VISÃO COMPUTACIONAL EMPREGADA AO MONITORAMENTO DO FLUXO DE VEÍCULOS PARA UM CAMPUS INTELIGENTE

Palavras-Chave: MONITORAMENTO, VEÍCULOS, VISÃO COMPUTACIONAL

Autores:

Maria Isabela Monteiro Servino, FT – UNICAMP

Profa. Dra. Talía Simões dos Santos Ximenes (orientadora), FT – UNICAMP

Dr. Luis Fernando Gomez Gonzalez (coorientador), IC - UNICAMP

INTRODUÇÃO:

Com o crescente aumento no número de veículos registrados no Brasil, o país tem enfrentado desafios significativos, como congestionamentos e a escassez de vagas de estacionamento [1], [2]. O Campus I de Limeira enfrenta desafios semelhantes, com uma demanda crescente por vagas de estacionamento. Em busca de soluções para problemas urbanos, as universidades têm adotado o conceito de *Smart Campus* sendo um exemplo a Unicamp Campinas que já implementou projetos a fim de modernizar o campus e aprimorar a experiência da sua comunidade [3].

Nesse contexto, propõe-se um projeto pioneiro para o Campus I de Limeira, utilizando visão computacional [4] para monitoramento de veículos, permitindo o acompanhamento do fluxo e obtenção de dados estatísticos. Essa iniciativa multidisciplinar visa promover a conectividade, otimização do espaço e melhora da mobilidade no campus, além de abrir caminho para futuras inovações.

METODOLOGIA:

Foram testadas diversas arquiteturas para implementar a solução proposta. Inicialmente, foram utilizadas placas ESP-32 CAM como câmeras de baixo custo conectadas via Wi-Fi, enviando imagens para uma máquina virtual na Oracle Cloud [5] onde um modelo HaarCascade no OpenCV era utilizado para a detecção de veículos [6]. No entanto, essa abordagem se mostrou lenta devido ao tempo de upload das imagens, além de ter problemas com proteção de dados e privacidade de informações pessoais (LGPD). Uma proposta alternativa com a placa Raspberry Pi também não atendeu às expectativas, pois não conseguia realizar a detecção em uma taxa de quadros aceitável [7], e apresentava um custo mais elevado. Como abordagem definitiva, optou-se por utilizar uma câmera IP VIP S4020 V2 da Intelbras, apresentada na Figura 1, fornecida pela Secretaria de Administração Regional (SAR) do campus em conjunto com um computador desktop.

A câmera foi instalada em um poste com vista para a entrada principal do campus e conectada usando um cabo de rede localmente instalado para transmissão do vídeo. O processamento acontece em um computador local, tal configuração permitirá a detecção de veículos com uma taxa de quadros adequada, maior que 5 qps (quadros por segundo), permitindo a atualização em tempo real do número de veículos presentes no campus. O computador se encontra na guarita principal do campus onde atuam os funcionários responsáveis pelo monitoramento e operação das cancelas, a guarita pode ser vista na Figura 4 localizada entre a entrada e a saída de veículos.

Figura 1 - Câmera Instalada – fonte: Coletânea do autor

A comunicação entre o computador e a câmera é realizada através do protocolo RTSP (*Real Time Streaming Protocol*) [8]. Usando Python 3 e a biblioteca OpenCV, ocorre a conexão e o processamento da imagem com a detecção dos veículos. A biblioteca também realiza transformações na imagem, exibindo informações úteis sobre a detecção e métricas relevantes. Além disso, as bibliotecas Numpy e PyQt5 também são empregadas para cálculos matemáticos e criação da interface gráfica que mostra a imagem manipulada pelo OpenCV.

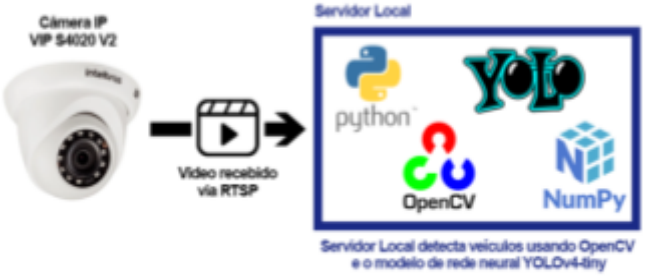

Figura 2 - Diagrama da arquitetura – fonte: coletânea do autor

Para a detecção de veículos, utiliza-se o modelo de rede neural YOLOv4-tiny, escolhido por sua velocidade e alta precisão em identificar várias classes de objetos, como carros, motos e bicicletas [9], [10]. A Figura 2 mostra o diagrama da arquitetura do processamento das imagens.

O código implementado possui um fluxo de funcionamento que começa criando uma janela com dois widgets utilizando a biblioteca PyQt5: uma imagem que exibe o último quadro processado e um timer inicia o processamento de um quadro novo conforme o *framerate* da câmera. São determinadas áreas de processamento de 416x416 pixels conforme especificado no modelo, dentro dessas áreas as regiões a serem ignoradas são demarcadas de preto para poupar processamento, em seguida tais áreas são redimensionadas para, e então e a detecção dos objetos é realizada, conforme pode ser observado na Figura 3.

Apenas as detecções de carros, motos e bicicletas com confiança acima de um limiar são consideradas, também ocorre o uso do algoritmo de Supressão Não Máxima (SNM) para remover detecções indesejadas ou em duplicidade.

Figura 3 – Detecção de veículos com SNM – fonte: Coletânea do Autor

Os centroides dos veículos são calculados e armazenados em um dicionário. A cada frame, os centroides detectados são associados ao dicionário com base na menor distância [11] em relação ao centróide do frame anterior. Se a distância ultrapassar um limite, é considerado um novo veículo e adicionado ao dicionário com uma nova chave associada.

Utilizando os últimos oito centroides de cada veículo, segmentos de reta são formados para cada veículo e são verificados se esses segmentos cruzam áreas delimitadas para adição ou remoção de veículos. Se houver interseção [12], a contagem de veículos é atualizada e o veículo é ignorado nas próximas detecções para evitar detecções em duplicidade. Para economizar memória, os veículos sem novos centroides detectados por mais de 30 quadros são eliminados.

A trajetória de cada veículo é definida pela concatenação de todos os segmentos de reta do veículo da forma $C_{Vi}C_{V(i+1)}$ onde CVi é o centroide de índice i do veículo V.

Figura 4 - Execução do código – fonte: coletânea do autor

A Figura 4 mostra a interface da aplicação com a contagem de veículos no canto superior esquerdo, sendo “C” carros, “M” motos e “B” bicicletas. Os retângulos amarelos delimitam as regiões de detecção enviadas para a rede neural, e em preto foi delimitada uma região ignorada para impedir a detecção dos veículos na rodovia próxima ao campus.

As listras verdes e vermelhas são as áreas de detecção para incrementar ou decrementar veículos, as linhas em verde-claro e branca representam as trajetórias dos carros delimitados pelos retângulos brancos. Essas linhas assumem cores aleatórias para descrever a trajetória de cada veículo, assim que cruzam as áreas verde ou vermelha elas se tornam branca evidenciando que aquele veículo foi contabilizado e está sendo ignorado nas detecções seguintes.

Além da detecção de veículos, foi desenvolvida uma aplicação Web para exibir a quantidade de veículos presentes e a disponibilidade de vagas de estacionamento no campus. A aplicação, hospedada em um servidor web externo, faz uso das linguagens PHP , HTML, CSS e Javascript e um banco de dados SQLite.

A Figura 5 mostra o diagrama da arquitetura completa da aplicação, destacando o processamento de imagem local, o envio dos dados para um servidor Web centralizado e o acesso do usuário final a esses dados.

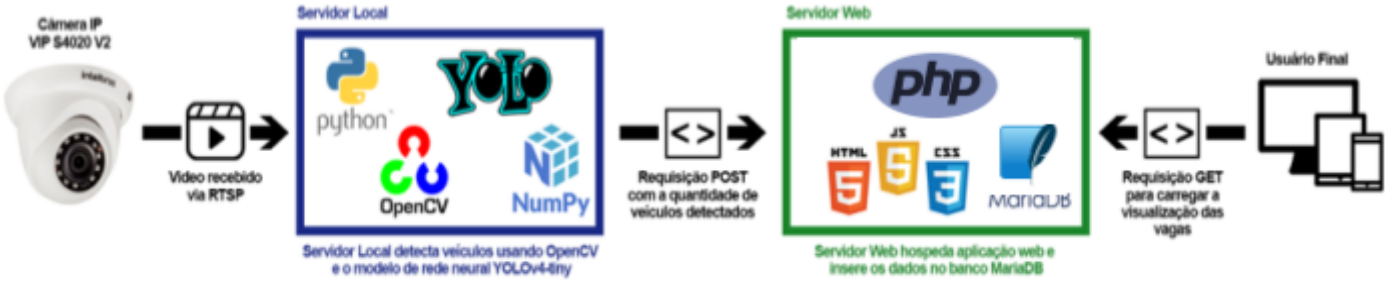

Figura 5 – Arquitetura completa da aplicação – fonte: coletânea do autor

O servidor local enviará requisições POST para o servidor Web por meio de uma API REST, que inserirá registros para cada veículo detectado, incluindo data, hora, o tipo de veículo (carro, moto ou bicicleta) e o tipo da detecção (entrada ou saída). A comunidade pode então contar com a informação da quantidade de vagas disponíveis em tempo real do campus e um histograma do fluxo de veículos daquele dia, conforme demonstrado na Figura 6.

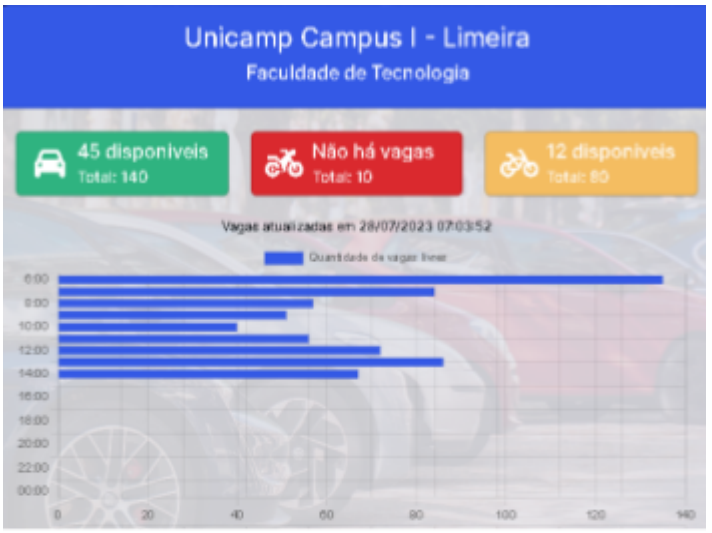

Figura 6 - Aplicação Web – fonte: coletânea do autor

RESULTADOS E DISCUSSÃO:

Durante a concepção da aplicação, desde as mudanças em sua arquitetura às otimizações do modelo, se tornou evidente a complexidade do problema, uma vez que é desafiador balancear a velocidade de processamento com o nível de precisão necessário e os recursos computacionais disponíveis. Após o desenvolvimento do modelo realizou-se a comparação da contabilização realizada entre a contagem manual de veículos e contagem realizada pelo algoritmo de detecção de veículos com um framerate de 8 qps em um notebook core i5 7200U de 7ª geração com 8GB de RAM, onde para as amostras de tempo de vídeo analisadas houve equidade entre os resultados obtidos.

Estima-se que a exatidão de detecção seja superior a 90% em um cenário de implantação definitiva, visto que as diversas variações de tempo, luminosidade e veículos compõe cenários imprevisíveis, e todas as horas de vídeo analisadas pela aplicação representam apenas algumas amostras das possíveis situações encontradas, nas quais obteve-se êxito e um desempenho efetivo.

Com os conhecimentos e autonomia adquiridos durante o desenvolvimento é perceptível o potencial para otimização deste modelo de detecção. É possível explorar o uso de outros modelos de redes neurais, ajustar o processamento da imagem para facilitar a detecção, otimizar o algoritmo de busca de centroides ou até mesmo substituir o método de detecção de interseção entre retas por uma interpolação polinomial com o auxílio de sistemas lineares para verificar as interseções.

CONCLUSÕES:

Espera-se que essa abordagem pioneira no Campus I de Limeira contribua na criação de um ambiente adaptado às necessidades dos membros da comunidade, melhorando a mobilidade de todos de forma eficiente, segura e sustentável. Além de envolver o corpo docente e discente no desenvolvimento de um espaço propício para a adesão de soluções inteligentes em prol de um *Smart Campus* que impacte positivamente a comunidade, ao mesmo tempo que demonstre nossa capacidade de desenvolvimento científico-tecnológico.

A expectativa é de que com a continuidade do trabalho e o desenvolvimento de futuras otimizações, o sistema de monitoramento e gestão de tráfego de veículos atinja níveis de velocidade e confiabilidade maiores a fim de contribuir com a autonomia da aplicação e corroborar com a sua utilização nas demais entradas do campus I Limeira, com possíveis expansões para o campus II ou até mesmo para uso em outras aplicações pela disponibilização do código no modelo de código aberto.

BIBLIOGRAFIA

[1] IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Frota de veículos. Rio de Janeiro: IBGE, 2013. Disponível em: https://cidades.ibge.gov.br/brasil/pesquisa/22/28120. Acesso em: 9 maio 2022.

[2] M.Y.I. Idris, Y.Y. Leng, E.M. Tamil, N.M. Noor and Z. Razak, 2009. Car Park System: A Review of Smart Parking System and its Technology. Information Technology Journal, 8: 101-113. Disponível em https://scialert.net/abstract/?doi=itj.2009.101.113. Acesso em 7 maio 2022.

[3] NEVES, Ana Régia de M.; SARMANHO, Kaê U.; MEIGUINS, Bianchi S. O papel da universidade na construção de cidades inteligentes e humanas. Revista Electronica de Sistemas de Informação, [s. l.], Mai-Ago 2017. Disponível em: https://www.proquest.com/openview/6dc5b34d62727ad904a4837ebadcc168/1?pq-origsite=gscholar&cbl=178195. Acesso em: 8 maio 2022.

[4] MILANO, Danilo de; HONORATO, Luciano Barrozo. Visão Computacional. Visão Computacional, [s. l.], 2010. Disponível em: https://www.academia.edu/download/35825905/2010_IA_FT_UNICAMP_visaoComputacional.pdf. Acesso em: 8 maio 2022.

[5] Escolha formas flexíveis de máquina virtual. Disponível em: https://www.oracle.com/br/cloud/compute/virtual-machines/. Acesso em: 20 de dezembro de 2022.

[6] OPENCV. OpenCV: Cascade Classifier. Disponível em: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. Acesso em: 4 de janeiro de 2023.

[7] Q-ENGINEERING. YoloV4 Raspberry Pi 4. Disponível em: https://github.com/Qengineering/YoloV4-ncnn-Raspberry-Pi-4. Acesso em: 16 de janeiro de 2023.

[8] RTSP. Disponível em: <https://pt.wikipedia.org/wiki/RTSP>. Acesso em: 16 de janeiro de 2023.

[9] BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. 22 abr. 2020.

[10] YOLOv4 Tiny Object Detection Model. Disponível em: https://roboflow.com/model/yolov4-tiny. Acesso em: 28 de janeiro de 2023.

[11] Euclidean distance. Disponível em: https://en.wikipedia.org/wiki/Euclidean_distance#Squared_Euclidean_distance. Acesso em: 12 de fevereiro de 2023.

[12] WIKIPEDIA CONTRIBUTORS. Line–line intersection. Disponível em: https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection. Acesso em: 20 de fevereiro de 2023.