

CENTRO UNIVERSITÁRIO FEI

SISTEMA ANTI-ROUBO PARA MERCADOS DE CONDOMÍNIO

Disciplina:
Visão Computacional
Isaac Jesus

Leonardo Quirino (RA 11.121.422-7)
Lucas Lagoeiro (RA 11.120.316-2)
Turma 75

São Bernardo do Campo
8º semestre / 2024

LISTA DE ILUSTRAÇÕES

| | |
|---|----|
| Figura 1 – Imagem de teste para o primeiro modelo | 7 |
| Figura 2 – Teste de reconhecimento de pessoas..... | 8 |
| Figura 3 – Imagens do treinamento | 9 |
| Figura 4 – Gráfico de comparação entre os modelos antigos com a YOLOv8 em relação a tamanho e velocidade de detecção..... | 10 |
| Figura 5 – Métricas do YOLOv8n para o dataset COCO | 10 |
| Figura 6 – Métricas do YOLOv8n para o dataset Open Images V7 | 11 |
| Figura 7 – Resultados gráficos | 12 |
| Figura 8 – Matriz de confusão..... | 13 |
| Figura 9 – Reconhecimento de pacote de molho de tomate | 14 |
| Figura 10 – Todos os pacotes do projeto..... | 15 |
| Figura 11 – Nós relacionados a câmera da entrada | 16 |
| Figura 12 – Tópicos relacionados a câmera da entrada..... | 16 |
| Figura 13 – Nós relacionados a câmera do objeto..... | 17 |
| Figura 14 – Tópicos relacionados a câmera do objeto | 17 |
| Figura 15 – Nós relacionados ao sistema central | 18 |
| Figura 16 – Tópicos relacionados ao sistema central | 18 |
| Figura 17 – Diagrama esquemático do funcionamento do sistema..... | 19 |
| Figura 18 – Pessoa entrando no minimercado..... | 22 |
| Figura 19 – Pessoa dentro no minimercado | 22 |
| Figura 20 – Pessoa para sair do no minimercado | 23 |
| Figura 21 – Pessoa saiu do no minimercado | 23 |
| Figura 22 – Após alguns segundos que a pessoa saiu do no minimercado | 24 |
| Figura 23 – Teste de produto na prateleira..... | 25 |
| Figura 24 – Teste de reconhecimento de produto não pago..... | 25 |
| Figura 25 – Teste de produto pago | 26 |
| Figura 26 – Câmera de entrada detecta a pessoa entrando e saindo..... | 27 |
| Figura 27 – Câmera do objeto detecta o produto sendo pago | 27 |
| Figura 28 – Tópico /steal após a pessoa sair | 28 |
| Figura 29 – Câmera de entrada detecta a pessoa entrando e saindo..... | 28 |
| Figura 30 – Câmera do objeto detecta o produto sendo roubado | 29 |
| Figura 31 – Tópico /steal após a pessoa sair | 29 |
| Figura 32 – Câmera de entrada detecta a pessoa entrando e saindo..... | 30 |

| | |
|---|----|
| Figura 33 – Pessoa pegando e devolvendo o produto | 30 |
| Figura 34 – Tópico /steal após a pessoa sair | 31 |
| Figura 35 – Câmera de entrada detecta a pessoa entrando e saindo..... | 31 |
| Figura 36 – Câmera do objeto detectando o produto na prateleira..... | 32 |
| Figura 37 – Tópico /steal após a pessoa sair | 32 |

SUMÁRIO

| | |
|---|-----------|
| 1. INTRODUÇÃO | 5 |
| 2. ESTADO DA ARTE..... | 5 |
| 3. TREINAMENTO | 6 |
| 3.1. TREINAMENTO E VALIDAÇÃO PARA RECONHECIMENTO DE PESSOAS..... | 6 |
| 3.2. TREINAMENTO E VALIDAÇÃO PARA RECONHECIMENTO DE PRODUTOS..... | 8 |
| 3.3. MÉTRICAS | 9 |
| 3.3.1. MODELO YOLOv8..... | 10 |
| 3.3.2. MODELO DOS PRODUTOS | 12 |
| 4. CONSTRUÇÃO DO ALGORITMO..... | 14 |
| 4.1. ESTRUTURA DO SOFTWARE | 15 |
| 4.2. ALGORITMOS EXPLICADOS DOS PRINCIPAIS NÓS | 19 |
| 5. VALIDAÇÃO E RESULTADOS..... | 21 |
| CASO 1 – HOVE PAGAMENTO DO PRODUTO..... | 27 |
| CASO 2 – HOVE ROUBO | 28 |
| CASO 3 – PESSOA PEGOU O ITEM, DEVOLVEU E SAIU | 30 |
| CASO 4 – PESSOA ENTROU E SAIU..... | 31 |
| 6. CONCLUSÃO..... | 32 |
| REFERENCIAS BIBLIOGRAFICAS | 34 |

1. INTRODUÇÃO

Mercados dentro de condomínios residenciais tem se tornando uma necessidade cada vez maior, afinal sempre que uma compra de emergência se faz necessária, eles tornam o processo muito mais prático. Apesar de serem inovadores, já que são mercados funcionais de autoatendimento, ainda há coisas que podem ser melhoradas.

O principal ponto de melhora que esses mercados podem receber é a sua segurança. Apesar de possuírem câmeras de segurança, o sistema antirroubo não é o melhor, já que não há funcionários trabalhando a todo o momento e podendo ficar de olho no que acontece lá dentro. Em alguns condomínios esse sistema depende de uma pessoa ter que olhar as câmeras a todo momento, e mesmo assim furtos podem acontecer.

Para resolver esse problema, propomos usar os conhecimentos adquiridos durante as aulas de Visão computacional e desenvolver um sistema que identifica quando furtos são cometidos nesses mercados e alerte, no mesmo momento, a empresa dona do mercado.

O primeiro passo é treinar o programa com um banco de dados de pessoas e outro de produtos comuns de mercados. Com o programa reconhecendo pessoas e produtos, o próximo passo é treiná-lo para identificar quando alguém entra e sai do mercado, e isso será feito adicionando uma linha onde a porta está localizada (passar uma vez pela linha retorna “*in*” e passar pela segunda vez retorna “*out*”).

O passo final é treinar o programa para reconhecer as ações dos clientes para distinguir o que é roubo e o que não é. Isso será feito ao preparar o programa para identificar se a pessoa pegou ou não o produto e se ela saiu e parou na máquina de pagar. Caso o cliente pare na máquina o programa será treinado para reconhecer a tela da máquina, já que muitas exibem uma tela verde de confirmação de pagamento.

2. ESTADO DA ARTE

Antes de começar a desenvolver o projeto, foram realizadas diversas pesquisas para um entendimento melhor do problema e de como ele poderia ser resolvido. Entre essas pesquisas foram encontrados três artigos que abordam problemáticas semelhantes à escolhida, desde furtos de objetos a reconhecimento de produtos.

O primeiro artigo analisado foi Detecção de Roubo de Computadores em Laboratório Usando Visão Computacional [1], feito por um grupo de sete alunos da Universidade Positivo. No artigo é apresentado um método capaz de detectar a remoção de computadores em laboratórios acadêmicos por meio de análise de vídeo em tempo real. O sistema desenvolvido

pelos alunos utiliza técnicas de visão computacional, como subtração de fundo e segmentação, para identificar computadores sendo levados por pessoas saindo do laboratório. O software alerta atividades suspeitas, como o transporte de gabinetes de computador na saída, contribuindo para prevenir furtos e minimizar danos acadêmicos.

O artigo *Computer Vision and Deep Learning for Retail Store Management* [2], escrito por Alessio Tonioni, aborda o uso de visão computacional para automatização de tarefas gerenciais em supermercados, como inventário e suporte ao cliente. Tonioni foca em reconhecimento automatizado de produtos em prateleiras, para melhorar a gestão e a experiência do cliente, e reconstrução 3D de ambientes, permitindo a navegação segura de agentes autônomos.

O último artigo estudado foi escrito por Ceren Gülra Melek, Elena Battini Sonmez e Songül Varlı. Intitulado de *Datasets and methods of product recognition on grocery shelf images using computer vision and machine learning approaches: An exhaustive literature review* [3], o artigo revisa a literatura sobre sistemas de reconhecimento de produtos em prateleiras de supermercados, abordando desafios como a obtenção de dados e a diversidade de produtos. Ele destaca benefícios, como monitoramento de layouts e estoque, além de melhorias na experiência de compra, especialmente para pessoas com deficiência visual. Métodos de visão computacional e aprendizado de máquina são analisados, comparando conjuntos de dados e abordagens disponíveis na literatura. O estudo fornece diretrizes e perspectivas para futuras pesquisas na área.

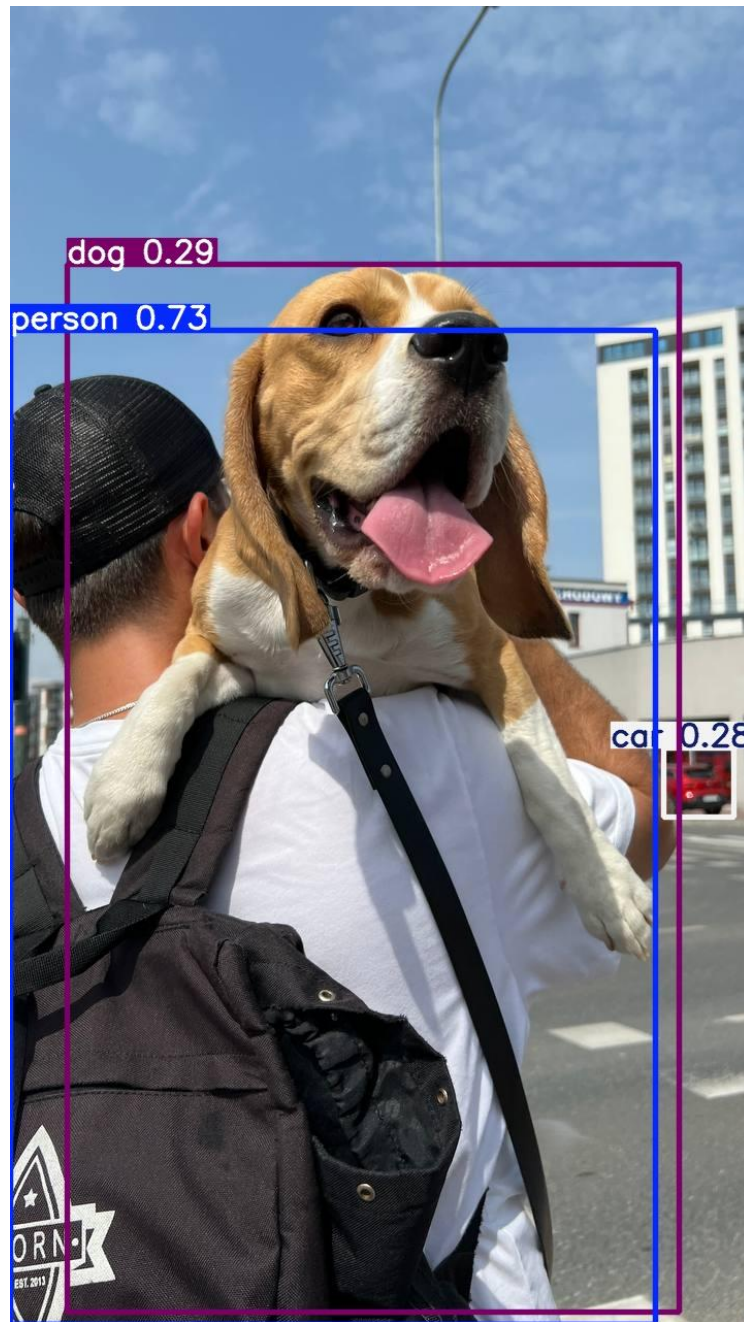
3. TREINAMENTO

Realizado o estudo, foi iniciado o desenvolvimento do projeto. A construção foi dividida em duas partes, a primeira parte é o preparo dos *datasets* e treinamento dos modelos para o reconhecimento de pessoas e de produtos, e a segunda parte é a construção dos algoritmos que realizarão a detecção de roubos no mercado.

3.1. TREINAMENTO E VALIDAÇÃO PARA RECONHECIMENTO DE PESSOAS

Para que o algoritmo fosse capaz de identificar um roubo, primeiro, ele teria que ser capaz de reconhecer uma pessoa. A biblioteca YOLO foi essencial para o desenvolvimento do projeto, já que ele disponibiliza um modelo pré-treinado capaz de reconhecer muitos objetos, inclusive pessoas. O modelo utilizado foi o YOLOv8n. A próxima imagem contém o resultado do primeiro teste feito para o modelo.

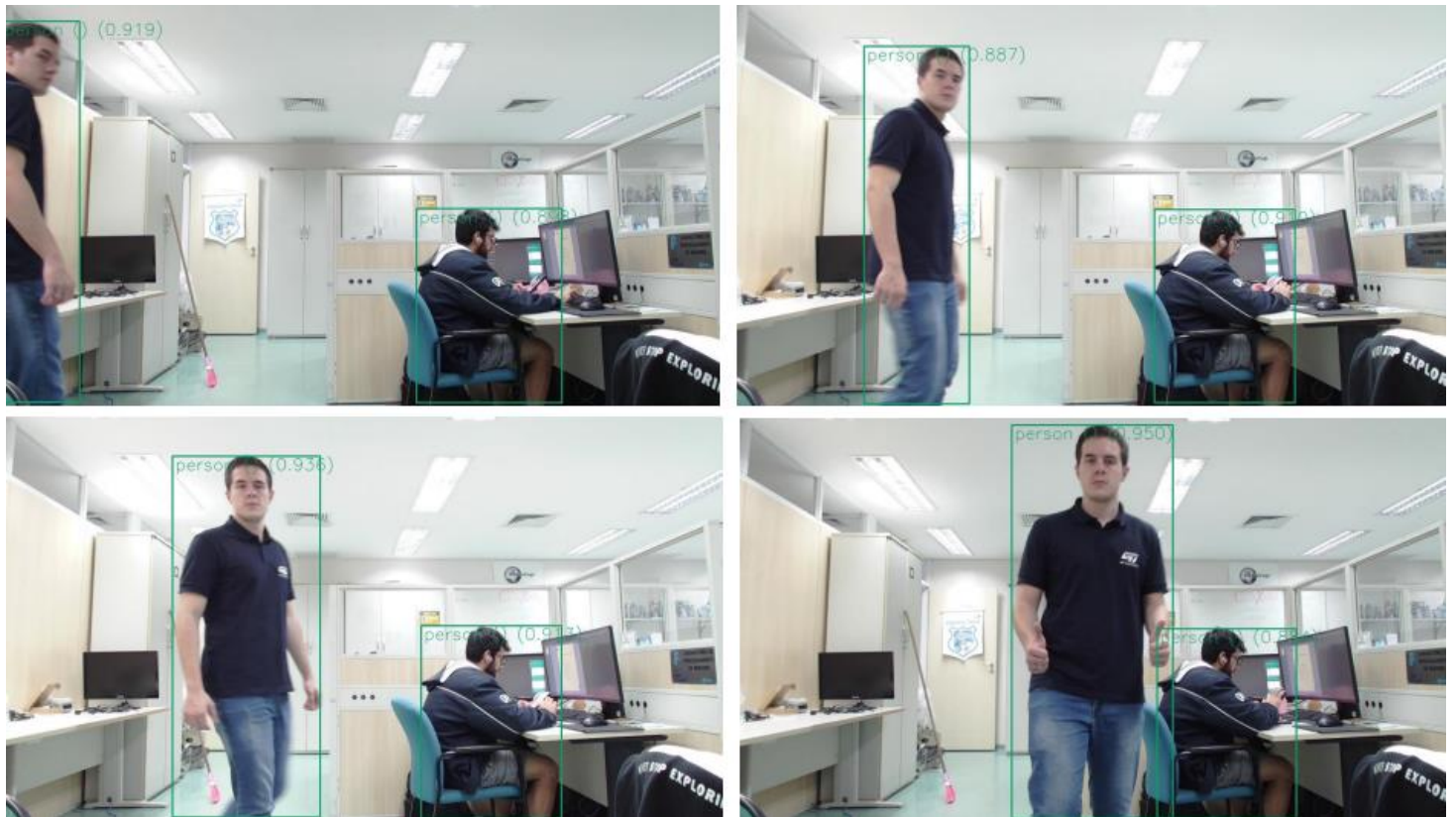
Figura 1 – Imagem de teste para o primeiro modelo



Fonte: Autor

Apesar de extremamente preciso no reconhecimento de pessoas, o modelo da YOLO também reconhece diversos elementos (desde objetos, até animais), o que normalmente não seria um problema, mas para esse projeto se torna. Além de exigir mais do algoritmo, a imagem de vídeo se torna poluída devido às diversas *bounding boxes* que o modelo usa para identificar os elementos. Para contornar esse problema, foi feita uma alteração no código para que apenas exibisse e identificasse a classe de pessoas.

Figura 2 – Teste de reconhecimento de pessoas



Fonte: Autor

3.2. TREINAMENTO E VALIDAÇÃO PARA RECONHECIMENTO DE PRODUTOS

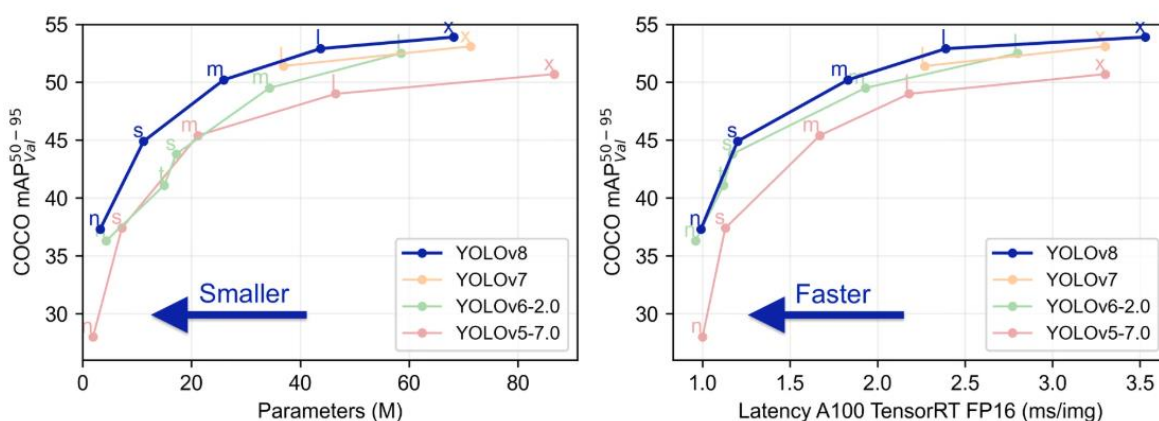
O próximo passo foi realizar o treinamento do modelo que iria identificar os produtos de mercado. O primeiro *dataset* encontrado [5] possuía um total de 3114 imagens e cinco classes, entre elas havia uma classe para Doritos. Para o treino, 2724 imagens foram separadas para treino, 261 para validação e 129 para teste, e foram configuradas 5000 épocas. Por mais que o número de épocas pareça elevado, não seriam usadas todas, já que o treino se encerra no momento em que o modelo demonstra que já ter aprendido a reconhecer os elementos. Durante a validação do modelo, notou-se que o modelo reconhecia um pacote de Doritos, mas apenas se o pacote estivesse em frente a um fundo branco. O reconhecimento ser apenas no fundo branco ocorre por conta das imagens dos pacotes de Doritos usados no *dataset* serem com fundo branco. Então foi descartada a possibilidade de usar esse *dataset*.

Após uma segunda busca por um *dataset* que possuísse imagens de itens de mercado, foi decidido usar o *dataset* desenvolvido pelos alunos do projeto Robô FEI @Home [6], já que contém diversas imagens de diferentes produtos (pó de café, água, condimentos etc.). O novo

3.3.1. MODELO YOLOv8

O modelo YOLOv8n.pt, apresenta uma grande vantagem de ser uma rede pré-treinada. Por esse motivo, foi retirado da documentação oficial o gráfico de comparação entre os modelos da YOLO, bem como as métricas de desempenho, conforme representado abaixo. [4]

Figura 4 – Gráfico de comparação entre os modelos antigos com a YOLOv8 em relação a tamanho e velocidade de detecção



Fonte: Ultralytics

Figura 5 – Métricas do YOLOv8n para o *dataset* COCO

| Model | size (pixels) | mAP ^{val} ₅₀₋₉₅ | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---------|---------------|-------------------------------------|---------------------|--------------------------|------------|-----------|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

Fonte: Ultralytics

Figura 6 – Métricas do YOLOv8n para o *dataset Open Images V7*

| Modelo | tamanho (pixéis) | mAPval 50-95 | Velocidade CPU ONNX (ms) | Velocidade A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---------|---------------------|-----------------|--------------------------------|-------------------------------------|---------------|--------------|
| YOLOv8n | 640 | 18.4 | 142.4 | 1.21 | 3.5 | 10.5 |
| YOLOv8s | 640 | 27.7 | 183.1 | 1.40 | 11.4 | 29.7 |
| YOLOv8m | 640 | 33.6 | 408.5 | 2.26 | 26.2 | 80.6 |
| YOLOv8l | 640 | 34.9 | 596.9 | 2.43 | 44.1 | 167.4 |
| YOLOv8x | 640 | 36.3 | 860.6 | 3.56 | 68.7 | 260.6 |

Fonte: Ultralytics

Ao analisar o gráfico e as tabelas fornecidas, é possível observar que o YOLOv8n se destaca pela sua eficiência em termos de recursos computacionais, apresentando um bom equilíbrio entre velocidade e desempenho no *dataset COCO*. O mAP50-95 (*mean Average Precision*, considerando IoU entre 50% e 95%) é uma métrica que avalia a precisão do modelo ao detectar objetos, atingindo 37,3% no COCO. Esse desempenho é alcançado com apenas 3,2M de parâmetros, indicando que o modelo utiliza pouco espaço de memória, enquanto o custo computacional, representado pelos FLOPs (Operações de Ponto Flutuante por segundo), é de 8,7B. Essas características resultam em velocidades de inferência extremamente altas, com latências de 80,4 ms em CPU (ONNX) e 0,99 ms na GPU (A100 TensorRT), tornando-o ideal para dispositivos com recursos limitados.

No *dataset Open Images v7*, observa-se uma redução significativa no desempenho do YOLOv8n, com o mAP50-95 caindo para 18,4%, o que indica maior dificuldade em generalizar para um *dataset* mais complexo e diverso. Nesse caso, o número de parâmetros aumenta ligeiramente para 3,5M, enquanto os FLOPs sobem para 10,5B, refletindo um pequeno aumento no custo computacional em relação ao COCO. Apesar dessa queda na precisão, o modelo mantém latências competitivas, registrando 142,4 ms em CPU e 1,21 ms na GPU. Esses valores destacam que, mesmo com a maior carga de processamento, o YOLOv8n ainda é eficiente em termos de velocidade e consumo de recursos.

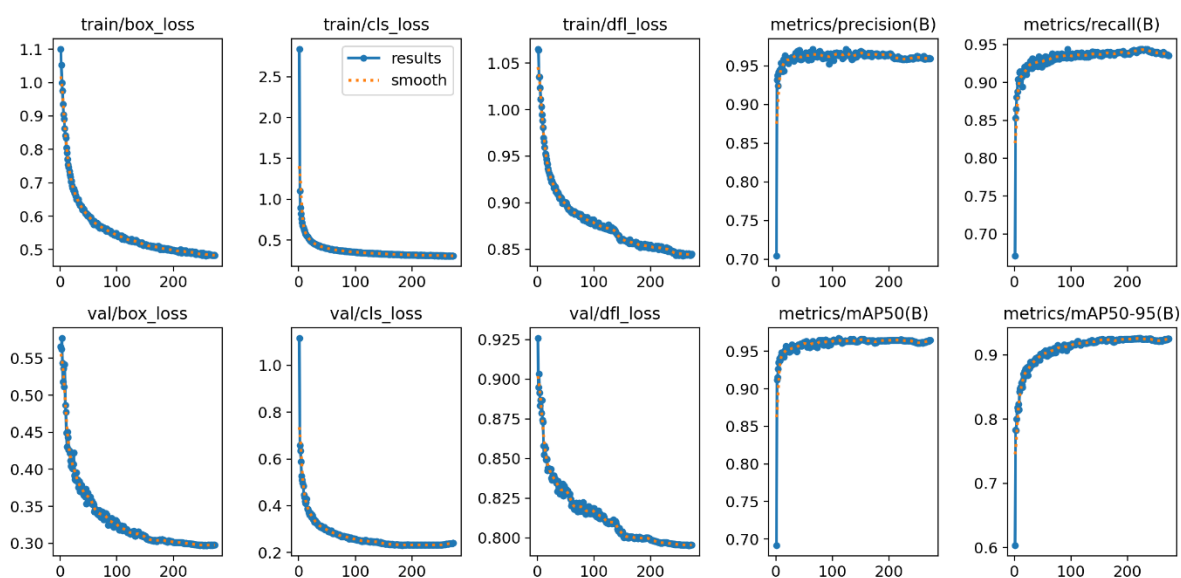
Como o sistema possivelmente será implementado em um ambiente embarcado com limitação de *hardware*, o baixo número de parâmetros, o custo computacional reduzido e a alta velocidade de inferência fazem dele uma escolha ideal. Além disso, o desempenho no *dataset*

COCO, mesmo sem atingir níveis de precisão extremamente altos, se mostrou adequado para a detecção básica de pessoas, garantindo a viabilidade do projeto em cenários reais.

3.3.2. MODELO DOS PRODUTOS

O treinamento do modelo de identificação de produtos retornou um conjunto de gráficos que estão relacionados as métricas de desempenho do modelo, esses gráficos foram usados para validar o desempenho do modelo após o treino.

Figura 7 – Resultados gráficos



Fonte: Autor

Ao analisar os gráficos, é notável que aqueles relacionados à fase de treinamento mostram que o modelo teve uma evolução estável e consistente em relação às perdas. As curvas de *train/box_loss*, *train/cls_loss* e *train/dfl_loss* apresentam quedas significativas ao longo das épocas, indicando que o modelo está aprendendo a ajustar as caixas delimitadoras, a classificar os objetos de maneira correta e a refinar as bordas das previsões. A redução de *train/cls_loss* de 2,5 para 0,5 é particularmente notável, sugerindo um aprendizado eficaz da tarefa de classificação.

Nos dados de validação, as perdas de *val/box_loss*, *val/cls_loss* e *val/dfl_loss* seguem padrões semelhantes aos do treinamento, mas começam com valores ligeiramente menores. Isso pode indicar que os dados de validação possuem características mais simples ou estão bem representados nos dados de treino. As curvas mostram boa estabilidade, com valores finais

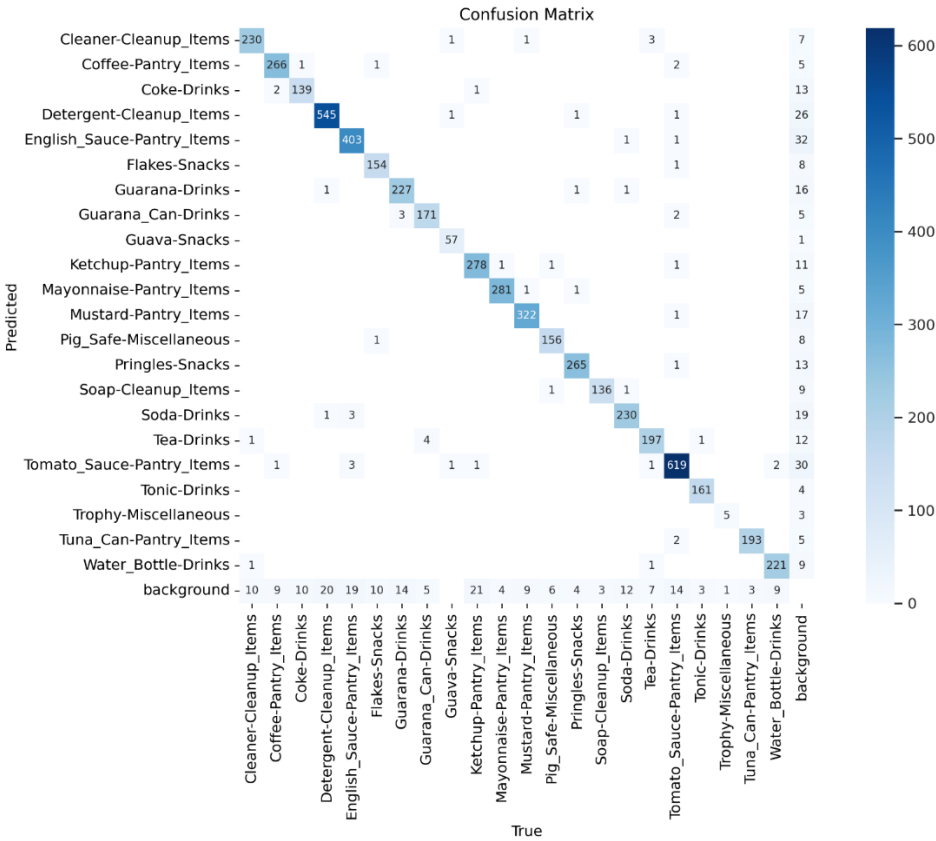
converging para níveis baixos, como $\sim 0,30$ para *val/box_loss* e $\sim 0,20$ para *val/cls_loss*. A ausência de aumento nas perdas de validação ao longo das épocas reforça que o modelo não está sofrendo *overfitting*, o que demonstra uma boa capacidade de generalização.

As métricas de desempenho, como precisão, *recall*, mAP50 e mAP50-95, apresentam resultados excelentes, atingindo valores finais acima de 90%. A precisão alta indica que a maioria das detecções do modelo são verdadeiras, enquanto o *recall* elevado sugere que poucos objetos relevantes estão sendo perdidos. A mAP50 indica o quanto o modelo é eficaz em prever caixas delimitadoras com um limiar de ao menos 50% de IoU (*Intersection over Union*), enquanto a mAP50-95 evidencia a capacidade do modelo de manter um bom desempenho para diferentes limiares de IoU. A rápida convergência dessas métricas nas primeiras épocas e a estabilização em níveis altos destacam o equilíbrio entre as detecções corretas e a consistência.

Em resumo, o treinamento foi bem-sucedido, com o modelo demonstrando estabilidade e generalização tanto nos dados de treino quanto nos de validação. As métricas de desempenho indicam que o modelo possui alta precisão e é robusto o suficiente para aplicações práticas.

Para ampliar a análise, foi gerada uma matriz de confusão de predição por valores corretos para cada classe de produto.

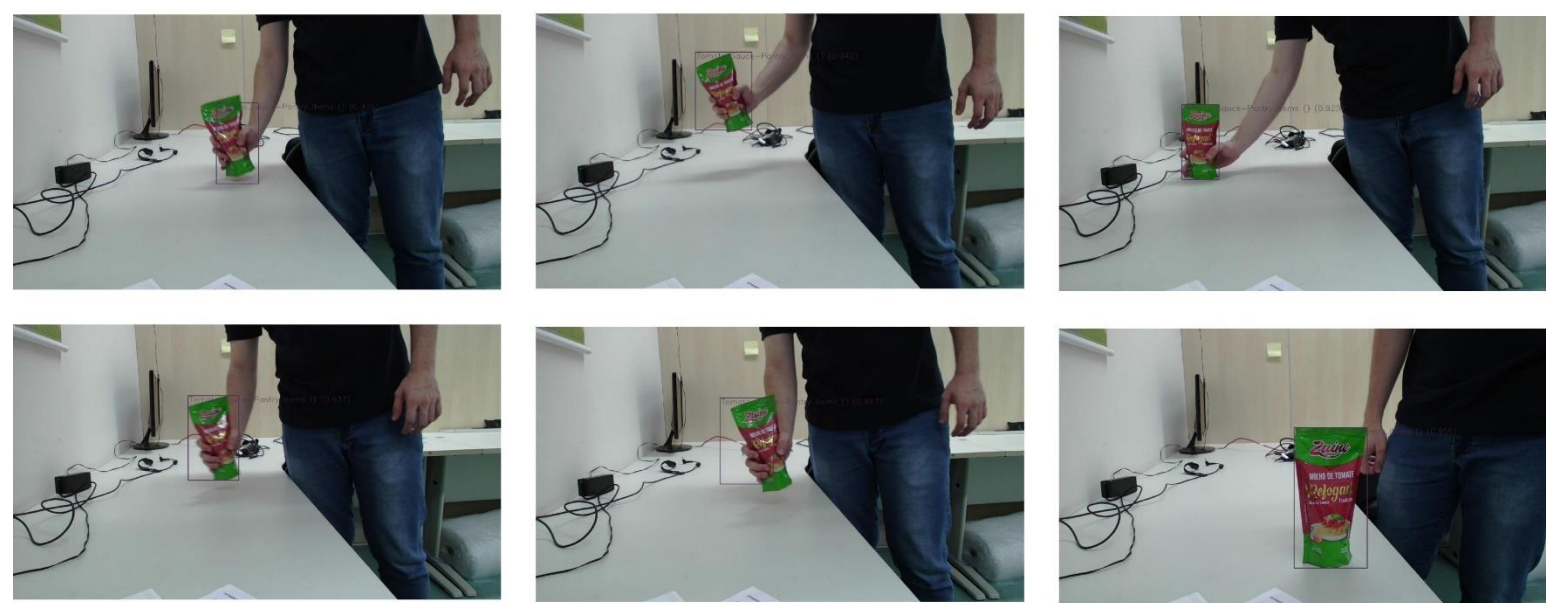
Figura 8 – Matriz de confusão



Fonte: Autor

Assim como o modelo para identificação de pessoas, o modelo de identificação de produtos foi configurado para identificar e destacar somente uma classe de produto. A classe escolhida foi a de molho de tomate, pois apresentou o maior valor de predição – como pode ser visto na matriz de confusão, e o modelo também foi ajustado para destacar se a identificação retornasse mais de 90% de acurácia. A seguir será demonstrado como foi feito o teste do modelo no reconhecimento do molho de tomate.

Figura 9 – Reconhecimento de pacote de molho de tomate



Fonte: Autor

4. CONSTRUÇÃO DO ALGORITMO

Com os modelos treinados, se deu início a próxima etapa do projeto. A solução pensada para solucionar o problema de furtos em mercados de condomínio consiste em utilizar duas câmeras para identificar quando uma pessoa entra e sai do mercado e se o pagamento do produto foi realizado. Para que isso fosse possível, foram desenvolvidos dois algoritmos, um para cada câmera.

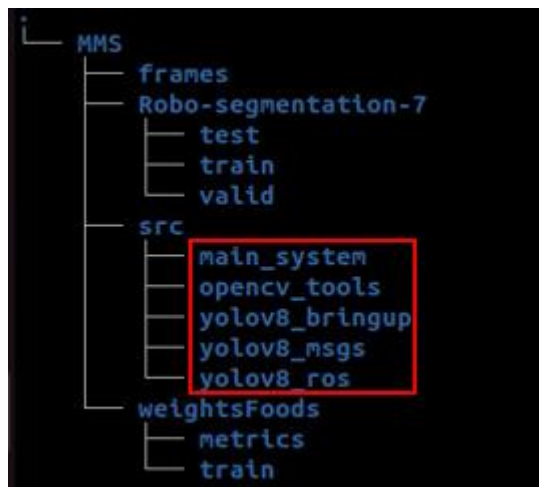
Foi utilizado o *middleware* ROS2 Humble, aproveitando as ferramentas de comunicação via tópicos e o ROS_DOMAIN_ID para comunicação entre dois dispositivos diferentes, mas conectados na mesma rede, de modo que as duas câmeras pudessem se comunicar e verificar em tempo real se a pessoa roubou ou comprou o produto em questão.

4.1. ESTRUTURA DO SOFTWARE

Para uma melhor compreensão da maneira que o sistema projetado funciona, é de suma importância entender toda a estrutura do *software*, como os pacotes, nós e tópicos utilizados no projeto.

No total, o projeto conta com 5 pacotes. O `yolov8_ros` é um pacote pronto adaptado para o projeto que conta com toda a lógica e capacidade da YOLOv8 para identificação e demarcação de *bounding boxes* de objetos e pessoas. Em seguida, há o pacote `yolov8_bringup`, que contém apenas os *launchers* dos nós do `yolov8_ros`. Há também outro pacote chamado `yolov8_msgs`, que reúne todas as *custom messages* utilizadas ao longo do projeto. O pacote `opencv_tools` foi criado especificamente para este projeto, nele há nós capazes de publicar a imagem real em tópicos e receber as imagens pós-processadas da YOLOv8, para que a lógica de detecção de pessoas possa ser implementada. Por fim, há o pacote `main_system`, cujo nó é responsável por atuar como o sistema central do projeto, pois é ele quem recebe as informações do estado e posição do produto e das pessoas que entram e saem do mercado, também realiza a validação das informações para identificar se foi ou não cometido roubo de produtos.

Figura 10 – Todos os pacotes do projeto

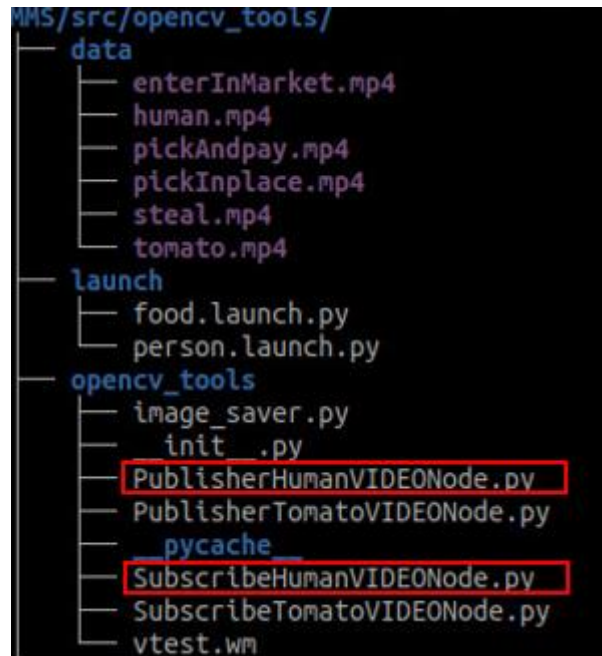


Fonte: Autor

Sobre os nós e tópicos, o pacote `opencv_tools` possui quatro nós importantes para o funcionamento do sistema, sendo dois deles para a câmera da entrada do minimercado. O nó `PublisherHumanVIDEONode.py` envia a imagem captada pela câmera através do tópico `/human_image_raw` e o `SubscribeHumanVIDEONode.py` a recebe processada pelo pacote da YOLOv8 através do tópico `/yolo/dbg_image_human` para realizar a lógica em cima da detecção

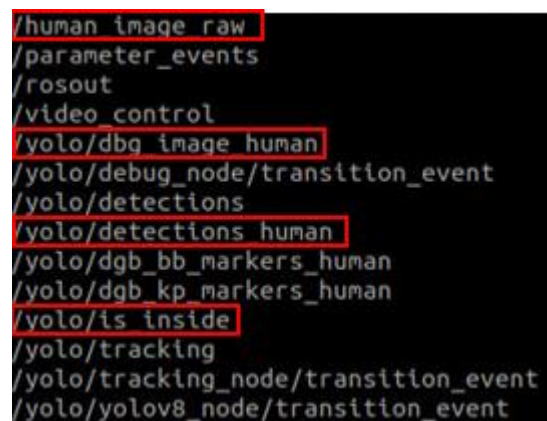
das pessoas e publicar para o tópico `/yolo/is_inside` com a informação de quantas pessoas entraram.

Figura 11 – Nós relacionados a câmera da entrada



Fonte: Autor

Figura 12 – Tópicos relacionados a câmera da entrada

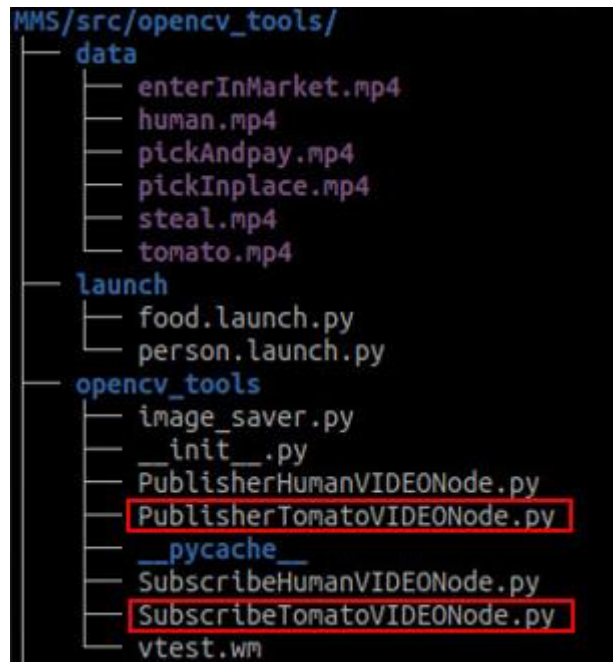


Fonte: Autor

Os dois nós restantes são para a câmera localizada na região onde ficam os produtos no minimercado. O nó `PublisherTomatoVIDEONode.py` é responsável por publicar a imagem real da câmera através do tópico `/tomato_image_raw`, e o `SubscribeTomatoVIDEONode.py` recebe a imagem processada pelo pacote YOLOv8 através do tópico `/yolo/dbg_image_food`, para que

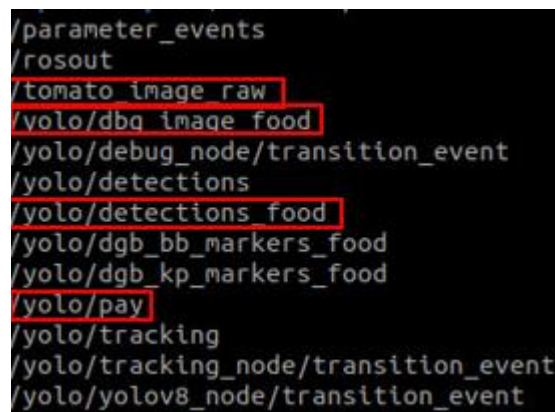
possa realizar a lógica em cima do objeto detectado e publicar no tópico `/yolo/pay` a informação sobre o estado do produto (se encontra na prateleira e se foi pago).

Figura 13 – Nós relacionados a câmera do objeto



Fonte: Autor

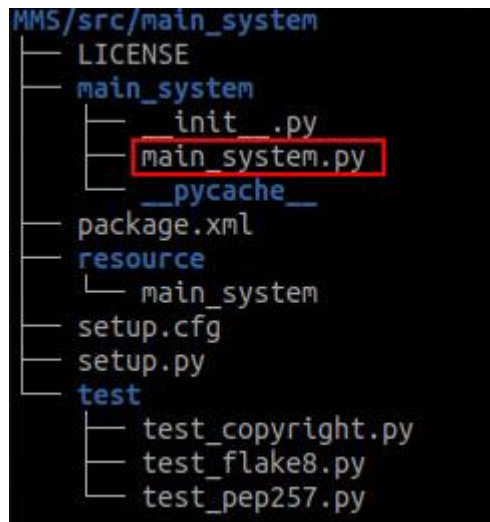
Figura 14 – Tópicos relacionados a câmera do objeto



Fonte: Autor

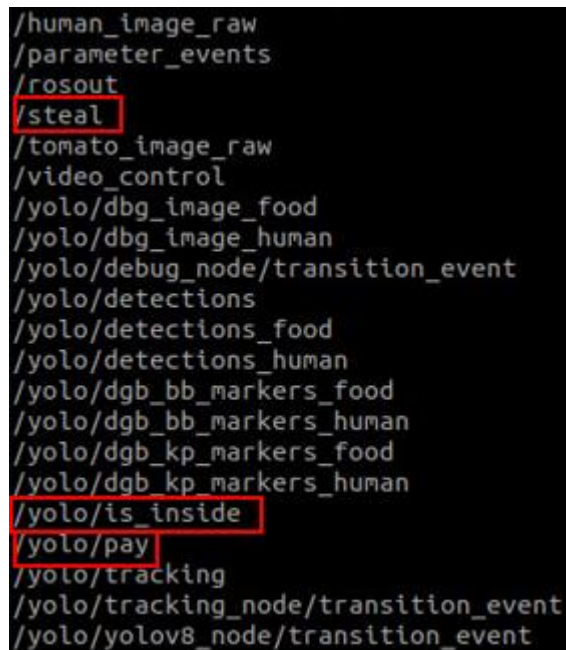
O nó `main_system.py`, do pacote `main_system`, é o responsável por verificar se a pessoa roubou ou não o produto, e faz isso por meio de dois *subscribers*, um no tópico `/yolo/is_inside` e outro no tópico `/yolo/pay` e publica no tópico `/steal` para finalmente identificar o roubo.

Figura 15 – Nós relacionados ao sistema central



Fonte: Autor

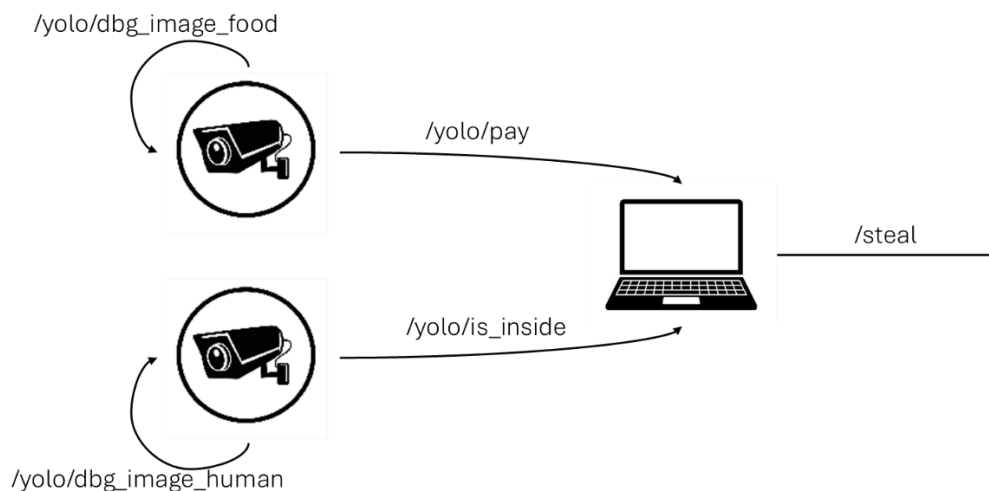
Figura 16 – Tópicos relacionados ao sistema central



Fonte: Autor

Com a estrutura do software explicada segue abaixo o diagrama esquemático simplificado do funcionamento do sistema.

Figura 17 – Diagrama esquemático do funcionamento do sistema



Fonte: Autor

4.2. ALGORITMOS EXPLICADOS DOS PRINCIPAIS NÓS

O primeiro algoritmo, relacionado ao nó *SubscribeHumanVIDEONode.py*, insere uma linha na imagem da câmera para separar a área de entrada e saída do mercado. Assim que uma pessoa entra no alcance da câmera, o algoritmo analisa se a pessoa está dentro ou fora do mercado ao seu centro cruzar a linha. Caso a pessoa tenha entrado, ele verifica se ela já não estava dentro para poder diferenciar se a pessoa está saindo ou entrando.

Caso ela esteja entrando, é incrementado o número total de pessoas dentro do minimercado e é declarada uma variável booleana igual a *True* para aquela pessoa específica. Agora, caso seja detectado que a pessoa esteja saindo, é decrementado o número de pessoas dentro do minimercado e a booleana designada a essa pessoa é declarada como *False*. Em seguida, é iniciado um *timer*; se esse tempo for maior que 3 segundos, a booleana declarada para essa pessoa é removida para que as listas desse algoritmo não ocupem tanta memória.

Essas informações sobre a presença de pessoas dentro do mercado e as booleanas para cada pessoa são publicadas no tópico */yolo/is_inside* para o compartilhamento dessa informação com o resto do sistema.

O segundo algoritmo, relacionado ao nó *SubscribeTomatoVIDEONode.py*, foi desenvolvido para operar na segunda câmera no reconhecimento do produto pago ou não. A ideia para o segundo algoritmo é ser capaz de identificar se o produto foi retirado da prateleira e em seguida foi checado no caixa. Ele ser capaz de identificar se o produto está na prateleira ou não é de eximia importância para o melhor funcionamento, já que uma pessoa pode entrar, olhar e sair.

Antes de começar o desenvolvimento da lógica de produto pago ou não, foram inseridos dois retângulos. O retângulo roxo tem o objetivo de delimitar a área onde o produto ficará quando na prateleira, e o retângulo quadrado marca a área de pagamento do produto. Com as áreas delimitadas e o modelo de reconhecimento de produto, foi feita a lógica de reconhecimento de roubo.

Foram configuradas duas variáveis booleanas, uma que atualiza seu estado de acordo com o produto na prateleira e uma que atualiza se o produto foi pago ou não. Para identificar se o produto está na prateleira ou não, o algoritmo rastreia a posição do centro da *bounding box*, enquanto as coordenadas do centro forem menores que as coordenadas das bordas do retângulo, a variável da prateleira permanece em *True*, indicando que o produto não foi pego. Caso as coordenadas do centro sejam maiores que as bordas do retângulo, o estado da variável muda para *False*, indicando que o produto foi pego.

O reconhecimento do produto pago ou não funciona de uma forma parecida. A partir do momento que a variável referente ao estado do produto na prateleira se torna *False*, a variável referente ao pagamento do produto entra no estado *False* também e, enquanto estiver neste estado, uma mensagem de aviso dizendo que o produto ainda não foi pago é exibida em repetição. Para que a câmera identifique o pagamento do produto, ela rastreia o centro do produto e no instante em que as coordenadas do centro ficam menores que as bordas do retângulo do pagamento, a variável referente ao pagamento se torna *True* e a mensagem de aviso informa que o produto está pago.

Com os dois algoritmos desenvolvidos, foi iniciado o terceiro algoritmo, relacionado ao nó *main_system.py*, onde é processo de identificação de roubo ou não. Os algoritmos são capazes de identificar quando uma pessoa entra e sai, se ela tirou o produto da prateleira ou não e se o pagamento realizado, e a junção dessas funções leva a lógica desse terceiro algoritmo. Quando uma pessoa entra em um mercado ela pode pegar um produto e pagar, pegar um produto e roubar, pegar um produto somente para olhar ou só entrar e sair. Esses foram os quatro casos pensados para o desenvolvimento da lógica.

Para o primeiro caso, o primeiro algoritmo lê o vídeo retornado pela câmera para identificar se a pessoa entrou no mercado, e, então, o segundo algoritmo entra em ação. Assim que a pessoa tira o produto da área da prateleira a mensagem de aviso informando que o produto não foi pago começa a ser exibida, então a pessoa realiza o pagamento do produto fazendo com que a mensagem de aviso mude e o segundo algoritmo alerte o sistema central sobre o pagamento. O primeiro algoritmo reconhece a saída da pessoa e envia também para o sistema central, onde essas informações são computadas e validadas que não houve roubo.

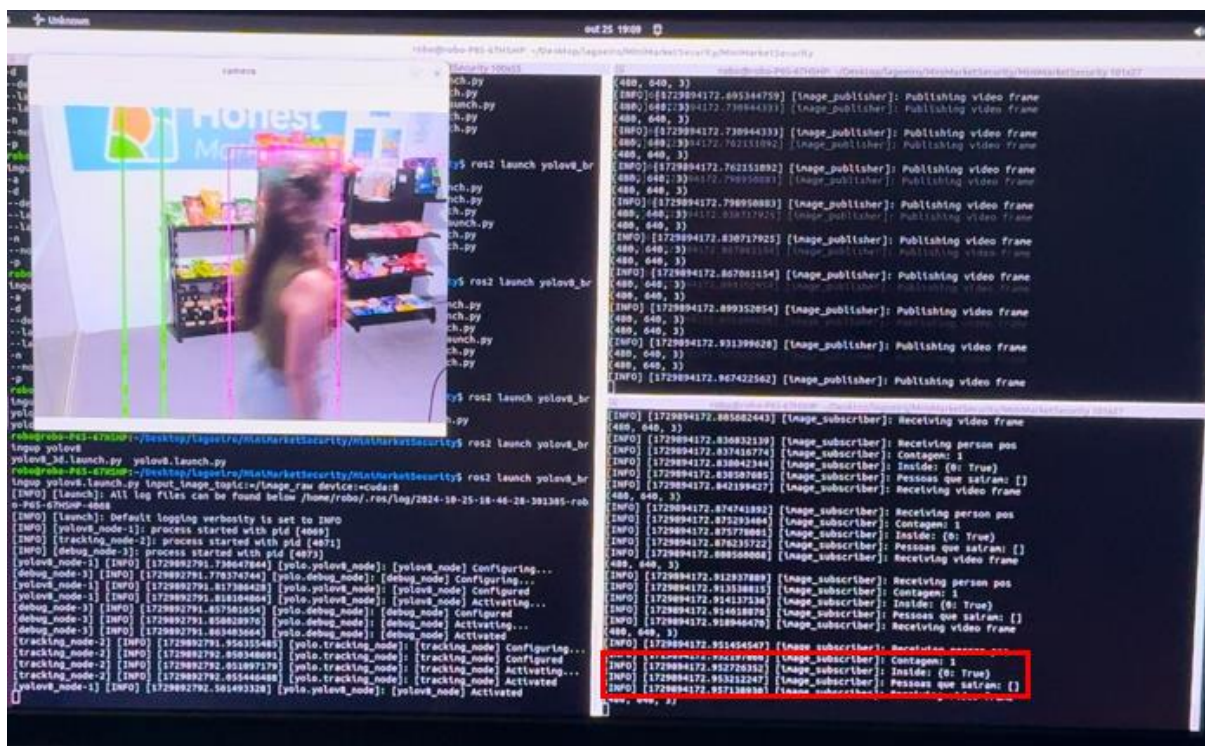
A lógica para o segundo caso funciona de uma maneira parecida. No momento em que a pessoa entra no mercado o algoritmo identifica e guarda essa informação, então ela remove o produto da prateleira e o segundo algoritmo identifica que o produto ainda não foi pago e envia essa informação para o terceiro algoritmo. Caso o terceiro algoritmo ainda não tenha recebido a informação de pagamento, mas identificou que a pessoa saiu, ele irá identificar como roubo.

Nos casos em que a pessoa entra e apenas olha, a variável do estado do produto na prateleira se faz essencial. Se o primeiro algoritmo identificar que uma pessoa entrou e o segundo estiver identificando que o produto ainda está na prateleira, a variável de pagamento do produto não é ativada e quando a pessoa sair não será identificado roubo. E caso a pessoa pegue o produto e depois o devolva à prateleira, a variável de pagamento é ativada dizendo que não foi pago, mas em seguida é desativada e a pessoa pode sair sem que seja alertado um roubo.

5. VALIDAÇÃO E RESULTADOS

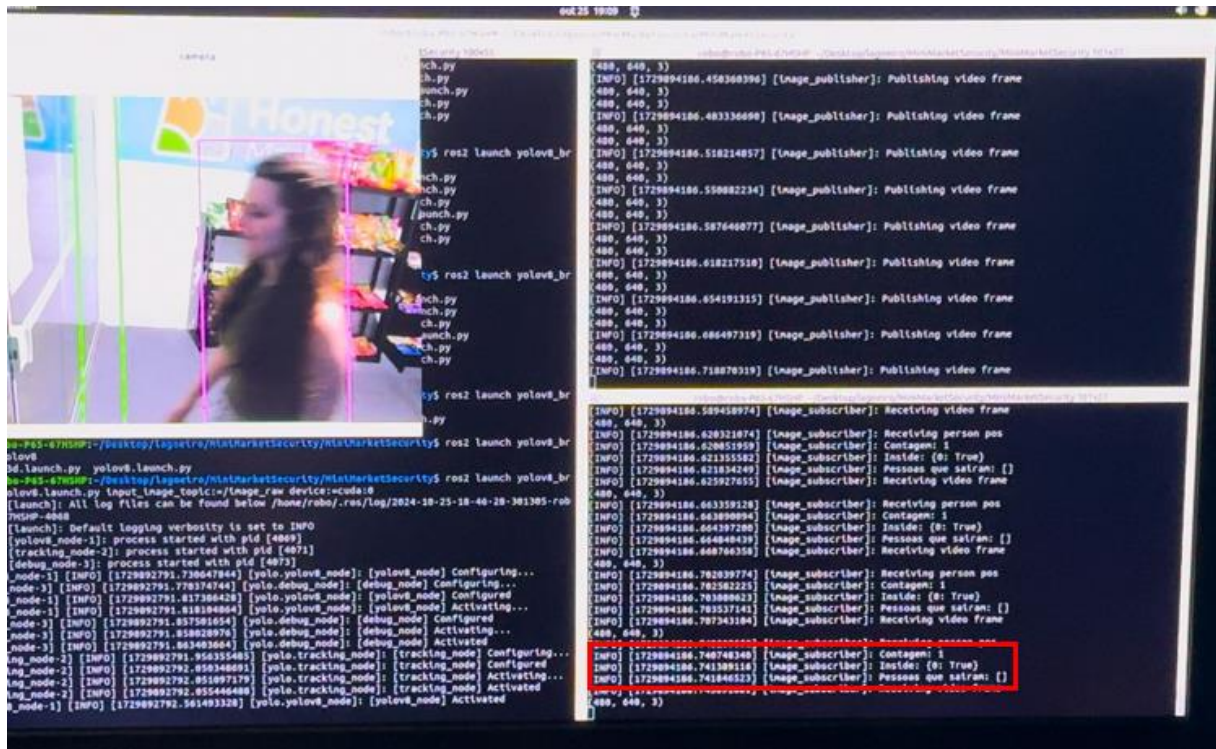
Antes da validação final do projeto, cada algoritmo foi testado individualmente. Para o teste do algoritmo que reconhece se a pessoa entrou ou não, foi utilizado o espaço do minimercado do condomínio Chácara das Fontes, que é o espaço planejado para realizar o teste final dos algoritmos. Então com uma câmera e um computador Avell para executar o código, os membros do grupo realizaram diversos testes para garantir que o algoritmo funcionasse da maneira desejada.

Figura 19 – Pessoa dentro no minimercado



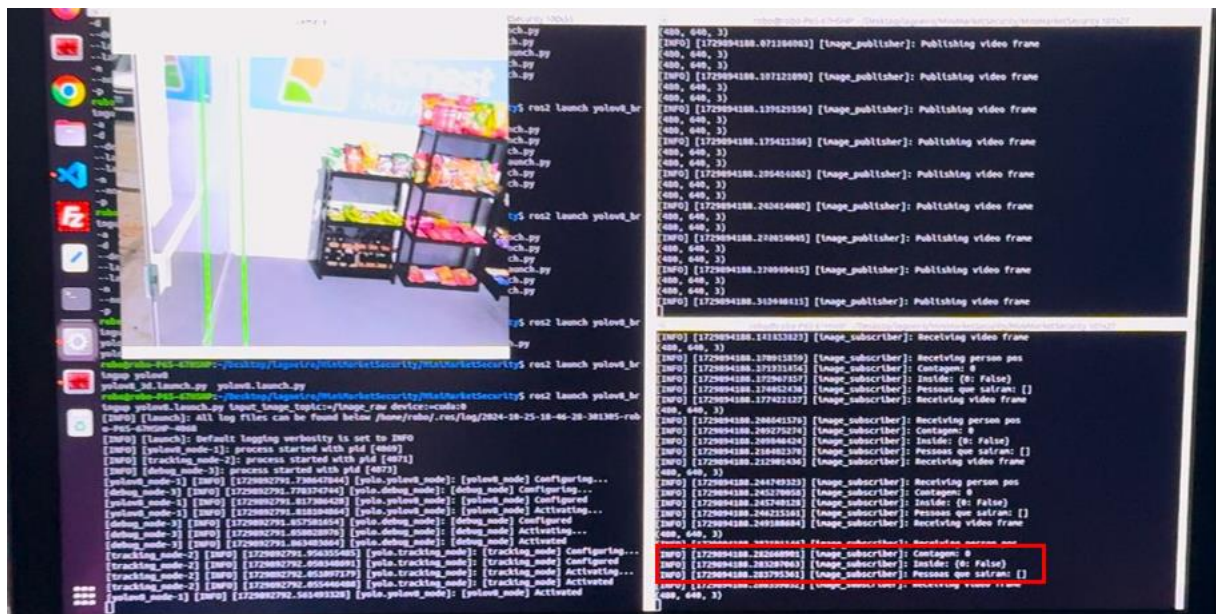
Fonte: Autor

Figura 20 – Pessoa para sair do no minimercado



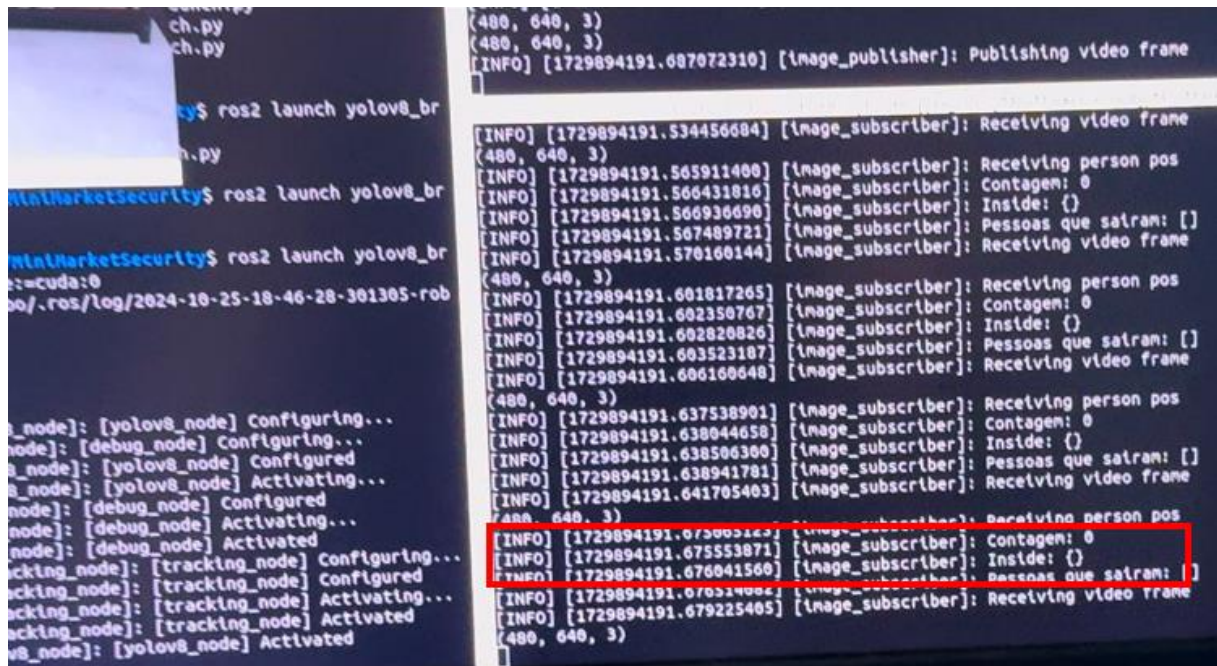
Fonte: Autor

Figura 21 – Pessoa saiu do no minimercado



Fonte: Autor

Figura 22 – Após alguns segundos que a pessoa saiu do no minimercado



```
ch.py
ch.py

$ ros2 launch yolov8_br
n.py

MiniMarketSecurity$ ros2 launch yolov8_br
MiniMarketSecurity$ ros2 launch yolov8_br
z=cuda:0
bo/.ros/log/2024-10-25-18-46-28-301305-rob

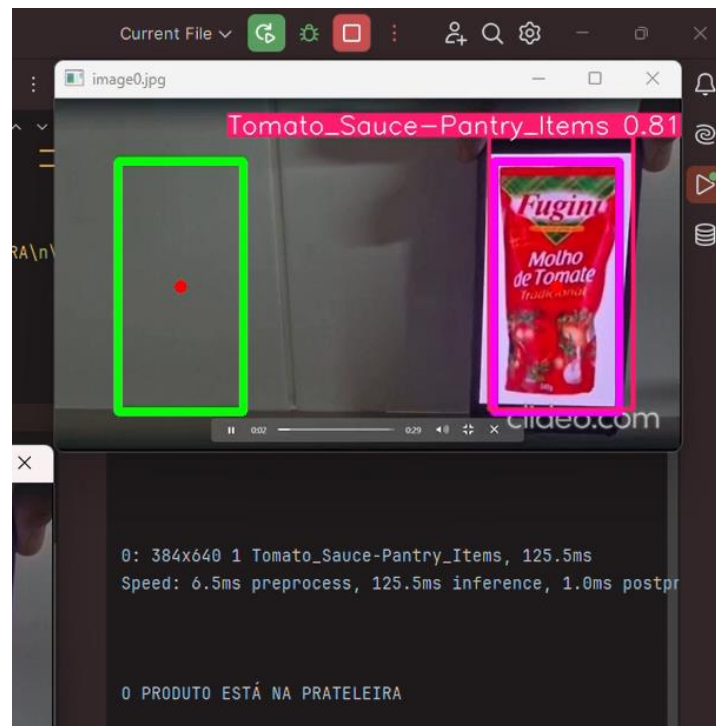
[INFO] [1729894191.534456684] [image_publisher]: Publishing video frame
(480, 640, 3)
(480, 640, 3)
[INFO] [1729894191.565911400] [image_subscriber]: Receiving video frame
[INFO] [1729894191.566431810] [image_subscriber]: Receiving person pos
[INFO] [1729894191.566936690] [image_subscriber]: Contagem: 0
[INFO] [1729894191.567489721] [image_subscriber]: Inside: {}
[INFO] [1729894191.570160144] [image_subscriber]: Pessoas que saíram: []
[INFO] [1729894191.601817265] [image_subscriber]: Receiving video frame
(480, 640, 3)
[INFO] [1729894191.602350747] [image_subscriber]: Receiving person pos
[INFO] [1729894191.602820826] [image_subscriber]: Contagem: 0
[INFO] [1729894191.603523187] [image_subscriber]: Inside: {}
[INFO] [1729894191.606160648] [image_subscriber]: Pessoas que saíram: []
(480, 640, 3)
[INFO] [1729894191.637538901] [image_subscriber]: Receiving video frame
[INFO] [1729894191.638044658] [image_subscriber]: Receiving person pos
[INFO] [1729894191.638506300] [image_subscriber]: Contagem: 0
[INFO] [1729894191.638941781] [image_subscriber]: Inside: {}
[INFO] [1729894191.641705403] [image_subscriber]: Pessoas que saíram: []
(480, 640, 3)
[INFO] [1729894191.673900142] [image_subscriber]: Receiving video frame
[INFO] [1729894191.675553871] [image_subscriber]: Receiving person pos
[INFO] [1729894191.676041560] [image_subscriber]: Contagem: 0
[INFO] [1729894191.676041560] [image_subscriber]: Inside: {}
[INFO] [1729894191.676041560] [image_subscriber]: Pessoas que saíram: []
(480, 640, 3)
[INFO] [1729894191.679225405] [image_subscriber]: Receiving video frame
(480, 640, 3)

[debug_node]: [yolov8_node] Configuring...
[debug_node]: [yolov8_node] Configured
[debug_node]: [yolov8_node] Activating...
[debug_node]: [yolov8_node] Activated
[tracking_node]: [tracking_node] Configuring...
[tracking_node]: [tracking_node] Configured
[tracking_node]: [tracking_node] Activating...
[tracking_node]: [tracking_node] Activated
[debug_node]: [yolov8_node] Activated
```

Fonte: Autor

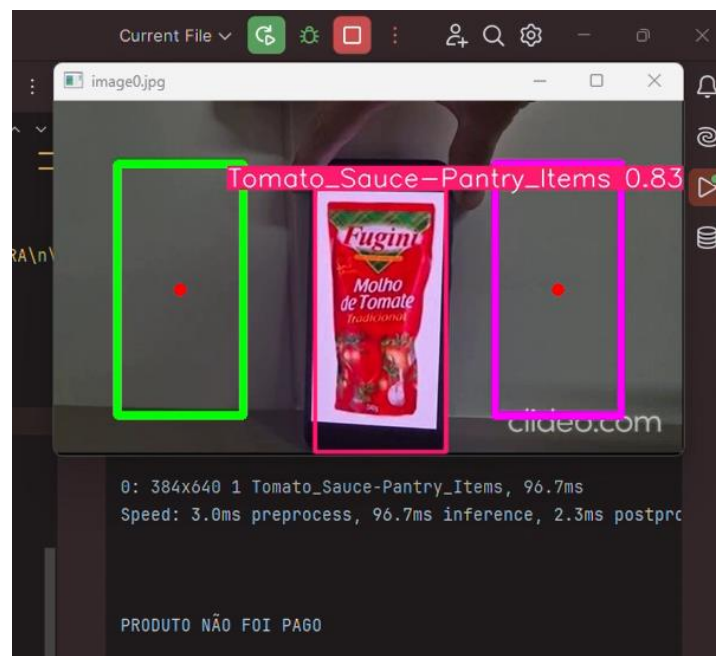
Para o algoritmo de reconhecimento de pagamento, os testes realizados foram mais simples, pois não se houve a necessidade de ir até o mercado, já que era possível testar apenas ajustando os retângulos da área da prateleira e de pagamento para os primeiros testes e depois, com o algoritmo validado, ajustar para a área do mercado. A seguir estão imagens demonstrando os testes

Figura 23 – Teste de produto na prateleira



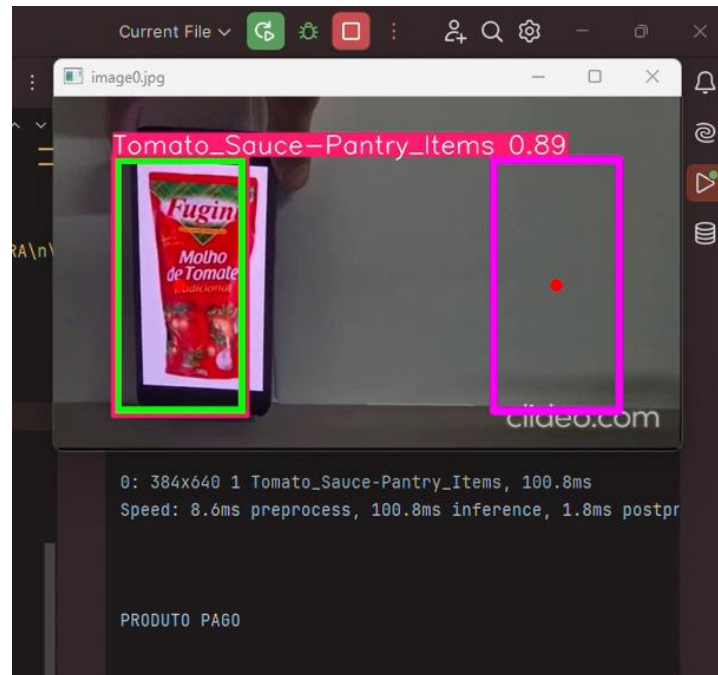
Fonte: Autor

Figura 24 – Teste de reconhecimento de produto não pago



Fonte: Autor

Figura 25 – Teste de produto pago



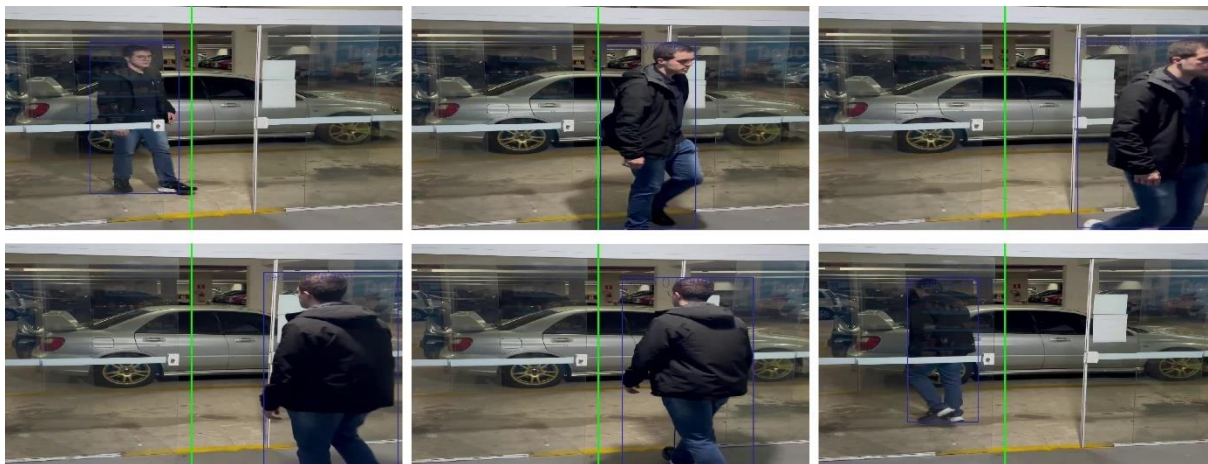
Fonte: Autor

Após os testes individuais de cada algoritmo, como validação final do projeto, foram feitas algumas gravações: uma para cada um dos casos mencionados anteriormente, simulando a câmera dos objetos, e, também, uma gravação de uma pessoa entrando e saindo, simulando a câmera de entrada.

No ambiente a primeira câmera foi posicionada em direção a porta, vale ressaltar que a câmera de entrada foi reposicionada para um local mais estratégico, de modo a ajudar no reconhecimento das pessoas. A segunda câmera foi posicionada ao lado do caixa, e o pacote de molho de tomate foi posicionado em uma prateleira perto da máquina de pagamento. Cada câmera estava enviando as imagens de vídeo para seu respectivo computador, e cada computador executava o seu respectivo código. Para cada um dos casos, o algoritmo foi capaz de reconhecer quando havia roubo e quando não havia.

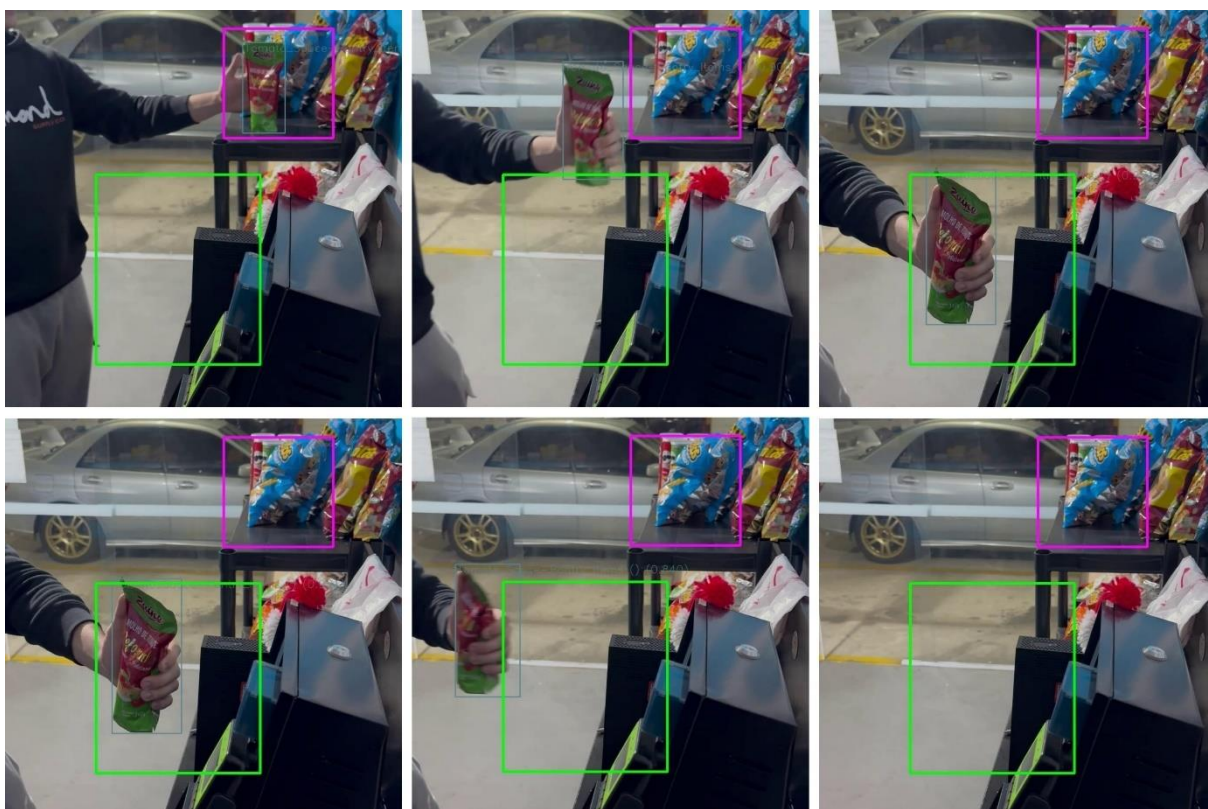
CASO 1 – HOUE PAGAMENTO DO PRODUTO

Figura 26 – Câmera de entrada detecta a pessoa entrando e saindo



Fonte: Autor

Figura 27 – Câmera do objeto detecta o produto sendo pago



Fonte: Autor

Figura 28 – Tópico /steal após a pessoa sair



Fonte: Autor

CASO 2 – HOUE ROUBO

Figura 29 – Câmera de entrada detecta a pessoa entrando e saindo



Fonte: Autor

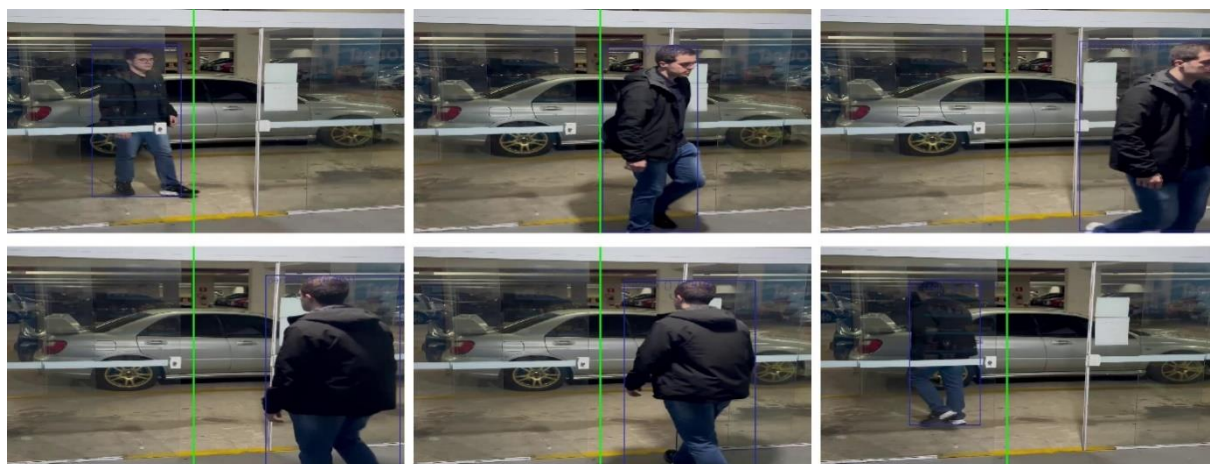
Fonte: Autor

[illegible]

Fonte: Autor

CASO 3 – PESSOA PEGOU O ITEM, DEVOLVEU E SAIU

Figura 32 – Câmera de entrada detecta a pessoa entrando e saindo



Fonte: Autor

Figura 33 – Pessoa pegando e devolvendo o produto



Fonte: Autor

Figura 34 – Tópico /steal após a pessoa sair



Fonte: Autor

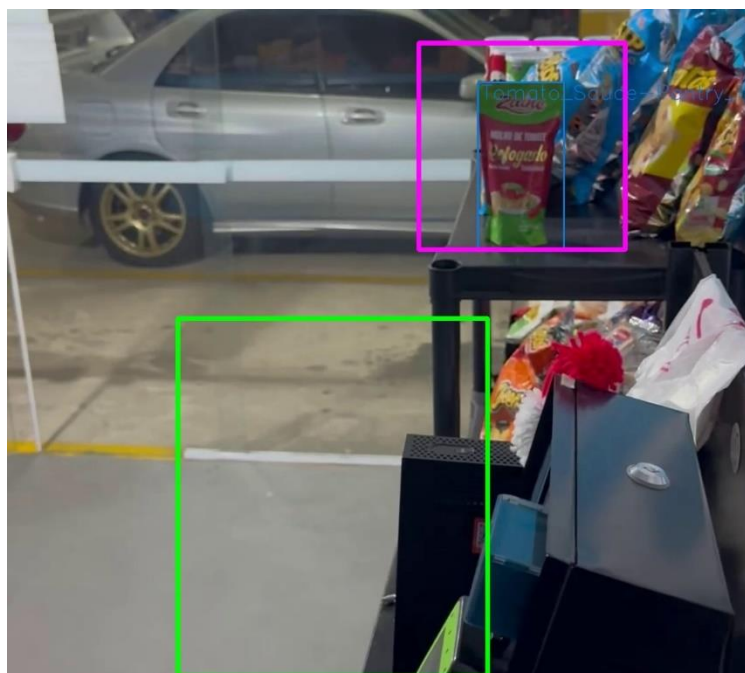
CASO 4 – PESSOA ENTROU E SAIU

Figura 35 – Câmera de entrada detecta a pessoa entrando e saindo



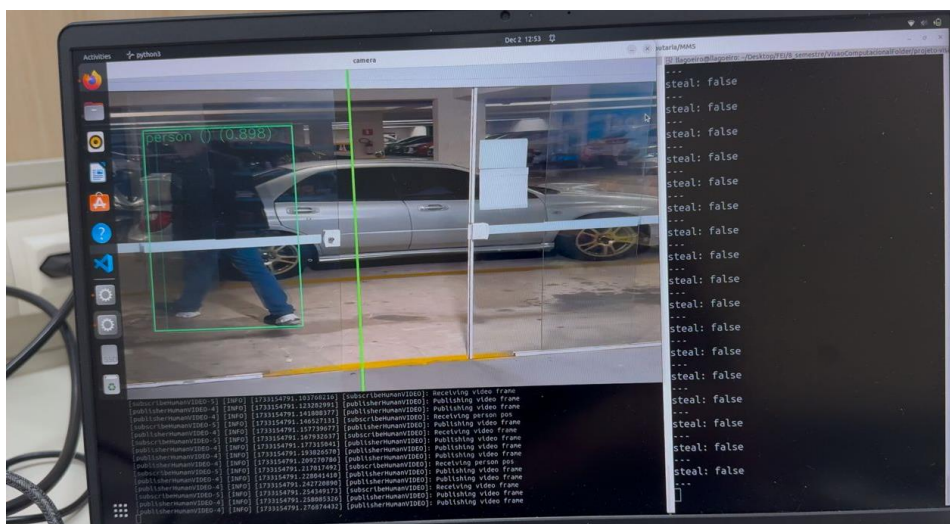
Fonte: Autor

Figura 36 – Câmera do objeto detectando o produto na prateleira



Fonte: Autor

Figura 37 – Tópico /steal após a pessoa sair



Fonte: Autor

6. CONCLUSÃO

Foi proposto solucionar os problemas de furtos de produtos que ocorrem em mercados de condomínios. Por meio do algoritmo desenvolvido e aqui descrito, o objetivo do projeto foi alcançado com êxito. Esse sucesso fica nítido ao observar os resultados obtidos na implementação simulada do projeto. O algoritmo desenvolvido demonstrou eficácia, tanto na

fase de testes quanto na implementação final, além de se provar viável nos quatro cenários propostos.

O projeto demonstrou as capacidades do que pode ser realizado com a Visão Computacional e o Aprendizado de Máquina, evidenciando sua eficácia na análise e interpretação de dados visuais em cenários complexos. Por meio de técnicas relativamente simples, foi possível realizar tarefas de identificação e monitoramento de maneira automatizada. Os resultados obtidos não apenas validaram a viabilidade técnica do projeto, mas também destacaram o impacto positivo dessa tecnologia em aplicações práticas.

Vale ressaltar a importância de se trabalhar com um modelo bem treinado. As métricas devem ser sempre analisadas durante o projeto, para que, durante a implementação real, o algoritmo trabalhe sempre com êxito. Foram realizados diversos treinos para que o modelo apresentasse as métricas apresentadas nesse relatório, que foram definidas como as melhores para essa aplicação.

Apesar do êxito do projeto, ainda há oportunidades de melhorias que poderão aumentar ainda mais o desempenho e eficiência do projeto. Aspectos como integração com um sistema embarcado, integração com sistema dos mercados de condomínio, ampliação da capacidade de identificação de produtos ao mesmo tempo e otimizar o sistema para operar com uma câmera só. Em suma, o projeto pode ser visto como um sucesso e abre diversas portas para futuras aplicações.

REFERENCIAS BIBLIOGRAFICAS

- [1] AZEVEDO, Itamar Junior de; CORDEIRO, Gabriel Andrade; GROCKOTZKI, Giovanni; et al. **Deteccção de Roubo de Computadores em Laboratório Usando Visão Computacional.** *Anais do Computer on the Beach*, 2020. Disponível em: <https://periodicos.univali.br/index.php/acotb/article/view/16737>. Acesso em: 04 nov. 2024.
- [2] TONIONI, Alessio. **Computer Vision and Deep Learning for Retail Store Management.** 2019. Tese (Doutorado) — Università di Bologna, Bologna, Itália. Disponível em: http://amsdottorato.unibo.it/8970/3/main_file.pdf. Acesso em: 04 nov. 2024.
- [3] MELEK, Ceren Gülra; SÖNMEZ, Elena Battini; VARLI, Songül. **Datasets and Methods of Product Recognition on Grocery Shelf Images Using Computer Vision and Machine Learning Approaches: An Exhaustive Literature Review.** *Engineering Applications of Artificial Intelligence*, v. 133, 2024. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197624006109>. Acesso em: 04 nov. 2024.
- [4] ULTRALYTICS. **YOLOv8 Performance Metrics.** Documentação oficial. Disponível em: <https://docs.ultralytics.com/pt/models/yolov8/#performance-metrics>. Acesso em: 14 out. 2024.
- [5] **TPJ Project. TPJ Dataset.** Roboflow, [s.d.]. Disponível em: <https://universe.roboflow.com/tpj-project/tpj/dataset/8>. Acesso em: 22 out. 2024.
- [6] MARTINS, Leo. **Robo Segmentation Dataset.** Roboflow, [s.d.]. Disponível em: <https://universe.roboflow.com/leo-martins-xpvd9/robo-segmentation/dataset/7>. Acesso em: 31 out. 2024.