http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2021.v13.n4.e375
ISSN on-line 2178-8332

Colloquium Exactarum

Submetido: 28/06/2021 Revisado: 16/12/2021 Aceito:21/02/2022

UTILIZANDO PROCESSAMENTO DE IMAGENS E YOLO PARA A CONSTRUÇÃO DE UM SISTEMA DE NAVEGAÇÃO DE UM DRONE COM APLICAÇÃO EM UMA INDÚSTRIA

Using image processing and YOLO to build a navigation system for a drone with an industrial application

João Vitor Sabino[1], Francisco Assis da Silva[1], Leandro Luiz de Almeida[1], Danillo Roberto Pereira[1], Almir Olivette Artero[2]

[1]Faculdade de Informática de Presidente Prudente, Unoeste - Universidade do Oeste Paulista, Presidente Prudente.
joaovitorbmx11@hotmail.com, chico@unoeste.br, llalmeida@unoeste.br, dpereira@analytics2go.com
[2]Faculdade de Ciências e Tecnologia, UNESP - Universidade Estadual Paulista Departamento de Matemática e Computação, Presidente Prudente.
almir@fct.unesp.br

RESUMO – Neste trabalho foi desenvolvido um sistema de navegação semiautônomo de um drone para uma indústria de caixas de papelão com a finalidade de auxiliar na contagem do estoque de bobinas de papelão. A metodologia desenvolvida possui quatro etapas principais, sendo a decodificação de QR Code, detecção de marcadores ópticos, sistema de navegação e movimentação do drone. Para a etapa de decodificação de QR Code foi utilizada a biblioteca pyzbar. Na etapa de detecção do marcador óptico foi utilizada a biblioteca YOLOv4 Tiny, que faz o uso de técnicas de aprendizagem de máquina para detectar objetos em tempo real. A YOLOv4 Tiny foi treinada utilizando um *dataset* personalizado com imagens dos marcadores ópticos e etiquetas em um ambiente de simulação fechado, obtendo uma taxa de acerto de 92,10%. A etapa do sistema de navegação é alimentada pela resposta da rede neural, na qual cada marcador tem uma função associada a ele. A última etapa depende do sistema de navegação, uma vez que este envia qual será o comando em que o drone deve seguir e a movimentação envia este comando ao drone.
Palavras-chave: Drone; YOLOv4 Tiny; visão computacional; aprendizado de máquina.

ABSTRACT – In this work we developed a semi-autonomous drone navigation system for a cardboard box industry, to assist in counting the stock of cardboard reels. The developed methodology has four main steps, being the QR Code decoding, optical marker detection, navigation system and drone movement. For the QR Code decoding step, the pyzbar library was used. In the optical marker detection step, the YOLOv4 Tiny library was used, which uses machine learning techniques to detect objects in real time. YOLOv4 Tiny was trained using a custom dataset with images of optical markers and labels in a closed simulation environment, achieving a hit rate of 92.10%. The navigation system step is fed by the response of the neural network, in which each marker has a function associated with it. The last step depends on the navigation system, since it sends which command the drone must follow and the movement sends this command to the drone.
Keywords: Drone; YOLOv4 Tiny; computer vision; machine learning.

1. INTRODUÇÃO

A gestão de inventários, quando bem realizada, pode aumentar a produtividade, reduzir custos operacionais e, por consequência, otimizar os lucros do negócio. Sendo assim, adotar o uso de tecnologia no estoque é uma das melhores maneiras de modernizar os processos e garantir que a empresa esteja apta a competir em seu mercado. Investir em tecnologia no estoque é uma das formas mais eficientes e com melhor custo-benefício para garantir um trabalho mais produtivo e rentável em cada etapa da logística (DRUMOND, 2020).

A indústria 4.0 é movida a dados, por meio deles que tecnologias como IoT (*Internet of Things*) ou Aprendizado de Máquina (*Machine Learning*) se fazem possíveis. O acúmulo e análise das informações faz com que os processos dentro das fábricas alinhadas com esse conceito sejam otimizados, a fim de aumentar a produção e diminuir os custos operacionais (PEDERNEIRAS, 2020).

O uso de drones já era muito comum em empresas do setor agrícola e de segurança. Mas dentro do contexto de Indústria 4.0 estes veículos aéreos não tripulados podem fazer a diferença ao aprimorar os processos nas fábricas. A popularidade do drone fez com que diferentes setores da economia testassem como a tecnologia poderia ser benéfica aos seus negócios. Em 2020, o Brasil contava com cerca de 73 mil drones cadastrados na Agência Nacional de Aviação Civil (ANAC), órgão que regulamenta o uso da ferramenta (SENIOR, 2020).

O uso crescente dos drones representa uma aplicação prática e combinada de Inteligência Artificial e Aprendizado de Máquina. Embora eles sejam capazes de fazer muito mais do que tarefas ligadas à vigilância, os avanços na detecção de objetos expandiram o uso desses equipamentos à análise de imagens e vídeos (WEDEMANN, 2019).

O conceito de Inteligência Artificial (IA) se refere à criação de máquinas – não necessariamente com corpo físico – com a habilidade de pensar e agir como seres humanos. Softwares que conseguem abstrair, criar, deduzir e aprender ideias. O objetivo geralmente está em facilitar tarefas do dia a dia, avançar pesquisas científicas e modernizar indústrias (CABRAL, 2018).

A Visão Computacional em conjunto com a Inteligência Artificial pode ser aplicada para o treinamento de computadores em busca da compreensão e interpretação do mundo visual. Isso pode ocorrer com o uso de Aprendizado Profundo (*Deep Learning*), a partir de imagens que permitem às máquinas reagirem e tomarem decisão de acordo com o que elas enxergam (TURCATO, 2019).

Este trabalho busca contribuir com o desenvolvimento de um sistema de navegação semiautônomo para um drone percorrer os corredores e as prateleiras do estoque de bobinas de papelão de uma fábrica de caixas de papelão e auxiliar na contagem desse estoque.

Essa fábrica de caixas de papelão em questão utiliza um tempo considerável para realizar a contagem das bobinas em estoque. Muitas vezes existem bobinas que não estão no controle, pois não foi feita a entrada deste material no inventário, isso faz com que sejam adquiridos novos materiais, gerando gastos a mais ao final do mês.

Dedicar um funcionário para realizar o controle de estoque no momento é algo dispendioso, visto que se trata de uma tarefa monótona e rotineira. Acredita-se que essa tarefa pode ser automatizada com o uso de drone, que percorreria os corredores e as prateleiras do estoque e faria a contagem das bobinas, sem interferência humana.

Para a realização do trabalho foi utilizado um drone modelo dji Tello (Figura 1) para percorrer os corredores e as prateleiras e fazer a varredura de todas as bobinas de papelão para contabilizar o estoque. As imagens capturadas pela câmera do drone são enviadas para um computador que é responsável por gerenciar as rotas e também por contabilizar o estoque. A partir dessas imagens, o algoritmo, desenvolvido neste

trabalho, busca detectar códigos QR Code, presentes nas etiquetas que são fixadas às bobinas de papelão no estoque da fábrica. Também foram utilizados marcadores ópticos ArUco (GARRIDO-JURADO *et al*., 2014) para serem utilizados como indicadores de direção ao drone, e com isso construir a rota gerenciada pelo sistema de navegação.

O trabalho está organizado da seguinte maneira. Na Seção 2 encontram-se trabalhos relacionados ao apresentando neste artigo. Na Seção 3 é apresentada a biblioteca YOLO, utilizada para a detecção dos marcadores ópticos e etiquetas em tempo real. Na Seção 4 são apresentados os métodos para decodificação de QR Codes, detecção dos marcadores ópticos, sistema de navegação desenvolvido neste trabalho e a movimentação do drone. A Seção 5 apresenta os experimentos e resultados obtidos a partir da metodologia desenvolvida. Por fim, na Seção 6 são feitas as considerações finais do trabalho.

Figura 1. Drone dji Tello.

Fonte: AIRBUZZ.ONE (2018).

2. TRABALHOS RELACIONADOS

Nesta seção são descritos, de forma sucinta, trabalhos encontrados na literatura que serviram de base de conhecimento para a realização deste.

O trabalho desenvolvido por Cavadas (2019) teve como objetivo a programação de um drone de forma individual, bem como também a programação de um enxame com quatro drones. Seu principal uso foi para fins educacionais, em que o autor ensina pessoas sem conhecimento prévio a programar missões simples para o drone, e com isso a programação avançada se torna algo mais fácil de ser aplicado.

O autor propõe controlar o enxame de drones por meio de movimentos corporais, e para isso foi usada a biblioteca OpenCV. Ele explica como conectar o drone a rede Wifi e configurar, para receber os comandos via Python, então é criada uma série de comandos para fazer com que o drone siga um roteiro de comandos em um *script*. Um problema enfrentado pelo autor foi o comportamento incorreto do drone, por causa de mensagens perdidas, e também a falta de sincronismo entre o enxame de drones.

Para o controle do enxame dos quatro drones foram desenvolvidas três aplicações. Na primeira aplicação foi possível enviar comandos para cada drone ou uma série de comandos e executá-los de uma só vez. A segunda aplicação leva em conta o posicionamento dos drones no espaço e foi usado um algoritmo para evitar a colisão entre os mesmos. A terceira aplicação envolve fazer movimentos mais precisos através de marcações no chão.

O trabalho desenvolvido por Saraiva (2018) teve como objetivo a criação de um sistema de navegação, usando Visão Computacional. O autor criou um drone capaz de percorrer um labirinto de forma autônoma, e desviar de obstáculos. No desenvolvimento do trabalho foi utilizada a controladora de voo Navio 2, juntamente com um Raspberry Pi 3 modelo B, além de uma câmera com 30 FPS, e a linguagem Python com a biblioteca de Visão Computacional OpenCV.

Em sua metodologia, o autor, a partir de cada imagem capturada pela câmera, converte essa para tons de cinza, aplica um filtro passa-baixa de desfoque, nesse caso foi usado o borramento Gaussiano, e foi utilizado o algoritmo Canny para a detecção de bordas. Como desejava-se obter os pontos cartesianos no plano, foi usado o algoritmo Probabilistic Hough Transform, que, ao invés de serem usados todos os pontos da imagem,

usa-se apenas uma parte deles para ser criada uma reta para o drone se guiar e locomover, representando uma economia de processamento.

3. YOLO (*You Only Look Once)*

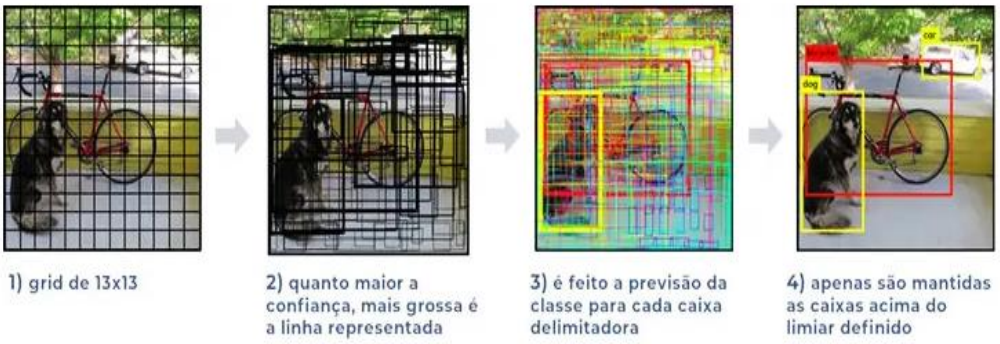
YOLO é uma biblioteca para detecção e reconhecimento de objetos de passada única (*single pass*) que utiliza uma rede neural convolucional (CNN) como extrator de características (*features*) (REDMON *et al*., 2016).

Diferente de outros métodos para detecção de objetos, como Region-based Convolutional Neural Networks (R-CNN) (GANDHI, 2018) que leva em média 50 segundos para processar uma imagem ou Faster R-CNN (GANDHI, 2018) que leva 0.2 segundos para processar a imagem (ALVES, 2020), a YOLO analisa a imagem uma única vez. Com isso, seu tempo de inferência é de apenas 0.05 segundos, se tornando viável para ser utilizado na detecção de objetos em tempo real (ALVES, 2020).

A YOLO em sua versão inicial já utilizava a imagem inteira para a detecção de cada objeto. Com isso é utilizada apenas uma rede neural convolucional que funciona da seguinte maneira: a imagem de entrada é redimensionada para 448x448 pixels e em seguida é dividida em uma grade SxS. Cada célula da grade é responsável por gerar caixas delimitadoras dos objetos, junto com um grau de confiança. Essa pontuação de confiança é obtida pela multiplicação entre a probabilidade do objeto e IoU (união sobre a intersecção) da célula da grade com a caixa.

Caso a pontuação seja zero, significa que não existem objetos dentro da caixa (Figura 2). As caixas selecionadas são aquelas que possuem o maior nível de pontuação (ZANGRANDI, 2019), (REDMON *et al*., 2016).

Figura 2. Funcionamento da biblioteca YOLO, usando uma grade de 13x13.

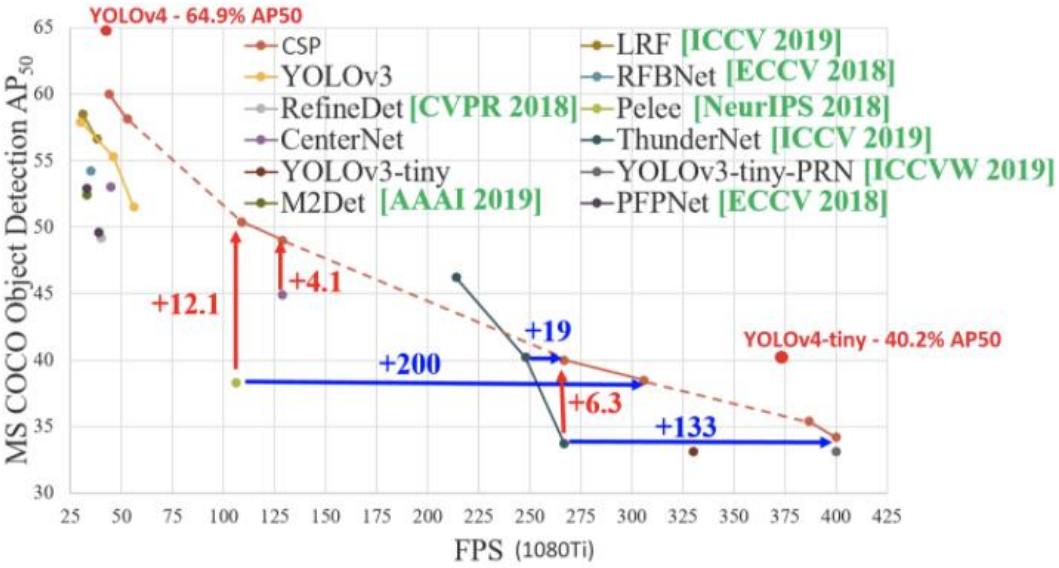


Fonte: Alves (2020).

Existem variações da biblioteca YOLO, como a YOLOv1, YOLOv2, YOLOv3, YOLOv4 e uma versão especial denominada de YOLOv4 Tiny, que foi escolhida para ser utilizada neste trabalho.

Desde a criação da YOLO, seu algoritmo foi sendo melhorado, e a cada nova versão, foi ficando com maior precisão e velocidade. No entanto, é possível observar no gráfico da Figura 3, em que foram realizados testes com a base de dados MS COCO (TECHZIZOU, 2020) utilizando uma placa de vídeo (GTX 1080Ti), que quanto maior a quantidade de FPS (frames por segundo) (eixo *x*) na rede neural, menor é a precisão de acerto (eixo *y*). Porém, é possível visualizar que a rede neural YOLOv4, com uma média de 25 a 30 FPS obtém uma precisão de 64,9%, enquanto a rede neural YOLOv3 obtém uma média entre 55% a 60%, utilizando a mesma quantidade de FPS.

Figura 3. Comparação de redes neurais convolucionais.



Fonte: Techzizou (2020).

A YOLOv4 Tiny (MELLO, 2021) foi projetada para dispositivos com poucos recursos computacionais, como memória, CPU e GPU. O número de acesso à memória é reduzido, mantendo o mesmo número de canais na saída das camadas de convolução.

Essa versão é muito mais rápida para se treinar, quando comparado com as outras versões, pois há uma redução na sua estrutura. A YOLOv4 Tiny utilizada neste projeto conta com 21 camadas de convolução, 3 camadas de *maxpool*, 2 camadas yolo, 11 camadas de rota e 1 camada net.

As camadas de convolução são as camadas na qual se extrai as características da imagem. Nestas camadas um filtro percorre a imagem de entrada realizando a operação de convolução sobre cada região e o resultado dessa operação é o conjunto de características da imagem.

A camada *maxpool* consiste em uma espécie de filtro que percorre a matriz de entrada (que representa a imagem) e obtém o valor máximo de cada região coberta pelo filtro.

A camada *yolo* é uma camada totalmente conectada na qual tem-se a classe de cada objeto.

Cada camada seguinte recebe como entrada o resultado da camada anterior, com a camada de rota o resultado da camada seguinte não vem da camada anterior e sim da camada definida pela camada de rota.

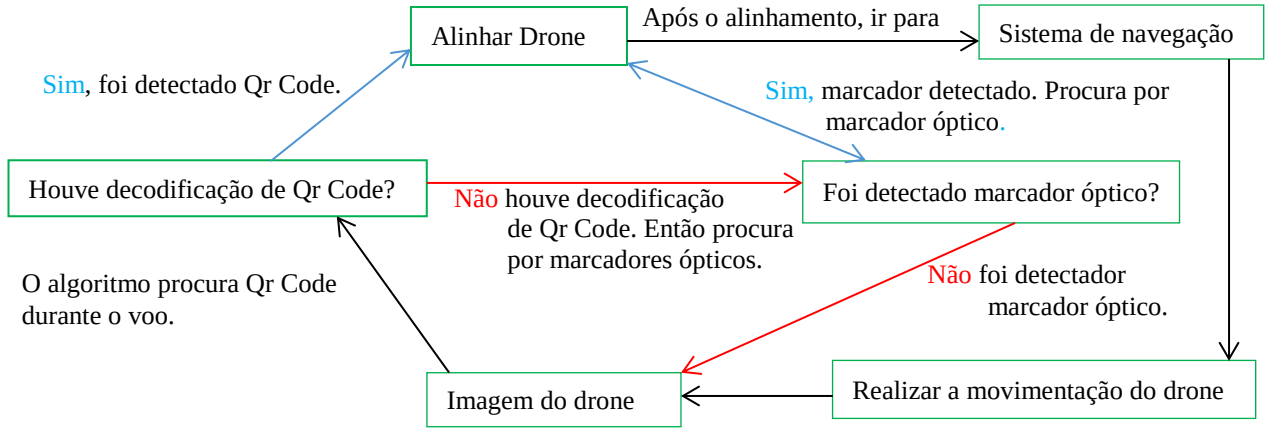
A camada *net* é a entrada da rede na qual são definidos alguns parâmetros como números de canais de cores, tamanho da imagem, quantidade de treinamento entre outros.

Para detecção de objetos em tempo real, a YOLOv4 Tiny é mais recomendada, pois, mesmo possuindo menos precisão do que a YOLOv4, seu tempo de inferência é mais rápido (MELLO, 2021; TECHZIZOU, 2020).

4. MÉTODO PROPOSTO

Nesta seção é descrito o funcionamento do método proposto, sendo dividido em quatro etapas: decodificação de QR Codes impressos nas etiquetas das bobinas de papelão; detecção de marcador óptico ArUco; sistema de navegação; e controle do drone. Na Figura 4 é apresentado um fluxograma representando essas etapas.

Figura 4. Fluxograma representando as etapas do método proposto.



Fonte: Os autores.

4.1 Decodificação de QR Codes

Para realizar a decodificação de QR Codes impressos nas etiquetas e fixados nas bobinas de papelão foi utilizada a biblioteca pyzbar (HUDSON, 2019). Esta biblioteca é compatível com a versão do Python 3.6.9. Versão essa utilizada no algoritmo de decodificação de QR Codes e na detecção de marcadores ópticos (ArUco).

Ao localizar uma etiqueta de uma bobina de papelão, são decodificadas as informações do QR Code para contabilizar o estoque. Na Figura 5, um retângulo na cor verde foi desenhado no entorno do QR Code na imagem para demonstrar a detecção, além de ser exibida na cor azul a informação decodificada.

Figura 5. Localização e decodificação de um QR Code presente em uma etiqueta.

Fonte: Os autores.

O QR Code presente na etiqueta serve para armazenar as informações contidas nos campos, como: o número do palete, número do pedido, quantidade e o identificador do palete (Figura 6).

Figura 6. Modelo de etiqueta presente nas bobinas da indústria. Essa etiqueta é usada nas bobinas e nos paletes.

Fonte: Os autores.

4.2 Detecção do marcador óptico

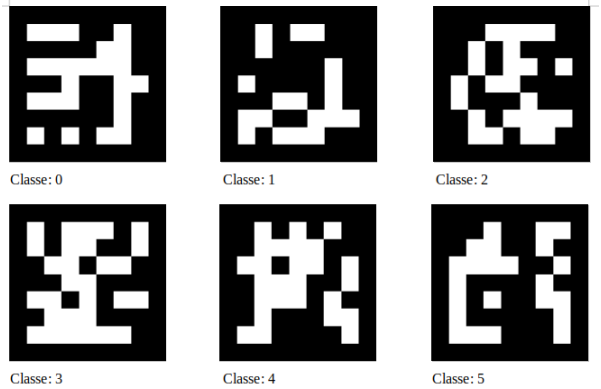
Para realizar a detecção dos marcadores ópticos foi utilizada a biblioteca YOLOv4 Tiny com um *dataset* de imagens de ArUco (Figura 7), construído neste trabalho.

Um marcador ArUco é um marcador quadrado composto por uma ampla borda preta e uma matriz binária interna. O tamanho do marcador determina o tamanho da matriz interna. Neste trabalho, foram utilizados marcadores do tamanho 7x7,

totalizando 49 regiões (células) (GARRIDO-JURADO *et al.*, 2014).

As imagens dos marcadores ópticos ArUco e etiquetas foram obtidas em um ambiente de simulação, em que foram capturadas imagens com ângulos e iluminações diferentes. Foram obtidas 81 imagens de 2048x1152 pixels por classe, num total de 569 imagens, divididas em seis classes diferentes, sendo elas: 0, 1, 2, 3, 4, 5.

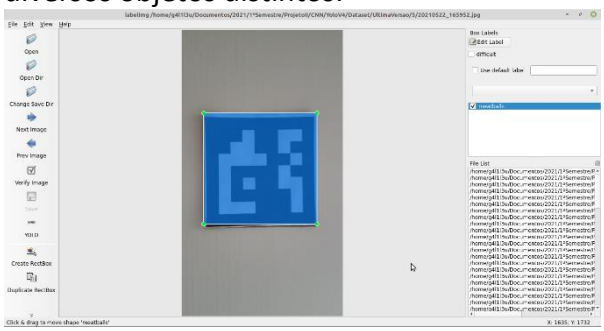
Figura 7. Marcadores ArUco no tamanho 7x7 utilizados neste trabalho.



Fonte: Os autores.

A preparação do *dataset* de imagens foi realizada com o uso do programa labelimage (TZUTALIN, 2018) para demarcar cada ArUco, pois a rede YOLO necessita de um arquivo texto para cada imagem contendo a localização do objeto a ser aprendido (Figura 8).

Figura 8. Programa labelimage, utilizado para demarcar o objeto de interesse, presente em uma imagem que pode conter diversos objetos distintos.



Fonte: Os autores.

A rede neural foi treinada no Google Colab (GOOGLE, 2021), pois oferece GPU gratuita por tempo limitado e funciona em um ambiente interativo chamado notebook Colab, que permite escrever e executar código.

O Google Colab conta com processadores Intel® Xeon® 2.30GHz, GPU modelo Tesla T4 com 12GB de RAM. Além disso, possui algumas bibliotecas pré-instaladas como o CUDA, Python, TensorFlow entre outras (RODRIGUES, 2018).

Os notebooks do Colab permitem aos usuários escrever e executar códigos em Python, assim como escrever textos para descrever o funcionamento do código. Esses notebooks são armazenados na conta do Google Drive do próprio usuário (GOOGLE, 2021).

O treinamento levou em média 11 horas para ser realizado. A rede foi treinada no *framework* Darknet (ALVES, 2020), usando os três canais de cores RGB, 14.000 épocas de treinamento e imagens com dimensões 608x608 pixels na entrada da rede.

Darknet é um *framework* de rede neural profunda de código aberto. É uma estrutura rápida e altamente precisa (a precisão do modelo treinado personalizado depende dos dados de treinamento, épocas, tamanho do lote e alguns outros fatores) para detecção de objetos em tempo real. Essa estrutura de rede é escrita na linguagem de programação C usando a biblioteca CUDA.

CUDA é a plataforma de computação paralela desenvolvida pela NVIDIA. É uma extensão para a linguagem de programação C, a qual permite que os programadores possam usar os poderes da unidade de processamento gráfico (GPU) para realizar algumas operações rapidamente (ARRUDA, 2011) (ALVES, 2020).

O erro da YOLOv4 Tiny durante o treinamento foi de 0,007518, ou seja, a rede no ambiente de simulação obteve 92,10% de acerto (Figura 9). Através do gráfico da Figura 10 é possível observar que, após 1.400 épocas de treinamento, a rede se mantém com 100% de acerto até a finalização do treinamento em 14.000 épocas. Ou seja, tem-se o limiar da curva de perda que está

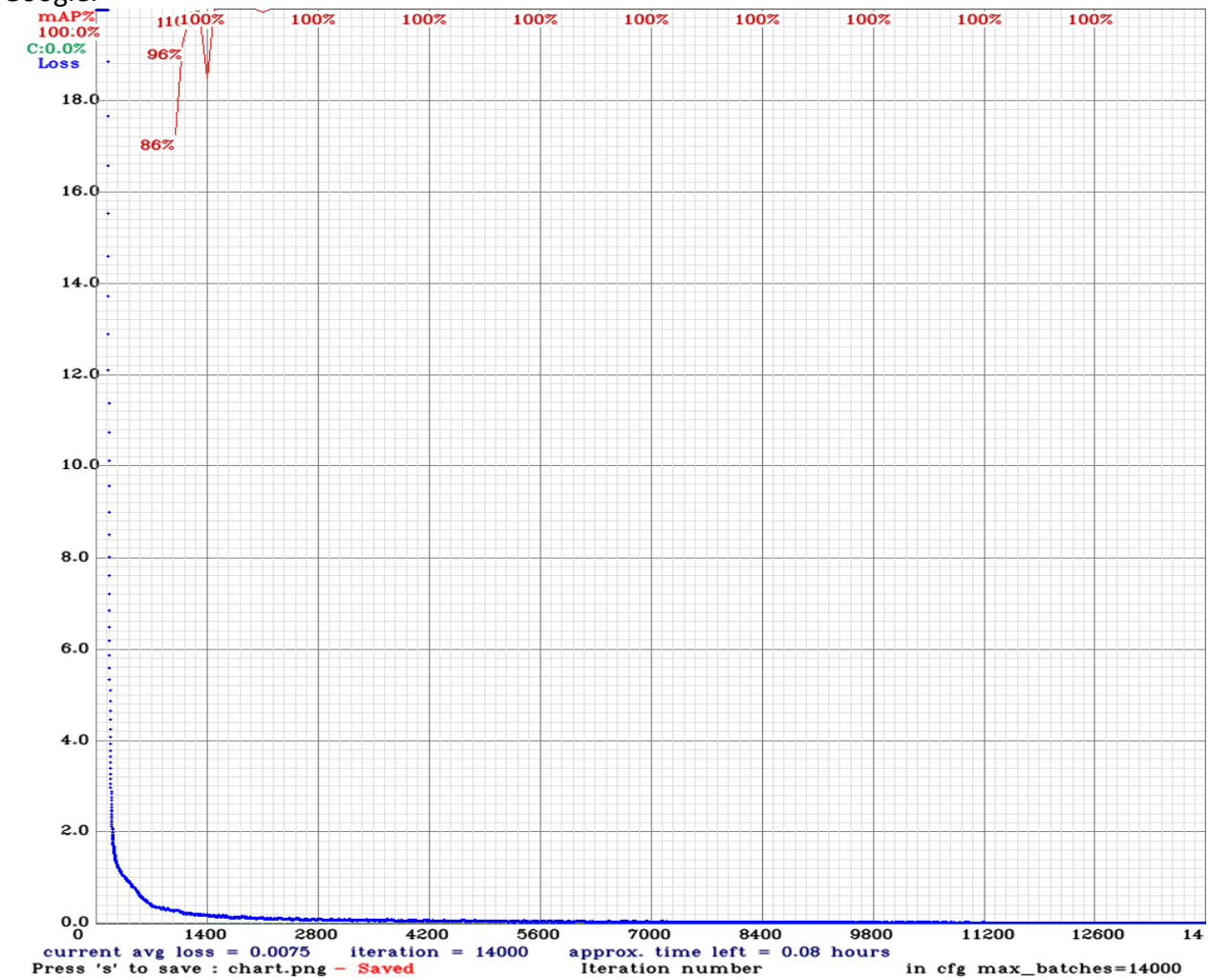
representada na cor azul e a porcentagem de acerto representada com a linha vermelha.

Figura 9. Resultado do treinamento da biblioteca YOLOv4 Tiny com o *dataset* de imagens dos marcadores ArUco utilizado neste trabalho.

```
 14000: 0.001739, 0.007518 avg loss, 0.000026 rate, 1.005926 seconds, 896000 images, 0.077999 hours left

 calculation mAP (mean average precision)...
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28

 detections_count = 116, unique_truth_count = 114
 rank = 0 of ranks = 116
 rank = 100 of ranks = 116
class_id = 0, name = 0, ap = 100.00%      (TP = 12, FP = 0)
class_id = 1, name = 1, ap = 100.00%      (TP = 17, FP = 0)
class_id = 2, name = 2, ap = 100.00%      (TP = 15, FP = 0)
class_id = 3, name = 3, ap = 100.00%      (TP = 21, FP = 0)
class_id = 4, name = 4, ap = 100.00%      (TP = 18, FP = 0)
class_id = 5, name = 5, ap = 100.00%      (TP = 15, FP = 0)
class_id = 6, name = 6, ap = 100.00%      (TP = 16, FP = 0)

 for conf_thresh = 0.25, precision = 1.00, recall = 1.00, F1-score = 1.00
 for conf_thresh = 0.25, TP = 114, FP = 0, FN = 0, average IoU = 92.10 %
```

Fonte: Os autores.

Figura 10. Gráfico de treinamento da biblioteca YOLOv4 Tiny com o *dataset* de imagens dos marcadores ArUco utilizado neste trabalho. Treinamento realizado na plataforma colaborativa do Google.



Fonte: Os autores.

4.3 Sistema de navegação

O sistema de navegação tem como princípio a resposta da rede neural, ou seja, quando um marcador óptico (ArUco) é detectado, ele possui uma classe (total de cinco classes diferentes), a qual é associada a um comando que deve ser interpretado pela função de movimentação do drone.

O sistema recebe da função da rede neural a área do marcador óptico, e com isso, se a área for muito próxima ao drone, o sistema envia um comando à movimentação que faz com que o drone se afaste evitando uma colisão. Esse sistema de colisão também é alimentado pela área do QR Code, quando identificado. Com isso, também faz o drone se aproximar e afastar.

Para cada marcador ArUco, um comando diferente é enviado. A rede neural possui cinco classes de marcadores ópticos e uma classe etiqueta. Um modelo de exemplo com números (na prática são marcadores ArUco) dispostos em uma sequência para sobrevoo do drone, representando um ambiente da indústria de caixas de papelão pode ser visto na Figura 11.

O marcador 0 indica o início de uma prateleira, com isso o drone se desloca da esquerda para a direita até encontrar o marcador óptico número 1.

O marcador 1 é referente a descida do drone, ou seja, quando detectado, o drone desce até encontrar outro marcador, que pode ser 2, 3, 4 ou 5.

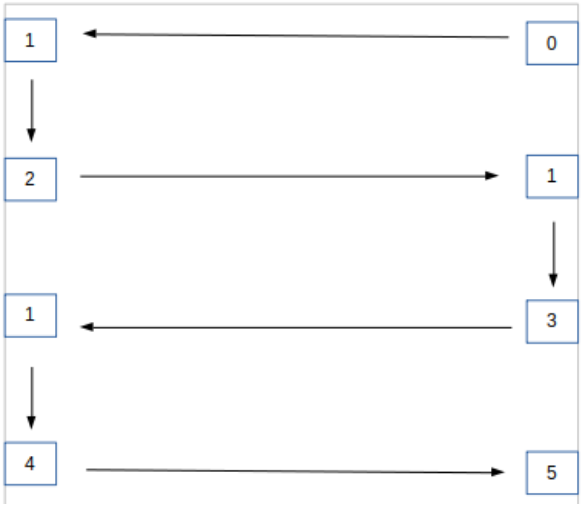
O marcador 2 é responsável por fazer o drone ir da esquerda para a direita, até encontrar o marcador óptico 1.

O marcador 3 é responsável por fazer o drone ir da direita para a esquerda, assim como o marcador 0, porém, esse não indica o início da prateleira.

Já os marcadores 4 e 5 indicam o fim da prateleira.

Se o marcador 4 for detectado antes do marcador 5 o drone vai da esquerda para a direita, e assim que for detectado o marcador 5 o drone irá pousar, pois dessa maneira toda a prateleira foi varrida. O mesmo princípio é aplicado se o marcador 5 for detectado antes do marcador 4, porém o drone vai da direita para a esquerda.

Figura 11. Modelo funcionamento do sistema de navegação.

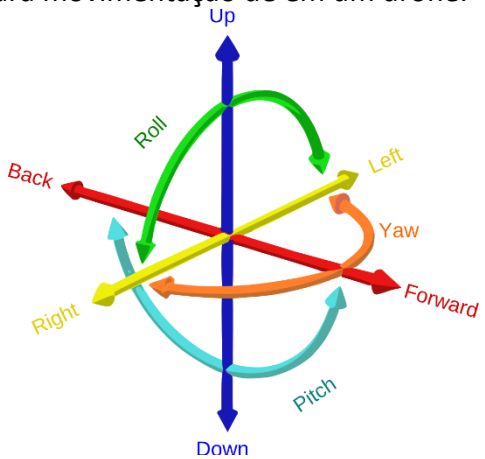


Fonte: Os autores.

4.4 Movimentação do drone

O princípio de comandar um drone são os seis graus de liberdade: sobe e desce, esquerda e direita, frente e trás, rotação esquerda e direita, inclinação para frente e para trás, e inclinação esquerda e direita (Figura 12).

Figura 12. Graus de liberdade possíveis para movimentação de em um drone.



Fonte: Stroski, (2020)

Para realizar a movimentação no drone são utilizados quatro canais de movimentação sendo eles o canal 1 (*Roll*), canal 2 (*Pitch*), canal 3 (*Up*/*Down*) e canal 4

(*Yaw*) e pode-se combinar o uso desses, a fim de obter todos os graus de liberdade.

5. EXPERIMENTOS

Os experimentos foram executados utilizando um computador com processador Intel® Core® i3 da 2ª geração com 6 GB de memória RAM utilizando a CPU.

O teste do sistema de navegação foi executado em tempo real e em um ambiente de simulação fechado (Figura 13). Foi utilizado o drone dji Tello, com sua entrada de vídeo redimensionada para 640x480 pixels, devido ao fraco poder computacional do computador utilizado no teste. A entrada da imagem na rede neural foi redimensionada para 608x608 pixels, conforme configurado no treinamento da rede neural na plataforma colaborativa do Google.

O drone foi posicionado no marcador óptico inicial 0, em seguida foi liberado para que o algoritmo de navegação executasse toda a trajetória de forma semiautônoma.

Durante o trajeto, o drone foi decodificando os QR Codes e detectando os marcadores ópticos. O experimento levou cerca de dois minutos e vinte segundos para ser concluído.

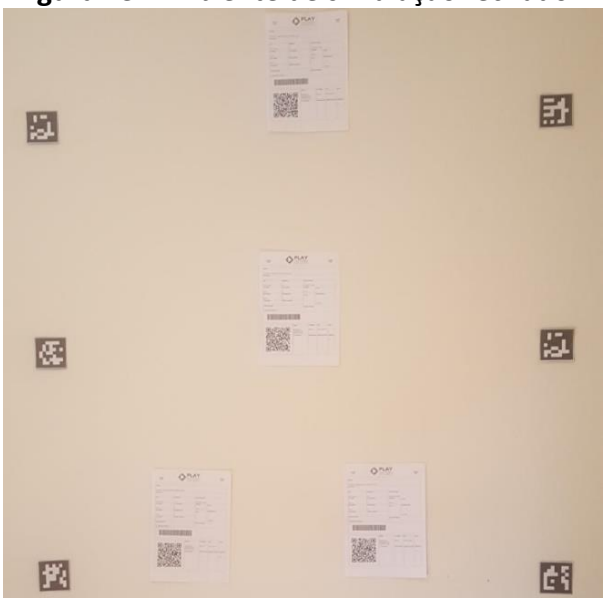
A rede neural foi ativada após o posicionamento correto do drone no local desejado, o ambiente de simulação contou com quatro etiquetas e seis marcadores ópticos.

A detecção dos marcadores ópticos sempre terá acertos acima de 0,85%, pois foi definida no algoritmo da rede neural a confiabilidade com no mínimo essa precisão.

Devido ao fraco poder computacional do computaodr utilizado para a execução do teste, o processamento do vídeo pela rede neural em tempo real sofre pequenos atrasos. Faz-se necessário um processador mais eficiente, ou seja, com maior número de núcleos*, v*isto que o programa como um todo utiliza várias *threads* simultaneamente para a execução da rede neural e comunicação com o drone.

A grande vantagem de se usar a rede neural convolucional YOLOv4 Tiny é não ser necessário ter uma placa de vídeo para o processamento de imagens em tempo real.

Figura 13. Ambiente de simulação fechado.



Fonte: Os autores.

6. CONSIDERAÇÕES FINAIS

A partir dos experimentos realizados, já se pode constatar que seja possível testar o sistema como um todo na indústria de caixas de papelão, para realizar a varredura de estoque e a contagem das bobinas de papelão.

A utilização de uma placa de vídeo ou um processador mais eficiente poderá apresentar melhores resultados, uma vez que esses contam com um maior número de núcleos de processamento para a execução da rede neural convolucional.

Para indústrias que contam com inventários maiores, uma alternativa seria a utilização de um drone com maior autonomia de voo, pois o modelo dji Tello conta somente com 15 minutos de autonomia.

A grande vantagem do algoritmo do sistema de navegação é que ele não depende exclusivamente desse modelo de drone, pois a rede neural pode ser reutilizada e o sistema de navegação também. Apenas o sistema de movimentação terá que ser alterado para se ajustar à forma de comunicação que esse

novo drone venha a utilizar para receber e enviar os dados.

Algumas limitações podem vir a ocorrer na detecção do marcador óptico. Caso a iluminação do local seja mínima, ou se os marcadores ópticos forem posicionados de forma irregular. Se não iniciar com o marcador zero, ou trocar a posição do marcador 1 que é responsável por fazer com que o drone desça, isso acarretará na desorientação do drone, ou a não detecção do mesmo.

REFERÊNCIAS

AIRBUZZ.ONE. **DJI Tello Review aircraft diagram**. 2018. Disponível em: https://airbuzz.one/dji-tello-review/dji-tello-review_aircraft_diagram/. Acesso em: 2 fev. 2020.

ALVES, G. **Detecção de Objetos com YOLO – Uma abordagem moderna:** Conceitos sobre IA. 2020. Disponível em: https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna. Acesso em: 3 jun. 2021.

ARRUDA, F. **NVIDIA CUDA**: o que é e como funciona. 2011. Disponível em: https://www.tecmundo.com.br/computacao-grafica/10507-nvidia-cuda-o-que-e-e-como-funciona.htm. Acesso em: 01 fev. 2021.

CABRAL, I. **Tudo sobre inteligência artificial: 10 fatos que você precisa saber:** Significado de IA é relacionado à capacidade de máquinas aprenderem a pensar e a agir como humanos. 2018. Disponível em: https://www.techtudo.com.br/listas/2018/05/tudo-sobre-inteligencia-artificial-10-fatos-que-voce-precisa-saber.ghtml. Acesso em: 2 ago. 2020.

CAVADAS, J. A. J. **Using Drone Tello Edu for educational purposes**. 2019. Tese (Master's degree in Applications and Technologies for Unmanned Aircraft Systems) - Universitat Politecnica de Catalunya Barcelonatech, Bracelona, 2019. Disponível em: https://upcommons.upc.edu/bitstream/handle/2117/173834/memoria.pdf?sequence=1&isAllowed=y. Acesso em: 19 nov. 2020.

DRUMOND, T. **Tecnologia no estoque:** conheça quais são as principais. 2020. Disponível em: https://blog.sogalpoes.com.br/?p=1078. Acesso em: 23 out. 2020.

GANDHI, R. **R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms: Understanding object detection algorithms**. 2018. Disponível em: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. Acesso em: 14 mar. 2021.

GARRIDO-JURADO, S. *et al.* Automatic generation and detection of highly reliable fiducial markers under occlusion. **Pattern Recognition**, v. 47, n. 6, p. 2280–2292, 2014. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320314000235?via%3Dihub. Acesso em: 14 mar. 2021. https://doi.org/10.1016/j.patcog.2014.01.005

GOOGLE**. O que é o Colaboratory?** 2021. Disponível em: https://colab.research.google.com/notebooks/intro.ipynb?hl=pt_BR#scrollTo=5fCEDCU_qrC0. Acesso em: 03 mar. 2021.

HUDSON, L. **Pyzbar.** 2019. Disponível em: https://pypi.org/project/pyzbar/. Acesso em: 7 ago. 2020.

MELLO, G. C. F. **Detecção e classificação facial em tempos de COVID-19**. 2021. Monografia (Bacharelado em Ciência da Computação) - Universidade Estadual de Londrina, Londrina, 2021.

PEDERNEIRAS, G. **Como funciona o uso de drones na Indústria 4.0**. 2020. Disponível em: https://www.industria40.ind.br/artigo/19377-como-funciona-o-uso-de-drones-na-industria-40. Acesso em: 15 ago. 2020.

REDMON, J. *et al.* You Only Look Once: Unified, Real-Time Object Detection. *In:* IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, Las Vegas. **Proceedings**..., Las Vegas:IEEE, 2016. p. 779-788. Disponível em: https://ieeexplore.ieee.org/document/7780460. Acesso em: 15 ago. 2020. https://doi.org/10.1109/CVPR.2016.91

RODRIGUES, S. F. **Google Colab- Guia do Iniciante**. 2018. Disponível em: https://medium.com/machina-sapiens/google-colab-guia-do-iniciante-334d70aad531. Acesso em: 15 jun. 2021.

SARAIVA, H. P. **Navegação Autônoma, reconhecimento e desvio de obstáculos com drones multirotores**. 2018. Monografia (Graduação Engenharia Elétrica) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2018. Disponível em: https://www.maxwell.vrac.puc-rio.br/34465/34465.PDF. Acesso em: 17 nov. 2020.

SENIOR. **Tecnologia: o** uso de drones na indústria 4.0. 2020. Disponível em: https://www.senior.com.br/blog/tecnologia-o-uso-de-drones-na-industria-4-0. Acesso em: 5 out. 2020.

STROSKI, P N. **O que são graus de liberdade?** 2020. Disponível em: https://www.electricalelibrary.com/2020/03/18/o-que-sao-graus-de-liberdade/. Acesso em: 2 ago. 2020.

TECHZIZOU, T. **YOLOv4 VS YOLOv4-tiny: Training YOLO for Object Detection**. 2020. Disponível em: https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec. Acesso em: 8 jan. 2021.

TURCATO, A. **Visão computacional:** o que é, aplicações e importância. 2019. Disponível em: https://crmpiperun.com/blog/visao-computacional/. Acesso em: 3 ago. 2020.

TZUTALIN, D. **Labelimage.** 2018. Disponível em: https://github.com/tzutalin/labelImg. Acesso em: 6 mar. 2021.

WEDEMANN, K. **O que drones, inteligência artificial e policiamento têm em comum?** 2019. Disponível em: https://canaltech.com.br/inteligencia-artificial/o-que-drones-inteligencia-artificial-e-policiamento-tem-em-comum-144653/. Acesso em: 2 mar. 2021.

ZANGRANDI, A G. M. **Identificação de regiões de texto em jornais históricos Germano-Brasileiros utilizando rede neural YOLO**. 2019. Tese (Doutorado em Ciência da Computação) - Universidade Federal do Paraná, Curitiba Paraná, 2019.