A EMPREGABILIDADE DA INTELIGÊNCIA ARTIFICIAL NA AUTOMAÇÃO DO SETOR LOGÍSTICO PARA CONTROLE DE CARGA

Júlia Magalhães Torres Calixto[1]
Marcelo Silva Corrêa[2]
Marcelo Arantes de Oliveira[3]

Resumo

O setor logístico apresenta demandas complexas, pois envolve uma movimentação grande de pessoas e serviços. Por conta do mercado nacional cada vez mais competitivo e agressivo, a busca pela inclusão digital na gestão das empresas é fundamental para otimizar processos e diminuir a possibilidade de erros humanos. Principalmente na etapa de carregamento durante a contagem de produtos acabados, atividade vital dentro da área que é feita de forma manual e diária que demanda muito tempo. Diante disto, no intuito de resolver problemas relacionados a tomadas de decisões, irregularidades no carregamento e melhora de gerenciamento de processos, este trabalho consiste no desenvolvimento de um sistema de controle de contagem automatizado que utiliza técnicas de *machine learning*, através da linguagem de programação Python, por meio de algoritmos que interpretarão vídeos obtidos por um sistema de câmeras ou por meio de envio de imagens para monitorar, possibilitando um melhor desempenho logístico em processos que antes eram feitos de forma manual.

Palavras-chave: Automatização. Inteligência Artificial. Logística. Contagem. Transporte.

THE EMPLOYABILITY OF ARTIFICIAL INTELLIGENCE IN AUTOMATION OF THE LOGISTICS SECTOR FOR LOAD CONTROL

[1]Bacharel em Sistemas de Informação pelo UGB/FERP.
[2]Bacharel em Sistemas de Informação pelo UGB/FERP.
[3]Especialista em Análise de Sistemas pela UNESA.

Abstract

The logistc sector shows complex demands because envolves a large movement of people and services, and with the national market increasingly agressive and competitive, the search for digital inclusion in business managment is essential to optimize process, decrease possibility of human errors, especially in the load phase during counting of finished products, a vital activity within the area that is is done manually and daily that demands a lot of time. Therefore, in order to solves problems related to decision making, irregularities in loading and improvement of process management, this work consists in the development of an automated counting control system that uses machine learning techniques, through the Python programmin language, using an algorithms that will interpret videos taken by a camera system or by sending images to monitor, enabling better logistical performance in processes that were previously done manually.

Keywords: Automation. Artificial intelligence. Logistics. Score. Transport.

Introdução

Com o mercado nacional e internacional cada vez mais competitivo e agressivo, empresas de todos os segmentos precisam buscar formas de lidar com a concorrência e permanecer comercialmente destacadas perante outras organizações da mesma área de atuação. Desta forma, com a necessidade de se destacar e ter maior visibilidade e preferência do consumidor final, a área de gestão precisa ser alinhada com a de inovação, tendo investimentos e modernização de processos e serviços. Sendo esse tópico de investimento em tecnologia ponto principal do estudo *Future Systems,* realizado pela Accenture*,* no qual mostrou que empresas líderes na integração de tecnologias em seus setores crescem duas vezes mais.

Segundo COYLE *et al*. (2002), a logística integrada, que interliga todo o processo de fabricação de um produto até a distribuição, deve possuir como meta a entrega de valores e satisfação ao cliente final por meio dos serviços que são prestados. Tal área é uma das mais importantes para qualquer organização nos dias de hoje pois é responsável pelo escoamento de mercadorias e insumos por

todo mundo. Assim, com o objetivo de se destacar existe um ponto chave: diminuir os custos e aumentar seus lucros e vendas de produtos junto da melhoria contínua da qualidade e prestação de serviços que vem de encontro posteriormente a sua posição no mercado internacional perante a concorrência.

Com o foco no setor de fabricação e distribuição de bebidas, dentro da área de logística, existe a locomoção do produto acabado (PA) que se dá através do carregamento, etapa no qual é realizada a preparação e separação dos itens para transporte. De acordo com BALLOU (2007), essa atividade deve ser realizada de modo eficiente e eficaz.

Sendo uma das atividades de suma importância dentro da área. E durante o carregamento existe a contagem de carga que é feita antes e depois do descarregamento, visando assegurar que não haja irregularidades e roubos. É uma forma de controle e fiscalização pois pode ocorrer muita das vezes falhas humanas por causa de falta de atenção ou má gestão acarretando desconforto aos clientes e parceiros.

A quantificação e verificação de produtos dentro das empresas é um processo rotineiro, principalmente empresas que possuem setores que entregam produtos para algum cliente final. E diante dessa problemática, o presente trabalho visa desenvolver um sistema de controle e automação que realiza a contagem de PA utilizando técnicas de *machine learning* e inteligência artificial para amenizar o percentual de falhas que ocorrem durante o carregamento, e a possibilidade da geração dados em tempo real para análise de KPI (*key performance Indicator* – indicador-chave de performance), pois, de acordo com FLEURY (2000), existem três razões para argumentar positivamente a eficácia da utilização de informações na criação de sistemas para a área logística, e a segunda razão, menciona a utilização da informação para redução de riscos, proposto pelo atual trabalho.

Referencial Teórico

Desde os primórdios, o conhecimento e a evolução do pensamento do homem se tornaram fatores importantes para o desenvolvimento e continuidade de existência de uma sociedade. O descobrimento do fogo, a invenção da roda, criação de ferramentas hoje consideradas simples, mas que possibilitaram o progresso, o crescimento do saber e da inteligência. A busca por vencer os obstáculos faz com que o homem continue sonhando e desenvolvimento ideias relacionadas a máquinas que pensem e ajam como o ser humano. E com essa ideia a tecnologia avançou consideravelmente.

BELLMAN (1978) define Inteligência Artificial (IA) como a mecanização de tarefas e atividades como tomada de decisão, solução de problemas e aprendizagem. E mesmo com os estudos e debates a respeito de IA terem se iniciado em 1943, ano que McCulloch e Walter Pitts apresentaram seu artigo denominado "*A logical calculus of the ideas immanent in nervous activity*" que menciona pela primeira vez redes neurais, e ano denominado como período de gestação da inteligência artificial, com a criação de redes neurais e o desenvolvimento de jogos de xadrez, hoje diversos fatores como a crescente demanda por informações e o desenvolvimento de computadores mais modernos para as indústrias, permitiram um grande avanço nesse campo.

No âmbito de IA, existem os agentes inteligentes, definidos como entidades que compreendem o mundo que está ao seu redor através de sensores e interagem neste mundo. Para POOLE *et al.* (1998), pode-se definir inteligência artificial como um campo de estudo de agentes inteligentes. Um agente pode ser um ser humano, robô, um termostato, agente de *software*. RUSSEL E NORVIG (2013), pesquisadores renomados no campo de informática, afirmam que é esperado que um agente perceba seu ambiente, se adapte a mudanças e utilizem seus conhecimentos para alcançar objetivos definidos.

Consequentemente, a implementação da IA e agentes inteligentes é fundamental na era da Indústria 4.0, termo apresentado pela primeira vez em 2011,

se referindo à quarta revolução industrial, que se baseia no processo de informatização e digitalização dos negócios fabris utilizando tecnologias emergentes da atualidade (LASI *et al., 2014).* Tecnologias que colaboram com tomadas de decisões e permitem otimizar operações logísticas.

SCHWAB (2018) defende que a Quarta Revolução Industrial é a junção de tecnologias de mundos digitais, físicos e biológicos. Sendo que conforme ele, o que sustenta e evolução da Indústria 4.0 é a velocidade de mudanças e o progresso. Progresso que está cada vez mais brusco no mundo contemporâneo, e mudanças que são necessárias para as empresas permanecerem fortes comercialmente perante o mercado nacional e internacional.

Em convergência com a Indústria 4.0, existe a logística 4.0, termo que segundo FRAGA *et al.* (2016) seria uma adaptação para a área de logística aos requisitos, menções e tecnologias da Indústria 4.0. Em tese, seria a evolução da logística tradicional, não é apenas uma expressão nova, e sim, um novo patamar da área que engloba conexões inteligentes, automação e integração de processos em tempo real. Conforme FRAGA *et al.* (2016), existem alguns pontos que a tecnologia pode contribuir com a área de logística: redução de perda de ativos, gerenciamento de estoques, garantia de eficiência de forma geral. O foco recai sobre manter a velocidade e a conexão. É possível afirmar que para essas coisas serem proporcionadas, a utilização de algumas soluções como computação em nuvem, *big data,* internet das coisas (IOT), visão computacional e aprendizado de máquina é fundamental e são grandes protagonistas da atual revolução promovendo também vantagens competitivas.

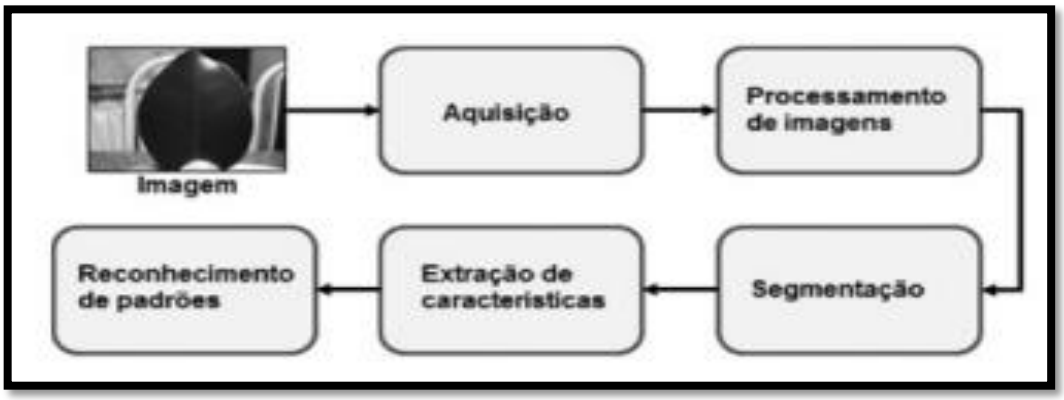
Dentro das organizações, quando se trata de gerenciamento de estoque e contagem de produtos, uma boa gestão é essencial. Saber quanto estocar, qual produto vai vencer e quanto e qual produto transportar se torna de grande importância e agrega valor à empresa. Desta forma, a área de inovação deve ser integrada, a implementação de novas tecnologias pode contribuir com o aumento da qualidade e oferecer benefício a todos. A diminuição de chance de erros e a garantia de eficiência e rapidez do processo se tornam realidade com a instalação

de *softwares* inteligentes que realizam operações como contagem, recebendo as informações por intermédio um sistema de visão, definido por STAIR e REYNOLDS (2006), como um sistema que permite que os robôs, ou outros equipamentos, recebam, processem e armazenem imagens virtuais e assim, disponibilizem relatórios em tempo real para gestores.

Visão Computacional

DAWSON (2014) menciona que o termo visão computacional pode ser atribuído à capacidade da realização de análises de vídeos e fotos por computadores no intuído de obter informações. A visão computacional, que teve início de suas pesquisas na década de cinquenta com os estudos de modelos estocásticos feitos por Ballard e Brown e publicado na obra *Computer Vision* no ano de 1982, é uma ciência que segundo os autores, estuda e desenvolve tecnologias que permitam que máquinas enxerguem e extraiam informações através de sensores. A Figura 1 ilustra as fases de um sistema de visão computacional proposto por BACKES e JUNIOR (2019). Um sistema de visão computacional apresenta um conjunto de várias fases que etapas predefinidas que são: aquisição, processamento, segmentação, extração e reconhecimento de padrões.

Figura 1**.** Etapas de um sistema de visão computacional.



Fonte: BACKES e JUNIOR (2019).

A aquisição é a primeira etapa de um sistema de processamento digital de imagens e as etapas seguintes dependem da realização dela. Nessa fase os dispositivos e sensores capturam imagens ou vídeos. Seguindo da etapa de processamento onde são aplicados recursos de pré-processamento para facilitar a leitura pelo modelo de visão computacional conforme BACKES e JUNIOR (2019), Figura 2.

Figura 2. Diferença entre uma imagem com ruído e após a filtragem do ruído.

Fonte: MAREGONI e STRINGHINI (2009).

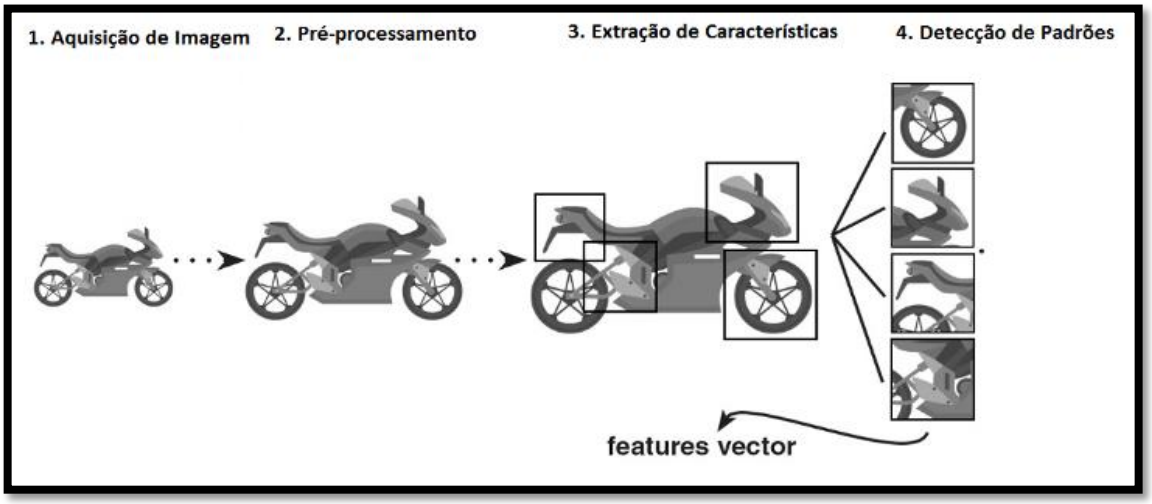
Segundo BACKES e JUNIOR (2019) a segmentação está ligada ao momento de fragmentação da imagem no qual os itens relevantes são separados. Com isso, após a etapa de segmentação a extração de característica encontra propriedades específicas daquele objeto como cor, textura, área. E após isso, como última etapa, o objeto é classificado ou agrupado de acordo com suas características conforme ilustrado na Figura 3.

A visão computacional é um tema que está em ascensão e ponto chave para o desenvolvimento de ferramentas que solucionem problemas que surgem mediante as necessidades da Indústria 4.0 com a criação de aplicações entre elas: câmeras inteligentes, realidade aumentada, transportes autônomos e controle de

qualidade industrial. Sendo assim, cada vez mais surgem novas bibliotecas e linguagens de programação que tornam o reconhecimento de padrões mais fáceis e ágeis como é o caso da biblioteca OpenCV.que foi desenvolvida para facilitar a aplicação de técnicas de processamento de imagem.

As bibliotecas que fazem parte do OpenCV contam com mais de dois mil algoritmos otimizados que são criados, segundo BARELLI (2018) para reconhecimento de faces, identificação de objetos e outros suportes relacionados à visão computacional. Sendo ele suportado pelas linguagens de programação: Python, Java, C++ e podendo ser suportado pelos sistemas operacionais Linux, Mac, Windows e Android.

Figura 3. Fluxo do processo de visão computacional.



Fonte: SUZANA VIANA (2020).

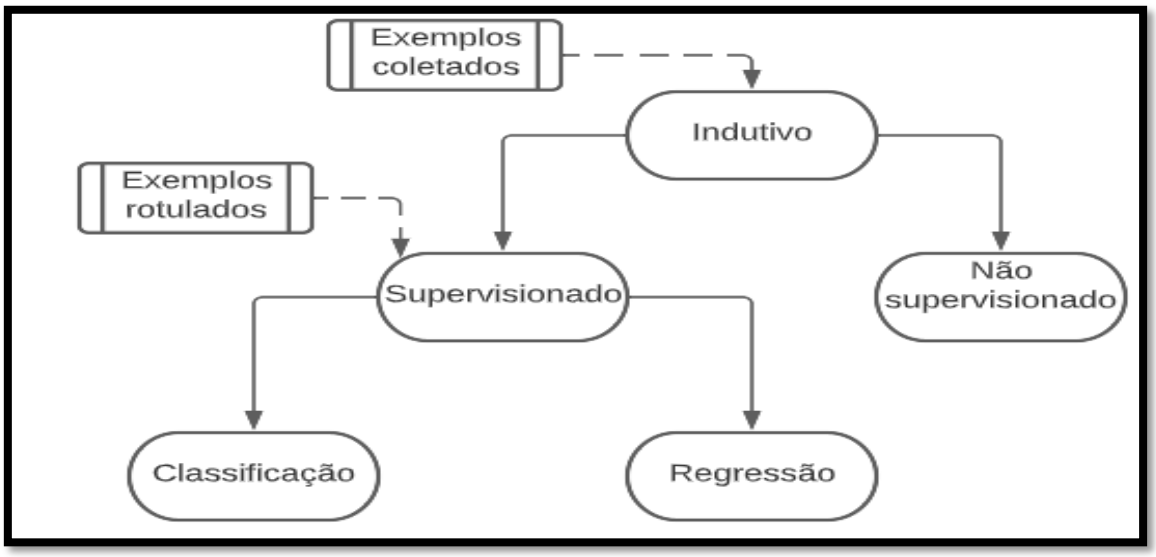
Aprendizado de Máquina

O aprendizado de máquina dentro do âmbito empresarial está cada vez mais presente e com isso tal tecnologia permitiu que empresas executassem tarefas de uma forma antes não pensada gerando produtividade e oportunidades de negócio para as empresas.

Segundo REZENDE (2003), o aprendizado de máquina pode ser definido como uma área relacionada a inteligência artificial que tem como objetivo o desenvolvimento de técnicas computacionais relacionadas ao aprendizado de forma a adquirir conhecimento de forma automática. Sendo um sistema baseado em aprendizado de máquina, um programa que toma decisões baseadas em experiência bem-sucedidas anteriormente.

O aprendizado de máquina segundo MONARD e BARANAUSKAS (2003) diz respeito a um aprendizado indutivo que diz respeito a um conjunto de dados que podem ser realizadas descobertas ou generalização para se ter um desfecho. Sendo esse aprendizado dividido em duas categorias: supervisionado e não supervisionado e possui ramificações conforme visto na Figura 4.

Figura 4. Aprendizado indutivo adaptado de MONARD e BARANAUSKAS(2003).



Fonte: Adaptação de MONARD e BARANAUSKAS (2003) elaborado pelos autores.

De acordo com RUSSEL e NORVIG (2014), um treinamento supervisionado por der definido como o agente observando um conjunto exemplos de pares de entrada e saída e acaba aprendendo uma função que faz o mapeamento da entrada para a saída. Tais algoritmos relacionam uma saída com uma entrada com base

em dados rotulado. Para cada saída é atribuído um rótulo, uma classe. Assim, é possível treinar uma rede neural para classificar automaticamente a saída dos dados. Quando um conjunto é definido pelos seus rótulos pré-definidos pode-se chamar o algoritmo de algoritmo de classificação. Desta forma, uma classe de produtos por exemplo pode ser treinada previamente neste algoritmo e assim que o algoritmo for executado novamente e o usuário enviar imagens que já estão pré-definidas, o algoritmo consegue definir qual a classe de produtos que o usuário está enviando conforme ilustrado na Figura 5.

Figura 5. Reconhecimento de produto utilizando aprendizado de máquina.

Fonte: Pesquisa dos Autores.

No treinamento de conjunto não supervisionado não é atribuído um rótulo para os dados de saída. Os algoritmos buscam com base no vasto número de dados, padrões e similaridades entre eles para se agrupar ao identificar os grupos. RUSSEL e NORVIG (2014) afirmam que o agente aprende os padrões de entrada embora não contenha nenhum feedback.

Para RUSSEL e NORVIG (1995) existem três razões para construir um algoritmo que aprenda, sendo a primeira, o fato de o programador não conseguir pensar em todas as prováveis situações que podem ocorrer com o agente inteligente. A segunda, que existe a necessidade constante de adaptação a mudanças e a última, nem todos os projetistas de *software* sabem qual ideia de solução será programada.

Materiais e Métodos

O projeto é caracterizado por um estudo exploratório e levantamento bibliográfico tendo como base revistas científicas disponíveis on-line a respeito do tema: inteligência artificial e logística integrada. Com base nos levantamentos bibliográficos, antes do desenvolvimento do *software* proposto, foi escolhida a metodologia ágil Kanban, criada na década de 40 pela Toyota.

Metodologia ágil pode ser definido como um conjunto de práticas para atender demandas de um projeto de forma eficiente. Para CAMARA (2005), metodologias ágeis buscam desenvolver *softwares* mais rapidamente e sem muita burocracia, por isso foi optado por seguir esse modelo. Sendo que a ferramenta utilizada foi o Trello, *software* que integra o quadro do Kanban em seus recursos e uma ferramenta de cronograma. O cronograma foi pensado e dividido em duas etapas: prazos de elaboração escrita do projeto e desenvolvimento.

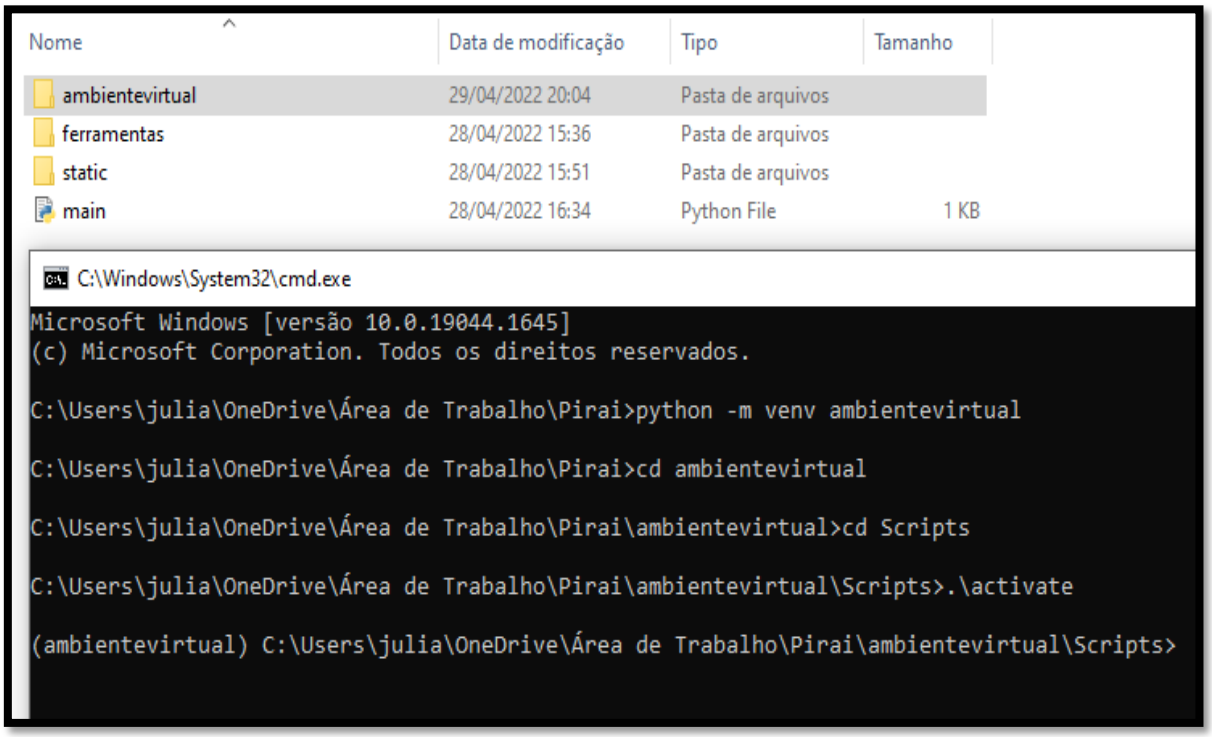
Para o esboço, prototipação inicial do *layout* do projeto e *mockups* da plataforma, ferramentas como a plataforma de design de interfaces gratuita Figma e o editor de imagens livre GIMP foram empregadas no intuito de proporcionar uma interface organizada e acessível. Oferecer uma boa experiência para o usuário é fundamental e deve ser pensado desde o início da confecção do projeto. Uma boa interação com os elementos da página evita frustrações e permite que os usuários se sintam bem ao utilizar.

Na questão do desenvolvimento do *software* foi necessário uma busca por tecnologias que estão em destaque no mercado quando se fala de visão computacional, pois o foco do projeto é voltado para o reconhecimento e contagem de produtos. A visão computacional busca replicar a visão humana utilizando *softwares* e *hardwares*. As imagens entram, são processadas e possuem um conjunto como objetos ou vetores como saída. Após pesquisas, *a* linguagem de programação Python, criada em 1991 por Guido van Rossum, foi considerada uma boa opção dentro desta área, se destacando por conta da sintaxe simples e

legibilidade. Foi utilizado sua biblioteca OpenCV, biblioteca gratuita de visão computacional que é normalmente empregada em análises de imagens e vídeos, possibilitando também tratamentos de cor.

Foi configurado um ambiente virtual que empacota todas as dependências que são utilizas dentro de um projeto e armazena em uma pasta. Assim, nenhuma dependência é instalada no sistema operacional do desenvolvedor e é possível puxar versões específicas de cada pacote sem impactar muito o sistema. A instalação e configuração do ambiente é exemplificada na figura 6. Todas as dependências utilizadas no projeto foram instaladas neste ambiente.

Figura 6. Criação do ambiente virtual e ativação.



Fonte: Pesquisa dos Autores.

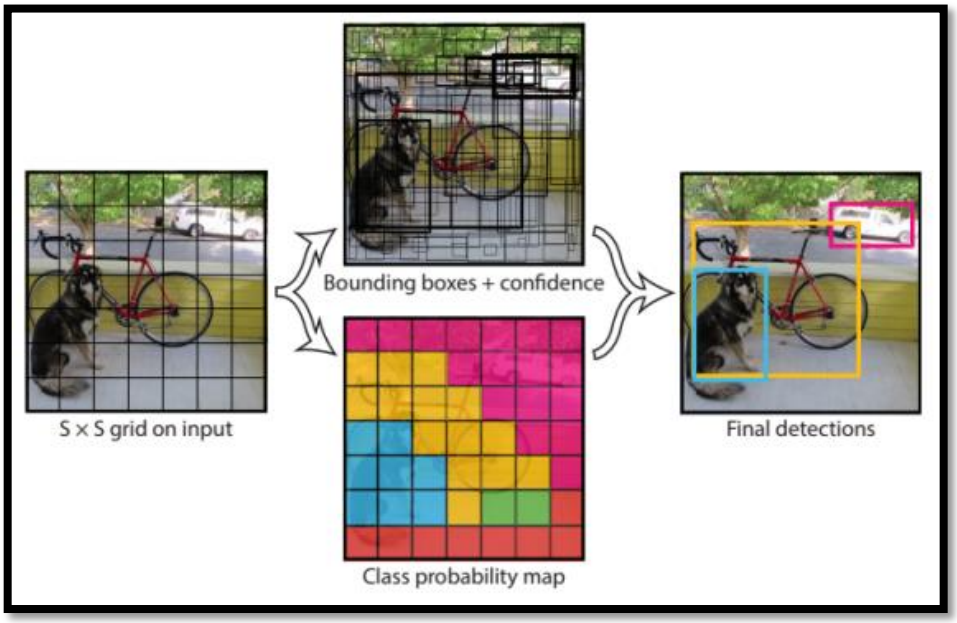
Para tornar possível a execução do projeto foi necessário conectar câmeras IPs ao computador. A conexão foi realizada via protocolo RTSP, sigla para *Real*

Time streaming Protocol (Protocolo com fluxo em tempo real), que torna possível a transferência de dados como áudio e vídeo em tempo real.

Após a instalação do ambiente virtual e procurar por uma câmera adequada ao protótipo do projeto, partiu-se para a instalação do OpenCV que conectaria a câmera e faria todo tratamento de imagem e reconhecimento. Inicialmente uma função foi criada para receber imagens, o OpenCV integrado nessa função aplica filtros cinzas e deixa a imagem mais fácil de ser lida. Após isso, em junção com a biblioteca YOLO (*you only look once)* que detecta objetos personalizados.

Celulares, imagens, faces de pessoas são exemplos de objetos que podem ser detectados. Com isso, os objetos das imagens detectados são contados e passam para uma base de dados. Um dos principais pontos da YOLO é que cada caixa que fica ao redor do objeto detectado precisa passar por etapas: a YOLO divide primeiramente a imagem em uma grade onde após isso cada quadrante recebe uma quantidade de caixas delimitadoras e após isso uma nota de confiabilidade para afirmar o que realmente é aquele objeto. Ver Figura 7.

Figura 7. Estrutura de funcionamento da YOLO.



Fonte: YOLO (2016).

Tal arquitetura ganhou bastante força em suas versões posteriores e sua grande vantagem é a velocidade de detecção, contudo, durante sua aplicação é fundamental aplicar o processo de *non-maximal supression,* responsável por ignorar detecções semelhantes. O famoso *dataset* da Microsoft COCO (Microsoft *Common Objects in Context)* foi utilizado para o treinamento do algoritmo por possibilitar a classificação de imagens e gerar um arquivo que será integrado com a YOLO e o OpenCV. Dentro do aprendizado supervisionado existe o segmento da classificação que de forma geral é utilizada quando as instâncias possuem classes já conhecidas e associadas com rótulos.

Assim, o primeiro passo do processo para criação de um modelo de treinamento é a obtenção de um conjunto de imagens. Para isso foram retiradas mais de 6 mil fotos a partir de frames de gravação com o algoritmo mostrado na Figura 8 e o resultado da coleta conforme a Figura 9. Na função “capFrameVideo” é passado um vídeo que é lido pela classe “VideoCapture” em diversos *frames.* Assim, na medida que foi ocorrendo o *loop,* as imagens eram salvas incrementalmente dentro de uma pasta

Figura 8. Algoritmo para coletar as imagens em loop

```
import cv2

pathSave = "./static/img/"
video = "static/video/teste2.mp4"

def capFrameVideo(video):
    cap = cv2.VideoCapture(video)
    number = 0

    if cap.isOpened():
        capture,img = cap.read()

        while capture:
            capture,img = cap.read()
            cv2.imshow("Janela", img)
            number += 1

            if cv2.waitKey(5) == 27:
                break

            cv2.imwrite(pathSave + "linha%d.jpg" % number, img)

    cap.release()
    cv2.destroyAllWindows()

capFrameVideo(video)
```

Fonte: Pesquisa dos Autores.

Figura 9. Coleta de imagens frame a frame.

Fonte: Pesquisa dos Autores.

Com isso, o próximo passo foi a rotulação das imagens utilizando a biblioteca “labelimg” do Python que foi instalada no ambiente virtual. É realizada a detecção das regiões de interesse que receberam as *labels* uma por uma conforme a Figura 10 demonstra. Como ferramenta de anotações de imagem gráficas, as anotações feitas na “labelimg” podem ser salvas em arquivos xml, PASCAL VOC, YOLO e CreateML. Após o salvamento, foi criado um arquivo chamado “classes.txt” na pasta que possuía as imagens. Tal arquivo define uma lista de nomes de classes aos quais os rótulos se referem.

Figura 10. Criação de rótulos usando LabelImg.

Fonte: Pesquisa dos Autores.

Depois da criação de rotulação e posteriormente do treinamento foi necessário o desenvolvimento de um processo de aprendizado profundo chamado *object tracking* (rastreamento de objeto) que rastreia o movimento de um objeto e estimula posições que ele vai assumir quando está em movimento em um vídeo. Ele atribui uma identificação única (ID) para cada objeto e observa o movimento dele em cada *frame* do vídeo, assim, evitando a detecção e contagem de um objeto de forma repetida mediante o exemplo ilustrado na Figura 11.

Figura 11. Exemplo de rastreamento do objeto em movimento e contagem.

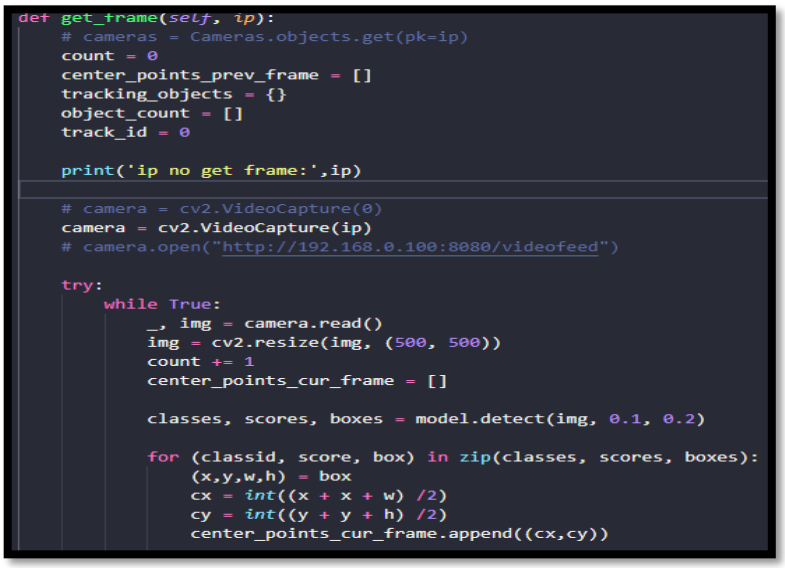
Fonte: Pesquisa dos Autores.

A função “get_frame” apresentada na Figura 12 está recebendo como parâmetro os endereços IPs das câmeras que foram configuradas na aplicação. Sendo essa configuração feita através da criação de uma tabela no *models* do *framework* Django, *framework web* Python de código aberto com *design* limpo e de fácil curva de aprendizado.

Para a conexão de múltiplas câmeras foi necessário desenvolver um processo de *multi-threading* ilustrado na Figura 13 para a leitura de vários quadros de uma vez só de forma que não é necessário ser sequencial como um processo normal. Enquanto um thread vai executando uma tarefa de leitura do próximo *frame*

do vídeo, o outro vai executando. A função “dynamic_stream” recebe os IPs que estão na tabela “Cameras” do *model* do Django e cria várias *urls* com a descrição fornecida também na tabela. Assim, além do processamento acontecer em *threading*, é possível abrir cada câmera individualmente no “html” da página.

Figura 12. Função get_frame ligada ao rastreamento do objeto.



```
def get_frame(self, ip):
    # cameras = Cameras.objects.get(pk=ip)
    count = 0
    center_points_prev_frame = []
    tracking_objects = {}
    object_count = []
    track_id = 0

    print('ip no get frame:',ip)

    # camera = cv2.VideoCapture(0)
    camera = cv2.VideoCapture(ip)
    # camera.open("http://192.168.0.100:8080/videofeed")

    try:
        while True:
            _, img = camera.read()
            img = cv2.resize(img, (500, 500))
            count += 1
            center_points_cur_frame = []

            classes, scores, boxes = model.detect(img, 0.1, 0.2)

            for (classid, score, box) in zip(classes, scores, boxes):
                (x,y,w,h) = box
                cx = int((x + x + w) /2)
                cy = int((y + y + h) /2)
                center_points_cur_frame.append((cx,cy))
```

Fonte: Pesquisa dos Autores

Figura 13. Processo de *threading* na função dynamic_stream.

```
def dynamic_stream(request, descricao):

    results = []
    threads = {}
    ips = {}

    cameras = Cameras.objects.all()
    itera = iteraip(cameras)

    for i in range(cameras.count()):
        seila = next(itera)
        t = Thread(target = get_frame, args=(seila.ip, results))
        threads[seila.descricao] = t
        ips[seila.descricao] = seila.ip
        t.start()

    i = threads[descricao]
    i.join()

    return StreamingHttpResponse(get_frame(i, ips[descricao]),content_type="multipart/x-mixed-replace;boundary=frame")
```

Fonte: Pesquisa dos Autores.

A aplicação foi separada em dois módulos de reconhecimento: por vídeo ao vivo e por envio de imagens. Sendo o processo de reconhecimento de imagem feito de forma diferente do vídeo em tempo real. Com as dependências da biblioteca OpenCV devidamente instaladas no ambiente virtual, foi criada uma função que recebe imagens e a classifica conforme é reconhecida. Caso o produto enviado seja algo que não está na base de dados da classificação, o produto aparece com o nome “desconhecido” conforme é ilustrado o código da Figura 14. Sendo que para a classe ser apresentada, ela é convertida para a coloração cinza utilizando o comando “cv2.COLOR_BGR2GRAY”.

Figura 14. Conversão para cinza e reconhecimento do produto.

```python
def detection_product(image):
    with open("names.names", "r") as f:
        class_names = [cname.strip() for cname in f.readlines()]
    recebimento = cv2.imread(f"media\count\{image}")
    cap = cv2.cvtColor(recebimento, cv2.COLOR_BGR2GRAY)

    modelWeightsPath = "counteyeapi\services\detection\yolov3_training_last.weights"
    modelConfigurationPath = "counteyeapi\services\detection\yolov3_testing.cfg"

    net = cv2.dnn.readNet(modelConfigurationPath, modelWeightsPath)
    model = cv2.dnn_DetectionModel(net)
    model.setInputParams(size = (416, 416), scale=1/255)


    predict = {
        "class": "Desconhecido"
    }
    try:
        classes, scores, boxes = model.detect(cap, 0.1, 0.2)
        for (classid, score, box) in zip(classes, scores, boxes):
            predict = {
                "class": class_names[0],
                "accuracy": score
            }
    except Exception as ex:
        pass

    return predict
```

Fonte: Pesquisa dos Autores.

Para a realização da contagem dos produtos outra função foi desenvolvida. Ela recebe a imagem que passa pela detecção e aplica a função “cv2.drawCounts” que tem na biblioteca OpenCV e aplica contorno ao redor da imagem que é

detectada e assim é possível se contar quantos objetos tem na imagem. Ambas as funções são chamadas na *views* do Django e são salvas com o método post, método de requisição suportado pelo protocolo HTTP, dentro da API (*Application Programming Interface)* desenvolvida para a aplicação.

Após todos os processos treinados e implementados em uma única câmera, parte-se para a proposta de conectar mais de um display de vídeo. Para a disponibilização de informações à página *web* foi necessário o desenvolvimento de uma API no modelo REST/FULL utilizando o framework Django Rest Framework, possibilitando então consultas na base de dados. Em relação ao armazenamento de informações, foi utilizado o banco de dados postgresql pela alta performance e baixa curva de aprendizagem por se tratar de um sistema relacional e ser gratuito que executa suas operações por comandos mais simples.

Neste projeto, a nível de *front-end*, parte da aplicação que interage diretamente com o usuário, foi desenvolvida uma página *web* programada em ReactJS, biblioteca em Javascript*,* criada em 2011 pelo Facebook, popularmente conhecida para criação de interfaces de usuário. A biblioteca foi escolhida por ser flexível e por permitir que os componentes em sua aplicação sejam reutilizados. Sendo que se pode definir como componentes uma função em Javascript que permite a divisão da interface em partes independentes. É possível utilizar um mesmo componente em outros lugares da aplicação.

Para a disponibilização da API do produto, foi utilizado um servidor Linux provido e gerenciado pela Amazon Lightsail que disponibiliza serviços em computação em nuvem e hospedagem de aplicações multiplataforma em diversas linguagens de programação. E para a hospedagem da aplicação *front-end* a plataforma Vercel[4] foi utilizada por ser gratuita e fácil de ser utilizada. Ambos os códigos foram organizados em *branches* de repositórios separados e privados na

[4] Disponível em: <https://counteye.vercel.app/> Acesso em: 27. maio 2022.

plataforma GitHub[5] para uma melhor organização e versionamento do código do produto.

Resultados

Com base no estudo levantado sobre a área de logística e a respeito da integração da inteligência artificial para gestão de empresas, foi desenvolvido um protótipo de *site* intitulado CountEye possuindo como principal função o reconhecimento e a quantificação de produtos acabados, disponibilizando os resultados em tempo real através de *dashboards* disponíveis na página. A aplicação inicialmente possui como público-alvo médias e grandes empresas do setor alimentício e de bebidas.

Para a construção da interface do sistema foi levado em consideração alguns princípios de usabilidade como: visibilidade, prevenção de erros, estética padrão e suporte ao usuário. Após o usuário acessar a *Landing Page* (Figura 15) e realizar o *login,* sendo que só são permitidos usuários que se cadastraram no sistema, ele será redirecionado a página inicial da aplicação que é composta por algumas guias. A página de “painel de controle” apresenta as últimas imagens ou gravações enviadas que são salvas no banco. Abaixo desse histórico breve de envios, existe uma área que é preenchida apenas quando se envia uma imagem após clicar no ícone de adicionar, localizado no canto inferior do meio, conforme demonstrado na Figura 16.

[5] Disponível em: <https://github.com/marcelo-s-correa/counteye_front> Acesso em: 27. mai. 2022.

Figura 15. Protótipo *Landing Page* CountEye.

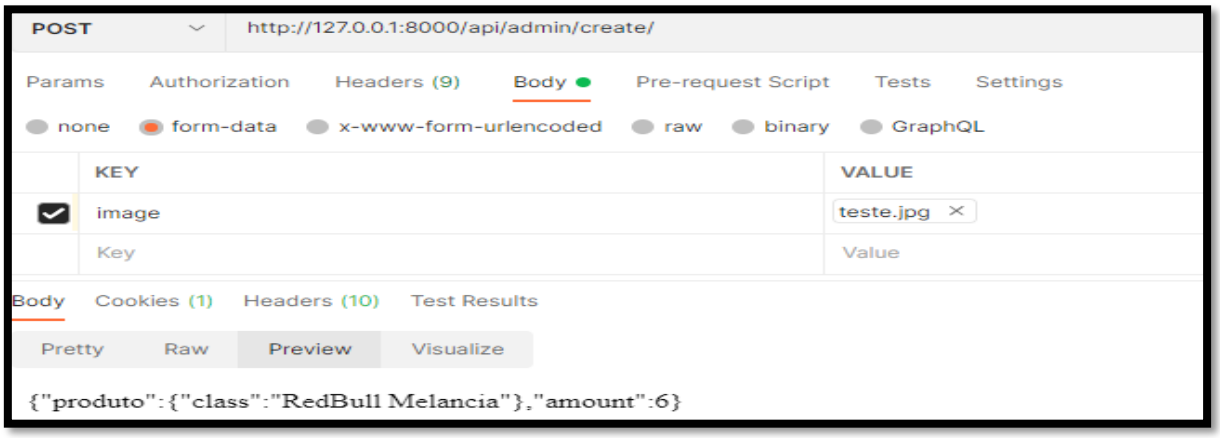
Fonte: Pesquisa dos Autores.

Figura 16. Protótipo do *dashboard* para enviar imagens.

Fonte: Pesquisa dos Autores.

Desta forma, ao selecionar a imagem ou vídeo e clicar em enviar, o painel é preenchido automaticamente com a detecção, informando qual o produto, e contagem. O primeiro teste com o programa em execução foi o reconhecimento de um energético: RedBull de Melancia que foi catalogado no dicionário de objetos do treinamento conforme exemplificado na Figura 17 com um teste sendo realizado no Postman, plataforma de teste e construção de API. O programa se mantém estável e consegue identificar o produto e contar adequadamente eles. Sendo possível identificar mais de um objeto por vez.

Figura 17. Teste sendo realizado no Postman.



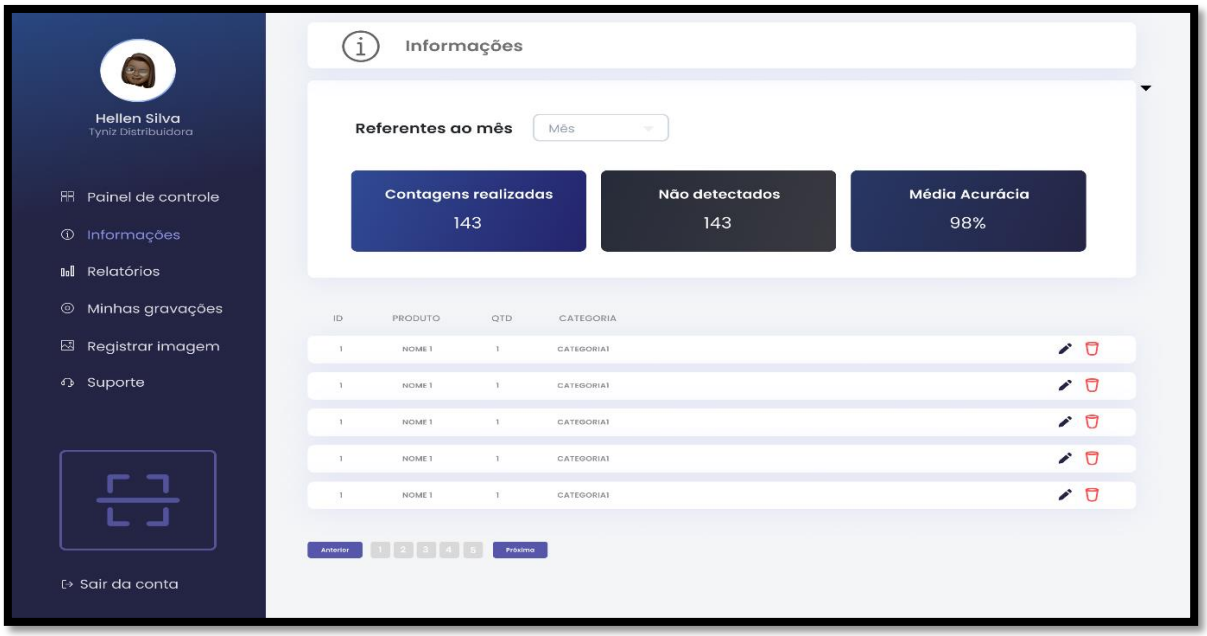
Fonte: Pesquisa dos Autores.

Foi necessário se pensar sobre as diversas variações de luminosidade que o objeto pode sofrer e assim impactar de forma negativa a identificação. Sendo necessário ser adicionado algumas imagens durante o treinamento que tentam simular como o objeto pode ficar nesses ambientes. É possível identificar que a iluminação do ambiente pode se tornar um fator de interferência no reconhecimento dos objetos. Sendo a ausência parcial ou total de luz, mais difícil do algoritmo detectar os objetos que estão na imagem/vídeo.

Destaca-se a importância da utilização de câmeras com uma boa resolução em HD para cima para captura de imagens no módulo de vídeo ao vivo. Câmeras com maior capacidade e qualidade podem impactar de forma positiva e trazer resultados significativos ao projeto. Dando sequência, foi verificado que na página de administrador do Django estava exibindo todos os envios de imagens e se o autor que estava realizando estava sendo exibido. Pois, a aplicação exibe apenas as informações pertinentes a cada usuário. A API foi configurada com permissões de usuário para poder realizar as operações de CRUD (*create, read, update* e *delete*). Além dos últimos envios aparecendo na página do administrador e no retorno da API, assim apresentado na Figura 18, foi desenvolvido um estoque

online que a cada envio de imagens, é acrescentado no somatório do estoque que é exibido na página *web* desenvolvida.

Figura 18. Protótipo da página de relatório.

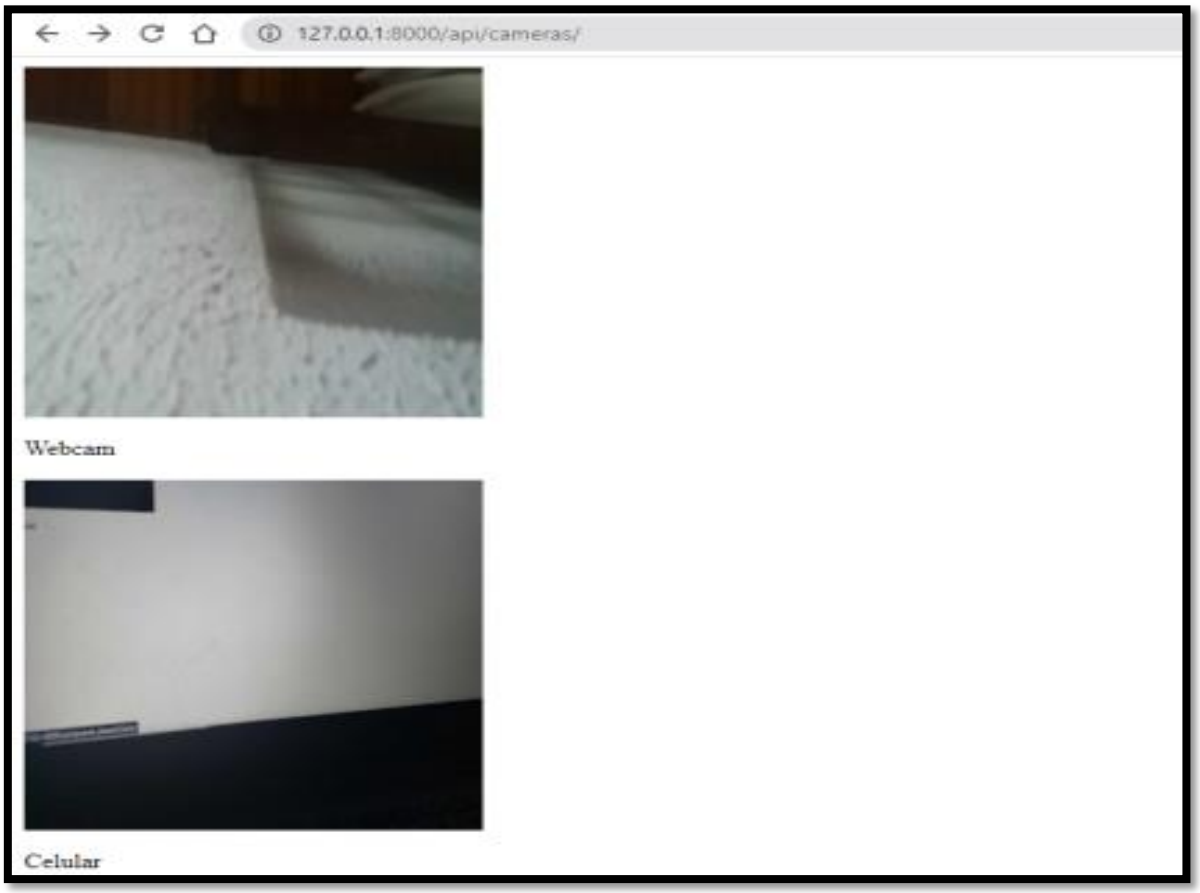


Fonte: Pesquisa dos Autores.

Nota-se que existe uma visualização personalizada de foto de perfil e nome dentro do sistema, tornando a interface atrativa a cada usuário. Na aba “informações” é possível observar os últimos envios e realizar operações neles. Existe a possibilidade de baixar relatórios antigos em formato de *excel* para exportar, assim como em “minhas gravações” que fica registrado apenas as imagens e vídeos enviados dentro da plataforma. Sendo que não existe apenas a possibilidade de enviar vídeos ou imagens na plataforma.

O CountEye, forma que foi nomeada o site, consegue fazer detecções ao clicando no ícone de *scanner* localizado no canto inferior esquerdo, isso em virtude da biblioteca de visão computacional OpenCV, utilizada no *back-end* do projeto para a captura das imagens. A Figura 19 demonstra o teste de conexão de mais de uma câmera no *back-end.*

Figura 19. Conexão de mais de uma câmera no *back-end*.



Fonte: Pesquisa dos Autores.

Para conectar a câmera na aplicação foi necessário utilizar a *tag* “iFrame” que é uma *tag* HTML utilizada para inserção de conteúdos externos em uma página *web*. Desta forma é possível carregar outra página sem a necessidade de abrir uma nova janela, ela é carregada dentro de outro ambiente sem comprometer os elementos de ambas as páginas.

Discussão

Com o cenário de competitividade acirrada no mercado empresarial, detalhes mínimos podem se tornar um diferencial garantindo uma posição de destaque perante os concorrentes, e a área da logística buscando inovação agrega grandes conquistas e valores à empresa. Assim, diante dos resultados obtidos com a elaboração da aplicação *CountEye*, foi possível afirmar que o desafio de ir além de uma simples plataforma construída para o setor logístico foi alcançado.

A possibilidade de recriar a forma de como o ser humano pensa e analisa dados, prevendo e evitando falhas, torna a inteligência artificial uma poderosa colaboradora em ferramentas que buscam o ganho de produtividade e redução de erros como o *CountEye* se propôs a fazer de forma simples e intuitiva. Sendo suficiente apenas o envio de imagens ou conexão com uma câmera para a realização de contagens automáticas feita pelo sistema. Ação que viabiliza a capacidade das empresas se destacarem e estarem um passo à frente das concorrentes, como também, permitindo o ganho de produtividade e informações em tempo real. É então importante destacar o diferencial do projeto: a criação de um espaço para atender as dificuldades de uma determinada área que demanda de inovações cada vez mais e necessita de ferramentas para análises de dados precisas e redução de falhas humanas e extravios de produto.

Vale ressaltar que dentro do escopo do projeto e diante do avanço e ascensão da era digital, se observa a necessidade da criação de interfaces intuitivas, acessíveis e práticas que engajem o consumidor. Ocorra a entrega de valor para o cliente. Assim, com a intensa circulação de informações por diversos meios, sendo elas por meio de linguagem textual e visual. Porém, isolados não representam grandes coisas. Precisam estar definidos em um contexto. Pontos, linhas, formas, texturas, cores, principais elementos que influenciam o ser humano e provocam sensações. A aplicação *CountEye* relaciona esses elementos em sua interface e proporciona uma forma de comunicação simplificada, ponto essencial dentro do projeto que foi proposto.

Entretanto, o projeto apresenta algumas limitações e alguns pontos foram observados durante seu desenvolvimento como o processo de treinamento dos modelos demorados e *hardwares* que suportem o sistema proposto.

Mesmo com a facilidade de desenvolvimento do treinamento, ele foi um processo demorado e repetitivo. Iniciando pelo processo de coleta de imagens para o treinamento, foram coletadas mais de seis mil imagens que receberam rótulos, uma por uma, utilizando a biblioteca *LabelImg* do Python que gerou um arquivo de texto que posteriormente passou por um algoritmo de treinamento que posteriormente se converteu em pesos.

No processo seguinte as imagens tiveram que ser zipadas em um arquivo para serem treinadas. Esta etapa no primeiro teste durou mais de 5 horas para gerar um peso. Sendo passados apenas alguns arquivos e não os 6 mil. Acredita-se que a demora foi causada pelo tamanho das imagens enviadas e pela complexidade do algoritmo de treinamento. Assim, caso outro produto precise ser registrado, terá que passar por essas etapas novamente que demanda tempo e espaço. O código do projeto é escalável e mutável, sendo possível acrescentar mesmo assim outros produtos.

A visão computacional é uma alavanca da inovação e proporciona diversos benefícios para diversos ramos de indústria, porém existe o ponto de limitações de *hardware* e não tecnologias em projetos ligados a visão computacional. Pois, em processamentos digitais de imagem a qualidade que esses *hardwares,* tal como câmeras, capturam as imagens precisam ser capazes de suportar as diversas necessidades do sistema como uma captura clara do ambiente que ela está instalada.

Considerações Finais

O presente trabalho teve como motivação identificar e controlar a quantidade de produtos que são carregados pela área de logística no intuito de reduzir extravios e evitar erros de contagem que ocorrem durante a etapa de carregamento. Compreender como funciona o carregamento, traçar os principais pontos críticos do processo de contagem foram alguns pontos investigados para a criação de uma página *web* denominada *CountEye* que utiliza inteligência artificial para realizar contagem de forma automática.

A proposta de se utilizar a inteligência artificial como auxiliar veio de seu crescimento e aperfeiçoamento nos últimos tempos e por estar cada vez mais ela está presente no cotidiano das pessoas e indústrias. A automatização de atividades permite a agilidade, produtividade e redução de erros, que são benefícios tanto para o empregado quanto para o empregador.

Com a questão de organizações possuírem diversos dados bruto, a aplicação promove a transformação de dados em informação em tempo real através de *dashboards*, o que irá auxiliar em tomadas de decisão mais inteligentes e assertivas por parte dos gestores. Utilizando-se dos conceitos de inteligência artificial e análise de dados foi possível criar uma forte ferramenta que promove um ambiente limpo, fácil e realmente útil aos usuários, que por sua vez, empresas e organizações ao usarem a plataforma, poderão estar um passo à frente de seus concorrentes. A utilização de técnicas inovadoras permite que as empresas se destaquem no mercado e melhorem o trabalho cada vez mais pensando no cliente final.

É de se recomendar para trabalhos futuros o teste e implementação do projeto em outras áreas sem ser da logística. Sendo o sistema reformulado para detectar e contar outros tipos de objeto como por exemplo, se em determinado ambiente quantas pessoas estão utilizando máscara em um ambiente fechado.
Assim, conclui-se que os objetivos específicos do trabalho foram alcançados e as tecnologias utilizadas durante a confecção em acordo com as especificações para

fazer do *CountEye* um vetor de transformação de processos antes feitos manuais, no mundo da logística.

Referências

BACKES, André Ricardo, JUNIOR, Jarbas Joaci e Mesquita Sá Junior. **Introdução à visão computacional usando matlab**. Alta Books, 2019.

BALLARD, Dana H., BROWN, Christopher M. ***Computer Vision***. *Prentice Hall*, 1982.

BALLOU, Ronald H. **Logística Empresarial:** transportes, administração de materiais, distribuição física. São Paulo: Atlas, 2007.

BARELLI, Felipe. **Introdução à Visão Computacional:** Uma abordagem prática com Python e *opencv*. Casa do Código, 2018.

BELLMAN, Richard E. **Na Introduction to Artifical Intelligence:** *Can Computers Think?* San Francisco: Boyd & Fraser Pub. Co., 1978.

CAMARA, Fabio. **Processos Ágeis e MSF**. 2005. Disponível em: http://www.linhadecodigo.com.br/artigo/833/processos-ageis-e-msf.aspx. Acesso em: 05 nov. 2021.

COYLE, J. J. et al. ***The management of business logistics:*** *a supply chain perspective*. 7. ed. p. cm. Thomson Learning, 2002.

DAWSON, Kenneth. ***A Practical Introduction to Computer Vision with opencv***. 1. ed. Wiley, 2014.

FLEURY, Paulo F. Logística Integrada. In: FLEURY, P.F., FIGUEIREDO, K., WANKE, P. (org.). **Logística Empresarial**. Coleção COPPEAD de Administração. Atlas: São Paulo, 2000.

FRAGA, Manoela, FREITAS, Matheus, SOUZA, Gilson. **Logística 4.0:** conceitos e aplicabilidade – uma pesquisa-ação em uma empresa de tecnologia para o mercado automobilístico. Caderno PAIC, v. 17 n. 1, 2016.

GHOSH, Bhaskar, WILSON, James, DAUGHERTY, Paul. ***Your Legacy or your legend?*** 1. ed. Accenture, 2020.

LASI, Heiner et al. **Industry 4.0**. 4. ed. Business & information systems engineering, 2014.

MARENGONI, Maurício, STRINGHINI, D. **Tutorial: Introdução à Visão Computacional usando opencv**. 16. ed. Revista de Informática Teórica e Aplicada, 2009.

MCCULLOCH, Warren S., PITTS, Walter. **A logical calculus of the ideas immanent in nervous activity**. 5. ed. Bulletin of mathematical biophysics, 1943.

MONARD, M. C.; BARANAUSKAS, J. A. **Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações**. São Paulo, Manoele, 1. ed. 2003.

POOLE, D., MACKWORTH, A. K., GOEBEL, R. **Computational Intelligence:** A Logical Approach. Oxford: Oxford University, 1998.

REDMON SANTOSH DIVVALA, R. G. J., FARHADI, A. **You only look once:** Unified, real-time object detection. Disponível em: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf. Acesso em: 07 set. 2021.

REZENDE, S. O. **Sistemas inteligentes:** fundamentos e aplicações. Editora Manole, 2003.

RUSSEL, Stuart, NORVIG, Peter. **Inteligência Artificial**. 3. ed. Elsevier Editora Ltda, 2013.

SCHWAB, Klaus. **A Quarta Revolução Industrial**. 1. ed. Edipro, 2018.

STAIR, Ralph M., REYNOLDS, George W. **Princípios de Sistemas de Informação**. São Paulo: Thomson, 2006.

VIANA, Suzana. **O pipeline de visão computacional**. 2020. Disponível em: https://suzana-svm.medium.com/o-pipeline-de-visao-computacional-com-python-opencv-adc70112f5ee. Acesso em: 03 maio 2022.