UNIVERSIDADE DE SÃO PAULO – USP
ESCOLA DE ENGENHARIA DE SÃO CARLOS – EESC
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

ALCEU DE PAIVA BRETTAS NETO

DETECÇÃO E SEGMENTAÇÃO DE INSTRUMENTOS CIRÚRGICOS EM BANDEJAS POR REDES NEURAIS CONVOLUCIONAIS

SÃO CARLOS
2025

ALCEU DE PAIVA BRETTAS NETO

DETECÇÃO E SEGMENTAÇÃO DE INSTRUMENTOS CIRÚRGICOS EM BANDEJAS POR REDES NEURAIS CONVOLUCIONAIS

Monografia apresentada ao curso de Engenharia Elétrica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do Título de Engenheira Eletricista.

Orientador: Prof. Dr. Marcelo Becker

SÃO CARLOS

2025

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Brettas Neto, Alceu
B845d Detecção e segmentação de instrumentos cirúrgicos em bandejas por redes neurais convolucionais / Alceu Brettas Neto; orientador Marcelo Becker. São Carlos, 2025.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. visão computacional. 2. detecção de objetos. 3. instrumentos cirúrgicos. 4. oclusão de objetos. 5. YOLO. 6. Segment Anything. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Alceu de Paiva Brettas Neto

Título: “Detecção e segmentação de instrumentos cirúrgicos em bandejas por redes neurais convolucionais”

Trabalho de Conclusão de Curso defendido e aprovado em 25/06/2025,

com NOTA 9,5 (Nove e Meio), pela Comissão Julgadora:

Prof. Associado Marcelo Becker - Orientador SEM/EESC/USP

Prof. Associado Valdir Grassi Junior - SEL/EESC/USP

Mestre Thiago Henrique Segreto Silva - EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

Este trabalho é dedicado ao meu pai e às dezenas de viagens para São Carlos que fizemos juntos.

AGRADECIMENTOS

Gostaria de, primeiramente, agradecer à minha família, por todo o apoio e conforto durante todos os anos da graduação, e por terem me proporcionado a vida que tenho.

Agradeço à minha mãe, Marivane, pelos conselhos, pelo cuidado constante e por me mostrar que o mais importante é estar bem com aqueles que amamos. Ao meu pai, Alceu, por todo o suporte, pela confiança, por ser meu exemplo e espelho, e por me ajudar a enxergar quem eu sou de verdade. À minha irmã, Isabela, por ser minha melhor amiga, minha inspiração, e por me mostrar que é possível sonhar alto quando se dedica a alcançar o sol.

Aos meus cachorros, Bono e Alecrim, por serem meu porto seguro em dias difíceis e por me lembrarem da beleza do mundo nas pequenas coisas.

À minha namorada, Ana Clara Garcia Braghini, por ser o brilho dos meus dias e a estrela que encantou o meu passado, ilumina o meu presente e guia o meu futuro.

À minha avó, Neuza, pelo apoio e carinho, mesmo à distância.

Agradeço ao meu orientador, professor Marcelo Becker, por me mostrar o poder da ciência e da engenharia na transformação do mundo.

Ao LabRoM e a todos os professores e pesquisadores que tornaram este trabalho possível. Em especial, agradeço a Thiago Segreto e João Pinheiro pelo imenso suporte e pelos ensinamentos nas áreas de visão computacional e inteligência artificial.

A todos os servidores e docentes da Universidade de São Paulo — especialmente do Departamento de Engenharia Elétrica e de Computação — que foram fundamentais na minha formação.

Aos grandes amigos que São Carlos me deu e que quero levar para a vida toda. Em especial, Ana Luísa Maia, Benito Palma, Tsuyoshi Sonobe, Rodrigo Kawakami, Luiz Gustavo Tacin, Paulo Chagas e Felipe Tommaselli. Esta trajetória não teria sido possível — e certamente não teria sido tão boa — sem vocês.

Ao Grupo SEMEAR, por ser meu ambiente favorito em São Carlos, por me mostrar quem quero ser no futuro profissional e por me permitir conhecer e me apaixonar pela robótica.

E, por fim, à Visagio e aos colegas de trabalho, por me proporcionarem o início da minha carreira e por contribuírem com a construção dos meus próximos passos.

RESUMO

Este trabalho apresenta uma solução para identificar e organizar instrumentos cirúrgicos que estão desordenados e sobrepostos, como frequentemente ocorre em bandejas durante procedimentos médicos. Para isso, foi utilizado um conjunto de imagens públicas, que passou por uma nova anotação com a ajuda de uma ferramenta moderna chamada Segment Anything Model (SAM). Três abordagens foram comparadas: uma versão tradicional da rede YOLO (YOLOv3), e duas versões mais recentes (YOLOv11), uma com detecção por caixas e outra com segmentação por máscaras. A versão com YOLOv11 e detecção por caixas teve o melhor desempenho geral, enquanto a versão com segmentação se destacou por permitir entender qual instrumento está por cima em casos de sobreposição, algo importante para a automação com robôs. O estudo mostrou que, com redes modernas e dados bem anotados, é possível avançar no desenvolvimento de sistemas que ajudem profissionais da saúde e futuramente permitam o uso de robôs na organização de instrumentos cirúrgicos.

Palavras-chave: visão computacional; YOLO; Segment Anything; detecção de objetos; oclusão de objetos; instrumentos cirúrgicos.

ABSTRACT

This work presents a solution to identify and organize surgical instruments that are often disordered and overlapping in trays during medical procedures. To achieve this, a public image dataset was re-annotated using a modern tool called the Segment Anything Model (SAM). Three approaches were compared: a traditional version of the YOLO network (YOLOv3), and two more recent ones (YOLOv11), one using bounding box detection and the other using segmentation masks. The YOLOv11 with bounding boxes showed the best overall performance, while the segmented version stood out for enabling the identification of which instrument is on top in overlapping scenarios—an important feature for robotic automation. The study shows that with modern neural networks and well-annotated data, it is possible to advance the development of systems that support healthcare professionals and, in the future, allow robots to assist in organizing surgical tools.

Keywords: computer vision; YOLO; Segment Anything; object detection; object occlusion; surgical instruments.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Durante procedimentos cirúrgicos, a atuação do instrumentador cirúrgico é essencial para garantir a segurança e a eficiência da operação. Este profissional é responsável por preparar o ambiente operatório, organizar e fornecer os instrumentos ao cirurgião no momento adequado, além de manter a assepsia e a ordem na sala de cirurgia. Sua presença contribui para a redução do tempo cirúrgico e diminui as chances de contaminação, sendo considerado o braço direito do cirurgião durante o procedimento.

A ausência ou falhas na atuação desses profissionais podem acarretar sérias consequências, como atrasos na cirurgia, aumento do risco de infecções e até comprometimento da segurança do paciente. Em situações emergenciais ou em locais com escassez de profissionais qualificados, a falta desse suporte pode impactar negativamente o desfecho cirúrgico. Portanto, a presença de um instrumentador capacitado é fundamental para o sucesso das intervenções cirúrgicas.

Durante a realização de procedimentos cirúrgicos, manter a organização da mesa de instrumentos representa um desafio constante. Com a dinâmica intensa da operação, instrumentos são utilizados, manipulados e reposicionados com frequência, o que leva a um cenário de sobreposição, acúmulo e eventual desorganização da mesa de instrumentos. Essa situação dificulta a visualização clara dos itens disponíveis, podendo gerar atrasos na entrega do instrumento solicitado pelo cirurgião e aumentando o risco de erros humanos ou contaminações. O problema se agrava em cirurgias longas ou com número elevado de instrumentos, em que o controle visual se torna mais complexo. A Figura 1, por exemplo, ilustra um cenário real da mesa de instrumentos cirúrgicos inicialmente bem organizada e que, no decorrer do procedimento, fica desorganizada com múltiplos instrumentos sobrepostos e dificultando sua identificação automática ou manual.

Nos últimos anos, a implementação de sistemas robóticos na área da saúde tem crescido exponencialmente em diversas aplicações, desde o suporte a cirurgias minimamente invasivas - como demonstram os robôs Da Vinci e Zeus - até o desenvolvimento de assistentes robóticos para apoio a cirurgiões e equipes de enfermagem. Nesse contexto, torna-se promissor explorar como soluções de visão computacional podem contribuir com sistemas robóticos ou semiautônomos para apoio às equipes médicas.

Diante desse cenário, surgiu o projeto Instrubot, desenvolvido no Laboratório de Robótica Móvel (LabROM) da Escola de Engenharia de São Carlos (EESC-USP), com o objetivo de investigar soluções automatizadas para a organização e manipulação de instrumentos cirúrgicos. A partir de um diagnóstico realizado em parceria com profissionais do Hospital das Clínicas de Ribeirão Preto, identificou-se que muitos dos instrumentos mais utilizados — como bisturis, pinças e tesouras — apresentam alta rotatividade na mesa de instrumentos, gerando situações

Figura 1 – Mesa de Instrumentos durante um procedimento cirúrgico

(a) Mesa organizada antes do procedimento

(b) Mesa desorganizada no decorrer do procedimento

Fonte: Elaborado pelo autor. Imagens obtidas junto ao Hospital das Clínicas de Ribeirão Preto (2024).

frequentes de sobreposição e desordem. Este trabalho, derivado do projeto Instrubot, concentra-se especificamente na tarefa de reconhecimento e detecção desses instrumentos em imagens reais de bandejas desorganizadas, avaliando diferentes abordagens de visão computacional, desde modelos baseados em detecção por caixas até técnicas mais recentes de segmentação orientada por máscaras.

2 OBJETIVOS

O presente Trabalho de Conclusão de Curso (TCC) tem como ponto de partida a dissertação de Lavado (2018), que demonstrou a viabilidade de identificar e ordenar instrumentos cirúrgicos em cenário de bin-picking por meio da arquitetura YOLOv3 combinada a redes auxiliares especializadas em inferência de oclusão. Com os avanços recentes em segmentação automática — como o Segment Anything Model (SAM) — e a consolidação de modelos de detecção mais robustos, como a família YOLOv11, este trabalho investiga se tais tecnologias permitem simplificar o fluxo de anotação, eliminar redes auxiliares, e, ao mesmo tempo, elevar o desempenho da detecção e do raciocínio espacial em bandejas cirúrgicas desorganizadas.

2.1 OBJETIVO GERAL

Avaliar o impacto do uso de máscaras segmentadas (geradas com auxílio do SAM e utilizadas para treinar modelos de segmentação YOLOv11) na tarefa de detecção e inferência de oclusão de instrumentos cirúrgicos, comparando três abordagens: (i) a linha de base com YOLOv3 e *bounding boxes*; (ii) um modelo YOLOv11 treinado apenas com caixas; e (iii) um modelo YOLOv11 treinado com máscaras segmentadas.

2.2 OBJETIVOS ESPECÍFICOS

1. Reanotar e estruturar um conjunto de dados compatível com segmentação:
    - Utilizar o CVAT, com apoio do modelo SAM, para gerar máscaras segmentadas para cada instrumento cirúrgico presente nas imagens.
    - Exportar os dados em formatos compatíveis com treinamento supervisionado em tarefas de detecção e segmentação (COCO + YOLO).
2. Treinar três pipelines contrastantes
    - YOLO v3 (Baseline/Lavado) modelo treinado com *bounding boxes*
    - YOLO v11 (caixas) modelo treinado apenas com *bounding boxes* (formato YOLO).
    - YOLO v11 (segmentação) modelo treinado com as máscaras segmentadas exportadas do CVAT.
3. Implementação de Algoritmo de inferência de oclusão:

- Aplicar o modelo YOLOv11-seg em imagens do conjunto de teste para obter máscaras segmentadas por classe.
- Analisar cada máscara quanto ao número de blobs e proporção da área visível.
- Classificar os objetos como TOP (não oclusos) ou BOTTOM (ocluídos) com base nessa análise.

4. Avaliar e comparar os modelos em critérios padronizados:
    - $mAP@0,50$, $mAP@[0,50:0,95]$, Precisão, Revocação e FPS de inferência
    - Acurácia do algoritmo de inferência de oclusão com base em validação cruzada manual.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos que embasam o desenvolvimento deste trabalho, com foco em visão computacional, redes neurais profundas e segmentação de imagens. São introduzidas as definições centrais da área, as técnicas tradicionais e modernas de detecção de objetos, além de uma revisão do estado da arte relacionado à identificação de instrumentos cirúrgicos. Essa fundamentação busca contextualizar a escolha das ferramentas e métodos empregados nos experimentos posteriores.

3.1 VISÃO COMPUTACIONAL

3.1.1 Conceitos Básicos

É vivida uma era em que a capacidade de cálculo cresce numa cadência exponencial, impulsionada por GPUs cada vez mais densas, *clusters* em nuvem acessíveis e grandes volumes de dados gerados por sensores e dispositivos móveis. Esse salto de hardware e dados sustenta avanços notáveis: modelos de deep learning já superam humanos em tarefas específicas, como a interpretação de mamografias (Park, 2020) ou a análise de imagens médicas em geral (Esteva, 2021).

Nesse contexto, visão computacional emerge como a disciplina que permite aos computadores extrair informações significativas de imagens, vídeos e outros conteúdos visuais – e agir com base nesses entendimentos (IBM Corporation, 2021). Enquanto a visão biológica converte fótons captados pela retina em impulsos elétricos que o córtex visual interpreta em frações de segundo, a visão computacional converte pixels em vetores numéricos, aplica algoritmos de detecção, classificação ou segmentação e devolve rótulos, coordenadas ou ações. A diferença reside tanto no meio (câmeras × olhos) quanto na "cognição" (regras matemáticas em GPUs × neurônios biológicos), mas o objetivo final é análogo: compreender o mundo visual para tomar decisões.

Para isso, considera-se o conceito muito importante de "*features*" que são descritores numéricos extraídos de uma imagem que condensam informação visual relevante (textura, cor, gradiente ou geometria). Esses vetores alimentam classificadores ou, nas CNNs modernas, são aprendidos automaticamente pelas primeiras camadas convolucionais, mas a função é a mesma: fornecer uma representação compacta, discriminativa e comparável entre imagens para obter informação e dados relevantes.

Outro conceito importante é o de "*pose*", o qual, por sua vez, descreve a posição e orientação de um objeto no espaço da câmera; em 2-D costuma vir como retângulo delimitador

(x, y, w, h) que circunscreve a projeção do objeto no plano da imagem e é conhecido como "*bounding box*". Estimar a pose correta é crucial quando o algoritmo precisa guiar um robô a realizar uma atividade.

Além das coordenadas da *bounding box*, detectores modernos como de YOLO devolvem um *score de confiança* $s \in [0, 1]$ (ou somente confiança) que quantifica, em termos probabilísticos, quão certo o modelo está de que *(i)* o retângulo contém efetivamente um objeto e *(ii)* esse objeto corresponde ao conjunto inferido (classe) (Redmon et al., 2016).

A partir do conceito de *bounding box*, o modelo de visão computacional - para detecção de objetos - infere um retângulo na imagem com sua interpretação de onde (pose) está o objeto desejado, mas, para avaliar quão precisa foi essa inferência, utiliza-se o indicador *IoU* (*Intersection over Union*) que mede a qualidade de dois retângulos, o proposto pelo modelo e o real da posição do objeto, através da razão entre a área de intersecção e a área de união desses retângulos. Em geral valores de *IoU* iguais ou superiores a $0,50$ são tradicionalmente aceitos como acertos (Everingham et al., 2010).

Dado a atribuição se o retângulo proposto pelo modelo está ou não correto, há 3 possibilidades de parâmetros para todas as imagens lidas, são eles (Sharma, 2022):

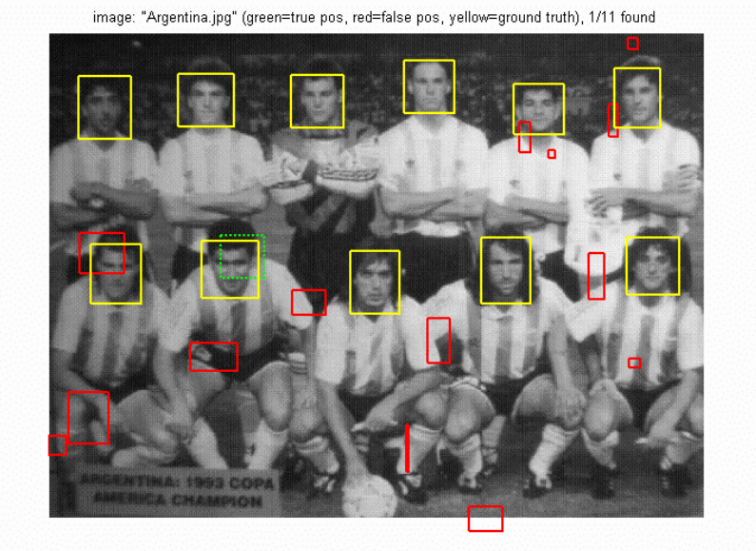
- ***True Positives* (TP)**: Número de vezes que o modelo corretamente atribuiu um *bounding box* em uma região da imagem com o objeto desejado
- ***False Positives* (FP)**: Número de vezes que o modelo incorretamente atribuiu um *bounding box* em uma uma região da imagem sem o objeto desejado
- ***False Negatives* (FN)**: Número de vezes que o modelo incorretamente deixou de atribuir um *bounding box* em uma uma região da imagem com o objeto desejado

A Figura 2 demonstra um exemplo de atribuições de um modelo de visão computacional e seus parâmetros de assertividade. Nele verifica-se em verde o número de rostos corretamente identificados, ou seja, número de *True Positives*, no caso $\text{TP} = 1$, em vermelho verifica-se o número de atribuições erradas do modelo, ou seja, número de *False Positives*, no caso $\text{FP} = 14$, e em amarelo o número de rostos que deveriam ser identificados, com isso obtem-se o número de *False Negatives*, no caso $\text{FN} = \text{Ground\_truth} - \text{TP} = 11 - 1 = 10$.

Com esses três parâmetros definidos, verifica-se duas métrica muito importantes para visão computacional: **Precisão (ou *Precision*)** e **Revocação (ou *Recall*)**. Precisão mede, dentre todas as predições positivas (que ele atribuiu) do algoritmo, qual fração corresponde de fato a objetos/partes verdadeiros, ou seja:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Figura 2 – Exemplo de modelo de detecção em imagem que tinha como objetivo detecção dos rostos dos jogadores



Fonte: Sharma (2022)

Já a Revocação mede dentre todos os objetos/partes reais existentes na imagem, qual fração foi efetivamente detectada pelo algoritmo.

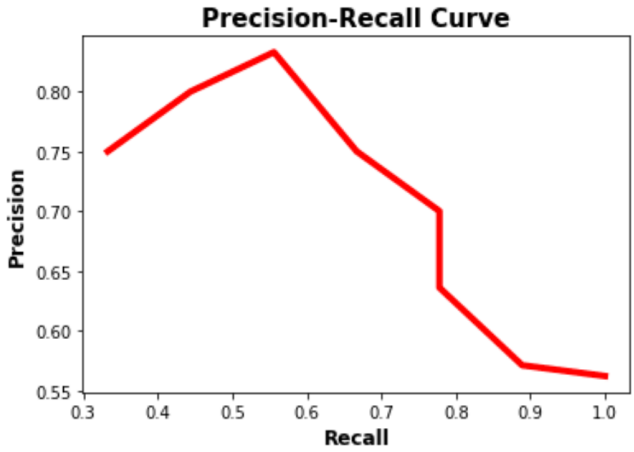
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

A partir disso, é possível traçar para todo algoritmo ou modelo de visão computacional, a curva P-R (*Precision - Recall*) e com ela verifica-se o indicador *AP* (***Average Precision***) que é dada pela área sob essa curva para uma classe (conjunto de mesmos objeto). Considerando todas as classes da base de dados, é possível calcular a média de APs, ou seja, a ***mean Average Precision*** (*mAP*) que é um dos indicadores mais importantes de qualidade de um modelo de visão computacional. A Figura 3 demonstra um exemplo de curva P-R e a seguir apresenta-se a equação de obtenção do *mAP*:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

Por fim, pode-se correlacionar os dois principais parâmetros do modelo, *mAP* (*mean Average Precision*) e *IoU* (*Intersection over Union*), fixando o $IoU = 0,50$, obtendo assim o indicador $mAP@0.5$ comumente usado (Everingham et al., 2010). Além disso, é comum utilizar-se o indicador $mAP@[0.5 : 0.95]$ com a média das *APs* para limiares *IoU* indo de $0,50$ a $0,95$ com passo de $0,05$ (Lin; Maire; Belongie, 2014).

Figura 3 – Exemplo de Curva P-R (*Precision - Recall*)

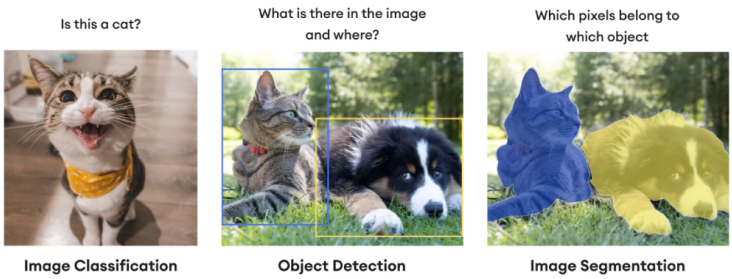


Fonte: Sharma (2022)

3.1.2 Classificação, Detecção e Segmentação

Tendo estabelecido os conceitos iniciais de imagem, feature, pose e as métricas de avaliação, convém notar que a prática contemporânea da visão computacional se organiza em três grandes vertentes. Cada vertente responde a uma pergunta relacionada ao que se quer obter da imagem. A seguir, descreve-se essas três áreas - classificação, detecção e segmentação - destacando seus objetivos, saídas típicas e métricas canônicas. Além disso, a Figura 4 apresenta um exemplo de resposta para cada uma das três vertentes de visão computacional.

Figura 4 – Principais vertentes da Visão Computacional: (a) *Image Classification* (b) *Object Detection* (c) *Image Segmentation*



Fonte: SuperAnnotate (2022)

Classificação de imagens responde apenas “o que há na cena?”. O modelo recebe uma matriz H×W×C (Altura em pixels H, Largura em pixels W e Canais de cores C) e devolve um rótulo único (ou vetor de probabilidades) que identifica a classe dominante identificada. Não há noção explícita de localização; múltiplos objetos competem e o mais saliente prevalece. A virada

histórica ocorreu com AlexNet, que alavancou CNNs profundas e GPUs para superar métodos de features manuais no desafio ImageNet (Krizhevsky; Sutskever; Hinton, 2012). As métricas concentram-se em acurácia top-1/top-5 (fração de previsões corretas sobre o total de amostras).

Detecção de objetos acrescenta a dimensão espacial, respondendo "o que e onde?". A saída é uma lista de tuplas $\langle$classe, *bounding box* e confiança). A avaliação combina precisão e revocação via área sob a curva (AP); realiza a media sobre todas as classes para obter *mAP*, normalmente em $IoU = 0,50$ (Everingham et al., 2010) ou na faixa $0,50 - 0,95$ (Lin; Maire; Belongie, 2014).

Segmentação de imagens exige granularidade por pixel, respondendo "o que, onde e com quais contornos?". Na segmentação semântica cada pixel recebe um rótulo de classe. Na segmentação por instâncias é preciso distinguir objetos individuais da mesma classe. A métrica predominante é IoU pixel-a-pixel, reportada como $mIoU$ ou AP de instância (SuperAnnotate, 2022).

Em síntese: classificação categoriza a cena como um todo; detecção localiza e rotula múltiplos objetos; segmentação refina a fronteira de cada objeto ou região, servindo de base para aplicações que exigem manipulação precisa — como o *bin-picking* cirúrgico que será apresentado nos capítulos seguintes.

3.1.3 Desafios Específicos do Trabalho

Um dos principais desafios do presente trabalho está relacionado ao problema de *bin-picking*, que consiste em identificar e remover corretamente objetos dispostos de forma desorganizada em um espaço compartilhado — como uma bandeja ou recipiente. No contexto cirúrgico, esse desafio é ainda mais sensível, pois os instrumentos metálicos possuem formatos finos e semelhantes, frequentemente dispostos de forma sobreposta e sem padrão fixo. Segundo Lavado (2018), o *bin-picking* aplicado a instrumentos cirúrgicos exige não apenas a correta identificação dos objetos, mas também a determinação da ordem de retirada, uma vez que muitos deles encontram-se parcialmente cobertos por outros e não podem ser removidos de forma arbitrária sem interferência física. Assim, a visão computacional precisa não só detectar os instrumentos presentes, mas também raciocinar sobre a sua disposição espacial para permitir uma manipulação segura e eficiente (Lavado, 2018).

Outro desafio diretamente relacionado é o da oclusão, que ocorre quando parte de um objeto está escondida por outro na imagem. Em tarefas de detecção ou segmentação, a presença de oclusões severas pode comprometer a acurácia do modelo, pois reduz a quantidade de informação visual disponível para cada objeto. A literatura distingue diferentes graus e tipos de oclusão — parciais, totais, intra-classe ou inter-classe —, sendo este um dos principais fatores limitantes

em aplicações reais de visão computacional (Drost et al., 2010). Para resolver esse problema, alguns trabalhos propõem o uso de múltiplas câmeras ou dados 3D; outros, como Lavado (2018), propõem redes dedicadas para inferir, a partir de imagens 2D, quais objetos estão por cima ou por baixo. Neste trabalho propõe-se a viabilidade de substituir essas redes auxiliares por uma análise geométrica baseada em máscaras geradas por modelos de segmentação de propósito geral.

3.2 *DEEP LEARNING*

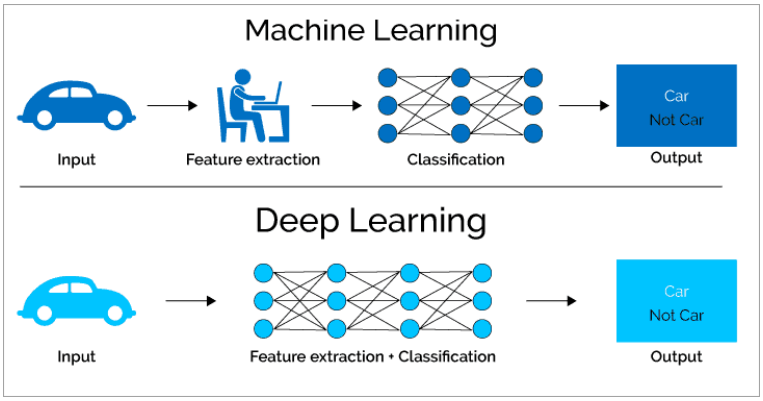
Nas últimas duas décadas o *deep learning* (DL) deixou de ser um tema de pesquisa restrito para se tornar o paradigma dominante em visão computacional, processamento de linguagem natural e robótica (LeCun; Bengio; Hinton, 2015). Redes profundas superaram os métodos clássicos de *machine learning* ao dispensar a engenharia manual de atributos e aprender, diretamente dos dados, representações hierárquicas capazes de capturar desde padrões de borda até composições semânticas complexas. Na prática, esse avanço viabilizou saltos de desempenho em tarefas de detecção de objetos, segmentação semântica e rastreamento — componentes essenciais para a automação de ambientes clínicos, onde a identificação correta de instrumentos e o entendimento de oclusões são pré-requisitos para a manipulação robótica segura de bandejas cirúrgicas (Lavado, 2018).

3.2.1 Aprendizado de Máquina Clássico

Antes da popularização das redes profundas, a maior parte dos sistemas de visão computacional recorria a um *pipeline* de duas etapas: primeiro extraíam-se atributos manuais, como *Histogram of Oriented Gradients* (HOG) (Dalal; Triggs, 2005) ou *Scale-Invariant Feature Transform* (SIFT) (Lowe, 2004); em seguida esses vetores eram alimentados em classificadores supervisionados — tipicamente *Support Vector Machines* (SVM) (Cortes; Vapnik, 1995) ou *Random Forests* (Breiman, 2001). Esse fluxo está ilustrado na Figura 5 junto do comparativo do pipeline de *Deep Learning*.

Embora robusto em bases restritas, o método clássico de aprendizado de máquina dependia fortemente da escolha do descritor e perdia desempenho diante de grandes variações de escala, iluminação ou oclusão - limitações críticas no contexto cirúrgico tratado neste trabalho. Tais gargalos motivaram o ressurgimento das redes neurais profundas, capazes de aprender representações diretamente dos dados brutos e dispensar engenharia manual de atributos.

Figura 5 – Diferença entre *Machine Learning* e *Deep Learning*



Fonte: Wolfewicz (2020)

3.2.2 Redes Neurais Artificiais (ANNs)

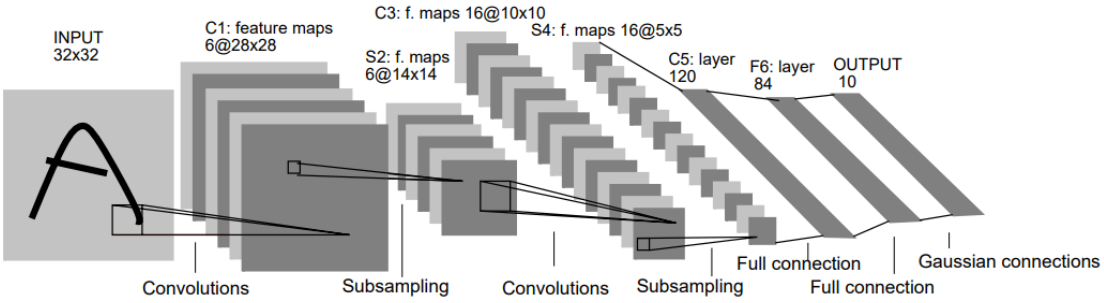
O primeiro modelo de rede neural plausível foi o *Perceptron* de Rosenblatt (1958), capaz de aprender classificações lineares por meio de ajuste iterativo de pesos. Embora marcante, o Perceptron não resolvia problemas não-lineares, o que levou a um hiato da tecnologia até o redescobrimento da retro-propagação (*back-propagation*) por Rumelhart, Hinton e Williams (1986). O algoritmo de retro-propagação habilitou o treinamento de *Multi-Layer Perceptrons* (MLPs), redes totalmente conectadas capazes de aproximar funções arbitrárias. Contudo, MLPs profundos enfrentavam o problema do *vanishing gradient*, aprendendo lentamente e exigindo grande poder computacional, o que limitou sua aplicação em visão por muitos anos (Schmidhuber, 2015). Esses desafios motivaram arquiteturas especializadas - em particular as *Convolutional Neural Networks* (CNNs) - que exploram os padrões em imagens e reduzindo o número de parâmetros necessários, superando as limitações dos MLPs em tarefas de reconhecimento visual.

3.2.3 Redes Convolucionais (CNNs)

Ao contrário dos MLPs totalmente conectados, as *Convolutional Neural Networks* exploram a estrutura espacial das imagens por meio de conexões locais da imagem, pesos compartilhados e operações de *pooling*, reduzindo drasticamente o número de parâmetros e conferindo invariância a translações (Nielsen, 2015). O modelo pioneiro foi o *LeNet-5* de LeCun et al. (1998), que já combinava camadas convolucionais e de *subsampling* para classificar dígitos manuscritos de cheques bancários usando a base MNIST — uma coleção de 60.000 imagens rotuladas de dígitos de 0 a 9. A intuição por trás do funcionamento dessas redes está demonstrado na Figura 6 e pode ser explicada de forma acessível, como no texto de Nakamoto (2021), que compara o processo a uma sequência de “peneiras” cada vez mais refinadas que vão identificando

traços simples até formar uma compreensão mais global da imagem.

Figura 6 – Arquitetura LeNet-5



Fonte: LeCun et al. (1998)

O salto decisivo veio com o *AlexNet* (Krizhevsky; Sutskever; Hinton, 2012): redes mais profundas, ativação ReLU, *dropout* e treinamento acelerado em GPU, alcançando vitória histórica no ImageNet, o que catalisou o "renascimento" do DL em visão.

Subsequentemente, o *VGGNet* (Simonyan; Zisserman, 2014) mostrou que pilhas de convoluções 3×3 simples, porém profundas (16–19 camadas), podiam obter ganhos sistemáticos, ao custo de maior carga computacional. O problema de degradação em redes muito extensas foi atenuado com as conexões de atalho (*skip connections*) do *ResNet* (He et al., 2016), permitindo treinar modelos com mais de 100 camadas sem perda de gradiente. Essas arquiteturas tornaram-se *backbones* padrão para detectores de objetos; a família YOLO, detalhada em seguida, que herda essa evolução e adiciona cabeças de regressão que transformam a detecção em um problema de predição direta, viabiliza aplicações em tempo real como a identificação de instrumentos cirúrgicos em bandejas desorganizadas.

3.2.4 YOLO: You Only Look Once

A arquitetura YOLO (You Only Look Once) revolucionou a tarefa de detecção de objetos ao propor uma abordagem unificada, rápida e precisa. Enquanto modelos anteriores como o R-CNN executavam a detecção em múltiplas etapas - geração de propostas, extração de features e classificação -, o YOLO reformulou o problema como uma única tarefa de regressão executada em tempo real, com uma única passada pela imagem. A rede divide a imagem em uma grade e, para cada célula, prevê simultaneamente as coordenadas das caixas delimitadoras (*bounding boxes*), a classe do objeto e uma medida de confiança, tornando possível realizar detecção com alta velocidade e boa acurácia em aplicações práticas (Redmon et al., 2016).

Teve seu lançamento em 2015, com trabalho realizado por Redmon et al. (2016) num contexto acadêmico e evoluiu por sucessivas melhorias ano a ano. A partir de 2020, com o lançamento da YOLOv5, o projeto passou a ser mantido pela empresa Ultralytics, que reescreveu o código em *PyTorch* e desenvolveu um ecossistema integrado com suporte a detecção, segmentação, rastreamento e estimativa de pose. Essa mudança democratizou o acesso à arquitetura, com uma interface simplificada e adaptável para diversos contextos. A sequência de versões mantidas pela Ultralytics - YOLOv5, YOLOv8 e mais recentemente a YOLOv11 - trouxe avanços contínuos em velocidade de inferência, estabilidade e desempenho em múltiplas tarefas (Ultralytics, 2024).

Uma melhoria importante de se realçar está relacionada ao backbone (rede de extração de características), em que todas as versões até a YOLOv4 utilizam o Darknet-53 como backbone e, a partir da família mantida pela Ultralytics, o código migra para PyTorch, mas o backbone permanece o mesmo até a adoção, nas versões mais recentes (YOLOv8/YOLOv11), de variantes *anchor-free* que dispensam as caixas de referência fixas "*anchor boxes*"para prever as *bounding boxes* finais, desse modo a rede prevê diretamente pontos-chave, vértices ou deslocamentos, simplificando o modelo e reduzindo os hiperparâmetros necessários (Ultralytics, 2023).

Neste trabalho, utiliza-se a versão YOLOv11 como base para os experimentos de detecção de instrumentos cirúrgicos em cenários de bin-picking e avaliação de oclusão.

3.2.5 Transformers em Visão

Os detectores da família YOLO mostram como as CNNs podem alcançar detecção em tempo real. Contudo, convoluções são inerentemente locais: cada neurônio “enxerga” apenas um pequeno campo. Para capturar relações de longo alcance surge o mecanismo de *self-attention* dos Transformers.

O *Vision Transformer* (ViT) (Dosovitskiy et al., 2020) subdivide a imagem em *patches* 16×16 e trata cada patch como um token, permitindo que dependências espaciais distantes sejam aprendidas sem esforço adicional de design. Embora o ViT exija pré-treino massivo, ele inaugurou uma linha que compete diretamente com CNNs em classificação.

O DETR (Carion et al., 2020) aplicou *self-attention* ao problema de detecção, eliminando âncoras e formulando o processo como correspondência bipartida entre *queries* e objetos. Para segmentação, o *Mask2Former* (Cheng; Schwing; Kirillov, 2022) unificou *semantic*, *instance* e *panoptic* segmentation.

O *Segment Anything Model* (SAM) - detalhado em seguida - combina um backbone ViT de larga escala a um decodificador *promptable*. Na integração do SAM à geração de *ground-truth*, obtêm-se máscaras precisas que potencializam o raciocínio de oclusão ausente nos detectores

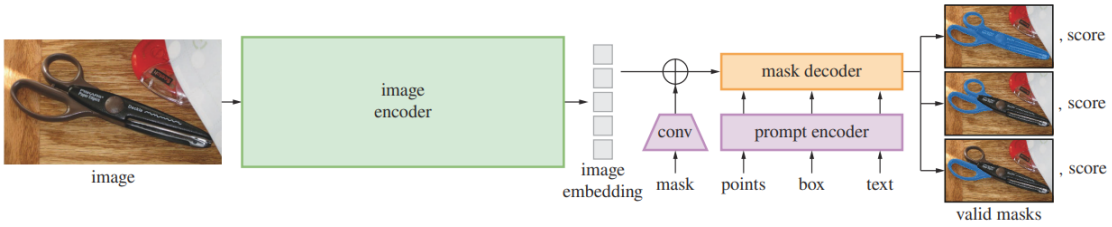
YOLO baseados apenas em caixas retangulares. Desse modo, Transformers complementam as CNNs: YOLO provê detecção rápida; SAM, contorno e hierarquia espacial - combinação crucial para que um robô decida qual instrumento retirar primeiro de uma pilha, por exemplo.

3.2.6 *Segment Anything*: Segmentação via Prompt e Zero-Shot

O modelo *Segment Anything* (SAM), desenvolvido pela Meta AI, representa um marco no avanço dos modelos fundacionais em visão computacional, com foco na tarefa de segmentação de imagens. Inspirado pelos modelos de linguagem natural, como Chat GPT, que generalizam via prompt engineering, o SAM introduz o conceito de segmentação acionável por prompt, onde é possível segmentar qualquer objeto a partir de um ponto, caixa, máscara ou mesmo prompt textual (Kirillov et al., 2023). Seu principal diferencial é a capacidade de atuar em regime *zero-shot*, ou seja, aplicar-se a novas distribuições de dados sem necessidade de re-treinamento, viabilizando aplicações em domínios nunca vistos. Essa abordagem permitiu o treinamento de um modelo altamente generalista, com um desempenho competitivo em diversas tarefas como segmentação interativa, segmentação de instâncias e object proposal, mesmo sem treinamento específico para elas.

Para habilitar essa generalização, o SAM foi treinado com a maior base de dados de segmentação já construída: o SA-1B, com mais de 1 bilhão de máscaras geradas automaticamente em 11 milhões de imagens de alta resolução. O modelo é composto por três partes principais: um codificador de imagem baseado em *Vision Transformers* (ViT), um codificador de prompt capaz de interpretar múltiplas formas de entrada (como pontos ou textos) e um decodificador leve que retorna máscaras de segmentação, como apresentado na Figura 7. Essa estrutura modular permite reutilizar a codificação da imagem para diferentes prompts, alcançando desempenho em tempo quase real. O SAM não apenas propõe uma nova arquitetura e tarefa de segmentação, mas também redefine o papel da segmentação automática como ferramenta para anotação assistida e construção de datasets, tornando-se peça-chave para acelerar o desenvolvimento de novos modelos e aplicações em visão computacional (Kirillov et al., 2023).

Figura 7 – Estrutura do modelo *Segment Anything* (SAM)



Fonte: Kirillov et al. (2023)

3.3 DETECÇÃO DE INSTRUMENTOS CIRÚRGICOS

A detecção automática de instrumentos cirúrgicos é uma aplicação promissora da visão computacional no contexto hospitalar, com potencial para auxiliar na automação de tarefas, rastreamento de materiais, inventário intraoperatório e sistemas robóticos assistivos. Diversas abordagens têm sido propostas ao longo dos anos, variando conforme o grau de complexidade computacional e a época. Esta seção apresenta um panorama dessas abordagens, organizadas em dois momentos principais: antes e depois do advento do aprendizado profundo (*Deep Learning*).

3.3.1 Abordagens pré-*Deep Learning*

O trabalho de Carpintero et al. (2010) descreve o desenvolvimento de uma enfermeira robótica de instrumentação concebida para apoiar a equipe em salas operatórias. O sistema integra reconhecimento de fala (IBM ViaVoice), visão computacional com uma câmera monocromática Sony XC-56 (659 × 494 px, 30 fps) e um braço Fanuc LR-Mate com pinça eletromagnética para manipular os instrumentos. A biblioteca Matrox Imaging Library (MIL), via módulo Model Finder, gera padrões de contorno e centroide que permitem identificar, mesmo sob oclusão parcial, sete instrumentos distintos em tempo real. Nos ensaios, o reconhecimento de voz atingiu 93,5% de acerto (n = 600) e a identificação visual obteve 98,1% de correspondência em 175 execuções, sem falsos positivos. As limitações residem no número restrito de instrumentos, na presença de apenas uma câmera fixa e na necessidade de comandos explícitos para cada ciclo; os autores propõem, como trabalho futuro, ampliar o conjunto de ferramentas, incorporar múltiplas câmeras e automatizar a recolocação quando o instrumento retorna à zona de troca (Carpintero et al., 2010).

O estudo de Tan et al. (2015), apresenta um sistema de triagem que usa o robô colaborativo Baxter para retirar instrumentos cirúrgicos “sujos” de uma bandeja caótica e depositá-los em recipientes separados por tipo. A percepção emprega uma câmera DSLR montada sobre a

área de trabalho: o módulo comercial KeyDot® lê barcodes para identificar cada instrumento e um algoritmo de template matching sobre mapas de distância estima sua pose 4-DOF; em seguida, um raciocínio de oclusão seleciona o objeto no topo da pilha e calcula o ponto de pega, transmitido a um gripper eletromagnético. Toda a lógica opera dentro de um ciclo Perception-Planning-Action orquestrado por máquina de estados finitos, dispensando intervenção humana durante a singulação e o depósito. Em testes com doze instrumentos, o algoritmo visual escolheu corretamente o item menos ocluído em 95% dos quadros; quando isso ocorria, a taxa de singulação chegava a 98%, mas o êxito global caiu para 80% devido a derrapagens provocadas pela conformidade do Baxter. Entre as limitações destacam-se a dependência de uma única câmera zenital, a exigência de barcodes visíveis e a sensibilidade à imprecisão cinemática do robô—pontos que os autores planejam mitigar com grippers mais robustos e métodos de detecção menos restritivos em trabalhos futuros (Tan et al., 2015).

3.3.2 Abordagens com Deep Learning

A dissertação de mestrado de Diana M. Lavado (2018) propõe um sistema de visão para, num tabuleiro desorganizado, detectar quatro instrumentos e decidir qual remover primeiro, preparando um robô pick-and-place para montar kits esterilizados. Após testar segmentações clássicas (Canny, Otsu, watershed), a autora constrói um dataset próprio e treina redes YOLOv2 e YOLOv3 (detecção) mais quatro YOLOv2 específicas de "*occlusion reasoning*". A melhor versão da YOLOv2, com 117 000 iterações, alcança mAP 90,06% e *IoU* 72,6% na detecção dos quatro instrumentos , enquanto a implementação posterior de YOLOv3 eleva a média para mAP 91,98% e *IoU* 78,9% em 49 000 iterações. Para raciocínio de oclusão, as redes obtêm precisão média entre 89% e 92%. O pipeline completo (detecção + oclusão + seleção) processa cada imagem em $\approx$ 10s num PC padrão, mas a etapa de detecção isolada roda a 0,2 s, sugerindo viabilidade de quase tempo real. As principais limitações identificadas são de haver apenas quatro classes, dependência de imagens estáticas de câmera zenital e necessidade de expandir o algoritmo de oclusão para múltiplos empilhamentos (Lavado, 2018).

O estudo de Kletz et al. (2019) investiga a segmentação e o reconhecimento de instrumentos em vídeos de laparoscopia ginecológica por meio de Mask R-CNN com backbone ResNet-101, treinada em um dataset próprio de apenas 333 quadros (540 × 360 px) contendo 561 máscaras que cobrem 11 tipos de instrumentos. Para separar ferramenta e fundo ("binary"), o modelo obteve AP50 = 82% e AP50:95 = 0,63 após 50 épocas, demonstrando boa localização mesmo com poucos exemplos. No cenário mais difícil de reconhecimento multi-classe, a precisão caiu para AP50 = 62% e AP50:95 = 0,43, com forte variação entre categorias (p.ex., Needle AP50: 0,22 versus Sealer-Divider AP50: 1,00). Os autores atribuem essas limitações ao

tamanho reduzido do conjunto de dados, ao desequilíbrio entre classes e à semelhança visual entre instrumentos finos, sugerindo ampliar a base e explorar técnicas adicionais de aumento de dados e de distinção de formas para elevar a robustez do sistema (Kletz et al., 2019).

O estudo de Deol et al. (2024) demonstra um modelo de visão computacional baseado no YOLOv9 treinado em um dataset próprio de 1004 imagens (13213 instâncias, 11 categorias de instrumentos) para automatizar a detecção e contagem de ferramentas e, assim, mitigar eventos de "item cirúrgico retido" (ICR). Após treinamento com ampliações de dados e 640×640px de entrada, o sistema alcançou precisão global de 98,5% e revocação de 99,9% na distinção instrumento-fundo, além de mAP@0,5 de 99,4%; por classe, mantendo desempenho robusto mesmo em imagens com sobreposição de ferramentas. Em vídeo dinâmico simulado, manteve contagem correta em todos os quadros estáveis, processando 40 fps numa GPU NVIDIA V100, sugerindo viabilidade em tempo real. Entre as limitações destacam-se o uso de cenário laboratorial com fundo azul padronizado, a cobertura restrita a 11 categorias e a ausência de validação clínica com variáveis como sangue, iluminação variável ou instrumentos raros; os autores propõem ampliar o dataset e testar o algoritmo em salas operatórias reais para avaliar robustez, implicações éticas e integração ao fluxo cirúrgico (Deol et al., 2024).

3.4 SÍNTESE E MOTIVAÇÃO

Em síntese, os avanços revistos neste capítulo — do surgimento das CNNs ao paradigma prompt-able do Segment Anything — revelam um movimento convergente rumo a modelos cada vez mais gerais, rápidos e capazes de operar sob fortes oclusões. Esses requisitos são exatamente os que emergem no cenário de bin-picking de instrumentos cirúrgicos discutido na Seção 3.1.3. Nos capítulos experimentais, demonstra-se como a combinação YOLOv11 + SAM atende a tais demandas, destacando ganhos de mAP@0,5 frente a abordagens anteriores.

4 MATERIAIS E MÉTODOS

Este capítulo descreve detalhadamente os materiais e procedimentos empregados no desenvolvimento e avaliação do presente trabalho. Inicialmente, é apresentado o conjunto de dados utilizado, derivado do estudo de Lavado (2018), seguido da metodologia adotada para a reclassificação das imagens com auxílio de ferramentas modernas de segmentação. Em seguida, são descritas as três abordagens de detecção comparadas - incluindo a linha de base original e duas variantes modernas com YOLOv11 -, bem como o algoritmo proposto para inferência de oclusão entre objetos. Por fim, é apresentada a metodologia de avaliação dos resultados, com base em métricas padronizadas na literatura de visão computacional.

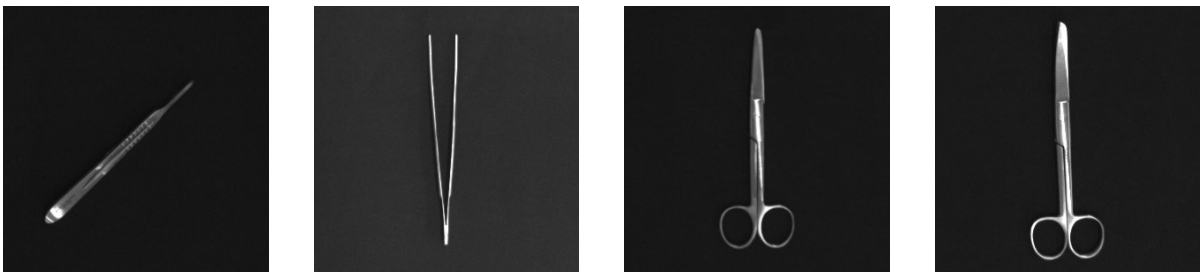
4.1 CONJUNTO DE DADOS

Durante o desenvolvimento do Projeto InstruBOT — Análise de Viabilidade para Automação em Procedimentos de Instrumentação Cirúrgica em Cirurgias Ortopédicas, realizado em 2024 pelo autor deste trabalho junto ao Laboratório de Robótica Móvel (LabROM) da Escola de Engenharia de São Carlos (EESC-USP), foi conduzido um mapeamento junto ao Hospital das Clínicas da Faculdade de Medicina de Ribeirão Preto com o objetivo de identificar os instrumentos de maior frequência nos procedimentos ortopédicos. O levantamento evidenciou que os instrumentos com maior fluxo de manipulação envolvem, majoritariamente, ferramentas de corte, como bisturis e tesouras, e instrumentos de preensão, como pinças.

Embora não tenha sido desenvolvido no escopo do InstruBOT, o conjunto de dados proposto por Lavado (2018) se mostra altamente representativo para o problema identificado. O dataset simula bandejas cirúrgicas desorganizadas contendo quatro classes principais de instrumentos - bisturi (Bisturi nº4), tesoura curva (Tesoura Mayo Curva), tesoura reta (Tesoura Mayo Reta) e pinça (Pinça de Dissecção Reta) - distribuídos de forma aleatória e frequentemente ocluídos, a Figura 8 apresenta as quatro classes. As imagens foram capturadas em diferentes configurações, permitindo a análise de variabilidade de posição, sobreposição e orientação dos objetos. A base foi originalmente organizada em três subconjuntos (train com 75% das imagens, val com 15% das imagens, test com 10% das imagens) e, neste trabalho, será utilizada como referência, com reclassificação moderna a partir de técnicas de segmentação automática, para análise comparativa com modelos de última geração.

A Tabela 1 apresenta a distribuição de cada uma das quatro classes e também quantas fotos elas possuem intersecção com dois ou mais objetos na mesma imagem.

Figura 8 – Exemplos dos quatro instrumentos cirúrgicos presentes no conjunto de dados.



(a) Bisturi nº4 (b) Pinça de Dissecção Reta (c) Tesoura Mayo Reta (d) Tesoura Mayo Curva

Fonte: Lavado (2018)

Tabela 1 – Distribuição da quantidade de imagens no conjunto de dados.

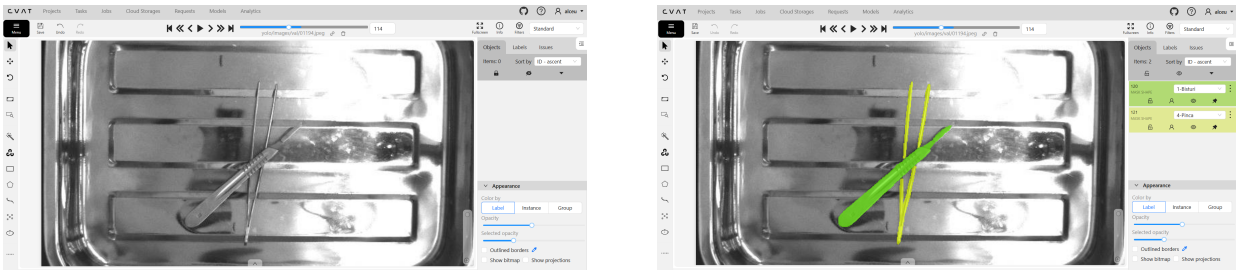
Instrumentos cirúrgicos presentes na imagem	**Qtd. de imagens**
Bisturi nº4 (individualmente)	**550**
+ Pinça de Dissecção Reta	71
+ Tesoura Mayo Reta	49
+ Tesoura Mayo Curva	64
Pinça de Dissecção Reta (individualmente)	**460**
+ Bisturi nº4	64
+ Tesoura Mayo Reta	76
+ Tesoura Mayo Curva	80
Tesoura Mayo Reta (individualmente)	**450**
+ Bisturi nº4	59
+ Pinça de Dissecção Reta	77
+ Tesoura Mayo Curva	79
Tesoura Mayo Curva (individualmente)	**550**
+ Bisturi nº4	69
+ Pinça de Dissecção Reta	117
+ Tesoura Mayo Reta	70
Todos os instrumentos	**100**

4.2 ANOTAÇÃO E CLASSIFICAÇÃO

Para viabilizar inferências relacionadas à oclusão e à ordenação espacial dos instrumentos - ou seja, identificar corretamente quais objetos estão sobrepostos e qual está acima ou abaixo na pilha - torna-se necessário utilizar anotações mais detalhadas do que aquelas fornecidas por caixas retangulares, ou *bounding boxes*. Redes treinadas exclusivamente com *bounding boxes* são limitadas na detecção de sobreposição parcial e não possibilitam a extração precisa dos contornos dos instrumentos, inviabilizando a segmentação por instância e, consequentemente, a análise da estrutura espacial da imagem.

Diante dessa limitação, optou-se por realizar a reanotação completa do conjunto de dados original utilizando a plataforma CVAT (Computer Vision Annotation Tool) - uma ferramenta open-source amplamente utilizada para anotação de imagens e vídeos em projetos de visão computacional. Embora o dataset de Lavado (2018) contasse apenas com anotações do tipo *bounding box*, essa representação se mostra insuficiente em contextos com oclusão, sobreposição entre instrumentos ou contornos irregulares. Dessa forma, todas as imagens foram reclassificadas com máscaras de segmentação completas por instrumento, de modo que cada instância presente na imagem fosse representada de forma precisa, pixel a pixel. A Figura 9 apresenta a plataforma do CVAT na anotação do dataset.

Figura 9 – Visualização do processo de anotação no CVAT.



(a) Plataforma do CVAT, sem anotaçoes (b) Plataforma CVAT, com classes anotadas

Fonte: Elaborado pelo Autor a partir do Dataset de Lavado (2018)

Para acelerar esse processo, foi utilizado o modelo *Segment Anything* (SAM), integrado ao CVAT, que permite a geração automática de máscaras a partir de prompts como cliques ou caixas delimitadoras. O operador apenas refinava ou corrigia manualmente as segmentações geradas, garantindo precisão sem comprometer produtividade. Ao final do processo, os dados foram exportados no formato COCO JSON, contendo as estruturas de imagens ("*images*"), anotações ("*annotations*") e categorias ou classes ("*categories*"), compatíveis com os pipelines de segmentação utilizados neste trabalho. Além disso, foi implementado um script de conversão

das máscaras em dois formatos complementares: caixas delimitadoras (YOLO .txt) e máscaras poligonais compactadas, possibilitando o uso tanto em modelos baseados em detecção quanto em segmentação.

4.3 TREINAMENTO DOS MODELOS

O conjunto de dados anotado foi dividido em três subconjuntos para os experimentos: treinamento (train) com 2.257 imagens, validação (val) com 300 imagens, e teste (test) com 453 imagens. A separação foi realizada aleatoriamente após a rotulagem, garantindo uma distribuição equilibrada das diferentes combinações de instrumentos em cada subconjunto.

O processo de treinamento foi realizado em um ambiente com GPU dedicada NVIDIA (modelo L40s), utilizando o framework Ultralytics (2024) implementado em Python 3.10. O código dos treinamentos, de validação e inferência, arquivos .YAML e demais algoritmos utilizados no trabalho encontram-se disponibilizado no repositório oficial do projeto[1]. Os modelos foram treinados por até 250 épocas, com o critério de escolha do melhor modelo sendo o maior valor de $mAP@0,50$ no conjunto de validação.

Na Tabela 2 são apresentados os principais hiperparâmetros utilizados durante os treinamentos das abordagens YOLOv3, YOLOv11 (caixas) e YOLOv11 (segmentação).

Tabela 2 – Parâmetros de treinamento dos modelos.

Parâmetro	**Valor**
Tamanho da imagem	640 × 480 pixels
Número de épocas	250
Batch size	16 (YOLOv3), 32 (YOLOv11)
Otimização	SGD (YOLOv3), Adam (YOLOv11)
Taxa de aprendizado inicial	0,01 (YOLOv3), 0,0001 (YOLOv11-bbox), 0,002 (YOLOv11-seg)
Critério de parada	Melhor $mAP@0,5$ no conjunto de validação
Aumento de dados	Aleatorização de escala, corte, rotação
Framework	Ultralytics + PyTorch 2.1.0
Dispositivo	GPU NVIDIA L40S

Por fim, após a realização do treinamento, utilizou-se outro hardware para inferência de modo a avaliar o tempo real em dispositivo limitado. Desse modo, a inferência foi realizada em um computador com Intel(R) Core(TM) i5-8265U em CPU de 1,60GHz e 8G de RAM.

[1] https://github.com/alceu-brettas/instrubot-bin-picking

4.4 ABORDAGENS DE DETECÇÃO

A partir do conjunto de dados reanotado e preparado, este trabalho propõe a comparação entre três abordagens distintas de detecção de instrumentos cirúrgicos. Cada uma utiliza uma arquitetura de rede neural diferente e estratégias variadas de supervisão, com o objetivo de avaliar os ganhos proporcionados por técnicas modernas de segmentação e detecção. A Tabela 3 apresenta um resumo comparativo entre os métodos analisados, destacando suas principais características técnicas. Em seguida, detalha-se individualmente cada uma das abordagens implementadas.

Tabela 3 – Comparativo entre as abordagens de detecção avaliadas

Característica	**YOLOv3 (BBox)**	**YOLOv11 (BBox)**	**YOLOv11 (Segmentação)**
Modelo principal	YOLOv3u	YOLOv11n	YOLOv11n
Tipo de anotação	*Bounding Box*	*Bounding Box*	Máscaras COCO + BBox
Segmentação de contorno	Não	Não	Sim
Arquitetura moderna e suportada	Não	Sim	Sim
Necessita redes auxiliares para oclusão	Sim	Sim	Não
Complexidade de implementação	Baixa	Baixa	Média
Flexibilidade para novos cenários	Baixa	Média	Alta

Neste projeto, foram utilizadas as variantes "*nano*"dos modelos de YOLO, por apresentarem menor complexidade computacional e maior velocidade de inferência, sendo adequadas para aplicações em tempo real.

4.4.1 Baseline-Lavado: YOLOv3 – Detecção com *Bounding Boxes*

A primeira abordagem implementada neste trabalho utiliza o modelo YOLOv3, conforme originalmente proposto por Lavado (2018) (que iniciou trabalho com YOLOv2 e percebeu

melhor desempenho na versão 3 da rede), porém com duas modificações importante: não foram utilizadas as quatro redes auxiliares de raciocínio de oclusão presentes na proposta original e a versão utilizada da YOLO foi a YOLOv3u desenvolvida após a integração da YOLO a Ultralytics. O modelo foi treinado com as anotações de *bounding boxes* geradas automaticamente a partir das máscaras criadas com o Segment Anything (SAM), garantindo consistência com as demais abordagens. A finalidade dessa abordagem é servir como linha de base histórica, mantendo a arquitetura da época, mas simplificando o pipeline e focando apenas na capacidade de detecção do modelo. Embora menos eficiente que modelos recentes como YOLOv11, sua inclusão permite avaliar a evolução técnica da arquitetura YOLO ao longo dos anos frente a um mesmo conjunto de dados. Para facilitar comparação com versões mais modernas do YOLO, realiza-se o treinamento com a YOLOv3u — reimplementação em PyTorch que mantém o backbone Darknet-53 original, mas incorpora melhorias de treinamento e suporte a GPUs modernas (Redmon; Farhadi, 2018). No trabalho de Lavado (2018) empregou-se a implementação oficial em C do framework Darknet.

4.4.2 Estado da Arte: YOLOv11 – Detecção com *Bounding Boxes*

A segunda abordagem consistiu na utilização do modelo YOLOv11, uma das versões mais recentes da arquitetura You Only Look Once mantida pela Ultralytics, configurada para a tarefa de detecção com base apenas em caixas delimitadoras (*bounding boxes*). O modelo foi treinado com os dados reanotados a partir do dataset original, agora em formato compatível com a estrutura da Ultralytics (arquivos .txt no estilo YOLO + YAML de configuração). Essa abordagem representa um salto em relação à linha de base anterior por empregar uma arquitetura mais moderna, eficiente e com suporte a treinamento em GPUs atuais, além de eliminar a necessidade de múltiplas redes auxiliares. Entretanto, por não considerar a máscara completa dos objetos, sua capacidade de avaliar o grau de sobreposição entre instrumentos é limitada, o que afeta diretamente a inferência de oclusão.

4.4.3 Estado: YOLOv11 – Segmentação com Máscaras

A terceira abordagem explorou o uso da arquitetura YOLOv11 estendida para tarefas de segmentação de instâncias, integrando ao treinamento as máscaras obtidas via Segment Anything (SAM). Utilizando o mesmo conjunto de imagens reanotadas no CVAT, mas agora com máscaras poligonais extraídas do formato COCO JSON, o modelo foi treinado em modo híbrido, permitindo aprender não apenas a localização por caixas, mas também a silhueta precisa de cada instrumento. Essa abordagem é a mais rica do ponto de vista de representação espacial e permite inferir com maior precisão o grau de oclusão entre objetos — inclusive viabilizando

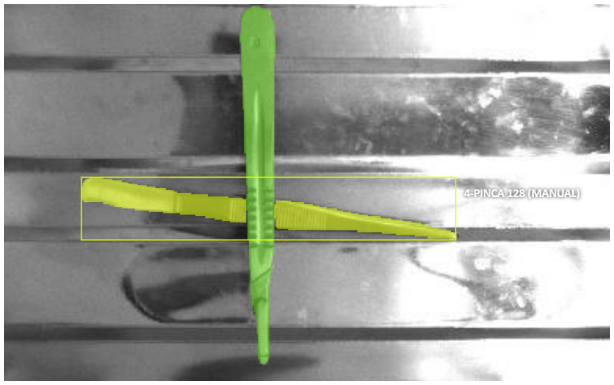
a implementação de algoritmos de sobreposição baseados em área visível. Além disso, essa configuração elimina a necessidade de redes auxiliares para inferência de oclusão, já que as próprias máscaras podem ser utilizadas para inferir relações de profundidade de forma analítica.

4.5 ALGORITMO DE OCLUSÃO

Com o objetivo de inferir a relação de oclusão entre instrumentos cirúrgicos presentes nas imagens — isto é, identificar quais estão visíveis integralmente e poderiam ser retirados primeiro por um manipulador robótico e quais estão parcialmente cobertos por outros — este trabalho propõe um algoritmo analítico baseado nas máscaras de segmentação geradas pelo modelo YOLOv11-seg durante a inferência. A estratégia adotada busca tirar proveito das informações espaciais contidas nas máscaras segmentadas, dispensando o uso de redes auxiliares ou classificações supervisionadas adicionais para determinar relações de sobreposição, e com isso, identificar qual objeto está por cima dos demais (TOP) e pode ser retirado primeiro da mesa, seguindo o conceito de *bin-picking* apresentado anteriormente.

O algoritmo parte da premissa de que, em cenários de oclusão, a máscara do instrumento inferior frequentemente apresenta descontinuidades — ou seja, está fragmentada em duas ou mais ilhas desconectados (blobs). Dessa forma, o primeiro critério adotado é a contagem do número de componentes conexas da máscara: caso haja apenas um blob, o objeto é classificado como TOP (vísivel); se todos os objetos possuírem mais de um blob, passa-se a uma segunda verificação. A Figura 10 demonstra a sobreposição de um bisturi (verde) que deixou uma pinça (amarelo) em oclusão com duas ilhas separadas da sua máscara (blobs).

Figura 10 – Exemplo de imagem com Pinça em oclusão e com ilhas (blobs) divididos



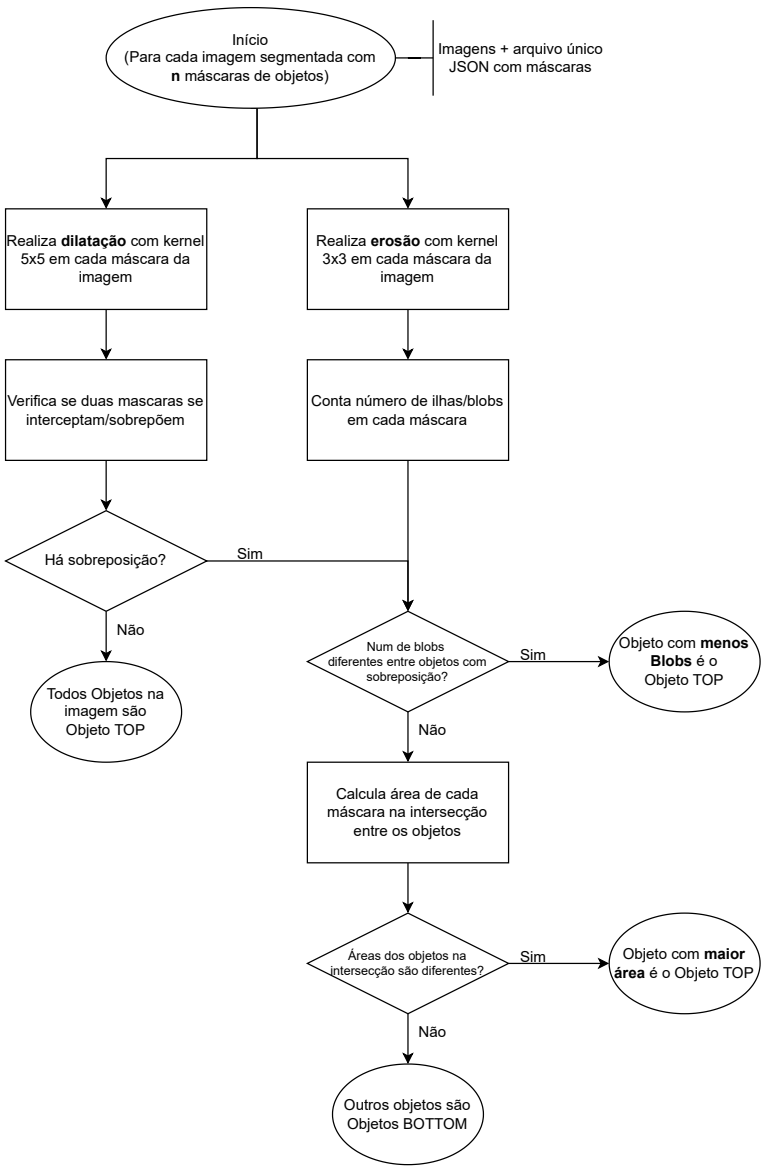
Fonte: Elaborado pelo Autor

Como podem existir dois objetos distantes em uma mesma imagem sem oclusão, antes da contagem de blobs é realizado uma dilatação das máscaras de todos os objetos com um kernel

5x5 de modo que, somente avalia-se a oclusão analisando os blobs caso haja sobreposição entre duas máscaras dilatadas. Além disso, após realizar os primeiros testes de avaliação de oclusão foi percebido que algumas máscaras inferidas pelo modelo, ao invés de serem geradas com blobs separados pela oclusão, possuíam um linha de poucos pixels conectando cada blob, desse modo, para mitigar esse problema foi realizado uma erosão com um kernel 3x3 nas imagens antes da realização da contagem de blobs.

Por fim, caso a contagem de blobs não consiga identificar o objeto TOP, o algoritmo realiza uma comparação da área vísivel de cada objeto no espaço de intersecção entre eles, de modo que, o objeto com maior área visível é classificado como TOP, os demais objetos são classificados como BOTTOM. Essa lógica, ilustrada no fluxograma da Figura 11, oferece uma inferência explicável, de baixo custo computacional e compatível com o pipeline de segmentação utilizado neste trabalho.

Figura 11 – Fluxograma do algoritmo de detecção de oclusão



Fonte: Elaborado pelo Autor

4.6 METODOLOGIA DE AVALIAÇÃO DE RESULTADOS

Para comparar objetivamente o desempenho das três abordagens propostas neste trabalho, foram definidos critérios quantitativos baseados em métricas amplamente utilizadas na literatura de detecção de objetos e segmentação semântica. A avaliação foi realizada exclusivamente sobre o conjunto de teste, que permaneceu isolado ao longo de todo o processo de treinamento e ajuste de hiperparâmetros, garantindo assim a imparcialidade dos resultados.

As métricas principais utilizadas são descritas na Tabela 4. Os indicadores $mAP@0,5$ e $mAP@[0,5:0,95]$ quantificam, respectivamente, a média da precisão para sobreposição mínima de 50% entre caixas preditas e reais, e a média para múltiplos limiares entre 50% e 95% com incremento de 5%, conforme definido no desafio COCO (Lin; Maire; Belongie, 2014). Já as métricas de precisão (*precision*) e revocação (*recall*) avaliam a confiabilidade e a abrangência das predições. Por fim, o indicador de FPS (frames por segundo) de inferência complementa a avaliação sob a ótica da eficiência computacional.

Tabela 4 – Métricas utilizadas para avaliação dos modelos

Métrica	**Descrição**
`mAP@0,5`	Média da precisão para *IoU* maior ou igual a 0,50 (Everingham et al., 2010)
`mAP@[0,5:0,95]`	Média das precisões para *IoU*s entre 0,50 e 0,95 com passo 0,05 (Lin; Maire; Belongie, 2014)
Precisão	Proporção de predições corretas entre todas as positivas
Revocação	Proporção de objetos reais corretamente detectados
FPS	Velocidade de inferência (quadros por segundo)

Além dos indicadores numéricos apresentados, também foram utilizadas as curvas *F1-Confidence* e *Precision-Recall* para cada classe dos modelos. A curva F1-Confidence permite identificar o ponto de confiança (*threshold*) em que o modelo atinge seu melhor equilíbrio entre precisão e revocação, medido pelo *F1-score*, que é a média harmônica dessas duas métricas. Já a curva *Precision-Recall* mostra a relação entre essas variáveis ao longo de diferentes limiares, permitindo avaliar a estabilidade do modelo em diferentes níveis de confiança. Essas representações gráficas enriquecem a análise ao evidenciar visualmente o desempenho por classe e a sensibilidade dos modelos frente a variações no limiar de detecção.

A avaliação objetiva dos modelos foi realizada utilizando o método .val() da biblioteca Ultralytics. Já para a avaliação da lógica de oclusão, foi desenvolvido um algoritmo específico que classifica cada objeto como visível (TOP) ou ocluído (BOTTOM), com base na integridade da máscara e proporção de área visível. A acurácia dessa classificação será estimada por meio de validação cruzada visual, e o algoritmo utilizado para identificação da oclusão está apresentado no Apêndice desse documento e também no github junto de todos os outros arquivos do projeto.

5 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os principais resultados obtidos com os três experimentos de detecção de instrumentos cirúrgicos: (i) YOLOv3 com *bounding boxes*, (ii) YOLOv11 com *bounding boxes* e (iii) YOLOv11 com segmentação por máscaras. São avaliados aspectos quantitativos, como métricas de precisão e velocidade, e qualitativos, por meio de exemplos visuais e análise da robustez em casos de oclusão.

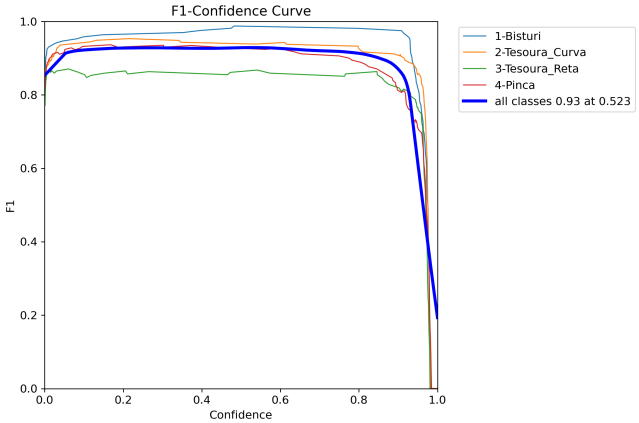
5.1 RESULTADOS QUANTITATIVOS

Nesta seção são apresentados os resultados quantitativos dos três experimentos realizados com diferentes abordagens de detecção. As métricas foram obtidas a partir do conjunto de teste, conforme detalhado no capítulo anterior.

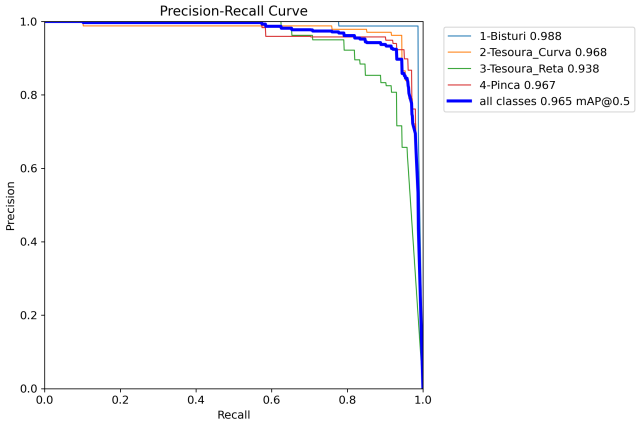
5.1.1 YOLOv3 – Detecção com *Bounding Boxes*

A primeira abordagem testada utilizou a arquitetura YOLOv3, treinada apenas com *bounding boxes* extraídas do dataset. A Figura 12 apresenta as curvas F1-Confidence (Confiança) e Precision-Recall para cada classe do modelo.

Figura 12 – Gráficos de desempenho da rede treinada com YOLOv3 com detecção de *bounding boxes* no subconjunto de validação (val)



(a) Curva da Confiança para cada Classe



(b) Curva da Precisão X Revocação para cada Classe

Fonte: Elaborado pelo Autor a partir do Dataset de Lavado (2018)

A Tabela 5 resume os principais resultados obtidos com esse modelo sobre o conjunto de validação.

Tabela 5 – Resultados da abordagem YOLOv3

Métricas globais da rede

Métrica	**Valor**
$mAP@0,50$	96,5%
$mAP@[0,50:0,95]$	94,1%
Precisão	95,4%
Revocação	90,7%
FPS (infer)	0,76 img/s

Matriz de Confusão com TP, FP e FN por Classe

Classe	**TP**	**FP**	**FN**	**Prec.**	**Revocação**
1-Bisturi	80	1	4	98,8%	96,5%
2-TesCurva	100	8	5	96,1%	91,7%
3-TesReta	61	11	6	91,8%	81,9%
4-Pinca	94	7	6	94,8%	90.4%

Nessa abordagem o objetivo é replicar a linha de trabalho de Lavado (2018), mas com duas atualizações significativas: (i) as *bounding boxes* de anotação foram extraídas a partir de segmentações geradas com o prompt de *Segment Anything* (SAM), mais precisas que o label manual original, e (ii) a versão utilizada da YOLO foi a YOLOv3u da Ultralytics, que incorpora melhorias de desempenho e estabilidade. Os resultados obtidos superaram os da baseline da autora: enquanto Lavado reportou um mAP de $91,98\%$, um *F1-score* de 94% e *IoU* médio de $78,93\%$, nesta abordagem foram alcançados mAP@0,50 de $96,5\%$, *F1-score* máximo de 93% (com confiança de $52,3\%$) e uma precisão média de $95,4\%$. A classe Tesoura Reta permaneceu como o maior desafio, apresentando revocação inferior às demais. Apesar da baixa taxa de inferência ($0,76$ img/s), o desempenho quantitativo comprova a solidez da rede como baseline e confirma que o uso de caixas oriundas de segmentação e a atualização de versão contribuíram para maior acurácia geral, em relação à versão tradicional utilizada no trabalho anterior.

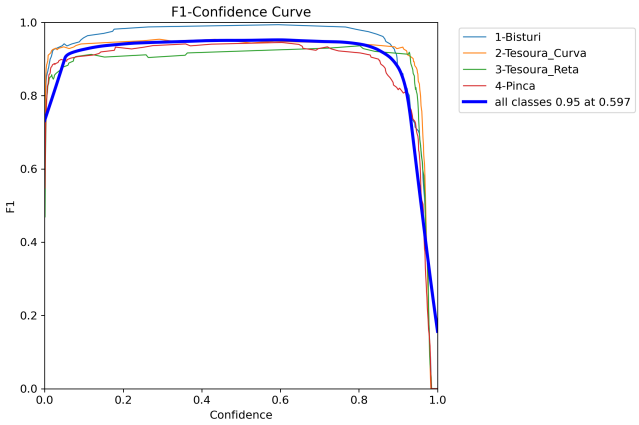
5.1.2 YOLOv11 – Detecção com *Bounding Boxes*

Na segunda abordagem, foi utilizada a versão mais recente da arquitetura YOLO, a YOLOv11, também treinada exclusivamente com *bounding boxes*. Esta versão apresenta diversas otimizações em relação às versões anteriores, principalmente em termos de desempenho computacional e precisão. A Figura 13 apresenta as curvas F1-Confidence (Confiança) e Precision-Recall para cada classe do modelo.

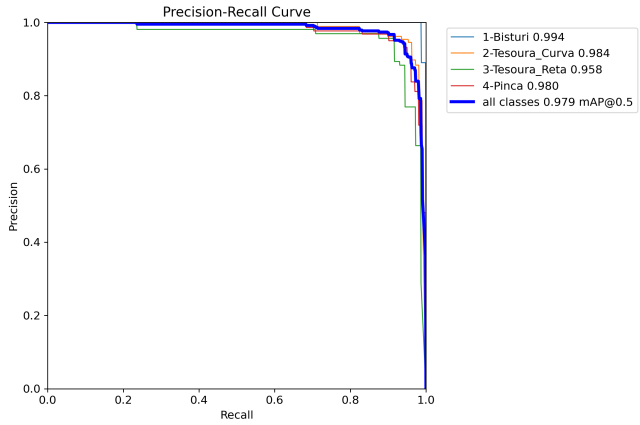
A Tabela 6 apresenta os resultados obtidos por essa configuração sobre o conjunto de validação.

Nessa segunda abordagem, utilizou-se a versão mais recente da família YOLO, a YOLOv11, também treinada com *bounding boxes*. Essa versão apresenta diversas otimizações computacionais e de arquitetura que se refletem nos resultados. O modelo obteve um mAP@0,50 de $97,9\%$, um *F1-score* máximo de 95% (com limiar de confiança de $59,7\%$), precisão de $96,1\%$ e revocação de $94,5\%$, com excelente estabilidade nas curvas *Precision-Recall*. A performance

Figura 13 – Gráficos de desempenho da rede treinada com YOLOv11 com detecção de *bounding boxes* no subconjunto de validação (val)



(a) Curva da Confiança para cada Classe



(b) Curva da Precisão X Revocação para cada Classe

Fonte: Elaborado pelo Autor a partir do Dataset de Lavado (2018)

Tabela 6 – Resultados da abordagem YOLOv11 com detecção de *bounding boxes*

Métricas globais da rede

Métrica	**Valor**
$mAP@0,50$	97,9%
$mAP@[0,50:0,95]$	94,4%
Precisão	96,1%
Revocação	94,5%
FPS (infer)	8,45 img/s

Matriz de Confusão com TP, FP e FN por Classe

Classe	**TP**	**FP**	**FN**	**Prec.**	**Revocação**
1-Bisturi	80	1	1	100%	98,7%
2-TesCurva	103	5	7	96,2%	93,5%
3-TesReta	66	6	9	93,4%	91,7%
4-Pinca	96	5	9	95,0%	98,0%

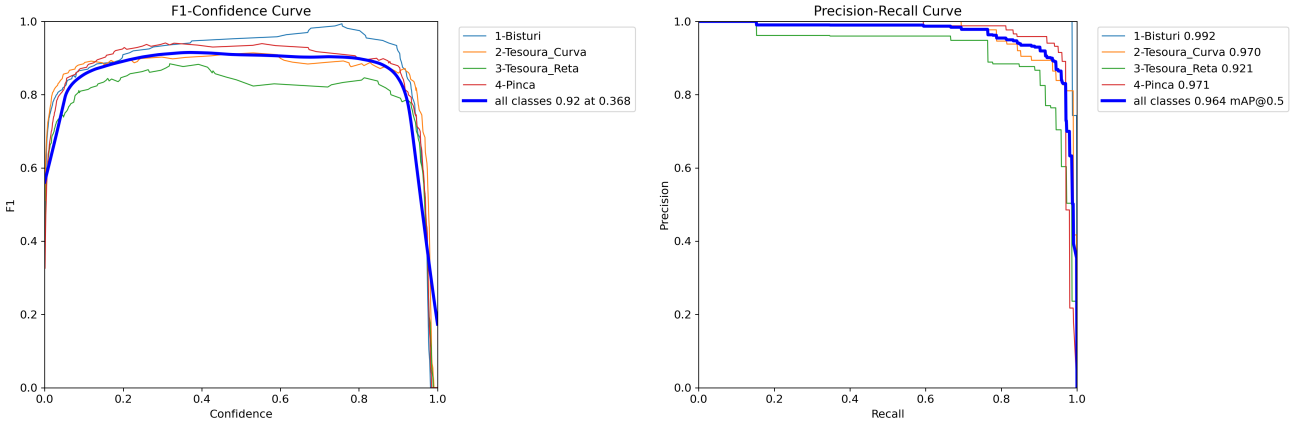
por classe foi alta e homogênea, com destaque para a classe Bisturi, que atingiu 100% de precisão e 98,7% de revocação. Além do ganho em acurácia, o modelo apresentou uma taxa de inferência de 8,45 img/s, evidenciando sua aplicabilidade em cenários com exigência de tempo real. Os resultados confirmam que, mesmo com anotações em formato de *bounding box*, a arquitetura YOLOv11 é substancialmente superior à YOLOv3 em termos de desempenho e eficiência

5.1.3 YOLOv11 – Segmentação com Máscaras

Na terceira abordagem, utilizou-se a variação da YOLOv11 capaz de lidar com segmentações densas, fornecendo não apenas caixas, mas também máscaras por pixel para cada instrumento detectado. Essa configuração visa capturar melhor os contornos dos instrumentos, especialmente em situações com oclusões parciais. A Figura 14 apresenta as curvas F1-Confidence

(Confiança) e Precision-Recall para cada classe do modelo.

Figura 14 – Gráficos de desempenho da rede treinada com YOLOv11 com segmentação por máscaras no subconjunto de validação (val)



(a) Curva da Confiança para cada Classe

(b) Curva da Precisão X Recall para cada Classe

Fonte: Elaborado pelo Autor a partir do Dataset de Lavado (2018)

Os resultados dessa abordagem estão resumidos na Tabela 7.

Tabela 7 – Resultados da abordagem YOLOv11 com segmentação por máscaras

Métricas globais da rede

Métrica	**Valor**
$mAP@0,50$	96,4%
$mAP@[0,50:0,95]$	93,8%
Precisão	89,8%
Revocação	93,9%
FPS (infer)	6,01 img/s

Matriz de Confusão com TP, FP e FN por Classe

Classe	**TP**	**FP**	**FN**	**Prec.**	**Revocação**
1-Bisturi	81	0	11	89,8%	98,8%
2-TesCurva	98	10	20	87,2%	93,5%
3-TesReta	64	8	16	87,0%	88,9%
4-Pinca	96	5	10	95,0%	94,4%

Na terceira abordagem, explorou-se o potencial da YOLOv11 com segmentação densa, capaz de gerar máscaras por pixel para cada instância detectada, além das tradicionais *bounding boxes*. Essa configuração busca representar com mais fidelidade os contornos dos instrumentos, especialmente em casos de oclusão parcial. O modelo alcançou mAP@0,50 de 96,4%, mAP@[0,50:0,95] de 93,8%, *F1-score* máximo de 92% com limiar de 36,8%, e revocação média de 93,9%. Contudo, apresentou a menor precisão entre as três abordagens (89,8%), indicando maior suscetibilidade a falsos positivos. A classe Tesoura Reta, novamente, mostrou os piores índices individuais. Com uma taxa de inferência intermediária (6,01 img/s), o modelo ainda é viável em aplicações práticas. Apesar de apresentar métricas numéricas ligeiramente inferiores

ao YOLOv11 com *boundingboxes*, esta abordagem traz uma vantagem qualitativa importante: a melhor definição de contornos, o que pode ser decisivo em cenários onde o posicionamento exato e a separação entre instrumentos são críticos.

5.1.4 Comparação entre Abordagens

A Tabela 8 sintetiza os principais indicadores das três abordagens testadas, permitindo uma comparação direta entre elas. Esta análise tem como objetivo destacar os ganhos de desempenho associados às evoluções arquiteturais (YOLOv3 $\rightarrow$ YOLOv11) e ao uso de segmentações em substituição às caixas tradicionais.

Tabela 8 – Comparação geral dos resultados das três abordagens

Métrica	**YOLOv3**	**YOLOv11 BBox**	**YOLOv11 Seg**
$mAP@0,50$	96,5%	97,9%	96,4%
$mAP@[0,50:0,95]$	94,1%	94,4%	93,8%
Precisão	95,4%	96,1%	89,8%
Revocação	90,7%	94,5%	93,9%
FPS (infer)	0,76 img/s	8,45 img/s	6,01 img/s

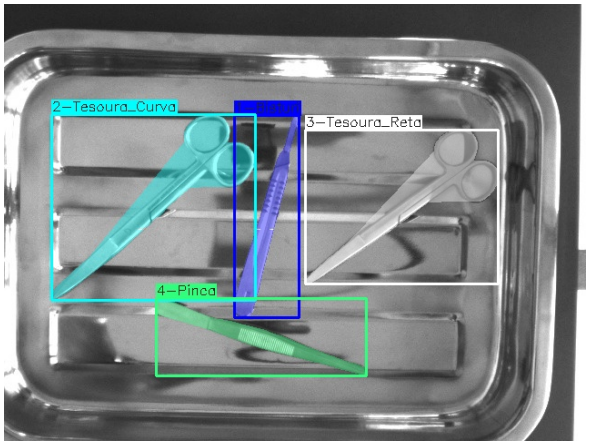
Os resultados demonstram ganhos expressivos com a transição de YOLOv3 para YOLOv11, com destaque para o aumento do mAP@0,50 (de $96,5\%$ para $97,9\%$) e um salto significativo na taxa de inferência (de $0,76$ para $8,45$ imagens por segundo), tornando a nova arquitetura mais adequada a aplicações em tempo real. A versão com segmentação apresentou desempenho próximo ao YOLOv11 com bounding boxes, mantendo revocação elevada ($93,9\%$) e acurácia competitiva, porém com menor precisão ($89,8\%$), possivelmente devido à maior sensibilidade das máscaras a bordas e sobreposições. Assim, o YOLOv11 com bounding boxes se destaca como a abordagem com melhor equilíbrio entre desempenho e eficiência, porém a abordagem com segmentação permite a identificação de oclusão e ordenação de instrumentos, como mostrado em seções posteriores.

5.2 RESULTADOS QUALITATIVOS E VISUAIS

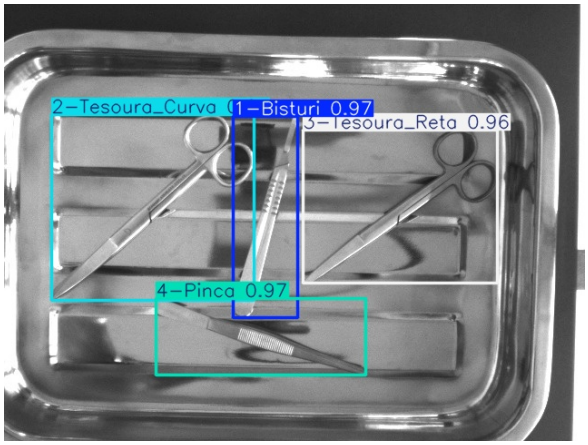
Nesta seção, são apresentados exemplos visuais obtidos nas inferências realizadas pelos três modelos. As imagens foram selecionadas para ilustrar acertos, erros, limitações e situações desafiadoras como oclusões parciais ou múltiplos instrumentos semelhantes.

Na Figura 15, observa-se que todas as abordagens foram capazes de detectar corretamente os quatro instrumentos presentes, porém com variações na qualidade da predição. O

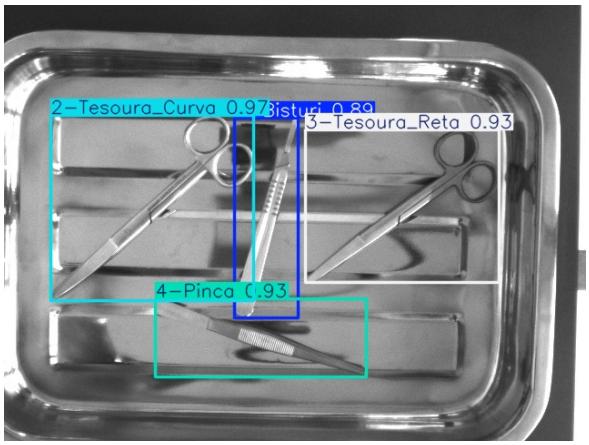
Figura 15 – Exemplo 1 de inferência das três abordagens + imagem *Ground-thruth* de referência



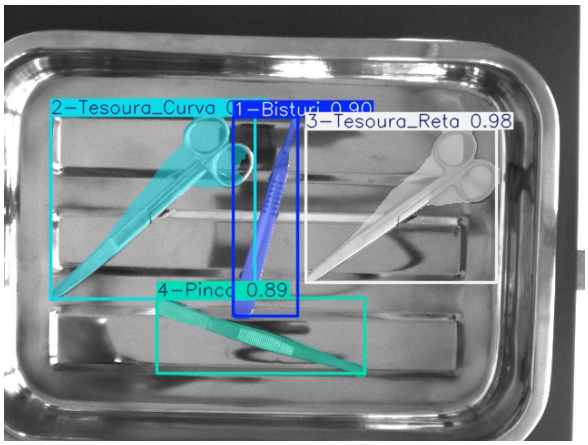
(a) *Ground-thruth* de referência



(b) YOLOv3 com detecção de *bounding boxes*



(c) YOLOv11 com detecção de *bounding boxes*



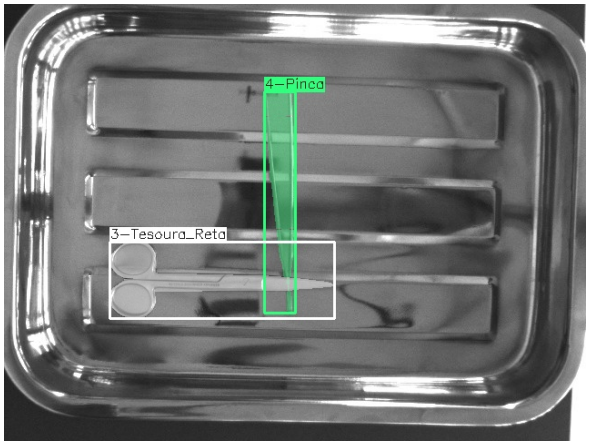
(d) YOLOv11 com segmentação por máscaras

Fonte: Elaborado pelo Autor

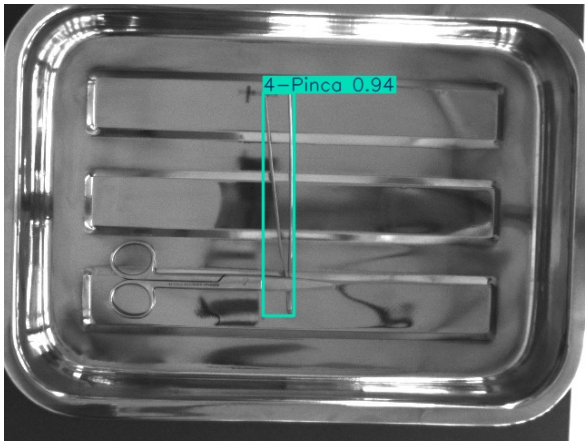
YOLOv3 apresentou caixas menos ajustadas aos objetos, enquanto o YOLOv11 com *bounding boxes* demonstrou maior precisão nas delimitações. Já a versão com segmentação por máscaras destacou-se por representar com maior fidelidade os contornos dos instrumentos, evidenciando com clareza que não há sobreposição entre o bisturi e a pinça, por exemplo - aspecto que, analisando *bounding boxes*, não é tão claro.

Já na Figura 16, todas as abordagens detectaram corretamente a pinça, mas nenhuma identificou a tesoura reta presente no *ground-truth*. As versões com bounding boxes mantiveram comportamentos similares, porém com uma confiança maior no YOLOv3, enquanto a segmentação por máscaras novamente destacou-se pelo contorno mais preciso do instrumento detectado. A falha na detecção da tesoura pode estar relacionada à superfície reflexiva da bandeja, que reduz o contraste e dificulta a diferenciação visual entre o instrumento e o fundo — um desafio

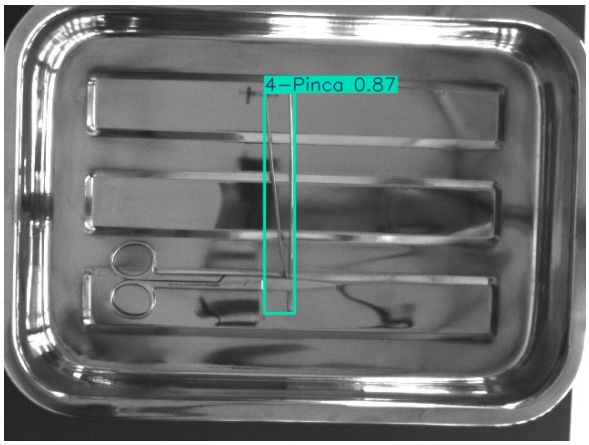
Figura 16 – Exemplo 2 de inferência das três abordagens + imagem *Ground-thruth* de referência



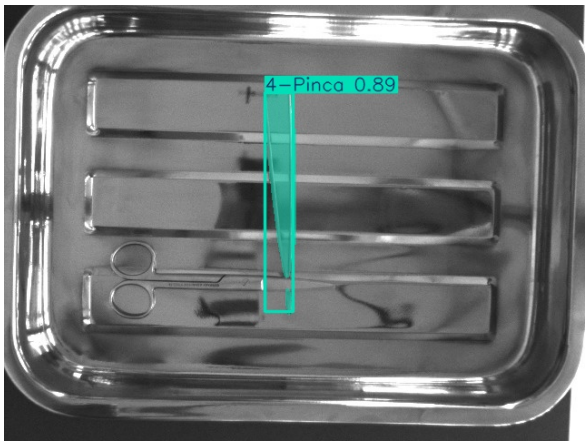
(a) *Ground-thruth* de referência



(b) YOLOv3 com detecção de *bounding boxes*



(c) YOLOv11 com detecção de *bounding boxes*



(d) YOLOv11 com segmentação por máscaras

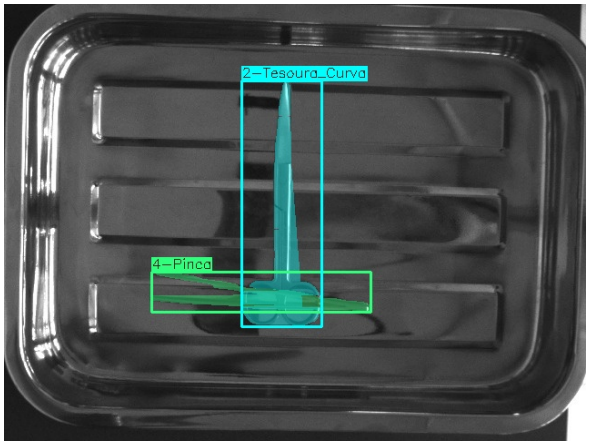
Fonte: Elaborado pelo Autor

recorrente em ambientes cirúrgicos reais.

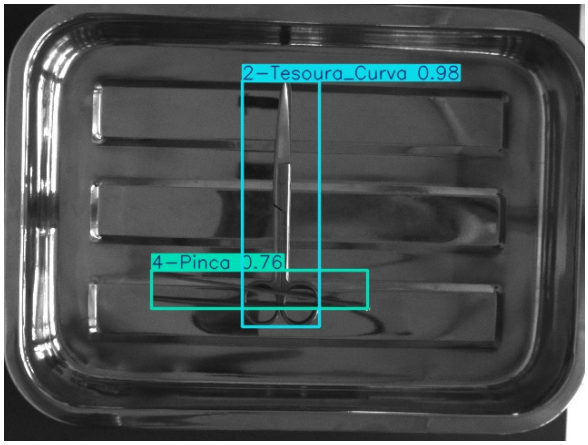
Na Figura 17, as três abordagens conseguiram detectar corretamente tanto a pinça quanto a tesoura curva mesmo com oclusão, mas com variações na qualidade da segmentação. As versões com *bounding boxes* apresentaram sobreposição significativa entre as predições, dificultando a distinção entre os instrumentos. Já a segmentação por máscaras evidenciou melhor separação entre os objetos, ainda que com leves imprecisões nas bordas. A proximidade física e o alinhamento dos instrumentos parecem ter representado um desafio adicional para as redes baseadas apenas em caixas.

Agora na Figura 18, os modelos enfrentaram maior dificuldade devido à forte sobreposição entre os dois instrumentos. O YOLOv3 detectou apenas a tesoura curva, ignorando a tesoura reta. O YOLOv11 com *bounding boxes* conseguiu identificar ambas, mas com baixa

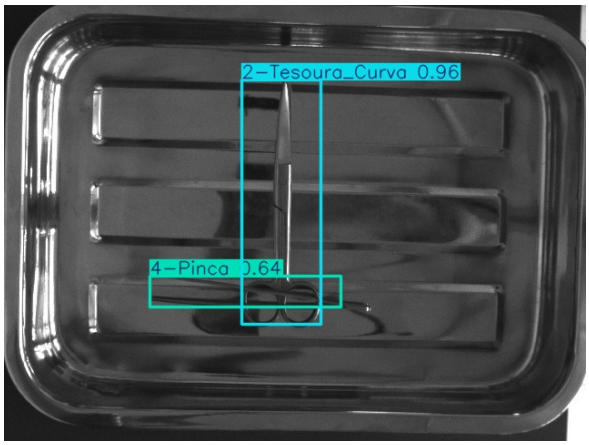
Figura 17 – Exemplo 3 de inferência das três abordagens + imagem *Ground-thruth* de referência



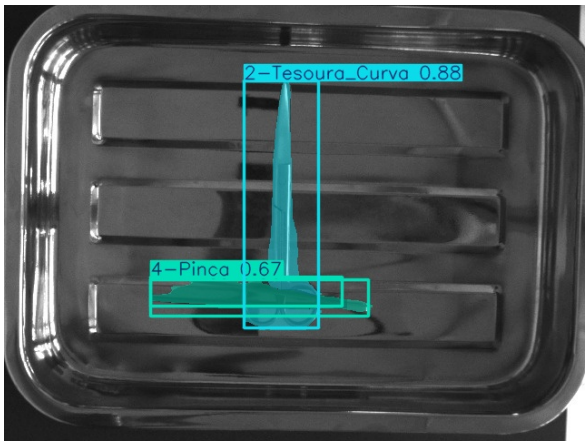
(a) *Ground-thruth* de referência



(b) YOLOv3 com detecção de *bounding boxes*



(c) YOLOv11 com detecção de *bounding boxes*



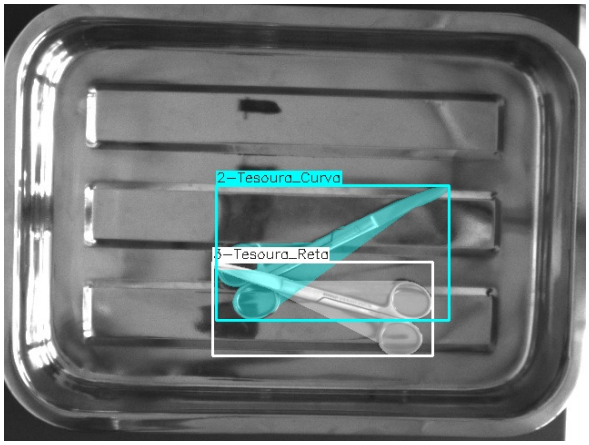
(d) YOLOv11 com segmentação por máscaras

Fonte: Elaborado pelo Autor

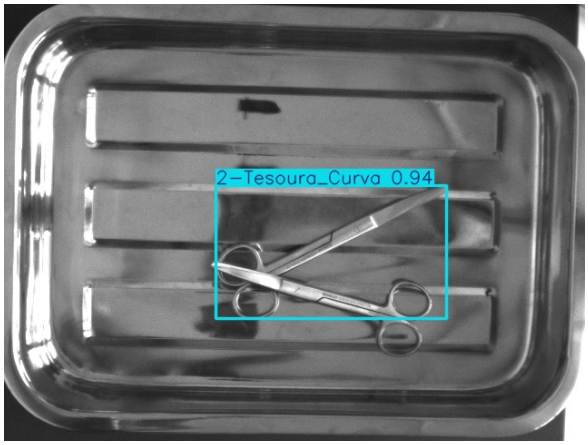
confiança na tesoura reta. Já a versão com segmentação foi a única a representar claramente os dois objetos, mesmo com leves variações nas pontuações de confiança. A semelhança visual entre as duas tesouras - aspecto muito comum em instrumentos cirúrgicos - pode ter contribuído para a redução da confiabilidade nas predições, especialmente em regiões ocluídas, reforçando o papel das máscaras na diferenciação espacial entre objetos semelhantes.

Por fim na Figura 19, apenas a versão segmentada do YOLOv11 foi capaz de identificar corretamente os dois instrumentos presentes — bisturi e pinça. As abordagens baseadas em *bounding boxes*, tanto com YOLOv3 quanto com YOLOv11, detectaram apenas o bisturi. A capacidade da segmentação em distinguir contornos mesmo com oclusões parciais permitiu a correta separação dos objetos, enquanto as demais abordagens parecem ter sido afetadas pela sobreposição e baixa diferenciação entre os elementos na cena, como o fundo da bandeja que

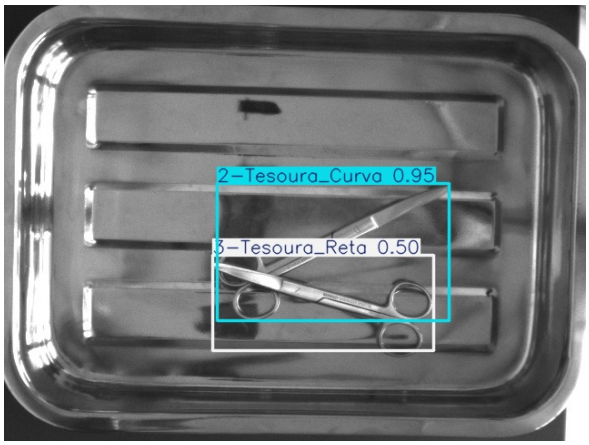
Figura 18 – Exemplo 4 de inferência das três abordagens + imagem *Ground-thruth* de referência



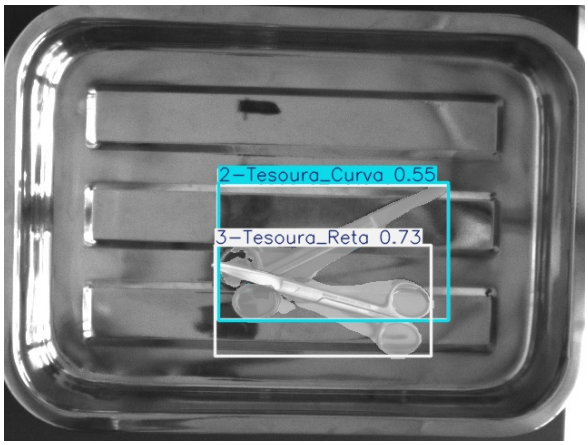
(a) *Ground-thruth* de referência



(b) YOLOv3 com detecção de *bounding boxes*



(c) YOLOv11 com detecção de *bounding boxes*

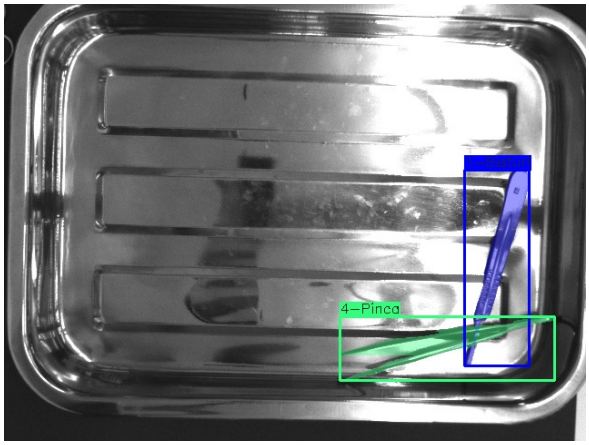


(d) YOLOv11 com segmentação por máscaras

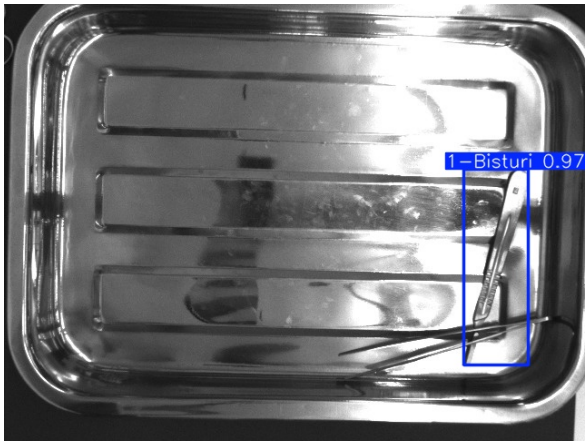
Fonte: Elaborado pelo Autor

deixa a pinça em baixo realce.

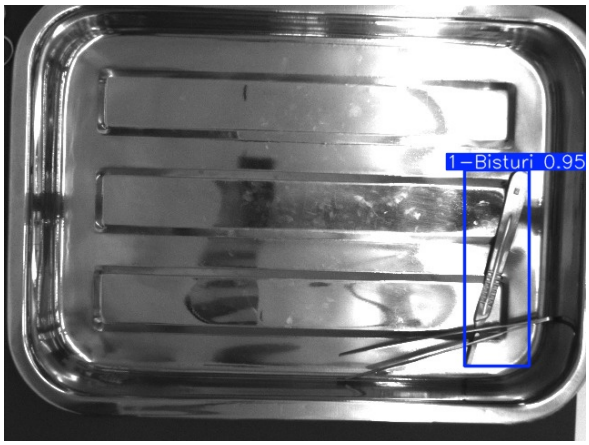
Figura 19 – Exemplo 5 de inferência das três abordagens + imagem *Ground-thruth* de referência



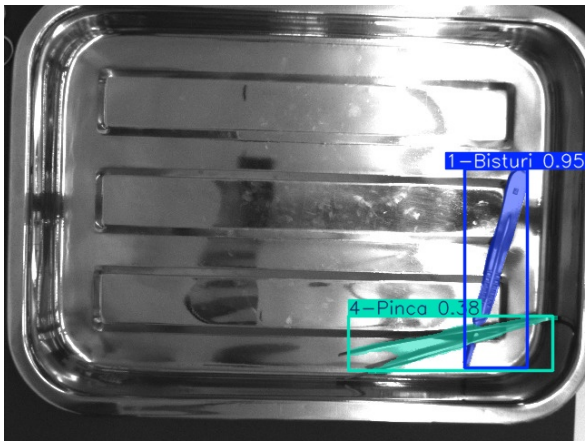
(a) *Ground-thruth* de referência



(b) YOLOv3 com detecção de *bounding boxes*



(c) YOLOv11 com detecção de *bounding boxes*



(d) YOLOv11 com segmentação por máscaras

Fonte: Elaborado pelo Autor

5.3 ANÁLISE DE OCLUSÃO

Com base nas inferências do modelo YOLOv11 com suporte a segmentação por pixel, foi possível aplicar o algoritmo de identificação de oclusão descrito no Capítulo 4. A análise focou em imagens do conjunto de teste que continham dois ou mais instrumentos visíveis e parcialmente sobrepostos.

Para cada par de instrumentos com interseção entre máscaras, o algoritmo estimou o instrumento em posição superior com base no critério de menor número de blobs. A decisão foi validada manualmente por inspeção visual das imagens, permitindo identificar acertos e falhas da lógica proposta.

O modelo de segmentação foi capaz de detectar **75 imagens com oclusão**, e o algoritmo

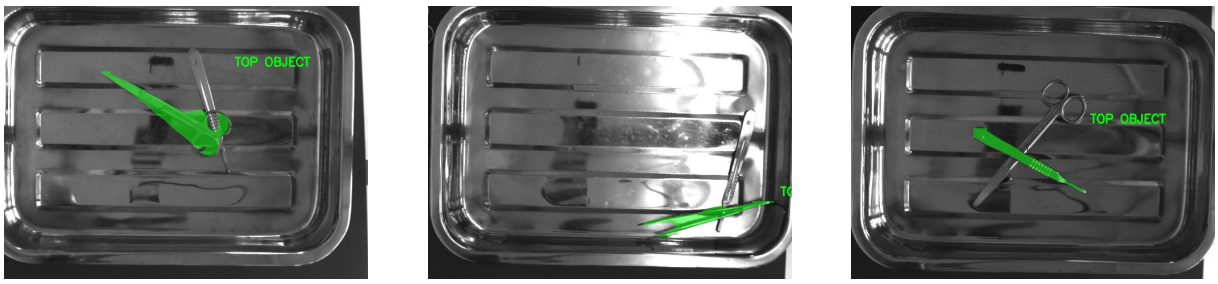
acertou corretamente o objeto TOP em **60 casos**, resultando em uma acurácia de aproximadamente **80%**. As falhas ocorreram, majoritariamente, em casos de:

- Não detecção do instrumento do topo da sobreposição.
- Máscaras incompletas ou com contornos imprecisos devido à segmentação imperfeita.
- Instrumentos com geometria muito semelhante e sobreposição frontal.

Vale ressaltar que por limitação da precisão e revocação, o modelo com segmentação treinado não teve exito na detecção de diversas imagens com oclusão que não foram contabilizadas aqui, por serem relacionadas ao desempenho do modelo em si e apresentadas nas seções anteriores do trabalho.

A Figura 20 apresenta exemplos de acertos do algoritmo, com o contorno verde marcando o instrumento classificado como estando por cima (TOP). Já a Figura 21 mostra casos de falha por motivos diferentes especificados.

Figura 20 – Exemplos de acertos na inferência de oclusão



(a) Tesoura Reta (em verde) acima de Bisturi

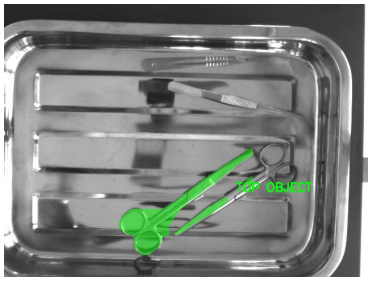
(b) Pinça (em verde) acima de Bisturi

(c) Bisturi (em verde) acima de Tesoura Curva

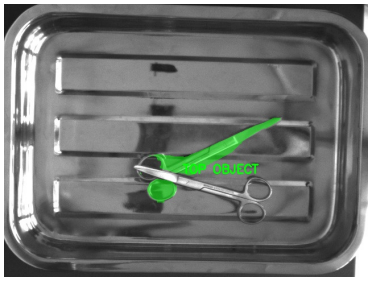
Fonte: Elaborado pelo Autor

Apesar das limitações, os resultados indicam que a estratégia de análise de oclusão baseada em segmentação densa e análise morfológica das máscaras pode ser uma alternativa simples e eficaz às redes auxiliares treinadas separadamente, como no trabalho de Lavado (2018).

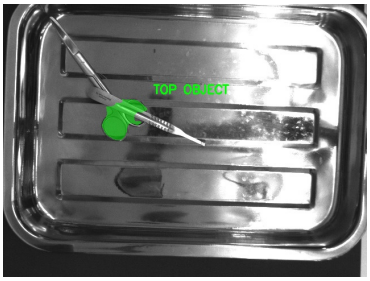
Figura 21 – Exemplos de falhas na inferência de oclusão



(a) Identificação de oclusão inexistente na imagem



(b) Geometria semelhante/ Não detecção da tesoura reta acima da tesoura curva



(c) Máscaras incompletas/Erro na segmentação dos objetos

Fonte: Elaborado pelo Autor

5.4 DISCUSSÃO DOS RESULTADOS

Os experimentos demonstraram ganho consistente de desempenho ao evoluir da arquitetura YOLOv3 para YOLOv11. O modelo YOLOv11 com *bounding boxes* apresentou o melhor equilíbrio entre acurácia (mAP@0,50 de $97,9\%$) e velocidade ($8,45$ img/s), superando a abordagem tradicional da literatura e tornando-se a opção mais viável para aplicações em tempo real.

Já a variante YOLOv11 com segmentação por máscaras manteve mAP elevado ($96,4\%$) e revocação superior a 93%, embora com leve queda de precisão. Apesar desse custo, as máscaras ofereceram vantagem qualitativa importante: permitiram identificar contornos e relações de oclusão que as caixas retangulares não capturam, aspecto explorado pelo algoritmo de detecção de sobreposição proposto neste trabalho.

Os exemplos visuais evidenciaram dois desafios recorrentes: reflexos na bandeja metálica, que reduzem contraste, e alta similaridade morfológica entre tesouras, fatores que impactaram a confiança das detecções. Ainda assim, a segmentação mostrou-se mais resiliente em cenários de sobreposição direta, indicando que vale o *trade-off* quando a separação precisa entre instrumentos é prioritária.

Trabalhos futuros incluem ampliar o conjunto de testes com variação de iluminação e fundos coloridos, explorar técnicas de supressão de reflexos e avaliar arquiteturas baseadas em Transformadores (p.ex. *Mask2Former*) para segmentação mais robusta. Também, vale reforçar, que o algoritmo de detecção de oclusão se mostrou uma solução interessante para avaliação de quais instrumentos deve-se retirar primeiro de pilhas por um robô autônomo, validando-o em ambiente real de esterilização para medir o impacto na taxa de erros de preensão e no tempo total de montagem dos kits cirúrgicos.

6 CONCLUSÃO

Este trabalho partiu do desafio de reconhecer e organizar instrumentos cirúrgicos em bandejas inevitavelmente desordenadas ao longo de um procedimento, problema já identificado no projeto Instrubot e em estudos anteriores de Lavado (2018), mas ainda sem solução suficientemente rápida e precisa para uso clínico em tempo real. A partir dessa motivação, definiu-se como objetivo avaliar se a combinação de máscaras segmentadas produzidas com o Segment Anything e as melhorias arquiteturais da família YOLOv11 permitiriam simplificar o pipeline de detecção, eliminar redes auxiliares e, ao mesmo tempo, elevar o desempenho frente à linha de base com YOLOv3.

Todas as imagens do conjunto original foram reanotadas no CVAT com apoio do SAM, resultando em um dataset compatível tanto com detecção por caixas quanto com segmentação instancial. Sobre esse material treinaram-se três pipelines contrastantes: YOLOv3 (baseline), YOLOv11 com *bounding boxes* e YOLOv11 com máscaras. A comparação direta mostrou ganhos expressivos: o YOLOv11-BBox obteve mAP@0,50 de $97,9\%$, revocação de $94,5\%$ e $8,45$ imagens/s, superando a linha de base em precisão e oferecendo velocidade mais de dez vezes maior . Já a variante YOLOv11-Seg manteve mAP elevado ($96,4\%$) e revocação de $93,9\%$, oferecendo contornos fiéis que se revelaram valiosos para raciocinar sobre sobreposições.

O uso de máscaras possibilitou ainda substituir quatro redes auxiliares de inferência de oclusão por um algoritmo analítico simples, baseado em contagem de blobs e proporção de área visível; primeiramente o algoritmo identificou 75 imagens com interseção de máscaras, e em seguida acertou qual o instrumento está no topo da pilha em 80% dos casos, demonstrando viabilidade prática sem custo adicional de inferência. Dessa forma, o presente estudo contribui com: (i) um pipeline unificado mais leve e rápido; (ii) um dataset reanotado de acesso público que abre caminho a pesquisas futuras em segmentação cirúrgica; e (iii) evidência quantitativa de que versões recentes da YOLO superam a configuração clássica mesmo quando treinadas apenas com caixas.

Ainda assim, persistem limitações. O Dataset possui imagens com iluminação fraca que dificultaram a detecção por reflexão com a bandeja metálica, sem variações cromáticas que são presentes no campo operatório real e com pouca variabilidade de instrumentos com geometrias semelhantes, fatores que afetaram sobretudo a classe Tesoura Reta e também representam o padrão em ambientes hospitalares. Também não foi realizada tunagem sistemática de hiperparâmetros: adotaram-se valores padrão da Ultralytics, o que sugere margem adicional de melhoria. Pesquisas futuras devem, portanto, expandir o conjunto de teste a ambientes clínicos variados, explorar abordagens baseadas em Transformers — como o Mask2Former —, otimizar

hiper-parâmetros de forma dirigida e integrar o pipeline a um robô físico para medir o impacto na redução do tempo de ciclo de pick-and-place em aplicação real.

Em síntese, a adoção de segmentação densa e da arquitetura YOLOv11 elevou a precisão de detecção, multiplicou a velocidade de processamento e eliminou módulos auxiliares, aproximando o projeto de seu propósito maior: aumentar a segurança e a eficiência no bloco operatório por meio da manipulação autônoma de instrumentos cirúrgicos.

REFERÊNCIAS

BREIMAN, Leo. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. Citado na página 22.

CARION, Nicolas et al. End-to-end object detection with transformers. In: **European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2020. p. 213–229. Citado na página 25.

CARPINTERO, E. et al. Development of a robotic scrub nurse for the operating theatre. In: **2010 3rd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics**. [S.l.: s.n.], 2010. p. 504–509. Citado na página 27.

CHENG, Bowen; SCHWING, Alexander; KIRILLOV, Alexander. Masked-attention mask transformer for universal image segmentation. **arXiv preprint arXiv:2202.04150**, 2022. Citado na página 25.

CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995. Citado na página 22.

DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: IEEE. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.], 2005. p. 886–893. Citado na página 22.

DEOL, Akshay et al. Artificial intelligence model for automated surgical instrument detection and counting: an experimental proof-of-concept study. **Patient Safety in Surgery**, BioMed Central, v. 18, n. 1, p. 7, 2024. Disponível em: https://doi.org/10.1186/s13037-024-00397-5. Citado na página 29.

DOSOVITSKIY, Alexey et al. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020. Citado na página 25.

DROST, Bertram et al. Model globally, match locally: Efficient and robust 3d object recognition. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, 2010. Citado na página 22.

ESTEVA, A. Deep learning-enabled medical computer vision. **npj Digit. Med**, 2021. Citado na página 17.

EVERINGHAM, Mark et al. The pascal visual object classes (voc) challenge. **International Journal of Computer Vision**, v. 88, n. 2, p. 303–338, 2010. Citado 4 vezes nas páginas 18, 19, 21 e 39.

HE, Kaiming et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. Citado na página 24.

IBM Corporation. **Computer Vision**. 2021. https://www.ibm.com/think/topics/computer-vision. White-paper institucional. Citado na página 17.

KIRILLOV, Alexander et al. Segment anything. **arXiv preprint arXiv:2304.02643**, 2023. Citado 2 vezes nas páginas 26 e 27.

KLETZ, Sabrina et al. Identifying surgical instruments in laparoscopy using deep learning instance segmentation. In: **2019 International Conference on Content-Based Multimedia Indexing (CBMI)**. [S.l.: s.n.], 2019. p. 1–6. Citado na página 29.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. **NIPS**, 2012. Citado 2 vezes nas páginas 21 e 24.

LAVADO, Diana Martins. **Sorting Surgical Tools from a Cluttered Tray – Object Detection and Occlusion Reasoning**. Dissertação (Mestrado) — Instituto Superior Técnico, Universidade de Lisboa, 2018. Citado 14 vezes nas páginas 15, 21, 22, 28, 30, 31, 32, 35, 41, 42, 43, 44, 51 e 53.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 22.

LECUN, Yann et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado 2 vezes nas páginas 23 e 24.

LIN, Tsung-Yi; MAIRE, Michael; BELONGIE, Serge et al. Microsoft COCO: Common objects in context. **ECCV**, p. 740–755, 2014. Citado 3 vezes nas páginas 19, 21 e 39.

LOWE, David G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, n. 2, p. 91–110, 2004. Citado na página 22.

NAKAMOTO, Pat. **Deep Learning Explained to Your Granny**. 2021. https://medium.com/@patnakamoto/deep-learning-explained-to-your-granny-abc123. Acesso em: 25 maio 2025. Citado na página 23.

NIELSEN, Michael. **Neural Networks and Deep Learning**. [s.n.], 2015. Acesso em: 25 maio 2025. Disponível em: http://neuralnetworksanddeeplearning.com/. Citado na página 23.

PARK, Alice. **Google's AI Bested Doctors in Detecting Breast Cancer in Mammograms**. 2020. https://time.com/5754183/google-ai-mammograms-breast-cancer/. Time Magazine, 1 jan. 2020. Acesso em: 25 mai. 2025. Citado na página 17.

REDMON, Joseph et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 779–788. Citado 3 vezes nas páginas 18, 24 e 25.

REDMON, Joseph; FARHADI, Ali. YOLOv3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018. Citado na página 35.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, American Psychological Association, v. 65, n. 6, p. 386–408, 1958. Citado na página 23.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. Citado na página 23.

SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. **Neural Networks**, Elsevier, v. 61, p. 85–117, 2015. Citado na página 23.

SHARMA, Aditya. **Mean Average Precision (mAP) Using the COCO Evaluator**. 2022. Disponível em: https://pyimagesearch.com/2022/05/02/mean-average-precision-map-using-the-coco-evaluator/. Citado 3 vezes nas páginas 18, 19 e 20.

SIMONYAN, Karen; ZISSERMAN, Andrew. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. [S.l.], 2014. Citado na página 24.

SuperAnnotate. **Image segmentation detailed overview**. 2022. https://www.superannotate.com/blog/image-segmentation-for-machine-learning#what-is-image-segmentation. Acesso em: 25 maio 2025. Citado 2 vezes nas páginas 20 e 21.

TAN, Huan et al. An integrated vision-based robotic manipulation system for sorting surgical tools. In: **2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)**. [S.l.: s.n.], 2015. p. 1–6. Citado na página 28.

ULTRALYTICS. **YOLOv8 – Model Documentation**. 2023. https://docs.ultralytics.com/models/yolov8/. Anchor-free split head; acessado em 03 jul 2025. Citado na página 25.

ULTRALYTICS. **YOLOv11 - Vision AI for the future**. 2024. https://github.com/ultralytics/ultralytics. Acesso em: 1 jun. 2025. Citado 2 vezes nas páginas 25 e 33.

WOLFEWICZ, A. **Deep learning vs. machine learning - What's the difference?** 2020. https://medium.com/levity/deep-learning-vs-machine-learning-whats-the-difference-e367803bb96d. Acesso em: 25 maio 2025. Citado na página 23.

APÊNDICE A – ALGORITMO DE DETECÇÃO DE OCLUSÃO

```
#!/usr/bin/env python3

from __future__ import annotations

import json
from collections import defaultdict, deque
from pathlib import Path
from typing import Dict, List

import cv2
import numpy as np
from tqdm import tqdm


# --------------
COCO_JSON   = Path("experiments/C_yolo11_seg/infer/
   segmentations_test.json")
OUTPUT_JSON = Path("experiments/C_yolo11_seg/occlusion/
   occlusion_results.json")
IMAGES_ROOT = Path("data/processed_seg/images/test")
OUT_DIR     = Path("experiments/C_yolo11_seg/occlusion/
   top_objects_only")
# --------------

# Kernels morfologicos
_DIL_KERNEL  = np.ones((5, 5), np.uint8)    # fecha gaps de ate 2 px
_EROS_KERNEL = np.ones((3, 3), np.uint8)    # quebra filetes de 1 px

# --------------
# Utilidades de mascara e grafo
# --------------
def polygon_to_mask_raw(poly: List[float], h: int, w: int) -> np.
   ndarray:
    """Poligono flatten -> mascara uint8 (0/1) sem pos-processo."""
    pts = np.array(poly, dtype=np.int32).reshape(-1, 2)
```

```
    m = np.zeros((h, w), dtype=np.uint8)
    cv2.fillPoly(m, [pts], 1)
    return m

def connected_components_count(mask_bool: np.ndarray) -> int:
    n, _ = cv2.connectedComponents(mask_bool.astype(np.uint8))
    return n - 1

def topo_sort(graph: Dict[int, set]) -> List[int]:
    indeg = {u: 0 for u in graph}
    for u, nbrs in graph.items():
        for v in nbrs:
            indeg[v] += 1
    q = deque([u for u, d in indeg.items() if d == 0])
    order: List[int] = []
    while q:
        u = q.popleft()
        order.append(u)
        for v in graph[u]:
            indeg[v] -= 1
            if indeg[v] == 0:
                q.append(v)
    return order  # bottom->top

def decide_occlusion(mask_dil, mask_ori, areas, ccs):
    """Retorna (grafo bottom->top, lista top->bottom)."""
    n = len(mask_ori)
    below = {i: set() for i in range(n)}
    for i in range(n):
        for j in range(i + 1, n):
            if not np.logical_and(mask_dil[i], mask_dil[j]).any():
                continue
            # Regra 0 - multi-componentes nunca eh topo
            if ccs[i] > 1 and ccs[j] == 1:
                below[i].add(j); continue
            if ccs[j] > 1 and ccs[i] == 1:
                below[j].add(i); continue
```

```
            # Regra 1 - fracao visivel
            inter = np.logical_and(mask_ori[i], mask_ori[j]).sum()
            r_i = inter / areas[i]
            r_j = inter / areas[j]
            if r_i < r_j:
                below[i].add(j)
            elif r_j < r_i:
                below[j].add(i)
            else:                           # empate -> mais partes fica
                embaixo
                if ccs[i] > ccs[j]:
                    below[i].add(j)
                elif ccs[j] > ccs[i]:
                    below[j].add(i)
    order_tb = list(reversed(topo_sort(below)))
    return below, order_tb

# --------------
# COCO helpers
# --------------
def load_coco(path: Path):
    with open(path, "r") as f:
        return json.load(f)

def build_index(coco: dict):
    anns_by_img = defaultdict(list)
    for ann in coco["annotations"]:
        anns_by_img[ann["image_id"]].append(ann)
    images_dict = {img["id"]: img for img in coco["images"]}
    anns_by_id  = {ann["id"]: ann for ann in coco["annotations"]}
    return anns_by_img, images_dict, anns_by_id

# --------------
# Analise principal
# --------------
def analyse_occlusions(anns_by_img, images_dict):
    results = []
```

```
    for image_id, anns in tqdm(anns_by_img.items(), desc="
       Analisando oclusoes"):
        h, w = images_dict[image_id]["height"], images_dict[
           image_id]["width"]
        mask_ori, mask_dil, areas, ccs, ann_ids = [], [], [], [],
           []
        for ann in anns:
            m_raw  = polygon_to_mask_raw(ann["segmentation"][0], h,
                w)
            m_ori  = cv2.erode(m_raw, _EROS_KERNEL, 1).astype(bool)
            m_dil  = cv2.dilate(m_raw, _DIL_KERNEL, 1).astype(bool)
            mask_ori.append(m_ori); mask_dil.append(m_dil)
            areas.append(int(m_ori.sum()))
            ccs.append(connected_components_count(m_ori))
            ann_ids.append(ann["id"])

        below, order_tb = decide_occlusion(mask_dil, mask_ori,
           areas, ccs)
        relations  = [{"top": int(ann_ids[j]), "bottom": int(
           ann_ids[i])}
                      for i, tops in below.items() for j in tops]
        components = {int(ann_ids[k]): int(ccs[k]) for k in range(
           len(ann_ids))}

        results.append({
            "image_id": int(image_id),
            "file_name": images_dict[image_id]["file_name"],
            "order_top_to_bottom": [int(ann_ids[k]) for k in
               order_tb],
            "relations": relations,
            "components": components,
        })
    return results

# --------------
# Overlay + export
# --------------
```

```
def draw_top_object(src: Path, ann_top: dict, h: int, w: int, dst:
    Path):
    img = cv2.imread(str(src))
    if img is None:
        print(f"  Falha ao ler {src}"); return
    m_raw = polygon_to_mask_raw(ann_top["segmentation"][0], h, w)
    m_ori = cv2.erode(m_raw, _EROS_KERNEL, 1).astype(bool)
    overlay = img.copy();  overlay[m_ori] = (0, 255, 0)
    cv2.addWeighted(overlay, 0.5, img, 0.5, 0, img)
    cnts, _ = cv2.findContours(m_ori.astype(np.uint8), cv2.
        RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if cnts:
        x, y, w_box, _ = cv2.boundingRect(cnts[0])
        cv2.putText(img, "TOP OBJECT", (x + w_box + 10, max(y - 10,
            20)),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
    dst.parent.mkdir(parents=True, exist_ok=True)
    cv2.imwrite(str(dst), img)

def export_outputs(occl_results, anns_by_id, images_dict):
    # JSON
    OUTPUT_JSON.parent.mkdir(parents=True, exist_ok=True)
    with open(OUTPUT_JSON, "w") as f:
        json.dump(occl_results, f, indent=2)
    print(f" JSON salvo em: {OUTPUT_JSON}")

    # Imagens
    OUT_DIR.mkdir(parents=True, exist_ok=True)
    n_exp = 0
    for rec in occl_results:
        if not rec["relations"]:
            continue
        top_id = rec["order_top_to_bottom"][0] if rec["
            order_top_to_bottom"] else None
        if top_id is None:
            continue
        ann_top  = anns_by_id[top_id]
```

```
        img_info = images_dict[rec["image_id"]]
        draw_top_object(IMAGES_ROOT / img_info["file_name"],
           ann_top,
                        img_info["height"], img_info["width"],
                        OUT_DIR / img_info["file_name"])
        n_exp += 1
    print(f"{n_exp} imagens exportadas em '{OUT_DIR}'.")

# --------------
# Main
# --------------
def main():
    coco = load_coco(COCO_JSON)
    anns_by_img, images_dict, anns_by_id = build_index(coco)
    occl_results = analyse_occlusions(anns_by_img, images_dict)
    export_outputs(occl_results, anns_by_id, images_dict)

if __name__ == "__main__":
    main()
```

Listing A.1 – Script occlusion infer