Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

Estimação de posição de um quadrirrotor utilizando o sensor *Kinect*

Autor: Tiago Avelino Ribeiro da Silva
Orientador: Dr. André Murilo de Almeida Pinto

Brasília, DF
2019

Tiago Avelino Ribeiro da Silva

Estimação de posição de um quadrirrotor utilizando o sensor *Kinect*

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. André Murilo de Almeida Pinto

Coorientador: Dr. Renato Vilela Lopes

Brasília, DF

2019

Tiago Avelino Ribeiro da Silva
Estimação de posição de um quadrirrotor utilizando o sensor *Kinect*/ Tiago Avelino Ribeiro da Silva. – Brasília, DF, 2019-
85 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. André Murilo de Almeida Pinto

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2019.

1. *Quadrirrotor*. 2. *Deep Learning*. I. Dr. André Murilo de Almeida Pinto. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Estimação de posição de um quadrirrotor utilizando o sensor *Kinect*

CDU 02:141:005.6

Tiago Avelino Ribeiro da Silva

Estimação de posição de um quadrirrotor utilizando o sensor *Kinect*

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 13 de dezembro de 2019 – Data da aprovação do trabalho:

Dr. André Murilo de Almeida Pinto
Dr. André Murilo de Almeida Pinto

Dr. Renato Vilela Lopes
Dr. Renato Vilela Lopes

Dr. Diogo Caetano Garcia
Dr. Marcus Vinícius Chaffim Costa

Dr. Henrique Marra Taira Menegaz
Dr. Henrique Marra Taira Menegaz

Brasília, DF
2019

Este trabalho é dedicado ao único Deus, ao meu pai, Jeronimo, à minha mãe, Jussara, à minha irmã, Amanda e à minha namorada Yasmin

Agradecimentos

Agradeço a Deus em primeiro lugar, por Ser a fonte de toda minha energia, todo conhecimento e toda sabedoria, por me capacitar durante o processo e por me manter firme durante as dificuldades. Agradeço por que sempre que precisei Ele nunca me abandonou e sempre apareceu com graça e fidelidade. Agradeço por proporcionar o entendimento necessário e alegria durante esses 5 anos, sendo todas as atividades por Ele, para Ele e por meio dEle, eternamente.

Agradeço também a minha família, meu Pai Jeronimo, e minha Mãe Jussara, que me ensinaram os caminhos a qual deveria percorrer e por todo apoio e amor que proporcionaram durante todos os anos. Agradeço minha irmã por sempre compartilhar os momentos tristes e felizes e sempre que possível me auxiliando nos pedidos aleatórios.

Agradeço a meu tio Almir e a minha tia Joselene, assim como agradeço ao meu tio Isaac e minha tia Uiara, por sempre me darem suporte e me acompanharem nos processos. Agradeço aos meus primos precursores no caminho da Engenharia Leonardo e Lucas, por sempre ser prestativo e me aconselharem com o aprendizado de suas jornadas. Agradeço também aos primos Isaac e Alexandre que sempre se importam e facilitam meus passos.

Em especial agradeço a meu primo Luis que durante esses anos, tem se tornado um irmão e que eu tenho o prazer de reconhecer com um amigo fiel e correto, no qual sei que posso contar. A lista de agradecimentos a família seria gigantesca, mas agradeço a todos, pois sei que compartilhamos uma união que não poderia reclamar.

Agradeço ao meu amigo Daniel por sempre ter uma visão além do tempo. Agradeço também a minha cunhada Letícia, por me ajudar em várias ideias malucas. Agradeço a meu amigo Rodrigo pelo companheirismo e por toda prestatividade durante esses anos.

Aos amigos da faculdade agradeço em especial a meu amigo Erick, que compartilhei diversas matérias e que sempre me ajudou a atingir as metas nas matérias. Agradeço também aos meus amigos Gabriel e Victor, que por mais que tenha conhecido os conhecido há pouco tempo foram fundamentais para minha graduação e me inspiram a ser melhor.

Agradeço também aos professores André e Renato, que proporcionaram os materiais necessários para construção do presente trabalho e pelas orientações nos últimos anos.

Por fim agradeço a minha namorada Yasmin, por sempre acreditar nas atividades que me proponho a fazer, por confiar nos meus projetos por me ensinar a sonhar. Muito além dos sonhos por começar a construir-los e me incentivar a não desistir, mesmo quando estes não parecem ser possíveis. Agradeço por todo apoio emocional e por não se desviar

dos propósitos e convicções.

"A glória de Deus está nas coisas encobertas; mas a honra dos reis, está em descobri-las. Os céus, pela altura, e a terra, pela profundidade, assim o coração dos reis é insondável. Provérbios 25:2,3

Resumo

O presente trabalho trabalho aborda a implementação de algoritmos de rastreamento e algoritmos de Aprendizado Profundo para rastreamento e detecção de um Veículo Aéreo Não Tripulado (VANT). Dois algoritmos de rastreamento foram escolhidos: Filtro de Correlação de Kernel (KCF) e Mínima Saída do Somatório do Erro Quadrático (MOSSE). Além de utilizar um algoritmo de alta velocidade para a detecção do quadrirrotor: Yolov3. O objetivo é extrair a posição do quadrirrotor de forma automática utilizando o sensor Kinect, adquirindo esta posição em tempo real. Um estudo dos algoritmos de Aprendizado de Máquina e Aprendizado Profundo é apresentado para a implementar um Rede Neural Convolucional. Além de realizar um estudo sobre as técnicas de rastreamento que serão utilizadas. Por fim serão apresentados os resultados obtidos, tanto para cada um dos algoritmos, quanto para a composição da abordagem final.

Palavras-chaves: Aprendizado de máquina. Aprendizado Profundo. MOSSE. KCF. PARROT Ar Drone 2.0. Sensor Kinect.

Abstract

The present work addresses the implementation of tracking algorithms and Deep Learning algorithms for traking and detecting of an Unmanned Aerial Vehicle (UAV). Two tracking algorithms were choseen: Kernel Correlation Filter (KCF) and Minimum Output Sum of Squared Error (MOSSE). In addition, a high-speed quadrotor detection algorithm was used: Yolov3. The goal is to extract a quadrotor position automatically using Kinect sensor, acquiring this position in real time. A study of Machine Learning and Deep Learning algorithms is presented to implement the Convolutional Neural Network. besides conducting a study on the tracking techniques that will be used. Finally, the results are displayed for both the algorithms and the composition of the final approach.

Key-words: Machine Learning. Deep Learning. KCF. MOSSE. PARROT AR. Drone. 2.0. Kinect Sensor.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CM	Centro de Massa
CNN	*Convolution Neural Network* (Rede Neural Convolucional)
CPU	*Central Processing Unit* (Unidade de Processamento Central)
CSRT	*Channel and Spatial Reliability*
DFT	Transformada Discreta de Fourier
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
GPU	*Graphical Processing Unit* (Unidade de Processamento Gráfica)
IoU	*Intersect over Union*
IMU	*Inertial Measurement Unit* (Unidade de Medição Inercial)
KCF	*Kernelized Correlation Filters* (Filtro de Correlação de Kernel)
LQR	*Linear Quadratic Regulator* (Regulador Linear Quadrático)
MIL	*Multiple Instance Learning*
MOSSE	*Minimum Output Sum of Squared Error* (Saída Mínima da Soma do Erro Quadrático)
PDF	*Probability Density Function* (Função de Densidade de Probabilidade)
PID	*Proportional Integral Derivative* (Proporcional Integral Derivativo)
RBF	Função de Base Radial
ReLU	*Rectified Linear Unit* (Unidade de Retificação Linear)
TLD	*Tracking, Learning and Detection*
UAV	*Unmanned Aerial Vehicle*
VANT	Veículo Aéreo Não-Tripulado
YOLO	*You Only Look Once*
2D	Duas Dimensões
3D	Três Dimensões

Lista de símbolos

Λ	Parâmetro de distorção radial.
$\in$	Pertence a.
A	Matriz de estados
$\odot$	Operação produto ponto a ponto.
η	Parâmetro da taxa de aprendizado.
Δ	Regra delta.
δ	Derivada parcial.
C_x	Matriz de circulação.
σ	Letra grega minuscula sigma.
ϵ	Erro médio quadrático.
p_i	Parâmetro distorção tangencial.
α	Solução da regressão de cume.
C_x	Matriz de circulação.
m	Filtro de correlação regularizado.
k	*Kernel* linear.
σ	Função sigmoide.

Sumário

1 Introdução

Os quadrirrotores são veículos do tipo asas móveis, ou helicóptero, que possui quatro rotores, sendo cada um posicionado em uma das hélices. Esse tipo de veículo é classificado como de decolagem e pouso vertical(VTOL, do inglês *Vertical Take-Off and Landing*), e ainda pairar em uma coordenada espacial fixa (*Hover*) (OLIVEIRA, 2014).O quadrirrotor possui algumas vantagens, sendo uma delas sua simetria, com simples construção e possuindo certa facilidade de detecção de uma possível falha, facilitando a manutenção. Por se tratar de um veículo VTOL não precisa de uma grande área para decolagem e aterrizagem.

No início do desenvolvimento estes veículos eram utilizados com intuito militar por conta do alto custo da tecnologia embarcada, porém com o avanço tecnológico e com a miniaturização do sistema o processo de fabricação foi facilitado e o custo dos sistemas embarcados foram reduzidos, permitindo que os quadrirrotores ocupassem diversas áreas tais como: lazer, utilizando as câmeras para realizar filmagens; segurança, realizando o monitoramento de propriedades e vigilância em incêndios; militar, atuando na identificação de bombas; entrega, realizando o transporte de cargas (VYAS, 2019) e muitas outras.

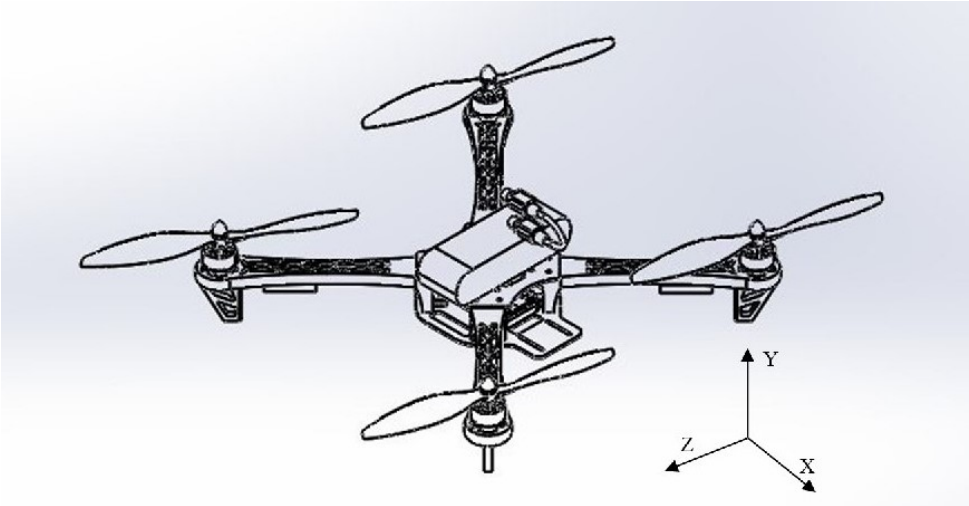


Figura 1 – Sistema de coordenadas de um quadrirrotor tipo F450 realizado no CAD.

Para garantir a estabilidade em voo do Veículo Aéreo Não-Tripulado (VANT) existem diversas técnicas de controle, implementadas em *software* e embarcadas em um microcontrolador. Esse microcontrolador executa múltiplos algoritmos, realizando cálculos de trajetória, estimação dos sensores, leitura dos sensores, garantindo a resposta em tempo limitado, para que não haja falha no controle. Dessa forma, para trabalhar com essa quantidade de algoritmos embarcados, é criado um sistema operacional primitivo

para executar cada aplicação, técnica comumente adotada em sistemas embarcados (programação *bare metal*).

Tendo em vista que para embarcar algoritmos de controle é necessário extrair dados provenientes da unidade de medição inercial (IMU, do inglês *Inertial Measurement Unit*, conjunto de sensores, tais como: acelerômetro, magnetômetro e giroscópio) e que esses dados podem sofrer com diferentes tipos de erros, tais como erros de derivação entre outros discutidos em (IV; WALL; BEVLY, 2005). Além disso, para ambientes internos, há a dificuldade da estimação da posição, por conta do bloqueio do sinal do Sistema de Posicionamento Global ((GPS), do inglês Global Positioning System). Uma forma de minimizar esses erros é utilizando técnicas de Visão Computacional e Inteligencia Artificial, que permitem o monitoramento da posição do VANT. Com essas técnicas é possível captar a posição real do quadrirrotor em qualquer momento, desde que este esteja sob o monitoramento por câmeras.

Para que haja a detecção do quadrirrotor de uma forma automática são aplicados diferentes métodos, algum desses são os métodos de Aprendizado de Máquina (ML, do inglês *Machine Learning*) e Aprendizado Profundo (DL, do inglês *Deep Learning*). Outra técnica que permite a automação do processo de monitoramento é a Rede Neural Artificial, que teve início em 1943 com a primeira formulação matemática do neurônio pelo neurofisiologista Warren McCulloch e o matemático Walter Pitts. A evolução das Rede Neurais artificiais permitiu aplicações inteligentes por meio de computadores, possibilitando que estes interpretem dados e tomem decisões com base nesse aprendizado (MAYO HASHAN PUNCHIHEWA, 2019).

A popularização das técnicas de Aprendizado de Máquina se deu por conta dos avanços na tecnologia das Unidades Centrais de Processamento (CPU, do inglês *Central Processing Unit*), que executam as operações lógicas e o controle das entradas e saídas. Inicialmente as CPUs possuíam apenas um núcleo de processamento, permitindo que apenas uma operação fosse realizada em um tempo específico. Posteriormente, com os avanços nas pesquisas e no desenvolvimento dos chips surgiram, as CPUs com vários núcleos de processamento, permitindo o início da programação com paralelismo, realizando mais de uma operação no mesmo período de tempo (BALTAZAR, 2018).

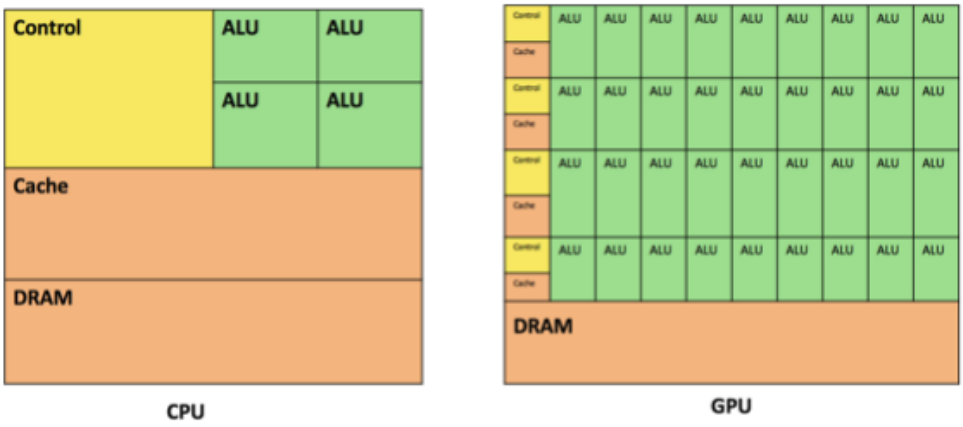


Figura 2 – Representação da CPU (a esquerda) e de uma GPU (a direita) (BALTAZAR, 2018).

Com o aparecimento das Unidades Gráficas de Processamento(GPU, *Graphical Processing Unit*) Fig. (2), que possuem uma quantidade maior de núcleos de processamento quando comparada com as CPUs, permitindo um maior poder de programação em paralelo (BALTAZAR, 2018), pode-se realizar o treinamento das redes neurais de uma forma mais rápida, auxiliando na popularização de Redes Neurais Artificiais com mais de uma camada oculta (Aprendizagem Profunda), sendo uma delas a *Convolution Neural Network* (CNN).

Com a viabilização das técnicas de Aprendizado de Máquina e Aprendizado Profundo surgiram novas formas de implementar soluções em diversas áreas, tais como: medicina, promovendo soluções na identificação de doenças; visão computacional, promovendo o reconhecimento de objetos, de animais, de pessoas, entre outros; reconhecimento da fala humana; sistemas de publicidade inteligente, que selecionam o tipo de propaganda para um determinado público; detecção de fraudes em documentos; na indústria da automação, por meio de dispositivos autônomos como robôs de fábricas; entre diversas outras aplicações (SANTANA, 2018).

Dentre essas aplicações uma de interesse é a de detecção e rastreamento da posição de um quadrirrotor em ambientes fechados, por conta das falhas dos sensores citadas e para garantir o funcionamento correto do controlador de voo antes que o veículo seja utilizado em um teste em ambiente aberto, o que poderia ocasionar a destruição do veículo por falha no controle,assim como lesões nas pessoas em volta do VANT. Para verificar o funcionamento do quadrirrotor pode se utilizar destas técnicas de detecção e rastreamento para verificar o funcionamento do controlador de posição, por meio da medição, por meio de câmeras, da posição real do quadrirrotor e utilizar esse dado como *setpoint* ideal para o quadrirrotor.

Um outro fator de interesse é utilizar os dados obtidos nessas técnicas para controle do quadrirrotor, enviando a posição dentro do ambiente identificado e avaliando o

comportamento do VANT na trajetória determinada.

1.1 Objetivos Gerais

Os objetivos gerais do presente trabalho são:

- Extrair a posição real do quadrirrotor em uma imagem.
- Realizar o rastreamento do VANT em tempo real.
- Realiza a detecção do quadrirrotor.

1.2 Objetivos Específicos

Visando atender os objetivos gerais, alguns objetivos específicos são especificados:

- Determinar os algoritmos que serão utilizados para o rastreamento.
- Construir um *dataset* para a detecção do quadrirrotor.
- Determinar o algoritmo de detecção que será utilizado.

1.3 Organização do trabalho

O texto do trabalho foi organizado em cinco capítulos introdução, referência bibliográfica, metodologia, resultados e conclusão. Um resumo de cada um desses é apresentado em seguida.

- **Capítulo 1 :** Neste capítulo é realizada a introdução dos conceitos mais importantes, além de realizar a contextualização em qual o atual trabalho esta inserido. São apresentados os objetivos gerais e específicos do trabalho e é detalhada a organização do trabalho.
- **Capítulo 2 :** Neste capítulo é apresentada a revisão bibliográfica que serviu de base para o desenvolvimento deste trabalho. São discutidas as técnicas de Aprendizado de Máquina e Aprendizagem Profunda para detecção do quadrirrotor, além de apresentar algoritmos de rastreamento e realizar uma introdução a unidade de medidas inercial (IMU). Além disso, é apresentado o *hardware* do sensor *Kinect* e uma discussão das limitações da câmera do sensor.
- **Capítulo 3 :** Neste capítulo é abordada a metodologia utilizada para elaborar o trabalho e são descrito os passos necessários para execução do trabalho.

- **Capítulo 4 :** Neste capítulo são apresentados resultados obtidos para a etapa de rastreamento e detecção, separadamente. Em seguida são apresentados os resultados da integração dos sistemas e a extração da posição.
- **Capítulo 5 :** Neste capítulo é realizada a conclusão para o presente trabalho.

2 Revisão Bibliográfica

Esta seção apresenta conceitos importantes de Processamento Digital de Imagem e Inteligência Artificial, para realizar a detecção e rastreamento de Veículos Aéreos Não-Tripulados do tipo quadrirrotor, incluindo discussões sobre conceitos iniciais de unidades de medidas inerciais e técnicas utilizadas que serviram de base para a elaboração do trabalho. Serão apresentados os trabalhos de identificação e rastreamento já realizados, com utilização de filtros diversos, assim como técnicas de identificação de movimentos primitivos, por meio de diversas transformadas (a transformada de Hough por exemplo), obtendo alguns movimentos de mãos e do corpo.

2.1 Fundamentação Teórica conceitos de Aprendizado de máquina e Aprendizado Profundo

Para entender o funcionamento dos algoritmos que serão descritos, é preciso inicialmente entender o funcionamento de técnicas de Inteligência Artificial. A Inteligência Artificial surgiu em 1943 com a definição do neurônio por meio de um circuito eletrônico (MCCULLOCH; PITTS, 1990). Evoluindo a um sistema que consegue aprender com experiências passadas, dinamicamente, sem intervenção humana, com base em alguma característica (*feature*) específica.

Um neurônio artificial tem seis componentes principais bem definidos, de forma a correlacionar um neurônio real e o artificial por meio de expressões matemáticas definidas em (Lippmann, 1987). Esses componentes são descritos abaixo e representados na Fig. (5).

1. **Nós de entrada** são os nós responsáveis em receber a informação do exterior e transmitir essa informação para um nó da Rede Neural Artificial (RNA).

2. A **Conexão** entre os nós da RNA, que pode ser ponderada de forma a sair da conexão o nó anterior multiplicado pelo peso da conexão, $x_1 w_1$ por exemplo.

3. Um **Somatório Ponderado** é utilizado para obter um valor conjunto de todos os nós de entrada com cada peso associado $y = \sum_{i=1}^{D}(w_i * x_i)$, em que x_i é a entrada do neurônio e w_i é o peso associado.

4. Posterior a isso é aplicada uma **Função de Ativação** que permite introduzir uma função não linear na saída do neurônio artificial ou *perceptron*, já que os dados reais

possuem em geral características não lineares. Um exemplo de função de ativação é a sigmóide da forma $\sigma(y) = 1/(1 - e^{-y})$, em que y é a saída da Função de Ativação.

5. Ainda é utilizado um ***Bias*** b_s, para realizar um último ajuste deslocando a função para a esquerda ou para a direita, ajustando melhor aos dados analisados.

6. A **Saída** do neurônio se dá pela associação do resultado da Função de Ativação com o somatório ponderado, obtendo $y = f(w_1x_1 + w_2x_2 + ... + w_ix_i + b_s)$.

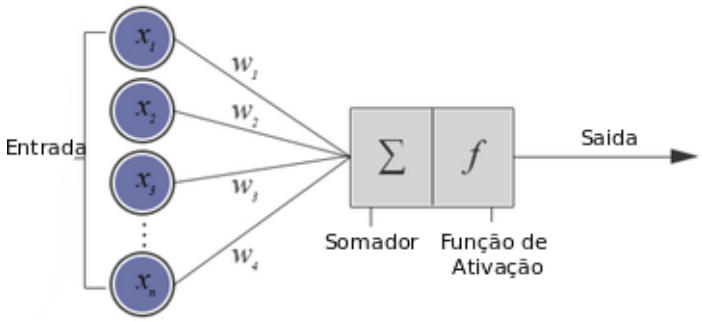


Figura 3 – Representação de um neurônio artificial, (WILLEMS, 2019)-modificado

Um ramo da Inteligência Artificial é o Aprendizado de Máquina (AM) (ML do inglês *Machine Learning*). Uma das principais atividades do AM se dá pela identificação de um conjunto de características que definem objetos, com um conjunto de dados parecidos, uma classificação.

Para exemplificar a classificação é proposto o seguinte exemplo, cachorro e gato pertencem a classe animais por apresentarem um conjunto de características em comum, mas ainda dentro da classe cachorro existem diferentes espécies.

Para realizar essa classificação os métodos tradicionais de AM se baseiam em três categorias diferentes, representadas na Fig. (6): Aprendizado Supervisionado, Aprendizado não Supervisionado e Aprendizado por Reforço.

1. **Aprendizado Supervisionado** funciona por meio de um rótulo (*label*) que facilita na classificação de um determinado dado em um conjunto, como por exemplo em um conjunto de imagens de animais, o rótulo cachorro facilita na identificação desse animal. Esse método trabalha tanto com classificação tanto com regressões de diferentes tipos.

2. **Aprendizado Não-Supervisionado** funciona por meio do reconhecimento de padrões de um determinado dado, caracterizando um conjunto por meio do número de

padrões já reconhecidos de um determinado grupo, um exemplo são os objetos apresentados no momento de uma compra online, de forma que esses objetos seguem o padrão do produto já comprado. Assim, lida com a ideia de Associação e *clustering*.

3. **Aprendizado por Reforço** se baseia em retribuir positivamente ações que estejam corretas e retribuir negativamente ações que estejam erradas, de forma que em algumas repetições o computador entenda qual ação deve tomar.

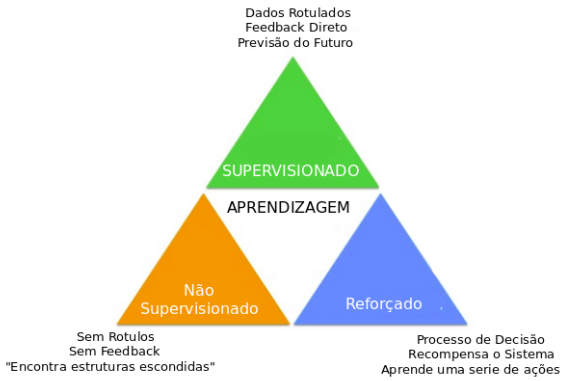


Figura 4 – Métodos de AM, (RASCHKA, 2015)-modificado.

Utilizando os princípios do Aprendizado Supervisionado para identificação é aplicada uma Rede Neural Artificial, que é o conjunto de *perceptrons*. Um esquema para uma Rede Neural Artificial é apresentado na Fig. (7), com apenas uma camada de neurônios oculta, o que diferencia de algoritmos de *Deep Learning* que trabalham com várias destas camadas.

Essas camadas funcionam por meio da conexão de vários Neurônios Artificiais, por meio de algoritmos do tipo *Feed-Forward*, Retropropagação do erro (do inglês *Backpropagation*) e Rede Neural Convolucional (CNN, do inglês *Convolutional Neural Network*). Será explicado o funcionamento do algoritmo CNN abaixo, já que este algoritmo utiliza os outros para sua implementação.

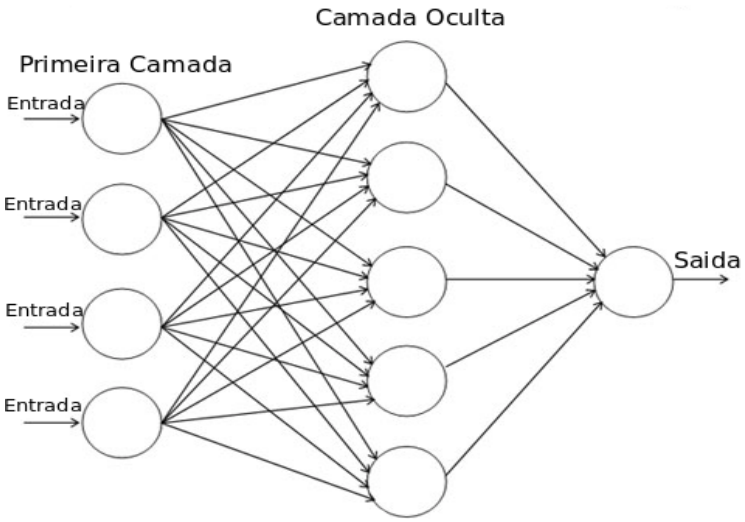


Figura 5 – Rede Neural Artificial com uma camada oculta, (MANZINI, 2017)-modificado

O algoritmo CNN é a base do funcionamento de algoritmos de Aprendizado Profundo que trabalham com a detecção de um determinado objeto em uma imagem, no caso de estudo deste trabalho um VANT.

Algoritmos do tipo CNN realizam cinco tipos principais de operações com o intuito de reconhecer padrões dentro de uma imagem, essas operações são: Convolução, Unidade de Retificação Linear (ReLU, do inglês *Rectified Linear Unit*), *Pooling*, *Feed-Forward* e o algoritmo de Retropropagação do erro.

Inicialmente é implementada uma convolução com um filtro, sem realizar a inversão de 180° do filtro utilizado, na primeira camada oculta, ou neste caso camada de convolução. É atribuído esse nome a essa camada, por conta da operação realizada ser relacionada com a convolução de dois sinais, porém tratando de uma convolução com um filtro invertido, mantendo a formulação, da Eq. 2.5, em que $g[x, y]$ é um filtro bidimensional invertido.

$$\sum_{n1=-\infty}^{\infty} \sum_{n2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]. \tag{2.1}$$

No final da convolução de cada camada é obtido uma imagem resultante de tamanho que pode ser variável, em detrimento da convolução realizada não ser efetuada com preenchimento ou ser feito um preenchimento com zeros para centralizar a convolução com o primeiro *pixel* adicional.

Cada camada oculta possui um filtro associado, de forma que na saída dessa camada obtenha se uma imagem reduzida para cada um dos neurônios associados. Para obter o resultado dessa camada são agregadas todas essas imagens.

Posterior a convolução realizada é aplicado uma função de ativação do tipo ReLU,

essa função é definida para introduzir uma não linearidade na decisão da Rede Neural Artificial, a função ReLU é definida por $f(x) = max(0, x)$ e retira do mapa de ativação, imagem em que a convolução foi realizada, os valores negativos que poderiam existir.

Após a etapa de ReLU, é aplicada a técnica de *Max Pooling*, que implica em aplicar um filtro na imagem, utilizando os valores máximos, criando uma nova imagem de tamanho definido, realizando uma sub amostragem da imagem em detrimento do tamanho do filtro que será aplicado na imagem, essa técnica pode ser observada na Fig. (8).

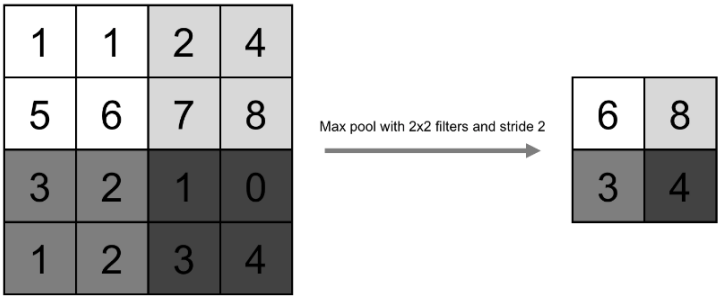


Figura 6 – Ilustração da técnica *Max Pooling*, (SHANMUGAMANI, 2019)-modificado

Após a etapa de *Max Pooling* é feito o *Feed-Forward* que implica em aplicar a função de ativação na imagem para cada nó $c = f(w_1x_1 + w_2x_2 + ... + w_ix_i + bs)$, com diferentes funções nesses nós e/ou diferentes pesos, esses pesos são iniciados de forma aleatória, uma distribuição gaussiana, por exemplo.

Em cada camada completamente conectada da RNA é aplicada essa técnica, extraindo características do dado analisado, sendo que no final da RNA será atribuído um valor para cada nó de saída e será comparado o valor dos nós de saída, que são a estimativa gerada para a imagem, com um vetor, ou matriz, que corresponde ao dado analisado. O vetor dos nós que mais se aproximar ao vetor real do dado será classificado como a resposta (SAZLI, 2006).

Por fim, é convertida a última matriz em um vetor de saída de forma que cada valor desse vetor corresponde a uma ponderação realizada pela CNN. Essa ponderação é responsável por detectar na imagem os padrões que correspondem ao objeto analisado.

Para realizar a atualização dos valores do vetor de saída é utilizado a Retropropagação. O princípio de funcionamento se baseia em atualizar os pesos do somatório ponderado, propagando o erro entre o vetor de saída e aos valores que representam perfeitamente o parâmetro de entrada da RNA. Dessa forma o erro associado aos valores do vetor final e ao valores que representam perfeitamente o objeto avaliado são minimizados (SAZLI, 2006).

Como inicialmente todos os pesos w_i, Fig. (5), são gerados aleatoriamente, ao utilizar uma imagem pelo algoritmo de uma Rede Neural Artificial profunda o vetor de

resposta não terá o casamento perfeito, com acurácia total, para a detecção da classe do treinamento.

Considerando uma Rede Neural Artificial com múltiplas camadas e considerando uma camada de saída j, o erro de saída para n-ésima pode ser definido como $e_j(n) = d_j - y_j(n)$, em que d_j é a saída predita para o neurônio avaliado, e y_j é a real saída, calculada pelos pesos iniciais.

A energia instantânea para o neurônio j pode ser obtida por $\epsilon_j(n) = \frac{1}{2}e_j^2(n)$, como na última camada os neurônios vistos pela saída são os próprios, o erro do sinal pode ser diretamente calculado, para todos os neurônios da camada j,

$$\epsilon(n) = \frac{1}{2}\sum_{j\in Q} e_j^2(n), \tag{2.2}$$

em que Q é o conjunto de todos os neurônios da camada de saída.

Supondo N padrões a serem analisados no treinamento o erro médio quadrado para a RNA é dado por:

$$\epsilon_{av} = \frac{1}{N}\sum_{n=1}^{N} \epsilon(n), \tag{2.3}$$

É importante ressaltar que o erro é associado para o conjunto de dados da RNA (função de ativação, pesos e entrada). A saída do neurônio j é definida por:

$$y_j(n) = f\left(\sum_{i=0}^{u} w_j i(n) y_i(n)\right), \tag{2.4}$$

em que o valor de u é o total de entradas do neurônio j, excluindo o *bias* da camada anterior, e f é a função de ativação do neurônio, assim como já explicado. Os pesos que serão atualizados e aplicados no neurônio j são proporcionais à derivada parcial do erro da energia instantânea, $\epsilon(n)$, com relação ao peso correspondente, por exemplo $\frac{\partial \epsilon(n)}{\partial w_j i(n)}$ e utilizando a regra da cadeia é obtida a Eq. 2.9,

$$\frac{\partial \epsilon(n)}{\partial w_j i(n)} = \frac{\partial \epsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial w_j i(n)}. \tag{2.5}$$

Pelas equações do erro de saída, da energia instantânea e Eq. 2.8 é factível obter as derivadas da regra da cadeia Eq. 2.9, obtendo:

$$\frac{\partial \epsilon(n)}{\partial e_j(n)} = e_j(n); \qquad \frac{\partial e_j(n)}{\partial y_j(n)} = -1; \tag{2.6}$$

já para o valor de $\frac{\partial y_j(n)}{\partial w_j i(n)}$ o resultado da derivação depende da função da ativação f, uma resposta genérica é obtida por:

$$\frac{\partial y_j(n)}{\partial w_j i(n)} = f'\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right) y_i(n), \tag{2.7}$$

em que, $f'\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right))$ é definido por:

$$f'\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right) = \frac{\partial f\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right)}{\partial\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right)}. \tag{2.8}$$

Estas derivações podem ser substituídas na Eq. 2.9 obtendo a Eq. 2.13,

$$\frac{\partial \epsilon(n)}{\partial w_j i(n)} = -e_j(n) f'\left(\sum_{i=0}^{u} w_{ij}(n) y_i(n)\right) y_j(n). \tag{2.9}$$

Com a correção $\Delta w_{ji}(n)$ aplicada à $w_{ji}(n)$ é possível definir a regra delta Eq. 2.14,

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} = \eta c_i \delta_j, \tag{2.10}$$

η corresponde ao parâmetro da taxa de aprendizado do algoritmo de retropropagação, que usualmente é colocado como constante no início do treinamento da Rede Neural Artificial (SAZLI, 2006).

Um exemplo do algoritmo de retropropagação pode ser observado na Fig. (9), em que são apresentadas algumas funções de ativação, com a saída da RNA com resultado de valor 1.

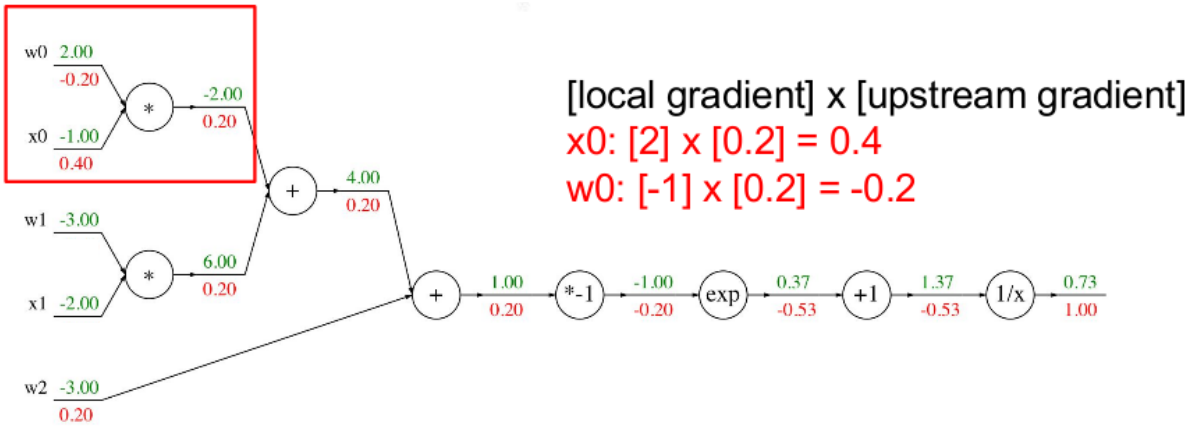


Figura 7 – Ilustração da técnica de retropropagação, (LI JUSTIN JOHNSON, 2017)-modificado

Para gerar essa área o processo típico de Aprendizado de Máquina é primeiramente treinada a Rede de Neurônios Artificiais com um conjunto de dados tanto positivos, contendo o dado a ser analisado, quanto negativos, não contendo esses dados, garantindo uma quantidade suficiente para uma boa análise probabilística.

Em seguida, deve-se utilizar um novo *dataset*, de forma a validar os coeficientes gerados na RNA, observando as medidas de erro. Por fim, deve-se utilizar um terceiro conjunto de dados, dados de teste, de forma a perceber o erro da sua Rede Neural Artificial, já minimizado na validação.

A saída de um algoritmo de classificação pode ser discreta, definindo de forma binária se é um objeto ou não, e ainda de forma probabilística garantindo um número de zero a um da probabilidade de pertencer a determinada classe avaliada.

2.1.1 Transfer Learning

Outra forma de realizar o treinamento da Rede Neural Artificial é por meio de algoritmos de *Transfer Learning*. *Transfer learning* consiste em utilizar conhecimento prévio de dados treinados ou características já identificadas, de um ou mais padrões já identificados para desenvolver uma hipótese para uma nova tarefa (I.Guyon et al., 2019).

Considerando um exemplo de duas pessoas aprendendo a tocar piano. Uma delas possui um extenso conhecimento de como tocar violão, enquanto que a outra nunca tocou nenhum instrumento musical. A pessoa com extenso conhecimento em tocar violão possuirá maior facilidade para aprender tocar piano, este é o cenário do *Transfer Learning*.

A necessidade do *Transfer Learning* é percebida com maior facilidade quando há a necessidade de realizar uma tarefa (seja detecção de um objeto ou a aprendizado de determinada informação) que não possui um vasto conjunto de dados.

Para aplicar as técnicas de *Transfer Learning* em algoritmos de Aprendizagem Profunda é necessário congelar os pesos de várias camadas da RNA que será implementada, de forma que durante o processo de retropropagação do erro, da nova classe que será associada a RNA, os pesos treinados por um *dataset* com uma quantidade reduzida de dados, modifiquem apenas camadas mais externas a RNA, tais como as camadas completamente conectadas.

A definição formal de *Transfer Learning* é dada por um domínio D, definido por um vetor de padrões X do espaço e uma distribuição marginal de probabilidade $P(X)$, em que $X = x_1, x_2, ..., x_n \epsilon X$. Para um determinado domínio uma tarefa τ é definida por duas partes, um espaço de rótulos Y e uma função de predição $F(.)$, que foi desenvolvida pelo par (x_i, y_i), em que $y_i \epsilon Y$ e $x_i \epsilon X$ e pelo vetor de padrões X.

Um domínio fonte é definido por $D_s = (x_{s1}, y_{s1}), ..., (x_{sn}, y_{sn})$, em que $x_{si} \epsilon X_s$ e $y_{si} \epsilon Y_s$, são a i-ésima data do vetor e o i-ésimo rótulo do vetor respectivamente. Da mesma forma um domínio alvo é definido por $D_t = (x_{t1}, y_{t1}), ..., (x_{tn}, y_{tn})$, em que $x_{ti} \epsilon X_t$ e $y_{ti} \epsilon Y_t$, são a i-ésima data do vetor e o i-ésimo rótulo do vetor respectivamente. Ambos domínios D_s e D_t possuem tarefas τ_s e τ_t com suas funções de predição $F_s(.)$ e $F_t(.)$.

Transfer Leaning pode ser então definido por um domínio fonte D_s com sua tarefa τ_s e um domínio alvo D_t e sua tarefa τ_t de forma que *Transfer Learning* é o processo de aperfeiçoamento da função de predição $F_t(.)$ utilizando as informações do domínio D_s e τ_s, $D_s \neq D_t$ em que e $\tau_s \neq \tau_t$, que é a definição de (Pan; Yang, 2010).

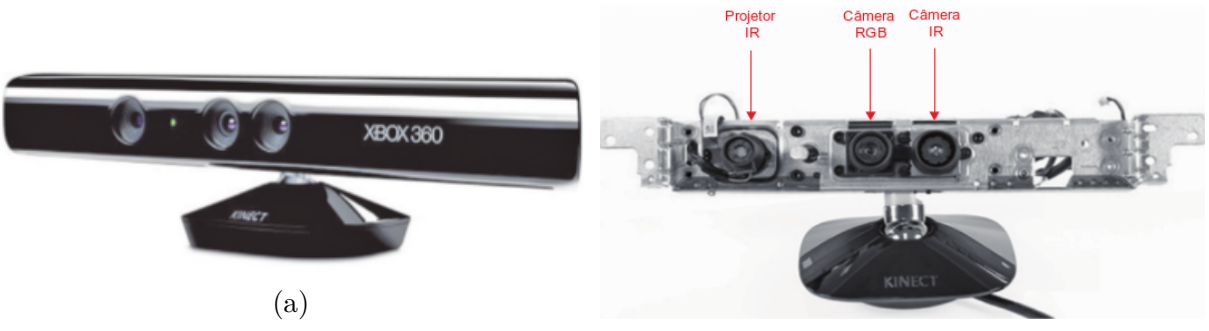
2.2 Sensor *kinect*

O sensor *Kinect* inicialmente foi desenvolvido pela empresa Microsoft e lançado em 4/11/2010 para o ambiente de jogos, com o intuito de melhorar a experiência, permitindo a interação corpo humano máquina, por meio do entendimento da movimentação do corpo humano.

Além disso a atividade de estimação de posição 3D, (que é uma atividade com certa dificuldade no campo de visão computacional, quando se tratando de câmeras comuns) é facilitada por meio do sensor *Kinect*, já que ao incorporar vários equipamentos de detecção (sendo eles: uma matriz de quatro microfones, um sensor de profundidade e uma câmera colorida que fornece a captura de movimentos 3D), esta dificuldade é facilitada, de forma a viabilizar diversas pesquisas no campo de visão computacional (ZHANG, 2012).

O sensor de profundidade consiste no projetor *infrared* (IR) combinado com a câmera IR, que é um sensor monocromático Semicondutor de Óxido de Metal complementar (CMOS). O projetor IR é um laser IR que passa por uma grade de difração resultando em um conjunto de pontos IR (ZHANG, 2012).

O sensor tem uma resolução de 640x480 pixels e uma taxa de quadros (*frames*) de 30fps (*frames* por segundos), além de uma precisão de 11 bits para o projetor IR. O sensor *Kinect* é mostrado na Fig. (3a) e os componentes são mostrados na Fig. (3b).



(a) (b)

Figura 8 – 3a Sensor *Kinect*, 3b sensor *Kinect* aberto, visão dos componentes. Ambas figuras editadas de (ZHANG, 2012)

Para que haja uma acurácia maior na aquisição dos dados pelas câmeras do *Kinect* é realizada a aproximação por câmeras *pinhole*, câmeras sem lente ou câmara estenopeica, com uma certa distorção na imagem, sendo os dois tipos principais de distorção as distor-

ções radiais e as distorções tangenciais, essas distorções podem causar um erro de medição elevado de acordo com (QI; LI; ZHENZHONG, 2010).

As distorções radiais ocasionam uma aparência curvilínea em linhas que deveriam ser retas Fig. (4), piorando quanto maior é a distância do centro da imagem. Estudando a geometria das câmeras *pinhole* é possível definir a distorção pelo modelo polinomial (PM, do inglês (*Polynomial Model*) (WANG; QIU; SHAO, 2009), definido abaixo:

$$r_n = r_d(1 + \lambda_1 r_d^2 + \lambda_2 r_d^4 + \lambda_3 r_d^6), \tag{2.11}$$

em que r_d e r_n são as distâncias entre os pontos distorcidos (x_d, y_d) e os pontos não distorcidos (x_n, y_n) com relação ao centro distorcido P, respectivamente; e λ_i é um parâmetro da distorção radial.

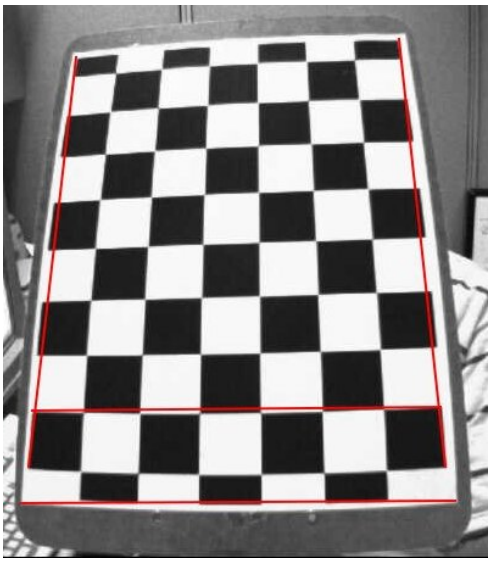


Figura 9 – Distorção radial (BRADSKI, 2000)

As distorções tangenciais são ocasionadas por conta de um alinhamento imperfeito da lente de captura da imagem, dessa forma alguns pontos da imagem aparentam estar mais próximos do que a realidade (BRADSKI, 2019). As distorções tangenciais podem ser definidas pelas Eq. 2.2 e Eq. 2.3.

$$x_{corrigido} = x_d + [2p_1 x_d y_d + p_2(r_d^2 + 2x_d^2)], \tag{2.12}$$

$$y_{corrigido} = y_d + [p_1(r_d^2 + 2y_d^2) + 2p_2 x_d y_d], \tag{2.13}$$

sendo p_i parâmetros de distorção tangencial.

Além desses dados ainda é necessário encontrar parâmetros intrínsecos e extrínsecos. Parâmetros intrínsecos são específicos da câmera utilizada, como por exemplo a

distância focal da lente (df_x, df_y) e o centro óptico (co_x, co_y). Com esses dados é factível montar uma matriz com os dados únicos da câmera utilizada, como a matriz observada na Eq. 2.4.

$$matriz\_camera = \begin{pmatrix} df_x & 0 & co_x \\ 0 & df_y & co_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.14}$$

Os parâmetros extrínsecos correspondem aos vetores de rotação e translação que realizam a transformação de um ponto tridimensional para o um sistema de coordenadas. Para encontrar esses parâmetros é preciso gerar algumas imagens de teste com um padrão bem definido em que as posições são conhecidas, como por exemplo um tabuleiro de xadrez Fig. (4).

Para que possa ser feita a correlação entre o sistema de coordenadas real para o sistema de coordenadas da imagem, encontrando os coeficientes de distorção para as Eq. 2.1, Eq. 2.2 e Eq. 2.3.

2.3 Revisão dos trabalhos correlatos

Explicados os principais conceitos é realizada a revisão dos trabalhos correlatos. Em (KAR; AMITABHA; GUHA, 2019) são utilizados métodos de segmentação na câmera de profundidade, para obter a diferença do corpo humano do fundo do local analisado, para que em seguida fossem aplicados os classificadores de Haar pelo algoritmo Viola-Jones, detectando se seria um ponto do corpo ou do rosto.

Apesar de obter bons resultados em detecção quando o objeto está parado Fig.(12), obtendo o esqueleto humano, o algoritmo falha na detecção quando há movimentos no sentido da câmera, além de trabalhar em uma taxa de 10 quadros por segundo.

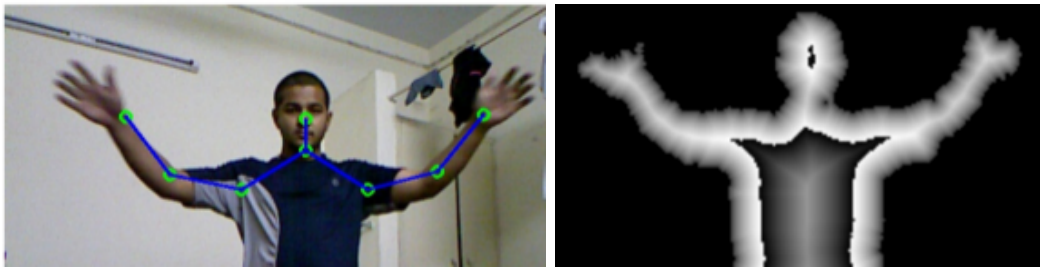


Figura 10 – Rastreamento de posição do corpo, (KAR; AMITABHA; GUHA, 2019)-modificado.

Em (BOUDJIT; LARBES; ALOUACHE, 2013) é utilizado a identificação do esqueleto humano Fig. (13), para gerar *setpoints* de arfagem, rolagem e guinada para o quadrirrotor, realizando a integração do conjunto *Kinect*, computador e quadrirrotor.

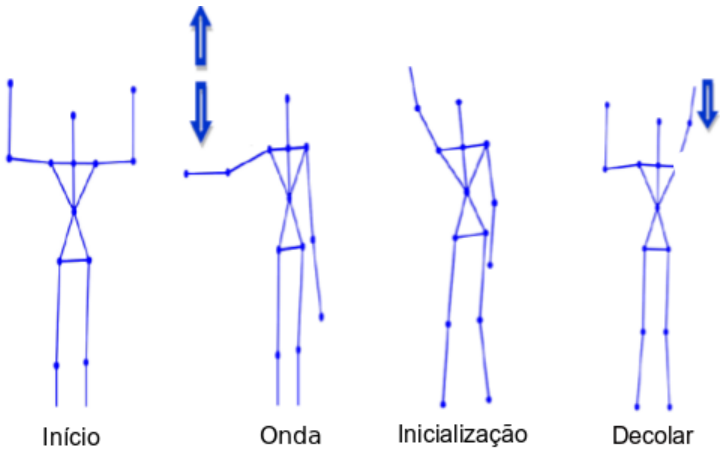


Figura 11 – Rastreamento de posição do corpo, (BOUDJIT; LARBES; ALOUACHE, 2013)-modificado.

Uma abordagem realizando a leitura da câmera de profundidade para o rastreamento de pequenos movimentos com as mãos é apresentado em (FRATI; PRATTICHIZZO, 2011), que encontra a caixa delimitadora onde a mão se encontra por meio de *threshold*, transformando em uma imagem binária. Em seguida é aplicado a função do OpenCv $cvFindContours()$, retornando o número de contornos da imagem para assim aplicar a convex Hull encontrando os contornos da mão.

Para encontrar a ponta dos dedos é aplicada a $cvConvexityDefects()$, retornando o valor da ponta dos dedos, sendo capaz de rastrear pequenas variações com a filtragem de Kalman, por meio da estimação da posição. As limitações dessa abordagem se dão por conta da limitação da proximidade da câmera para realizar o *threshold* e a necessidade de não ocorrer oclusão do objeto, necessitando que este esteja sempre a uma distância fixa.

Uma outra maneira de realizar a identificação do quadrirrotor é por meio de algoritmos de Aprendizado de Máquina (AM) (*Machine Learning*) e Aprendizado Profundo (*Deep Learning*) utilizando o reconhecimento de padrões (*pattern recognition*).

Para determinar se é ou não o objeto avaliado, é realizado um estudo estatístico adquirido por meio de padrões, retirados de um conjunto de dados (*dataset*) e por meio desses realizar previsões futuras validando a análise estatística realizada (SIVA, 2018).

Em (Lee; Gyu La; Kim, 2018) é utilizada uma CNN utilizando também o classificador de Haar baseado em padrões, da biblioteca OpenCv, para realizar a detecção. Com uma CNN com duas camadas de convolução e duas camadas totalmente conectadas, obtendo uma taxa de acerto de 90%.

Outro trabalho de identificação é o de (Rozantsev; Lepetit; Fua, 2017) se baseia na distinção de objetos voadores, utilizando uma câmera acoplada a um quadrirrotor. É aplicado a técnica de Aprendizado de Máquina por meio do classificador de AdaBoost, conjuntamente com um estabilizador para a câmera do quadrirrotor, obtendo uma taxa de

acerto de 75% para o *dataset* de VANTs analisados. Dentre os dados analisados o algoritmo utilizado é eficaz para imagens com o fundo ruidoso, ou ainda em imagens tiradas com uma câmera distante, com o propósito de evitar colisões entre objetos voadores.

Um trabalho semelhante é o de (Saqib et al., 2017) que aplica uma CNN com aprendizado profundo com base na *faster RCNN*, que possui uma boa acurácia em imagens de longa distância, obtendo 60% de taxa de acerto, porém necessita de um grande poder computacional, gastando tempo para realizar a detecção, não sendo adequada para aplicações *real-time*.

Outro trabalho que realiza a detecção de quadrirrotores é o (Aker; Kalkan, 2017), que utiliza o algoritmo *You Only Look Once version 2* (YOLOv2), baseado em CNNs, porém caracterizando a imagem com apenas uma ajuste de pesos pela Rede Neural Artificial. Essa abordagem permite classificar mais de um objeto voador, possuindo uma acurácia de 90% com uma maior velocidade de operação, por conta de aplicar a imagem apenas uma vez em sua CNN.

Uma explicação dos conceitos de Aprendizado de Máquina e a diferenciação com Aprendizado Profundo é melhor explorada no subseção 2.4, nesta subseção é tratado o trabalho de (Carrio et al., 2018), que realiza a detecção do VANT e o mapeamento da posição tridimensional por meio de uma CNN do tipo YOLOv2, em que a aquisição é feita por câmeras de profundidade.

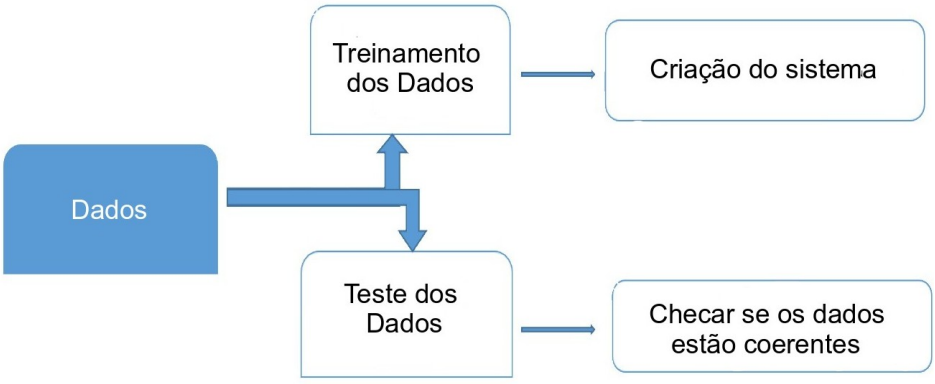


Figura 12 – Funcionamento básico de um reconhecimento de padrões, (SIVA, 2018)-modificado.

Após a identificação do quadrirrotor são utilizadas técnicas de localização ou rastreamento, que permitem por meio de uma imagem ou conjunto de imagens em sequência (video) determinar a posição do veículo no espaço, considerando a posição inicial a câmera, alguns desses algoritmos são: Lucas Kanade (Siong et al., 2009), MIL, do inglês *Multiple Instance Learning* (Babenko; Yang; Belongie, 2009), KCF, do inglês *Kernelized Correlation Filter* (Henriques et al., 2015), TLD, do inglês *Tracking Learning Detection* (Kalal; Mikolajczyk; Matas, 2012), MOSSE, do inglês *Minimum Output Sum of Squared*

Error (Bolme et al., 2010).

Dentre os algoritmos de rastreamento analisados em (JANKU et al., 2016) o algoritmo que mais se destacou foi o MIL apresentando uma melhor precisão com relação a *bounding box* e conseguindo se restabelecer com uma oclusão parcial. Apesar dos resultados apresentados do MIL, um algoritmo que possui resultados de rastreamento ainda mais preciso e com uma velocidade ainda melhor é o KCF, por conta de ser baseado nos princípios do MIL, em que múltiplas amostras positivas possuem grandes posições sobrepostas (MALLICK, 2017).

Na seção 2.2.2, é descrito o método de (SANTANA; SANTOS; VIANA, 2016) que descreve uma técnica parecida com a situação de interesse deste TCC, que é uma aplicação de rastreamento de posição, utilizando um computador conectado a um sensor *Kinect*.

2.3.1 *Kinect* fixado no teto

Uma das técnicas empregadas para rastreamento, que utiliza o sensor *Kinect*, realiza a identificação em três dimensões de um círculo vermelho no centro do VANT, o sensor é fixado no teto e apontado no sentido do piso, sendo o piso o fundo em que o quadrirrotor está situado.

Um outro componente utilizado é um computador fora do alcance das câmeras do sensor, sendo que neste componente são realizados todas as estimações de filtros de Kalman, além de realizar o processamento de imagem para identificar o círculo vermelho e ainda realizar a conversão das distâncias do círculo para milímetros, utilizando o método *pinhole*. A comunicação do sensor *Kinect* com o computador é realizada via uma interface USB. O sistema utilizado é ilustrado na Fig. (10).

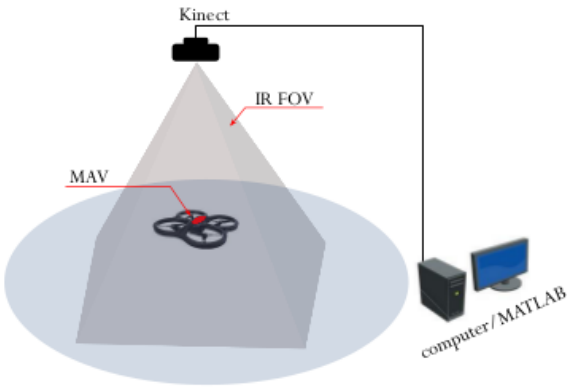


Figura 13 – Esquemático do sistema montado (SANTANA; SANTOS; VIANA, 2016)

2.4 Detecção do quadrirrotor

Dentre as técnicas avaliadas de identificação de VANTs, utilizando *Machine Learning* e *Deep Learning*, uma que melhor atende os requisitos do presente trabalho (que são o mapeamento da posição bidimensional e/ou tridimensional em tempo real) por trabalhar especificamente com o mapeamento da posição 3D por meio da Rede Neural Artificial YOLOv2, é descrita em (Carrio et al., 2018) e será explicada nesta seção.

O método proposto pelo artigo para a detecção do VANT diferentemente dos demais citados que trabalham com imagens no espaço de cores RGB, trabalha com a variação da intensidade da profundidade de um ambiente, de forma a obter uma diferenciação maior entre o fundo da imagem e o VANT avaliado. Dessa forma, foi utilizado um *dataset* que possuía as imagens da câmera de profundidade avaliada, incluindo o fundo e uma imagem segmentada com o apenas o quadrirrotor incluso.

Baseado do *dataset* foi treinada uma Rede Neural Artificial profunda, com base no algoritmo de YOLOv2 (Redmon; Farhadi, 2017) que é um dos algoritmos de detecção de objetos mais rápidos, obtendo uma das melhores respostas de velocidade e precisão de acordo com (Carrio et al., 2018).

O algoritmo *You Only Look Once* (YOLO) tem o princípio de funcionamento por meio de uma Rede Neural de Convolução aplicada apenas uma vez na imagem analisada, obtendo as caixas delimitadoras para o objeto. É aplicada uma única Rede Neural de Convolução para a imagem inteira de tamanho SxS, dividindo a imagem em uma grade SxS. Quando o centro do objeto está em uma das células da grade a predição da detecção do objeto é realizada por esta célula da grade, de tamanho mxm.

A grade de detecção é definida por cinco parâmetros, o centro da imagem (x, y), a largura e comprimento, respectivamente, (w, h) da caixa e a confiança C_f da predição, que é definida por meio da *Intersection Over Union (IOU)*. São geradas duas caixas delimitadoras (*bounding boxes*), para cada grade mxm (*grid*). Essas caixas geradas recebem uma probabilidade de confiança com base no número de classes C avaliadas no treinamento da Rede Neural Artificial, determinando o objeto em detrimento da confiança recebida Eq. 2.15 (Redmon et al., 2016), Fig. (16). A predição do algoritmo Yolo é um tensor de dimensão $SXSX(B * 5 + C)$ contando a detecção dos objetos:

$$C_f = P_r(objeto) * IoU, \tag{2.15}$$

em que IoU é a métrica que permite avaliar quão próximo da caixa delimitadora perfeita a caixa prevista pela Rede Neural Artificial está, definida na Eq. 2.16.

$$IoU = \frac{R_s}{R_c}, \tag{2.16}$$

em que R_s é a região sobreposta, área da caixa verde na Fig. (15), e R_c é a região combinada, soma das áreas das caixas subtraindo a interseção (caixa verde).

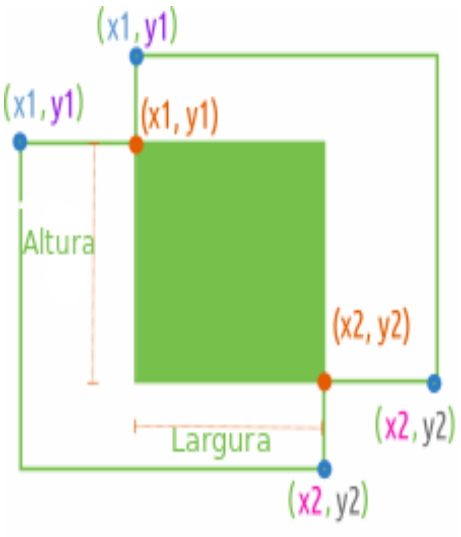


Figura 14 – Métrica IoU, (RESTREPO, 2019)-modificado.

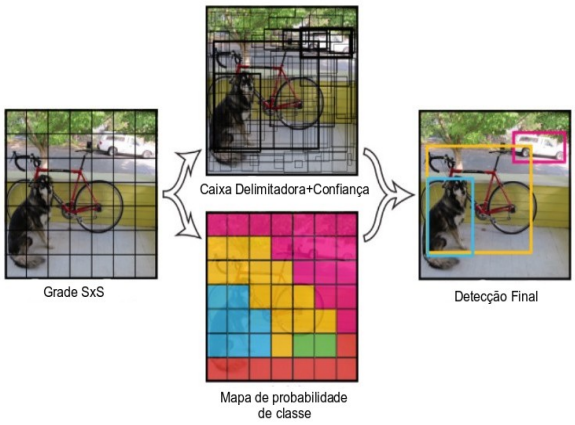


Figura 15 – Caixas delimitadoras do algoritmo Yolo, (Redmon et al., 2016)-modificado.

O algoritmo YOLOv3 (versão atualizada do YOLO) utiliza alguns procedimentos para otimizar o acerto e a velocidade do algoritmo, assim como mais camadas de convolução. Esses procedimentos estão descritos em (REDMON; FARHADI, 2018). Além disso, é necessário realizar um procedimento para minimizar os erros do processo de casamento dos pontos de profundidade.

O método proposto para minimizar esses erros de acurácia é baseado em encontrar um ponto $P_i \in P$ em que o mapa das profundidades melhor indique o centro do VANT, em que $P = \{P_1, P_2, ...P_n\}$ são os pontos 2D dentro de uma *bounding box*, com as profundidades associadas $Z = \{Z_1, Z_2, ...Z_n\}$. Isso é realizado escolhendo um P_i tal que $i = argmin(|Z_i - Z_{ref}|)$. Para encontrar Z_{ref} dois métodos são propostos:

1. Escolher o ponto de profundidade 2D mais próximo ao valor da média dos 25% menores pontos dentro da caixa delimitadora $Z_{ref} = mean(Z_i) \quad \forall \quad Z_i < Q_1$, em que Q_1 é o primeiro quartil de Z.

2. Escolher o ponto de profundidade 2D mais próximo do valor da mediana dos 25% menores pontos dentro da caixa delimitadora $Z_{ref} = median(Z_i) \quad \forall \quad Z_i < Q_1$.

Escolhido o método que melhor se adéque, é factível realizar a estimação da localização 3D correspondente ao ponto $P_i(u, v)$ com discrepância d_i, utilizando uma câmera de profundidade. Para realizar essa estimação é utilizada a Eq. 2.16

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{T}{c_x^e - c_x^d - d} \begin{pmatrix} u - c_x^e \\ v - c_x^d \\ f^e \end{pmatrix}, \tag{2.17}$$

em que f^e é a lente focal esquerda, $C^e(c_x^e, c_y^e)$ é o centro óptico da câmera esquerda e $C^d(c_x^d, c_y^d)$ é o centro óptico da câmera da direita.

2.5 Fundamentação teórica do algoritmo de rastreamento KCF

Após a identificação do VANT é possível aplicar um algoritmo de rastreamento, já que o objeto foi encontrado, para garantir a estabilidade da detecção. O objetivo do rastreamento é estimar o estado x_k, especificado pela equação da dinâmica $x_k = f_k(x_{k-1}, v_k)$ por meio de medidas z_k, definidas como $z_k = h_k(x_k, n_k)$, relacionadas diretamente com os estados a ser estimado.

Em geral f_k e h_k são funções não lineares e variantes no tempo e as sequências de ruído v_k e n_k são independentes e com distribuição idêntica. Logo estimar o estado x_k é equivalente a construir a função de densidade de probabilidade (PDF, do inglês) $p(x_k|z_{1:k})$ (Comaniciu; Ramesh; Meer, 2003).

As sequências de ruído são definidas em detrimento das condições das suas funções, por exemplo, se f_k e h_k são lineares e a sequência é uma Gaussiana, a solução ótima perfeita é fornecida por um filtro de Kalman.

O algoritmo *Kernelized Correlation Filters* (KCF) se baseia na detecção do vetor de direção entre os *frames* atuais e os *frames* de um estado anterior conhecido, procurando na vizinhança o mesmo conjunto de pontos em cada quadro consecutivo, quando esses pontos são encontrados a *bounding box* é movimentada para esses novos pontos realizando assim o rastreamento.

Para entender como o KCF constrói os blocos limitadores, é preciso entender o algoritmo de regressão linear, regressão do cume, que tem o objetivo de encontrar uma

função $f(z) = m^T z$, que minimiza o erro quadrático sobre as amostras x_i e suas metas de regressão y_i (Henriques et al., 2015).

$$\min_w (\sum_i (f(x_i) - y_i)^2 + \lambda \|m\|^2, \tag{2.18}$$

em que λ é um parâmetro de regularização que controla o *overfitting*, e o minimizador é dado por

$$m = (X^T X + \lambda I)^{-1} X^T y, \tag{2.19}$$

como dito anteriormente o minimizador m a matriz X tem uma amostra por linha x_i, cada elemento y tem suas metas de regressão y_i e I é a matriz identidade. Para simplificar o trabalho em uma análise na frequência, é utilizada a versão complexa da Eq. 2.19.

$$m = (X^H X + \lambda I)^{-1} X^H y, \tag{2.20}$$

X^H representa a Hermitiana transposta de X.

2.5.1 Deslocamento Cíclico

Em geral um número grande de sistemas lineares devem ser calculados para resolver o problema da equação 2.20, o que pode ser um fator impeditivo para o cálculo em tempo real do minimizador, desta forma um caso especial de x_i que supera esse problema é apresentado, por meio da técnica de Deslocamento Cíclico (do inglês *Cyclic Shifts*).

A definição dessa técnica para uma dimensão é apresentada, estendida diretamente para imagens bidimensionais (Henriques et al., 2015). Considerando um vetor de tamanho $nx1$ contendo um fragmento do objeto de interesse x, chamado de amostra base. O objetivo é treinar um classificador, por meio da amostra base e a translação dessa amostra, obtendo amostras virtuais Fig. (17).

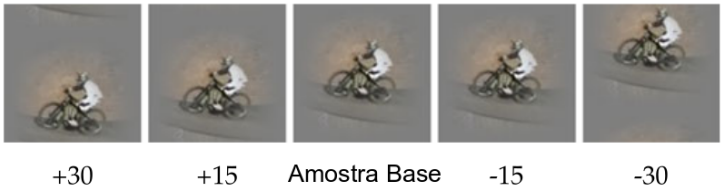


Figura 16 – Exemplo de Deslocamento Cíclico. Editada de: (Henriques et al., 2015)

A matriz com os valores permutados da amostra base é apresentada na Eq. 2.21. O produto de $Cx = [x_n, x_1, x_2, \ldots, x_{n-1}]^T$ é responsável por realizar o deslocamento de

uma unidade, realizando uma pequena translação,

$$C = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}. \tag{2.21}$$

2.5.2 Matriz Circulante

Por conta da operação de *Cyclic Shifts* é possível obter o sinal x periodicamente a cada n deslocamentos, $\{C^u x | u = 0, \dots, n-1\}$, com uma translação mais brusca C^u. Outro fator observado é que a primeira metade dos valores se movimenta na direção positiva, enquanto que a segunda metade dos valores encontra-se caminhando na direção negativa. Para calcular uma regressão, com amostras deslocadas, pode-se utilizar a periodicidade do sinal para definir uma matriz de circulação C_x, Eq. 2.22,

$$C_x = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ x_n & x_1 & x_2 & \dots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \dots & x_{n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_2 & x_3 & x_4 & \dots & x_1 \end{bmatrix}. \tag{2.22}$$

Um exemplo da matriz de circulação Eq. 2.22 pode ser observado na Fig. (17). A característica mais importante é que todas as matrizes de circulação são transformadas em matrizes diagonais, pela transformada discreta de Fourier (DFT), independente do vetor que as gerou (Henriques et al., 2015).

$$C_x = J diag(\hat{x}) J^H, \tag{2.23}$$

em que a matriz J é uma matriz constante independente de x, conhecida como matriz DFT e é a matriz única que calcula a DFT de qualquer vetor e $\hat{x}$ é a DFT do vetor x.

Trabalhando com dados treinados em formato de *cyclic shifts* é vantajoso, já que permite trabalhar com operações do tipo *element-wise* (operações ponto a ponto, no caso de multiplicação entre matrizes se trata de uma multiplicação elemento a elemento), por se tratar de uma matriz diagonal. Aplicando a Eq. 2.23 em Eq. 2.20 obtém-se:

$$C_x^H C_x = J diag(\hat{x}^* \odot \hat{x}) J^H. \tag{2.24}$$

Já que as matrizes diagonais são simétricas, a Hermitiana apenas deixa um termo complexo conjugado $\hat{x}^*$ e a multiplicação matricial $J^H J = I$. O termo $\odot$ representa a

operação produto *element-wise* entre matrizes diagonais. Aplicando a Eq. 2.24 em Eq. 2.20 obtém-se a Eq. 2.25, que tem como solução um filtro de correlação regularizado (Henriques et al., 2015).

$$m = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}. \tag{2.25}$$

2.5.3 Regressão não linear por meio do *Kernel Trick*

Uma forma de obter uma análise mais poderosa é utilizar o Truque de *Kernel* (*Kernel Trick*), que permite realizar a interpretação de um problema não linear em um problema linear, por meio do mapeamento em um espaço de dimensão mais elevada, por exemplo $\Omega = I = \mathbb{R}^2 \rightarrow F = \mathbb{R}^3$ (HOFMANN, 2006).

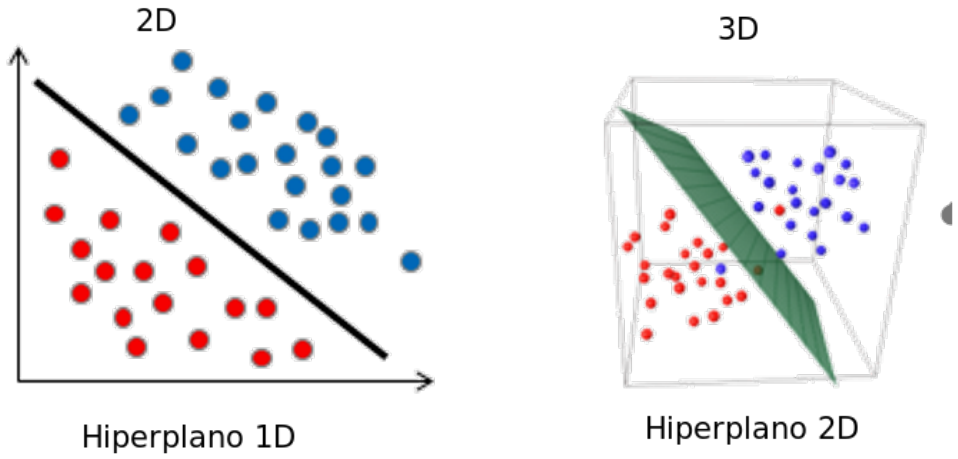


Figura 17 – Exemplificação do truque de kernel, (SHARMA, 2019)-modificado.

Isso permite utilizar uma regressão não linear $f_{(z)}$, de forma que o problema de otimização permanece linear, só que em um sistema de duas variáveis. O truque de Kernel é expresso na Eq. 2.26, que mapeia um problema com características não lineares em um espaço (com dimensão elevada) linear de características $\varphi(x_i)$, Fig. (18), tornando a função m na combinação linear das amostras, em que α está em um espaço bidimensional e é o responsável pela correlação em um novo espaço,

$$m = \sum_i \alpha_i \varphi(x_i). \tag{2.26}$$

Escrevendo o algoritmo em termo de um produto ponto a ponto $\varphi^T(x)\varphi(x') = k(x, x')$, que são calculados pela função *Kernel* k, como por exemplo uma Gaussiana. A resposta para a regressão de cume Eq. 2.23, por meio do truque de *Kernel* é a Eq. 2.28.

O produto ponto a ponto entre todos os pares de amostra geralmente é armazenado em uma matriz *Kernel* K de tamanho nxn e com elementos $K_{i,j} = k(x_i, x_j)$,

A principal vantagem desse truque é utilizar um espaço de dimensão superior de forma implícita, sem instanciar um vetor nesse espaço, logo $f(z)$ pode ser escrita como:

$$f(z) = m^T z = \sum_{i=1}^{n} \alpha_i k(z, x_i). \tag{2.27}$$

A desvantagem dessa abordagem é que a complexidade do truque de *Kernel* cresce para o número de amostras, como pode ser observado na Eq. 2.27, o que inviabilizaria o cálculo, já que o número de amostras é o valor que auxilia em um bom rastreamento, por conta de aumentar a quantidade de padrões percebidos.

2.5.4 Regressão Rápida de *Kernel*

Para evitar a desvantagem do truque de *Kernel* utiliza-se uma abordagem diagonalizada da Eq. 2.28, definindo alguns tipos específicos de *kernel*, como por exemplo: *Radial Basis Function kernels*, *Dot-product kernels*, *Additive kernels*, *Exponentiated additive kernels*. A solução para a versão de *kernel* da regressão de cume é dada por:

$$\alpha = (K + \lambda I)^{-1} y, \tag{2.28}$$

em que K é a matriz de *Kernel*, α é o vetor com coeficientes α_i, que representa a solução no espaço bidimensional. Definidos os *Kernels* que permitem montar uma matriz K circular é possível estender a solução linear para diagonalizar a função definida na Eq. 2.28 de forma a obter a Eq. 2.29.

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}, \tag{2.29}$$

em que $\hat{k}^{xx}$ é a primeira linha da matriz K aplicada a DFT. Para melhor entender $\hat{k}^{xx}$ é realizada uma definição mais genérica, aplicando a correlação de *kernel* em dois vetores arbitrários x e x', obtendo o vetor:

$$\hat{k}^{xx'} = k_i(x', C_x^{i-1} x), \tag{2.30}$$

que representa a correlação entre x e sua representação no domínio de Fourier, porém com a rotação cíclica já apresentada. Essa abordagem permite calcular apenas um vetor $n \times 1$, limitando o número de amostras a serem calculadas pelo truque de *kernel*, resolvendo a desvantagem apresentada.

2.5.5 Rastreamento múltiplo

Para uma avaliação mais ampla e rápida, visando avaliar diversos locais diferentes na mesma imagem, utiliza-se a segmentação desta imagem, aplicando um *cyclic shift* para as partes, obtendo um conjunto de *kernels* assimétricos K^z, para cada local da imagem em que há uma previsão de objeto a ser rastreado, de forma que K^z é dado por $k = (C^{i-1}z, C^{i-1}x)$, que é circular, possuindo as mesmas propriedades para a matriz circular da subseção 2.5.2 *kernels* citados na subseção 2.5.4.

Similar a subseção 2.5.4 é possível definir o *kernel* pela primeira linha da matriz circular, obtendo $K^z = C_x(k^{x,z}$, em que $k^{x,z}$ é a correlação de *Kernel*, como definido anteriormente na Eq. 2.30. Pela Eq. 2.27 é possível calcular a função de regressão para os objetos candidatos como:

$$f(z) = (K^z)^T \alpha, \tag{2.31}$$

em que $f(z)$ é um vetor que possui todos os valores da operação de *cyclic shift*, a resposta final para todos os objetos que serão rastreados. Para calcular de forma eficiente é realizada a diagonalização, obtendo a Eq. 2.32.

$$\hat{f}(z) = \hat{k}^{xz} \odot \hat{\alpha}, \tag{2.32}$$

em que $\hat{f}(z)$ é o vetor que contém todos os *Cyclic Shifts* de z e $\hat{k}^{xz}$ é a correlação de *kernel* para x e z. Essa operação pode ser percebida como uma filtragem espacial sobre os valores de *kernel* k^{xz}. Todos os $f(z)$ são combinações lineares dos valores vizinhos do *kernel* k^{xz} ponderados pelos coeficientes aprendidos α. Como se trata de uma operação de filtragem é melhor formulado no domínio de *Fourier*.

2.5.6 Correlação rápida de *Kernel*

Um tipo específico de *Kernel* é o Função de Base Radial (RBF, do inglês *Radial Basis Function*) gaussiano, que tem o formato $k(x, x') = h(\|x - x'\|^2)$, para alguma função h, logo os elementos $k^{xx'}$ são definidos como:

$$k_i^{xx'} = k(x', C^{i-1}x) = h(\|x' - C^{i-1}x\|^2), \tag{2.33}$$

que é um caso especial de *kernel* de produto ponto a ponto, o que pode ser observado expandindo a norma euclidiana levando a Eq. 2.34,

$$k_i^{xx'} = h(\|x\|^2 + \|x'\|^2 - 2x'^T C^{i-1}x). \tag{2.34}$$

Para um caso especial para um *kernel* gaussiano definido por $k^{xx'} = exp(-1/\sigma^2\|x - x'\|^2)$ é possível obter a Eq. 2.35 de (Henriques et al., 2015),

$$k^{xx} = exp(-1/\sigma^2(\|x\|)^2 + (\|x'\|)^2 - 2\mathcal{F}^{-1}(\hat{x}^* \odot \hat{x}'). \tag{2.35}$$

Uma característica obtida pelo trabalho em um espaço bidimensional é o fato de atuar em vários canais simplesmente somando-os no domínio de *Fourier* essa característica se estende para o caso de um *Kernel* linear $k(x, x') = x^T x'$ obtendo a Eq. 2.36.

$$k^{xx'} = \mathcal{F}^{-1}(\sum_c \hat{x_c}^* \odot \hat{x_c}'), \tag{2.36}$$

em que c é o número de canais. Dessa forma é possível substituir a Eq. 2.36 em Eq. 2.29 e Eq. 2.32 de forma a obter o resultado do rastreador de correlação linear Eq. 2.25 obtendo a solução para o filtro KCF.

2.6 Fundamentação teórica do algoritmo de rastreamento MOSSE

Assim como o algoritmo de rastreamento KCF o algoritmo MOSSE, também é implementado no domínio de Fourier, de forma que as operações que seriam convoluções, no domínio de Fourier são ponto a ponto (Bolme et al., 2010). Em que $F = \mathcal{F}(f)$ e o filtro $H = \mathcal{F}(h)$. E a multiplicação ponto a ponto no espaço de Fourier dos filtros $G = F \odot H^*$.

Para encontrar o filtro que mapeia as entradas de treinamento para as saídas de treinamento desejadas, MOSSE calcula o filtro H que minimiza o somatório do erro quadrático entre a atual saída da convolução e a saída desejada da convolução. Esse problema de minimização é definido pela Eq. 2.37

$$\min_H^* \sum_i ||F_i \odot H^* - G_i||^2. \tag{2.37}$$

Para resolver o problema de otimização é necessário considerar que a função a ser otimizada é obtida por variáveis complexas. Cada elemento de H é indexado por w e v, e pode ser calculado de forma independente por conta das operações no domínio de Fourier serem ponto a ponto. Re-escrevendo as funções em H_{wv}, H^*_{wv}. E enquanto tratando as variáveis independentes $H^*_{wv} = 0$ logo

$$0 = \frac{\partial}{\partial H^*_{wv}} \sum_i |F_{iwv} H^*_{wv} - G_{iwv}|^2. \tag{2.38}$$

Calculando a norma euclidiana da equação 2.38 é obtida a Eq. 2.39

$$H = \frac{\sum_i F_i \odot G_i^*}{\sum_i F_i \odot F_i}, \tag{2.39}$$

em que H é o filtro que resolve o problema de otimização apresentado na Eq. 2.37.

2.7 Sensores do quadrirrotor

Para realizar o controle do quadrirrotor é necessário primeiramente efetuar a medição e/ou estimação do vetor de estados do sistema, para que possa ser então aplicada uma técnica de controle. São utilizados sensores de medição inercial para realizar esta operação, são esses o giroscópio, o acelerômetro e o magnetômetro Fig. (11), conhecidos como *Inertial Measurement Unit* (IMU).

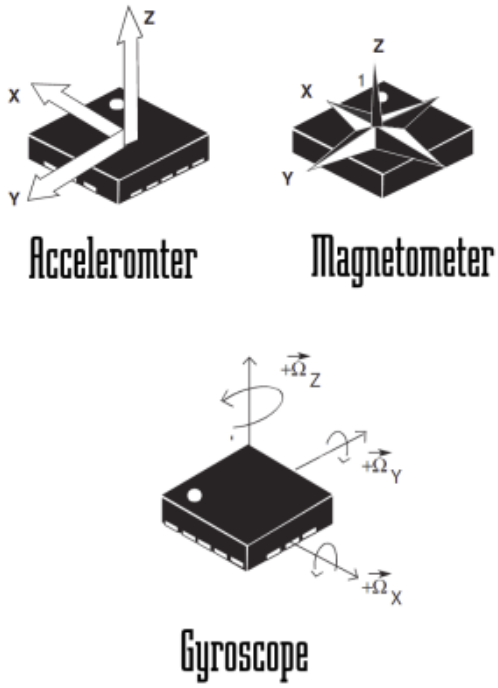


Figura 18 – Principais sensores IMU (OZZMAKER, 2019).

O giroscópio (ou girômetro) é um sensor de extrema importância para que todo o sistema de controle funcione de uma forma adequada, já que as medidas de velocidade angular são realizadas por meio deste sensor e dessas velocidades é possível estimar as acelerações angulares (DOLCI, 2014).

Por conta de necessitar dos dados de arfagem, guinada e rolagem, obtendo os dados nos três eixos, um IMU para um VANT utiliza três giroscópios, um para cada eixo como mostrado na Fig. (11).

Assim como o giroscópio, o acelerômetro é de fundamental importância para a estimação dos estados do quadrirrotor, pois este detecta a aceleração linear sobre o eixo no qual foi inserido. Dessa forma são necessários 3 acelerômetros para estimar as acelerações sobre os eixos x, y e z.

O magnetômetro funciona por meio da medição do campo magnético da terra, obtendo uma referência das orientações norte, sul, leste e oeste. Por conta disso a IMU utiliza essas orientações para auxiliar na estimação do ângulo de *yaw*.

As medidas desses sensores de forma isolada não são suficiente para gerar o vetor de estados do veículo. O giroscópio não é capaz de realizar a medição dos ângulos diretamente, mas a velocidade nestes ângulos, o que leva à propagação do erro no cálculo dos estados atuais.

O acelerômetro por sua vez não é capaz de garantir estabilidade quando as acelerações são elevadas, além deste sensor não ser capaz de realizar a medição da aceleração em *yaw*. Por fim o magnetômetro pode sofrer de diferentes tipos de interferência magnética (PATRãO, 2015).

Outros tipos de erro dos sensores são os de integração, por conta da propagação da incerteza da constante de integração que varia, os de derivação (posição e velocidade) por conta de um ruído de banda larga, que estão descritos em (IV; WALL; BEVLY, 2005).

Outro tipo de sensor é o sensor de posicionamento global (GPS), utilizado para vôos autônomos, sendo muito utilizado para vôos *outdoor* (ambiente aberto). Uma das principais utilidades deste sensor é para controle em *softwares* de navegação, em que pode-se enviar a posição correta (*waypoints*) para que o VANT possa seguir. Isso é realizado pela comparação entre a posição obtida para o veículo e a posição de comando. Um problema para esse sensor são ambientes *indoor* (ambiente fechado) por conta da medição ser bloqueada pelas paredes do ambiente.

3 Metodologia

Esta seção apresenta a metodologia que foi utilizada para o desenvolvimento desse trabalho, detalhando cada atividade finalizada e as que ainda serão realizadas. Além disso foram definidas as técnicas, dentre as avaliadas, que serão utilizadas para o trabalho futuro, assim como alguns resultados preliminares. Para isso foram definidas as seguintes etapas.

3.1 Etapa 1 - Revisão bibliográfica

A primeira etapa do trabalho foi realizar uma ampla revisão bibliográfica, analisando cada técnica já desenvolvida para a execução das atividades propostas, sendo esta a atividade mais longa do projeto.

Foi necessário uma introdução sobre o quadrirrotor e sobre o sensor *Kinect* e suas distorções. Uma revisão de técnicas de Inteligência Artificial, aprendizado de máquina, aprendizagem profunda e rastreamento foram realizadas com o intuito de compreender os conceitos, permitindo a aplicação em quadrirrotores com métodos que estão sendo aplicados na atualidade.

3.2 Etapa 2 - Definição do problema

Por conta de possuir seis graus de liberdade, três de rotação e três de translação, e quatro entradas (rotação dos motores) o quadrirrotor é caracterizado como um sistema sub-atuado. Os graus de liberdade de rotação e translação são acoplados, tornando a natureza dinâmica do sistema não linear, sendo este um problema interessante para a teoria de controle (TIWARI, 2017)

Diferentes tipos de controladores são aplicados para resolver esses problemas, dois bem conhecidos na literatura são o PID e o LQR. Esses controladores se baseiam na fusão dos dados dos sensores IMU (Zul Azfar; Hazry, 2011), realizando operações de derivação e integração que podem levar a uma tomada de decisão errônea pelo controlador do VANT. Além de que possíveis erros no levantamento dos coeficientes físicos do sistema geram um erro no *setpoint* passado pelo controlador (WILSON; GöKTOğAN; SUKKARIEH, 2015).

Uma forma de reduzir esses problemas é utilizar técnicas de Inteligência Artificial aplicadas ao monitoramento de posição, para realizar a elaboração dessas técnicas é necessário o conhecimento de visão computacional e a correta adequação dos algoritmos a limitações de tempo, o que é um desafio, já que esses algoritmos muitas vezes demandam

grande poder computacional. Além da dificuldade de garantir a medição da posição em um espaço tridimensional por meio das câmeras do sensor *Kinect*.

Esses requisitos podem ser atendidos por meio da aquisição das imagens do sensor *Kinect*, que permite uma obtenção tridimensional da posição do VANT. Para realizar a análise em tempo real são combinadas as técnicas de detecção, que precisam de um poder computacional maior do que as técnicas de rastreamento, que não necessitam desse poder computacional, dessa forma obtendo as posições nas três dimensões x, y, z. Podendo, com isso, avaliar o *setpoint* dado pelo controlador embarcado.

3.3 Etapa 3 - Estudo de casos

Os estudos de caso apresentaram a implementação da detecção por meio do algoritmo de aprendizagem profunda YOLOv3 e para o rastreamento o método do KCF. Utilizando esses métodos é possível realizar a estimação da posição do quadrirrotor, conhecendo os parâmetros da câmera utilizada.

Dessa forma para a obtenção das posições será utilizada primeiramente a aplicação da YOLOv3 pelos resultados obtidos há uma velocidade de $30 fps$, que é a velocidade de trabalho do sensor *Kinect* (Redmon; Farhadi, 2017), explicando como será feita a implementação do algoritmo na subseção 3.3.5, que realiza a criação das caixas delimitadoras que serão utilizadas no algoritmo de rastreamento.

Em seguida será implementada tanto a técnica do KCF, escolhida dentre os algoritmos de rastreamento, por conta de obter os melhores resultados de precisão e acurácia (Henriques et al., 2015), quanto a técnica de rastreamento MOSSE, já que possui os melhores resultados para velocidade (Bolme et al., 2010).

Os passos da implementação foram explicados na subseção 3.3.1 e na seção 3.3.2. Para finalmente obter as posições x, y e z do centro de massa do quadrirrotor, como descrito na seção 2. O diagrama da abordagem apresentada pode ser visualizado na Fig. (19).

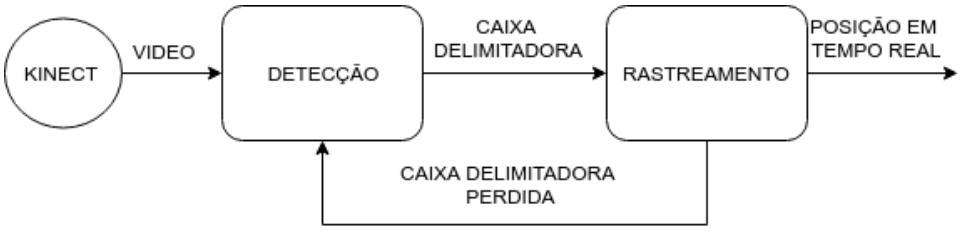


Figura 19 – Diagrama da abordagem que será implementada.

Ambos métodos serão implementados na linguagem de programação python, utilizando a biblioteca OPENCV (BRADSKI, 2000), que já possui uma ampla implementação

de algoritmos de detecção e rastreamento. Para aplicar essa biblioteca no presente trabalho de conclusão do curso, será realizada a edição, quando necessário da biblioteca, respeitando os requisitos estabelecidos. Para a implementação desses será utilizado o editor de texto *Visual Studio Code* da empresa *Microsoft*.

3.3.1 Caso 1 - *Kernelized Correlation Filters*

Tendo em vista que o processo de detecção realiza a identificação do objeto analisado e por conta disso precisa de um maior poder computacional, é necessário aplicar o segundo estudo de casos desse trabalho, realizando o processo de rastreamento do quadrirrotor em tempo integral, não se preocupando com a identificação do objeto, mas sim com a variação dos pixels ao redor da caixa delimitadora.

Para realizar o rastreamento do quadrirrotor pelo algoritmo do KCF foram definidas duas etapas, a primeira etapa é aplicar o rastreamento do quadrirrotor com a criação manual da caixa delimitadora, pegando o primeiro *frame* de vídeos pré gravados, selecionando o quadrirrotor que será rastreado, por meio de uma caixa delimitadora e em seguida aplicando o rastreamento KCF, obtendo o rastreamento do quadrirrotor para o vídeo analisado. Todos os passos podem ser observados na Fig. (25).

Figura 20 – Diagrama do rastreamento realizado.

A segunda etapa é aplicar o algoritmo de rastreamento para um vídeo em tempo real, filmado pelo sensor *Kinect*, obtendo a posição tridimensional do quadrirrotor. Com essa etapa finalizada, será realizada a integração do sistema com o algoritmo da subseção 3.3.5 o tornando um sistema automatizado Fig. (19).

3.3.2 Caso 2 - *Minimum Output Sum of Squared Error*

Tendo em vista que o processo de detecção realiza a identificação do objeto analisado e por conta disso precisa de um maior poder computacional, é necessário aplicar o segundo estudo de casos desse trabalho, realizando o processo de rastreamento do quadrirrotor em tempo integral, não se preocupando com a identificação do objeto, mas sim com a variação dos pixels ao redor da caixa delimitadora.

Para realizar o rastreamento do quadrirrotor pelo algoritmo do MOSSE foram definidas duas etapas, a primeira etapa é aplicar o rastreamento do quadrirrotor com a criação manual da caixa delimitadora, pegando o primeiro *frame* de vídeos pré gravados, selecionando o quadrirrotor que será rastreado, por meio de uma caixa delimitadora e em seguida aplicando o rastreamento MOSSE, obtendo o rastreamento do quadrirrotor para o vídeo analisado. Todos os passos podem ser observados na Fig. (21).

Figura 21 – Diagrama do rastreamento MOSSE realizado.

A segunda etapa é aplicar o algoritmo de rastreamento para um vídeo em tempo real, filmado pelo sensor *Kinect*, obtendo a posição tridimensional do quadrirrotor. Com essa etapa finalizada, será realizada a integração do sistema com o algoritmo da subseção 3.3.5 o tornando um sistema automatizado Fig. (19).

3.3.3 Caso 3 - Sensor *Kinect*

Para realizar a extração da posição de forma correta é necessário estabelecer o posicionamento do sensor *Kinect* e o sistema de coordenadas da extração da posição. Para isso foi realizado uma representação do sistema e pode ser observada na Fig. 22.

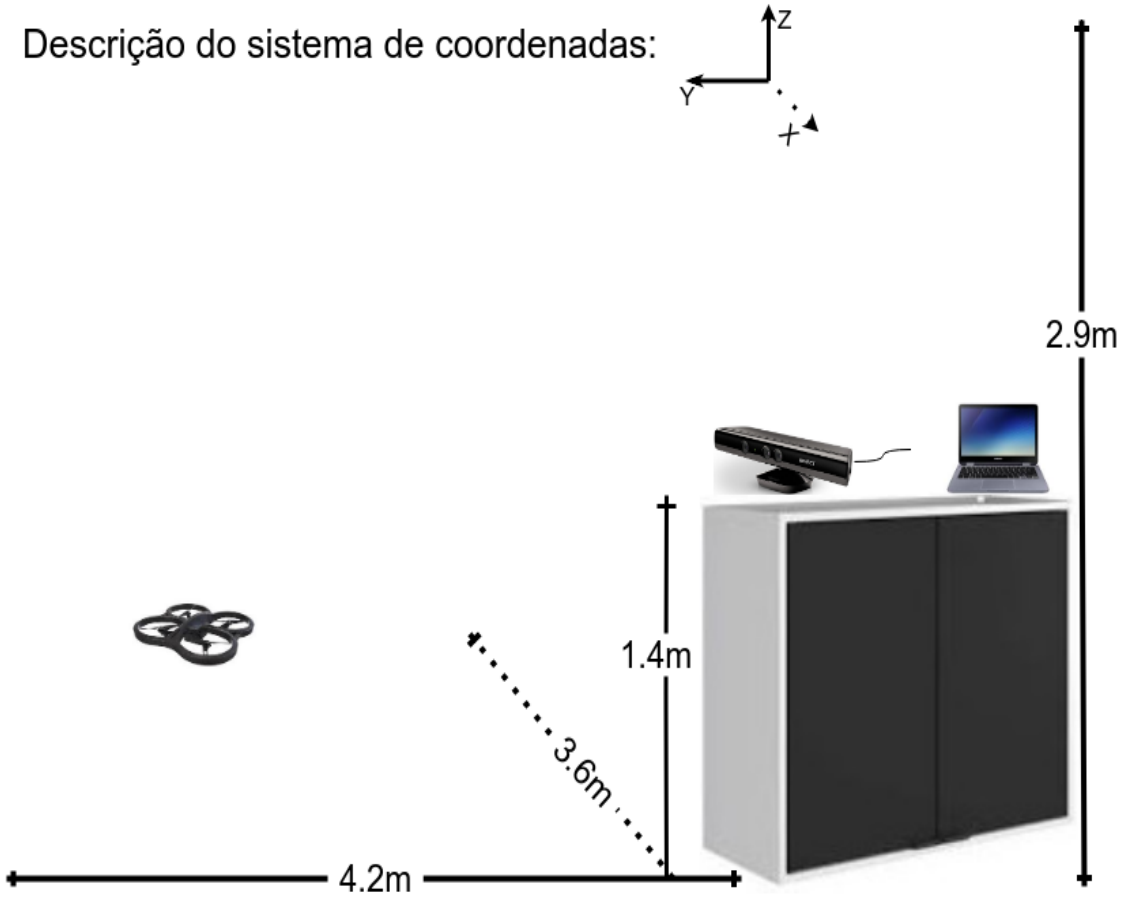


Figura 22 – Representação do posicionamento do sistema de aquisição da posição, utilizando o sensor *Kinect*, além de apresentar o sistema de coordenadas da posição tridimensional.

É importante ressaltar que no estágio de detecção a posição do *Kinect* pode se localizar em qualquer posição, desde que não haja a obstrução da câmera, porém seria necessário a correção da angulação para o cálculo da posição. Por conta disso o cálculo da posição no presente trabalho considera uma posição horizontal do sensor *Kinect*.

Além disso é preciso realizar a calibração da câmera do sensor como descrito na seção 2.1 para que o posicionamento esteja ajustado, sem as distorções da câmera do

sensor.

3.3.4 Caso 4 - Construção do *dataset*

Para realizar a detecção do quadrirrotor é necessário a construção de um *dataset*. O *dataset* deve ser composto por imagens adquiridas pelo sensor *Kinect* do quadrirrotor de estudo deste trabalho (Parrot Ar. Drone 2.0).

Visando extrair as imagens necessárias para a detecção do Parrot Ar. Drone 2.0 duas etapas são necessárias:

1. Vídeo quadrirrotor: Inicialmente será realizado diversos vídeos do quadrirrotor, de forma a obter o máximo de imagens possíveis, obtendo imagens em diversos ângulos e posições do Parrot Ar. Drone 2.0.
2. Subamostragem: Em seguida será implementada a subamostragem dos frames do vídeo, obtendo uma imagem por segundo do quadrirrotor, para constituição do *dataset*.

Por meio dessa subamostragem será colocada a caixa delimitadora manualmente em cada uma das imagens do *dataset*. Para armazenar a posição em que o quadrirrotor se encontra, e posteriormente enviar para a etapa de *Transfer Learning* é necessário criar um arquivo com as posições do centro, da largura e da altura da caixa delimitadora, realizando uma normalização pela largura e altura da imagem colorida que será submetida no algoritmo de treinamento, esses valores normalizados podem ser observados na Eq. 3.1.

$$centroX = \frac{X_{bb}}{L}; \qquad centroY = \frac{Y_{bb}}{A}; \qquad L_n = \frac{L_{bb}}{L}; \qquad A_{bb} = \frac{A_{bb}}{A}; \tag{3.1}$$

em que $centroX$ e $centroY$ são os centros da caixa delimitadora na coordenada x e y normalizados respectivamente; X_{bb} e Y_{bb} são os centros da caixa delimitadora na coordenada x e y respectivamente; A_{bb} e L_{bb} são a altura e a largura da caixa delimitadora normalizada respectivamente e L e A são a largura e altura da imagem colorida.

3.3.5 Caso 5 - *You Only Look Once*

Para realizar a detecção do quadrirrotor, específico do estudo desse trabalho, será levantado um *dataset* utilizando as câmeras do sensor *kinect*, este contendo as imagens das câmeras no espaço de cor RGB, como explicado em 3.3.4.

Com esse conjunto de dados levantados será realizado o treinamento da YOLOv3 construída, realizando as modificações necessárias em algum arquivo de pesos já treinados da YOLOv3, implementando a técnica de *Transfer Learning*.

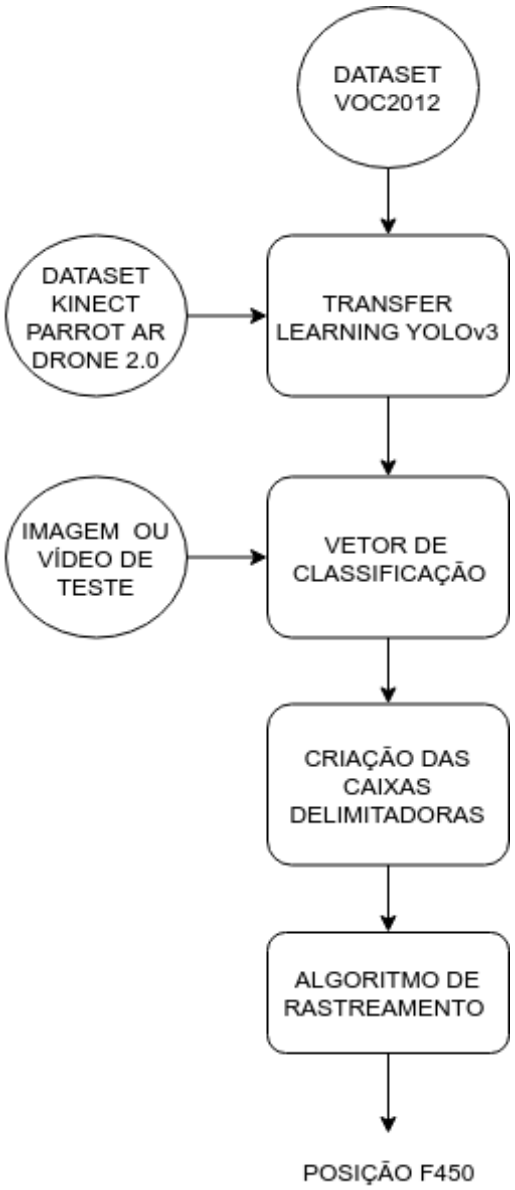


Figura 23 – Passos para realizar a detecção do quadrirrotor em uma imagem, tanto para vídeos pré gravados, quanto para vídeos adquiridos em tempo real.

Conhecendo os parâmetros da câmera utilizada, pelo processo de calibração realizado, é possível aplicar a Eq. 2.17 obtendo as posições tridimensionais de um vetor de posições próximos ao centro de de massa da caixa delimitadora, estimado pela Rede Neural.

Com o funcionamento do sistema será aplicado inicialmente a vídeos pré gravados do quadrirrotor, validando o funcionamento do sistema, para que em seguida seja realizada a análise em tempo real por meio da integração com o algoritmo de rastreamento. Todos os passos para o levantamento da rede neural de convolução são descritos na Fig. (23).

3.4 Etapa 6 - Integração do Sistema

Posterior a validação de ambas as etapas (rastreamento e detecção) é construído um sistema integrado entre a etapa de rastreamento e a etapa de detecção. O sistema integrado será implementado no intuito de utilizar tanto a velocidade de rastreamento dos algoritmos apresentados, quanto a autonomia na detecção da YOLOv3.

Para o rastreamento serão implementados ambos(KCF e MOSSE) algoritmos e verificado visualmente qual dos algoritmos possui o melhor resultado na detecção do quadrirrotor, sem que este pare de rastrear o objeto em condições adversas. O diagrama da integração pode ser visualizado tanto na Fig. 19 quanto na Fig .24

Figura 24 – Passos para realizar a integração do sistema de detecção e rastreamento do quadrirrotor, em um vídeo.

A validação do sistema será realizada com marcações em posições conhecidas do

quadrirrotor em um ambiente controlado, de forma que cada uma das marcações se encontram a um metro de distância da posição inicial. Logo a posição do quadrirrotor será estimada em relação à posição inicial do mesmo como apresentado na Eq .3.2

$$x_{estimado} = x_{atual} - x_{inicial}; \quad y_{estimado} = y_{atual} - y_{inicial}; \quad z_{estimado} = z_{atual} - z_{inicial}, \quad (3.2)$$

em que $x_{estimado}, y_{estimado}$ e $z_{estimado}$ seguem o sistema de coordenadas apresentado na Fig .**??**.

4 Resultados

Esta seção apresenta os resultados que foram obtidos no decorrer deste trabalho, para a etapa de rastreamento, para a calibração do sensor *Kinect*, para a detecção do quadrirrotor na imagem e para o processo de equalização de histograma. Além disso foi realizada uma discussão dos resultados obtidos para cada uma das etapas citadas.

4.1 Rastreador utilizando KCF

Para realizar o rastreamento inicial do quadrirrotor, sem que houvesse um algoritmo de detecção anteriormente implementado, foi necessário selecionar uma caixa delimitadora manualmente, cercando o objeto no primeiro frame do vídeo e selecionando o quadrirrotor, para iniciar o rastreamento do KCF.

É importante ressaltar que nesse ponto o algoritmo ainda não identificou a existência do quadrirrotor, já que este algoritmo de rastreamento não possui RNA associada, logo apenas realiza um estudo comparativo e probabilístico com os pixeis ao redor da caixa delimitadora, para assim realizar o rastreamento. O rastreamento inicialmente foi realizado utilizando a câmera na representação RGB, obtendo as caixas em tempo real. O diagrama do funcionamento pode ser observado na Fig. (25).

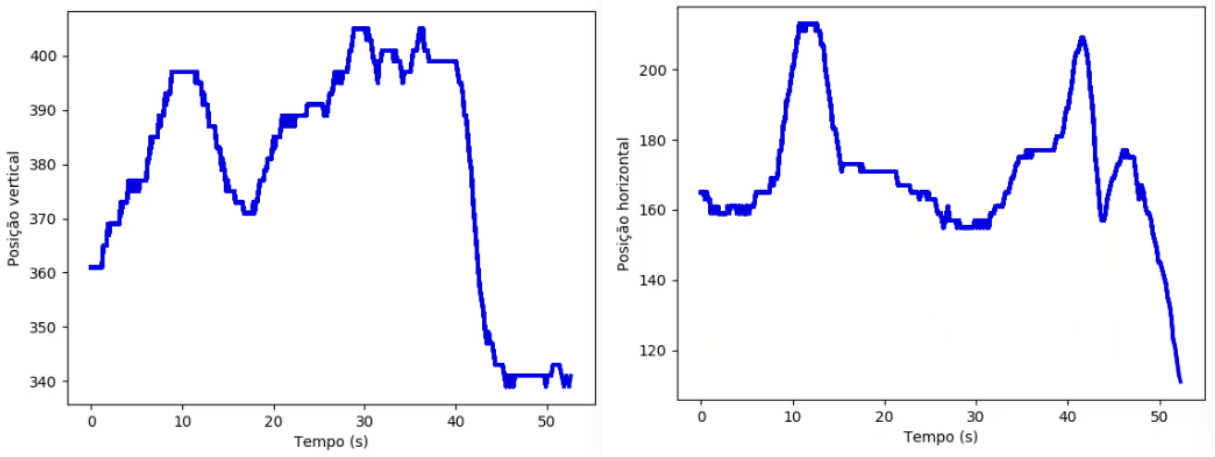
Figura 25 – Resultado obtido aplicando o KCF, com um vídeo pré-gravado.

Aplicado as etapas do diagrama foram obtidos os resultados do rastreamento do

objeto selecionado, além de obter a velocidade de execução do algoritmo em Frames por segundo (Fps), em detrimento do tamanho da caixa delimitadora selecionada. Isso pode ser observado pela Fig. 25, em que inicialmente é colocada uma caixa delimitadora de forma manual, obtendo o conjunto de pixeis a serem rastreados.

Posteriormente é observável o funcionamento do algoritmo KCF, em que um novo frame da imagem é rastreado pela técnica, com uma velocidade de $86Fps$ como pode ser observado em (Tracker. . . , 2019).

Com a utilização desta técnica é possível estimar o centro geométrico da caixa delimitadora, obtendo os valores em pixeis da posição estimada do quadrirrotor. Foi realizado um vídeo (Tracker. . . , 2019), em que as posições verticais e horizontais do centro de massa da caixa delimitadora em que o quadrirrotor se encontra são calculadas. O valor do rastreamento dessas posições pode ser visualizado na Fig. 26a e na Fig. 26b



(a) (b)

Figura 26 – Gráficos da posição vertical Fig. 26a e horizontal Fig. 26b, medidas através do rastreamento do KCF, no vídeo (Tracker. . . , 2019).

Alguns aspectos foram observados ao utilizar o KCF, este rastreador obtêm um bom resultado de precisão e velocidade do algoritmo, quando o fundo da imagem não possui a mesma coloração do objeto rastreado, já que o rastreador KCF não realiza a estimação da posição do quadrirrotor.

Quando ocorre uma aproximação da coloração do quadrirrotor com a coloração do fundo da imagem, o rastreador tem uma falsa acertação, de forma que o algoritmo de rastreamento erra o objeto que está rastreando, e rastreia uma parte fixa do fundo da imagem.

Um outro fator negativo com relação ao rastreamento do KCF é que este filtro não faz um ajuste no tamanho da caixa delimitadora, quando há uma variação de profundidade maior que a atual profundidade da caixa delimitadora.

Dessa forma quando há um deslocamento para o fundo da imagem, o rastreador

pode novamente obter uma falsa acertação, já que a maior parte da imagem dentro da caixa delimitadora está composta pelo fundo da imagem e não pelo quadrirrotor.

Um outro fator importante é que o algoritmo foi implementado inicialmente em vídeos pré gravados. Essa restrição inicial permite que, caso o cálculo da caixa delimitadora fosse terminado antes da aquisição do próximo frame o vídeo seja acelerado. Esta característica não é possível de ser atingida quando o filme está sendo adquirido em *real time*.

4.2 Rastreador utilizando MOSSE

Assim como no caso do rastreador KCF, o rastreador MOSSE foi implementado, sem que houvesse um algoritmo de detecção anteriormente atuando. Dessa forma foi selecionada uma caixa delimitadora manualmente, cercando o objeto no primeiro frame do vídeo, iniciando o rastreador MOSSE.

Aplicado as etapas do diagrama foram obtidos os resultados do rastreamento do objeto selecionado, além de obter a velocidade de execução do algoritmo em Frames por segundo (Fps), em detrimento do tamanho da caixa delimitadora selecionada.

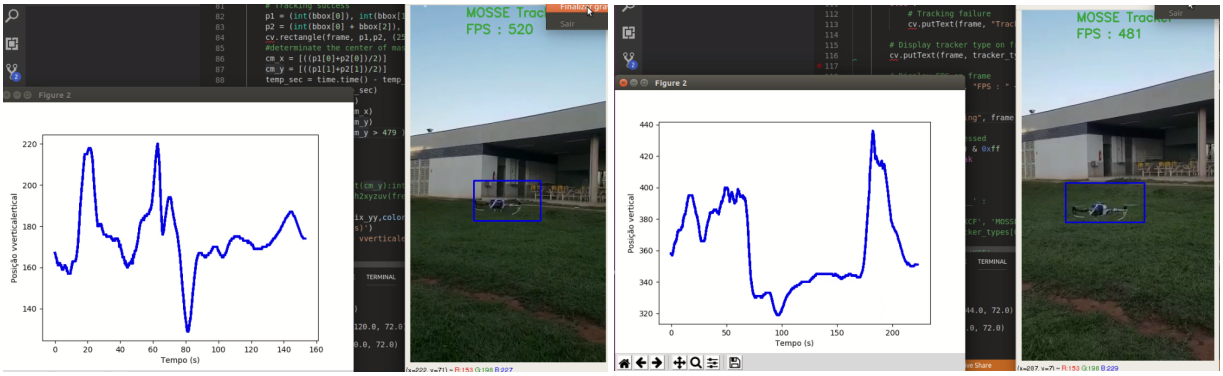
Os valores de Fps podem ser observados na Fig. 27a e na Fig. 27b, em que inicialmente é colocada uma caixa delimitadora de forma manual, obtendo o conjunto de pixeis a serem rastreados.

Posteriormente é observável o funcionamento do algoritmo KCF, em que um novo frame da imagem é rastreado pela técnica, com uma velocidade por volta de $500Fps$ como pode ser observado em (MOSSE. . . , 2019) e em (MOSSE. . . , 2019).

Com a utilização desta técnica é possível estimar o centro geométrico da caixa delimitadora, obtendo os valores em pixeis da posição estimada do quadrirrotor. Foi realizado um vídeo (Tracker. . . , 2019), em que as posições verticais e horizontais do centro de massa da caixa delimitadora em que o quadrirrotor se encontra são calculadas. O valor do rastreamento dessas posições pode ser visualizado na Fig. 27a e na Fig. 27b

Alguns aspectos foram observados ao utilizar o MOSSE, este rastreador obtêm um bom resultado em velocidade. Por conta da velocidade de execução do algoritmo MOSSE o rastreamento é feito de forma mais efetiva do que o algoritmo KCF, de forma que o MOSSE tem uma menor taxa de falsa aceitação além de rastrear o objeto em movimentos com uma maior velocidade.

Um outro fator negativo com relação ao rastreamento do MOSSE é que este filtro também não faz um ajuste no tamanho da caixa delimitadora, quando há uma variação de profundidade maior que a atual profundidade da caixa delimitadora, dessa forma quando há um deslocamento para o fundo da imagem, o rastreador pode novamente obter



(a) (b)

Figura 27 – Gráficos da posição vertical Fig. 27a e horizontal Fig. 27b, medidas através do rastreamento do KCF, no vídeo (MOSSE. . . , 2019) e no vídeo (MOSSE. . . , 2019).

uma falsa acertação, já que a maior parde da imagem dentro da caixa delimitadora está composta pelo fundo da imagem e não pelo quadrirrotor.

Um outro fator importante é que o algoritmo foi implementado inicialmente em vídeos pré gravados. Essa restrição inicial permite que, caso o cálculo da caixa delimitadora fosse terminado antes da aquisição do próximo frame o vídeo seja acelerado. Esta característica não é possível de ser atingida quando o filme esta sendo adquirido em *real time*.

4.3 Calibração do sensor *Kinect*

Visando extrair a posição tridimensional de forma a obter um resultado com alta acurácia e precisão é realizada a calibração do sensor *Kinect*, para isso foi impresso um imagem de um tabuleiro de xadrez de tamanho $7X9$ (7 cruzamentos entre o quadrados com coloração totalmente preta e totalmente branca na horizontal e 9 cruzamentos entre quadrados com coloração totalmente preta e totalmente branca na vertical) e com área de $4cm^2$, como observado na Fig. 28.

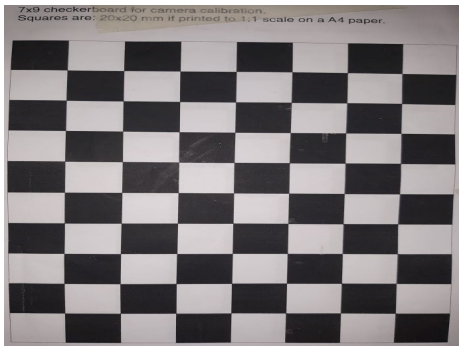
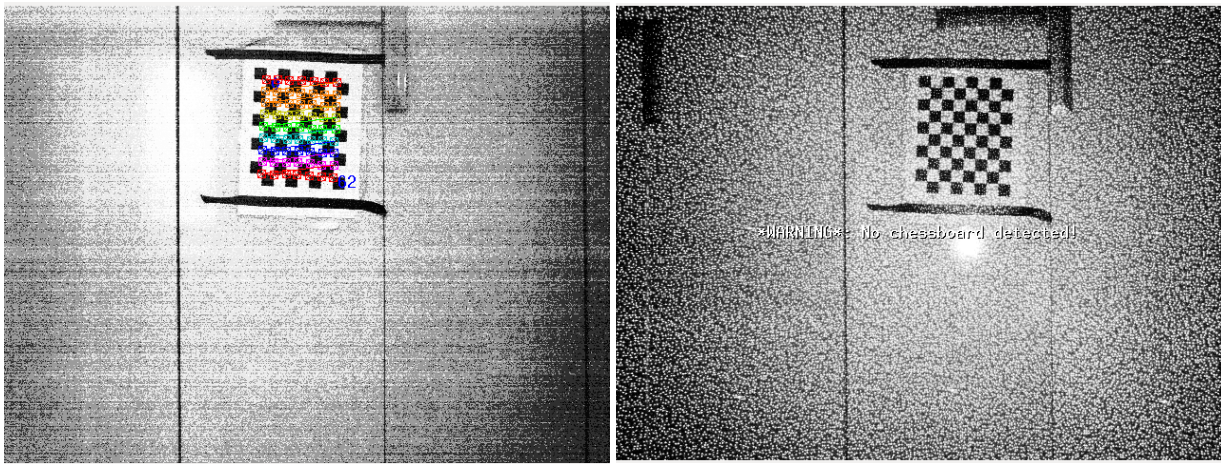


Figura 28 – Tabuleiro de xadrez $7X9$ impresso em uma folha A4, com área dos quadrados de $4cm^2$, utilizado para calibração da câmera.

Para realizar a detecção dos pixeis com coloração preta e coloração branca nas duas câmeras do sensor (câmera colorida e câmera de profundidade) foi encontrado um problema inicial, em que a câmera colorida realizava a detecção da variação da intensidade destes pixeis, enquanto que a câmera de profundidade não conseguia realizar esta detecção.

Essa falha na detecção ocorre por conta de que a câmera do sensor *Kinect* é composta de um projetor IR e um receptor IR, responsáveis por projetar e receber a reflexão desses raios, obtendo os valores de profundidade. Porém a projeção IR no momento da detecção gera diversos ruídos na recepção dos raios IR, impossibilitando a detecção dos quadrados de coloração diferente.

Para evitar com que esse ruído impedissem a identificação da variação dos pixeis na imagem é necessário bloquear o projetor IR fisicamente, como pode ser observado na Fig. 30. Ao bloquear a fonte do sensor *kinect* é necessário fornecer uma fonte de raios IR para iluminar o ambiente e logo ser recebido no receptor do *Kinect*. É possível perceber a diferencia das imagens do sensor com o projetor IR bloqueado Fig. 29a para o sensor com o projetor IR desbloqueado na Fig. 29b.



(a) (b)

Figura 29 – Detecção da variação dos padrões no tabuleiro de xadrez na Fig. 29a. E a falha na detecção por conta da projeção dos raios IR na Fig. 29b, em que o projetor IR não estava bloqueado.

Figura 30 – Imagem do sensor *kinect* com o projetor infravermelho coberto para não haver interferência no receptor infravermelho.

Foi colocado o tabuleiro em várias posições na imagem adquirida por ambas câmeras (receptor IR e câmera colorida), armazenando a detecção da transição dos quadrados brancos para os pretos em cada uma dessas posições, como pode ser observado na Fig. 31.

(a) (b)

Figura 31 – Detecção da variação da coloração dos quadrados no tabuleiro de xadrez pela câmera colorida do sensor *Kinect* na Fig. 31a e na Fig. 31b.

Com a identificação dos padrões no tabuleiro é possível obter a matriz de parâmetros intrínsecos do sensor *Kinect*. Após a aquisição de 15 imagens do tabuleiro de xadrez são obtidos os dados da distorção da câmera e os dados intrínsecos são apresentados na Tab .1.

Tabela 1 – Tabela de parâmetros intrínsecos obtidos para o sensor *Kinect*.

co_x	320.060293 ± 0.013	λ_2	$6.516358e^{-01}$
co_y	235.091900 ± 0.013	λ_3	$5.126044e^{-04}$
df_x	570.067270 ± 0.063	p_1	$5.773943e^{-02}$
df_y	446.851786 ± 0.072	p_2	$4.502992e^{-01}$
λ_1		$-5.149702e^{-01}$	

em que df_x e df_y é a distância focal da lente, co_x e co_y é o centro óptico, λ_1, λ_2 e λ_3 são os parâmetros da distorção radial e p_1 e p_2 são os parâmetros da distorção tangencial esses parâmetros permitem obter a matriz de valores intrínsecos que é apresentada na Eq.4.1.

$$matriz\_camera = \begin{pmatrix} 570.067270 & 0 & 320.060293 \\ 0 & 446.851786 & 235.091900 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.1}$$

4.4 Detecção do quadrirrotor

Para realizar a detecção do quadrirrotor foi construído um *dataset* por meio de três filmagens do quadrirrotor, utilizando a câmera colorida do sensor *Kinect*. Garantindo que o Parrot Ar. Drone 2.0 estivesse em diversas posições dentro da imagem.

Aplicando os parâmetros que descrevem a localização da caixa delimitadora descritos na seção 3.3.4 e as 792 imagens do *dataset* e é possível realizar o *Transfer Learning* com a utilização de uma estrutura de software livre para treinamento de *Redes Neurais Artificiais* que pode utilizar tanto GPUs quanto CPUs para realizar o treinamento a Darknet.

Na estrutura da Darknet as redes neurais são implementadas para o treinamento, de forma a promover uma completa estrutura para aplicar tanto o treinamento da rede, quanto o treinamento por *Transfer Learning*.

Para realizar o *Transfer Learning* verificou-se a necessidade de configurar o GPU do sistema, já que o tempo para cada um do conjunto de imagens analisado foi de aproximadamente $5200s$ quando não havia um GPU associado.

Houveram diversos problemas com a integração da interface gráfica do Ubuntu com a GPU, de forma que foi preferido alugar uma máquina virtual na plataforma Amazon *Elastic Compute Cloud* (EC2), que é um serviço web que disponibiliza diversos sistemas com capacidade computacional, já configurados e seguros para o uso, sendo que alguns desses serviços possuem aplicações especificas para treinamento de Redes Neurais Artificiais.

A máquina escolhida possui as características apresentadas na Tab. 2, além de possuir estas características o modelo da GPU utilizada é NVIDIA K80 de alta performance com 2.496 núcleos de processamento paralelo, processadores Intel Xeon E5-2686 v4. Com a utilização da GPU no treinamento da Rede Neural Artificial o tempo de processamento para cada conjunto de imagem é de em média $27.22s$, tornando a velocidade de operação em 173.33 vezes mais rápida.

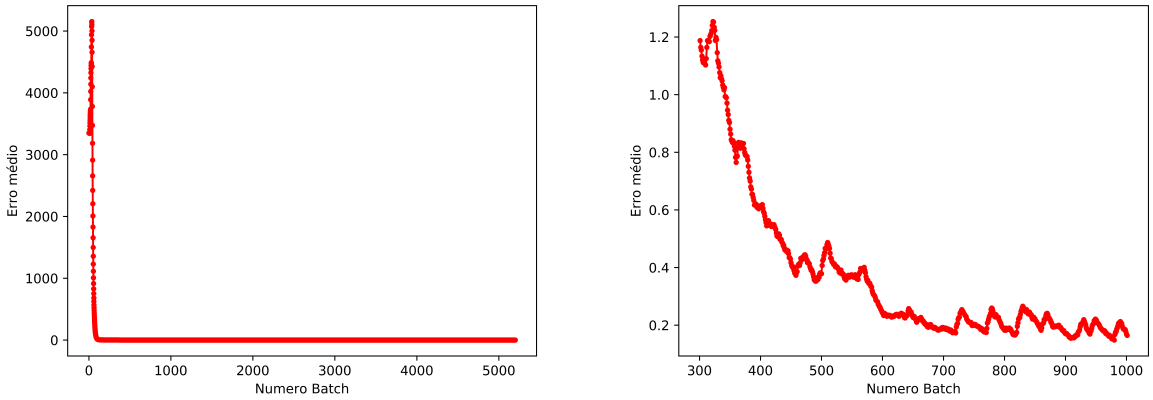
Tabela 2 – Parâmetros da máquina na nuvem escolhida para realizar o *Transfer Learning*

vCPU	Mem (GiB)	GPU	Memória da GPU (GiB)	Instância
4	61	1	12	p2.xlarge

Para acessar a máquina no Servidor Web da Amazon (AWS, do inglês *Amazon Web Service*) é realizada uma comunicação Secure Shell (SSH), que permite um acesso seguro a um serviço de rede utilizando um terminal, dentro da máquina basta executar a Darknet (REDMON, 2013–2016) realizando as configurações de todas as camadas da YOLOv3.

Um outro fator importante para implementar o *Transfer Learning* é a utilização de pesos pré treinados de forma a utilizar os benefícios de um *dataset* treinado por semanas para uma aplicação específica com a utilização de um *dataset* menor contendo o novo objeto de estudo. Para o presente trabalho foi utilizado o *dataset* Visual Object Classes Challenge 2012 (VOC2012) (EVERINGHAM et al., 2019), que contêm 20 classes 11,530 imagens e 27,450 caixas delimitadoras.

Durante o processo do treinamento é possível verificar a média do erro entre a resposta real dos pesos treinados e a resposta predita pela YOLOv3. Essa verificação é realizada para que, quando a média desse erro for menor que um valor limite, o treinamento seja terminado, antes que esse alcance o número máximo de interações programadas. O valor da média do erro, pelo número de batch é apresentado na Fig. 32a e Fig. 32b.



(a) (b)

Figura 32 – Erro médio entre os pesos da ultima camada da YOLOv3 e o valor real dos pesos esperados para um conjunto de imagens específicas, analisadas durante o processo de *Transfer Learning*. Para a Fig. 32a é apresentado o erro médio para o treinamento completo, enquanto que para a Fig. 32b é apresentado para os valores do número de batch no intervalo de 300 a 1000.

É possível perceber na Fig .32a que o valor limite do treinamento foi de 5200, porém na Fig .32b o valor do erro médio é menor que um por volta do tricentésimo número batch. Posterior ao treinamento é obtido os pesos da rede, associado a classe quadrirrotor, acrescentada no *dataset* VOC2012.

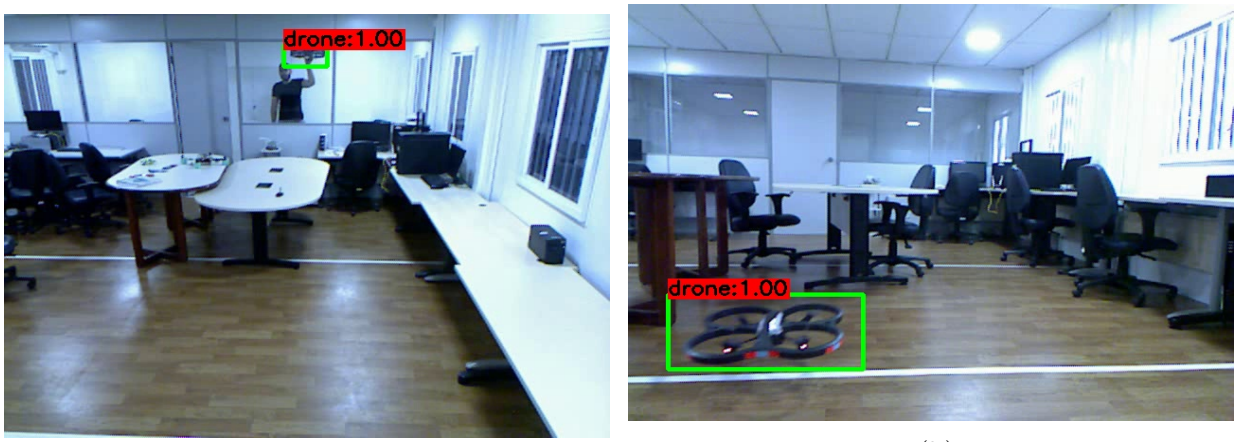
Por meio dos pesos treinados, foi implementado o algoritmo de detecção do quadrirrotor, em que os frames do vídeo advindo do Kinect foram utilizados de forma a detectar o Parrot Ar. Drone 2.0 como pode ser observado em (YOLOV3. . . , 2019).

Algumas imagens da detecção do Parrot Ar. Drone podem ser encontradas na Fig. 33. É importante ressaltar que foi colocado um limiar de detecção elevado para definição da posição do quadrirrotor, de forma que o valor comparativo entre peso real para uma determinada imagem e peso estimado pela rede deve ter uma porcentagem de 99%, para

que a YOLOv3 classifique o objeto como um quadrirrotor.

Este valor foi definido de forma empírica, já que em alguns momentos foi percebida uma falsa acertação do algoritmo de detecção, porém quando o limiar de detecção foi incrementado, o valor da métrica de *mean average precision* (mAP) foi de $89,87\%$. Este valor de mAP foi obtido utilizando as imagens de teste do *dataset*, que são 10% do total do *dataset* (os 90% das imagens restantes foram utilizadas para o *Transfer Learning*).

As imagens do conjunto de teste foram escolhidas aleatoriamente, dentre as imagens totais do *dataset* para que não houvesse uma seleção seletiva das imagens que seriam testadas e validariam o funcionamento da rede, garantindo que haja variação, nas imagens de teste, para a detecção do Parrot Ar. Drone 2.0.



(a) (b)

Figura 33 – Exemplo do funcionamento do algoritmo YOLOv3 para o quadrirrotor Parrot Ar. Drone 2.0, treinado por meio de *Transfer Learning* para o conjunto de imagens de teste do *dataset*.

Apesar dos bons resultados na detecção do quadrirrotor em diversos ambientes e posições, e além disso o algoritmo de detecção ser um dos mais rápidos, como explicado na seção 2, ainda possui um valor lento de execução, demorando em média $2s$ por imagem para a detecção. Para o rastreamento em *real time* é necessário o envio da posição 3D do quadrirrotor a uma taxa de no mínimo $50ms$, o que poderia ser atingido pela utilização de hardware dedicado por exemplo.

Uma outra forma de obter uma resposta com um tempo de execução menor é por meio da integração do sistema conjunto (rastreador e detecção), de forma que a restrição de tempo é respeitada e o algoritmo atua autonomamente.

4.5 Integração do sistema

Por meio dos resultados promissores em ambas etapas, rastreamento e detecção do Parrot Ar. Drone 2.0, foi realizada a integração dos dois sistemas. Para isso foi ob-

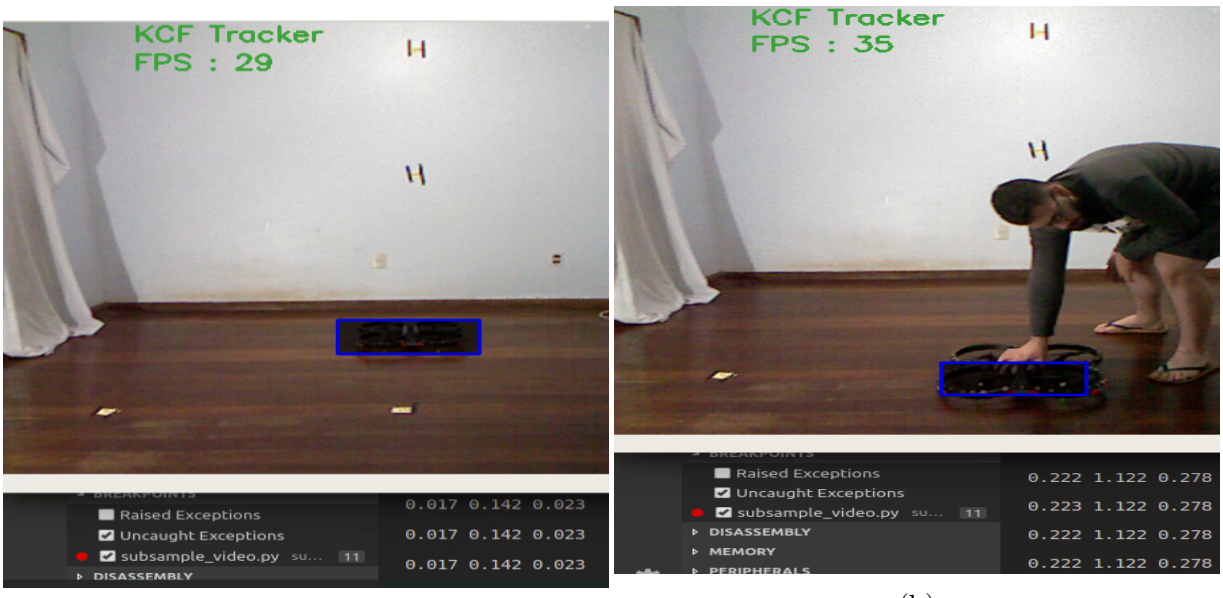
tida a detecção do quadrirrotor de forma automática, por meio da YOLOv3, em que a caixa delimitadora adquirida pela Rede Neural Artificial em um intervalo de $2s$, como apresentado na seção 4.4.

Com isso é possível enviar a caixa delimitadora de forma automática para o algoritmo de rastreamento obtendo em média uma resposta de $38Fps$ ou ainda $26ms$ das posições nos três eixos (x, y, z). Para validar a integração do sistema foram feitas marcações a cada um metro do centro dol Parrot Ar. Drone 2.0, em uma sala de $4.2m$ de comprimento por $3.6m$ de largura e $2.9m$ de altura.

Em seguida foi movimentado o quadrirrotor manualmente pelas medições e verificada as posições extraídas pelo algoritmo integrado, utilizando o modelo de câmera pinhole do sensor *Kinect*. Esses valores podem ser observados no vídeo (YOLOv3. . . , 2019) e no vídeo (Sistema. . . , 2019), em que foi utilizado tanto o algoritmo de rastreamento KCF quanto o algoritmo de rastreamento MOSSE, respectivamente.

Um exemplo da extração da posição pode ser observado na Fig. 34, em que os valores apresentados no terminal são x, y e z respectivamente, em metros. Os valores foram adquiridos por meio da variação da posição inicial do quadrirrotor, para a posição atual do mesmo na imagem.

Assim, quando o quadrirrotor se deslocava no sentido de uma das marcações a variação real deveria ser de $1m$, porém foi observado o erro em média da aquisição de $20cm$, entre a posição esperada e a real.



(a) (b)

Figura 34 – Exemplo da extração da posição em x, y e z respectivamente, com a utilização da câmera do sensor *Kinect*, em que foi gerada a variação na posição y e foi verificada a variação de $1.12m$ para uma variação real de $1m$.

Um problema percebido para a estimação da posição do quadrirrotor é que, em diversos momentos do rastreamento, o quadrirrotor não esta completamente alinhado com a caixa delimitadora, de forma que, a posição caso seja extraída no centro geométrico da caixa delimitadora não seria correspondente com a real.

Além disso, a geometria do quadrirrotor não é completamente uniforme possuindo diversos furos, que permitem, caso haja um desalinhamento quadrirrotor e caixa delimitadora, a aquisição da profundidade no fundo da imagem e não no quadrirrotor.

Visando minimizar estes erros de medição, a estimação da posição é realizada por meio de um método estocástico, em que é retirado um intervalo de valores dentro da caixa delimitadora em volta do centro geométrico da caixa delimitadora. Dentro deste intervalo estima-se que o menor valor extraído da câmera de profundidade seja o ponto em que o quadrirrotor se encontra, dessa forma este valor é considerada a posição do quadrirrotor.

Validada a aquisição das posições para o Parrot Ar. Drone 2.0 foram realizados testes de voo, para os dois tipos de rastreadores. Foi possível perceber que apesar do algoritmo de rastreamento KCF possuir uma boa precisão, quando é atualizada a caixa delimitadora, o algoritmo diversas vezes não consegue acompanhar a velocidade de quadrirrotor e por isso começa a rastrear o fundo da imagem ou simplesmente perde a referência do quadrirrotor.

Para minimizar a variação dentre os pixeis estudados na caixa delimitadora, foi realizada uma atenuação nos mesmos $(0.25 * pixel_{real})$, de forma que a imagem rastreada mantivesse a maior uniformidade possível. Este método auxiliou na melhoria do rastreamento, porém não de forma eficaz, já que, caso haja uma variação brusca nos pixeis da imagem o rastreador KCF não é capaz de rastrear o quadrirrotor.

Um vídeo com o sistema integrado extraindo a posição pode ser observado na em (FLIGHT..., 2019) em que ainda não havia sido feita a atenuação dos pixeis e que a velocidade em *Fps* é calculada apenas para o tempo de execução do algoritmo KCF e não para todas as operações considerando a velocidade de aquisição do sensor *Kinect*.

Utilizando o rastreador MOSSE os resultados foram melhores, já que a velocidade de execução do algoritmo é mais rápida, também foi implementado para o mosse a atenuação nos pixeis de $0.55 * pixel_{real}$, para que os pixeis na caixa delimitadora fossem uniformizados. O rastreamento do algoritmo integrado utilizando o MOSSE pode ser observado em (TEST..., 2019).

5 Conclusão

Este trabalho trouxe a proposta de implementar duas abordagens, uma de detecção e uma de rastreamento, utilizando o sensor *Kinect* para realizar a medição da posição tridimensional do quadrirrotor. Implementar a abordagem de detecção possibilita o entendimento de Redes Neurais Profundas, por meio de CNNs que utilizam filtros convolutivos para a formação dos seus nós. A segunda abordagem possibilita o entendimento de técnicas de rastreamento para a obtenção, constante, da caixa delimitadora em que o quadrirrotor se encontra, viabilizando a estimação da posição 3D em tempo real.

É importante ressaltar que as propostas apresentadas possuem certo nível de complexidade para serem implementadas, porém apresentaram um bom resultado, tanto no rastreamento, quanto na detecção do quadrirrotor avaliado.

Os resultados que foram obtidos neste trabalho foram satisfatórios, considerando a revisão que teve de ser construída durante o projeto. O estudo de Redes Neurais, Aprendizado de Máquina, Aprendizado Profundo e de técnicas de rastreamento foi necessário para entender as abordagens que poderiam ser realizadas, além de verificar diferentes tipos de trabalhos, já realizados na área, para assim definir o escopo deste.

É importante ressaltar que a validação das medições da posição 3D, que foram obtidas pelo sistema, foram realizada inicialmente com medições fixas, na parede ou no piso, no próprio ambiente fechado. Para uma validação com uma confiança maior deveria ser utilizado um sistema de detecção mais robusto, o laboratório do departamento de educação física da UNB possui um sistema de captura de movimentos do tipo VICON, porém houveram reformas no prédio que impossibilitaram a validação do sistema no ambiente.

Para as técnicas de rastreamento utilizadas o algoritmo de rastreamento MOSSE obteve melhores resultados que do que o algoritmo de rastreamento KCF, garantindo que a caixa delimitadora mantivesse o quadrirrotor por mais tempo. O rastreamento pelo MOSSE obteve um maior tempo sem a necessidade de utilizar a re-alimentação do sistema (não necessitou de utilizar a realimentação do algoritmo de Detecção tanto quanto o KCF).

Os resultados obtidos se devem por conta de que o algoritmo KCF é computacionalmente mais exigente, de forma que quando o quadrirrotor faz movimentos bruscos, principalmente quando há uma variação da coloração do fundo da imagem, o algoritmo não consegue acompanhar a velocidade exigida. O que não ocorre com o MOSSE por ser computacionalmente menos exigente.

Para o treinamento do algoritmo de detecção da YOLOv3 os resultados foram

bastante satisfatórios, já que mesmo com o *dataset* com um conjunto de dados extraído em um único ambiente o algoritmo de detecção é capaz de detectar em diversos tipos de ambientes. Esta precisão do algoritmo se deve ao treinamento utilizando o *Transfer Learning*, que permite utilizar os dados de mais de 27000 imagens.

Além disso o algoritmo da YOLOv3 apresentou uma boa velocidade de detecção, como mostrado na seção 4.4, porém para implementar a re-alimentação do sistema é necessário que o quadrirrotor esteja estático, já que o *setpoint* enviado pela RNA tem um atraso de $2s$.

Outro ponto positivo do monitoramento pela câmera do sensor *Kinect* é que o erro associado a medição é de menos de $1m$ enquanto que o erro dos sensores geralmente é associado em metros. Além disso caso seja necessário estimar a posição pela integração do dado do sensor, quando o GPS não está presente, são gerados os erros, já apresentados.

Foi possível manter a restrição de tempo com a utilização das técnicas de rastreamento, apresentando uma nova abordagem para detecção e rastreamento de quadrirrotores, por meio dos resultados são pensados trabalhos futuros de controle, em que deverá ser implementado a posição adquirida há um controlador restritivo, visando acabar com o problema de derivação dos controles atuais.

O sistema apresentando sensor *Kinect*, conectado ao computador, permite uma boa análise da posição do quadrirrotor por meio das imagens adquiridas, porém o sistema utilizando apenas um sensor *Kinect* se limita à uma profundidade de $5m$. Para utilizar o sistema em locais abertos é necessário utilizar uma fusão sensorial composta de mais sensores aumentando a profundidade que pode ser analisada.

Por meio desse sistema é possível fazer uma fusão sensorial com os demais sensores da IMU do quadrirrotor, permitindo que sejam projetados algoritmos de controle mais eficientes que os atuais, por conta da garantia da posição do quadrirrotor.

É importante ressaltar que o sistema deve ser colocado em um ambiente em uma única tonalidade de cor, tonalidade esta diferente da tonalidade do quadrirrotor, de forma que o sistema de rastreamento não seja influenciado diretamente pela fundo em que o quadrirrotor se encontra.

Um outro fator a ser analisado é que futuros *datasets* devem conter mais de um quadrirrotor, gerando uma maior robustez na verificação das imagens pela Rede Neural Artificial.

Referências

Aker, C.; Kalkan, S. Using deep networks for drone detection. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. [S.l.: s.n.], 2017. p. 1–6. Citado na página 45.

Babenko, B.; Yang, M.; Belongie, S. Visual tracking with online multiple instance learning. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 983–990. ISSN 1063-6919. Citado na página 45.

BALTAZAR, G. *CPU vs GPU in Machine Learning*. 2018. Disponível em: <https://www.datascience.com/blog/cpu-gpu-machine-learning>. Citado 3 vezes nas páginas 15, 26 e 27.

Bolme, D. S. et al. Visual object tracking using adaptive correlation filters. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2010. p. 2544–2550. ISSN 1063-6919. Citado 3 vezes nas páginas 45, 54 e 58.

BOUDJIT, K.; LARBES, C.; ALOUACHE, M. Control of flight operation of a quad rotor ar . drone using depth map from microsoft kinect sensor. In: . [S.l.: s.n.], 2013. Citado 3 vezes nas páginas 15, 43 e 44.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. Citado 3 vezes nas páginas 15, 32 e 58.

BRADSKI, G. *Camera Calibration*. 2019. Disponível em: <https://docs.opencv.org/3.4.3/dc/dbb/tutorial_py_calibration.html>. Citado na página 32.

Carrio, A. et al. Drone detection using depth maps. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2018. p. 1034–1037. ISSN 2153-0866. Citado 2 vezes nas páginas 45 e 46.

Comaniciu, D.; Ramesh, V.; Meer, P. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 25, n. 5, p. 564–577, May 2003. ISSN 0162-8828. Citado na página 48.

DOLCI, R. Avaliação metrológica de girômetros mems e sua aplicação em testes de inclinação de modelos de embarcações. In: *Universidade Federal de Santa Catarina, Brasil*. [S.l.: s.n.], 2014. Citado na página 42.

EVERINGHAM, M. et al. *"The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results", howpublished = "http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html"*. 2019. Citado na página 74.

FLIGHT TEST FOR KCF TRACKER AND YOLOv3. 2019. Disponível em: <https://www.youtube.com/watch?v=ej41_ICTIQQ>. Citado na página 77.

FRATI, V.; PRATTICHIZZO, D. Using kinect for hand tracking and rendering in wearable haptics. In: . [S.l.: s.n.], 2011. p. 317–321. Citado na página 43.

Henriques, J. F. et al. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 37, n. 3, p. 583–596, March 2015. ISSN 0162-8828. Citado 7 vezes nas páginas 15, 45, 49, 50, 51, 54 e 58.

HOFMANN, M. *Support Vector Machines — Kernels and the Kernel Trick*. 2006. Citado na página 51.

I.Guyon et al. Unsupervised and transfer learning, challenges in machine learning, volume 7. *Unsupervised Learning*, p. 314, Sep 2019. Citado na página 40.

IV, W. S. F.; WALL, J. H.; BEVLY, D. M. Characterization of various imu error sources and the effect on navigation performance. In: . [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 26 e 43.

JANKU, P. et al. Comparison of tracking algorithms implemented in opencv. *MATEC Web of Conferences*, v. 76, p. 04031, 01 2016. Citado na página 45.

Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 7, p. 1409–1422, July 2012. ISSN 0162-8828. Citado na página 45.

KAR, A.; AMITABHA, A.; GUHA, P. Skeletal tracking using microsoft kinect. 06 2019. Citado 2 vezes nas páginas 15 e 43.

Lee, D.; Gyu La, W.; Kim, H. Drone detection and identification system using artificial intelligence. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2018. p. 1131–1133. ISSN 2162-1233. Citado na página 44.

LI JUSTIN JOHNSON, S. Y. F.-F. *Lecture 4:Backpropagation and Neural Networks*. 2017. Disponível em: <http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf>. Citado 2 vezes nas páginas 15 e 39.

Lippmann, R. An introduction to computing with neural nets. *IEEE ASSP Magazine*, v. 4, n. 2, p. 4–22, Apr 1987. ISSN 0740-7467. Citado na página 34.

MALLICK, S. *Home*. 2017. Disponível em: <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/>. Citado na página 46.

MANZINI, N. 2017. Disponível em: <https://www.nicolamanzini.com/single-hidden-layer-neural-network/>. Citado 2 vezes nas páginas 15 e 36.

MAYO HASHAN PUNCHIHEWA, J. E. e. J. M. H. *History of Machine Learning*. 2019. Disponível em: <https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html>. Citado na página 26.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. 1943. *Bulletin of mathematical biology*, v. 52, p. 99–115; discussion 73, 02 1990. Citado na página 33.

MOSSE tracker horizontal position in pixels. 2019. Disponível em: <https://www.youtube.com/watch?v=Gp1cdnJhJYs>. Citado 3 vezes nas páginas 16, 69 e 70.

MOSSE tracker vertical position in pixels. 2019. Disponível em: <https://www.youtube.com/watch?v=HouKLHYVxK4>. Citado 3 vezes nas páginas 16, 69 e 70.

OLIVEIRA, W. S. *VEÍCULO AÉREO NÃO TRIPULADO QUADRIRROTOR: Aspectos Construtivos*. [S.l.], 2014. Citado na página 25.

OZZMAKER. *BerryIMUv2-accelerometer, gyroscope, magnetometer*. 2019. Disponível em: <https://www.tindie.com/products/ozzmaker/berryimuv2-accelerometer-gyroscope-magnetometer/>. Citado 2 vezes nas páginas 15 e 42.

Pan, S. J.; Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 10, p. 1345–1359, Oct 2010. ISSN 2326-3865. Citado na página 41.

PATRãO, S. M. C. Ferramenta de teste e validação para algoritmos de fusão sensorial. In: *Instituto Superior de Engenharia de Coimbra, Portugual*. [S.l.: s.n.], 2015. Citado na página 43.

QI, W.; LI, F.; ZHENZHONG, L. Review on camera calibration. In: . [S.l.: s.n.], 2010. p. 3354 – 3358. Citado na página 31.

RASCHKA, S. *Learning • Labeled data •*. 2015. Disponível em: <https://www.slideshare.net/SebastianRaschka/nextgen-talk-022015/8-Learning_Labeled_data_Direct_feedback>. Citado 2 vezes nas páginas 15 e 35.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>. Citado na página 73.

Redmon, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788. ISSN 1063-6919. Citado 2 vezes nas páginas 15 e 47.

Redmon, J.; Farhadi, A. Yolo9000: Better, faster, stronger. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. p. 6517–6525. ISSN 1063-6919. Citado 2 vezes nas páginas 46 e 58.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018. Citado na página 48.

RESTREPO, R. *Intersect over Union (IoU)*. 2019. Disponível em: <http://ronny.rest/tutorials/module/localization_001/iou/>. Citado 2 vezes nas páginas 15 e 47.

Rozantsev, A.; Lepetit, V.; Fua, P. Detecting flying objects using a single moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 5, p. 879–892, May 2017. ISSN 0162-8828. Citado na página 44.

SANTANA, L. M. S.; SANTOS, D. A.; VIANA, I. B. A low-cost position and velocity estimation system for multirotor aerial vehicles using a static kinect sensor. nstituto Tecnológico de Aeronáutica, Praça Mal. Eduardo Gomes, 50. Vila da Acácias, 12228-900. São José dos Campos, São Paulo, Brasil, p. 7, 2016. Citado 3 vezes nas páginas 15, 41 e 46.

SANTANA, M. *Deep Learning: do Conceito às Aplicações*. 2018. Disponível em: <https://medium.com/data-hackers/deep-learning-do-conceito-%C3%A0s-aplica%C3%A7%C3%B5es-e8e91a7c7eaf>. Citado na página 27.

Saqib, M. et al. A study on detecting drones using deep convolutional neural networks. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. [S.l.: s.n.], 2017. p. 1–5. Citado na página 45.

SAZLI, M. A brief review of feed-forward neural networks. *Communications, Faculty Of Science, University of Ankara*, v. 50, p. 11–17, 01 2006. Citado 2 vezes nas páginas 37 e 39.

SHANMUGAMANI, R. *Deep Learning for Computer Vision*. Packt Publishing, 2019. Disponível em: <https://learning.oreilly.com/library/view/deep-learning-for/9781788295628/d6fe3d14-d576-40e3-b76d-c60bc316ba80.xhtml>. Citado 2 vezes nas páginas 15 e 37.

SHARMA, M. *Support Vector Machines – Not for the faint-hearted*. 2019. Disponível em: <https://storybydata.com/datacated-weekly/support-vector-machines-not-for-the-faint-hearted/>. Citado 2 vezes nas páginas 15 e 51.

Siong, L. Y. et al. Motion detection using lucas kanade algorithm and application enhancement. In: *2009 International Conference on Electrical Engineering and Informatics*. [S.l.: s.n.], 2009. v. 02, p. 537–542. ISSN 2155-6822. Citado na página 45.

Sistema integrado utilizando o Rastreador MOSSE. 2019. Disponível em: <https://www.youtube.com/watch?v=IWIQw3qqo9E>. Citado na página 76.

SIVA, C. *Machine Learning and Pattern Recognition*. 2018. Disponível em: <https://dzone.com/articles/machine-learning-and-pattern-recognition>. Citado 3 vezes nas páginas 15, 44 e 45.

TEST FLIGHT WITH MOSSE TRACKER AND YOLO_v3. 2019. Disponível em: <https://www.youtube.com/watch?v=l5l6udnbBys&t=1s>. Citado na página 77.

TIWARI, A. Position control of an unmanned aerial vehicle from a mobile ground vehicle. 01 2017. Citado na página 57.

Tracker KCF position in pixels. 2019. Disponível em: <https://www.youtube.com/watch?v=C_O3lGdmBtA>. Citado 3 vezes nas páginas 16, 68 e 69.

VYAS, K. *A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles (UAVs)*. 2019. Disponível em: <https://interestingengineering.com/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs>. Citado na página 25.

WANG, A.; QIU, T.; SHAO, L.-T. A simple method of radial distortion correction with centre of distortion estimation. *Journal of Mathematical Imaging and Vision*, v. 35, p. 165–172, 11 2009. Citado na página 32.

WILLEMS, K. *Keras Tutorial: Deep Learning in Python*. 2019. Disponível em: <https://www.datacamp.com/community/tutorials/deep-learning-python>. Citado 2 vezes nas páginas 15 e 34.

WILSON, D.; GöKTOğAN, A.; SUKKARIEH, S. Guidance and navigation for uav airborne docking. In: . [S.l.: s.n.], 2015. Citado na página 57.

YOLOv3 and KCF algorithm for automatic drone tracking. 2019. Disponível em: <https://www.youtube.com/watch?v=nnblIAuLSJE>. Citado na página 76.

YOLOV3 DL algorithm for drone detection. 2019. Disponível em: <https://www.youtube.com/watch?v=d15U01PYSuU>. Citado na página 74.

ZHANG, Z. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, v. 19, n. 2, p. 4–10, Feb 2012. ISSN 1070-986X. Citado 3 vezes nas páginas 15, 31 e 32.

Zul Azfar, A.; Hazry, D. A simple approach on implementing imu sensor fusion in pid controller for stabilizing quadrotor flight control. In: *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*. [S.l.: s.n.], 2011. p. 28–32. Citado na página 57.