* (SCCB) com processamento em único chip fornecendo imagens *full-frame*, subamostradas ou dimensionadas com 8 a 10 bits em diversos formatos. Opera com taxa de 15 / 30 FPS, com controle completo sobre a qualidade da imagem, formato de saída e transferência de dados. Todas as funções de processamento de imagem, incluindo controle de exposição, gama, equilíbrio de branco, saturação de cor, controle de matiz, cancelamento de pixel branco, cancelamento de ruído e outras funções, são programáveis através da interface SCCB, justificativa principal para a utilização desta câmera no hardware de captura.

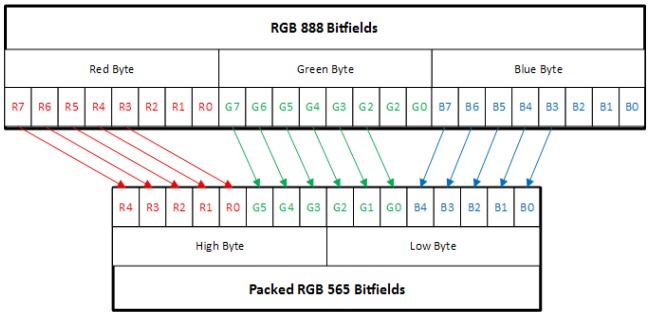
Figura 27 – Câmera OV2640.

Fonte: Ominivision (2015)..

A mesma apresentar os requisitos necessários a etapa de captura das imagens reduzindo a sobrecarga de processamento em etapas posteriores, porém, faz-se necessário implementar uma interface de controle para gerenciamento das operações.

Esta interface, pode ser diretamente conectada a controlador, como mostra a Figura 28.

Figura 28 – Codificação dados imagem.



Fonte: Packed.

Esta interface, de controle (em geral microprocessada) utilizada para configuração de registadores internos da câmera (formato de saída, controle de exposição, balanço de branco, ganho, entre outras características).

Podemos conectar diretamente a câmera ao dispositivo controlador, através das conexões do que permite a leitura direta dos dados da câmera, porém, apresenta alta demanda de tempo entre captura dos *pixels* e transferência dos dados ao sistema. A solução encontrada para contornar esta dificuldade foi a utilização do módulo de interface ArduCam.

A *Shield* para módulos de câmeras (*ArduCam),* permite o interfaceamento da câmera com o modulo de controle adequando o controle de captura com a transferência de dados. É responsável pelo gerenciamento da varredura de *pixels* da câmera, deixando o módulo de controle responsável somente pela transferência e gerenciamento do local. A Figura 29 apresenta está placa de controle.

Figura 29 - Placa de controle de câmera universal para Arduino – ArduCAM.

Fonte: Arducam (2019).

O fabricante desta interface disponibiliza uma biblioteca de código-fonte de software para utilização com diversos dispositivos, incluindo as plataformas de prototipagem Arduino.

A interface de prototipagem Arduino Uno é comumente utilizada como modulo de controle. Podem ser utilizadas outras arquiteturas de processamento como Arduino Mega ou *Raspberry pi*. A programação da plataforma Arduino foi realizada pelo IDE fornecida pelo fabricante da plataforma.

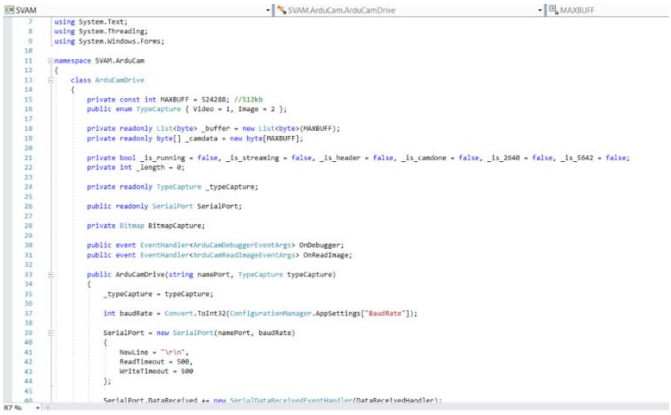
3.1.2 Interface de Captura

Para obtenção das imagens utilizando o *hardware* de captura foram implementadas algoritmos controle no módulo Arduino Uno apresentados na Figura 30 que possibilitam configurar dados das câmera (resolução, taxa de transferência e formato de dados) e no sistema operando em alto nível (*drive* de comunicação, funções de decodificação dos dados de imagem, funções de arquivamento das imagens em disco) na Figura 31.

Figura 30 – Algoritmo controle no módulo Arduino Uno.

Fonte: Arduino.

Figura 31 – Sistema de operação.

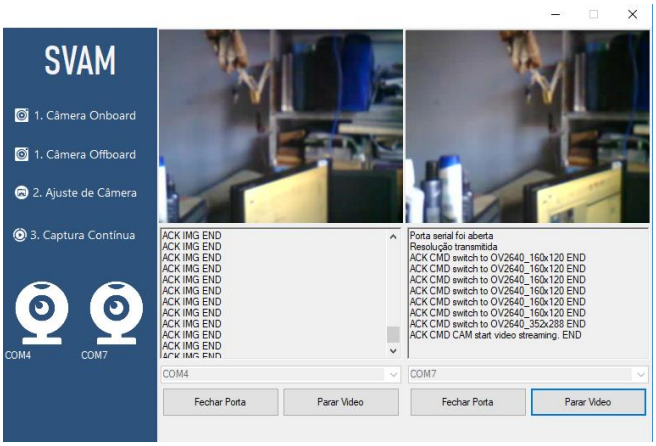


Fonte: Autor, 2019.

No sistema em alto nível, temos a opção de utilizar o *hardware* de Captura definido neste trabalho (utilizando a opção de Câmera *Offboard*), como também optar pelo uso de outro dispositivo de captura que possa conectar-se ao sistema via USB (opção de Câmera *Onboard*).

O funcionamento destes módulos está apresentado na Figura 32.

Figura 32 – Funcionamento dos módulos.



Fonte: Autor, 2019.

3.2 Detecção

A detecção de objetos engloba duas características distintas do processamento de imagens, a classificação de objetos e a localização destes na imagem.

Como o problema é altamente complexo, algoritmos de aprendizado de máquina

tornaram-se parte integrante da pesquisa de detecção de objetos. (JENSEN, SELVIK, 2016, p. 2).

A complexidade do problema de detecção proposto neste trabalho está na detecção de um objeto de morfologia complexa (a alface) em campo (diferentes condições de ambientes) e em tempo real (com tempo de resposta crítico).

Tradicionalmente, reconhecimento de objetos para aplicações de visão de máquina baseia-se em combinações de processos, como limiar, mascaramento, segmentação de cores, detecção de bordas e filtragem. Essas abordagens funcionam bem quando a iluminação é previsível, a aparência do objeto está bem definida e oclusões são mínimas, como em ambientes de fábrica. No entanto, o ambiente agrícola é frequentemente o caso em que nenhuma destas condições são cumpridas (WILLIAMS et al, 2019 p. 142).

Para um sistema que irá operar nestas condições é imprescindível que ele possua características de assertividade e velocidade apropriados. Precisamos de uma estratégia de ação que seja eficiente para estas condições.

Ainda segundo (WILLIAMS et al, 2019 p. 04), métodos manuais tais como *Histogram of Oriented Gradients* (HOG), *Scale-Invariant Feature Transform* (SIFT), combinados com classificadores como *Adaptive Boosting* (AdaBoost) ou *Support Vector Machines* (SVM) tem a desvantagem de possuir um alto custo computacional.

Uma maneira de resolver esse problema seria através da utilização de Redes Neurais Artificiais (ANN). Em teoria, podemos fazer uso de Redes Neurais convencionais para analisar imagens, mas na prática, analisando através da perspectiva computacional é altamente dispendioso. Por exemplo, uma rede neural convencional tentando processar uma imagem pequena (30x30 pixels) ainda precisaria de 0,5 milhões de parâmetros e 900 entradas. (RAVINDRA, REIS, 2017).

Atualmente, é o de mais inovador em sistemas de reconhecimento de objetos em tempo real, está de acordo com um compromisso entre velocidade e assertividade (KÜHNE, 2018).

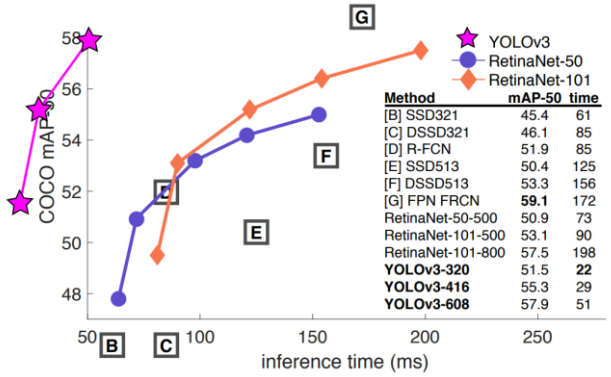
Para o atendimento das características de velocidade e assertividade necessárias ao sistema, precisamos modelar uma rede de alto desempenho capaz de processar imagens em tempo real e detectar os objetos de interesse com grau de acerto adequado.

As redes convolucionais se destacam na tarefa de processamento de altos volumes de dados como os de imagens e desempenho com baixo tempo de resposta.

Em termos de desempenho, o YOLOv3 se destaca pela velocidade de detecção. A rede YOLO tem resposta da ordem de 50ms.

De acordo com o trabalho de Redmon e Farhadi (2018b), além do aumento na capacidade de detectar objetos pequenos, a versão 3 chega a ser cerca de três vezes mais rápida em relação ao algoritmo *Single Shot Multibox Detector* - SSD. Dessa forma, o YOLOv3 continua sendo especialista na execução em tempo real. A Figura 33 apresenta um gráfico de acurácia x velocidade. Pode-se perceber o quão rápido é o tempo de execução possui o YOLOv3, considerando IoU de 50%. (MARQUES, 2018, p. 29).

Figura 33– Comparação desempenho da YOLOv3 (acurácia x velocidade).



Fonte – Redmon e Farhadi (2018b).

O algoritmo de detecção escrito para o desenvolvimento do sistema utiliza o método *DnnInvoke.ReadNetFromDarknet*, da plataforma de desenvolvimento EMGU CV. Ele recebe os frames da imagem capturados pelo Hardware de Captura e efetua a detecção do objeto desejado. Este método necessita de dois parâmetros principais. O *cfgFile*, arquivo .cfg com a descrição textual da arquitetura da rede e o *darknetModel* arquivo de pesos (*weights*) de aprendizado da rede.

O arquivo de pesos (*weights*) é obtido pelo treinamento que deve ser efetuado em um conjunto de dadas de imagens previamente selecionado e preparado contendo o objeto a ser detectado.

O treinamento da rede Darknet Yolo será apresentado na próxima seção.

3.2.1 Configurando o ambiente

Para o treinamento da rede utilizando a arquitetura *Darknet* há a necessidade da instalação do *framework*. Ele pode ser obtido em https://github.com/pjreddie/darknet. Esta plataforma possui duas dependências principais:

- OpenCV: para operação com dados de imagem;

- CUDA: para computação em GPU.

Estas dependências são opcionais, mas promovem uma gama de facilidades de uso como aumento da variedade de imagens suportadas e aceleração do processamento do treino.

Para compilação do sistema base utilizou-se a plataforma de desenvolvimento C# MSVC 2015/2017/2019 em um computador Intel® Core™ i7-8550U CPU @ 1,80Ghz com 16,0 GB(RAM) com Windows 10 Pro - Sistema Operacional 64 bits e adaptador de vídeo NVIDIA GeForce MX150.

A instalação com CUDA permite acelerar o processamento da ordem de 500 vezes desde que corretamente configurado. Para isto, deve-se instalar o pacote CUDA 10.0 da NVIDIA, (https://developer.nvidia.com/cuda-toolkit-archive), e alterar a linha GPU=1 no arquivo *Makefile*.

Para utilização da OpenCV, deve-se primeiramente instalar a biblioteca de funções para em seguida compila-la junto a *Darknet*. A configuração de uso da OpenCV é feita no mesmo *Makefile* anteriormente citado na linha OPENCV = 1. A biblioteca utilizada nesta implementação é a versão 2.4.10.

Após a instalação da plataforma podemos utilizar seus modelos pré-treinados para detecção das classes. Uma classe é uma categoria de objeto identificável pelo classificador.

3.2.2 Treinamento da rede

A *Darknet* possui uma série de modelos com categorias pré-treinadas, porém, nossa necessidade é o reconhecimento de uma categoria específica de objeto. Para isso, devemos treinar a rede com dados próprios.

O treinamento é feito apresentando-se um conjunto de imagens preparadas previamente. Este preparo consiste em gerar um arquivo .txt para cada imagem do treinamento localizado no mesmo diretório das imagens, contendo valore de coordenadas do centro e dados da largura e altura do objeto. A figura 34 demonstra a descrição do objeto contém também o rótulo da classe do objeto.

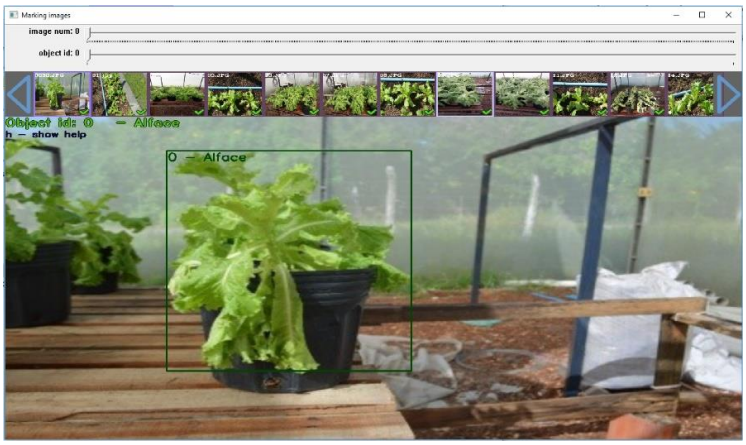
Figura 34 - Arquivo de referência da imagem.

```
1 0.716797 0.395833 0.216406 0.147222
0 0.687109 0.379167 0.255469 0.158333
1 0.420312 0.395833 0.140625 0.166667
```

Fonte: AlexeyAB, 2019.

Estes dados podem ser gerados manualmente ou com uso de ferramentas. Neste foi utilizado o *Yolo_Mark*, uma ferramenta visual para geração destes arquivos, conforme a Figura 35, fornecido por AlexeyAB e obtido em https://github.com/AlexeyAB/Yolo_mark,.

Figura 35 - Ferramenta Yolo_mark.



Fonte: Autor, 2019.

Recomenda-se que para um bom treinamento, que o conjunto de imagem seja composto por pelo menos 300 imagens de cada categoria a ser identificada.

Para o treinamento da rede, assim como para a avaliação dos resultados, foram coletados imagens da alface em três fontes distintas: buscas na internet, cultivo de campo realizado na horta didática pertencente ao departamento de Fitotecnia e em cultivo protegido(estufa), localizada em área próxima à estação meteorológica, pertencente ao departamento de Engenharia Agrícola, ambos vinculados a Universidade Federal do Ceará .

As imagens obtidas no cultivo de campo e no cultivo em estufa foram obtidas com câmera Nikon modelo D3200, conforme a Figura 36, e com o próprio Hardware de Captura já em fase de testes da Etapa de captura, conforme a Figura 37.

Figura 36 – Utilizando o aparato de Hardware de Captura.

Fonte: Autor, 2019.

Figura 37– Utilizando a câmera Nikon modelo D3200.

Fonte: Autor, 2019.

Com o conjunto de imagens definidas, criam-se os arquivos necessários ao treinamento. São eles: o arquivo de rótulos (*.names*) com o nome das classes a serem treinadas (no nosso caso Alface), de dados (*.data*) com informações da quantidade de classes, *set* imagens de treino e validação (arquivos*train.txt* e *test.txt*), definição das pastas dos dados de treinamento e saída dos pesos do aprendizado(*backup/*).

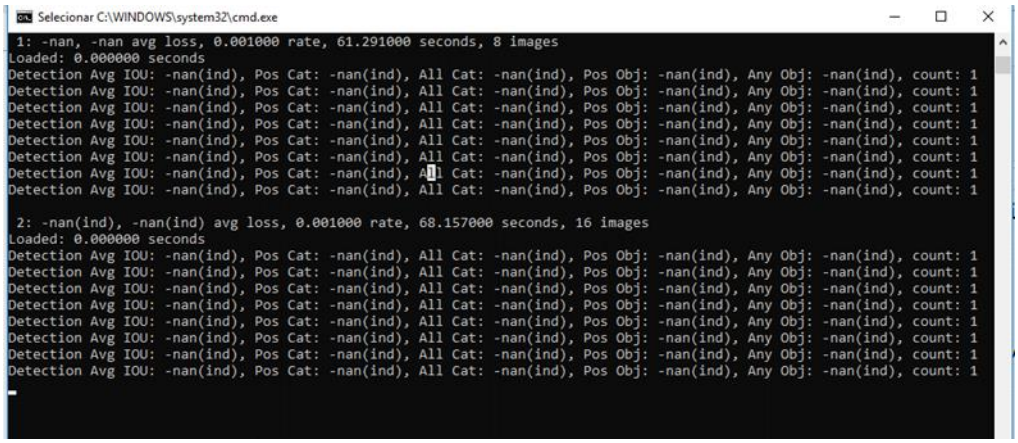
O próximo passo do treinamento é a configuração do arquivo da rede (*.cfg*). Para uma configuração de treino padrão, devem ser alteradas as linhas referenciadas abaixo definindo-as com os seguintes valores:

- *batch*=64;
- *subdivisions*=8;
- *max_batches* para número de classes*2000(no nosso caso, 2000 visto que temos somente uma classe a ser treinada);
- *steps* para 80% e 90% de *max_batches*(no nosso caso *steps*=1600,1800)
- *classes*=1 nas 3 *[yolo]-layers* do arquivo;
- [*filters*=18] (correspondendo ao (número de classes + 5) x3 nas 3 camadas [*convolutional*] que antecedem a *[yolo] layer*.

Configurada a rede e o conjunto de dados há a necessidade de uma pré-carga dos pesos das camadas convolucionais para iniciar o treinamento. O arquivo de pesos utilizado nesta fase é o *darknet53.conv.74*, disponível em https://pjreddie.com/media/files.

O treinamento da rede é iniciando com a execução do programa *Darknet*, pela linha de comando *darknet detector train data/obj.data yolo-obj.cfg darknet53.conv.74*. Os resultados parciais do treinamento são mostrados conforme a figura 38.

Figura 38 – Resultado parcial do treinamento.



Fonte: Autor, 2019.

Durante o treinamento, haverá indicações de erro que, quando não diminuir mais

o valor de avg e deve-se parar o treinamento. Estes dados são verificados nas linhas de saída do processamento. Nesta linha temos:

9002: 0.211667, 0.060730 avg, 0.001000 rate, 3.868000 segundos, 576128 imagens Carregados: 0.000000 segundos

Onde:

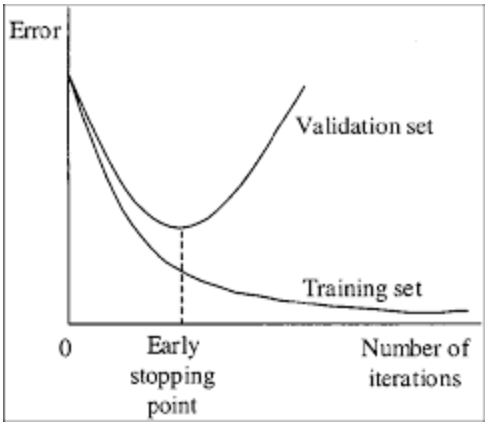
9002 - número de iteração (número de lote)

0,060730 avg - perda média (erro)

Usualmente, são necessárias 2000 iterações para cada classe (objeto). Esta indicação demonstra a existência de erros. Durante o treinamento, são apresentados vários indicadores de perda média (average loss). O treinamento deve ser encerrado quando não houver mais redução do Avg (0.XXXXXXX avg). O valor médio pode ser de 0,05 (para um modelo pequeno e um conjunto de dados fácil) a 3,0 (para um modelo grande com conjunto de dados complexo).

Porém, dois erros podem ocorrer: se durante o treinamento você vê valores *-nan* para o campo de perda média (Avg), então o treinamento está ocorrendo com erro. Um outro erro que pode ocorrer é o *Overfitting* quando você pode detectar objetos em imagens do conjunto de dados de treinamento, mas não pode detectar objetos em quaisquer outras imagens. Os pesos ideais são os obtidos no *Early Stopping Point* Figura 39.

Figura 39 – Gráfico de erro apresentando. Overfitting e Early Stopping.



Fonte: AlexeyAB/darknet (2016).

O treinamento também pode ser acompanhado no *chart* de precisão das médias (*mAP*)

Analisando os valores apresentados na execução, temos o termo *-nan* nas linhas

de saída que indica o erro de cálculo. Um outro ponto a se observar é o tempo de treinamento. No caso em estudo, cada interação está consumindo uma média de 68 segundos, o que nos dá uma previsão de cerca de 38 horas de treinamento, além do volume de dados gerado em memória para este processo.

Isto ocorre devido a falha na configuração da GPU, na compilação do *framework darknet*. Este fato é comum visto que o próprio desenvolvedor da *darknet* alerta para este evento.

A ausência de utilização da GPU inviabilizou o treinamento nesta máquina *desktop* . Este fato levou a busca de uma solução de processamento de dados que otimizasse o treinamento da rede.

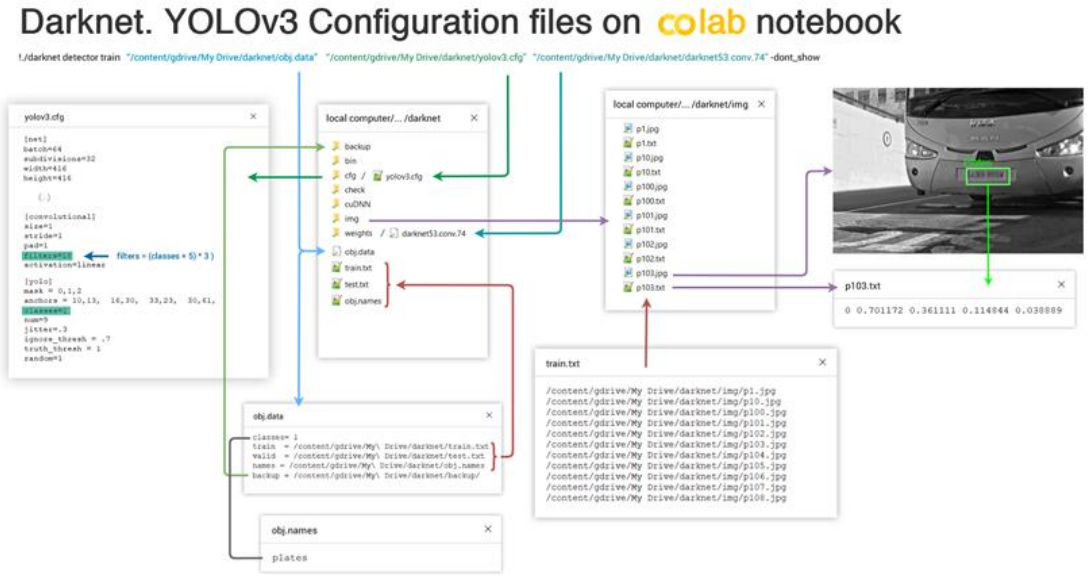
A solução encontrada foi a utilização de uma máquina virtual (MV) em nuvem.

Existe disponível diversos serviços de máquina virtual como o *Microsoft Azure, Pipeline Notebook* e outros. O *Google Colab* é um serviço de nuvem gratuito para ensino e pesquisa de aprendizado de máquina. Ele fornece um *dashboot* em tempo de execução totalmente configurável para aprendizagem profunda e acesso livre a uma GPU robusta (Ibáñez, 2017).

O *Google Colab* é um ambiente *Jupyter notebook* configurável e de uso gratuito com algumas limitações tais como tempo limite de uso em 12h e conjunto de dados limitado a 550Mb, porém, com a vantagem de utilizar uma GPU com 12Gb de *RAM* otimizada. Estas limitações ainda podem ser resolvidas “linkando” o Colab com o *Google drive*.

A configuração do treinamento da rede no Colab passa pela atualização do sistema, conecção do Colab ao Google Drive, instalação e verificação do CUDA e do framework *darknet* e da OpenCV para então efetuar o treinamento da rede. A preparação e configuração dos dados de treinamento no Colab segue a estrutura apresentada na Figura 40.

Figura 40 - Configuração dos dados de treinamento no Colab.

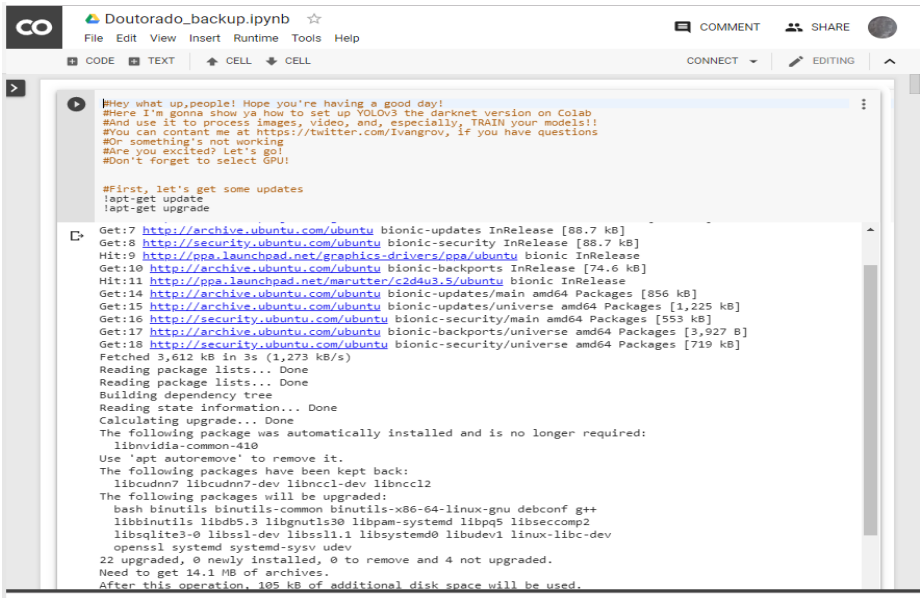


Fonte: Colab (2019).

A sequência de configuração passo a passo do Colab para utilização da *Darknet Yolo* pode ser encontrada em <https://colab.research.google.com/drive/1lTGZsfMaGUpBG4inDIQwIJVW476ibXk_#scrollTo=wkzMqLZV-rF5>. Uma configuração alternativa pode ser encontrada em <https://github.com/ivangrov/YOLOv3GoogleColab/blob/master/YOLOv3_GoogleColab.ipynb.>

A Figura 41 apresenta o arquivo de configuração do Colab.

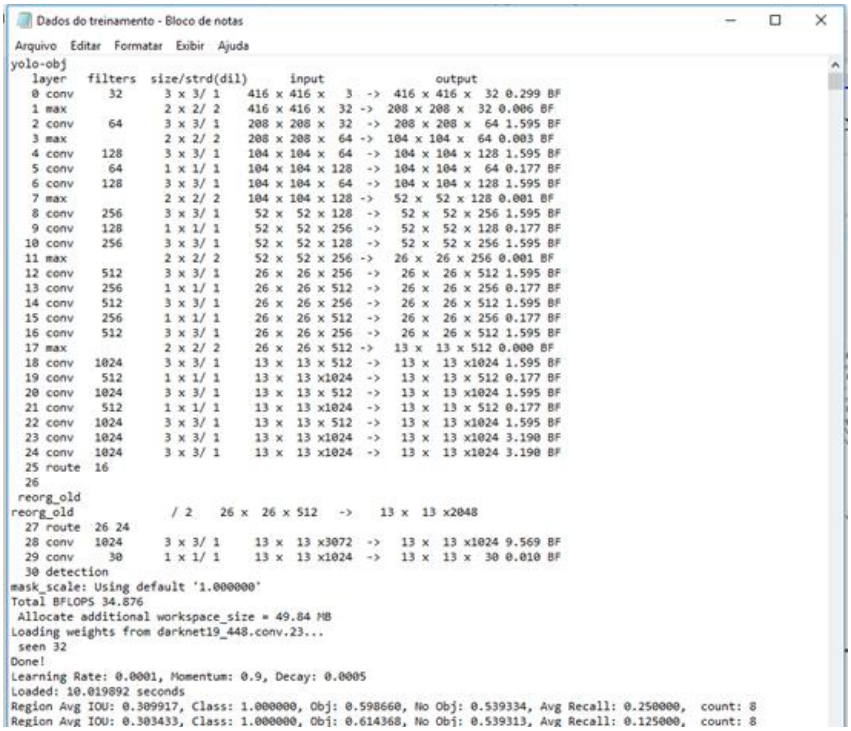
Figura 41 – Arquivo de configuração do Colab.



Fonte: Colab.

Após a configuração do Colab e de todos os arquivos de dados do treinamento, deu-se início ao treinamento da rede. As Figuras 42 e 43 abaixo apresentam o início do treinamento com a carga das camadas convolucionais e os dados de erro médio do treinamento.

Figura 42 – Início do treinamento no Colab.



Fonte: Autor, 2019.

Figura 43 - Dados de erro médio do treinamento.

```
 194: 0.125247, 0.114326 avg loss, 0.001000 rate, 6.227332 seconds, 12416 images
Loaded: 0.000056 seconds
Region Avg IOU: 0.745297, Class: 1.000000, Obj: 0.626495, No Obj: 0.004845, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.754397, Class: 1.000000, Obj: 0.624803, No Obj: 0.004784, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.815500, Class: 1.000000, Obj: 0.670629, No Obj: 0.005403, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.771100, Class: 1.000000, Obj: 0.449201, No Obj: 0.005149, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.725347, Class: 1.000000, Obj: 0.532527, No Obj: 0.005219, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.807489, Class: 1.000000, Obj: 0.706599, No Obj: 0.004636, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.767799, Class: 1.000000, Obj: 0.554045, No Obj: 0.004351, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.825639, Class: 1.000000, Obj: 0.663374, No Obj: 0.004676, Avg Recall: 1.000000,  count: 8

 195: 0.107433, 0.113637 avg loss, 0.001000 rate, 6.220009 seconds, 12480 images
Loaded: 0.000034 seconds
Region Avg IOU: 0.741049, Class: 1.000000, Obj: 0.689309, No Obj: 0.005597, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.740259, Class: 1.000000, Obj: 0.702995, No Obj: 0.005407, Avg Recall: 0.875000,  count: 8
Region Avg IOU: 0.736734, Class: 1.000000, Obj: 0.548446, No Obj: 0.006204, Avg Recall: 1.000000,  count: 7
Region Avg IOU: 0.728034, Class: 1.000000, Obj: 0.629438, No Obj: 0.005345, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.764374, Class: 1.000000, Obj: 0.796212, No Obj: 0.006015, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.737193, Class: 1.000000, Obj: 0.565166, No Obj: 0.005060, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.744064, Class: 1.000000, Obj: 0.718873, No Obj: 0.005731, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.768768, Class: 1.000000, Obj: 0.465140, No Obj: 0.004758, Avg Recall: 1.000000,  count: 8

 196: 0.110109, 0.113284 avg loss, 0.001000 rate, 6.222400 seconds, 12544 images
Loaded: 0.000040 seconds
Region Avg IOU: 0.719524, Class: 1.000000, Obj: 0.575982, No Obj: 0.005390, Avg Recall: 0.875000,  count: 8
Region Avg IOU: 0.769940, Class: 1.000000, Obj: 0.464106, No Obj: 0.004441, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.750109, Class: 1.000000, Obj: 0.721843, No Obj: 0.005452, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.752062, Class: 1.000000, Obj: 0.681781, No Obj: 0.005133, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.779282, Class: 1.000000, Obj: 0.800951, No Obj: 0.004651, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.804845, Class: 1.000000, Obj: 0.494564, No Obj: 0.005817, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.738625, Class: 1.000000, Obj: 0.590503, No Obj: 0.005088, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.744307, Class: 1.000000, Obj: 0.726909, No Obj: 0.005119, Avg Recall: 1.000000,  count: 8
```

Fonte: Autor, 2019.

Podemos perceber na Figura 50, que o valor do erro médio, se mostra mais

adequado a uma condição de aprendizado da rede. Houve também melhoria do tempo de processamento das interações.

Com o treinamento da rede efetuado, são gerados os valores de pesos (*weigths*) localizados na pasta backup. De acordo com os dados do treinamento, pode se verificar que o Avg (*Average loss*) calculado na última interação é de 0,007964.

3.3 Estimação de Distâncias

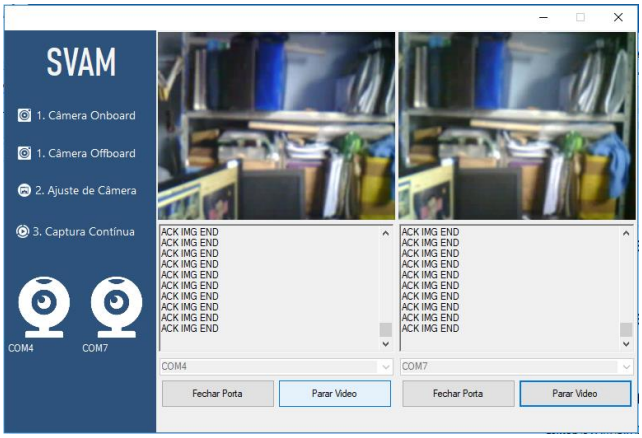
Para a estimar a distâncias com melhor acurácia utiliza-se de um sistema de visão estéreo com *hardware* e conjunto de operações de processamento de imagens (captura estéreo, calibração, retificação, cálculo de disparidades e detecção de distância).

Como já definimos o *hardware*, aqui são descritas as operações de processamento implementadas.

3.3.1 Captura Estéreo

O sistema SVAM utiliza a informação de saída do *buffer* das câmeras, para efetuar a captura. Primeiramente, deve-se determinar quais câmeras serão utilizadas na captura, se *Onboard* (para cameras USB tipo *webcam*) ou *Offboar* (para cameras serial OV2640). Na figura 44 observa-se a captura da imagem estéreo pelo sistema.

Figura 44 – Captura de imagem estéreo pelo sistema.



Fonte: SVAM.

Para a captura das imagens foram utilizadas as funções e estruturas de dados da biblioteca *EMGUCV*. Com a *class Emgu.CV.VideoCapture*, podemos definir as funções de

captura das câmeras e importá-las em uma matriz (*Mat*) para operações em seus pixels.

A figura 45 apresenta trecho do código para exibição de vídeo em *EMGU*.

Figura 45 – Trecho do código de exibição de vídeo em EMGU.

```
private void FormStereoCameraCalibration_Load(object sender, EventArgs e)
{
    m_formMain = (FormMain)Application.OpenForms[0];

    m_pathFileQMatrix = ConfigurationManager.AppSettings["FileQMatrix"];
    m_pathFileImagesRight = ConfigurationManager.AppSettings["FileImagesRight"];
    m_pathFileImagesLeft = ConfigurationManager.AppSettings["FileImagesLeft"];
    m_pathFolderImages = ConfigurationManager.AppSettings["FolderImages"];

    if(m_formMain.CamL.Value.Tipo ==  CamInfo.TipoCam.Onboard)
    {
        m_videoCaptureL = new VideoCapture(int.Parse(m_formMain.CamL.Value.Codigo));
        m_videoCaptureR = new VideoCapture(int.Parse(m_formMain.CamR.Value.Codigo));
        Application.Idle += ProcessFrameOnboard;
    }
    else
    {
        m_arduCamVideoCaptureL = new ArduCamDrive(m_formMain.CamL.Value.Codigo, ArduCamDrive.TypeCapture.Video);
        m_arduCamVideoCaptureR = new ArduCamDrive(m_formMain.CamR.Value.Codigo, ArduCamDrive.TypeCapture.Video);

        m_arduCamVideoCaptureL.OpenSerialPort();
        m_arduCamVideoCaptureR.OpenSerialPort();

        m_arduCamVideoCaptureL.StartCapture();
        m_arduCamVideoCaptureR.StartCapture();

        Application.Idle += ProcessFrameOffboard;
    }
}
```

Fonte: Autor, 2019.

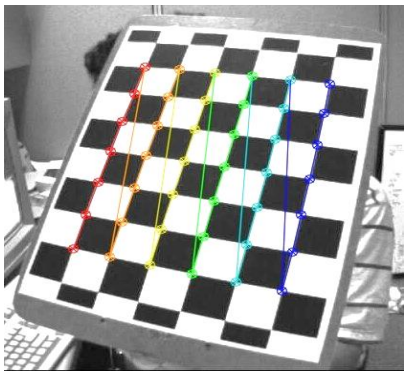
Definido o padrão de câmera, selecionamos a opção de Ajuste de Câmera para iniciarmos a calibração do sistema.

A captura de câmeras em estéreo visão deve ser ajustada para ocorrer simultaneamente na câmera a esquerda e a direita. Então, o salvamento das imagens ocorre somente se um par de imagens (a esquerda e a direita) do *chessborder* for detectado nas câmeras.

3.3.2 Calibração

Existem inúmeros métodos e ferramentas de calibração de câmeras. O mais utilizado é a de tabuleiro de xadrez (*ChessBorder*). Esta ferramenta é utilizada devido a facilidade de detecção dos cantos (*cornes*) dos quadrados do tabuleiro de xadrez. Na biblioteca o método *CvInvoke.FindChessboardCorners()* é responsável por esta tarefa. A Figura 46 mostra um exemplo de tabuleiro de xadrez (*ChessBorder*).

Figura 46 - Tabuleiro de Xadrez.

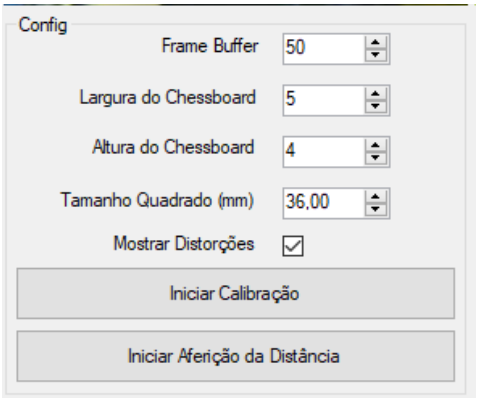


Fonte: OpenCV

Para garantir a simultaneidade na captura das imagens, necessária a operação de calibração do sistema, quando é efetuada a detecção de cantos nas imagens sincronizadamente o salvamento das mesmas é efetuado.

Para este procedimento, deve-se definir o número de pontos internos (Horizontal e Vertical) do *chesboarder,* e o tamanho do quadrado em mm. Para esta operação foi utilizado um de parâmetros 6 x 5 x 36 mm (Figura 47).

Figura 47 – Configuração da calibração.

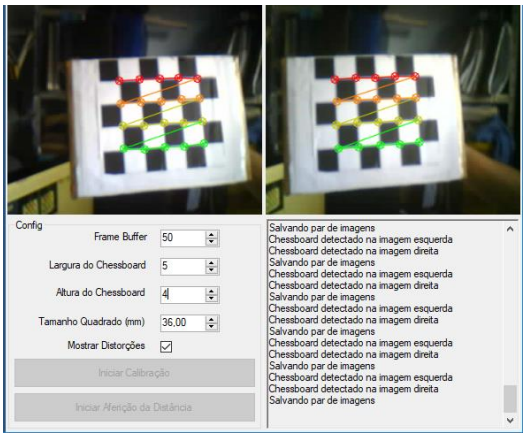


Fonte: Autor, 2019.

No caso de se utilizar outra configuração de *Chessboarder,* pode-se efetuar uma nova calibração alterando-se os valores nos campos. O campo *Frame Buffer* define o número de pares de imagem a serem capturadas.

Nesta configuração, como mostra a Figura 48, as câmeras devem estar posicionadas de forma que os eixos principais sejam paralelos e os planos de imagem horizontalmente alinhados.

Figura 48 – Processo de calibração no sistema.



Fonte: Autor, 2019.

Após a captura de instantâneos (*snapshoot*) temos como resultado uma matriz de rotação **R**, um vetor de translação **T** e as matrizes essencial **E** e fundamental **F**. As funções *LeftCameraCalibrate()* e *RightCameraCalibrate()* da biblioteca EMGUCV permitem o cálculo destes parâmetros.

As câmeras são calibradas individualmente com os conjuntos de imagens das câmeras a esquerda e a direita e para depois serem calibradas em estéreo através do método *CvInvoke.StereoCalibrate()*.

Assumindo que cada par de imagens direita/esquerda foi adquirido com o tabuleiro visível por ambas as câmeras, o final desse processo devolve o vetor de rotação e translação da câmera direita em relação à esquerda e otimiza os parâmetros de forma a minimizar o erro de reprojeção.

3.3.3 Retificação e disparidade por correspondência

Com a calibração estéreo pode-se determinar a disparidade de um pixel em um par de imagens, descobrindo qual pixel na imagem da direita corresponde a um pixel na imagem da esquerda. Isto pressupõe um perfeito alinhamento vertical entre as imagens. Porém, em um sistema estéreo, mesmo com câmeras montadas em estrutura única, pequenos desalinhamentos verticais podem existir, quanto mais em sistemas que utilizam câmeras individuais.

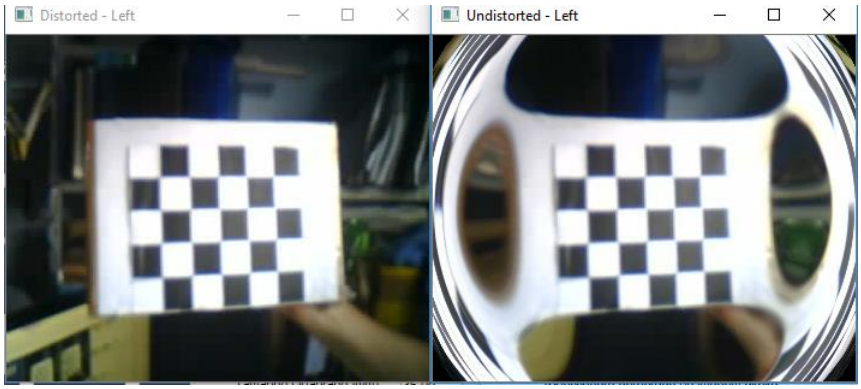
A calibração também pode resolver esse problema calculando-se as matrizes de rotação **R** e de translação **T** da câmera esquerda em relação a câmera direita. Utilizando-se estas matrizes de transformação, as duas imagens podem ser exatamente alinhadas. Na

EMGU CV, este processo de alinhamento chama-se retificação estéreo, e é realizado pela função *StereoCalibrateAndRectify()* em conjunto com as funções *CvInvoke.StereoCalibrate(), CvInvoke.StereoRectify()*. Estas funções recebem as informações de calibração individual e estéreo das câmeras (R1, R2, P1, P2), e fornecem as saídas(Q) sendo:

- R1 - Matriz de rotação aplicável à câmera esquerda para alinhá-la
- R2 - Matriz de rotação aplicável à câmera direita para alinhá-la
- P1 - Matriz aparente de alinhamento da câmera esquerda unida a uma translação para origem do sistema de coordenadas calibrado (sistema de coordenadas da câmera esquerda). Para a câmera esquerda [0 0 0]T
- P2 - Matriz aparente de alinhamento da câmera direita para unida a uma translação para origem do sistema de coordenadas calibrado (sistema de coordenadas da câmera esquerda). Para a câmera direita [-T 0 0] T onde T é a distância entre as origens das câmeras
- Q matriz de mapeamento de disparidade-para-profundidade.

Depois de obter essas transformações, deve-se aplicá-las à imagem correspondente. A Figura 49 mostra a função *CvInvoke.Undistort() que* calcula mapas de pixels tomando as matrizes de saída da função *CvInvoke.StereoRectify()* como entradas. A função aplica o mapa de pixels a uma Imagem para corrigi-la.

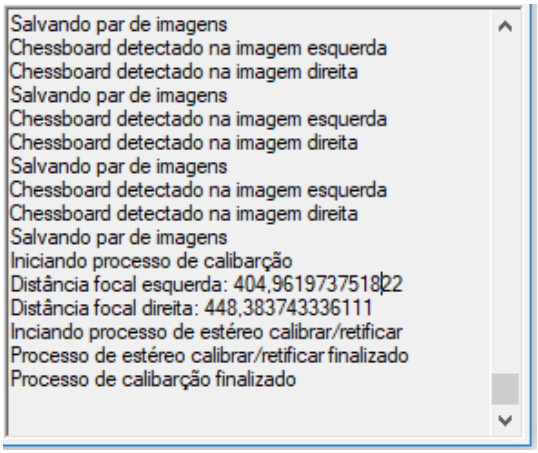
Figura 49 – Ajuste de distorção de câmeras.



Fonte: Autor, 2019.

A Figura 50 - mostra o status de calibração do sistema.

Figura 50 – Status de calibração do sistema.



Fonte: Autor, 2019.

3.3.4 Cálculo de Disparidades

Após estes procedimentos obtém-se *pixels* correspondentes nas duas imagens e calculam-se as disparidades. Na biblioteca EMGU CV, o algoritmo *Semi-Global Block Matching* (SGBM) da classe *StereoSGBM* utiliza uma técnica de programação dinâmica para tornar a correspondência mais robusta. O algoritmo utiliza parâmetros associados tais como:

- *SADWindowSize*: O tamanho do bloco de corresponder (deve ser um número ímpar)
- P1 e P2: parâmetros que controlam a suavidade da disparidade calculada.
- *speckleWindowSize* e *speckleRange*: Esses parâmetros controlam o filtro de *speckle*. *SpeckleWindowSize* indica o tamanho máximo de uma região de disparidade para que seja considerada como um *speckle*, enquanto *speckleRange* indica a máxima variação.
- *minDisparity* e *numberOfDisparities:* Juntos controlam a configuração de disparidade estéreo.

3.3.5 Detecção de Distância.

Para a estimação da distância com sistema de estéreo visão a função *reprojectImageTo3d()* efetua o calcula das coordenadas 3D tomando a disparidade e a matriz de mapeamento disparidade-profundidade Q gerada por CvInvoke.StereoRectify().

Pode-se perceber a relação disparidade-profundidade onde objetos com maior disparidade são coloridos com tons mais próximos ao branco, à medida que eles escurecem a disparidade diminui devido ao efeito paralaxe. Em conformidade com Cheng (2011), pode-se perceber que a profundidade tem propriedade inversa a disparidade, onde nas mesmas imagens os tons mais próximos ao branco são os de menor profundidade.

4. RESULTADOS E DISCUSSÃO

Para avaliação do sistema foram efetuados ensaios para cada fase de desenvolvimento do mesmo.

4.1 Hardware de Captura

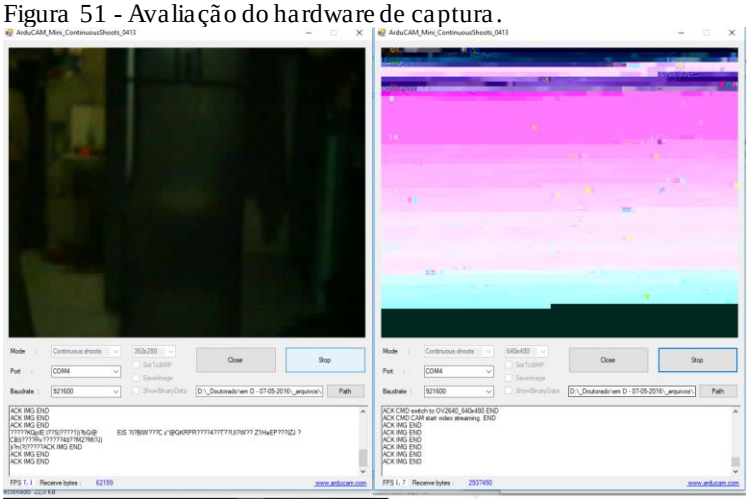
Na elaboração do *Hardware* de captura, foram avaliados a montagem do aparato, o funcionamento do hardware, a comunicação com o sistema em alto nível, a velocidade de tráfego de dados e captura e arquivamento das imagens.

A montagem em sapata padrão facilita o encaixe / desencaixe do suporte e a estrutura em guia prismática permite o ajuste da distância entre câmera na configuração do *baseline* de estéreo visão.

O funcionamento foi verificado com o *baudrate* máximo (921600 bps) e avaliou-se a captura em diferentes tamanhos de quadro. O modelo de câmera OV2640 opera com os tamanhos de *frame* de "160x120", "176x144", "320x240", "352x288", "640x480", "800x600", "1024x768", "1280x1024" e "1600x1200" *pixels*.

Com o aumento do tamanho do quadro tem-se o acréscimo do volume de dados trafegando entre o *Hardware* e o sistema, e consequente redução da taxa de transferência. Assim chegou-se ao tamanho de quadro de 352x288 *pixels* como mais adequado para operação, tendo como resposta uma taxa de 7,1 fps. Acima deste tamanho, a taxa de transferência cai para 1,7 fps. Estes resultados são apresentados na Figura 51, e foram obtidos a partir de programa do fabricante da interface Arducam.

Figura 51 - Avaliação do hardware de captura.



Fonte: Autor, 2019

A Figura 52 apresenta uma imagem captura e arquivada pelo conjunto do hardware e do sistema de captura.

Figura 52- Avaliação do hardware de captura.

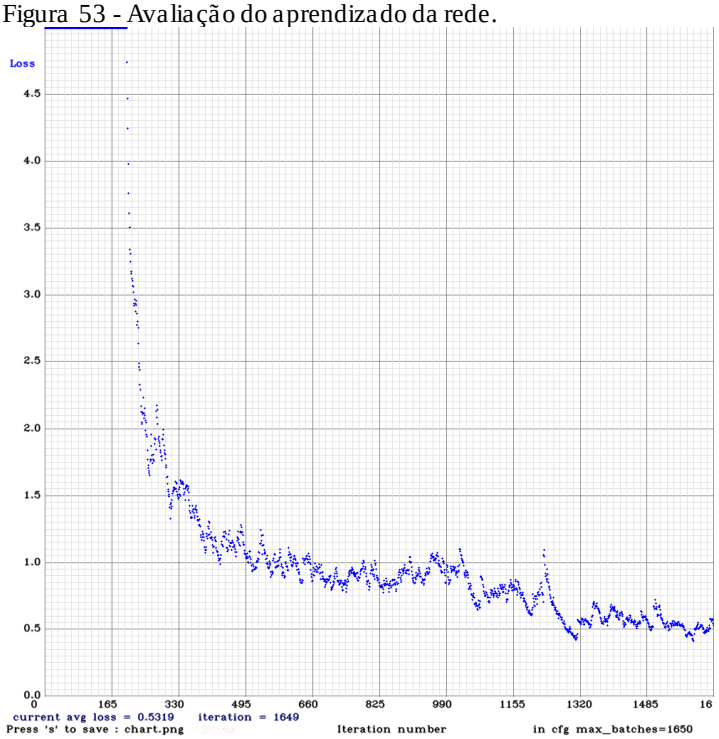
Fonte: Autor, 2019

4.2 Detecção de objetos

Foi analisado o treinamento da rede, o desempenho na identificação da alface e o tempo de resposta na detecção.

O treinamento da rede gera como resultado o arquivo de pesos (*weights*) de carga das camadas da rede para classificação da alface. Acompanhando o aprendizado pelo gráfico(*chart)* de precisão das médias (*mAP*), o primeiro treinamento apresentou resultados com bastante oscilação da convergência de aprendizado conforme a Figura 53, com Avg médio de 0.5319. este valor é considerado alto para acurácia do sistema.

Figura 53 - Avaliação do aprendizado da rede.



Fonte: Autor, 2019

Este resultado advém de o conjunto de dados apresentado no treinamento não ser adequado ou não estar devidamente configurado como também da definição das camadas da rede. Detecções geradas por este resultado de treinamento apresentam grande margem de erro de detecção como pode-se observar na Figura 54.

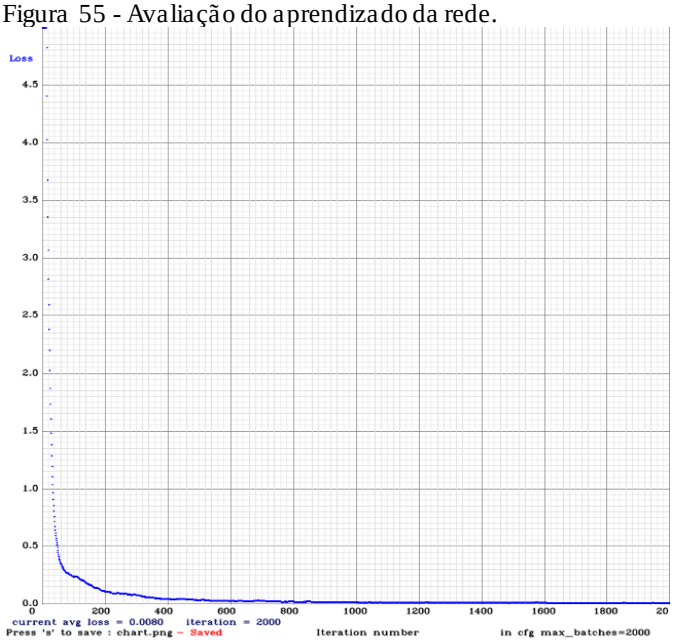
Figura 54 – Detecção com Arquivo não otimizado.

Fonte: Autor, 2019

Decorreu da falha no treinamento a ocorrência de falsos positivos detectados. Isso ocorre devido aos pesos(*weights*) não estarem adequados acarretando em novo treinamento com ajuste dos dados ou parâmetros da rede.

Redefinindo o conjunto de dados de treinamento, obteve-se o gráfico (*chart)* de precisão das médias (*mAP*) com AVG médio para 0,0080 apresentado na Figura 55.

Figura 55 - Avaliação do aprendizado da rede.



Fonte: Autor, 2019.

Figura 56 – Exemplo de detecção.

Fonte: Autor, 2019.

Para avaliar a detecção da rede treinada foi utilizado um conjunto formado por 100 imagens (do treinamento da rede, e imagens não conhecidas pelo sistema). Estas foram captadas em diferentes condições ambiente(diferença de luminosidade). Além disso, foi efetuado edição dos arquivos de imagem com tratamento do brilho simulando uma alta exposição (sobre exposto) e uma baixa exposição (sub exposta). A Figura 57 apresenta exemplo das imagens da alface utilizadas.

São verificados nesta avaliação o percentual de confiança (% de confiança) e o tempo decorrido (em ms) na detecção.

O próprio *framework darknet* fornece as condições necessárias a verificação da detecção com a apresentação destes parâmetros.

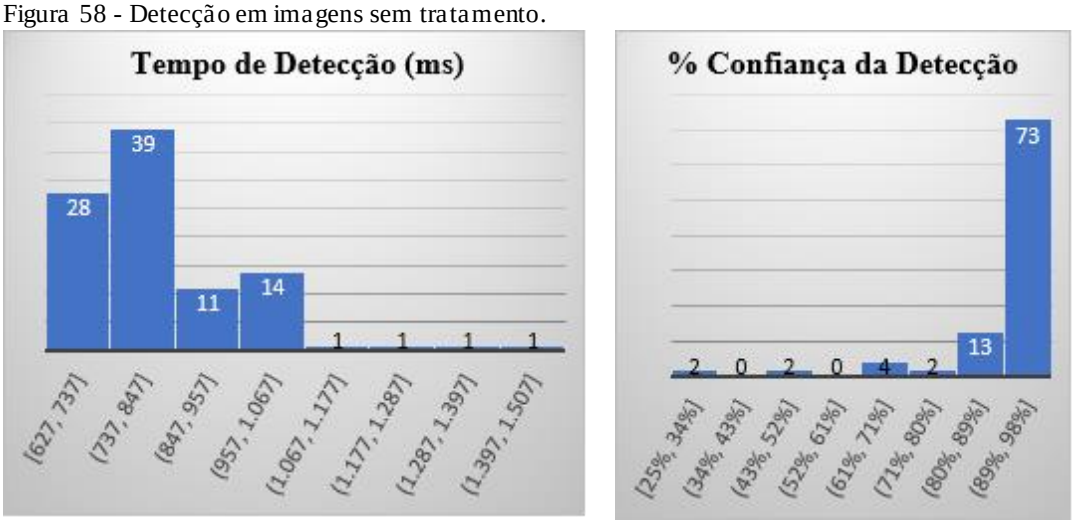
Figura 57 – Exemplo imagens a – Normal, b – sobre exposta, c – sub exposta.

a b c

Fonte: Autor, 2019.

Após efetuado a verificação, foram obtidos os resultados apresentados nas Figuras 58 a 60 a seguir. Pela análise dos valores obtidos, tem-se que em imagens sem tratamento (condição normal de luminosidade ambiente), em torno de 86 imagens foram detectadas com confiança acima de 80%. Tem-se ainda que 67 delas foram detectadas em menos de 850 ms, a média de tempo na detecção foi de 864,29 ms.

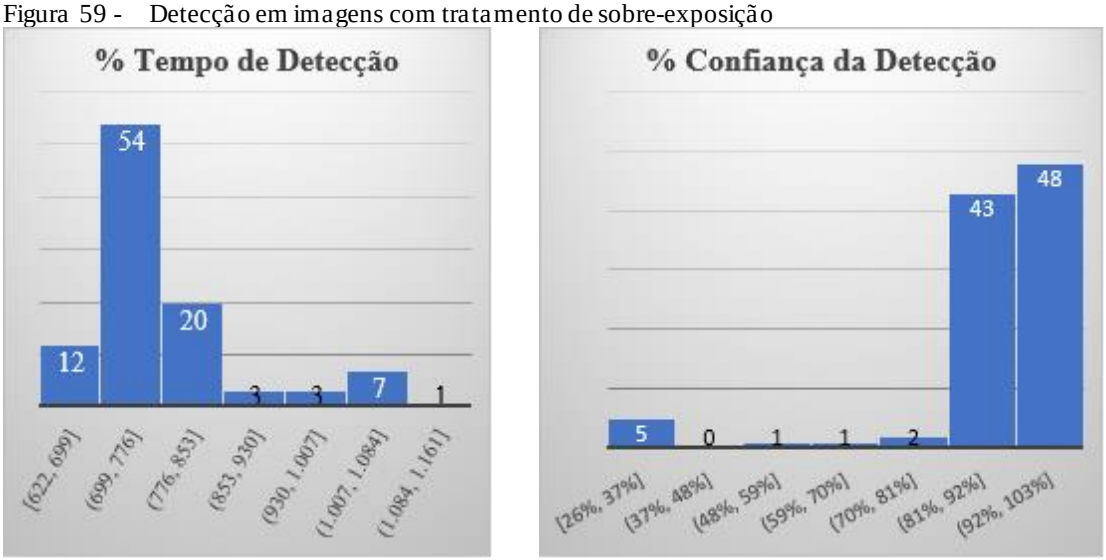
Figura 58 - Detecção em imagens sem tratamento.



Fonte: Autor, 2019.

Pela análise dos valores obtidos em imagens com tratamento de sobre exposição (condição de alta luminosidade ambiente), conforme Figura xx em torno de 91 imagens foram detectadas com confiança acima de 80%. Tem-se ainda que 86 foram detectadas em torno de 850 ms, a média de tempo na detecção foi de 788,25 ms.

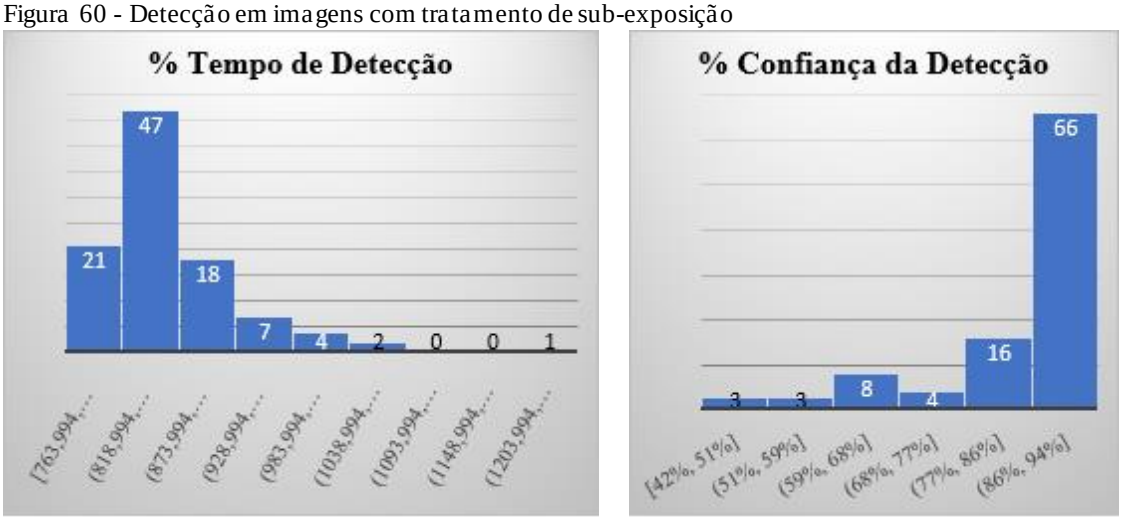
Figura 59 - Detecção em imagens com tratamento de sobre-exposição



Fonte: Autor, 2019.

Pela análise dos valores obtidos em imagens com tratamento de sub-exposição (condição de baixa luminosidade ambiente), conforme Figura xx, 82 imagens foram detectadas com confiança em torno dede 80% acima. Tem-se ainda que 86 delas foram detectadas em torno de 850 ms, a média de tempo na detecção foi de 744 ms.

Figura 60 - Detecção em imagens com tratamento de sub-exposição



Fonte: Autor, 2019.

Esta análise foi realizada com imagens estáticas. Para a execução em vídeo em tempo real, temos que a cada segundo, são lançados ao sistema em torno de 14 *frames* (taxa de 7,1 fps para cada câmera), que devem ser processados simultaneamente para o atendimento da detecção em estéreo visão.

Avaliando a detecção no sistema desenvolvido, o tempo de resposta não está adequada a sua utilização, devido ao atraso na atualização das saídas. Além disto, foi visto em alguns dos testes realizados, que não ocorreu sincronicidade da detecção conforme Figura 61, ou a detecção com valores de confiança diferentes para as câmeras esquerda e direita conforme Figura 62.

Suspeita-se que o motivo da inconformidade seja a não utilização de GPU. A não sincronicidade e a diferença de confiança na detecção pode estar sendo gerada pela diferença de análise em frames sucessivos. A utilização de GPU em um computador *desktop* depende da configuração do hardware que não foi obtido êxito neste.

Figura 61 - Detecção sem sincronicidade.

Fonte: Autor, 2019.

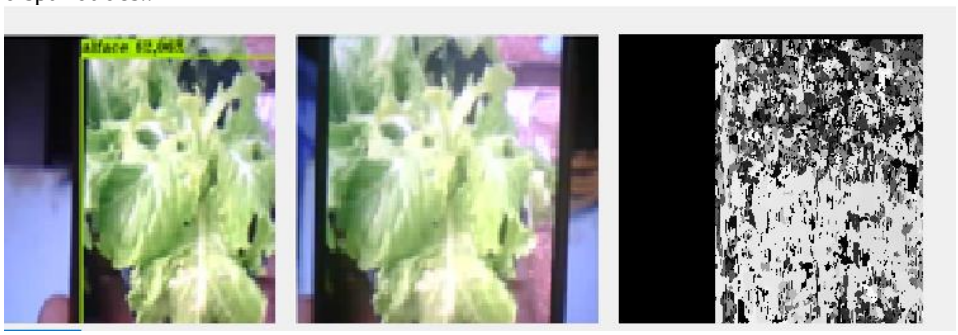
Figura 62- Detecção com diferença de confiança.

Fonte: Autor, 2019.

4.3 Estimação da distância

Com a matriz Q de mapeamento de disparidade, a estimação da distância é realizada a partir da detecção do objeto e do mapa de disparidade como apresentado na Figura 63.

Figura 63 – Mapa de disparidade: A – Imagem a esquerda, B – Imagem a direita, C – Mapa de disparidades..



A B C

Fonte: Autor, 2019.

Foram realizados posicionando -se a imagem a ser identifica a distancias de 10, 20, 30 cm do hardware de captura, e coletando-se o valor de distância fornecido pelo sistema. Na Figura 64 apresentamos duas medições realizadas a 30 cm.

Notam-se divergências da ordem de 11 a 20 % do valor de distância real para a distância fornecida pelo sistema. Estes valores são verificados quadro a quadro da imagem.

O valor de erro de medição pode também estar sendo comprometido pelo retardo da detecção dado o não uso de GPU em um computador *desktop*.

Figura 64 – Medição de distância.

Fonte: Autor, 2019.

5. CONCLUSÃO

O sistema de visão computacional para máquinas agrícolas – SVAM, foi desenvolvido, utilizando-se de técnicas de processamento de imagens e inteligência artificial, que possibilitou a captura, detecção e estimação de distancias para a hortícola da alface, tanto em campo como em ambiente protegido.

Foram especificados os componentes constituintes do sistema de visão computacional (hardware de captura) e elaborado o aparato. Também foram desenvolvidos os algoritmos de processamento da captura que obtiveram taxas de transferência da ordem de 7,1 fps operando em visão estéreo.

Forma elaborados algoritmos para processamento de imagem e reconhecimento de culturas, por meio de rede neural convolucional com taxas de assertividade da ordem de 80% e velocidades de detecção inferiores a 800 ms, em imagens estáticas, não tendo seu desempenho atestado em captura contínua.

Codificou-se os algoritmos para localização e estimação de distância que obtiveram os resultados apresentados.

Avaliando-se os resultados obtidos e considerando a não utilização da GPU, o que pode melhorar estes resultados, concluímos ser possível fornecer parâmetros de localização de hortícolas a dispositivos colhedores.

REFERÊNCIAS

ALBIERO, Daniel.; GONÇALVES, Flávio.R.F.; BEZERRA, José. H. P. **Sistema de visão estereo para identificação e localização de hortículas.** Depositante: UNIVERSIDADE FEDERAL DO CEARA. BR n. 10 2018 010359 8. Depósito: 22 mai. 2018. Concessão: 14 jun. 2018.

APHORT-Associação Portuguesa de Hotelaria, Restauração e Bebidas. **Hortículas**. Porto, 2010. Disponível em: https://www. http://www.aphort.com/index.php. Acesso em: 1 jul 2010.

AXIS COMMUNICATIONS. **Camera elements – Sensores de imagem.** Lund. 2016. Disponível em: https://www.axis.com/pt/pt/learning/web-articles/technical-guide-to-network-video/image-sensors Acesso em: 14 dez 2016.

BENGIO, Y.; COURVILLE, A.; GOODFELLOW, I. **Deep Learning**. Cambridge, MA. MIT Press,1ª ed. (2016).

BISHOP, C. **Pattern Recognition and Machine Learning**. 1ª ed, Berlin. Editora SPRINGER, 2001.

BRAHMNHATT, S. **Practical OpenCV**. New York. Apress. 2013.

CHENG, H.; NA, P.; LI, H.; ZHANG, Z. **Stereo image rectification algorithm for multi-view 3d display**.*[s. l.]* International Conference on 3D Imaging (IC3D), p. 1–5, 2011.

CONAB [COMPANHIA NACIONAL DE ABASTECIMENTO]. **Boletim Prohort revela que clima quente mantém tomate campeão de preços baixos.** Brasília- DF. Disponível em: https://www.conab.gov.br/ultimas-noticias/2718-boletim-prohort-revela-que-clima-quente-mantem-tomate-campeao-de-precos-baixos-na-ceasas. Acesso em: 30 jun. 2019.

CNA [Confederação da Agricultura e Pecuária do Brasil]. **Agropecuária supera obstáculos e segue liderando a economia brasileira em 2016.** Brasília- DF. Disponível em: http://www.cnabrasil.org.br/noticias/agropecuaria-supera-obstaculos-e-segue-liderando-economia-brasileira-em-2016 Acesso em: 14 jan 2017.

COSTA, C. P.; SALA, F. C. A evolução da alfacicultura brasileira. Brasília- DF. **Horticultura Brasileira,** v.23, n. 1 (Artigo de capa). 2005.

CRUVINEL, P. E. **Instrumentação agropecuária no agronegócio brasileiro do século XXI**: **Parte 1**. São Carlos. Disponível em: http://www.embrapa.br/noticias/artigos/2000/artigo. Acesso em: 28 ago. 2014.

DEMÉTRIUS A. S.; KÉDYMA M. **Estudo da Identificação de Emoções Através da Inteligência Artificial**. Cachoeiro de Itapemirim. Multivix. Disponivel em: http://www.emgu.com/wiki/index.php/Emgu_CV. Acessado em: 10 jun 2019.

ECMA. (2012). **C# Language Specification**, Genebra. 4 ed., Ecma International.

EMGUCV. **Main Page**. *[s. l.],* 2013. Disponivel em: http://www.emgu.com/wiki/index.php/Emgu_CV. Acessado em: 10 Jan 2019.

GUSTAVO, P.; DUARTE, I.; NASCIMENTO, B. **Pib da Agropecuaria Cresce 0,1% em 2018 Limitado Por Greve dos Caminhoneiros [Estado de São Paulo].** São Paulo, 28 fevereiro 2019. Disponível em: https://economia.estadao.com.br/noticias/geral,pib-da-agropecuaria-cresce-0-1-em-2018-limitado-por-greve-dos-caminhoneiros,70002739157. Acesso em: 30 jun. 2019.

FERNANDES, J. L. – **Aplicação de Algoritmos de Otimização Estocástica à Correspondência Entre Imagens em Visão Estéreo.** Niterói, Universidade Federal Fluminense, 2010.

FILGUEIRA F. A. R. **Novo Manual de Olericultura: Agrotecnologia moderna na produção e comercialização de hortaliças.** Viçosa, 2ª ed. UFV. 412p. 2003.

GARDIMAN R. Q.; DINIZ I. S.; BRUGINSKI R. F. **Implementação de um Sistema de Visão Estéreo e Triangulação como Técnica para Determinação de Distância.** Sorocaba, Unesp – Universidade Estadual Paulista – Campus Sorocaba.

GONZALEZ, R.C. e Woods, R.E. e Eddins, S.L. **Digital Image Processing using MATLAB**, Londres. Pearson 2 ed, 2006.

GUEDES, Í. M. R. **A horticultura brasileira precisa de uma revolução branca**. Brasília – DF. Disponível em: https://www.embrapa.br/busca-de-noticias/-/noticia/8955575/ Acesso em: 15 jan 2017.

HAYKIN, Simon. **Redes Neurais: Princípios e Práticas**. Porto Alegre. Bookman 2 ed, 900p., 2009.

GOODFELLOW I; BENGIO Y; COURVILLE A. **Deep learning:.** Cambridge, MA**.** The MIT Press, 2016, 800 pp, ISBN: 0262035618

IBÁÑEZ, D. **How to train YOLOv3 using Darknet on Colab 12GB-RAM GPU notebook and speed up load times**. *[s.l.],* DEV-ibanez.info**.** 2017. Disponível em: http://blog.ibanyez.info/blogs/coding/20190410-run-a-google-colab-notebook-to-train-yolov3-using-darknet-in/ Acesso em: 12 maio. 2019.

JENSEN, S; SELVIK, A, L. **Using 3D graphics to train object detection systems**. Norwegian , 76 f. Dissertation (Master of Science in Computer Science) - Computer and Information Science, Norwegian University of Science and Technology, 2016.

KODAK. **Câmera**. *[s.l.],* New York. Disponível em http://wwwca.kodak.com/default.htm. Acesso em 14 jan 2017.

KÜHNE, F. YOLO: um sistema para detecção de classes de objetos em tempo real. **Profissionais TI**. São Paulo. 2018. Disponível em: https://www.profissionaisti.com.br/2018/07/yolo-um-sistema-para-deteccao-de-classes-de-objetos-em-tempo-real/. Acesso em: 25 maio. 2019.

LIMA, G. F..**Controle de Tempertatura de um sistema de Baixo Custo Utilizando a Placa**

Arduino. Natal, IX congresso de iniciação cientifica do IFRN, 2013.

LIMA, M. E. **Avaliação do desempenho da cultura da alface (*Lactuca sativa*) cultivada em sistema orgânico de produção, sob diferentes lâminas de irrigação e coberturas do solo.** Seropédica, 63f. Dissertação (Mestrado em Ciências) – Universidade Federal Rural do Rio de Janeiro, 2007.

LISA Lab: **Convolutional Neural Networks (LeNet) - DeepLearning 0.1 documentation**. *[s.l.]*. Disponível em: http://deeplearning.net/tutorial/lenet.html. Acesso em: 31 de jun 2019.

LU, J. Sang, N. **Detecting citrus fruits and occlusion recovery under natural illumination conditions**. Amsterdam. Computers and Electronics in Agriculture. 2014.

MARENGONI, M.; STRINGHINI D. **Introdução a visão computacional usando openCV**. Berlin. RITA, v. XIII, n.1, 2009.

MARQUES, E. A. L.: **Estudo sobre Redes Neurais de Aprendizado Profundo com Aplicações em Classificação de Imagens**. Brasília, Universidade de Brasília Departamento de Estatística 2016.

MASCARENHAS, M. H. T.; ROCHA, FE de C. **Panorama da mecanização na olericultura brasileira**. Brasília. Embrapa Milho e Sorgo-Artigo em periódico indexado (ALICE), 1997.

MARR, D. V. **A computational investigation into the human representation and processing of visual information**. Cambridge, MA. S.1: The MIT Press, 1982.

MEHTA, S.S; BURKS, T.F. **Vision-based control of robotic manipulator for citrus harvesting**. Amsterdam. Computers and Electronics in Agriculture. 2014

MELO, M. **Circuito Carioca de Feiras Orgânica.***[s.l.]* Disponível em: .http://maps.google.com.br/maps/ms?ie=UTF8&oe=UTF8&msa=0&msid= 216945583591336355959.0004d59ee8d1046a09da0. Acessado em: 18 de agosto de 2013.

MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO-MAPA. **Agropecuária Brasileira em Números**. Brasília, DF. Disponível em: http://www.agricultura.gov.br/assuntos/politica-agricola/agropecuaria-brasileira-em-numeros . Acesso em: 30 jun. 2019.

MORETTI, C. L. **Boas práticas agrícolas para a produção de hortaliças**. Brasilia, DF. Embrapa. Horticultura Brasileira, v. 21, n. 2, p. 1-27, 2003.

MORRIS T. **Computer Vision and Image Processing**. Palgrave, Macmillan. 2004.

NEVES, R. J. M da S. **Condução Visual de Embarcações Autónomas**. Porto, Universidade do Porto, 2013.

NUNES, A, C, S. **Biofortificação de plantas de alface (Lactuca sativa L.) geneticamente modificada para aumento do teor de folato.** Brasília, DF, . 134 f. Dissertação (Mestrado em Biologia Molecular) Instituto de Ciências Biológicas, Universidade de Brasília, 2009.

OLIVEIRA, S. K. L. *et al.* **Cultivo de alface com proteção de agrotêxtil em condições de altas temperaturas e luminosidade**. Mossoró. Revista Caatinga, v.19, n.12, p.112-116, 2006.

OPENCV. **OpenCV: About OpenCV**. *[s.l.]*. Disponivel em: https://opencv.org/about/. Acessado em: 17 Ago 2017. . (2013).

OPENCV. **OpenCV Documentation**. *[s.l.]*.Disponível em: http://docs.opencv.org/3.2.0/index.html. Acesso em: 14 set 2016.

PUTTI, F, F. **Produção da cultura de alface irrigada com água tratada magneticamente.** Botucatu, 123 f. Dissertação (Mestrado Irrigação e Drenagem) - Faculdade de Ciências Agronômicas, Universidade Estadual Paulista Júlio de Mesquita Filho, 2014.

QUEIROGA, R. C. F. *et al.* Produção de alface em função de cultivares e tipos de tela de sombreamento nas condições de Mossoró. Mossoró. **Horticultura Brasileira**, v.19, n.3, p.324, 2001.

QUEIROS, J. E. R de, GOMES, H. M. **Introdução ao Processamento Digital de Imagens**. Berlin. RITA, v. VIII, n1, 2001.

RAVINDRA, S; REIS, P. **Como as Redes Neurais Convolucionais realizam o reconhecimento de imagem**. São Paulo. InfoQ. 2017. Disponível em: https://www.infoq.com/br/articles/redes-neurais-convolucionais/. Acesso em: 10 maio 2019.

RIA - **Robotics Industries Association**. Michigan. Disponível em: http://www.ria.org. Acesso em: 28 ago. 2014.

RYDER, E. J. 2002. **The new salad crop revolution.** Alexandria. Disponível em: http://www.hort.purdue.edu/newcrop/ncnu02/v5-408.html. Acesso em: 10 set. 2018.

SANCHES, R. M., **Desenvolvimento de um sistema de planejamento de trajetória para veículos autônomos agrícolas**. São Carlos, 2012.

SANTOS, R. M.: **Um Estudo de Procesamento de Imagens com OPENCV**. Niterói UNIVERSIDADE FEDERAL FLUMINENSE, 2011.

SANTOS, H, R. **Viabilidade de produção da alface americana no município de Pentecoste, CE.** Fortaleza, 50 f. Dissertação (Mestrado Fitotecnia), Departamento de Fitotecnia, Universidade Federal do Ceará, 2013.

Security Camera King. **Tecnologias CCD/CMOS**. Boca Raton, FL. Disponível em: http://www.securitycameraking.com/securityinfo/cmos-vs-ccd/. Acesso em: 09 dez 2016.

SILVA, G. D. **Desenvolvimento de um Sistema de Visão Computacional para o Estudo do Comportamento de Animais Experimentais**: Belo Horizonte. 2008. Disponível em: http://cpdee.ufmg.br/defesas/343M.PDF. Acesso em: 03 de Setembro de 2010.

SILVA, J.M.C. **Distribution of Amazonian and Atlantic birds in gallery forests of the cerrado region, South America.** Miami. Ornitologia Neotropical 7:1-18, 2008.

SILVA, V. F. *et al.* Comportamento de cultivares de alfaces em diferentes espaçamentos sob temperatura e luminosidade levadas. Brasília, DF **Horticultura Brasileira**, v.18, n.3, p. 183-187, 2000.

SNA[Sociedade Nacional de Agricultura]. **Maior mostra da horticultura brasileira antecipa inovações em tecnologia agrícola**. Rio de Janeiro. Disponível em: http://sna.agr.br/maior-mostra-da-horticultura-brasileira-antecipa-inovacoes-em-tecnologia-agricola/. Acesso em: 09 janeiro 2017.

SCURI, A. E. **Fundamentos da Imagem Digital**.Rio de janeiro, ecgraf/PUC-Rio, 2002.

Vadivambal, R.; Jayas, D. S. **Bio-Imaging Principles, Techniques,and Applications.** Boca Raton, CRC PRESS, FL, 2016.

VIAN, Carlos Eduardo De Freitas, ANDRADE JÚNIOR, Adilson Martins. **Evoluçao Histórica da Indústria de Máquinas Agrícola no Mundo:** Orígens e Tendencias**.** PIRACICABA**,**.ESALQ USP – SP, 2013.

WILLIAMS, H. A. M; JONES, M. H; NEJATI, M, et al. **Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms**. Cambridge England, Biosystems engineering**.** v 181 p. 140 - 156, 2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S153751101830638X?via%3Dihub. Acesso em: 29 jun. 2019.

WIKIMEDIA COMMONS. **File:Nikon F SLR camera with NIKKOR-S Auto 1,4 f=5,8cm.JPG**. *[s.l.]*. Disponível em : https://commons.wikimedia.org/wiki/File:Nikon_F_SLR_camera_with_NIKKOR-S_Auto_1,4_f%3D5,8cm.JPG. Acessado em: 09 janeiro 2017.

YURI, J. E. *et al.* **Comportamento de cultivares e linhagens de alface americana em Santana da Vargem (MG), nas condições de inverno**. Brasília, Horticultura Brasileira, v.22, n.2, p.322-325, 2004ª