http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2023.v15.e405
ISSN on-line 2178-8332

Colloquium Exactarum

Submetido: 19/12/2022 Revisado: 11/07/2023 Aceito: 21/08/2023

ALGORITMO PARA A DETECÇÃO DE ÁRVORES URBANAS A PARTIR DE IMAGENS 360

ALGORITHM FOR THE DETECTION OF URBAN TREES FROM 360 IMAGES

Gustavo Garcia de Campos[1], Francisco Assis da Silva[1], Leandro Luiz de Almeida[1], Almir Olivette Artero[2], Ricardo Luís Barbosa[3], Alan Kazuo Hiraga[4]

[1]Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática de Presidente Prudente – FIPP, Presidente Prudente, SP. [2]Universidade Estadual Paulista - UNESP, Departamento de Matemática e Computação, Presidente Prudente. [3]Universidade Federal de Uberlândia - UFU, Instituto de Geografia, campus de Monte Carmelo, MG. [4]Integral Soluções LTDA, Presidente Prudente, SP.
Email: gugcampos@gmail.com, chico@unoeste.br, llalmeida@unoeste.br, almir.artero@unesp.br, rluisbarbosa@gmail.com, alan@integralsol.com.br

RESUMO – As árvores são indispensáveis à vida humana, absorvem dióxido de carbono e liberam oxigênio, protegem os ecossistemas, reduzem a erosão e ajudam na redução de temperatura do ambiente. A identificação manual de árvores em vias públicas necessita de gastos e tempo para o registro e administração dos dados coletados, visto que as regiões urbanas podem ser muito amplas. Neste trabalho foi desenvolvido um método para a detecção de árvores em regiões urbanas a partir de um vídeo 360. Uma rede neural YOLO foi treinada para detectar as árvores nos quadros do vídeo equiretangular (imagens 360). Técnicas de Visão Computacional com o auxílio da biblioteca OpenCV foram utilizadas no desenvolvimento de algoritmos para segmentar as regiões que enquadram as árvores detectadas no campo de visão retilínea (projeção gnomônica), com o propósito de verificar se as árvores estão nas calçadas. Os resultados obtidos apresentaram em torno de 80% de acerto na detecção de árvores usando a YOLO, e uma precisão de 71% no algoritmo que verifica se as árvores estão na calçada.
Palavras-chave: Detecção de árvores; YOLO; Imagem Equiretangular; Imagem 360; Visão Computacional.

ABSTRACT – Trees are indispensable to human life, they absorb carbon dioxide and release oxygen, protect ecosystems, reduce erosion and help to reduce the environment temperature. The manual identification of trees on public roads requires expense and time for recording and managing the data collected, since urban regions can be very large. We developed in this paper a method for the trees detection in urban areas from a 360 video. A YOLO neural network was trained to detect the trees from frames of the equirectangular video (360 images). We used Computer Vision techniques with the OpenCV library to develop algorithms to segment the regions that fit the detected trees in the rectilinear field of view (gnomonic projection), in order to verify if the trees are on the sidewalks. The results obtained showed around 80% success in detecting trees using YOLO, and an accuracy of 71% in the algorithm that checks if the trees are on the sidewalk.
Keywords: Tree Detection; YOLO; Equirectangular Image; 360 Image; Computer Vision.

1. INTRODUÇÃO

As árvores são responsáveis pela disseminação de uma ecoestrutura sistemática de clima, vegetação e polarização de diversidades ecológicas. Elas são organismos essenciais do planeta, desenvolvem um papel primordial para controle da temperatura, transformam o gás carbônico em oxigênio, controlam as chuvas e promovem a biodiversidade. Além da possibilidade de produzir frutos, fibras, sementes, madeira, resina e inúmeros tantos outros benefícios e funções (IBFLORESTAS, 2021).

Diante disso, é relevante o estudo sobre a possibilidade de levantamento de informações geográficas sobre árvores, para serem utilizada, por exemplo, pela administração municipal. Segundo Barbosa *et al*. (2018), o inventário de árvores urbanas é importante para o conhecimento das espécies existentes e a geolocalização contribui para uma gestão eficiente. O uso de imagens digitais, principalmente imagens panorâmicas e imagens 360, pode auxiliar no levantamento urbano de árvores.

Segundo Gledhill *et al*. (2003) um panorama é uma única imagem grande angular do ambiente ao redor da câmera. Normalmente, o panorama envolve completamente a câmera no plano horizontal e podem ter aproximadamente 120° no campo de visão vertical ou 180° para criar uma esfera completa. Os panoramas podem ser amplamente utilizados em áreas como robótica, visão computacional, vigilância e realidade virtual. Eles também são impulsionados por aplicações comerciais como entretenimento, TV interativa, imóveis e turismo virtual.

A partir da captura de imagens e com o uso de algumas técnicas computacionais é possível fazer a detecção das árvores em uma cena. Essa atividade de detecção de objetos em imagens é uma das principais áreas de aplicação de Processamento de Imagens e Visão Computacional e é usado em muitas aplicações diferentes (TREIBER, 2010).

Para encontrar objetos em imagens, redes neurais artificiais estão sendo muito utilizadas, como é o caso do trabalho de muitos trabalhos encontrados na literatura. No trabalho de Itakura e Hosoi (2020) é proposto um método de medição de árvores utilizando câmeras esféricas 360º com detecção usando uma rede YOLO v2 e uma abordagem fotogramétrica chamada estrutura do movimento (*structure from motion*) para realizar a reconstrução da imagem e medição estrutural. Segundo os autores, as árvores foram detectadas com as informações obtidas das imagens 3D reconstruídas, obtendo como resultado o diâmetro e altura do tronco das árvores. No trabalho de Xie *et al*. (2019) é proposto um *framework* para a detecção de árvores em imagens urbanas, utilizando técnicas de detecção baseadas no modelo de rede neural R-CNN (*Region-based Convolutional Neural Network*). O método proposto aborda o desafio da oclusão e detecções falsas em cenários com muitas árvores. Para isso, os autores introduzem uma nova função de perda de treinamento que suprime detecções falsas, forçando as propostas associadas às árvores verdadeiras a se localizarem compactamente entre si e se afastarem de outras árvores verdadeiras que não são seus alvos. Além disso, o trabalho apresenta um módulo de atenção de partes, para lidar efetivamente com partes de árvores ocultas durante a detecção.

Apesar da importância da aplicabilidade do reconhecimento de árvores e sua identificação em imagens de forma automática, essa atividade é pouco explorada. Este trabalho busca contribuir com um algoritmo para a detecção de árvores urbanas a partir de quadros (*frames*) de vídeo 360 capturados nas vias de uma cidade. A detecção das árvores ocorre em dois estágios, a detecção no vídeo equiretangular (imagens 360) utilizando a rede neural YOLO (*You Only Look Once*) e a verificação se as árvores estão realmente nas calçadas das vias urbanas. É importante saber se cada árvore detectada está na calçada para que se possa, em outro momento, anotar a localização geográfica e construir, por exemplo, um mapa de urbanização da cidade.

Após esta seção introdutória, o trabalho está organizado da seguinte maneira. Na Seção 2 são apresentados os conceitos que fundamentam a base teórica para este trabalho. Na Seção 3 é detalhado todo o processo desenvolvido. Na Seção 4 são apresentados os resultados e, por fim, na Seção 5 encontram-se as considerações finais.

2. CONCEITOS FUNDAMENTAIS

Nesta seção são abordados os conceitos fundamentais para o entendimento da rede neural YOLO utilizada para detecção das árvores, juntamente com esclarecimentos sobre imagem equiretangular e projeção retilínea (gnomônica).

2.1. YOLO

Segundo Shinde, Kothari e Gupta (2018), YOLO (*You Only Look Once*) é uma abordagem avançada de detecção de objetos. A rede YOLO aplica uma única CNN (*Convolutional Neural Network*) à imagem inteira e divide a imagem em grades. A previsão das caixas delimitadoras e a respectiva pontuação de confiança são calculadas para cada grade. Essas caixas delimitadoras são analisadas pela pontuação de confiança prevista (Figura 1).

Figura 1. Ilustração do funcionamento da YOLO mostrando as divisões das grades.

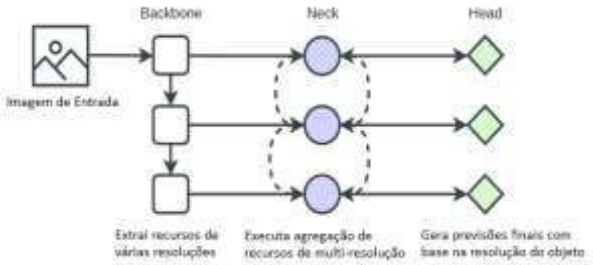
Fonte: Alves (2020).

Diferente dos algoritmos que percorrem várias vezes a mesma imagem, em busca dos objetos treinados, a YOLO apenas precisa percorrer a imagem uma única vez (PIEMONTEZ, 2022).

Os detectores de objetos podem ser divididos em duas categorias: Detectores de Um Estágio (*One-Stage Detector*) e Detectores de Dois Estágios (*Two-Stage Detector*). Os detectores de dois estágios separam a tarefa de localização e classificação de objetos para cada caixa delimitadora. Detectores de um estágio fazem previsões para localização e classificação de objetos ao mesmo tempo. A YOLO é um detector de um estágio (SOLAWETZ, 2020).

Um modelo de detecção contém um *backbone*, *neck* e *head*. O módulo de *backbone* explora os recursos essenciais de diferentes resoluções, e o módulo de *neck* funde os recursos de diferentes resoluções. Finalmente, vários módulos *head* realizam a detecção de objetos em diferentes resoluções. Estrutura demonstrada na Figura 2 (KATEB *et al*., 2021).

Figura 2. Estrutura de um detector de objetos.



Fonte: Adaptado de (KATEB *et al*., 2021).

2.1.1. Backbone

A rede de *backbone* para um detector de objetos normalmente é pré-treinada na classificação ImageNet. O pré-treinamento significa que os pesos da rede já foram adaptados para identificar características relevantes em uma imagem, embora sejam ajustados na nova tarefa de detecção de objetos (SOLAWETZ, 2020).

YOLOv4 usa especificamente CSPDarknet53 como seu *backbone*. Ele opera em uma estratégia CSPNet de dividir o mapa de características que consiste em DenseBlock em duas metades e, em seguida, mesclá-los por meio de uma hierarquia de estágio cruzado. A parte anterior contorna a camada base e é usada como entrada da próxima camada de transição (RAJPUT, 2021).

2.1.2. Neck

Segundo Bouraya e Belangour (2021), o *neck* do detector de objetos refere-se às camadas adicionais existentes entre o *backbone* e a *head*. Seu papel é coletar mapas de características de diferentes

estágios. Os modelos de *neck* são compostos por vários caminhos de cima para baixo e vários caminhos de baixo para cima. A ideia por trás dessa agregação de características existente neste modelo é permitir que as características de baixo nível interajam mais diretamente com as características de alto nível. Eles alcançam agregação e interação de características em muitas camadas, pois a distância entre os dois mapas de recursos é grande.

2.1.3. Head

De acordo com Bouraya e Belangour (2021), *head* é última parte do modelo de detecção de objetos, nela é feita a previsão das caixas delimitadoras e das classes dos objetos. Pela YOLO ser um detector de um estágio, ela possui altas velocidades de inferência, onde o modelo prevê as caixas delimitadoras em uma única etapa.

2.2. Imagem Equiretangular

As projeções de imagens equiretangulares mapeiam as coordenadas de latitude e longitude de um globo esférico diretamente nas coordenadas horizontais e verticais de uma grade, onde essa grade é aproximadamente duas vezes mais larga que alta. As projeções equiretangulares podem mostrar todo o ângulo de visão vertical e horizontal de até 360 graus, como pode ser observado na Figura 3 (MUTHA, 2017).

Figura 3. Imagem equiretangular obtida a partir de um quadro obtido a partir do vídeo 360 utilizado pelos autores.

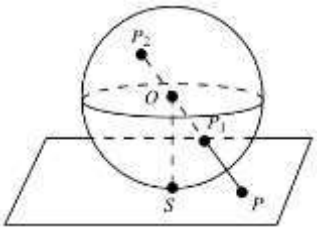
Fonte: Os autores.

2.3. Projeção retilínea (gnomônica)

A projeção retilínea (gnomônica) é obtida através da projeção do ponto P_1 (ou P_2) na superfície de uma esfera no centro *O* da mesma para o ponto *P* em um plano que é tangente a um ponto *S*, como demonstrado na Figura 4 (WEISSTEIN, 2022).

As projeções retilíneas são usadas para extrair um NFoV (*Normal Field of View*) de uma imagem equiretangular.

Figura 4. Projeção esférica usada para a obtenção da projeção retilínea (gnomônica).



Fonte: (WEISSTEIN, 2022).

2.4. Equações de transformação plana

As equações de transformação das coordenadas planas são dadas pela Equação 1 e Equação 2, onde (λ, Φ) = (0, 0).

$$x = \frac{\cos\phi \sin(\lambda - \lambda_0)}{\cos(c)} \qquad (1)$$

$$y = \frac{\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi \cos(\lambda - \lambda_0)}{\cos(c)} \quad (2)$$

onde *c* é a distância angular do ponto (*x, y*) do centro da projeção dada pela Equação 3.

$$\cos(c) = \sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi \cos(\lambda - \lambda_0) \quad (3)$$

enquanto a Equação 4 e a Equação 5 são as equações inversas para mapear o ponto (*x, y*) no plano para a esfera.

$$\phi = \sin^{-1}(\cos c \ \sin\phi_1 + \frac{y \sin c \cos\phi_1}{\rho} \quad (4)$$

$$\lambda = \lambda_0 + \tan^{-1}\left(\frac{x \sin c}{\rho \cos\phi_1 \cos c - y \sin\phi_1 \sin c}\right) \quad (5)$$

onde, $\rho = \sqrt{x^2 + y^2}$ e $c = \tan^{-1}\rho$

Na Figura 5 é ilustrada uma representação do mapeamento da imagem equirretangular na imagem retilínea.

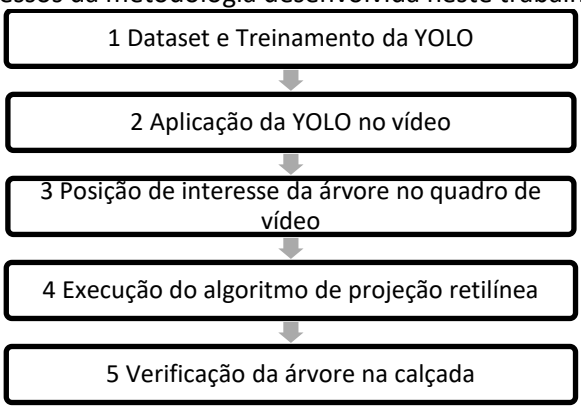
Figura 5. Ilustração gráfica do mapeamento da imagem equiretangular/esférica na imagem retilínea.

Fonte: Adaptado de (MUTHA, 2017).

3. MÉTODO PROPOSTO

O fluxograma apresentado na Figura 6 apresenta a sequência das etapas do método proposto, que se resume basicamente ao treinamento da YOLO, aplicação da rede treinada no vídeo para detectar as árvores presentes na cena, busca pela posição da árvore no quadro de vídeo, realizar a projeção retilínea obtendo a região da árvore e por fim, realizar a verificação se a árvore está presenta na calçada da via urbana.

Figura 6. Fluxograma de processos da metodologia desenvolvida neste trabalho.



Fonte: Os autores.

3.1. Dataset e Treinamento da YOLO

Para a utilização da arquitetura de rede YOLO foi necessário fazer algumas configurações nos valores de parâmetros da rede como: Valor de *max batches* para 6.000, *width* e *height* da imagem para 608.

Todo o conjunto de treinamento, validação da rede YOLO, são de imagens de árvores obtidas através de um *dataset* público chamado Open Images Dataset (2022). Na Figura 7 são mostradas algumas imagens do *dataset*.

Figura 7. Imagens do Open Images Dataset.

Fonte: Open Images Dataset (2022).

Foram utilizadas no total 10.400 imagens, sendo usadas 8.000 imagens para treinamento da rede YOLO e 2.400 para validação.

A rede foi treinada em 6.000 épocas. Todo o processo foi realizado na nuvem do Google, o Google Colaboratory, com o ambiente configurado em GPU. Na Figura 8 é apresentada uma imagem utilizada para treinamento com as regiões contendo árvores demarcadas por retângulos.

Figura 8. Imagem utilizada no treinamento da YOLO.

Fonte: Open Images Dataset, 2022.

Para comprovar o comportamento da rede, foi aplicado o resultado do treinamento da YOLO em algumas imagens que não foram utilizadas durante o treinamento. Esses testes não foram contabilizados nos experimentos realizados. Na imagem da Figura 9, a YOLO obteve uma predição de 77% de que o objeto detectado é uma árvore.

Figura 9. Exemplo de imagem de predição pela YOLO.

Fonte: Os autores.

3.2. Aplicação da YOLO no vídeo

Após a finalização do treinamento da rede, os pesos obtidos foram aplicados nos quadros do vídeo 360 utilizado. O vídeo possui três minutos e cinquenta e seis segundos e possui a filmagem em 360 graus das vias urbanas de uma cidade, com quadros 2.560 x 1.280 pixels de resolução. Na Figura 10 é representada a aplicação dos pesos da YOLO em um dos quadros do vídeo, em que se pode notar sete árvores detectadas na mesma cena.

Figura 10. YOLO aplicada em um quadro do vídeo.

Fonte: Os autores.

3.3. Posição de interesse da árvore no quadro de vídeo

Para cada cena (quadro do vídeo) em que forem detectadas árvores, faz-se necessário saber se a posição de cada árvore está na lateral do carro (direita ou esquerda) no eixo central da câmera. A partir dos eixos da câmera, que está fixada sobre um carro, foram estipuladas retas para que se pudesse determinar de forma empírica coordenadas de regiões de interesse no eixo esquerdo (E) e no eixo direito (D) (Figura 11). Quando o centro da largura de algum *bounding box* que contém uma árvore detectada pela YOLO estiver sob as regiões de interesse na imagem, representadas na Figura 11 por (E) e (D), é considerado que esta árvore foi detectada na imagem equirretangular. A partir daí um algoritmo para obter a projeção retilínea (gnomônica) do lado esquerdo ou direito é executado.

Figura 11. Regiões de interesse estipuladas empiricamente nos eixos da câmera.

Fonte: Os autores

3.4. Execução do algoritmo de projeção retilínea

Para a execução do algoritmo que obtém a imagem com projeção retilínea (gnomônica) (Seções 2.3 e 2.4) é fornecido como parâmetro o lado a qual foi encontrada a árvore no quadro do vídeo. Foi determinado que a imagem retilínea tenha uma resolução de 1.000 x 500 pixels. Na Figura 12 é mostrado um quadro de vídeo contendo uma árvore no lado direito detectado pela YOLO que está no eixo da câmera.

Figura 12. Exemplo de quadro do vídeo contendo uma árvore no lado direito.

Fonte: Os autores.

Na Figura 13 tem-se a imagem gerada após a execução do algoritmo de projeção retilína (gnomônica). Nessa imagem pode-se visualizar a árvore posicionada no centro da cena, que é a árvore que está no eixo da câmera na imagem equiretangular (Figura 12) detectada pela YOLO.

Figura 13. Imagem retilínea contendo uma árvore no eixo da lateral direita da câmera.

Fonte: Os autores.

3.5. Verificação da árvore na calçada

Para verificar se árvore está na calçada, foi desenvolvido um algoritmo aplicando técnicas de processamento de imagem. Inicialmente a imagem foi convertida para do espaço de cor RGB (*Red Green Blue*) para tons de cinza. Em seguida foi aplicado o filtro de suavização gaussiana na imagem em tons de cinza, com kernel de (25, 25) e sigma 0, obtendo uma nova imagem. O resultado pode ser visualizado na Figura 14.

Figura 14. Imagem em tons de cinza com a aplicação da suavização gaussiana.

Fonte: Os autores.

Em seguida é aplicado um algoritmo que realiza a divisão entre a imagem em tons de cinza e a imagem da suavização gaussiana da Figura 14 com a intenção de destacar a silhueta do tronco da árvore (Figura 15). Novamente foi aplicada uma suavização gaussiana, porém, agora com um kernel de (3, 3) (Figura 16).

Figura 15. Imagem resultante da aplicação da divisão entre a imagem em tons de cinza e a imagem da suavização gaussiana (Figura 14).

Fonte: Os autores.

Figura 16. Imagem com a suavização gaussiana aplicada no resultado da Figura 15.

Fonte: Os autores

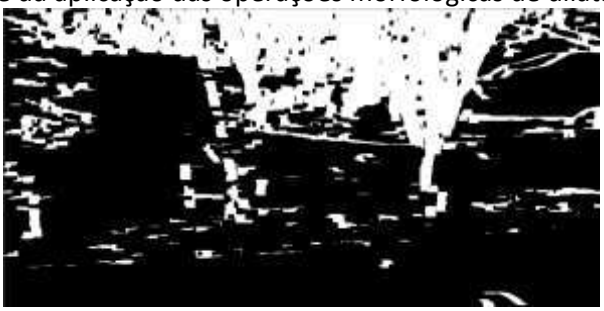
Na sequência, foi aplicado um *threshold* Otsu (OTSU, 1979) com valor limiar de 100. O Resultado do *threshold* pode ser visualizado na Figura 17.

Figura 17. Resultado da aplicação do *threshold* Otsu.

Fonte: Os autores.

A próxima tarefa foi realizar operações morfológicas de dilatação e erosão a partir da imagem da Figura 17 com objetivo de eliminar elementos pequenos e destacar as regiões de interesse, que seria o tronco da árvore. Inicialmente, foi aplicada uma operação de dilatação com elemento estruturante de tamanho (1, 4) com duas iterações, seguida de uma operação de erosão com elemento estruturante de tamanho (8, 4), também com duas iterações. O resultado das operações morfológicas de dilatação e erosão pode ser visualizado na Figura 18.

Figura 18. Imagem resultante da aplicação das operações morfológicas de dilatação e erosão.



Fonte: Os autores.

Sob a imagem resultante (Figura 18) foram aplicadas uma operação de erosão e uma de dilatação com elemento estruturante de tamanho (1, 4) com duas iterações. Na Figura 19 tem-se o resultado das operações de erosão e dilatação seguida do negativo da imagem.

Figura 19. Imagem resultante da aplicação da erosão e dilatação sob a imagem da Figura 18, seguida da operação para inverter a cor (negativo).

Fonte: Os autores.

A partir da Figura 19, foram realizadas a busca dos contornos presentes na imagem usando o algoritmo *Border Following* (SUZUKI; BE, 1985). Na Figura 20 tem-se o resultado contendo todos os contornos.

Figura 20. Contornos encontrados utilizando o algoritmo *Border Following*.

Fonte: Os autores.

Na sequência, apenas as regiões detectadas com altura maior que 40 pixels e largura maior que 200 pixels são mantidas (Figura 21).

Figura 21. Regiões de contornos de interesse resultantes.

Fonte: Os autores.

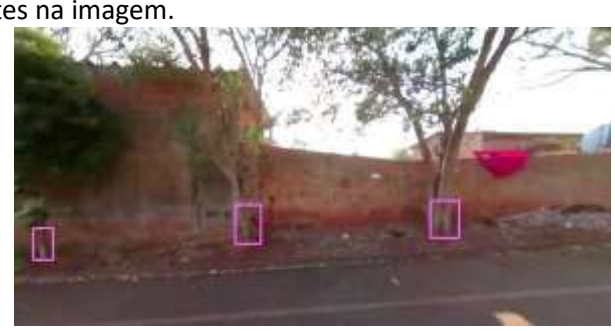
Por fim, como é procurado o tronco da árvore na calçada, foi estabelecido empiricamente, a partir da análise de várias imagens, uma região considerada válida. Essa região da calçada deve estar com a coordenada *y* entre 290 e 405 pixels.

Figura 22. Após eliminação das partes acima e abaixo da região da calçada.

Fonte: Os autores

A partir dos contornos filtrados na etapa anterior, foi calculado um retângulo mínimo envolvendo essas regiões, que possivelmente tem um tronco de árvore. Na Figura 23 é apresentado o resultado final.

Figura 23. Retângulos representando as regiões encontradas pelo algoritmo contendo possivelmente troncos das árvores presentes na imagem.



Fonte: Os autores.

O último passo do algoritmo é saber se a árvore detectada pela YOLO (representada pela Figura 12) está realmente na calçada e no eixo da câmera. Para isso os retângulos mostrados na Figura 23 são comparados com uma posição de coordenada *x* próxima do centro da imagem retilínea. Essa posição foi determinada a partir da análise de várias imagens retilíneas de resultado. No caso do exemplo da Figura 23, a árvore buscada está posicionada sob o retângulo do meio.

4. RESULTADOS

No vídeo 360 utilizado nos experimentos do trabalho, foram contadas 125 árvores na lateral esquerda, sendo 98 árvores nas calçadas, e 140 árvores na lateral direita, sendo 109 nas calçadas. Muitas dessas árvores foram detectadas com a rede YOLO treinada, como pode ser observado na Tabela 1. Foram consideradas válidas as árvores detectadas pela YOLO quando estas estiverem no eixo da câmera. As demais árvores detectadas nos quadros e que não estavam no eixo da câmera foram descartadas.

Outro resultado buscado neste trabalho foi a validação da presença das árvores existentes na calçada, executado pelo algoritmo desenvolvido e apresentado na Seção 3.5. Os resultados são apresentados na Tabela 2. Algumas árvores podem ser detectadas com a YOLO, mas não se encontram na calçada da via, o que tem que ser descartado pelo algoritmo, como pode ser observado na Figura 24.

Tabela 1. Resultados da detecção das árvores utilizando a YOLO.

	Árvores lateral esquerda	**Árvores lateral direita**
Existentes	125	140
Detectadas	103	113
% Acertos	82,4	80,7

Fonte: Os autores.

Tabela 2. Resultados da validação das árvores encontradas na calçada.

	Árvores lateral esquerda calçada	**Árvores lateral direita calçada**
Existentes	98	109
Detectadas	70	88
% Acertos	71,4	80,7

Fonte: Os autores.

Figura 24. Exemplo de árvore detectada no quadro de vídeo (indicada pela seta), mas não se encontra na calçada.

Fonte: Os autores.

Os resultados do método proposto em identificar e confirmar a presença de árvores nas calçadas são considerados significativos, uma vez que a detecção de objetos de forma automática em imagens 360 é uma tarefa desafiadora devido à complexidade das cenas panorâmicas. A estratégia de considerar apenas as árvores detectadas no eixo da câmera como válidas também se mostrou eficaz para evitar falsos positivos e melhorar a acurácia do método.

Em suma, os resultados obtidos e as metodologias empregadas neste estudo representam uma contribuição para a área de visão computacional, especificamente na detecção de objetos em imagens 360.

5. CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto um método para realizar a detecção de árvores urbanas a partir de quadros de vídeo 360. O método foi dividido em duas etapas, a detecção das árvores nos quadros utilizando uma rede YOLO, que foi treinada, e a validação da presença das árvores nas calçadas. Nessa segunda etapa, foi desenvolvido um algoritmo utilizando técnicas de processamento digital de imagens.

A detecção da árvore utilizando os pesos da rede YOLO treinada atingiu valores acima de 80%. Esse valor ainda pode ser melhorado com o aprimoramento da rede utilizada, adicionando novas imagens que retratam árvores em cidades ao *dataset*, e realizar um novo treinamento.

O algoritmo desenvolvido utilizado para validar a presença das árvores nas calçadas também se mostrou promissor, uma vez que a taxa de acerto foi superior a 71%. Um pré-processamento com algumas

técnicas para remoção de sombras nas imagens poderia melhorar os resultados, já que a sombra foi um dos fatores que prejudicaram a detecção do tronco da árvore.

REFERÊNCIAS

ALVES, G.; **Detecção de Objetos com YOLO - Uma abordagem moderna**. 2020. Disponível em: https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna. Acesso em: 17 dez. 2022.

BARBOSA, R. L.; GALLIS, R. B. A.; HIRAGA, A. K.; SILVA, F. A. Quantificação e Georreferenciamento Semiautomático de Árvores Urbanas. **Revista da Sociedade Brasileira de Arborização Urbana - REVSBAU**, Curitiba, v.13, n.4, p.41-53, 2018. https://doi.org/10.5380/revsbau.v13i4.65046

BOURAYA, S.; BELANGOUR, A. Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study. **International Journal of Advanced Computer Science and Applications**. 2021. Acesso em: 14 dez. 2022. https://doi.org/10.14569/IJACSA.2021.0121119

GLEDHILL, D.; TIAN, G. Y.; TAYLOR, D.; CLARKE, D. Panoramic imaging—a review. **Computer & Graphics**, v. 27, n. 3, p. 435-445, 2003. https://doi.org/10.1016/S0097-8493(03)00038-4

IBFLORESTAS; **Quais são as partes da árvore e as suas funções?**. s.d. Disponível em: https://www.ibflorestas.org.br/conteudo/quais-sao-as-partes-da-arvore?utm_source=google-ads&utm_medium=cpc&utm_campaign=nativas-c&keyword=%22a%20importancia%20da%20arvore%27&creative=429664467327&gclid=Cj0KCQjwnoqLBhD4ARIsAL5JedLWmXY4yn5p5yCqHCmlb8VdrncdeSzJeWCaaDI-sgT8VQVOighB6_YaAsDoEALw_wcB. Acesso em: 03 dez. 2022.

ITAKURA, K.; HOSOI, F. Automatic Tree Detection from Three-Dimensional Images Reconstructed from 360° Spherical Camera Using YOLO v2. **Remote Sensing**, 12(6), 2020. https://doi.org/10.3390/rs12060988

KATEB, F. A.; MONOWAR, M. M.; HAMID, A.; OHI, A. Q.; MRIDHA, M. F. FruitDet: Attentive Feature Aggregation for Real-Time Fruit Detection in Orchards. **Agronomy**. 11(12), 2021. https://doi.org/10.3390/agronomy11122440

MUTHA, N. **How to map Equirectangular projection to Rectilinear projection**. 2017. Disponível em: http://blog.nitishmutha.com/equirectangular/360degree/2017/06/12/How-to-project-Equirectangular-image-to-rectilinear-view.html. Acesso em: 11 dez. 2022.

OPEN IMAGES DATASET. Disponível em: https://storage.googleapis.com/openimages/web/index.html. Acesso em: 19 dez. 2022.

OTSU, N. A Threshold Selection Method from Gray-Level Histograms. **IEEE Transactions on Systems, Man, And Cybernetics**, v. 9, n. 1, p. 62-66, 1979.. https://doi.org/10.1109/TSMC.1979.4310076

PIEMONTEZ; **YOLO para Detecção de Objetos – Visão Geral**. 2022. Disponível em: https://visaocomputacional.com.br/yolo-para-deteccao-de-objetos-visao-geral/. Acesso em: 13 dez. 2022.

RAJPUT. **Working of YOLOv4 Algorithm**. 2021. Disponível em: https://medium.com/@shraddhapattanshetti161998/working-of-yolov4-algorithm-76a5e75f188b#:~:text=YOLOv4%20Specifically%20uses%20CSPDarknet53%20as. Acesso em: 09 dez. 2022.

SHINDE, S.; KOTHARI, A.; GUPTA, V. YOLO based Human Action Recognition and Localization. **Procedia Computer Science**, 133, p. 831–838. 2018. https://doi.org/10.1016/j.procs.2018.07.112

SOLAWETZ; **YOLOv4 - An explanation of how it works**. 2020. Disponível em: https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/#yolov4-backbone-network-feature-formation. Acesso em: 10 dez. 2022.

SUZUKI, S.; BE, K. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, 30(1), 32–46, 1985. https://doi.org/10.1016/0734-189X(85)90016-7

TREIBER, M. An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications. **Springer-Verlag London Limited**, 2010.

WEISSTEIN. **Gnomonic Projection**. s.d. Disponível em: https://mathworld.wolfram.com/GnomonicProjection.html. Acesso em: 13 dez. 2022.

XIE, Q.; LI, D.; YU, Z.; ZHOU, J.; WANG, J. Detecting Trees in Street Images via Deep Learning with Attention Module**. IEEE Transactions on Instrumentation and Measurement**, v. 69, n. 8, p. 5395-5406, 2019. https://doi.org/10.1109/TIM.2019.2958580