Aplicação de Visão Computacional para Identificação de Códigos de Contêineres

Hugo Tomazi
UNIVALI
Itajaí - SC - Brasil
hugo.busnardo@edu.univali.br

Alex Rese
T.I.
Trust Group/UNIVALI
Itajaí - SC - Brasil
alexrese@univali.br

Rodrigo Lyra
Lab. Inteligência Aplicada
UNIVALI
Itajaí - SC - Brasil
rlyra@univali.br

Felipe Viel
Lab. de Sistemas Embarcados e Distribuídos
UNIVALI
Itajaí - SC - Brasil
viel@univali.br

Anderson Martins
T.I.
Trust Group
Itajaí - SC - Brasil
anderson.martin@trust.group

RESUMO

A evolução tecnológica tem desempenhado um papel essencial na otimização do setor logístico, promovendo a automação de procedimentos e impulsionando a competitividade no mercado. Um dos setores que se destaca no mercado é o de gestão da cadeia de suprimentos, porém, a dificuldade em controlar a entrada e saída de contêineres devido ao alto tráfego de caminhões nos pátios de armazenamento resulta em filas e atrasos. Com o desenvolvimento de uma ferramenta que facilite o controle de contêineres, é possível superar esse desafio. A aplicação de um software capaz de realizar a leitura automática e precisa do código identificador do contêiner em tempo real traz ganhos operacionais significativos. Neste contexto, este trabalho apresenta uma aplicação destinada à extração do código identificador de contêineres por meio de vídeos em tempo real com requisito de processamento de 30 quadros por segundo. Utilizando uma abordagem de pesquisa aplicada, foi elaborado um software que emprega métodos de reconhecimento de objetos e técnicas de processamento de imagens. Após uma análise da literatura, o método mais apropriado para o reconhecimento de objetos, o YOLOv8, foi selecionado. Adicionalmente, uma pesquisa de métodos de processamento de imagens foi conduzida para aprimorar a precisão da biblioteca de OCR Tesseract durante a etapa de reconhecimento de caracteres. Um terminal foi construído e equipado com duas câmeras para a realização e validação do software desenvolvido. Os resultados obtidos apresentaram uma taxa de precisão de 80% com apenas uma das câmeras, junto a análise dos casos de erro visando identificar melhorias futuras.

PALAVRAS-CHAVE

Códigos de contêineres, Visão Computacional, Identificação de Padrões, Reconhecimento de Caracteres.

1 Introdução

Nos últimos anos, a convergência entre o campo da tecnologia e a logística tornou-se uma associação fundamental para as estratégias das empresas. Integrar e alavancar as inovações tecnológicas tem sido reconhecida como um diferencial competitivo, impulsionando a eficiência e a competitividade.

Segundo Bowersox, Closs e Ballou em 2001 [1] a integração destas áreas é um fator determinante para potencializar a eficiência logística, um conceito que se mantém relevante até os dias atuais. No entanto, apesar do reconhecimento da importância estratégica da Tecnologia da Informação (TI), muitas empresas ainda não exploraram plenamente seu potencial, conforme apontado por Andersen e Segars em 2001 [2]. Em análise comparativa entre os portos brasileiros e os do BRICS, conduzida por Ribeiro, Fraga e Clarkson em 2017 [3], evidenciou-se o potencial transformador das TIs no cenário logístico. Identificaram soluções concretas para desafios persistentes, como a alocação de berços, detecção de objetos e distribuição de cargas.

Paralelamente, a influência da inteligência artificial (IA) nas atividades logísticas, tem sido objeto de grande destaque. Nadimpalli em 2017 [4] evidencia benefícios tangíveis ao empregar na previsão e identificação de obstáculos na cadeia de suprimentos, resultando em maiores probabilidades de alcançar resultados positivos. Ressaltando a importância crescente da IA como um recurso fundamental para otimização e eficácia na gestão logística. Segundo Heilig e Voß em 2016 [5], nos períodos de maior demanda, os portos frequentemente enfrentam desafios de congestionamento nos pátios quanto nos portões terminais, resultantes do volume considerável de caminhões que chegam para operações. Esses pontos levantados permitem evidenciar a crescente necessidade na logística global de sincronizar diversos sistemas e agilizar a coleta de informações, visando coordenar eficientemente processos produtivos e serviços de valor agregado. Além disso, permitisse melhorar o atendimento dos requisitos de clientes por redução de custos, execução ágil dos serviços, confiabilidade

De acordo com a Confederação Nacional da Indústria (CNI), a eficiência logística do comércio exterior se apresenta como o principal fator impactante na competitividade das exportações brasileiras [6]. Dentro desse contexto, a ineficiência nos processos

de manuseio e embarque de cargas nos portos emerge como um dos elementos prejudiciais à competitividade dessas exportações.

Estudos realizados identificam diversas áreas críticas que agravam esse panorama desafiador. Destacam-se a complexidade e a sobrecarga documental exigida para operações de exportação, somadas aos procedimentos burocráticos para liberação e despacho de cargas. Estes processos, frequentemente carentes de padronização, consomem um tempo substancialmente maior do que o necessário. Esses fatores reforçam a urgência e a relevância de investir em ferramentas que possam agilizar os procedimentos burocráticos inerentes às operações logísticas [6].

Com a crescente integração da tecnologia na automação de processos e atividades, os portos se viram compelidos a se adaptar para atender às demandas dos clientes, que incluem a necessidade de respostas ágeis e a redução de custos [6]. Como resultado, portos têm buscado a implementação de uma variedade de tecnologias para automatizar suas operações e garantir um nível de serviço que atenda aos padrões estabelecidos.

Dentre as tecnologias adotadas, destacam-se a identificação por radiofrequência (RFID) e os sistemas de reconhecimento óptico de caracteres (OCR) [5]. Soluções são comumente empregadas nos portões de entrada e saída, permitindo a identificação automatizada dos códigos dos contêineres. Essa automação parcial dos procedimentos administrativos e de verificação resulta em um aumento da capacidade dos portos em lidar com um maior volume de contêineres, enquanto reduz a necessidade de intervenção humana nesses processos.

Atualmente o OCR pode ser utilizado para extrair textos de documentos, livros, manuscritos e imagens [15]. Esse é um campo da visão computacional explorado a décadas. Os primeiros sistemas de OCR comerciais surgiram em meados de 1950 e em 1954 a primeira máquina de OCR foi instalada. Em 1990, com os avanços tecnológicos da área, o processamento de imagens e reconhecimento de padrões foi combinado com a inteligência artificial para criar sistemas de OCR extremamente eficazes. Atualmente, com a melhoria da precisão dos equipamentos eletrônicos como câmeras, scanners e demais avanços na área da computação é possível desenvolver sistemas que aplicam as técnicas de OCR satisfatóriamente [16].

Este artigo apresenta uma contribuição na área de visão computacional e identificação de caracteres, utilizando técnicas avançadas como aprendizado de máquina e OCR. Busca-se extrair automaticamente esses códigos a partir de imagens capturadas por câmera de vídeo. A solução desenvolvida foi aplicada e validade em ambiente real.

2 Problematização

A modernização do setor logístico enfrenta desafios consideráveis, evidenciados na literatura especializada. Um dos problemas centrais reside na burocratização e na lentidão dos procedimentos de entrada e saída de caminhões nos pátios, frequentemente resultando em longas filas e atrasos. O processo manual de cadastramento dos códigos de contêineres é propenso a erros humanos, exigindo, em alguns casos, recursos dedicados ininterruptamente para registrar as movimentações durante 24 horas por dia, gerando ainda mais custos. Além disso, a falta de atualização imediata nos sistemas após esses registros manuais prejudica a integração e eficiência dos sistemas de gerenciamento de armazéns e transporte [7][8].

Diante desses desafios, empresas frequentemente recorrem a soluções de mercado onerosas, envolvendo por vezes custos elevados de implementação e mensalidades. Esse modelo de negócios pode representar um ônus financeiro significativo para as empresas, limitando sua capacidade de inovar e desenvolver soluções personalizadas. É essencial buscar alternativas que permitam um acesso mais flexível aos códigos de identificação de contêineres, sem depender excessivamente de serviços pagos de terceiros. Uma estratégia promissora pode ser investir em capacidades internas de desenvolvimento, explorando tecnologias de visão computacional e aprendizado de máquina [6].

Este artigo apresenta uma alternativa aliando empresa (Trust Group) e pesquisa (Trabalho de Conclusão de Curso - Univali) no desenvolvimento de uma solução: a aplicação técnica de visão computacional para automatizar a leitura de códigos de contêineres. Essa abordagem visa reduzir custos operacionais, com contratação de serviços empresas terceiras eliminando a necessidade de recursos dedicados e mensalidades. Busca-se automatizar os processos de reconhecimento de carga, melhorando a eficiência geral das operações sem exigir os altos investimentos associados às soluções tradicionais do mercado.

3 Desenvolvimento

A necessidade por soluções ágeis e eficientes direcionou o foco deste estudo para o desenvolvimento de uma aplicação que visa a automatização do processo de identificação de contêineres. Assim a abordagem proposta está dividida em cinco fases que compõem a execução da aplicação sendo: (i) obtenção das imagens; (ii) detecção do código do contêiner; (iii) pré-processamento, (iv) detecção dos caracteres e (v) verificação de integridade. Na Figura 1 é possível visualizar o fluxo de execução das etapas descritas. Para desenvolver a aplicação foi adotado o uso da linguagem Python pois esta conta com uma grande gama de bibliotecas voltadas as áreas de visão computacional.

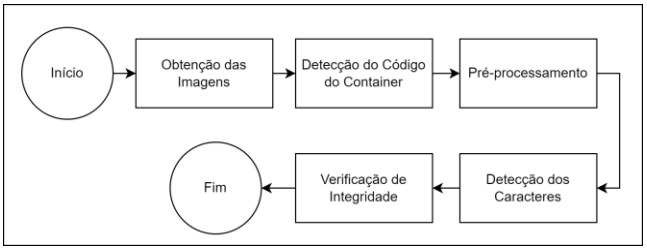

Figura 1: Fluxo geral de execução da aplicação proposta.

3.1 Obtenção das Imagens

A primeira fase do processo recebe imagens como entrada para passar pelo processo de processamento. Para capturar essas imagens, um terminal foi construído na entrada de um centro de armazenamento situado no bairro Salseiros, em Itajaí - Santa Catarina, pertencente à empresa Trust Group. Este terminal foi equipado com duas câmeras e um refletor de luz, utilizado durante os períodos noturnos de entrada e saída de fluxo. As câmeras foram estrategicamente posicionadas e reguladas para capturar os códigos dos contêineres transportados pelos caminhões que circulam na área. Na Figura 2 pode ser observado o terminal a partir de duas perspectivas distintas, demonstrando o posicionamento geral da estrutura e das câmeras instaladas.

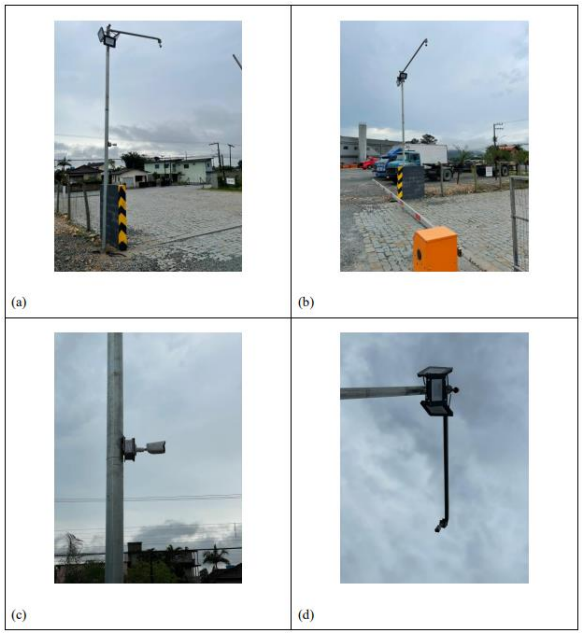

Figura 2: Terminal construído: (a) Visão geral do terminal; (b) Visão geral do terminal em outra perspectiva; (c) Câmera lateral posicionada; (d) Câmera superior e refletor posicionados.

As câmeras foram posicionadas de forma a capturar a parte superior e lateral do contêiner. Na Figura 3 (a) é possível visualizar o ângulo de captura da câmera superior, já a Figura 3 (b) evidencia o ângulo de captura da câmera lateral. Durante a execução inicial da aplicação desenvolvida, foi possível perceber que em alguns casos as imagens geradas pelas câmeras continham o código do contêiner borrado devido à alta velocidade em que o caminhão trafega abaixo do terminal. Para resolver este problema foi necessário aumentar a velocidade do obturador das câmeras para o equivalente a 1/2000. Ao aumentar a velocidade de abertura e fechamento do obturador, diminui-se o tempo de exposição da câmera a luz, gerando uma imagem sem esmaecimento.

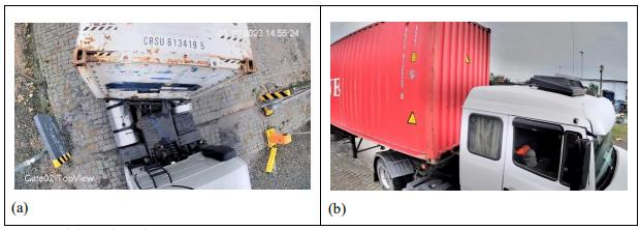

Figura 3: Ângulo capturado pelas câmeras: (a) Imagem capturada pela câmera superior; (b) Imagem capturada pela câmera lateral.

Para captar as imagens das câmeras foi utilizado o software FFmpeg. O FFmpeg suporta entradas como imagens, arquivos de vídeo e streams de câmeras (protocolo RTSP). Este foi adotado devido a sua variedade de entradas, confiabilidade e desempenho, ao qual resulta no ganho de flexibilidade ao desenvolver a aplicação facilitando as etapas de testes e execução em campo. Ao receber um arquivo de vídeo ou stream de câmera como entrada, o FFmpeg produz uma saída binária quadro a quadro do vídeo. Na aplicação desenvolvida foi utilizada a biblioteca "ffmpeg-python" para capturar a saída gerada pelo FFmpeg. As imagens capturadas são repassadas para a próxima fase do projeto descrito na seção 3.2.

3.2 Detecção do Código do Contêiner

A fase de detecção do código do contêiner visa extrair os locais de interesse das imagens capturadas na fase anterior (3.1). Para realizar a detecção do local de interesse foi treinado e utilizado um modelo de detecção de objetos. O modelo escolhido para uso foi o YOLOv8 disponibilizado pela Ultralytics [9]. A Ultralytics disponibiliza 5 variações do modelo YOLOv8 no formato "PyTorch", nomeadas como YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l e YOLOv8x. A diferença entre elas é o tamanho de cada um dos modelos sendo que o menor deles é o YOLOv8n contendo 3.2 milhões de parâmetros e o maior é o YOLOv8x contendo 68.2 milhões de parâmetros. Visando obter uma boa taxa de processamento mantendo uma qualidade de inferência aceitável, foi optado por treinar o modelo YOLOv8s contendo 11.2 milhões de parâmetros.

Para treinar o modelo, foi realizada uma pesquisa na literatura a procura de bases de dados contendo imagens de contêineres com seus devidos códigos visíveis. No site Roboflow [10] foram encontradas várias bases de dados públicas. Em sequência foram baixadas as bases e unidas, gerando uma nova base de dados com aproximadamente 2000 imagens. Foi realizado um treinamento inicial da rede utilizando as bases recuperadas da internet. Ao executar a aplicação desenvolvida percebeu-se uma falha ao detectar os códigos de contêineres dos caminhões que trafegam abaixo do terminal construído (descrito na seção 3.1).

A partir desta observação, foram capturadas e rotuladas cerca de 880 imagens das câmeras instaladas no terminal construído para refazer o treinamento do modelo visando corrigir a falha que afeta diretamente a acurácia da aplicação. Entre as 2880 imagens contidas na base de dados final, existem diversas características que

as diferenciam. Como exemplo dessas características, pode ser citado o ângulo de captura do código, contêineres com diversas texturas (ferrugem, amassados e outros), imagens capturadas durante o dia/noite, desfoques e outros. Entre essas características, existem imagens de fundo (comumente chamadas de background) a qual não contém o código do contêiner em evidência. Dessa forma é possível treinar o modelo para atuar em cenários onde o código não existe, melhorando assim a sua acurácia final. Em seguida foi realizada uma revisão final para excluir imagens repetidas e por fim foi realizado a geração das classes da base de dados de maneira a detectar três classes de textos. As classes identificadas são referentes ao código do proprietário juntamente com o carácter identificador do equipamento, número serial e por fim, dígito verificador. Para realizar a identificação das classes das imagens foi utilizado o software YoloLabel na versão 1.2.1 ao qual oferece simplicidade e rapidez na tarefa [11]. A Figura 1 exibe exemplos de imagens contidas na base de dados já etiquetada. Para realizar o treinamento do modelo a base de dados foi aleatorizada e separada, mantendo 70% das imagens para treinamento, 20% para teste e 10% para validação. Ao receber o quadro da etapa de obtenção das imagens (3.1), é realizada a inferência deste utilizando o modelo de rede neural treinado extraindo as classes detectadas e obtendo o recorte dos locais de interesse contendo o código do contêiner. Os recortes obtidos são repassados para etapa de pré-processamento (3.3).

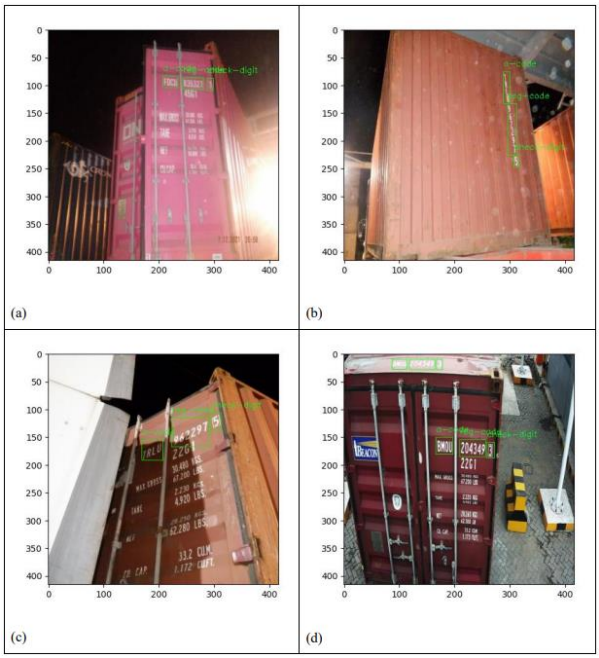

Figura 4: Exemplo de imagens contidas na base de dados etiquetada: (a) Código identificado na porta do contêiner e na horizontal; (b) Código identificado na vertical; (c) Código detectado na porta do contêiner a partir de uma perspectiva diferente de (a); (d) Código identificado respectivamente na parte de trás e em cima do contêiner.

3.3 Pré-processamento

Na etapa de pré-processamento, são aplicados alguns métodos de tratamento de imagens aos locais de interesse extraídos na etapa anterior contendo os textos alvo (descrita na seção 3.2). Os métodos de processamento de imagens são aplicados visando melhorar a acurácia do reconhecimento de caracteres que será executado na fase seguinte descrita na seção 3.4. Na primeira versão da aplicação foram implementadas quatro técnicas de processamento de imagem sendo: (i) aumento de resolução, que consiste em um processo de interpolação para incrementar o número de pixels; (ii) esmaecimento Gaussiano, método de suavização que atenua ruídos e detalhes, aplicando um filtro com pesos definidos por uma distribuição Gaussiana; (iii) conversão para escala de cinza, técnica que transforma uma imagem colorida em escala de cinza e (iv) limiarização, processo que converte uma imagem em preto e branco. No entanto, foi observado que a limiarização não produziu os resultados esperados, comprometendo a eficácia no reconhecimento de caracteres. Diante dessa constatação, foi optado por eliminar o desfoque Gaussiano e a limiarização, substituindo-os pela técnica de correção de alinhamento de texto, processo que ajusta a inclinação ou distorção do texto em uma imagem, alinhando-o corretamente para facilitar o reconhecimento óptico de caracteres. Com este ajuste, as técnicas finais de processamento de imagem adotadas foram: (i) conversão para escala de cinza; (ii) o aumento da resolução espacial e (iii) correção de alinhamento de texto.

De acordo com Gilbey et al. em 2021 o modelo Tesseract (utilizado na fase seguinte, descrita na seção 3.4) utiliza imagens com 300 dpi em seu treinamento [12]. Sendo assim, é necessário realizar o aumento da resolução espacial das imagens recortadas na etapa anterior (descrita na seção 3.2) para equalizar a essa medida visando obter um resultado satisfatório. O algoritmo para realizar o aumento da resolução é descrito na documentação da biblioteca OpenCV como "*pixel area relation*" [13].

Quanto a etapa de conversão para escala de cinza, essa visa diminuir a quantidade de informações referentes a cores contidas nas imagens. Já a correção de alinhamento, é aplicada em momentos em que o código estiver desalinhado com a angulação da câmera, sendo necessário corrigi-lo visando obter um melhor resultado na etapa de detecção de caracteres (3.4). Para executar o alinhamento do texto é realizado um cálculo para descobrir o ângulo entre duas partes do código detectado na etapa de detecção do código do contêiner (3.2). O cálculo do ângulo é realizado com base no ponto central das duas partes detectadas, dessa forma é calculado o arco tangente entre os dois pontos, resultando assim no ângulo necessário para rotação da imagem. A função para calcular o arco tangente é implementada nativamente na biblioteca *math* do Python. A partir do ângulo descoberto é realizada a correção da rotação das imagens de forma que o texto fique alinhado a 180 graus. Vale destacar que esse fluxo de pré-processamento utiliza também algoritmos presentes na biblioteca OpenCV. Na Figura 5 é possível visualizar a transformação da imagem conforme é aplicado cada método de pré-processamento.

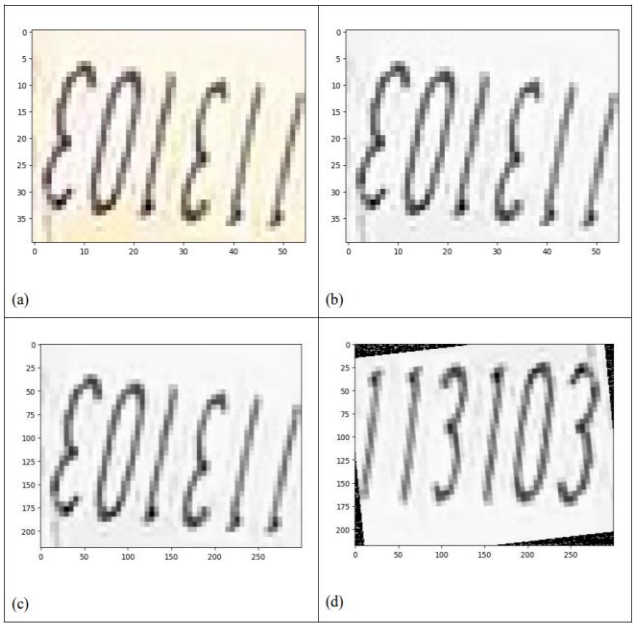

Figura 5: Execução do pré-processamento: (a) Imagem original; (b) Imagem em (a) convertida para escala de cinza; (c) Imagem em (b) com a resolução aumentada; (d) Imagem em (c) com o alinhamento corrigido.

3.4 Detecção dos Caracteres

Essa etapa visa detectar e reconhecer o texto contido na imagem pré-processada na fase anterior (descrita na seção 3.3). Para realizar a detecção dos caracteres é utilizada a biblioteca Tesseocr a qual possibilita a execução do modelo de OCR Tesseract no Python. Essa biblioteca foi escolhida devido ao seu histórico de uso e existência, além da sua fácil implementação.

O reconhecimento dos caracteres é realizado com base na classe da imagem a qual é submetida a detecção no momento. Sendo assim, caso a imagem corresponda ao dígito verificador ou código serial do contêiner, é aplicado um filtro em tempo de execução para detectar apenas dígitos. Quando se trata do código do proprietário, o filtro aplicado reconhecerá apenas letras maiúsculas. Esse é aplicado para evitar erros relacionados à similaridade entre letras e números, exemplo as letras "O" e "I" tem um grau de similaridade com os números "0" e "1" respectivamente.

3.5 Validação do Dígito Verificador

A fase de verificação de integridade realiza a validação dos caracteres reconhecidos na etapa anterior (descrita na seção 3.4). É necessário realizar essa etapa para garantir que o texto extraído corresponda a um código de contêiner válido. Para realizar a verificação são realizados alguns testes nos textos extraídos de maneira a verificar se estes estão minimamente no formato esperado. Na ISO 6346 de 1995, é especificado o padrão que o código de identificação do contêiner deve seguir [14]. Neste trabalho é realizada a validação dos textos referentes as classes detectadas pela rede YOLOv8. As classes são o código do proprietário (ao qual inclui o identificador da categoria do equipamento), número de serial e dígito verificador. Na Tabela 1, é especificado de maneira detalhada a regra verificada para cada classe.

Código do Proprietário	Número Serial	Dígito Verificador
Composto por 4 letras maiúsculas ao qual a última letra deve ser "U". A sequência de caracteres deve estar na lista de códigos BIC.	Composto por 6 digítos	Composto por 1 dígito.

Tabela 1: Composição textual das classes de texto detectadas.

O código *Bureal International des Containers* (BIC) supervisiona os padrões para contêineres intermodais, comumente chamados de contêineres de transporte. Consiste em uma sequência de três letras maiúsculas que correspondem ao código do proprietário do contêiner. Esse código é adicionado em uma base de dados centralizadora e disponibilizada na internet. Cada texto extraído na etapa de detecção dos caracteres (3.2) passa pelos testes de composição básica das classes.

Na etapa de verificação de integridade, é implementada uma sessão de detecções. A sessão de detecções dura um período de 10 segundos e é inicializada quando alguma classe é detectada pelo classificador executado na etapa de detecção do código do contêiner (3.2). Uma vez inicializada a sessão de detecções, todos os textos detectados na etapa de detecção de caracteres (3.4) são salvos em vetores correspondentes a cada classe do código detectado. Quando a sessão de detecções é finalizada, um novo vetor contendo uma combinação para cada parte do código detectado é criado e ordenado de forma decrescente pelo somatório da quantidade de vezes que cada parte do código foi detectado. A ordenação decrescente do vetor tem como objetivo maximizar as chances de encontrar o resultado correto, dado que quanto mais vezes um texto é detectado em uma seção de detecções, maiores serão as chances deste corresponder ao texto correto. Por fim, o vetor final é percorrido e cada combinação é submetida a uma validação de integridade utilizando o dígito verificador. A função do dígito verificador é servir como um parâmetro para checar se os caracteres detectados nas classes "código do proprietário" e "número serial" são válidos. Na ISO 6346 de 1995 também é determinado o método ao qual deve ser utilizado para realizar a verificação do texto utilizando o dígito verificador. Se a combinação validada corresponder corretamente ao dígito verificador detectado, é assumido que se chegou ao resultado correto.

4 Análise e Resultados

Para realizar a análise dos dados foi necessário salvar automaticamente os vídeos e resultados que serão gerados pela aplicação desenvolvida, permitindo assim que a análise possa ser efetuada posteriormente.

Para executar a captura do vídeo, todas as sessões de detecção (descrita na seção 3.5) foram gravadas. A inicialização da sessão começa quando ao menos uma das classes de texto (código do proprietário, código serial ou digito verificador) é detectada na etapa de detecção do código do contêiner. Os quadros capturados são salvos gerando um arquivo de vídeo no formato MP4.

O resultado das detecções é salvo em um arquivo de texto ao qual contém os dados do código detectado. Entre os dados salvos estão data, hora e texto detectado. A partir do vídeo gerado e dos dados coletados, é realizada uma análise manual de cada vídeo e criada uma base de dados contendo as informações da data e hora de detecção, código detectado, acerto positivo ou negativo e se o código se encontra rasurado (leitura, detecção e classificação comprometidas). Um código é considerado rasurado quando um ou mais códigos existentes na gravação estão com a leitura comprometida. A leitura pode ser comprometida devido à qualidade do vídeo ou a condições de preservação do código do contêiner (ferrugens, manchas, amassados, desgaste e outros). Os dados foram coletados em um período de 42 dias (27/09/2023 à 07/11/2023). Neste período foi possível capturar a passagem de 356 caminhões com carregamento de contêineres marítimos. Dentre os vídeos gravados, encontram-se variações na iluminação (decorrentes do horário e clima), clima (chuvoso, nublado e ensolarado) e ventanias, ou mesmo o deslocamento dos próprios caminhões, as quais esporadicamente causam tremores na estrutura montada, ocasionando ruídos.

A aplicação desenvolvida foi testada em campo utilizando o terminal construído e descrito na seção 3.1. Durante o período de testes, foram identificados aperfeiçoamentos relevantes que poderiam melhorar o desempenho da aplicação desenvolvida. Sendo assim, as gravações coletadas foram separadas em 4 lotes e nomeados de lote 1, 2, 3 e 4 respectivamente. Cada lote representa uma versão da aplicação que foi testada em campo e contém uma alteração em configurações que gerou uma melhoria. Na Tabela 2, é possível visualizar as alterações realizadas de forma resumida, sendo a primeira coluna o lote, a segunda o período, a terceira a quantidade de detecções e a quarta a alteração realizada. É importante ressaltar que no terceiro lote foi necessário um período maior de avaliação devido à quantidade de caminhões que passaram pelo terminal, a qual foi consideravelmente menor que nos outros períodos devido ao fechamento dos portos referente ao grande volume de chuvas na região do vale do Itajaí. Fator que impactou consideravelmente na quantidade de testes.

Lote	Período de Avaliação	Quantidade Avaliado	Alterações
Lote 1	27/09 à 01/10	21	Versão inicial, nenhuma alteração
Lote 2	02/10 à 08/10	108	Técnicas de Pré-processamento, Treinamento da Base de Dados
Lote 3	09/10 à 26/10	80	Treinamento da Base de Dados, Configuração do Obturador da Câmera
Lote 4	27/10 à 07/11	147	Configuração de Bitrate da Câmera

Tabela 2: Resumo das alterações por lote, período de avaliação e quantidade de detecções avaliadas.

Na Tabela 3 é possível visualizar os resultados referentes a porcentagem de acerto, porcentagem de códigos com rasura e porcentagem de detecções com falso positivo obtidos na análise de cada lote. Os falsos positivos são contabilizados quando o código detectado pela aplicação não condiz com o código real contido no contêiner. A porcentagem de rasura se refere à quantidade de códigos com rasura quando considerado o montante total.

Durante os testes do lote 1, foi identificado que a aplicação em sua primeira versão não conseguiu obter um resultado satisfatório. Após realizar uma análise dos vídeos deste lote, chegou-se à conclusão de que os códigos localizados na parte superior do contêiner (capturados pela câmera superior do terminal) não estavam sendo identificados. Isso devido à falta de imagens que contemplam o ângulo superior do contêiner para o treinamento do modelo de detecção de objetos. A falta de imagens que exibem o código a partir deste ângulo ocasiona no mau desempenho da detecção do código do contêiner, invalidando a maioria das imagens recuperadas das câmeras. Percebeu-se também que as etapas de pré-processamento da imagem não estavam gerando um bom resultado. A técnica de limiarização e esmaecimento Gaussiano geravam imagens ilegíveis devido a fatores externos como iluminação e ruídos. Ao analisar os vídeos, foi possível identificar também que a biblioteca Tesseract não estava desempenhando um bom resultado devido à falta de alinhamento do texto na imagem, problema esse identificado na maioria dos casos. A partir destas análises, foram realizadas algumas alterações na aplicação desenvolvida. Foi removida a técnica de limiarização e esmaecimento Gaussiano da etapa de pré-processamento e adicionada uma técnica de correção de alinhamento da imagem (3.3). Foi também realizado um novo treinamento do modelo de

detecção de objetos. Para isso foram selecionadas algumas imagens obtidas no lote 1 e inseridas na base de imagens já existente (3.2).

Ao analisar o segundo lote de vídeos, foi possível identificar uma melhoria de 57% em relação ao primeiro lote, evidenciando que as ações tomadas refletiram positivamente no desempenho da aplicação. Por outro lado, percebe-se que a porcentagem de falsos positivos aumentou consideravelmente no lote 2. Ao visualizar os vídeos gerados nesse lote, percebeu-se que em alguns momentos o modelo de detecção de objetos treinado detectava alguns textos contidos na imagem como falso positivo, afetando negativamente o alinhamento do texto. Percebeu-se também que em alguns casos a câmera gerou imagens distorcidas devido à velocidade em que os caminhões trafegam. Para minimizar esse problema foi necessário alterar as configurações do obturador das câmeras de maneira a remover os ruídos gerados. Decidiu-se também por, novamente, selecionar algumas imagens capturadas no lote 2 para adicioná-las a base de imagens de treinamento do modelo. Ao final, a base de imagens contém aproximadamente 2880 imagens conforme descrito na seção 3.2. Em seguida foi realizado um novo treinamento do modelo de detecção de objetos e adicionado o mesmo a aplicação.

No lote 3 é possível perceber a melhoria do sistema em relação ao lote 2. O terceiro lote obteve uma porcentagem de 73% de acerto, apresentando um aumento de 16% quando comparado com o lote 2. Quando analisado a porcentagem de falsos positivos, é possível perceber uma diminuição de 5% nesse indicador, evidenciando novamente que as ações tomadas impactaram positivamente nos resultados da aplicação. Por outro lado, percebeu-se que as imagens captadas pelas câmeras muitas vezes decaiam a qualidade. Ao analisar com mais detalhes foi identificado que a taxa de bits (bitrate) das câmeras estava muito baixa (1024 KB por segundo). Sendo assim, foi realizada a correção aumentando a configuração referente ao bitrate das câmeras para 8192 KB por segundo.

Ao analisar o lote 4 foi identificado que a aplicação obteve uma taxa de 80% de acertos, evidenciando um aumento de 7% quando comparado com o lote 3. Foi possível identificar uma diminuição de 6% na taxa de rasuras, resultado esse que vem do ajuste realizado na configuração referente ao bitrate das câmeras.

Foi observado durante a execução da aplicação que os códigos que se encontram alinhados na vertical não estavam sendo detectados. Este fato evidência que o uso da biblioteca de OCR Tesseract não alcançou um desempenho aceitável ao detectar textos alinhados verticalmente. Outro fato que afeta negativamente os resultados é a rasura do texto. Analisando os dados referentes a porcentagem de códigos com rasura, pode-se perceber que quase metade dos contêineres que trafegam pelo terminal tem algum tipo de rasura no código. A rasura pode ser originada de defeitos na lataria do contêiner ou perca de qualidade da imagem. Em alguns casos foi possível perceber uma queda na qualidade das imagens devido a intermitências da rede de internet local e velocidade em que o caminhão trafega, chegando a aproximadamente 25 km/h. Após executar o levantamento dos dados envolvendo apenas os casos de erro, é possível identificar que a maioria dos erros pode ser originada da rasura existente na imagem. Para analisar a relação da rasura com os erros foi observado todos os vídeos, independente do lote, foi identificado que 67,5% dos erros da aplicação podem ter relação com a rasura. Outro fator que pode afetar negativamente no desempenho da aplicação tem relação com contêineres nacionalizados. Os contêineres nacionalizados são desativados e utilizados apenas para carregamento de cargas em rodovias, não podendo trafegar em transporte marítimo. Para identificar estes, é necessário que a lista de códigos BIC da aplicação esteja atualizada, desta maneira evita-se a detecção de falso positivos.

Lote	Porcentagem de Acerto	Porcentagem de Rasura	Porcentagem de Falso Positivo
Lote 1	0%	47%	4%
Lote 2	57%	54%	12%
Lote 3	73%	52%	7%
Lote 4	80%	46%	7%

Tabela 3: Evolução dos resultados da aplicação desenvolvida.

5 Considerações Finais

Neste trabalho foi desenvolvida uma aplicação de visão computacional para reconhecimento e leitura de códigos de contêiner suportando uma taxa de processamento de 30 quadros por segundo. A partir das pesquisas realizadas no campo de reconhecimento de caracteres, detecção de objetos e processamento de imagens foi possível identificar os métodos e etapas que melhor atendem o projeto. Além disso, a aplicação desenvolvida foi testada em campo e seus resultados foram recuperados para uma análise geral de acurácia.

Ao realizar a análise final da aplicação foi possível obter uma acurácia de 80%. É possível também identificar que praticamente metade dos contêineres analisados contém algum tipo de rasura, seja ela provinda de razões externas (amassados, arranhões…) ou de oscilações na qualidade das imagens capturadas pelas câmeras (perca de pacotes, oscilações de rede, tremores e outros). Foi possível identificar também que a biblioteca Tesseract não está atingindo um bom resultado ao detectar os códigos de contêineres posicionados na vertical. A partir dos resultados obtidos por meio das análises é possível identificar alguns pontos de melhoria na aplicação que podem ser estudados em trabalhos futuros. O primeiro ponto tem relação com o estudo da melhoria da imagem captada pela câmera. Pode-se realizar estudos na área de recuperação de imagens de modo a melhorar a qualidade da imagem obtida removendo seus ruídos. Outra melhoria que pode impactar positivamente nos resultados da aplicação tem relação com a leitura do código do contêiner na vertical. A partir de pesquisas relacionadas a métodos e bibliotecas de OCR voltados a detecção de textos na vertical é possível obter uma melhoria nos resultados.

AGRADECIMENTOS

Agradecemos o apoio financeiro a empresa Trust Group e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) – Processo Nº 140368/2021-3.

REFERÊNCIAS

[1] Bowersox, D. J.; Closs, D. J. (2001) Logística Empresarial. O Processo de integração da cadeia de suprimento. São Paulo: Atlas

[2] Andersen, Torben; Segars, Albert. The impact of IT on decision structure and firm performance. In: Zhang, Han. Information & Management. Atlanta, Georgia: Elsevier, 2001. p. 85-100

[3] Ribeiro, Fraga, Clarkson. Gestão de portos brasileiros e do BRICS: uma análise comparativa sobre seus problemas logísticos e a resolução por meio da tecnologia da informação. Exacta – EP, São Paulo, v. 15, n. 2, p. 335-351, 2017.

[4] Nadimpalli, Meenakshi. Artificial intelligence risks and benefits. International Journal of Innovative Research in Science, Engineering and Technology, v. 6, n. 6, 2017.

[5] Heilig, Leonard; Voß, Stefan. Information systems in seaports: a categorization and overview. Information Technology and Management, v. 18, p. 179-201, 2016.

[6] Andrade, Robson; Desafios à Competitividade das Exportações Brasileiras. Brasília: Confederação Nacional da Indústria, 2022.

[7] Santos, Eduardo Biagi Almeida. As dificuldades logísticas de acesso e de movimentação de cargas do porto de santos. IX Simpósio de Excelência em Gestão e Tecnologia, 2012.

[8] Barbosa, Danilo; Careta, Catarina; Musetti, Marcel. A tecnologia da informação e comunicação na logística: estudo de caso em uma cadeia de suprimentos. XIV SIMPEP, 2007.

[9] Jocher, Gean. Ultralytics. 2023. Disponível em: https://github.com/ultralytics/ultralytics. Acesso em: 18 dez. 2023.

[10] Roboflow, Equipe. Roboflow: everything you need to build and deploy computer vision models. Everything you need to build and deploy computer vision models. 2023. Disponível em: https://roboflow.com/. Acesso em: 18 dez. 2023.

[11] Kwon, Yonghye. Yolo-Label. 2018. Disponível em: https://github.com/developer0hye/Yolo_Label. Acesso em: 18 dez. 2023.

[12] Gilbey, Julian D.; Schönlieb, Carola-Bibiane. An end-to-end Optical Character Recognition approach for ultra-low-resolution printed text images. arXiv 2105.04515 versão online, 2021. Disponível em: https://arxiv.org/abs/2105.04515

[13] Team, Opencv. OpenCV. 2011. Disponível em: https://opencv.org/. Acesso em: 18 dez. 2023.

[14] ISO 6346. Freight containers: coding, identification and marking. Coding, identification and marking. 1995. Disponível em: https://cdn.standards.iteh.ai/samples/59778/8692d98b90f649c5921ca3b4ab4a0c34/ISO-6346-1995-Amd-3-2012.pdf. Acesso em: 18 dez. 2023.

[15] MITTAL, Rishabh; GARG, Anchal. Text extraction using OCR: a systematic review. In: Second International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2020. p. 357-362.

[16] MANDAVIA, K. et al. Optical Character Recognition Systems for Different Languages with Soft Computing. Spring. Inter. Publish., v. 352, p. 9-41, 2017. MAÇADA, FELDENS, SANTOS. Impacto da tecnologia da informação na gestão das cadeias de suprimentos – um estudo de casos múltiplos. Gest. Prod., São Carlos, v. 14, n. 1, p. 1-12, jan.-abr. 2007.