UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

BRUNO NOVELLI BONELA

CONTAGEM DE PESSOAS NO INTERIOR DE EDIFICAÇÕES POR TÉCNICAS DE VISÃO COMPUTACIONAL

VITÓRIA
2021

BRUNO NOVELLI BONELA

CONTAGEM DE PESSOAS NO INTERIOR DE EDIFICAÇÕES POR TÉCNICAS DE VISÃO COMPUTACIONAL

Parte manuscrita do Projeto de Graduação do aluno **Bruno Novelli Bonela**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Patrick Marques Ciarelli

VITÓRIA
2021

BRUNO NOVELLI BONELA

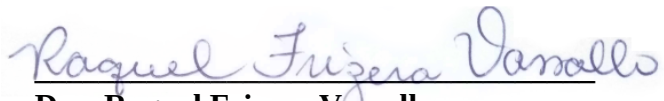
CONTAGEM DE PESSOAS NO INTERIOR DE EDIFICAÇÕES POR TÉCNICAS DE VISÃO COMPUTACIONAL

Parte manuscrita do Projeto de Graduação do aluno **Bruno Novelli Bonela**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

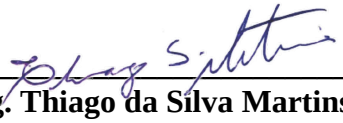
Aprovada em 12 de julho de 2021.

COMISSÃO EXAMINADORA:

Prof. Dr. Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Orientador



Dra. Raquel Frizera Vassallo
Universidade Federal do Espírito Santo
Examinador



Eng. Thiago da Silva Martins
Vale S.A.
Examinador

Aos meus pais, Cleuvera e Manoel.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus pela oportunidade de estar escrevendo este trabalho e estar ao meu lado em todos os momentos.

Aos meus pais, que me incentivaram e apoiaram em todos os momentos. Que apesar de todos os estresses, me suportaram e sustentaram das mais diversas formas, sempre com carinho, paz e otimismo.

Aos meus avós, que estiveram sempre ao meu lado e torcendo pelo meu sucesso. À minha avó Palmerinda, que no decorrer do desenvolvimento deste trabalho passou a torcer e orar por mim lá do céu.

Aos meus amigos de curso, os quais estiveram do meu lado sempre contribuindo para o meu aprendizado. Além de momentos de descontração, onde me auxiliaram muito nesses anos muito cansativos. Em especial, Franco Marchiori, Letícia Assis, Marianne Marinho, Matheus Franco e Thais Marchesi.

À minha namorada, Yasmin Paiva, por todo o apoio dado para a conclusão deste trabalho e curso. Que mesmo passando por momentos muito delicados e desafiadores, nunca deixou de me apoiar e motivar.

Aos professores e servidores do Departamento de Engenharia Elétrica.

Aos colegas de trabalho da Vale, que me apoiaram no desenvolvimento desse projeto sempre que necessário.

Ao professor e orientador Patrick Ciarelli, por todo o conhecimento compartilhado e apoio para o desenvolvimento deste trabalho. E também à Universidade Federal do Espírito Santo, por ser minha casa nos últimos anos, por todos os momentos em que vivi no campus Goiabeiras.

Dedico ainda esse trabalho a todos que perderam suas vidas para o COVID-19 e no desastre de Brumadinho, os quais foram as principais motivações para o desenvolvimento desde trabalho.

RESUMO

No decorrer da história, várias catástrofes em edificações aconteceram sem que fosse possível afirmar a quantidade de pessoas em seus interiores, havendo a necessidade de colocar a vida humana em risco para realizar a varredura em caso de uma evacuação. Esses acontecimentos poderiam ser evitados com a presença de um sistema que realizasse essa contagem e fornecesse os dados em uma rede externa para as equipes de apoio. Soluções utilizando cartões de acesso, sensores infravermelhos e outros métodos de contagens já foram aplicados em algumas localidades com o intuito de controlar o fluxo de pessoas. Entretanto, com o passar do tempo, novas tecnologias surgiram possibilitando soluções mais inteligentes. O objetivo do sistema desenvolvido nesse trabalho foi realizar a contagem de pessoas por meio de técnicas de Aprendizado Profundo aplicadas a uma Rede Neural Convolucional. Para a solução, foi utilizado o YOLOv4, o qual é um método de detecção de objetos em tempo real mantendo uma precisão igual ou superior aos outros métodos que realizam a mesma tarefa. Foi desenvolvido também, no escopo desse projeto, um portal para informar os dados analisados pelo sistema, um banco de dados para armazenar informações e uma API para disponibilizar os mesmos para outros sistemas, se necessário. A solução proposta neste trabalho utilizou um modelo previamente treinado com a base de dados pública MS COCO e câmeras de segurança em todas as entradas/saídas de uma edificação, as quais estavam conectadas em uma rede local ou internet transmitindo seus vídeos capturados em tempo real para o sistema. A precisão alcançada pelo sistema aqui desenvolvido foi de 97,3%.

Palavras-chave: Detecção de pessoas. Visão computacional. Aprendizado de máquina. Rede neural convolucional.

ABSTRACT

Over the course of history, several catastrophes in buildings have taken place without it being possible to affirm the number of people in the interior, being necessary to put human life at risk in order to carry out the sweep in the event of an evacuation. These events could be avoided with the presence of a system that performed this count and provided the data on an external network for the support teams. Solutions using access cards, infrared sensors and other methods of counting have already been employed in some locations in order to control the flow of people. However, over time, new technologies have emerged enabling more intelligent solutions. The objective of the system developed on this work is to perform the counting of people through the techniques of Deep Learning applied to a Convolutional Neural Network. For the solution, YOLOv4 was used, which is a method of detecting objects in real time, maintaining an accuracy equal to or greater than other methods that perform the same task. It was also developed in the scope of this project a portal to inform the data obtained by the system, a database to store the data and an API to make them available to other systems, if necessary. The solution proposed in this work used a model previously trained using the public database MS COCO and security cameras in all building entrances / exits, that were connected on a local network or internet transmitting your captured videos in real time to the system. The accuracy achieved by the system developed here was 97.3%.

Keywords: People detection. Computer vision. Machine learning. Convolutional neural network.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Networks*
API	*Applicattion Programming Interface*
CPD	Centro de Processamento de Dados
CNN	*Convolutional Neural network*
cuDNN	*NVIDIA CUDA® Deep Neural Network Library*
FPS	*Frames Per Second* (Quadros por Segundo)
GCC	*GNU Compiler Collection*
IoU	*Intersection Over Union*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
mAP	*mean Average Precision*
MVC	*Model-View-Controller*
PHP	*Hypertext Preprocessor*
PoE	*Power over Ethernet*
REST	*Representational State Transfer*
ROC	*Receiver Operating Characteristic*
RTSP	*Real Time Streaming Protocol*
SORT	*Simple Online and Realtime Tracking*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFES	Universidade Federal do Espírito Santo
WEB	*World Wide Web*
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação e Objeto de Pesquisa

O poder de processamento dos computadores tem crescido de forma contínua a cada ano, o que possibilitou o desenvolvimento e implantação de novos sistemas computadorizados em diversas aplicações do dia a dia da sociedade. Uma área que floresceu com o aumento do poder computacional foi a da Visão Computacional. Aplicações nesta área buscam interpretar cenas em imagens ou vídeos de forma semelhante ao que o ser humano é capaz de fazer, como identificar um objeto, uma ação ou um contexto. “A Visão Computacional visa usar câmeras para analisar e entender cenas no mundo real. Essa disciplina estuda problemas metodológicos e algorítmicos, assim como tópicos relacionados à implementação de soluções desenvolvidas” (KETTLE, 2014, p. vii, tradução nossa).

Uma aplicação que tem ganhado atenção nos últimos anos como potencial uso da Visão Computacional é o de videomonitoramento. Vários fatores influenciaram o crescimento deste tipo de sistema, como os altos índices de furto e invasão de ambientes privados e o barateamento e aumento da qualidade das câmeras. A Visão Computacional pode automatizar tarefas que comumente são feitas por pessoas ao analisarem vídeos de segurança. Com isso, existe a possibilidade de uma gama de aplicações de detecção inteligente de elementos que compõem as imagens/vídeos, tais como: identificação de pessoas foragidas, detecção de atividades suspeitas, reconhecimento de pedestres, etc. (KLETTE, 2014).

A dificuldade em contabilizar a quantidade de pessoas em uma edificação prejudica o resgate em caso de evacuações emergenciais. Com o acesso a informações de quantas pessoas existem no local, é possível minimizar o tempo das equipes de resgate em ambiente de risco (BARROS, 2017).

Observada a relevância do conhecimento da quantidade de pessoas em um edifício, surge a seguinte pergunta que guiou a realização deste trabalho: é possível elaborar um sistema confiável para contagem de pessoas em tempo real por meio de videomonitoramento?

O foco deste trabalho é abordar o problema de contagem de pessoas em um recinto utilizando técnicas de Aprendizado Profundo em uma Rede Neural Convolucional (CNN, do inglês *Convolutional Neural Network*), mais especificamente o detector de objetos *You Only Look Once* (YOLO). O recebimento das imagens a serem processadas dar-se-á pelo processamento assíncrono de câmeras de videomonitoramento conectadas a uma mesma rede local ou a internet utilizando o protocolo de comunicação *Real Time Streaming Protocol* (RTSP). As câmeras em conjunto devem suprir o sistema com imagens de todas entradas e saídas, que utilizando da rede neural e do algoritmo de rastreamento de objetos irá identificar a quantidade de pessoas que saem e entram no recinto (a soma de entradas menos a soma de saídas da edificação resultará na quantidade de pessoas no interior da mesma). Esses dados são disponibilizados aos usuários via site na internet ou via *Application Programming Interface* (API) para algum outro sistema que possa utilizá-los.

1.2 Justificativa

A falta de informação de quantas pessoas estão no interior de um edifício é algo que pode ocasionar problemas em uma eventual evacuação. Existem alguns métodos praticados na rotina de companhias para auxiliar na evacuação (GASPAR; LEITÃO; REZENDE, 2018), como:

- Um líder por ambiente: o qual tem a função de garantir que todos conseguirão sair em segurança;
- Contagem de pessoas utilizando o registro de acesso com fechaduras nas portas ou catracas.

A contagem utilizando sistemas é algo complexo e custoso para as companhias. Os custos de instalação de câmeras que realizam o mesmo procedimento que o proposto neste trabalho são mais de 24 vezes o custo de um sistema de monitoramento comum. (AXIS, 2019).

Para sistemas em tempo real, como o desenvolvido neste trabalho, existe a necessidade de servidores, os quais geram um custo de infraestrutura de Centro de Processamento de Dados (CPD) para suportar os equipamentos e manter seu pleno funcionamento (VARELLA, 2019).

Além de todos os quesitos de segurança envolvidos na contagem de pessoas em um edifício, é possível também utilizar estes dados para controle de fluxo e acompanhar possíveis

descumprimentos das limitações do ambiente em relação à quantidade das mesmas (BARROS, 2017).

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho foi o desenvolvimento de um sistema de contagem de pessoas em uma edificação que possui mais de um acesso e presença de câmeras de segurança digitais. O sistema desenvolvido é de fácil configuração e portabilidade, para que possa ser replicado em novas localidades sem a necessidade de uma nova instalação física, desde que exista uma infraestrutura de comunicação das câmeras de segurança com a rede de internet.

1.3.2 Objetivos Específicos

Como objetivos específicos, deseja-se que o sistema a ser desenvolvido disponibilize ao usuário funcionalidades que permitam:

- Verificar as detecções e rastreamentos, sendo realizados em tempo real na tela de gerenciamento;
- Analisar a quantidade e o fluxo de pessoas que entram e saem da edificação a qualquer momento por meio de uma interface *World Wide Web* (WEB).

1.4 Classificação da Pesquisa

Do ponto de vista de sua natureza, o atual trabalho classifica-se como uma pesquisa aplicada, pois seu objetivo final foi gerar um algoritmo para contagem de pessoas em uma edificação. Quanto aos objetivos, classifica-se como uma pesquisa explicativa, onde há necessidade de aprofundamento da realidade, baseando-se na análise de diferentes vídeos gerados em condições adversas.

Do ponto de vista dos procedimentos técnicos adotados, este trabalho possui caráter experimental, pois definiu-se o objeto de estudo como sendo a análise de imagens para que

fosse possível realizar a contagem de pessoas por meio de técnicas de Visão Computacional. Por fim, a abordagem do problema se deu de forma quantitativa.

1.5 Estrutura do Texto

O presente trabalho está estruturado da seguinte maneira:

a) **Introdução**: a primeira seção tem como objetivo contextualizar a problemática abordada por esse trabalho, apresentando o problema, indicando suas aplicações no cotidiano e as ideias iniciais de solução. Em adição, são expostos os objetivos gerais e específicos para a realização deste projeto de graduação;
b) **Embasamento Teórico**: nesta seção são apresentados os conceitos fundamentais para entendimento do sistema proposto, tais como: CNN, RTSP, YOLOv4, dentre outros;
c) **Método Proposto**: aqui serão descritas todas as etapas do sistema proposto para a resolução do problema abordado;
d) **Experimentos e Resultados**: nesta seção serão descritas as bases de dados, as métricas e os experimentos, assim como os resultados e suas análises;
e) **Conclusão**: na seção final deste trabalho serão indicadas todas as conclusões relativas aos resultados obtidos, além de apresentar propostas de melhorias para estudos futuros.

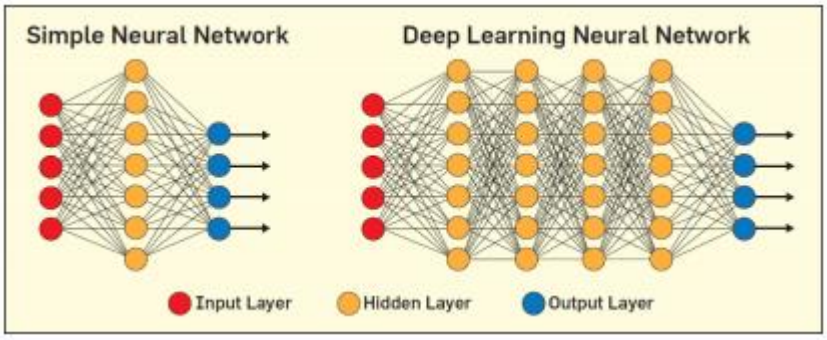
2 EMBASAMENTO TEÓRICO

2.1 Aprendizado Profundo

Deep learning (aprendizado profundo) tem sido muito estudado e implementado nos últimos anos, representando uma evolução significativa nos métodos de inteligência artificial. Sua grande evolução tem por origem a implementação de algoritmos de treinamento de redes neurais que permitem treinar várias camadas conectadas (LECUN; BENGIO; HINTON, 2015).

Uma rede de aprendizado profundo se diferencia devido à quantidade de camadas internas para processamento das entradas iniciais, como mostrada na Figura 1. Tal quantidade de camadas faz com que as regras de predição deixem de ser explícitas, entretanto, adiciona um maior grau de liberdade para que a rede consiga aprender sistemas mais complexos e menos lineares (FACURE, 2017).

Figura 1 – Arquitetura de redes neurais: (a) rede neural rasa e (b) rede neural de aprendizado profundo



(a) (b)

Fonte: Edwards (2018).

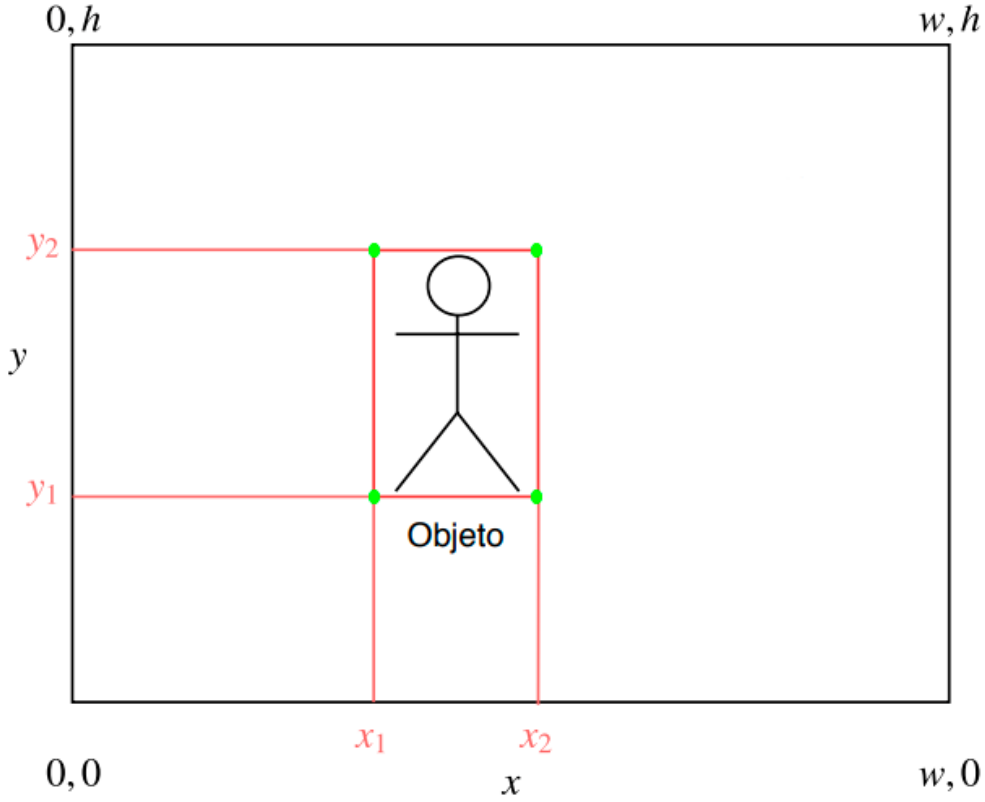
2.2 Detecção de Pessoas em uma Imagem

Sistemas de visão computacional podem ser utilizados para a análise de mais de um objeto na mesma imagem. O processo mais usual de predição é a classificação de uma imagem quanto ao seu tipo, associando rótulos para descrever o que há na imagem ou se possui ou não uma característica.

Alguns sistemas permitem localizar a posição do objeto na imagem, sendo cada vez mais complexo no caso de múltiplos objetos na mesma imagem. Sistemas deste tipo usam modelos de detecção de objetos que recebem como entrada uma imagem digital (ou *frames* de um vídeo) com um ou mais objetos na cena e geram uma saída de uma ou mais *bounding boxes* (tradução, caixas delimitadoras), normalmente definidas por 4 pontos limitando a região do objeto e um rótulo de identificação do tipo de objeto (“aprendido” no treinamento do modelo) para cada uma das *bounding boxes*.

Na Figura 2 pode ser observado o detalhamento de uma *bounding box*, no qual um objeto que está contido em uma imagem digital w × h é limitado por 4 pontos, que quando unidos por 4 seguimentos de retas, formam um retângulo.

Figura 2 – Detalhamento da estrutura de uma *bounding box*



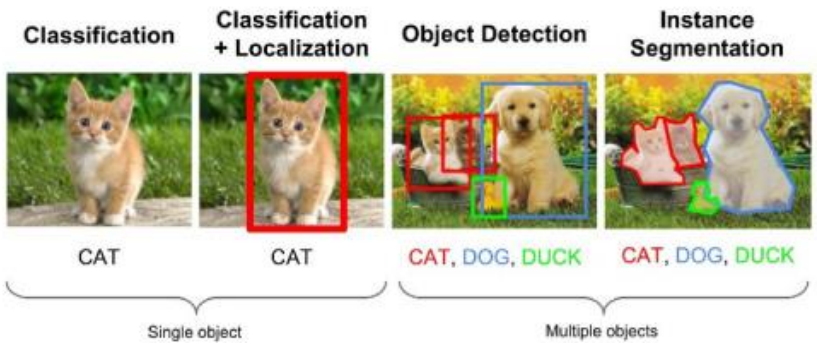
Fonte: Produção do próprio autor.

Há também a possibilidade de não detectar apenas a localização de um objeto, mas os exatos pixels que o compõem. Essa técnica é chamada de Segmentação de Instância (do inglês *Instance Segmentation*) que efetua a classificação do objeto em nível de pixel, conseguindo então definir

quais os pixels que se referem ao mesmo e sua classe, tratando vários objetos de uma mesma classe de forma individual. Uma comparação entre a detecção de objetos e a *Instance Segmentation* pode ser observada na Figura 3. Para este trabalho será utilizada apenas a detecção de objetos, visto que a identificação dos pixels referentes a cada um deles não apresenta nenhum ganho para o sistema.

A detecção da localização do objeto na imagem foi crucial para realização desse projeto, pelo fato da necessidade de perceber se o movimento que está sendo realizado pela pessoa é de saída ou entrada, pois é muito comum em edificações que a entrada e a saída sejam realizadas pelo mesmo local do edifício. Com essa aquisição de localização, pode-se fornecer esses dados para um algoritmo de *tracking*, por exemplo, e com isso analisar a movimentação da pessoa no local desejado.

Figura 3 – Comparação entre tipos de objetivos na visão computacional



Fonte: Ouaknine (2018).

Alguns trabalhos interessantes, identificados na literatura, relacionados ao problema de detecção, contagem e rastreamento de objetos em imagens digitais foram de:

- Bathija e Sharma (2019), que realizaram a detecção de objetos utilizando a YOLO e seu rastreamento utilizando o Rastreamento Simples *Online* e em Tempo Real (SORT, do inglês *Simple Online and Realtime Tracking*), sem uma efetiva contagem dos objetos em uma região. Para esse trabalho a rede foi treinada para detectar 6 diferentes classes de objetos, e foi possível perceber a necessidade de um treinamento com um grande banco de imagens baseado nos resultados para o uso de 800 imagens. Além disso, é afirmado que o desempenho do rastreamento é dependente do desempenho dos detectores.

- Ahmad, Ning e Tahir (2019), que se preocuparam em realizar a detecção de pedestres em tempo real utilizando o YOLOv3. A precisão da detecção de pedestres foi de 98,1%, 87,3% de r*ecall* e 89,78% de m*ean Average Precision* (mAP). Para obter esses resultados foi utilizado um modelo previamente treinado com a base de imagens do MS COCO. Além disso, o trabalho conseguiu atingir valores altos de *frames per second* (FPS), o que é desejável para sistemas de execução em tempo real.
- Cordeiro, Paula Filho, Ojeda e Valiati (2019), que apresentam uma proposta para a contagem do fluxo de pedestres, informando quantas entraram e saíram de uma delimitada região. Para esse trabalho foi utilizada a YOLOv3, a qual foi treinada para detectar 200 diferentes classes de objetos. Como o intuito era a detecção de pessoas, foi utilizado um verificador de classes para aceitar apenas a desejada. Nesse trabalho é informado apenas que para a contagem de multidões a precisão é de 50%.
- Nogueira (2018), que analisou diferentes redes neurais para a contagem de pessoas com câmera de segurança. É possível perceber que a YOLOv3 obteve um menor erro dentre as redes analisadas, porém com o menor valor de FPS. As demais tiveram valores consideravelmente maiores de erro.

2.3 YOLO

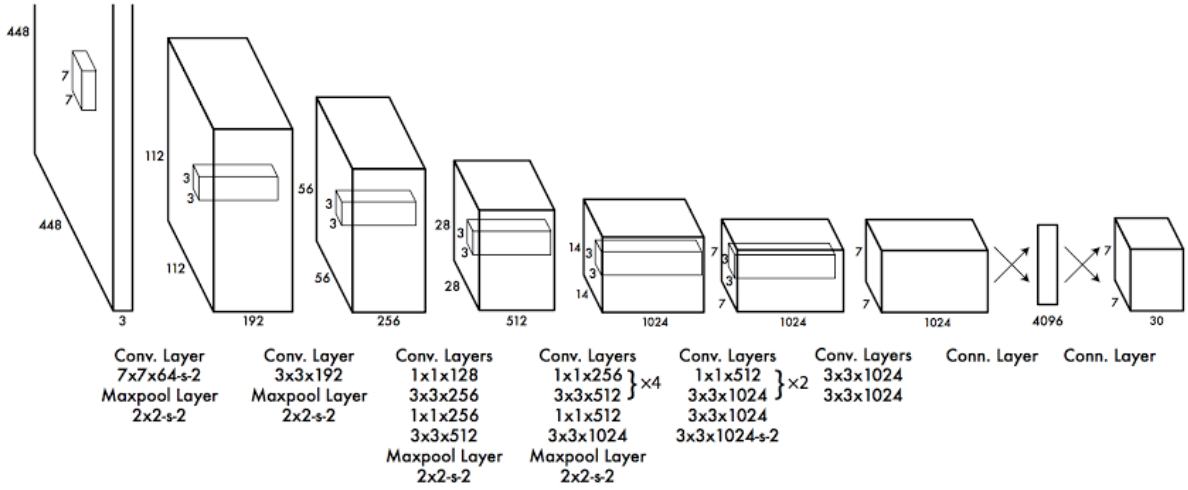
You Only Look Once (em português, Você Só Olha Uma Vez), ou comumente chamada de YOLO, é uma arquitetura que se apresentou como destaque quando o objetivo é a detecção de objetos em tempo real (NOGUEIRA, 2018).

Como passo inicial, a YOLO divide a imagem em uma grade de $S \times S$ células, onde para cada célula é predita uma quantidade limitada de *bounding boxes* (caixas) (B). Cada uma das *bounding boxes* tem uma nota de confiança associada a cada uma das C classes (objetos) que a rede pode detectar (NOGUEIRA, 2018).

Um dos principais conceitos da YOLO é construir uma rede neural para reduzir o volume de saída para $S \times S \times (B \times C)$. Por exemplo, em um cenário onde se tem 15 classes para predição ($C = 15$), duas *bounding boxes* por célula ($B = 2$) e uma grade de 7×7 células ($S = 7$), tem-se,

ao final, uma saída de rede com dimensão 7 × 7 × 30, conforme mostrado na Figura 4 (NOGUEIRA, 2018).

Figura 4 – Arquitetura da CNN aplicada na YOLO



Fonte: Redmon, Divvala, Girshick e Farhadi (2016).

A estratégia escolhida pela arquitetura de dividir diversas caixas por célula, faz com que haja objetos sendo detectados em diferentes *bounding box* repetidas vezes, como apresentado na Figura 5. Com isso, torna-se necessária a utilização do processo de *Non-maximal Suppression* (em português, Supressão do Não Máximo), que basicamente consiste em duas etapas:

a) Ordenar as predições com base em suas notas de confiança;
b) Começar com as detecções de maior confiança, ignorando as subsequentes que possuem uma sobreposição maior que 50% com a mesma classe.

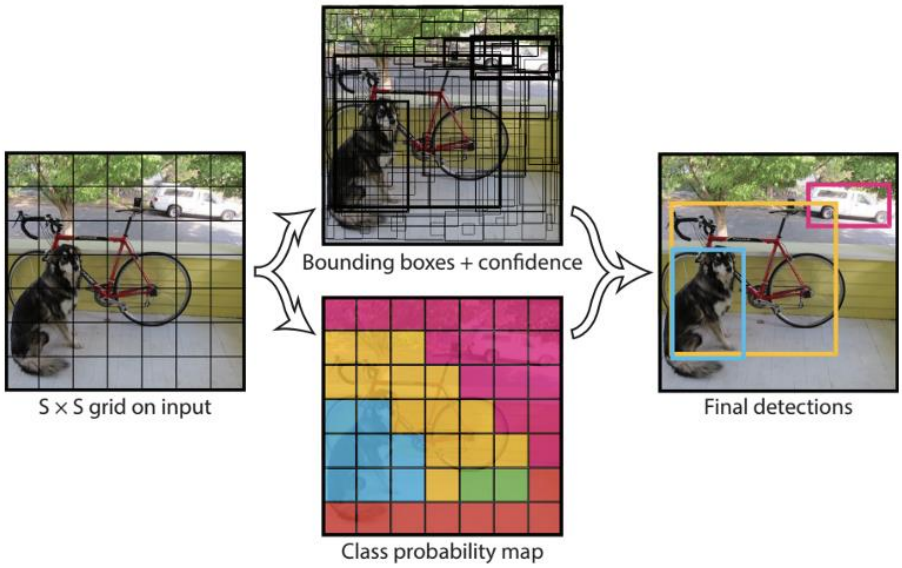
A grande vantagem que há na escolha da YOLO nesse projeto, é a sua alta velocidade na predição de uma imagem que é passada pela rede convolucional apenas uma vez. Redes deste formato são chamadas de *Single Shot Detectors*, as quais são referências nos casos de soluções que necessitam de realizar análises em tempo real (REDMON; FARHADI, 2018).

Para o treinamento da YOLO, é necessária a instalação do *Software Development Kit* (SDK), softwares e dependências do Python. Esses são:

- CMake;
- CUDA;
- NVIDIA cuDNN;

- OpenCV;
- GPU with CC;
- GCC;
- Darknet.

Figura 5 – Estrutura da YOLO



Fonte: Redmon e Farhadi (2018).

Após a instalação de todos os itens acima, é necessário o preenchimento dos arquivos de configuração fornecidos para o treinamento da YOLO. Para esse trabalho foi escolhida a versão 4, no qual melhora AP e FPS em 10% e 12%, respectivamente (BOCHKOVSKIY; WANG; LIAO, 2020).

Ela é fornecida e configurada para detectar e classificar 80 classes quando treinada com a base de imagens do MS COCO, contendo imagens rotuladas contendo ou não os objetos, a qual foi utilizada neste trabalho (LIN et al., 2014).

Após isso, é necessária a configuração de uma base de dados que informe para a rede neural imagens digitais e um arquivo informando os *bounding boxes*, marcando os objetos de interesse nas imagens para o aprendizado da rede.

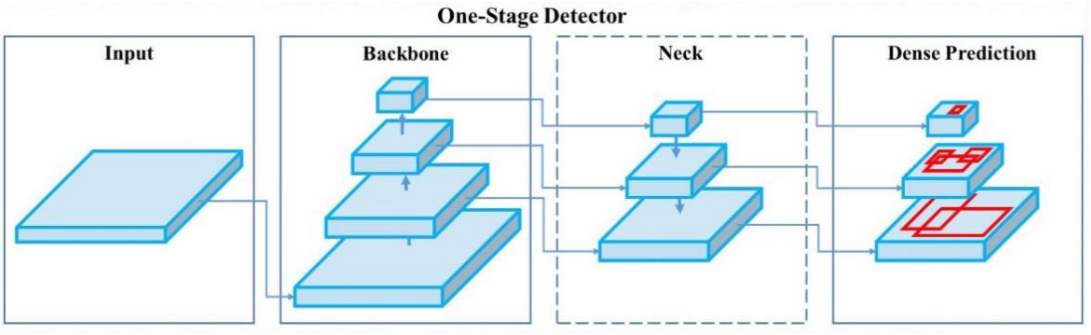
O modelo de rede neural fornecido foi pré-treinado adotando uma taxa de aprendizagem de 0,0013 e uma total de 500.500 iterações, sendo que nas iterações 400.000 e 450.000 a taxa de aprendizagem foi dividida por 10. Essa é uma das estratégias adotadas pela YOLO com o objetivo de melhorar sua precisão, visto que após para a quantidade de iterações já realizadas, a rede possui a capacidade de realizar a detecção e a partir desse ponto realizamos um treinamento de características mais específicas (BOCHKOVSKIY; WANG; LIAO, 2020).

Sistemas modernos de detecção de objetos usualmente consistem de 4 componentes (GUO et al., 2020): a) *backbone*, para extrair características semânticas; b) *neck*, para fundir características multi-nível; c) rede de região proposta (do inglês *region proposal network*), para gerar regiões propostas para detecção (usualmente usada em detectores de dois estágios); d) *head*, para classificação de objetos e localização de *bounding boxes*. A YOLOv4 utilizada neste trabalho possui 3 componentes, sendo a arquitetura base definida para melhor velocidade e precisão de detecção:

- Backbone: CSPDarknet53 (WANG et al., 2020);
- Neck: SPP (HE; ZHANG; REN; SUN, 2015), PAN (LIU et al., 2018);
- Head: *Dense Prediction*: YOLOv3 (REDMON; FARHADI, 2018).

Essa arquitetura pode ser visualizada na Figura 6.

Figura 6 – Arquitetura do detector de objetos



Fonte: Bochkovskiy, Wang e Liao (2020).
Nota: Adaptado pelo autor.

2.4 Rastreamento em Tempo Real

O rastreamento dos objetos detectados em tempo real é realizado utilizando o SORT, no qual se torna possível o acompanhamento do objeto por períodos mais longos, mesmo quando em oclusão, reduzindo efetivamente o número de troca de identidade (WOJKE; BEWLEY; PAULUS, 2017).

O algoritmo SORT propõe a combinação do filtro de Kalman e o método húngaro de otimização para o rastreamento de objetos em um conjunto de *frames*. Ele utiliza um vetor de *bounding boxes* como entrada e consegue rastrear os múltiplos objetos que possam existir em tempo real, alcançando bons resultados com baixo custo de processamento. Basicamente, o algoritmo utiliza os pontos detectados, constrói uma predição do posicionamento de cada ponto para os próximos quadros e assimila aquele ponto com maior proximidade com a sua predição. É necessário utilizar uma matriz associativa de custos de rastreamento para manter o registro dos pontos, e a partir disso, ser possível obter o índice de menor custo (WOJKE; BEWLEY; PAULUS, 2017).

O algoritmo de rastreamento pode apresentar dificuldades ao tentar rastrear os objetos quando ocorrem oclusões. Esse caso pode acontecer principalmente quando duas pessoas entram ou saiam muito próximas, ou carregando algum objeto capaz de obstruir a visão da câmera sobre a pessoa.

Para esse trabalho, é proposto que cada câmera carregue um algoritmo de rastreamento independente para que não tenha mistura dos *frames* recebidos de diferentes localidades para análise, visto que o mesmo necessita de analisar o frame anterior para a respectiva câmera. Além disso, será utilizado o DeepSORT (WOJKE; BEWLEY; PAULUS, 2017), o qual adiciona *Matching Cascade* (pseudocódigo na Figura 7) e uma nova confirmação de trajetória ao SORT.

O *Matching Cascade* é dedicado a resolver a situação em que o alvo fica em oclusão por um longo período, realizando uma comparação entre as detecções e a predição do filtro de Kalman. Para que a detecção atual corresponda à *bounding box* de rastreamento que está mais próxima

do presente momento, ao realizar essa tentativa, ele dará prioridade à *bounding box* com o tempo de desaparecimento mais curto (WOJKE; BEWLEY; PAULUS, 2017).

Figura 7 – Pseudocódigo do *Matching Cascade*

Listing 1 Matching Cascade

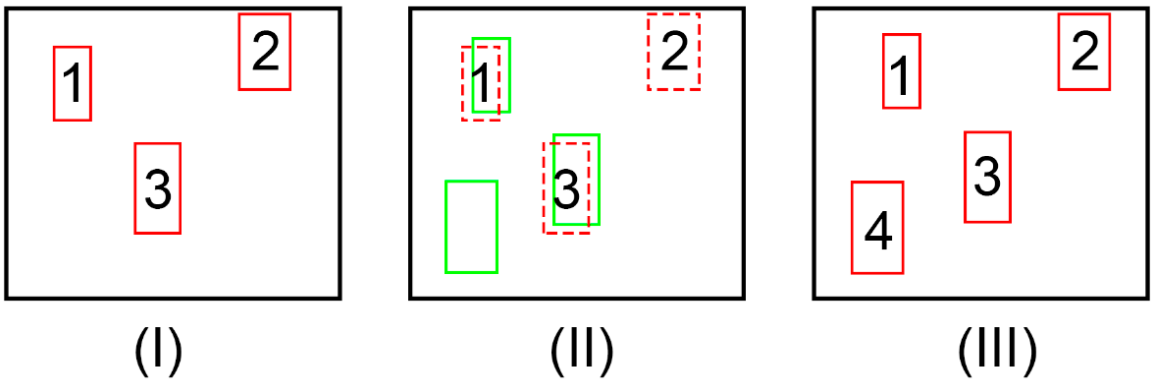
Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6
Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
for $n \in \{1, \ldots, A_{\max}\}$ **do**
  Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
  $[x_{i,j}] \leftarrow \text{min\_cost\_matching}(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
end for
return $\mathcal{M}, \mathcal{U}$

Fonte: Wojke, Bewley e Paulus (2017).

Na Figura 8 podemos observar um fluxo de análise do algoritmo de rastreamento, no qual ele recebe em (I) as predições do filtro de Kalman, realiza a comparação das predições com as detecções do *frame* atual em (II), resultando nos objetos rastreados em (III).

Figura 8 - Exemplo de uma análise do algoritmo de rastreamento



Fonte: Produção do próprio autor.

Com isso, há as seguintes situações possíveis:

a) Detecção e correspondência de rastreamento, ou seja, uma detecção confirmada. Os alvos de rastreamento contínuo comuns pertencem a esta situação. Existem alvos nos dois quadros antes e depois, que podem ser combinados (objetos 1 e 3 na Figura 8);

b) Detecção não encontrou um rastreamento correspondente, ou seja, uma detecção sem correspondência. Quando um novo alvo aparece repentinamente na imagem, a detecção não consegue encontrar um alvo correspondente na trilha anterior, e com isso é criado um novo objeto no rastreamento (objeto 4 na Figura 8);
c) O rastreamento não encontrou uma detecção correspondente, ou seja, rastreamento incompatível. O alvo rastreado continuamente excede a área da imagem e com isso não pode corresponder a nenhuma detecção atual. Caso esse rastreamento não encontre uma detecção correspondente por mais de 30 *frames*, esse objeto será deletado do algoritmo.

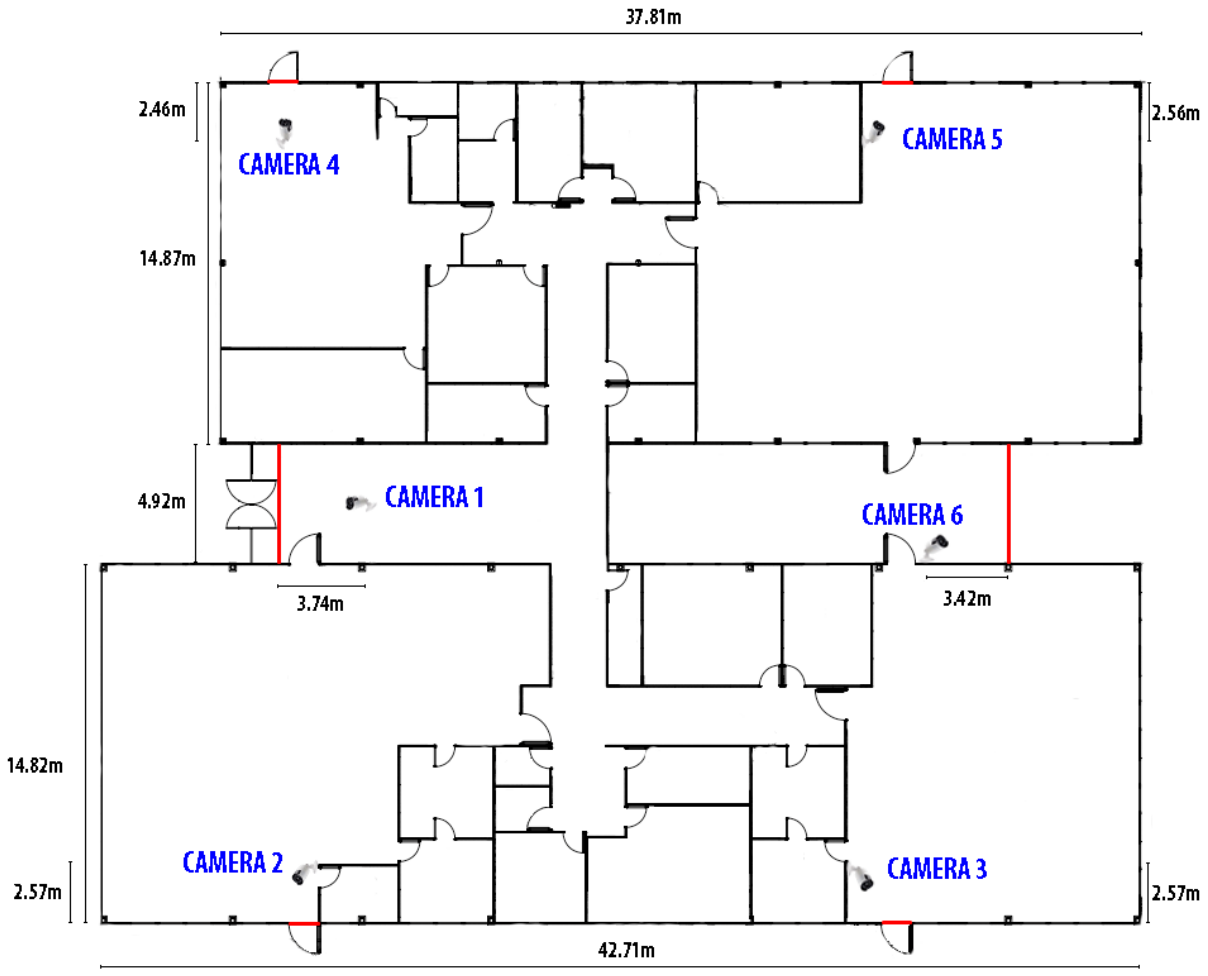
3 SISTEMA PROPOSTO

O sistema projetado neste trabalho foi dividido em módulos que juntos completam todos os objetivos esperados.

3.1 Ambiente e Base de Dados

A aquisição de imagens foi realizada através de 6 câmeras digitais de segurança instaladas sem possibilidade de movimentação em uma edificação utilizada como ambiente de teste. As câmeras estão localizadas a 2,90 metros de altura em relação ao piso e em diferentes distâncias em relação à linha de limitação de ambientes (linhas em vermelho na Figura 9) localizadas em cada uma das saídas, conforme a Figura 9.

Figura 9 – Planta do ambiente utilizado no projeto e disposição das câmeras de segurança. Linhas em vermelhas são as entradas/saídas do recinto

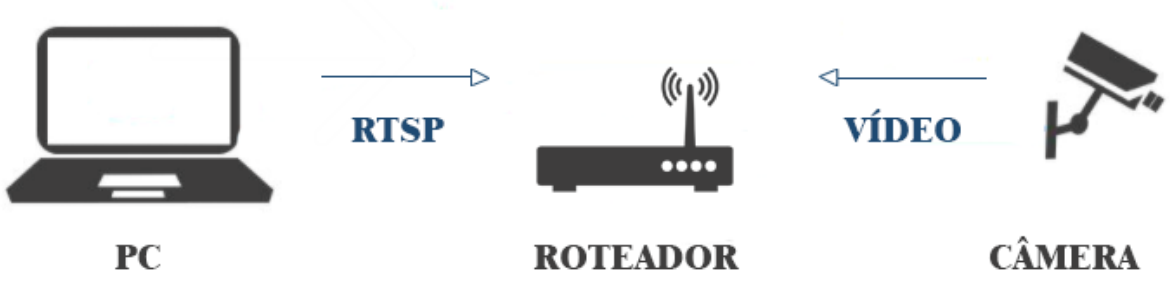


Fonte: Produção do próprio autor.

As imagens que compõem os vídeos estão disponíveis em uma Interface de Programação de Aplicativos (API, do inglês *Application Programming Interface*), hospedada em um servidor na rede local do edifício de teste. Elas são fornecidas com largura de 352 pixels e altura de 240 pixels. Essa definição é a padrão para as câmeras de segurança utilizadas em sua camada de transmissão rápida, tendo assim uma maior quantidade de FPS e menor alocação de banda da rede.

O servidor provê as imagens utilizando do protocolo RTSP para a transmissão de *streams*, conforme a Figura 10. Internamente, ele utiliza os protocolos *User Datagram Protocol* (UDP) ou *Transmission Control Protocol* (TCP) e possui um link para o fornecimento do conteúdo de mídia em tempo real no formato rtsp://caminho_servidor/cam_ID.

Figura 10 – Estrutura de aquisição de imagens com protocolo RTSP



Fonte: Produção do próprio autor.

Podem ser vistos na Figura 11 dois exemplos de imagens da base de dados deste trabalho. Nem todos os *frames* utilizados foram armazenados por questões de direito de imagem das pessoas presentes. Para realização de testes, ajustes e exemplos foram geradas imagens com pessoas que autorizaram o uso ou tiveram seus rostos ofuscados. Além disso, todas as conexões com as câmeras apresentam a necessidade de informar usuário e senha para serem estabelecidas, evitando assim acessos e gravações indevidas.

Figura 11 – Exemplo de imagens da base de dados

Fonte: Produção do próprio autor.

3.2 Contador de Pessoas

O módulo contador de pessoas foi desenvolvido em *Python 3.6* e é o responsável por unificar as principais funções em um fluxo de tarefas. Ele tem a capacidade de receber os parâmetros:

a) ID da câmera a ser analisada;
b) Se o servidor de imagens será síncrono ou assíncrono;
c) Se o usuário deseja salvar as imagens geradas em formato de vídeo;
d) Exibir modo de desenvolvedor, no qual o mesmo consegue obter mais informações na tela.

Após realizar o carregamento das configurações, o modelo do YOLOv4 é instanciado para que as detecções possam ser geradas, e o modelo de rastreamento DeepSORT é carregado, tal que, pode-se iniciar o recebimento de imagens pelo servidor.

O sistema não inicia a contagem de pessoas em zero, ele possui um *backup* em formato de arquivo de texto que relaciona a câmera desejada com a última contagem de pessoas que foi realizada, fazendo um somatório.

Ao receber uma imagem do servidor, o algoritmo inicia a detecção de objetos e, ao realizar a detecção de uma pessoa (demais classes detectadas pela rede não são analisadas, sendo ignoradas via código), é enviado para o DeepSORT realizar o rastreamento da pessoa, que retorna uma *bounding box* associada a mesma.

Cada indivíduo recebe uma identificação se já foi computado em alguma contagem de entrada ou saída do local. A verificação se o sentido de movimento é de entrada ou saída de uma localidade é definida pelo usuário (dependendo do posicionamento da câmera, o sentido do movimento pode mudar, e com isso foi necessária a implementação da possibilidade de escolha do usuário).

O sistema é capaz de detectar o sentido do movimento em relação às imagens analisadas. É utilizado um ponto médio da *bounding box* da pessoa identificada e realizada a intersecção entre retas. Sendo elas:

a) Reta 1 limitando a saída e entrada do ambiente, normalmente alinhada horizontalmente na altura média da imagem;
b) Reta 2 entre a posição anterior e atual da detecção da pessoa. São armazenados os últimos pontos de localização da pessoa e feito o cálculo do coeficiente angular entre o décimo ponto (ou o mais antigo caso não tenha o decimo ponto) anterior com o atual. Com isso, é gerado um seguimento de reta a ser utilizado na verificação de intersecção.

Se a pessoa estiver no sentido de saída da edificação e o ponto médio dela estiver além deste ponto de intersecção, é considerada a sua saída. Procedimento similar, mas com sentido inverso é feito para contar a entrada.

Essas retas podem ser observadas na Figura 12, que no caso apresenta a reta 2 com coeficiente angular positivo (pessoa saindo para a definição atual do usuário) em intersecção com a reta 1. Além disso, pode ser observado que foi classificado como pessoa o rastreamento presente, conforme indicada pela *bounding box* branca.

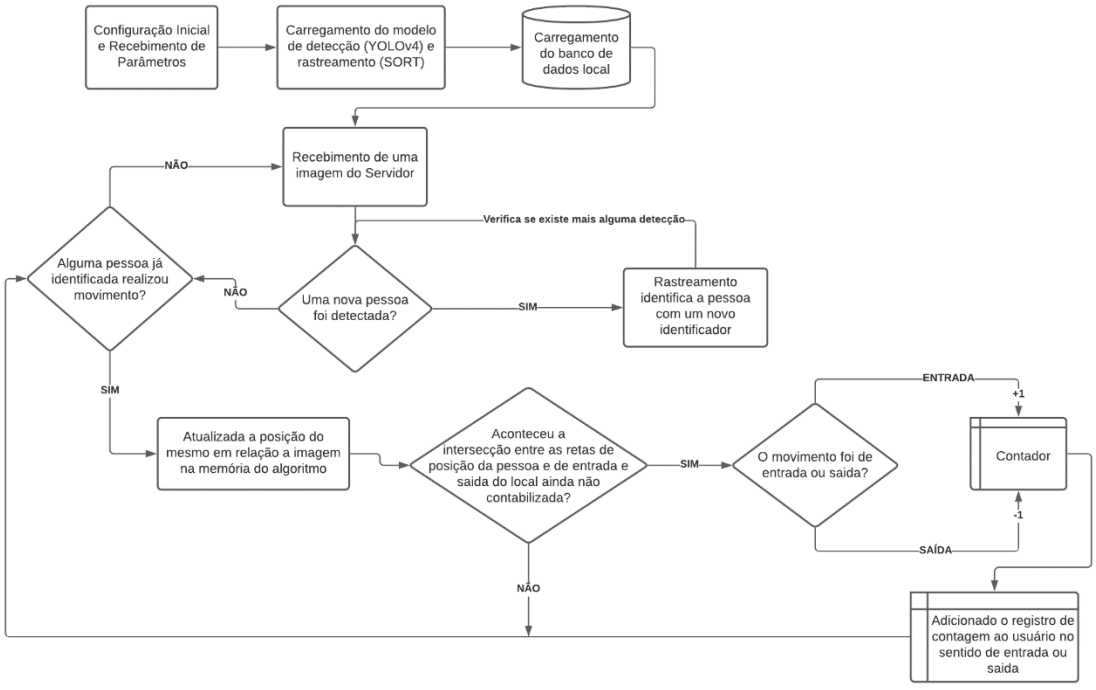
Como para o rastreamento é necessário o armazenamento dos vetores das *bounding boxes* brancas, foi adicionado um campo no qual é possível verificar se uma determinada pessoa já foi computada pelo contador (ou seja, se já existiu um ponto de intersecção das retas 1 e 2) e qual foi o sentido do movimento. Com isso, é verificado se existiu um acontecimento de saída ou entrada de forma duplicada pela mesma pessoa. Deste modo, evita-se que uma mesma pessoa saia ou entre duas vezes consecutivas no ambiente, evitando assim um possível erro de contagem.

Figura 12 – Retas utilizadas para verificação de intersecção para detecção de entrada ou saída

Fonte: Produção do próprio autor.

Após a detecção, é enviado para o gestor de dados as informações de quantas pessoas entraram e quantas pessoas saíram de cada uma das câmeras para serem armazenadas. Inicialmente estas informações são salvas em um arquivo de texto local, e em seguida enviadas para a API. O fluxo pode ser observado na Figura 13.

Figura 13 – Fluxograma de funcionamento do Contador de Pessoas



Fonte: Produção do próprio autor.

3.3 Contador de Pessoas – API e Módulo WEB

A API é responsável por realizar o recebimento dos dados de entrada e saída de uma nova pessoa, além da identificação de qual o ID da câmera a ser analisada. A API está desenvolvida em PHP utilizando o framework *Codeigniter* e seguindo o estilo de arquitetura de Transferência Representacional de Estado (REST, do inglês *Representational State Transfer*). Tornando a mesma uma API *RestFul* devido ao seu comprometimento com as regras desse estilo, algo relevante quando desenvolvido um *backend*.

Os dados aprovados pela API a serem armazenados no banco de dados são diretamente enviados para o gestor de banco de dados, que neste trabalho foi utilizado o MariaDB, um software de banco de dados criado sobre a base do MySQL com o intuído de manter o código aberto (BARTHOLOMEW, 2015).

Além da análise para armazenamento dos dados, a API é responsável por fornecer aos demais sistemas as informações ali contidas. Isso é possível utilizando um usuário e senha para que sejam limitados os acessos à informação. Neste trabalho, o módulo WEB é o único autorizado a consumir os dados da API. Todos são transmitidos em formato JSON, conforme a Figura 14.

Figura 14 – Dados fornecidos pela API em JSON


```
{
    "peopleInside": 22,
    "peopleFlowToday": 55,
    "peopleFlowWeek": 55,
    "peopleFlowMonth": 55,
    "updatedAt": "20 dias atrás"
}
```


Fonte: Produção do próprio autor.

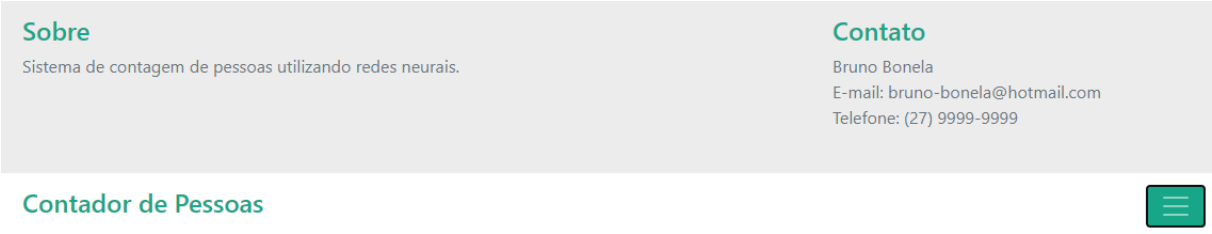
O Módulo WEB é responsável por deixar os dados visíveis aos usuários de forma organizada e saneada. Com isso, é possível realizar a unificação das câmeras de diferentes localidades, mas que juntas monitoram todas as entradas/saídas do edifício. Além disso, é possível realizar algumas análises de fluxo de pessoas em relação ao tempo, conforme observado na Figura 15.

Figura 15 – Módulo WEB do Contador de Pessoas

Fonte: Produção do próprio autor.

Na Figura 16 é mostrado o cabeçalho da página WEB que contém as informações básicas sobre o sistema e os dados de contato do desenvolvedor.

Figura 16 – Menu com informações do projeto e contato do desenvolvedor



Fonte: Produção do próprio autor.

Ele pode ser acessado através de qualquer plataforma que tenha acesso à internet e de qualquer tamanho ou formato de tela, uma vez que foi desenvolvido utilizando técnicas de responsividade.

Toda a API e módulo WEB foram desenvolvidos em mesmo código utilizando o *framework* CodeIgniter 3 (BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY, 2019), hospedado

em um servidor Apache com PHP 7. CodeIgniter é baseado no padrão de desenvolvimento *Model-View-Controller* (MVC). MVC é uma abordagem de software que separa a lógica do aplicativo da apresentação. Na prática, permite que suas páginas da WEB contenham um mínimo de scripts, já que a apresentação é separada do script PHP (LI; KARNAN; CHISHTI, 2017):

- O modelo representa suas estruturas de dados. Normalmente, suas classes de modelo conterão funções que o ajudam a recuperar, inserir e atualizar informações em seu banco de dados;
- A visão é a informação que está sendo apresentada a um usuário e que normalmente será uma página da WEB. Entretanto, no CodeIgniter, ela também pode ser um fragmento de página como um cabeçalho ou rodapé;
- O Controlador atua como um intermediário entre o Modelo, a Visualização e quaisquer outros recursos necessários para processar a solicitação HTTP e gerar uma página da WEB.

O CodeIgniter tem uma abordagem bastante flexível para MVC, uma vez que os modelos de dados não são necessários. Caso não seja necessária a separação adicional, ou seja, conhecido que a manutenção de modelos requer mais complexidade do que desejado, é possível ignorá-los e construir seu aplicativo minimamente usando controladores e visualizações. O mesmo também permite a incorporação de scripts já existentes, ou até mesmo bibliotecas centrais desenvolvidas para o sistema, permitindo uma grande maleabilidade do *framework*.

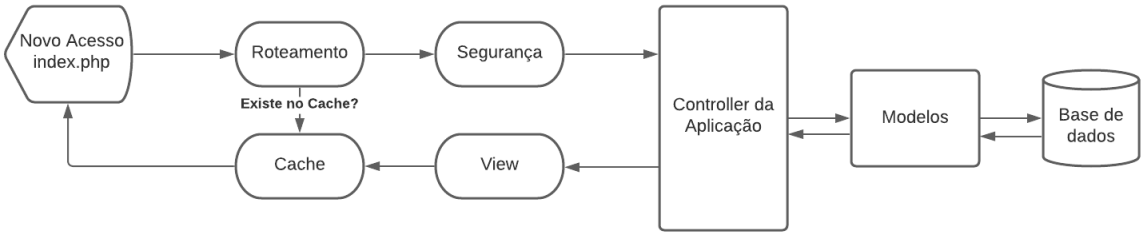
Além disso, é um sistema instanciado dinamicamente, fracamente acoplado com alta singularidade de componentes. Ele busca simplicidade, flexibilidade e alto desempenho em um pacote compacto.

Na Figura 17 são visualizados os seguintes elementos:

a) Index.php: serve como um controlador primário, iniciando os recursos básicos necessários para rodar o *framework*;
b) Roteamento: análise da requisição HTTP para determinar o caminho a ser percorrido por ela;
c) Cache: caso o arquivo já exista, ele é enviado para o solicitante sem que sejam necessárias novas interações com o *Controller*;

d) Segurança: análise da requisição para verificar se cumpre todos os requisitos de segurança;
e) *Controller*: carrega o modelo e as principais bibliotecas, além de conter todas as regras de negócio;
f) *View*: é renderizada a informação a qual deve ser mostrada ao usuário;
g) *Models*: é realizada toda a conexão com a base de dados.

Figura 17 – Fluxo baseado na arquitetura do CodeIgniter 3



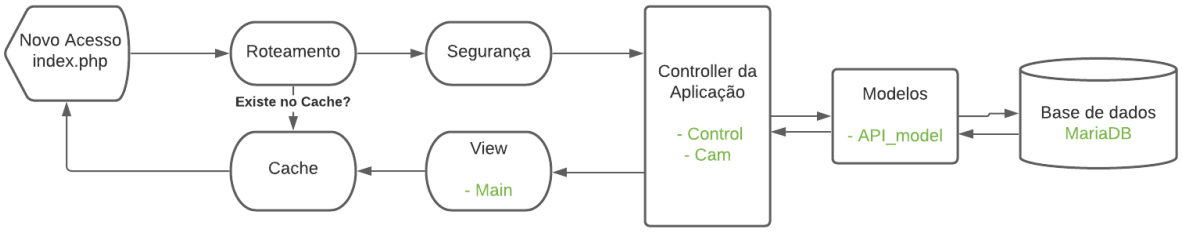
Fonte: Produção do próprio autor.

A estrutura utilizada no trabalho proposto é baseada no *framework*, sendo adicionados alguns novos elementos para que sejam possíveis aplicar as funcionalidades ao sistema:

- Controladores:
  - Control: fornece os dados no formato da Figura 14;
  - Cam: fornece os dados de quantas pessoas entraram e saíram para o último registro no sistema e recebe novos dados das câmeras;
- Visualizadores:
  - Main: entrega para o usuário a tela com as informações do sistema, conforme Figura 15;
- Modelos:
  - API_model: fornece os dados para os controladores contidos no banco de dados.

É possível assim, redesenhar o fluxo apresentado na Figura 17, adicionando as implementações do sistema e obter um novo diagrama apresentado na Figura 18.

Figura 18 – Fluxo baseado na arquitetura do sistema



Fonte: Produção do próprio autor.

4 EXPERIMENTOS E RESULTADOS

Foram utilizados para os experimentos, vídeos gravados durante diferentes períodos na mesma localidade, para que fosse possível aumentar a quantidade de detecções em um curto período para os testes sem a necessidade de estar no local de teste. Porém, para análise final deste trabalho, o sistema permaneceu executando por 7 dias durante 24 horas, totalizando 168 horas de processamento. Ao término de cada um dos dias de expediente de serviço do edifício analisado, foi calculada a diferença entre o número de pessoas que foram identificadas entrando no ambiente e o número de pessoas que foram identificadas saindo dele. Esse experimento utilizou o sistema proposto por completo, de forma a não utilizar gravações de terceiros, apenas imagens digitais vindas através das câmeras de segurança utilizando o protocolo RTSP.

Na Figura 19 é mostrada a interface gráfica do módulo Contador de Pessoas no modo de desenvolvedor, onde podem ser observadas as *bounding boxes* azuis (adicionada pela YOLO indicando a detecção de pessoa) e brancas (adicionadas pelo DeepSORT mostrando uma pessoa sendo rastreada). É possível perceber uma diferença nítida entre a detecção e o rastreamento. Além disso, podem-se observar os caminhos percorridos (gerados em cores aleatórias, por exemplo, na *Camera number* 0 que tem uma cor lilás para um caminho percorrido e o outro com um tom verde) por pessoas enquanto localizadas nas imagens (*frames*). Em verde, na parte inferior das *bounding boxes*, pode ser observado que o objeto detectado em questão é uma pessoa (classe informada pela rede neural) e sua respectiva confiança.

Figura 19 – Interface do Contador de Pessoas no modo desenvolvedor

Fonte: Produção do próprio autor.

4.1 Equipamentos Utilizados

Os experimentos foram realizados em uma máquina física contendo sistema operacional Windows 10 com as seguintes especificações:

- Processador: Intel Core i7 8700;
- Memória RAM: 32 GB;
- 2x Placas de Vídeo: GTX 1080 TI;
- Armazenamento: SSD Corsair Force LE 240 GB + HD 2 TB;
- Ambiente para execução e desenvolvimento das linguagens Python, PHP e SQL instalados.

As 6 câmeras de segurança utilizadas possuíam as seguintes especificações:

- Fabricante: Intelbras;
- Modelo: VIP S3020 G3;
- Tecnologia IP com suporte a *Power over Ethernet* (PoE);
- Tecnologia IR;
- Sensor: 1/4’;
- Lente: 2,6 mm;
- Resolução: 1 MP;
- Proteção: IP67.

4.2 *Intersection over Union* (IoU)

Uma das formas de comparar as relações entre duas localizações é por meio da métrica IoU que computa a razão entre a intersecção e união das áreas de duas *bounding boxes*. A métrica foi popularizada no contexto de visão computacional a partir do desafio Pascal VOC (EVERINGHAM et al., 2009).

O valor de IoU é dado pela equação (1).

$$IoU = \frac{area(b1 \cap b2)}{area(b1 \cup b2)} \quad (1)$$

Onde *b*1 e *b*2 representam duas diferentes *bounding boxes*, sendo uma do objeto detectado pelo modelo e a outra marcada manualmente. Esse valor então é comparado a um valor mínimo definido (IoU_{min}), e se $IoU \geq IoU_{min}$, considera-se que as duas *bounding boxes* correspondem ao mesmo objeto.

4.3 Supressão dos Não Máximos

O caso de detecções repetidas pode ser minimizado utilizando a técnica chamada de Supressão dos Não Máximos (NMS, do inglês *Non-maximum Suppression*). Dado um conjunto de localizações que descrevem uma mesma ocorrência de um objeto, com valores de confiabilidade associados a cada localização, apenas a *bounding box* com maior confiabilidade é mantida.

Dessa forma, é possível limitar um objeto a ter no máximo uma detecção associada a ele e a mesma será a que apresentar uma maior confiabilidade.

4.4 Critérios Utilizados na Detecção

Uma detecção de objeto está ligada diretamente à sua posição em relação à imagem a ser analisada.

Se uma detecção for comparada às localizações de referência da base de dados de validação e não obter um valor mínimo de sobreposição, considera-se que a localização não corresponde a um objeto da classe detectada. Se mais de uma detecção obter sobreposição acima do valor mínimo em relação a uma mesma referência, considera que ocorreram detecções repetidas. E, caso nenhuma detecção obter a sobreposição mínima em relação a uma referência específica, ocorreu falha na detecção. A avaliação do sistema de detecção de objetos depende de uma métrica que considere detecções corretas, falsas detecções e falhas de detecção.

A mAP é uma métrica popularizada no contexto a partir do desafio Pascal VOC (EVERINGHAM et al., 2009). O seu cálculo é realizado através das medidas de precisão, obtidas ao aplicar um sistema de detecção em imagens de teste que possuem seus objetos

anotados. Entretanto, para este trabalho está sendo utilizada apenas uma classe de objetos, a de pessoas. Com isso, o mAP não é um bom parâmetro de análise. Em seu lugar, serão utilizadas outras medidas que são comumente utilizadas: precisão e *recall*.

Por definição, a soma da quantidade de detecções corretas e detecções falsas resulta no número total de detecções. A partir da quantidade de detecções de cada tipo observada em uma imagem, a medida de precisão é calculada como a relação entre a quantidade de detecções corretas e o número total de detecções. Já a medida do *recall* é calculada como a razão entre detecções corretas e o número de pessoas na imagem. O cálculo da precisão não leva em consideração falhas na detecção e o cálculo do *recall* não leva em consideração detecções falsas (DAVIS; GOADRICH, 2006).

O valor de precisão é dado pela equação (2).

$$precisão = \frac{VP}{VP + FP} \tag{2}$$

Já o valor de *recall*, pode ser encontrado pela equação (3).

$$recall = \frac{VP}{VP + FN} \tag{3}$$

Onde os verdadeiros positivos (VP) são as detecções corretamente rotuladas, falsos positivos (FP) são as detecções incorretamente rotuladas e falsos negativos (FN) são as pessoas que não foram detectadas.

Os valores de precisão e *recall* serão relevantes para a análise de qualidade do sistema.

4.5 Base de Dados de Validação

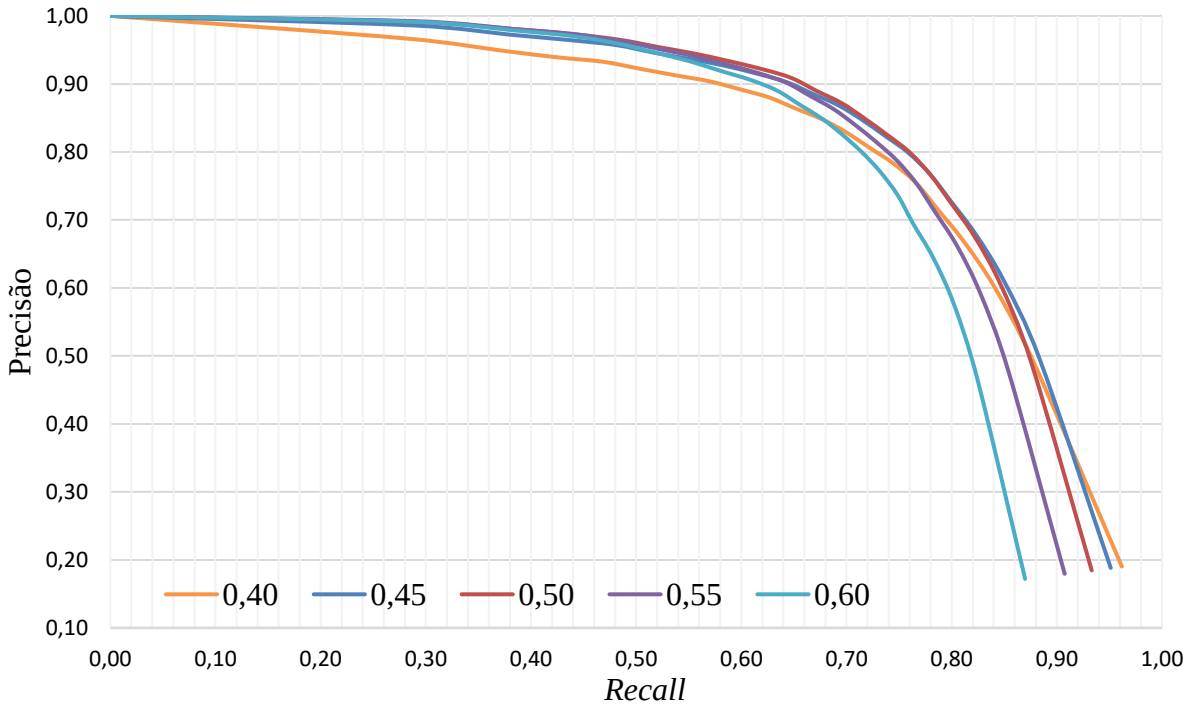
Para validar o funcionamento e desempenho do modelo previamente treinado para detecção de pessoas no sistema foi utilizada a base de validação de dados do MS COCO (LIN et al., 2014),

que apresenta uma grande quantidade de imagens digitais de pessoas e suas respectivas localizações marcadas de forma manual a serem comparadas com as geradas pela rede neural.

Pelo Gráfico 1 pode ser observada a curva de precisão em relação ao *recall* para diferentes valores de IoU. As curvas foram geradas variando o valor de *confidence threshold* (valor mínimo de confiança que o modelo de rede neural YOLO tem de um objeto detectado) de 0 a 1 em intervalos de 0,05.

Para conseguir uma melhor relação de precisão e *recall*, foi realizada uma análise no Gráfico 1 que apresenta essa relação para diferentes valores de IoU. Com isso, foi escolhida a curva de IoU = 0,50, na qual se obteve melhores resultados. Após isso, foi identificado um ponto que otimizasse a relação deles, sendo esse próximo à região na qual a curva começa apresentar uma queda. Sendo assim, foi adotado um valor de *Confidence threshold* de 0,50.

Gráfico 1 – Curva PR para diferentes valores de *IoU*



Fonte: Produção do próprio autor.

Foi escolhida a curva PR (Precisão x *Recall*) para realizar essas definições, baseado nos estudos de Saito e Rehmsmeier (2015), nos quais apresentam uma análise comparativa entre os gráficos

Receiver Operating Characteristic (ROC) e PR. Em seus estudos, tais autores afirmam que a plotagem de PR permite uma avaliação precisa e interpretação intuitiva do desemprenho prático do classificador.

4.6 Resultados

A Figura 20 mostra dois *frames* nos quais existem pessoas transitando na frente das câmeras. Conforme pode ser observado, para pessoas próximas à câmera (entre a reta de delimitação de saída ou entrada e a base inferior da imagem), a YOLO as detectou com alta porcentagem de confiança (acima de 99%). Mesmo para pessoas mais afastadas (como a pessoa de fundo no *frame* à direita), a confiança de detecção retornada é elevada. Este mesmo comportamento foi observado em vários outros *frames*. Estes resultados indicam que a YOLO conseguiu detectar de forma confiável as pessoas que entravam e saíam do ambiente. O valor médio de FPS foi de 19. Lembrando que, a captura de *frames* fornecidos pelas câmeras é realizada de forma assíncrona e com isso alguns *frames* não são analisados, mantendo assim o sistema em tempo real.

Em nenhum momento durante os testes foram identificados uma falsa detecção, sendo assim, nenhum objeto diferente de uma pessoa foi rotulado como pessoa. Isso não garante que a rede neural não possa cometer esse tipo de erro, pois o ambiente analisado é controlado e tende a ter trânsito de poucos tipos de objetos que poderiam ser confundidos pela rede.

Figura 20 – Exemplos de confiança de detecção

Fonte: Produção do próprio autor.

Os dados da análise realizada pelo sistema no prédio de teste podem ser observados no Quadro 1, no qual estão apresentados os erros em relação ao fluxo de pessoas do dia. O fluxo total de

pessoas no período de 30/07/2019 a 07/08/2019 (com exceção dos dias não úteis) foi de 1161 pessoas no edifício de acordo com a ferramenta. Em todos os dias dos testes foram realizadas contagens manuais verificando a quantidade de pessoas ainda presentes no interior da edificação, a fim de conferir se o sistema continha informações condizentes com a realidade, que é do prédio estar vazio ao final do expediente. Com isso, foi possível perceber que no final do dia existiam pequenas diferenças em relação à quantidade real de pessoas no interior do edifício. Além disso, vale lembrar que podem ter acontecido falhas que foram supridas por outras falhas. Como, por exemplo, uma contagem de entrada que falhou, e uma contagem de saída que também falhou e que se anulam. Permanecendo, assim, a contagem aderente mesmo com a presença de falhas. Não foi possível mensurar essa quantidade de possíveis falhas no período de teste de 168 horas.

De acordo com o Quadro 1, pode-se calcular um erro médio de 2,70%, chegando a uma acurácia média do sistema de 97,30%. Se comparado com o trabalho de Cordeiro, Paula Filho, Ojeda e Valiati (2019), que obtiveram uma acurácia de 50% para identificação e contagem de pessoas, o sistema proposto demonstra ser mais adequado ao ambiente em que foi implementado, diferente do que ocorreu no caso do trabalho citado.

Além disso, é possível observar que o erro apresentado pelo sistema foi sempre positivo, sendo assim, uma contagem inferior de saídas em relação a entradas. Isso pode ocorrer, pois, a entrada de pessoas é realizada em diferentes períodos, entretanto, a saída é realizada de forma conjunta após o expediente de trabalho. Com isso, podem ocorrer um maior índice de oclusões e uma não contabilização correta de saídas.

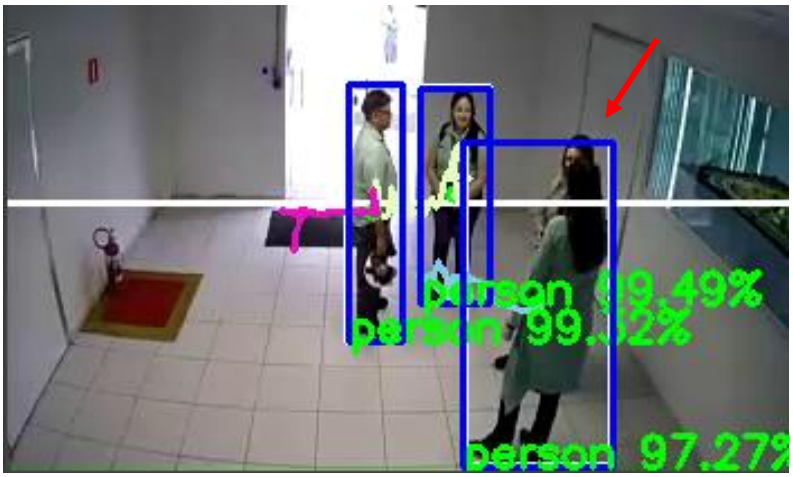
Pode-se observar na Figura 21 uma falha na detecção, no qual uma pessoa ficou oclusa atrás de outra. Infelizmente situações como essa pode acontecer. Nestes casos não é possível realizar o rastreamento das pessoas. Além disso, o NMS, que, em geral melhora a qualidade do *tracking*, pode prejudicar o sistema nestas situações, visto que ele tende a entender os dois objetos apenas como um. Para esse trabalho não serão analisadas formas de corrigir esse problema, podendo assim ser um trabalho futuro. Os ambientes analisados continham iluminação artificial, e com isso, não existiram problemas relacionadas a iluminação.

Quadro 1 – Resultados dos testes realizados com as 6 câmeras

Data e Hora	Diferença entre o número de entradas e saídas identificadas	Erro
30/07/2019 16:30	4	3,13%
31/07/2019 16:30	5	2,76%
01/08/2019 16:30	7	3,78%
02/08/2019 16:30	5	2,65%
05/08/2019 16:30	3	3,26%
06/08/2019 16:30	5	3,36%
07/08/2019	0	0%

Fonte: Produção do próprio autor.

Figura 21 – Exemplo de oclusão de uma pessoa



Fonte: Produção do próprio autor.

5 CONCLUSÕES

O objetivo principal desse trabalho foi implementar um sistema de contagem de pessoas em edificações utilizando uma rede neural e o rastreamento de objetos. Para isso, utilizou-se a YOLOv4 para realizar as detecções e o DeepSORT para o rastreamento.

A técnica proposta foi avaliada utilizando métricas e um modelo da YOLOv4 previamente treinado. Como resultado, as pessoas próximas às câmeras foram detectadas pela YOLO com uma confiança média superior a 95%, e a contagem de pessoas que entram e saem no recinto obteve uma acurácia média de 97,30%, quando utilizadas imagens dos locais de testes com as câmeras de segurança próprias em condições boas de iluminação. O referido valor demostra que o sistema tem potencial para ser usado no ambiente, embora seja desejado elevar o valor da acurácia.

Todavia, destaca-se que o ambiente usado para análise possui iluminação controlada e pouco fluxo de objetos que poderiam ser confundidos com pessoas pelo sistema. Em condições mais adversas do ambiente, como ofuscamento e falta de iluminação, podem ocorrer um desempenho menor do sistema. Uma situação que fez cair o desempenho do sistema foi quando ocorreu oclusão total e/ou parcial das pessoas. Por exemplo, duas pessoas ocupando regiões muito próximas, podem ser consideradas a mesma pessoa. Isso pode provavelmente ser evitado com o uso de uma estrutura física no ambiente a ser monitorado, para que apenas uma pessoa realize a passagem por vez.

Para trabalhos futuros é interessante mencionar que o desempenho do sistema em relação às múltiplas câmeras instaladas pode ser melhorado utilizando técnicas de reidentificação de pessoas, e com isso facilitar o sistema de rastreamento e contagem de pessoas. Além disso, a utilização de duas retas horizontais como limiar de decisão para identificar a entrada ou saída do ambiente pode evitar alguns erros de contagem identificados. Sendo assim, o sistema só iria realizar a contagem quando a pessoa ultrapassasse ambas as retas. Outra abordagem que pode ser adotada, é a integração com outros tipos de sistema de contagem de pessoas, constituindo assim um sistema hibrido.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMAD, F.; NING, L.; TAHIR, M. An Improved D-CNN Based on YOLOv3 for Pedestrian Detection. *In*: IEEE International Conference on Signal and Image Processing (ICSIP), 4., 2019, Wuxi. **Proceedings** […]. Wuxi: IEEE, 2019. p. 405-409. Disponível em: http://dx.doi.org/10.1109/siprocess.2019.8868839. Acesso em: 20 fev. 2021.

AXIS. **Contagem de pessoas**. 2019. Disponível em: https://www.axis.com/pt-pt/solutions-by-application/people-counting. Acesso em: 24 mar. 2019.

BARROS, N. **Análise crítica do sistema de saída de emergência aplicado no projeto de arquitetura: estudo de caso**. 2017. 116 f. Dissertação (Mestrado em Arquitetura e Urbanismo) – Curso de Arquitetura e Urbanismo, Programa de Pós-Graduação em Arquitetura e Urbanismo, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2017.

BARTHOLOMEW, D. **Getting started with MariaDB**. 2. ed. Birmingham: Packt Publishing Ltd., 2015. 140 p.

BATHIJA, A.; SHARMA, G. Visual Object Detection and Tracking using YOLO and SORT. **International Journal of Engineering Research & Technology (IJERT).** Mumbai, p. 705-708. 01 nov. 2019. Disponível em: https://www.ijert.org/research/visual-object-detection-and-tracking-using-yolo-and-sort-IJERTV8IS110343.pdf. Acesso em: 19 mar. 2021.

BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. 2020. Disponível em: https://arxiv.org/abs/2004.10934. Acesso em: 20 jun. 2020.

BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY. **CodeIgniter Overview.** 2019. Disponível em: https://codeigniter.com/userguide3/overview/index.html. Acesso em: 13 mar. 2021.

CORDEIRO, A. F.; PAULA FILHO, P. L.; OJEDA, C. A. U.; VALIATI, G. R. Rastreamento e contagem de pedestre em tempo real por meio de imagens digitais. *In*: CONGRESSO LATINO-AMERICANO DE SOFTWARE LIVRE E TECNOLOGIAS ABERTAS, 16.,2019, Porto Alegre. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 146-149. Disponível em: https://sol.sbc.org.br/index.php/latinoware/article/view/10350. Acesso em: 15 fev. 2020.

DAVIS, J.; GOADRICH, M. The relationship between Precision-Recall and ROC curves. *In*: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 23., 2006, Pittsburgh. **Proceedings** […]. New York: Association for Computing Machinery, 2006. p. 233-240. Disponível em: http://dx.doi.org/10.1145/1143844.1143874. Acesso em: 1 jun. 2021.

EDWARDS, C. **Deep Learning Hunts for Signals Among the Noise**, [s. l.], 2018. Disponível em: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852. Acesso em: 21 out. 2019.

EVERINGHAM, M; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The Pascal Visual Object Classes (VOC) Challenge. **International Journal of Computer Vision**, [s. l.], v. 88, n. 2, p. 303-338, 9 set. 2009. Disponível em: http://dx.doi.org/10.1007/s11263-009-0275-4. Acesso em: 13 jan. 2020.

FACURE, M. **Introdução às Redes Neurais Artificiais**, [s. l.], maio 2017. Disponível em: https://matheusfacure.github.io/2017/03/05/ann-intro/. Acesso em: 22 out. 2019.

GASPAR, I.; LEITÃO, K. B. M.; REZENDE, A. L. T. IMPLANTAÇÃO DE ATIVIDADES DE BRIGADA DE INCÊNDIO NA UNIDADE ESCOLAR. **Semioses**, [s. l.], v. 12, n. 1, p. 100-113, 2018. Disponível em: https://revistas.unisuam.edu.br/index.php/semioses/article/view/59/11. Acesso em: 5 jun. 2021.

GUO, J.; HAN, K.; WANG, Y.; ZHANG, C.; YANG, Z.; WU, H.; CHEN, X.; XU, C. Hit-Detector: Hierarchical Trinity Architecture Search for Object Detection. *In*: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2020, Seattle. **Proceedings** […]. Seattle: IEEE, 2020. p. 11405-11414. Disponível em: https://openaccess.thecvf.com/content_CVPR_2020/html/Guo_Hit-Detector_Hierarchical_Trinity_Architecture_Search_for_Object_Detection_CVPR_2020_paper.html. Acesso em: 21 fev. 2021.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [s. l.], v. 37, n. 9, p. 1904-1916, 1 set. 2015. Disponível em: http://dx.doi.org/10.1109/tpami.2015.2389824. Acesso em: 30 jan. 2020.

KLETTE, R. **Concise Computer Vision: an introduction into theory and algorithms**. Auckland: Springer, 2014.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, [s. l.], v. 521, n. 7553, p. 436-444, 27 mai. 2015. Disponível em: https://doi.org/10.1038/nature14539. Acesso em: 14 nov. 2020.

LI, X.; KARNAN, S.; CHISHTI, J. A. An empirical study of three PHP frameworks. *In*: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS AND INFORMATICS (ICSAI), 4., 2017, Hangzhou. **Proceedings** […]. Hangzhou: IEEE, 2017. p. 1636-1640. Disponível em: http://dx.doi.org/10.1109/icsai.2017.8248546. Acesso em: 20 jan. 2020.

LIN, T.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P; ZITNICK, C. L. Microsoft COCO: Common Objects in Context. *In*: EUROPEAN CONFERENCE ON COMPUTER VISION, 13., 2014, Zurique. **Proceedings** […]. Zurique: Springer, 2014. p. 740-755. Disponível em: https://link.springer.com/content/pdf/10.1007%2F978-3-319-10602-1_48.pdf. Acesso em: 14 jan. 2020.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path Aggregation Network for Instance Segmentation. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2018, Salt Lake City. **Proceedings** […]. Salt Lake City: IEEE, 2018. p. 8759-8768. Disponível em: http://dx.doi.org/10.1109/cvpr.2018.00913. Acesso em: 25 fev. 2021.

NOGUEIRA, A. B. **Análise de viabilidade no uso de Deep Learning para contagem de pessoas com câmeras de segurança**. 2018. 78 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Curso de Engenharia de Controle e Automação, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2018.

OUAKNINE, A. Review of Deep Learning Algorithms for Object Detection. *In*: Medium. **Zyl Story**. [s. l.], 5 fev. 2018. Disponível em: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852. Acesso em: 23 out. 2019.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016, [s. l.]. **Proceedings** […]. [S. l.]: IEEE, 2016. p. 779-788. Disponível em: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html. Acesso em: 13 dez. 2019.

REDMON, J.; FARHADI, A. **Yolov3: An incremental improvement**. 2018. Disponível em: https://arxiv.org/abs/1804.02767. Acesso em: 13 dez. 2019.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **Plos One**, [s. l.], v. 10, n. 3, p. 1-21, 4 mar. 2015. Disponível em: http://dx.doi.org/10.1371/journal.pone.0118432. Acesso em: 9 jun. 2021.

VARELLA, W. A. **Infraestrutura de TI**. Editora Senac São Paulo, 2019.

WANG, C.; LIAO, H. M.; WU, Y.; CHEN, P.; HSIEH, J.; YEH, I. CSPNet: a new backbone that can enhance learning capability of CNN. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS (CVPRW), 2020, Seattle. **Proceedings** […]. Seattle: IEEE, 2020. p. 1571-1580. Disponível em: http://dx.doi.org/10.1109/cvprw50498.2020.00203. Acesso em: 15 mar. 2021.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. *In*: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017, Beijing. **Proceedings** […]. Beijing: IEEE, 2017. p. 3645-3649. Disponível em: http://dx.doi.org/10.1109/icip.2017.8296962. Acesso em: 26 mar. 2021.