

Detecção de ação utilizando câmeras de segurança

1st Ananda Mendes Souza

departamento de computação

Universidade Federal de Ouro Preto

Ouro Preto, Brasil

2nd Maria Laura Moreira Raimundo

departamento de computação

Universidade Federal de Ouro Preto

Ouro Preto, Brasil

3rd Rafael Coelho Monte Alto

departamento de computação

Universidade Federal de Ouro Preto

Ouro Preto, Brasil

Abstract—Atualmente cerca de 12 milhões de imóveis no Brasil contam com câmeras de vigilância, o principal motivo é a questão da segurança que o monitoramento traz visto que proporcionam o monitoramento e gravação (na maioria dos casos) do seu ambiente e em alguns casos a própria câmera tem sensor de perigo e avisa aos órgãos responsáveis, além disso as gravações são documento de provas para a polícia. No âmbito do transito as aplicações móveis colaborativas têm ajudado condutores de veículos de diversas formas. Dito isso, este trabalho propõe a detecção de movimento e classificação automóveis e pessoas, assim como outros objetos capturados em cenas.

Index Terms—câmeras de vigilância, automóveis, monitoramento, pessoas, detecção, movimento

I. INTRODUÇÃO

As câmeras de vigilância tem inúmeras funções, principalmente para segurança, porém se mostraram ótima ferramenta no período de pandemia da covid-19, uma vez que é possível fazer a contagem de pessoas, bem propício ao momento em que locais não podem atingir sua lotação máxima devido a pandemia. Nesse caso, é possível criar a regra e as câmeras de segurança auxiliam para esse monitoramento.

O enfoque deste trabalho é a detecção de movimentos e objetos. Aplicações nesta área têm demandado tarefas como contagem, determinação da velocidade, classificação e rastreamento de veículos, avaliação do tráfego, detecção de acidentes, entre outros. Esses desafios possuem características muito particulares em relação à vigilância eletrônica em geral. As características geométricas dos veículos e da estrada são informações valiosas para buscar a detecção automática dos mesmo.

Este trabalho foi desenvolvido com o objetivo de buscar soluções simples e eficientes para resolver algumas tarefas de vigilância , além de sua detecção, a classificação de objetos. Utilizando técnicas como, tratamento das imagens tirando ruídos, extraíndo pontos de interesse e segmentação de imagens baseada em subtração de fundo, utilizando o framework OpenCV para o processamento das imagens e codificado na linguagem Python.

II. TRABALHOS RELACIONADOS

O artigo [1] propõe um método de detecção e reconhecimento de ações de vários indivíduos em um mesmo vídeo de vigilância. Uma série de treinamento de Histogramas de

Gradiente Orientado, Histograms of Oriented Gradient (HOG), são usados para representar as categorias de ações humanas.

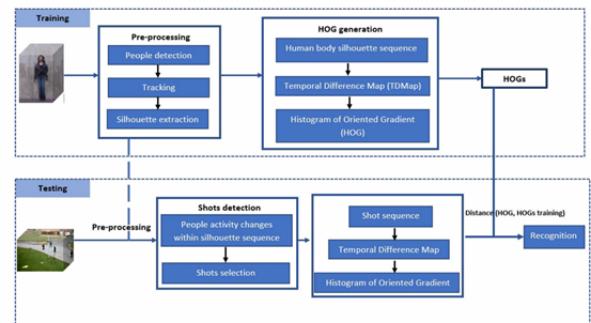


Fig. 1: Treinamento e Aplicação. Fonte: Autor

Os corpos são detectados usando um algoritmo de subtração de fundo de vídeo (background-subtraction-based approach). Então, cada pessoa tem suas ações analisadas separadamente. São chamados de shots as partes homogêneas dos movimentos da pessoa. Essas partes homogêneas são reconhecidas a partir de um algoritmo de similaridade de histograma de frames (similarity histogram between frames). Os picos dos histogramas são momentos de transição e os shots são captados nos momentos entre pares de picos.

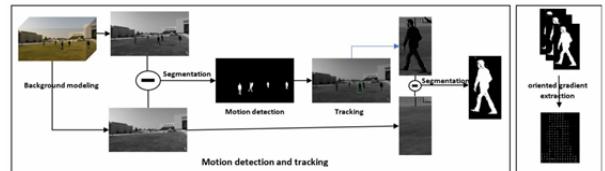


Fig. 2: Subtração de fundo e detecção de sujeitos. Fonte: Autor

Em seguida, cada shot é usado na identificação de ação baseado nos HOGs. A classificação da ação pode ser feita a partir de dois métodos: (1) comparação e análise da similaridade da medida do cosseno do HOG da ação atual com o da série de treinamento. (2) utilização de um modelo de rede neural convolucional, convolutional neural network (CNN) model.

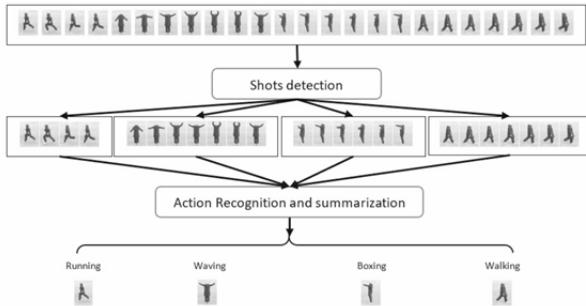


Fig. 3: Separação em shots e reconhecimento da ação. Fonte: Autor

O artigo [2] refere a uma aplicação para monitoramento da velocidade de veículos em tempo real a partir de imagens obtidas de câmeras de segurança convencionais. Para auxiliar no reconhecimento dos veículos, foram aplicadas algumas técnicas de processamento de imagens utilizando a biblioteca OpenCV.

A primeira parte no tratamento das imagens foi feita a partir da subtração do fundo, para isso foi feita a diferença entre duas matrizes. Para cada frame do vídeo é retirada a imagem da via sem nenhum objeto, facilitando o reconhecimento dos veículos e pedestres que estão trafegando na via.



Fig. 4: Imagem utilizada na subtração do fundo - Fonte: Autor

Na segunda parte foi feita a redução de ruídos, retirando os objetos desnecessários, isso foi feito com o apoio da função GaussianBlur. Na terceira parte, por feita a binarização da imagem, ou seja, ao invés de tons de cinza a imagem passa a ter apenas duas cores preto e branco.



Fig. 5: Imagem resultante binarizada - Fonte: Autor

A dilatação é a última parte utilizada no tratamento das imagens. Após binarizar a imagem é utilizada a operação morfológica de dilatação a fim de juntar os “pedaços” do objeto encontrado, para que se torne um objeto único e consequentemente poder classificá-lo como um veículo e posteriormente medir sua velocidade.

Após o processamento das imagens foi feito o cálculo de velocidade e a seleção dos pontos de interesse. Com isso foi possível realizar a classificação dos objetos de interesse presentes na via.

O artigo consegue detalhar o processo de tratamento de imagens com uma riqueza de detalhes, porém o monitoramento se desenvolve sob uma perspectiva apenas.

O estudo [3] apresenta um método de identificar eventos e classificá-los como normal ou anormal. Um comportamento

anormal pode significar risco ou ameaça a outras pessoas. Os autores seguiram a seguinte linha de análise: Fragmentação da Cena em Frames, Extração de Fundo, Extração do Primeiro Plano, Detecção de Movimento e Classificação de Ação.



Fig. 6: Passos do método. Fonte: Autor

Trabalhos correlatos foram analisados brevemente no artigo quanto a diferentes métodos de classificação. Os dois métodos mais populares são: modelos de recursos artesanais, hand-crafted features based models, e modelos de deep learning. Existem bancos de dados públicos usados para detecção de anomalias em vídeos, classificados em violência ou interação. Dois dos bancos de dados de violência mais famosos são os de luta em filme (movie-fight dataset) e de luta de hockey (hockey fight dataset). O primeiro contém mais de 200 clipes de cenas de filmes de ação em diversas resoluções. O segundo conta com mais de 1000 vídeos de jogos de hockey classificados se há luta ou não. Agora quando falamos de bancos de dados de interação, existem o UCSD dataset, The Avenue dataset, The Subway dataset, etc.

O foco do artigo [4] são as detecções de armas, incêndios e pichações. Para isso utiliza-se da ferramenta YoloV5 na detecção de objetos pequenos treinando em datasets distintos de tal modo a verificar a possibilidade de utilização do Yolo na análise de incidentes em tempo real.

Os datasets utilizados para detecção de armas foram o Granada (dataset com quase 3 mil imagens de pistola), o U500 para pré-treino da rede, formado por 500 imagens de pistolas e rifles criados artificialmente no Unity. Já para o fogo foi o FiSmo e é composto da união de outros datasets junto de imagens de Flickr, Youtube e simulações, contém 1604 imagens contendo fogo e 3952 não contendo fogo para fins de treinamento. E para o Grafite foi utilizado o FormasGraffitiDataset, com 516 imagens extraídas de vídeos de bombing ou de vídeos contendo grafite legal extraídos frame a frame do Youtube e selecionados manualmente. Os frames foram divididos em 3 classes: Spray, Tagger e o Grafite.

A principal vantagem deste artigo é o treinamento com diferentes datasets e a conexão com centrais de monitoramento dos bombeiros e polícia para alarmes em caso de ocorrência. Porém, há necessidade de aprimoramento na precisão das detecções em imagens reais, já que muitos datasets são canônicos (colocando os objetos em posição de destaque), o que não reflete as situações cotidianas.

O objetivo do artigo [5] é a detecção de anomalias a partir de um dataset de vídeos. Ao todo, foram coletadas 128 horas de duração. No artigo, as anomalias sinalizam cenas de roubo, brigas, acidentes de carros e etc. Ao todo, obteve-se 13 tipos de anomalias. Os resultados foram obtidos através de uma técnica de aprendizado supervisionado chamado de

MIL (multiple-instance learning). Para o pré-processamento, foi feita a rotulação de cada anomalia presente nos vídeos fornecidos para teste.

O passo seguinte consistiu em dividir o conjunto de dados em treino e teste. Dentro do artigo é apresentado o modelo matemático utilizado para realizar o treinamento da rede. Além disso, os resultados são comparados com técnicas similares de alta qualidade.

A forma como é feita a classificação das cenas, deixou pouco evidente a forma como a fase de rotulação das anomalias foi realizada. Por outro lado, o passo a passo da formulação do algoritmo de IA bem como a fase de testes e comparações são bem descritas. No que se refere ao YOLO, o artigo [6] é uma boa referência e trata de um tema bastante atual que é o Covid, trazendo outra utilidade para as câmeras de segurança como elemento de ajuda na saúde. O algoritmo busca capturar as áreas de interesse, que no caso são os rostos, e detectar se as pessoas estão usando máscara ou não. O estudo obteve uma acurácia de 62% com 144 frames por segundo.



Fig. 7: Identificando o uso de máscaras. Fonte: Autor

O artigo [7] o possui como objetivo utilizar imagens de circuitos internos de TV para rastrear o posicionamento de partes do corpo de humanos durante a execução do vídeo, modelar seu comportamento e interpretar as ações realizadas. Além disso, apresenta uma nova base de dados, com o foco inteiramente em cenas com violência registradas em casas lotéricas do Brasil, que conta com aproximadamente 47.280 quadros com e sem violência. Por fim, é apresentada a performance do método baseado em pose track na base de dados construída.

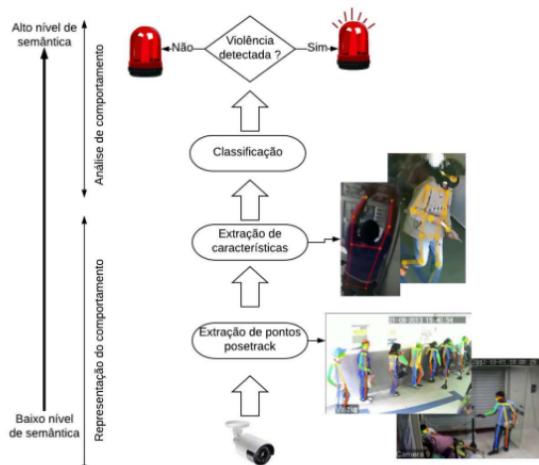


Fig. 8: Processo metodológico para classificação de violência

A metodologia do artigo, propõe a utilização de Pose Flow associado com Pose Tracking para detectar e rastrear humanos e a posição dos membros de seus corpos durante o processamento do vídeo em tempo real. Assim, extraíndo características sobre as ações realizadas por estas pessoas, sendo possível modelar um contexto sobre cena e identificar ações suspeitas, não necessariamente contendo violência. Na prática, utilizando estes algoritmos na base proposta, conseguiremos extraír pontos sobre a postura de várias pessoas em cena e rastreá-las pelo vídeo. Esta extração será armazenada em um arquivo contendo todos os pontos do seus membros durante a execução do vídeo. Posteriormente, serão classificadas manualmente por pessoas, onde, para cada humano detectado no vídeo, existem pontos distintos.

A partir da classificação é possível concluir se trata-se de violência ou não.

O trabalho [8] apresenta o uso de dois modelos de redes neurais trabalhando em conjunto a fim de efetuar a tarefa de detecção de pedestres, rastreamento e contagem por meio de imagens digitais.

O processo de detecção de pedestres ficou a cargo da rede YOLO (You-Only-Look-Once), que utiliza a rede Darknet-53 como *backbone* à rede YOLO realiza a detecção por meio do uso de *Bounding Boxes* (BB), caixas delimitadoras que são espalhadas pela imagem a fim de obter características locais do objeto a ser reconhecido, assim obtendo probabilidade de tais características pertencerem às classes informadas previamente no treinamento.

A rede de rastreamento proposta, utiliza uma abordagem por meio da aprendizagem de filtros de correlação discriminativos, utilizando uma estimativa de escala genérica, assim possibilitando sua aplicação em diversos métodos de rastreamento. Foram repassadas as coordenadas das detecções para uma rede de rastreamento, possibilitando definir sua trajetória e assim efetuar a contagem referente a entrada ou saída do pedestre de um determinado local.

III. MODELO PROPOSTO

Desenvolvemos dois modelos para o problema de detecção de ação em câmeras de segurança. O primeiro utiliza de uma biblioteca chamada OpenCV para encontrar, em um vídeo, espaços, blocos ou elementos que estejam se movimentando. O segundo modelo proposto é mais complexo e traz uma classificação dos objetos que estão se movimentando na cena.

A. Primeira Modelo

Para começar a entender o funcionamento do OpenCV e o objetivo do trabalho, começamos implementando um programa simples em python. Funções e conceitos utilizados serão explicados a seguir:

1) OpenCV

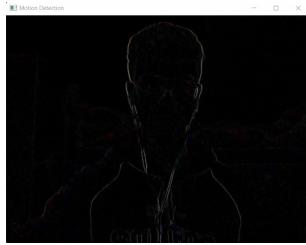
OpenCV (Open Source Computer Vision) é uma biblioteca de programação, de código aberto e inicialmente desenvolvida pela Intel com o objetivo de tornar a visão computacional mais acessível a desenvolvedores e hobistas. Atualmente possui mais de 500 funções, pode ser utilizada em diversas linguagens

de programação (C++, Python, Ruby, Java...) e é usada para diversos tipos de análise em imagens e vídeos, como detecção, tracking e reconhecimento facial, edição de fotos e vídeos, detecção e análise de textos, etc.

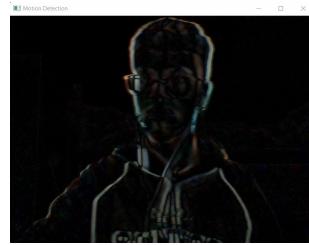
Por ter uma documentação simples e fácil de entender, por encontrar muitos exemplos de utilização das funções, por esses motivos, foi determinado que seria utilizada neste projeto.

2) Captura de vídeo e tratamento da imagem

A captura de vídeo da webcam é dada usando a função VideoCapture do OpenCV e o parâmetro correspondente 0. A função read é usada para extrair uma imagem dentro do vídeo referenciado. Definimos, então, dois frames (1 e 2) que serão utilizados para comparação ao decorrer da execução do programa.



(a) Sujeito imóvel.



(b) Sujeito em movimento.

Fig. 9: Diferença absoluta.

Para tratar a imagem extraída do vídeo, aplicamos uma série de funções. Uma delas é a diferença absoluta dos dois frames. A função absdiff irá calcular a diferença entre as duas imagens ou matrizes, retornando um resultado. Logo em seguida utilizamos da função cvtColor e o parâmetro COLOR_BGR2GRAY para transformar a imagem resultante em escala de cinza.



(a) Sujeito imóvel.



(b) Sujeito em movimento.

Fig. 10: Imagem em escala preto e branco.

3) Gaussian Blur

Um desfoco é aplicado para facilitar a detecção do objeto que se movimenta. A função GaussianBlur implementa a "técnica de borrar" através de uma função de Gauss, onde é calculada a média ponderada do pixel com relação aos seus pixels vizinhos.



Fig. 11: Ilustração da função GaussianBlur. Fonte: Ilustração adaptada da Documentação do OpenCV 3.0.0-dev.

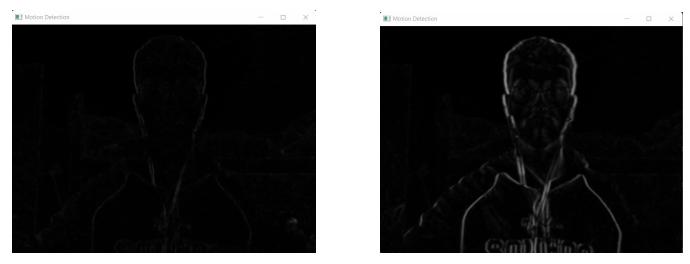


Fig. 12: Gaussian Blur.

Cheung e Kamath (2004) classificaram as técnicas de modelagem de fundo, em recursiva e não-recursiva. Modelagem de fundo é a base da subtração de fundo, é nesta etapa que a imagem de referência é construída.

- Modelagem Recursiva: As técnicas recursivas não armazenam um buffer para estimar o fundo, o modelo de fundo é atualizado recursivamente através de cada frame de entrada. Por não armazenar buffer não é necessário ter grande capacidade de memória, porém se houver um erro de modelagem, este pode alterar os resultados por um longo período de tempo.
- Modelagem Não-Recursiva: As técnicas não-recursivas armazenam uma determinada quantidade de imagens no buffer e estima o fundo baseado na variação temporal de cada pixel dentro do buffer.

4) Threshold

A função Threshold ou limiarização, tem como objetivo separar os pixels em duas regiões através de um limiar, a região de interesse, ou seja, a região onde os objetos de interesse se destacam e a região de contraste com a região anterior.

A imagem é representada por uma matriz $I(i,j)$ onde i é o número de linhas e j é o número de colunas, cada elemento da matriz representa a cor de um pixel, que pode assumir o valor 0 ou 1 para imagens binárias, de 0 a 255 para imagens em tons de cinza e três valores de 0 a 255 para imagens coloridas, esse processo também é conhecido como binarização.

Com o resultado do filtro de ruídos, as imagens foram binarizadas, ou seja, ao invés de tons de cinza a imagem passa a ter apenas duas cores preto e branco, para isso foi utilizada a

propriedade cv2.THRESH_BINARY. A função cv2.threshold realiza a binarização da imagem, mas para que isso seja possível é necessário que a imagem esteja em tons de cinza.

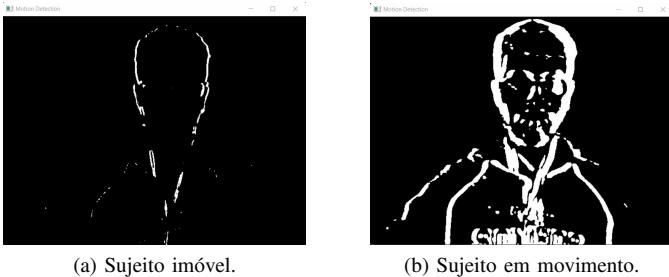


Fig. 13: Threshold.

5) Dilatação

A dilatação é o ultimo passo no tratamento das imagens. Após binarizar a imagem é utilizada a operação morfológica de dilatação a fim de juntar os “pedaços” do objeto encontrado e para preencher todas as lacunas, ajudando a reconhecer os melhores contornos da imagem.

A dilatação é um conjunto de todos os elementos x , de forma que as translações da reflexão de B por x , sobreponham-se ao conjunto A por pelo menos um elemento não-nulo.

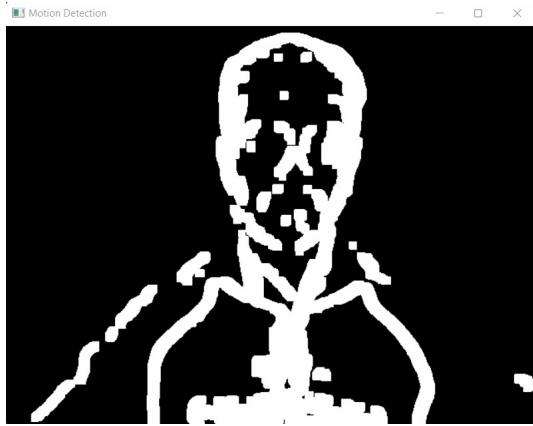


Fig. 14: Operação de Dilatação.

6) Detecção de Contornos

Neste ponto do processamento os objetos se movimentando serão detectados com a ajuda dos seus contornos. A função findContours se encarrega de encontrar contornos presentes na imagem. Contornos podem ser descritos como curvas ligando pontos contínuos que possuem mesma cor ou intensidade.

O próximo passo é iterar esses contornos e filtrar os indesejados. Contornos com áreas muito pequenas são vistos como ruídos e são ignorados. O restante é validado e destacado usando um retângulo que será desenhado na tela.

A junção de todas as operações descritas nesta seção resultam em um algoritmo capaz de identificar elementos se movendo no vídeo de uma webcam. Tais elementos são

identificados na tela e uma mensagem atualizando o status do vídeo também aparece.

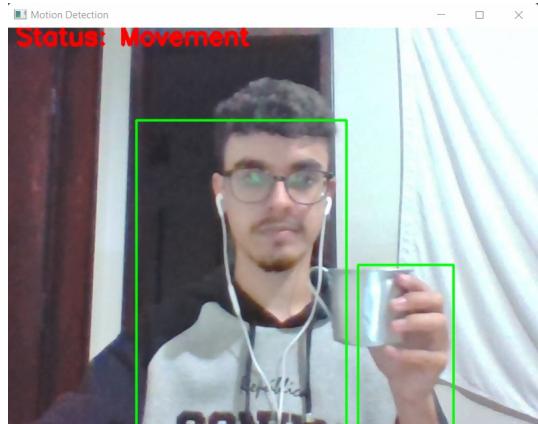


Fig. 15: Detecção de movimento.

B. Segundo Modelo

O segundo modelo proposto trabalha com o Yolo (You Only Look Once). YOLO é um método de detecção de objetos de passada única (single pass) que utiliza uma rede neural convolucional como extrator de características (features).

Após o seu lançamento em 2015, o YOLO foi logo reconhecido como uma técnica inovadora pois através de uma abordagem totalmente nova foi capaz de obter uma precisão igual ou superior ao dos outros métodos de detecção de objetos da época, porém com uma velocidade de detecção muito superior.

1) Funcionamento do Yolo

Para o seu funcionamento o YOLO utiliza uma rede neural profunda, cuja arquitetura é chamada de Darknet, que é o mesmo nome do framework utilizado para implementar o detector. Esse framework foi desenvolvido pelo próprio criador do YOLO, Joseph Redmon.

O YOLO trata a detecção de objetos como um simples problema de regressão.

Primeiro, o algoritmo divide a imagem em um grid de $S \times S$ células. Cada uma dessa células é responsável por fazer a predição de, por padrão do Yolov3, 3 caixas delimitadoras (B), para caso haja mais de um objeto naquela célula. Também é retornado a pontuação de confiança que diz o quanto de certeza ele tem que aquela caixa delimitadora contenha um objeto, YOLOv3, no total, usa 9 caixas de âncora. Três para cada escala. Para cada caixa, a célula também faz a previsão de uma classe. Isso funciona como se fosse um classificador: é fornecido um valor de probabilidade para cada uma das classes possíveis. O valor de confiança para a caixa delimitadora e a predição da classe são combinados em uma pontuação final, que vai dizer a probabilidade dessa caixa conter um objeto específico. Acontece que a maioria dessas caixas terá um valor de confiança extremamente baixo, então por isso geralmente se considera apenas as caixas cuja pontuação final seja 30% ou mais, esse valor de 30% é o threshold.

2) Métodos do Yolo

A função de configuração é usada para definir os caminhos para os pesos YOLO, arquivo .cfg. O método cv2.dnn.readNetFromDarknet é usado para carregar os pesos salvos na rede. Depois de carregar os pesos, extraia a lista de todas as camadas usadas em uma rede usando um modelo net.getLayerNames.

Dentro do loop de processamento de imagem, pega-se um único quadro de vídeo e o processa para detecção. Inicialmente defini-se as dimensões do quadro de vídeo (W, H). Depois, usa-se o método cv2.dnn.blobFromImage para carregar frames em um lote e executá-los pela rede. A função blob realiza uma subtração média, escala e troca de canal em um quadro.

As saídas da camada do YOLO consistem em um conjunto de valores. Esses valores ajudam a definir qual objeto pertence a qual classe. De cada detecção, obtém-se uma caixa delimitadora com o centro X, o centro Y, a largura e a altura da caixa para detecção na saída.

IV. EXPERIMENTOS

Os dados de entrada utilizados foram 4 vídeos de câmeras de segurança encontrados em bancos de dados na internet. Cada vídeo mostra um ambiente diferente: uma rodovia movimentada, um shopping, uma escola e uma rua. Uniformizamos a duração de todos os vídeos de entrada, deixando-os com 5 segundos. Dessa forma os experimentos a serem realizados a seguir seriam coerentes.

Analisamos o tempo de duração do processamento de cada vídeo de entrada e notamos que ele mudava a cada vez que o programa era acionado. Rodamos o programa, então, 5 vezes para cada vídeo e guardamos a média do tempo de duração. Vários fatores poderiam influenciar esse tempo de processamento, como as especificações técnicas de hardware da máquina utilizada ou quantidade ou tamanho dos objetos detectados no vídeo.

A saída do algoritmo era um arquivo csv que descrevia todos os objetos classificados em cada frame, assim como a precisão dessa classificação. Outra saída era um vídeo mp4 que mostrava, frame pro frame, as regiões onde se encontravam cada objeto detectado. A segunda saída foi desconsiderada nos experimentos por demandar bastante tempo de processamento e não ser relevante quando estamos falando do yolo e de detecção de objetos por si só.

Para cada arquivo csv gerado, extraímos todos os tipos de objetos que foram detectados na cena, assim como uma média da acurácia de cada elemento. As precisões máxima e mínima encontradas também foram ressaltadas. Essas informações foram tratadas com o Google Colab.

O YOLOv3 utilizou de um arquivo .names que listava todos os 80 tipos de objetos que poderiam ser encontrados e detectados com sucesso na cena. Entre eles: pessoa, bicicleta, carro, avião, ônibus, trem, barco, cachorro quente, bolo, sanduíche, cadeira, sofá, relógio e geladeira. Obtivemos os seguintes resultados:

- Rodovia: 888 objetos detectados.

Tempo médio: 40.45466 segundos

Acurácia média: 84.85%

Acurácia máxima: 99.74%

Acurácia mínima: 50.03%

Acurácia por tipo de objeto: Carro 87.35%; Pessoa 61.69%; Caminhão 58.04%

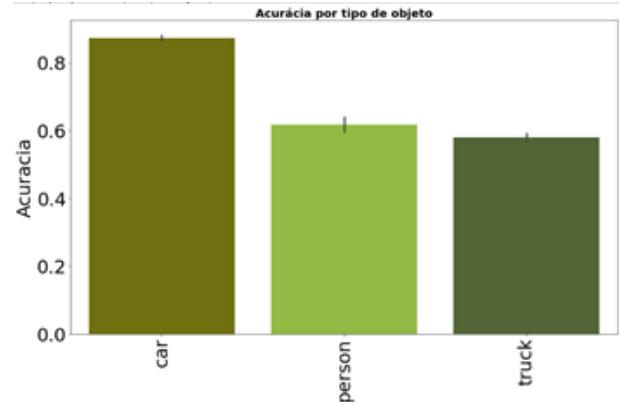


Fig. 16: Acurácia por tipo de objeto na rodovia. Fonte: Autor.

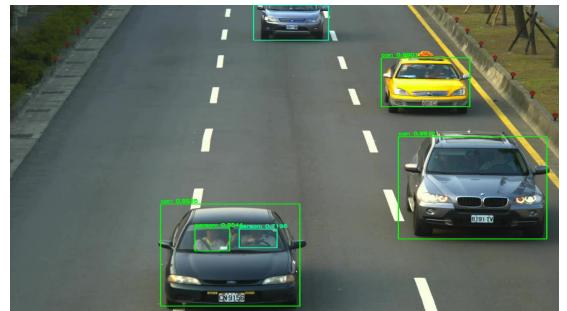


Fig. 17: Imagem retirada do vídeo de saída.

- Shopping: 3060 objetos detectados.

Tempo médio: 34.971032 segundos

Acurácia média: 79.12%

Acurácia máxima: 99.22%

Acurácia mínima: 50.01%

Acurácia por tipo de objeto: Pessoa 79.12%

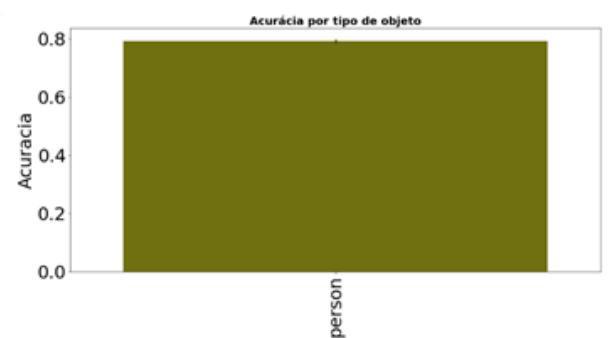


Fig. 18: Acurácia por tipo de objeto no shopping. Fonte: Autor.



Fig. 19: Imagem retirada do vídeo de saída.

- Escola: 727 objetos encontrados.
Tempo médio: 22.938556 segundos
Acurácia média: 86.19%
Acurácia máxima: 99.84%
Acurácia mínima: 50.02%
Acurácia por tipo de objeto: Mochila 60.57%; Parquímetro 50.27%; Pessoa 86.47%

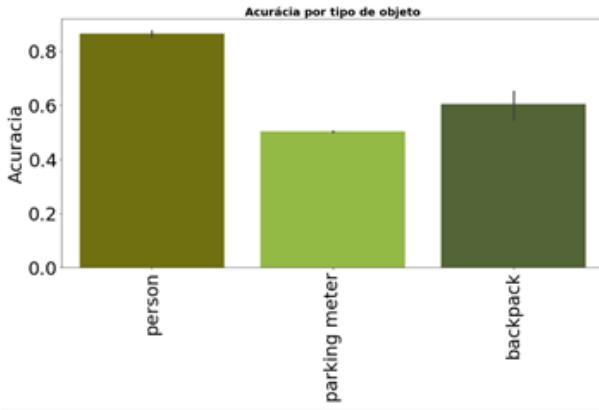


Fig. 20: Acurácia por tipo de objeto na escola. Fonte: Autor.

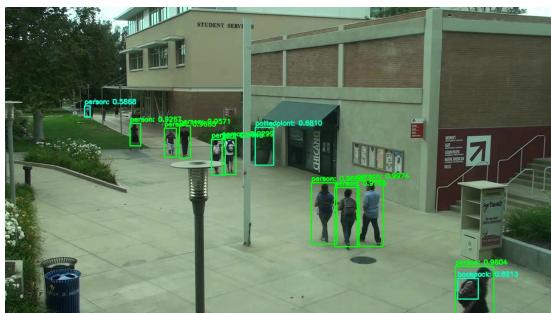


Fig. 21: Imagem retirada do vídeo de saída.

- Rua: 620 objetos encontrados.
Tempo médio: 38.355236 segundos
Acurácia média: 82.92%
Acurácia máxima: 99.99%
Acurácia mínima: 50.05%
Acurácia por tipo de objeto: Ônibus 57.09%; Carro 90.66%; Pessoa 76.90%; Caminhão 85.34%; Guarda-chuva 92.66%

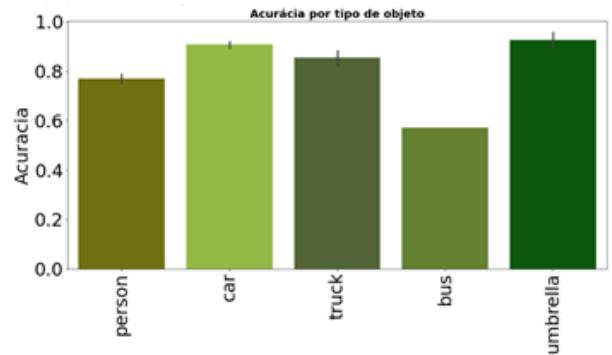


Fig. 22: Acurácia por tipo de objeto na rua. Fonte: Autor.



Fig. 23: Imagem retirada do vídeo de saída.

Para obter esses dados foi utilizado o arquivo .csv que o Yolov3 gera e foram tratados com o Google Colab.

V. CONCLUSÃO

O objetivo deste trabalho foi o desenvolvimento de uma aplicação capaz de medir detectar e classificar veículos e objetos através do uso de câmeras de segurança convencionais e técnicas de visão computacional.

A vantagem do YOLO frente aos outros métodos é que o mesmo faz as previsões da classe com uma única passada na rede. Diferente dele, os outros principais sistemas de detecção de objetos faziam a detecção através da divisão da imagem em várias partes e depois em cada pedaço da imagem se executava um classificador, em cada uma dessas regiões.

Comparando com esses outros métodos, o YOLO possui uma abordagem completamente diferente. Ele não é um classificador tradicional que foi reajustado para ser um detector de objetos, mas sim foi feito para ser capaz de rodar por toda a imagem apenas de uma vez, fazendo isso de uma forma inteligente. Desse modo, com o método YOLO é possível realizar a detecção em tempo real, ou quase real dependendo das configurações de hardware da máquina onde está sendo rodado.

Um diferencial do trabalho, é que a aplicação utiliza câmeras de segurança convencionais, por ter um valor baixo de aquisição, o projeto se torna viável de aplicação. Por realizar o monitoramento com imagens de câmeras de segurança convencionais também será possível realizar com câmeras mais sofisticadas.

Portanto, conclui-se o trabalho com sucesso, pois os objetivos propostos foram alcançados e apresentados como base para trabalhos relacionados ou para o aperfeiçoamento deste trabalho.

REFERÊNCIAS

- [1] ELHARROUSS, O. et al. A combined multiple action recognition and summarization for surveillance video sequences. *Applied Intelligence*, Springer, v. 51, n. 2, p. 690–712, 2021.
- [2] JUNIOR, J. T. Monitoramento veicular utilizando câmeras de segurança convencionais e visão computacional. 2016.
- [3] MAHDI, M. S.; MOHAMMED, A. J.; JAFER, M. M. Unusual activity detection in surveillance video scene. *Journal of Al-Qadisiyah for computer science and mathematics*, v. 13, n. 3, p. Page–92, 2021.
- [4] MOURA, N.; CLARO, D. B.; GONDIM, J. M. Análise experimental para a detecção de objetos em vídeos de câmeras de vigilância. 2021.
- [5] SULTANI, W.; CHEN, C.; SHAH, M. Real-world anomaly detection in surveillance videos. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.I.: s.n.], 2018. p. 6479–6488.
- [6] FIGUEIREDO, E. B.; SILVA, E. P. da. Combate ao covid19: Detecção em tempo real de indivíduos sem máscara em ambiente escolar por meio de deep learning. In: SBC. *Anais do XV Brazilian e-Science Workshop*. [S.I.], 2021. p. 113–120.
- [7] SOARES, P. G. S. d. C. *Um sistema para detecção de violência baseado em métodos de Pose Track*. Dissertação (B.S. thesis) — Brasil, 2019.
- [8] CORDEIRO, A. F. et al. Rastreamento e contagem de pedestre em tempo real por meio de imagens digitais. In: SBC. *Anais do XVI Congresso Latino-Americano de Software Livre e Tecnologias Abertas*. [S.I.], 2019. p. 146–149.
- [9] SEN-CHING, S. C.; KAMATH, C. Robust techniques for background subtraction in urban traffic video. In: SPIE. *Visual Communications and Image Processing 2004*. [S.I.], 2004. v. 5308, p. 881–892.