Universidade Politécnica
Castelo Branco
Escola Superior de Tecnologia

Estruturação de Roteiros de Fabrico Suportada em Técnicas de *Machine Learning*

Hugo Miguel Fé Santos

Nº 20234477

Orientadores

Professor Doutor Pedro Miguel Baptista Torres

Eng.º Rui M. C. Santos

Trabalho de Projeto apresentado à Escola Superior de Tecnologia da Universidade Politécnica de Castelo Branco, desenvolvido em parceria com a empresa Durit Metalurgia do Tungsténio, Lda, para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em Engenharia e Gestão Industrial, realizada sob a orientação científica do Professor Doutor Pedro Miguel Baptista Torres, da Universidade Politécnica de Castelo Branco e coorientação do Eng.º Rui Santos da Durit.

Setembro 2026

Composição do júri

Presidente do júri

Nuno Octávio Garcia Fernandes; Professor Coordenador.

Escola Superior de Tecnologia da Universidade Politécnica de Castelo Branco

Arguente

Armando Lopes Ramalho; Professor Coordenador;

Escola Superior de Tecnologia da Universidade Politécnica de Castelo Branco

Orientador(es):

Pedro Miguel Baptista Torres; Professor Adjunto;

Escola Superior de Tecnologia da Universidade Politécnica de Castelo Branco

Rui M. C. Santos;

Diretor de Engenharia e Desenvolvimento de Produtos da Durit

Agradecimentos

Ao concluir este trabalho, quero deixar o meu agradecimento a todos os que, de alguma forma, contribuíram para que ele fosse possível.

À Universidade Politécnica de Castelo Branco (UPCB), pela oportunidade de frequentar esta licenciatura e pelas condições proporcionadas ao longo do meu percurso académico.

À Escola Superior de Tecnologia de Castelo Branco (ESTCB) e aos seus docentes, pelos conhecimentos transmitidos ao longo do curso de Engenharia e Gestão Industrial, que constituíram a base deste trabalho.

Ao meu orientador, Professor Doutor Pedro Miguel Baptista Torres, pela disponibilidade, pelo acompanhamento ao longo de todo o projeto e pelas sugestões que ajudaram a dar-lhe rumo e rigor.

Ao meu coorientador, Eng.º Rui Santos, Diretor de Engenharia e Desenvolvimento de Produtos da Durit, pela confiança depositada neste projeto, pela partilha do seu conhecimento técnico e por me ter dado acesso aos desenhos e ao contexto real que estiveram na base deste trabalho. Estendo este agradecimento à Durit Metalurgia do Tungsténio, Lda, pela parceria e pela abertura com que me recebeu.

Aos meus colegas de curso, pela entreajuda e companheirismo.

Por fim, e de forma muito especial, à minha família e aos meus amigos, pelo apoio incondicional, pela paciência e pelo incentivo constante, sobretudo nos momentos em que conciliar trabalho, estudo e projeto parecia mais difícil.

A todos, o meu sincero obrigado.

Resumo

A leitura e transcrição manual de cotas, tolerâncias geométricas e dimensionais (GD&T) e tabelas de metadados a partir de desenhos técnicos de engenharia é um processo moroso e propenso a erros, sobretudo quando aplicado a grandes volumes de desenhos exportados de sistemas CAD. Este projeto tem como objetivo o desenvolvimento de um pipeline automático para a extração desta informação a partir de desenhos técnicos em formato PDF, reduzindo a intervenção manual e aumentando a fiabilidade dos dados extraídos.

A solução desenvolvida combina deteção de objetos com *deep learning* e processamento de texto vetorial. Um modelo YOLOv8, treinado especificamente para este domínio, deteta as regiões de interesse em cada desenho — peças cilíndricas, outras peças, tabelas de metadados e legendas. Sobre essas regiões é aplicado um conjunto de regras geométricas e expressões regulares que reconstrói as cotas e respetivas tolerâncias, agregando corretamente valores nominais, ajustes e desvios, e que interpreta a estrutura das tabelas a partir da sobreposição espacial entre palavras e colunas. Os resultados são exportados em formato Excel, prontos a integrar nos processos produtivos.

Um aspeto central da abordagem seguida é a extração do texto diretamente da camada vetorial do PDF, através da biblioteca *PyMuPDF*, em vez do recurso a Reconhecimento Ótico de Caracteres (OCR). Uma vez que os desenhos processados são exportados diretamente de sistemas CAD — e não digitalizações —, o texto já existe de forma pesquisável dentro do próprio ficheiro, o que permite evitar os erros de reconhecimento e o custo computacional associados a motores de OCR. O trabalho inclui uma análise comparativa entre esta abordagem e as técnicas de OCR identificadas no estado da arte — desde motores clássicos como o *Tesseract*, a pipelines industriais como o *PaddleOCR*, até arquiteturas *Transformer* como o *TrOCR*, e trabalhos que combinam deteção de objetos com OCR no mesmo domínio de aplicação —, discutindo as vantagens da extração vetorial em precisão e eficiência computacional, bem como as suas limitações quando aplicada a desenhos digitalizados ou de qualidade inferior.

Os resultados obtidos demonstram que a combinação de deteção de objetos com extração vetorial de texto constitui uma alternativa viável e mais leve às abordagens baseadas em OCR ou em modelos de visão-linguagem, para o caso específico de desenhos técnicos em formato vetorial, contribuindo para a automatização do controlo de qualidade e da digitalização de processos de engenharia. Os resultados devem, contudo, ser lidos à luz de um constrangimento importante: o dataset utilizado (44 desenhos, 101 imagens anotadas) é ainda pequeno, sendo necessário aumentá-lo substancialmente para consolidar a generalização do modelo e para suportar, em trabalho futuro, a estruturação de roteiros de fabrico otimizados a partir da informação extraída.

Palavras chave

Extração automática de informação; Desenhos técnicos de engenharia; YOLOv8; OCR; Extração de texto vetorial de PDF

Abstract

Manually reading and transcribing dimensions, geometric and dimensional tolerances (GD&T), and metadata tables from engineering technical drawings is a slow and error-prone process, particularly when applied to large volumes of drawings exported from CAD systems. This project aims to develop an automated pipeline for extracting this information from technical drawings in PDF format, reducing manual intervention and increasing the reliability of the extracted data.

The developed solution combines deep learning-based object detection with vector text processing. A YOLOv8 model, trained specifically for this domain, detects the regions of interest in each drawing — cylindrical parts, other parts, metadata tables, and legends. Over these regions, a set of geometric rules and regular expressions reconstructs dimensions and their tolerances, correctly aggregating nominal values, fits, and deviations, and interprets table structure based on the spatial overlap between words and columns. The results are exported to Excel, ready to be integrated into production processes.

A central aspect of the approach is extracting text directly from the PDF's vector layer using the PyMuPDF library, instead of relying on Optical Character Recognition (OCR). Since the processed drawings are exported directly from CAD systems — rather than scanned — the text already exists in searchable form within the file itself, avoiding the recognition errors and computational cost associated with OCR engines. The work includes a comparative analysis between this approach and the OCR techniques identified in the state of the art — from classic engines such as Tesseract, to industrial pipelines such as PaddleOCR, to Transformer architectures such as TrOCR, and works that combine object detection with OCR for the same application domain — discussing the advantages of vector extraction in terms of accuracy and computational efficiency, as well as its limitations when applied to scanned or lower-quality drawings.

The results obtained show that combining object detection with vector text extraction constitutes a viable and lighter alternative to OCR-based or vision-language model-based approaches, for the specific case of vector-format technical drawings, contributing to the automation of quality control and the digitalization of engineering processes. These results should, however, be read in light of an important constraint: the dataset used (44 drawings, 101 annotated images) is still small, and must be substantially expanded to consolidate the model's generalization and to support, in future work, the structuring of optimized manufacturing routings from the extracted information.

Keywords

Automatic information extraction; Engineering technical drawings; YOLOv8; OCR; Vector PDF text extraction

Índice geral

1. INTRODUÇÃO 1

1.1 Enquadramento e motivação 1
1.2 Problema e objetivos 1
1.3 Abordagem e contribuições 2
1.4 Organização do documento 2

2. ESTADO DA ARTE 3

2.1 Deteção de objetos com *deep learning* 3
2.2 Extração de texto: OCR versus extração vetorial de PDF 3
2.3 Deteção e reconhecimento de estrutura de tabelas 4
2.4 Extração automática de informação de desenhos técnicos de engenharia 4
2.5 Síntese e posicionamento do trabalho 5

3. METODOLOGIA 7

3.1 Visão geral do pipeline 7
3.2 Dataset e anotação dos desenhos técnicos 8
3.3 Treino do modelo de deteção YOLOv8 9
3.4 Extração vetorial de texto com *PyMuPDF* 10
3.5 Reconstrução de cotas de peças cilíndricas 10
Classificação de spans em tipos de cota 10
Junção de símbolos e tolerâncias por proximidade geométrica 11
Filtro do envelope exterior 11
Redução às cotas principais 12
3.6 Deteção e extração de tabelas de metadados 12
3.7 Geração automática de tabela de metadados a partir da legenda 12
3.8 Exportação dos dados e formato de saída 13
3.9 Integração num pipeline único: extrair_cotas_tab.py 13
3.10 Síntese do capítulo 15

4. DESENVOLVIMENTO DO CÓDIGO: DA PROVA DE CONCEITO AO PIPELINE DE PRODUÇÃO 17

4.1 Metodologia iterativa e controlo de versões 17
4.2 Do extrator de tabelas v4.5 ao v6.0: de posição inicial a caixa delimitadora completa 17
4.3 Robustez e tratamento de casos-limite 17
4.4 Limitações conhecidas e trabalho por integrar 18

5. DIAGNÓSTICO DA DETEÇÃO YOLO: COBERTURA DA CLASSE PECA_C 19

5.1 Metodologia do diagnóstico 19
5.2 Resultados globais 19
5.3 Peças não-cilíndricas: o gate está correto, não é uma falha 20
5.4 Ausência total de deteção: os quatro casos a investigar 20
5.5 Casos-limite: peca_c presente, mas abaixo do limiar 21
5.6 Síntese e recomendações 21

6. VALIDAÇÃO EMPÍRICA: EXTRAÇÃO VETORIAL VS. OCR 23

6.1 Metodologia de comparação 23
6.2 Resultados 24
6.3 Análise granular: tipo de cota e orientação do texto 25
6.4 Fiabilidade da confiança do OCR como filtro de qualidade 26

6.5 Taxonomia dos erros observados 26
6.6 Efeito do filtro de sanidade do pipeline de produção 28
6.7 Discussão e comparação com a literatura 28
6.8 Limitações do estudo 29

7. RESULTADOS DO PIPELINE FINAL 31

7.1 Metodologia de avaliação 31
7.2 Resultados globais 31
Exemplos ilustrativos de desenhos processados 32
7.3 As 58 linhas de tabela por resolver: uma limitação a investigar 35
7.4 Discussão e próximos passos 35

8. CONSTRANGIMENTOS E TRABALHO FUTURO 37

8.1 O constrangimento do dataset 37
8.2 Consolidar e alargar a extração de informação 38
8.3 Da extração de informação a roteiros de fabrico otimizados 38
8.4 Síntese 39

9. CONCLUSÕES 41

REFERÊNCIAS 43

Índice de figuras

Figura 3.1 Visão geral do pipeline de extração automática de informação 8
Figura 3.2 Exemplo de deteção YOLOv8 sobre um desenho técnico real do dataset 9
Figura 3.3 Reconstrução de uma cota composta a partir de spans de texto independentes 11
Figura 6.1 Precisão, recall e tempo total de execução por fonte de texto 25
Figura 6.2 Exemplo de erro de OCR 27
Figura 7.1 Distribuição das linhas com categoria atribuída, por categoria de resultado 32
Figura 7.2 Peça cilíndrica com extração bem-sucedida 33
Figura 7.3 Desenho com tabela de componentes (Stückliste) 33
Figura 7.4 Caso-limite de deteção 34
Figura 7.5 Peça não-cilíndrica (peca_v) 34

Índice de símbolos

Ø, ⌀, ø, ∅	Diâmetro (variantes gráficas do mesmo símbolo, conforme a fonte do desenho de origem)
R	Raio
M	Rosca métrica (prefixo ISO, ex. M8×1,25)
±	Desvio de tolerância (limites superior/inferior em torno do valor nominal)
°	Grau (unidade de ângulo)
×	Multiplicação / fator (ex. em chanfros “1×45°” e em fatores de lentidão relativa)
%	Percentagem
Ra	Rugosidade superficial média (parâmetro normalizado)
A, B	Letras de referência (datum) em tolerâncias geométricas
≥, ≤	Maior ou igual a / menor ou igual a (usado nos limiares de confiança do modelo)
mm	Milímetro
s	Segundo (unidade de tempo)

Lista de tabelas

Tabela 5.1 Diagnóstico da deteção de peca_c nos 44 desenhos do lote, por categoria 20
Tabela 5.2 Casos-limite: confiança de peca_c abaixo do limiar de produção 21
Tabela 6.1 Precisão, recall e tempo de execução por motor de extração 24
Tabela 6.2 Recall por tipo de cota / orientação do texto 25
Tabela 6.3 Confiança OCR em leituras corretas vs. espúrias, por motor 26
Tabela 6.4 Taxonomia das leituras espúrias por motor..... 27
Tabela 6.5 Efeito do filtro plausivel_dimensao() na remoção de falsos positivos 28
Tabela 7.1 Distribuição das linhas geradas pelo pipeline unificado, por categoria..... 32

Lista de abreviaturas, siglas e acrónimos

ACM	Association for Computing Machinery (editora citada nas Referências)
ASME	American Society of Mechanical Engineers
CAD	Computer-Aided Design (desenho assistido por computador)
CIRP	International Academy for Production Engineering (revista citada nas Referências)
CRNN	Convolutional Recurrent Neural Network
CTC	Connectionist Temporal Classification
DBNet	Differentiable Binarization Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DETR	DEtection TRansformer
DIN	Deutsches Institut für Normung (instituto alemão de normalização)
DPI	Dots Per Inch (pontos por polegada)
ERP	Enterprise Resource Planning (sistema de gestão empresarial)
GD&T	Geometric Dimensioning and Tolerancing (tolerâncias geométricas e dimensionais)
GPU	Graphics Processing Unit (unidade de processamento gráfico)
HTML	HyperText Markup Language
IoU	Intersection over Union (interseção sobre união)
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
OBB	Oriented Bounding Box (caixa delimitadora orientada)
OCR	Optical Character Recognition (reconhecimento ótico de caracteres)
PDF	Portable Document Format
R-CNN	Region-based Convolutional Neural Network (e variantes Faster R-CNN, Mask R-CNN, Cascade R-CNN)
SAP	Sistema de gestão empresarial (ERP) mencionado a propósito de um código de referência
SVM	Support Vector Machine (máquina de vetores de suporte)
SVTR	Scene Text Recognition with a Single Visual model
SW	Schlüsselweite (termo alemão para entre-faces, usado em roscas/sextavados)
TrOCR	Transformer-based Optical Character Recognition
YOLO	You Only Look Once

1. Introdução

Este trabalho de projeto, desenvolvido em parceria com a Durit — Metalurgia do Tungsténio, Lda, aborda a extração automática de informação de desenhos técnicos de componentes em metal duro, como primeiro passo para a estruturação de roteiros de fabrico assistida por técnicas de *machine learning*. Este capítulo apresenta o enquadramento e a motivação do trabalho, o problema e os objetivos, a abordagem seguida e a organização do documento.

1.1 Enquadramento e motivação

Comentado [HM1]:

O presente projeto, surgiu inicialmente com o objetivo de contribuir para a estruturação de roteiros de fabrico através da aplicação de técnicas de *machine learning*. A definição destes roteiros, que corresponde à identificação da sequência de operações necessárias ao fabrico de um componente, é uma etapa importante no planeamento da produção e depende da correta interpretação das características técnicas de cada peça.

No entanto, após a análise das necessidades da empresa e em articulação com a equipa da Durit, concluiu-se que, antes de avançar para a estruturação automática dos roteiros de fabrico, seria necessário desenvolver um mecanismo capaz de extrair e organizar a informação presente nos desenhos técnicos. Assim, o âmbito do projeto foi direcionado para a extração automática de cotas e metadados de desenhos técnicos exportados de sistemas CAD, com especial enfoque em peças cilíndricas e em desenhos associados a clientes alemães.

Estes desenhos contêm informação essencial para o fabrico e controlo de qualidade, nomeadamente cotas, tolerâncias dimensionais e geométricas (GD&T), material, dureza, peso e referências de identificação da peça e do cliente. A recolha e transcrição manual destes dados é um processo moroso e suscetível a erros, dificultando o seu tratamento sistemático. A automatização desta tarefa permite reduzir o esforço manual, melhorar a consistência da informação recolhida e facilitar a sua utilização posterior.

Deste modo, a extração de cotas e metadados constitui uma primeira etapa para alcançar o objetivo inicialmente proposto. Ao disponibilizar a informação dos desenhos técnicos de forma estruturada e digital, pretende-se criar uma base que permita, em trabalho futuro, relacionar as características técnicas das peças com as operações, máquinas e ferramentas necessárias ao seu fabrico. Assim, o projeto responde a uma necessidade identificada pela Durit e estabelece as bases para a futura estruturação e otimização de roteiros de fabrico com recurso a técnicas de aprendizagem automática.

1.2 Problema e objetivos

O problema abordado é a extração automática de cotas, tolerâncias e metadados (por exemplo, material, dureza, peso, número de desenho e escala) de desenhos técnicos em formato PDF exportados de sistemas CAD, produzindo dados estruturados prontos a integrar nos processos produtivos. Estes desenhos colocam dificuldades específicas: cotas compostas por vários elementos de texto independentes (símbolo de diâmetro, valor nominal e desvios de tolerância), texto em diferentes orientações, tabelas de metadados e de componentes, e layouts distintos entre famílias de peças.

O objetivo geral é desenvolver um pipeline automático que reduza a intervenção manual e aumente a fiabilidade dos dados extraídos. Como objetivos específicos, pretende-se: (i) treinar um detetor YOLOv8 que localize as regiões de interesse dos desenhos (peças cilíndricas, outras peças, tabelas e legendas); (ii) extrair o texto diretamente da camada vetorial do PDF, evitando o recurso a OCR, e reconstruir cotas e tolerâncias através de regras geométricas e expressões regulares; (iii) interpretar as tabelas de metadados e exportar os resultados para Excel; (iv) comparar empiricamente a extração vetorial com motores de OCR; e (v) identificar os constrangimentos do trabalho e as direções de desenvolvimento futuro, em particular no sentido da estruturação de roteiros de fabrico. A estruturação de roteiros de fabrico que dá título ao projeto é, assim, o objetivo de enquadramento: o âmbito concretizado neste trabalho é a extração automática da informação dos desenhos, que constitui a sua condição prévia, sendo a passagem para roteiros proposta no capítulo 8.

1.3 Abordagem e contribuições

A abordagem combina deteção de objetos com *deep learning* e processamento de texto vetorial: o YOLOv8 delimita apenas as regiões de interesse, e o texto é sempre lido diretamente da camada vetorial do PDF através da biblioteca *PyMuPDF*. As principais contribuições são: um pipeline unificado de extração de cotas e de tabelas de metadados; um diagnóstico quantificado da cobertura da deteção; uma comparação empírica entre a extração vetorial e motores de OCR (nomeadamente *EasyOCR*, *PaddleOCR* e *docTR*) sobre o mesmo lote de desenhos; e a identificação, com dados, dos limites do estudo. O principal deles é a dimensão do dataset (44 desenhos e 101 imagens anotadas), que condiciona a generalização dos resultados e cujo alargamento é o requisito central do trabalho futuro.

1.4 Organização do documento

O documento está organizado em nove capítulos. Após esta introdução, o capítulo 2 revê o estado da arte em deteção de objetos, extração de texto, deteção de tabelas e extração de informação de desenhos técnicos. O capítulo 3 descreve a metodologia e o pipeline desenvolvido, e o capítulo 4 documenta a evolução do código, da prova de conceito ao pipeline de produção. O capítulo 5 diagnostica a cobertura da deteção YOLO, o capítulo 6 compara empiricamente a extração vetorial com motores de OCR e o capítulo 7 apresenta os resultados do pipeline final. O capítulo 8 discute os constrangimentos do trabalho e as direções de trabalho futuro e, por fim, o capítulo 9 apresenta as conclusões.

2. Estado da Arte

A extração automática de informação a partir de desenhos técnicos de engenharia — cotas, tolerâncias geométricas e dimensionais (GD&T), materiais e tabelas de metadados — tem vindo a ser abordada, nos últimos anos, através da combinação de técnicas de deteção de objetos baseadas em *deep learning* com métodos de processamento de texto e de imagem. Esta secção revê o estado da arte em três frentes complementares às técnicas utilizadas neste projeto — deteção de objetos, extração/reconhecimento de texto e deteção de tabelas — e analisa em detalhe os trabalhos mais próximos da abordagem aqui desenvolvida.

2.1 Deteção de objetos com *deep learning*

As arquiteturas de deteção de objetos dividem-se essencialmente em duas famílias: os detetores *two-stage*, como o Faster R-CNN, que primeiro geram propostas de regiões e só depois classificam cada uma, e os detetores *one-stage*, como a família YOLO (*You Only Look Once*), que preveem simultaneamente a localização e a classe dos objetos numa única passagem pela rede. Os primeiros tendem a ser mais precisos, mas mais lentos; os segundos oferecem inferência em tempo real com uma perda de precisão normalmente pequena, o que os torna preferíveis em aplicações práticas de processamento de grandes volumes de documentos, como é o caso deste projeto. O YOLO evoluiu ao longo de várias versões, desde o modelo original até ao YOLOv8, utilizado neste trabalho para detetar peças cilíndricas, outras peças, tabelas e legendas nos desenhos técnicos, e mais recentemente até variantes como o YOLOv11, incluindo a variante YOLOv11-OBB (*oriented bounding boxes*), capaz de detetar anotações rodadas — uma limitação que os detetores clássicos, orientados a caixas alinhadas com os eixos, não resolvem adequadamente [1], [2].

2.2 Extração de texto: OCR versus extração vetorial de PDF

Depois de detetada uma região de interesse, é necessário extrair o texto que nela consta. Quando o documento de origem é uma imagem digitalizada (scan), essa extração exige recurso a motores de OCR (*Optical Character Recognition*). Ao longo da última década, o OCR evoluiu de motores clássicos baseados em redes convolucionais e LSTM bidirecional com decodificação CTC — como o *Tesseract*, leve mas com fraco desempenho em layouts complexos — para pipelines especializados em duas etapas que combinam deteção de texto e reconhecimento (por exemplo, o *PaddleOCR*, que usa DBNet para deteção e CRNN/SVTR para reconhecimento), e, mais recentemente, para arquiteturas *Transformer* end-to-end como o *TrOCR*, que atingem desempenho estado da arte em texto manuscrito à custa de um custo computacional consideravelmente mais elevado [3]. No entanto, quando o documento de origem é um PDF vetorial exportado diretamente de um sistema CAD — como é o caso dos desenhos técnicos processados neste projeto, o texto já existe como camada vetorial pesquisável dentro do próprio ficheiro, pelo que pode ser extraído diretamente através de bibliotecas como o *PyMuPDF*, sem qualquer erro de reconhecimento e com um custo computacional muito inferior ao de qualquer motor de OCR. Esta distinção é relevante porque grande parte dos trabalhos publicados nesta área assume a existência de digitalizações e recorre por isso a OCR — por exemplo em [4], os autores combinam deteção YOLOv7 com reconhecimento de texto via *Tesseract*-OCR para interpretar desenhos técnicos 2D — enquanto a

abordagem seguida neste projeto explora a natureza vetorial dos ficheiros de origem para evitar essa etapa.

2.3 Deteção e reconhecimento de estrutura de tabelas

A deteção de tabelas em documentos e o reconhecimento da sua estrutura interna constituem um subcampo próprio dentro da análise automática de documentos. Em [5], num survey abrangente sobre o tema, os autores traçam a evolução destas técnicas desde métodos heurísticos baseados em linhas, espaçamento e palavras-chave, passando por classificadores de *machine learning* clássicos como o SVM, até aos métodos de *deep learning* atualmente dominantes — nomeadamente detetores baseados em região, como o *Faster R-CNN*, o *Mask R-CNN* e o *Cascade R-CNN*, detetores *one-stage* adaptados especificamente para tabelas (como o designado "*YOLO-table*"), e arquiteturas baseadas em *Transformer* como o DETR. Para o reconhecimento da estrutura interna das tabelas (linhas, colunas e células), os autores destacam redes totalmente convolucionais, redes neuronais em grafo e arquiteturas *encoder-decoder* com atenção capaz de mapear diretamente a imagem de uma tabela para uma representação HTML. Esta revisão sustenta teoricamente a escolha, também neste projeto, de um detetor *one-stage* (YOLOv8) para identificar as tabelas de metadados presentes nos desenhos técnicos, complementado por um conjunto de regras geométricas — baseadas na sobreposição horizontal entre palavras e colunas — para reconstruir a estrutura da tabela a partir do texto extraído.

2.4 Extração automática de informação de desenhos técnicos de engenharia

O problema mais próximo do abordado neste projeto — a extração automática de cotas, tolerâncias e outras anotações de desenhos técnicos de engenharia — tem sido objeto de trabalhos crescentes nos últimos anos, refletindo o interesse da indústria em digitalizar processos ainda largamente manuais de controlo de qualidade e de tolerância assistida por computador. Em [6], os autores foram dos primeiros a propor uma solução direcionada a este problema com o *DigiEDraw*, um protótipo que recorre a *clustering* geométrico (DBSCAN) para agrupar e distinguir diferentes tipos de informação dimensional num desenho, obtendo uma taxa de *recall* superior a 88% num processo de manufatura real — sem, no entanto, recorrer a *deep learning*. Em [4], os autores avançaram para uma abordagem em camadas baseada em *deep learning*, combinando três modelos YOLOv7 (para vistas, grupos de anotação e anotações individuais) com reconhecimento de texto via *Tesseract*-OCR, alinhados com a norma ASME Y14.5-2018 de GD&T, atingindo entre 70% e 85% de precisão nas diferentes fases do pipeline, mas cobrindo apenas cinco categorias básicas de informação. Em [7], os autores focaram-se especificamente na extração de tolerâncias geométricas em desenhos legados, combinando processamento de imagem digital — filtragem e correção de inclinação — com *deep learning* para reconhecimento de caracteres, visando colmatar uma lacuna identificada nos pipelines modernos de Tolerância Auxiliada por Computador.

Mais recentemente, em [1], [2], os autores propuseram dois pipelines híbridos muito próximos entre si, ambos baseados no mesmo *dataset* de 1367 desenhos mecânicos anotados em nove categorias (GD&T, tolerâncias gerais, cotas, materiais, anotações técnicas, raios, rugosidade, roscas e blocos de título): na primeira versão, um detetor YOLOv11-OBB identifica as regiões de interesse e um transformer visão-linguagem

(*Donut*) interpreta o conteúdo de cada uma, produzindo uma saída estruturada em JSON com 94,77% de precisão em GD&T e uma taxa de alucinação de 5,23%; na segunda versão, os autores comparam o *Donut* com um segundo modelo visão-linguagem, o *Florence*-2, mantendo a arquitetura de deteção; também aqui o *Donut* é o melhor dos dois modelos, com 88,5% de precisão, 99,2% de recall e uma taxa de alucinação de 11,5% — valores globais, que não são diretamente comparáveis com os 94,77% obtidos apenas na categoria GD&T do primeiro trabalho. Ambos os trabalhos ilustram a tendência atual da área para pipelines híbridos que combinam um detetor de regiões rápido com um modelo de linguagem-visão pesado para a interpretação semântica do conteúdo — uma abordagem poderosa e genérica, mas que exige datasets anotados de grande dimensão e acarreta custos computacionais e riscos de alucinação que os métodos baseados em regras não têm.

2.5 Síntese e posicionamento do trabalho

Em síntese, a literatura revista mostra uma evolução clara de métodos geométricos e heurísticos [6] para pipelines híbridos que combinam deteção de objetos com *deep learning* e, mais recentemente, modelos de visão-linguagem de grande dimensão [1], [2], passando por soluções intermédias que ainda recorrem a OCR clássico [4], [7]. Este projeto posiciona-se nesta linha de investigação através de uma abordagem intencionalmente mais leve: em vez de aplicar OCR ou um modelo de linguagem-visão pesado sobre imagens renderizadas do desenho, o pipeline desenvolvido explora a natureza vetorial dos PDFs exportados de CAD para extrair o texto diretamente com o *PyMuPDF*, delegando ao YOLOv8 apenas a tarefa de deteção de regiões (peças, tabelas e legendas) e reservando a interpretação semântica — reconstrução de cotas, junção de tolerâncias, classificação de tipos de cota — a um conjunto de regras geométricas e expressões regulares. Esta opção reduz significativamente o custo computacional e elimina o risco de alucinação típico dos modelos generativos, ao preço de uma menor capacidade de generalização a desenhos digitalizados ou a formatos muito distintos dos utilizados no treino — uma limitação discutida nas secções 6.8 e 8.2 e retomada nas conclusões (capítulo 9).

3. Metodologia

Este capítulo descreve o método seguido no desenvolvimento do pipeline de extração automática de informação a partir de desenhos técnicos de engenharia. O projeto foi desenvolvido com base num conjunto de 44 ficheiros PDF de desenhos técnicos disponibilizados pela empresa Durit — Metalurgia do Tungsténio, exportados diretamente do sistema CAD utilizado internamente e, por essa razão, com uma camada de texto vetorial pesquisável (não digitalizações). O trabalho combina três componentes complementares: (i) um modelo de deteção de objetos YOLOv8, treinado especificamente para localizar peças, tabelas e legendas nos desenhos; (ii) a extração direta do texto vetorial do PDF através da biblioteca *PyMuPDF*, evitando qualquer recurso a OCR (cf. secção 2.2); e (iii) um conjunto de regras geométricas e expressões regulares que interpretam, classificam e reconstroem essa informação textual — cotas, tolerâncias, tabelas de metadados — de forma estruturada, exportando o resultado para ficheiros Excel. As secções seguintes descrevem sucessivamente a visão geral da arquitetura do pipeline (3.1), a construção e anotação do dataset (3.2), o treino do modelo de deteção (3.3), a extração vetorial de texto (3.4) e, em detalhe, os três scripts desenvolvidos para reconstrução de cotas (3.5), deteção e extração de tabelas (3.6) e geração automática de tabelas de metadados a partir da legenda (3.7), terminando com uma descrição do formato de exportação dos dados (3.8), a integração destes três scripts num pipeline único de produção (3.9) e uma síntese do capítulo (3.10).

3.1 Visão geral do pipeline

O pipeline desenvolvido processa cada PDF página a página, seguindo uma sequência comum aos três scripts descritos neste capítulo: primeiro, cada página é renderizada para uma imagem através de get_pixmap() do *PyMuPDF*, a uma resolução fixa (tipicamente entre 120 e 150 DPI, consoante o script), e submetida ao modelo YOLOv8, que devolve as caixas delimitadoras (*bounding boxes*) e a classe de cada objeto detetado — peça cilíndrica (peca_c), outra peça (peca_v), tabela (tabela) ou legenda (legenda), entre outras. As coordenadas destas deteções, calculadas no espaço de pixels da imagem renderizada, são reconvertidas para o espaço de coordenadas do PDF (em pontos) dividindo-as pelo fator de escala DPI/72, o que permite relacionar diretamente cada deteção com o texto vetorial nativo da página. A partir daqui os três scripts divergem consoante o objetivo: extração de cotas de peças cilíndricas, deteção e extração de tabelas de metadados já organizadas em grelha, ou reconstrução de uma tabela de metadados a partir de uma legenda em formato de lista rótulo/valor. Em qualquer dos casos, o texto nunca é lido a partir da imagem — é sempre obtido diretamente da camada vetorial do PDF, sendo o YOLOv8 usado apenas para delimitar as regiões relevantes da página (cf. discussão da secção 2.5 sobre esta opção de arquitetura). A Figura 3.1 ilustra esta sequência de etapas, desde o PDF de origem até à exportação para o template Excel de produção, evidenciando o ponto em que o pipeline se divide nos três scripts especializados (extrair_cotas_c.py, extrair_tabela.py e gerador_tabelas_yolo.py) e o ponto em que estes voltam a convergir no programa unificado extrair_cotas_tab.py (secção 3.9).

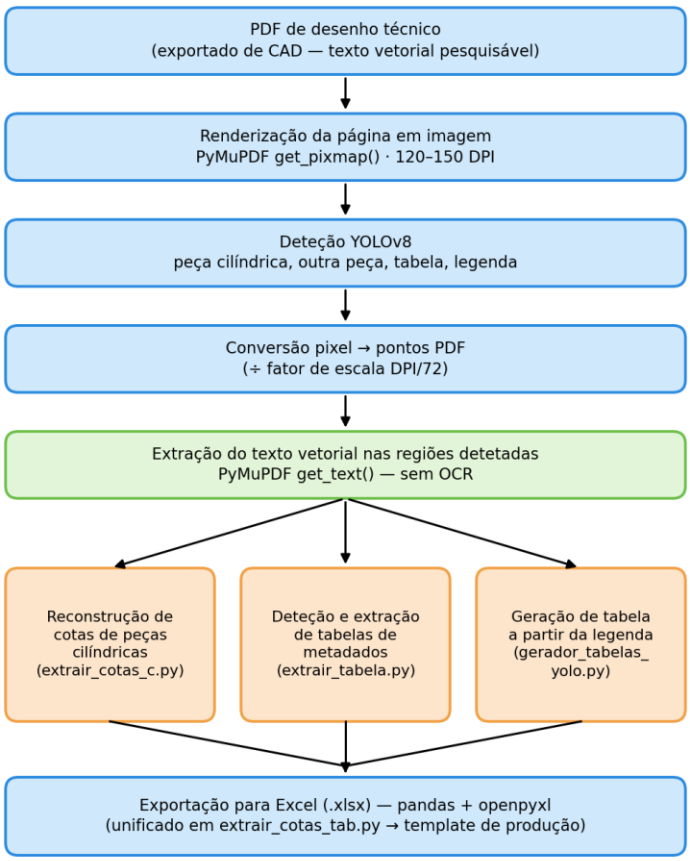


Figura 3.1 — Visão geral do pipeline de extração automática de informação (cf. secção 3.1). O YOLOv8 delimita apenas as regiões de interesse; o texto é sempre lido diretamente da camada vetorial do PDF, nunca da imagem renderizada, pelo que o OCR é evitado (cf. secção 2.2).

3.2 Dataset e anotação dos desenhos técnicos

O dataset do projeto é composto pelos 44 desenhos técnicos (ficheiros PDF) fornecidos pela Durit, cobrindo uma variedade de peças cilíndricas e não cilíndricas em metal duro, cada um com uma ou mais páginas — vista principal, vistas de detalhe ampliadas e, na maioria dos casos, uma legenda ou tabela de metadados no bloco de título. Para o treino do modelo de deteção, as páginas destes PDFs foram renderizadas como imagens e anotadas manualmente na ferramenta *Label Studio* [8], perfazendo um total de 101 imagens anotadas com caixas delimitadoras (*bounding boxes*) para as classes de interesse: peça cilíndrica (peca_c), outras peças (peca_v), uma classe genérica de peça (peca), vistas superior e lateral (vista_sup, vista_lat), tabela (tabela), legenda (legenda) e, dentro da legenda, os diferentes campos do bloco de título (material, grau/dureza, peso, número de cliente — sap, número de desenho do cliente — mod3d,

designação, escala e acabamento). Esta anotação manual constitui a etapa mais morosa do processo, mas é indispensável, uma vez que treina o modelo a distinguir estas classes especificamente no contexto dos desenhos técnicos da empresa — um domínio visual distinto de datasets genéricos de deteção de objetos. A Figura 3.2 ilustra o resultado desta deteção sobre um desenho técnico real do dataset, com as caixas delimitadoras das classes peça cilíndrica, tabela e legenda sobrepostas à página renderizada.

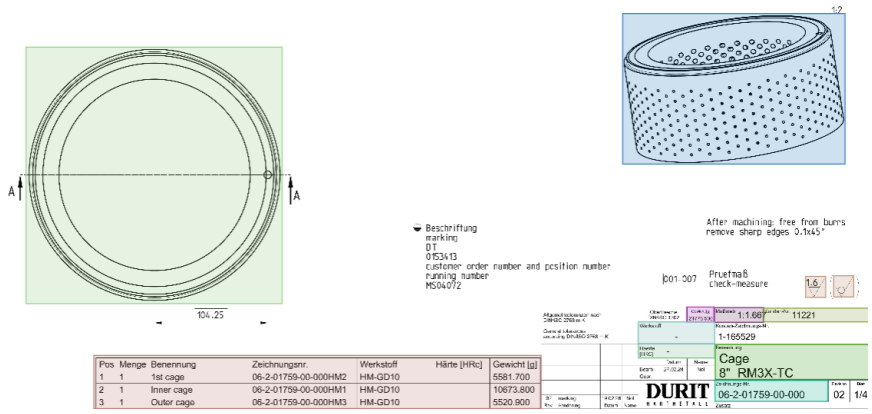


Figura 3.2 — *Exemplo de deteção YOLOv8 sobre um desenho técnico real do dataset (secção 3.2), com as bounding boxes das classes peça cilíndrica, tabela e legenda sobrepostas à página renderizada.*

Importa sublinhar que a dimensão deste dataset é o principal constrangimento do trabalho. Um conjunto de 44 desenhos, e apenas 101 imagens anotadas, é reduzido para treinar um detetor de objetos que se pretende robusto a diferentes famílias de peças, *layouts* de folha e estilos de cotagem. É provável que algumas classes — por exemplo, as peças não-cilíndricas, as tabelas de componentes ou certos campos da legenda — disponham de poucos exemplos, e desenhos da mesma família partilham frequentemente o mesmo *layout*, pelo que existe o risco de o desempenho medido ser otimista face a desenhos novos. Para comparação, o dataset usado por Khan et al. [1] contém 1367 desenhos anotados, cerca de trinta vezes mais do que o disponível neste projeto. São por isso necessários muito mais dados, tanto para treinar o modelo como para constituir conjuntos de validação e de teste independentes (cf. secção 6.8 e capítulo 8).

3.3 Treino do modelo de deteção YOLOv8

O treino do modelo foi realizado a partir das anotações exportadas do *Label Studio*, utilizando um *notebook Google Colab* — que tira partido do acesso gratuito a GPU para acelerar o treino —, partindo dos pesos pré-treinados do YOLOv8 [9] e afinando-os (*fine-tuning*) sobre as 101 imagens anotadas. O processo foi iterativo: uma primeira versão do modelo (designada treino1 no código, utilizada nos scripts de deteção de tabelas) foi complementada por uma segunda versão (treino2), com deteção refinada da classe peca_c, na sequência de uma análise dos erros de deteção mais frequentes — um processo de diagnóstico documentado em detalhe no capítulo 5. Em todos os *scripts* do pipeline, o limiar de confiança mínimo para aceitar uma deteção (parâmetro CONF) foi definido em 0,25 para a generalidade das classes, tendo sido reduzido para 0,05 apenas

no script de geração automática de legendas (secção 3.7), de forma a maximizar a sensibilidade de deteção das classes peca_c/peca_v nesse contexto específico, em que o custo de uma deteção adicional incorreta é baixo — é usada apenas para classificar o tipo de peça, não para delimitar uma região de exclusão de texto. No pipeline unificado (secção 3.9), que substitui estes *scripts* em produção, é usado um limiar único de 0,25 para todas as classes. Dada a dimensão reduzida do *dataset*, o desempenho do modelo de deteção é avaliado neste trabalho sobretudo de forma funcional — através do diagnóstico de cobertura da classe peca_c apresentado no capítulo 5 e do comportamento do pipeline nos capítulos 6 e 7 —, e não sobre um conjunto de teste independente.

3.4 Extração vetorial de texto com *PyMuPDF*

Como discutido na secção 2.2, os PDFs processados neste projeto são exportados diretamente do sistema CAD e contêm, por isso, uma camada de texto vetorial pesquisável, o que torna desnecessário o recurso a OCR. A extração de texto é feita através do método *get_text("dict")* da biblioteca *PyMuPDF* [10], que devolve a estrutura completa da página decomposta em blocos, linhas e spans de texto, cada um com a respetiva caixa delimitadora em coordenadas de pontos PDF e o vetor de direção da linha — a partir do qual se infere a orientação do texto (horizontal ou vertical), necessária para mais tarde distinguir comprimentos de diâmetros nas peças cilíndricas. Antes de classificar cada span de texto, o pipeline exclui duas categorias de região: (i) as áreas correspondentes às classes não dimensionais detetadas pelo YOLOv8 (bloco de título, tabela, contorno da própria peça), com uma pequena margem de segurança à volta de cada caixa; e (ii) toda a área abaixo da linha onde se situam os campos do bloco de título, calculada a partir da posição vertical mínima dessas deteções. Só os spans de texto que sobrevivem a este filtro — ou seja, os que se encontram na área de desenho, fora das caixas excluídas, onde as cotas são tipicamente colocadas à volta do contorno da peça — são considerados candidatos a cotas.

3.5 Reconstrução de cotas de peças cilíndricas

O *script* *extrair_cotas_c.py* é responsável por identificar as cotas, tolerâncias e outras anotações dimensionais nas páginas classificadas como peça cilíndrica. A deteção da classe peca_c funciona como porta de entrada (gate) de todo o processo: uma página só é processada se esta classe for detetada com confiança igual ou superior a 0,25; caso contrário, a página é ignorada, por se assumir que não contém uma peça cilíndrica. As subsecções seguintes descrevem as quatro etapas principais deste script.

Classificação de spans em tipos de cota

Cada span de texto retido pelo filtro da secção 3.4 é classificado, por um conjunto de expressões regulares, num de sete tipos: diâmetro (presença de um símbolo ⌀/Ø/ø/∅, ou um valor seguido de uma designação de ajuste ISO, como "20 j6"), rosca (padrão M seguido de número, por exemplo M8x1.25), raio (R seguido de número), chanfro (padrão "1x45°" ou ângulo isolado), entre-faces (padrão "SW" seguido de número, correspondente ao termo alemão Schlüsselweite), linear (um número simples) ou geométrico (um valor linear inferior a 0,05 mm sem desvios associados, tipicamente uma tolerância de forma ou posição junto de uma letra de referência/datum). São ainda filtrados, antes da classificação, tokens que correspondem a datas, códigos de revisão ou letras de datum isoladas (A, B, A-B), que não são cotas.

Junção de símbolos e tolerâncias por proximidade geométrica

Como o extrator de texto do *PyMuPDF* devolve muitas vezes o símbolo de diâmetro como um *span* separado do número a que pertence, o script junta-os sempre que os respetivos centros estão a menos de 25 pontos de distância em x e em y, formando por exemplo "⌀20" a partir de dois *spans* distintos — o mesmo procedimento é aplicado ao prefixo "SW". Da mesma forma, os desvios de tolerância (por exemplo, "+0.009", "-0.004" ou "±0.2") e os complementos de chanfro (por exemplo, "x45°") são associados à cota base geometricamente mais próxima, desde que a distância euclidiana entre os respetivos centros não exceda 55 pontos (parâmetro RAIO_TOLERANCIA). Isto permite reconstruir cotas compostas, como "20 j6 (+0.009/-0.004)", a partir de três ou quatro spans de texto originalmente independentes, que de outra forma apareceriam fragmentados e sem significado no resultado final. A Figura 3.3 ilustra este processo para a cota "⌀20 j6 (+0,009/-0,004)", reconstruída a partir de três spans independentes: o símbolo de diâmetro, o valor nominal com a respetiva classe de tolerância e os desvios superior e inferior.

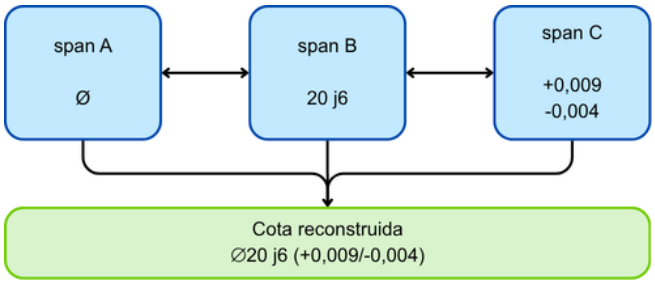


Figura 3.3 — Reconstrução de uma cota composta a partir de spans de texto independentes, por proximidade geométrica (cf. secção 3.5): o símbolo de diâmetro junta-se ao valor nominal quando os centros distam menos de 25 pontos; os desvios de tolerância associam-se à cota base num raio de 55 pontos (RAIO_TOLERANCIA).

Filtro do envelope exterior

Uma peça cilíndrica tem tipicamente cotas relativas ao seu contorno exterior (o comprimento total, os diferentes diâmetros externos) mas também cotas relativas a *features* internas — furos de centro, entalhes de desbaste, roscas — frequentemente representadas em vistas de detalhe ampliadas, a uma escala diferente da vista principal. Para separar estes dois grupos, o *script* não filtra pelo valor do texto (um valor de rugosidade como "3.2" é indistinguível, enquanto *string*, de uma cota de 3,2 mm), mas sim pela região da página onde a anotação se encontra. São identificadas âncoras de vista de detalhe — texto que indica uma escala de ampliação superior à escala principal do desenho (lida junto ao rótulo "*Maßstab*" do bloco de título) ou notas normalizadas específicas de detalhes construtivos (ISO 6411, para furos de centro; DIN 509, 332 e 76, para gargantas de desbaste e roscas) — e, à volta de cada âncora, é construída uma caixa delimitadora a partir da geometria vetorial da página (linhas e formas desenhadas, obtidas via get_*drawings*()) num raio de 230 pontos. Qualquer cota cujo centro caia dentro de uma destas caixas é excluída do envelope exterior. Como rede de segurança adicional, valores que coincidem com a série normalizada de rugosidade Ra (0,05, 0,1, 0,2, 0,4, 0,8, 1,6, 3,2, 6,3, 12,5 e 25) só são mantidos como cota se existir, na sua proximidade, uma ponta de seta de cota colinear — identificada geometricamente entre

os traçados preenchidos da página como uma forma fina e alongada, com comprimento entre 6 e 14 pontos e uma razão comprimento/largura superior a 1,7.

Redução às cotas principais

Por fim, quando ativada (parâmetro REDUZIR_PRINCIPAIS), esta etapa reduz a lista de cotas do envelope exterior às que caracterizam mais diretamente as dimensões principais da peça: o maior comprimento linear horizontal e, para cada diâmetro externo distinto, um único representante — agrupados pelo valor nominal (ignorando o ajuste ISO associado) ou mantendo cada ajuste como um diâmetro distinto, consoante a configuração escolhida. Raios, chanfros e tolerâncias geométricas são deixados fora desta redução, por não constituírem, em geral, dimensões estruturais da peça. O resultado final de cada PDF — cotas com tipo, valor, orientação, página e coordenadas — é exportado para um ficheiro Excel individual através do pandas e do *openpyxl*.

3.6 Deteção e extração de tabelas de metadados

Sempre que o bloco de metadados de um desenho já está organizado numa grelha tabular, este é processado pelo script *extrair_tabela.py*, que combina a deteção do YOLOv8 para a classe tabela com um mecanismo de recurso baseado em texto: quando o modelo não deteta nenhuma tabela na página, o script procura, entre as palavras extraídas com *get_text("words")*, linhas que contenham pelo menos três das cinco palavras-chave típicas do cabeçalho da tabela em alemão (*Pos, Menge, Benennung, Zeichnungsnr., Werkstoff*), construindo uma região candidata a partir da extensão dessas palavras. Dentro da região de tabela, detetada ou estimada, as palavras são agrupadas em linhas por proximidade do centro vertical — com uma tolerância derivada da altura média do texto na região — e a linha de cabeçalho é identificada como aquela que contém pelo menos quatro das colunas esperadas. A partir dessa linha são definidas as colunas reais da tabela, não pela posição horizontal inicial de cada palavra, como em versões anteriores do script, mas pela caixa delimitadora completa de cada cabeçalho, expandida até ao ponto médio com a coluna vizinha — a principal melhoria desta versão (v6.0) face a abordagens anteriores, tornando a atribuição de células mais robusta a pequenas variações de alinhamento entre desenhos diferentes. Cada palavra é depois atribuída à coluna com maior sobreposição horizontal com a sua caixa delimitadora, e concatenada às restantes palavras já atribuídas a essa mesma célula. A extração de linhas termina automaticamente ao encontrar texto de rodapé conhecido (por exemplo, menções a tolerâncias gerais ou à escala do desenho), e uma etapa final de limpeza remove linhas e colunas vazias, normaliza espaços em branco, elimina linhas residuais de texto técnico não tabular e reformata a coluna de peso para um valor numérico simples. O resultado é exportado para Excel, com uma folha de resumo — página, confiança da deteção, número de linhas e colunas de cada tabela encontrada — e uma folha por tabela estruturada.

3.7 Geração automática de tabela de metadados a partir da legenda

Nem todos os desenhos apresentam os metadados numa grelha tabular: em muitos casos, esta informação surge antes como uma legenda de pares rótulo/valor dispersos pelo bloco de título. Para estes casos, o script *gerador_tabelas_yolo.py* reconstrói os mesmos metadados a partir da legenda, complementando o *script* anterior — quando o

modelo deteta que a página já contém uma tabela estruturada, esta é ignorada aqui, para evitar duplicação de resultados. O *script* parte de um dicionário de rótulos esperados, tal como surgem no bloco de título dos desenhos, em alemão: *Werkstoff* (material), *Härte* (grau/dureza), *Gewicht* (peso), *Kunden-Nr.* (número de cliente), *Kunden-Zeichnungs-Nr.* (número de desenho do cliente) e *Benennung* (designação). Para cada rótulo, o algoritmo localiza primeiro o *span* de texto correspondente e, a partir da sua posição, procura o valor associado segundo três heurísticas espaciais, por ordem de prioridade: um valor imediatamente abaixo do rótulo e alinhado à esquerda (o caso mais comum no formato dos desenhos da Durit), um valor imediatamente acima (heurística de recurso) ou, por fim, um valor à direita, à mesma altura. Candidatos que correspondem a ruído conhecido — números de revisão, notações de escala como "2:1", datas no formato dd.mm.aa — são descartados, assim como valores atribuídos, por engano, a mais do que um campo em simultâneo, caso em que ambos são descartados, por se considerar indício de erro de correspondência. O script classifica ainda o tipo de peça de cada página (peça cilíndrica ou outra peça) diretamente a partir das classes peca_c/peca_v devolvidas pelo YOLOv8, consolidando todos os campos recolhidos por página numa única linha de um ficheiro Excel final, com uma ordem de colunas fixa.

3.8 Exportação dos dados e formato de saída

Em qualquer dos três *scripts*, o resultado final é persistido em ficheiros Excel (.xlsx) através das bibliotecas pandas e *openpyxl*, um formato escolhido por ser diretamente utilizável pela equipa de produção da empresa sem necessidade de ferramentas adicionais. Os scripts de extração de cotas e de deteção de tabelas produzem um ficheiro por PDF processado, nomeado com o mesmo nome do desenho de origem, o que facilita o cruzamento posterior dos resultados com os desenhos originais; o script de geração de legendas consolida, em vez disso, todas as páginas processadas num único ficheiro, com uma linha por página. Esta escolha de granularidade reflete a forma como cada tipo de resultado é tipicamente utilizado a jusante: as cotas e as tabelas são normalmente consultadas peça a peça, enquanto os metadados de legenda se prestam a uma análise agregada, em lote, sobre todo o *dataset*.

3.9 Integração num pipeline único: *extrair_cotas_tab.py*

Os três *scripts* descritos nas secções 3.5 a 3.7 foram desenvolvidos e validados de forma independente, cada um responsável por um tipo de conteúdo (cotas de peças cilíndricas, tabelas de metadados em grelha, e legendas em formato rótulo/valor) e cada um a exportar o seu próprio ficheiro Excel por PDF processado. Para uso em produção, estes três *scripts* foram unificados num único programa, *extrair_cotas_tab.py*, que os substitui na prática: percorre cada PDF uma única vez, corre o modelo YOLOv8 uma só vez por página — em vez das duas a três passagens exigidas quando cada script corria de forma independente — e escreve o resultado diretamente nas linhas de uma cópia do ficheiro Excel usado como modelo mestre pela equipa de produção da Durit (*exemplo_dados.xlsx*), guardada como *dados_preenchidos.xlsx*, em vez de criar uma pasta com um ficheiro por PDF. Não são gerados ficheiros intermédios: a única saída é o template preenchido, acompanhado de um ficheiro de avisos (*_avisos.txt*) com os casos que requerem confirmação manual. O capítulo 4 documenta em detalhe o processo de desenvolvimento que conduziu a esta consolidação.

Cada linha escrita no template é identificada pela coluna Designação, usada como chave para decidir se uma linha é nova ou deve atualizar uma já existente. Esta chave segue uma hierarquia de recurso: prioritariamente o campo mod3d (número de desenho do cliente, *Kunden-Zeichnungs-Nr.);* na sua ausência, o campo sap (número de cliente, *Kunden-Nr.*); e, como último recurso, a própria designação da peça (*Benennung*) combinada com o número de página. Este terceiro nível não identifica de forma inequívoca o desenho de origem, pelo que é sempre registado como aviso para confirmação manual — uma opção deliberada para não perder silenciosamente linhas de desenhos sem os campos habituais, à custa de gerar um pequeno número de casos que exigem revisão humana em vez de uma correspondência automática errada.

O *script* introduz ainda uma regra de classificação de material ausente dos *scripts* originais: quando o valor extraído do campo *Werkstoff* começa pelo prefixo HM- (*Hartmetall*, metal duro em alemão), a peça é classificada como MD (Metal Duro) na coluna Materiais do template, e o grau que resta depois de remover o prefixo é escrito em Grau - MD 1; nos restantes casos — tipicamente uma referência numérica sem prefixo, como um número de material DIN (por exemplo, 1.2344) — a peça é classificada como Aço e o valor bruto extraído passa diretamente para o grau. Esta regra foi definida a partir da convenção de nomenclatura observada nos desenhos da Durit e não consta na descrição da extração de metadados feita na secção 3.7. Numa versão anterior do script (1.11) as duas etiquetas estavam trocadas; o erro foi corrigido na versão 1.12 aqui descrita (cf. secção 7.2). Quanto às cotas, as dimensões principais de cada peça cilíndrica (secção 3.5) são escritas em três colunas do template: *MD-Comprimento* (maior comprimento horizontal), *MD-Diam Ext.* (maior diâmetro) e *MD-Diam Int.* (menor diâmetro, quando existe mais do que um); os diâmetros intermédios são registados no ficheiro de avisos e todas as cotas passam ainda pelo filtro de plausibilidade plausivel*_dimensao()* (secção 6.6).

A alteração mais significativa face aos três *scripts* originais diz respeito às tabelas de componentes (*Stückliste*): em vez de exportar cada tabela detetada para uma folha Excel própria, como fazia *extrair_tabela.py* (secção 3.6), cada linha da tabela passa a gerar diretamente a sua própria linha no template mestre — Designação a partir de *Zeichnungsnr.*, Descrição Hierarquia a partir de *Benennung* e Hierarquia Básica a partir de Pos. Como o desenho de detalhe de cada componente listado numa *Stückliste* está normalmente noutro PDF do mesmo lote, e não na página onde a tabela é encontrada, estas linhas são inicialmente escritas sem cota, assinaladas com o valor “Sem cota (tabela)”. Numa segunda passagem, executada depois de todos os PDFs processados, o script tenta localizar o PDF correspondente a cada uma destas designações — por correspondência exata do nome do ficheiro ou, na sua ausência, por normalização do texto (remoção de espaços, hífens e pontos) — e, quando o encontra, extrai dele as cotas do componente do mesmo modo que faria para uma página processada normalmente. Esta segunda passagem é a resposta direta ao problema de atribuição de cotas por peça em desenhos de montagem, identificado numa fase anterior do projeto como uma limitação não resolvida.

Uma vez que o YOLOv8 deteta por vezes a mesma tabela duas vezes, com caixas delimitadoras quase coincidentes, o script aplica uma deduplicação por IoU (*Intersection over Union*) com um limiar elevado (0,92): caixas cuja sobreposição excede este valor são tratadas como a mesma deteção e reduzidas a uma só, enquanto caixas apenas

parcialmente sobrepostas são mantidas em separado, uma vez que por vezes uma delas lê corretamente uma linha da tabela que a outra lê mal. Quando a mesma posição da tabela (Pos) acaba por ser lida duas vezes a partir de deteções diferentes, uma função de qualidade decide qual das duas leituras fica no template, penalizando números de desenho com espaços a mais — sinal típico de erro de segmentação de colunas — e privilegiando registos com os campos Werkstoff e Benennung preenchidos.

3.10 Síntese do capítulo

Em síntese, a metodologia seguida neste projeto assenta na complementaridade entre um modelo de deteção de objetos — YOLOv8, afinado em duas iterações de treino (secção 3.3) sobre 101 imagens anotadas manualmente a partir dos 44 desenhos técnicos fornecidos pela empresa — e três *scripts* de pós-processamento, depois unificados num pipeline único (secção 3.9), que exploram diretamente o texto vetorial nativo do PDF, sem qualquer etapa de OCR, para reconstruir cotas, tabelas de metadados e legendas estruturadas. Esta arquitetura reflete o posicionamento do trabalho já discutido na secção 2.5: em vez de delegar a interpretação semântica do conteúdo a um modelo pesado de visão-linguagem, a inteligência do pipeline reside sobretudo num conjunto interpretável de regras geométricas e expressões regulares, aplicadas sobre um texto extraído sem qualquer erro de reconhecimento. Os capítulos seguintes avaliam esta metodologia em três frentes complementares: o capítulo 5 diagnostica a cobertura da deteção YOLO da classe *peca_c*, que funciona como porta de entrada para todo o pipeline de extração de cotas; o capítulo 6 valida empiricamente a extração vetorial de texto, comparando-a com diferentes motores de OCR sobre o mesmo conjunto de desenhos; e o capítulo 7 avalia o pipeline unificado como um todo.

4. Desenvolvimento do Código: Da Prova de Conceito ao Pipeline de Produção

Este capítulo documenta a evolução do código do pipeline, complementando a descrição funcional do capítulo 3 com o processo que a ela conduziu: a forma como os três *scripts* iniciais foram sucessivamente refinados a partir de problemas concretos encontrados no dataset da Durit, e como acabaram por ser consolidados no pipeline único apresentado na secção 3.9. Esta perspetiva de desenvolvimento é sustentada diretamente pelas próprias versões do código — os comentários de cabeçalho de cada script documentam, versão a versão, o que mudou e porquê.

4.1 Metodologia iterativa e controlo de versões

Cada componente do pipeline foi desenvolvido como um *script* autónomo e versionado (*extrair_cotas_c.py*, *extrair_tabela.py*, *gerador_tabelas_yolo.py*), com cada nova versão preservada como ponto de recuperação em vez de substituir a anterior. Os números de versão refletem diretamente o esforço de afinação sobre dados reais: o extrator de tabelas chegou à sua sexta versão principal (v6.0) e o gerador de tabelas a partir da legenda à sétima (v7), cada uma delas corrigindo casos concretos observados nos 44 desenhos da Durit em vez de partir de uma especificação completa à priori. Esta opção por scripts únicos e autocontidos — sem divisão em módulos importados entre si — foi uma escolha deliberada, que privilegia a possibilidade de correr, ler e depurar cada etapa do pipeline isoladamente, à custa de alguma duplicação de código entre *scripts* (visível, por exemplo, na lógica de classificação de cotas, repetida quase verbatim entre *extrair_cotas_c.py* e o módulo de cotas do *script* unificado).

4.2 Do extrator de tabelas v4.5 ao v6.0: de posição inicial a caixa delimitadora completa

O próprio código documenta a motivação da revisão mais significativa do extrator de tabelas. Até à versão 4.5, cada palavra era atribuída à coluna cujo cabeçalho começava mais próximo da sua posição x inicial — uma abordagem que falhava sempre que pequenas variações de alinhamento entre desenhos diferentes (comuns em blocos de título preenchidos manualmente ou exportados por versões distintas do CAD) deslocavam ligeiramente o início de uma palavra para além da fronteira esperada da coluna. A versão 6.0 substituiu este critério pela sobreposição horizontal (overlap) entre a caixa delimitadora completa de cada palavra e a caixa de cada coluna, expandida até ao ponto médio com a coluna vizinha — um critério mais tolerante a pequenos desvios, por comparar áreas em vez de um único ponto. Esta mudança, referida no próprio cabeçalho do script como a razão de ser da versão 6.0, tornou a extração de tabelas substancialmente mais robusta à variabilidade real dos desenhos, à custa de nenhuma alteração percetível na arquitetura geral do *script*.

4.3 Robustez e tratamento de casos-limite

Ao longo das várias versões, o pipeline acumulou um conjunto de regras defensivas motivadas por casos-limite encontrados no dataset, e não antecipadas no desenho inicial: listas de exclusão (IGNORAR, IGNORAR_METADADOS, IGNORAR_COTAS) com termos alemães do bloco de título (*Datum, Bearb., Gepr., Maßstab*) e texto de rodapé (Allgemeintoleranzen, DIN/ISO 2768) que de outra forma seriam lidos como cotas ou

metadados válidos; expressões regulares que filtram números de revisão ("02"), datas ("10.02.17") e notações de escala ("2:1") antes de qualquer um destes ser mal classificado como uma cota linear; e o filtro de plausibilidade *plausivel_dimensao()* — discutido em detalhe na secção 6.6 — que rejeita valores acima de 1000 mm ou sequências de seis ou mais dígitos sem separador decimal, tipicamente números de cliente ou código SAP colados por engano a uma cota. A cada etapa do processamento por página está ainda associado um bloco try/except independente, de forma que uma falha isolada (por exemplo, uma página cujo render para imagem falha) não interrompe o processamento do resto do PDF nem do lote.

4.4 Limitações conhecidas e trabalho por integrar

Nem toda a lógica desenvolvida ao longo do projeto está integrada no pipeline único. A extração de cotas para peças não-cilíndricas (*peca_v*), desenvolvida num *script* próprio (*extrair_cotas_peca_v.py*) — que segue uma abordagem distinta, de *shortlist* de candidatas por eixo pensada para confirmação humana em vez de atribuição automática rígida — existe como função placeholder (*extrair_cotas_v*) no *script* unificado, devolvendo sempre valores em branco; o próprio código documenta esta limitação, apontando que a lógica completa "não estava disponível para integrar" no momento da consolidação. Na prática, isto significa que qualquer página classificada como *peca_v* é registada no template com o valor "Peça variada", sinalizando que as suas dimensões (largura, comprimento, altura) têm de ser preenchidas manualmente. Esta é a limitação mais relevante do pipeline de produção atual, e o candidato mais direto para trabalho futuro: integrar essa lógica de shortlist no script unificado permitiria estender automaticamente a extração de cotas às peças não-cilíndricas do *dataset*.

5. Diagnóstico da Deteção YOLO: Cobertura da Classe peca_c

Este capítulo investiga, em detalhe, a cobertura da deteção do YOLOv8 para a classe *peca_c*, que funciona como porta de entrada (gate) obrigatória para todo o pipeline de extração de cotas descrito na secção 3.5: uma página só é processada se esta classe for detetada com confiança igual ou superior a 0,25 — caso contrário, é ignorada e fica sem qualquer cota extraída, independentemente do motor de extração utilizado. Este diagnóstico foi motivado pela constatação, durante os testes do pipeline, de que uma parte do *dataset* ficava sistematicamente sem cotas, levantando a questão de saber se essa ausência se devia a peças genuinamente não cilíndricas, a falhas de deteção do modelo, ou a um limiar de confiança demasiado exigente. Para responder a esta questão, o modelo YOLOv8 foi executado sobre a totalidade dos 44 desenhos técnicos do *dataset* sem aplicar o gate, registando todas as classes e respetivas confianças detetadas em cada página, a dois níveis de limiar. As secções seguintes descrevem a metodologia deste diagnóstico, os resultados globais obtidos e a análise detalhada de cada uma das categorias de páginas identificadas, terminando com uma síntese e um conjunto de recomendações.

5.1 Metodologia do diagnóstico

Foi escrito um *script* auxiliar (*diagnostico_deteccao_yolo.py*) que corre o mesmo modelo YOLOv8 sobre cada página de cada PDF, mas sem aplicar o gate: em vez de descartar a página quando peca_c está ausente, regista todas as classes detetadas e a respetiva confiança, a dois níveis de limiar — o mesmo usado em produção (conf ≥ 0,25) e um limiar exploratório muito mais permissivo (conf ≥ 0,05), usado apenas para perceber se *peca*_c está "quase lá" ou completamente ausente. Isto permite distinguir, para cada PDF sem cotas, entre três situações possíveis: o modelo não detetou nada relacionado com uma peça, detetou uma peça, mas de outro tipo (não *peca_c*), ou detetou *peca_c* mas com confiança insuficiente.

5.2 Resultados globais

Dos 44 desenhos técnicos testados, 29 (65,9%) têm pelo menos uma página onde *peca_c* é detetado com confiança ≥0,25, e por isso atravessam o gate normalmente. Os restantes 15 (34,1%) ficam sem cotas — mas, esta ausência não tem uma única causa. A Tabela 5.1 desagrega estes 15 casos em três categorias com implicações muito diferentes.

Tabela 5.1 — Diagnóstico da deteção de peca_c nos 44 desenhos do lote, por categoria (inclui os 15 PDFs sem cotas detetadas).

Categoria	Nº PDFs	%	Exemplos
Passam o gate (peca_c ≥0,25 nalguma página)	29	65,9%	01100-00024-012-A2, 06-0-05838-00-001-02, 06-0-06280-00-000, 08-2-06607-15-000-01, 18-0-04754-00-000-02, entre outros
peca_v detetado — peça não-cilíndrica, fora do âmbito	9	20,5%	02-1-00344-00-000-01, 06-0-06660-00-000-03, 06-1-01849-00-000-01, 06-2-00346-00-000-06, 06-2-01868-00-000-01, 07-0-03814-00-000-05_, 15-2-00173-02-000-01, 17-0-02412-00-000-02, 28-0-01710-00-000-01
Nenhuma classe de peça detetada — a investigar	4	9,1%	04310-00031-000-B1_(FC37763_MD_REV_C), 06-0-06734-00-000-01, 10220-00002-000-A1_(R_20_T_800_A11), 13-0-00418-00-126-01
peca_c presente mas <0,25 — caso-limite	2	4,5%	18-0-03330-00-000-03 (0,10), 28-0-01702-00-000-01 (0,09)

A leitura mais importante desta tabela é que apenas 4 dos 44 desenhos (9,1%) representam uma falha de deteção genuinamente preocupante — casos em que o modelo YOLO não encontrou nenhuma evidência de peça, de qualquer tipo, mesmo ao limiar exploratório de 0,05. A maioria dos "zero cotas" (9 dos 15 casos, 20,5% do lote) corresponde a desenhos de peças não-cilíndricas (classe peca_v detetada com confiança elevada, entre 0,91 e 0,98), para os quais o gate está a funcionar exatamente como previsto: este pipeline foi concebido para extrair cotas de peças cilíndricas/torneadas, e corretamente exclui peças de outro tipo geométrico desta comparação. Somando estas duas categorias aos 29 PDFs processados normalmente, 86,4% do lote (38 em 44 PDFs) reflete o comportamento pretendido do pipeline — apenas os restantes 13,6% (6 PDFs, as duas últimas categorias da Tabela 5.1) merecem atenção adicional.

5.3 Peças não-cilíndricas: o gate está correto, não é uma falha

Nos 9 casos desta categoria, a classe *peca_v* surge sempre com confiança alta (0,91-0,98), tipicamente a par das classes de vista *vista_sup* e *vista_lat*, coerentes com uma peça prismática, maquinada por fresagem. Não há, nestes casos, qualquer ambiguidade: são desenhos de peças que não são de revolução, fora do âmbito da extração de cotas cilíndricas que este projeto compara. Alargar a extração a peças não-cilíndricas passa por integrar no pipeline o ramo paralelo, com regras de cotagem próprias, que já existe como script separado (*extrair_cotas_peca_v.py*, cf. secção 4.4) — e não por um ajuste ao gate atual.

5.4 Ausência total de deteção: os quatro casos a investigar

A categoria mais relevante para efeitos de melhoria do pipeline é a dos 4 desenhos em que nenhuma classe de peça (nem peca_c, nem peca_v, nem a classe genérica peca) foi detetada — em dois deles (10220-00002-000-A1 e 13-0-00418-00-126-01) esta ausência mantém-se mesmo ao limiar exploratório de 0,05, o que é atípico: uma página com um desenho normal de peça costuma gerar pelo menos alguma deteção fraca de

uma classe de peça. Estes quatro PDFs justificam inspeção visual direta para determinar se se trata de um problema de generalização do modelo a um layout de desenho fora da distribuição de treino, ou se, pelo contrário, estas páginas não correspondem de facto a um desenho técnico de peça (por exemplo, uma folha de rosto, um índice, ou uma vista muito distante/reduzida).

5.5 Casos-limite: peca_c presente, mas abaixo do limiar

Em 2 dos 15 PDFs sem cotas, peca_c é de facto detetado, mas com confiança muito baixa — visível apenas no teste exploratório a 0,05 (Tabela 5.2).

Tabela 5.2 — Casos-limite: confiança de peca_c abaixo do limiar de produção (0,25).

PDF	Confiança peca_c	Observação
18-0-03330-00-000-03	0,10	só visível a conf ≥0,05
28-0-01702-00-000-01	0,09	só visível a conf ≥0,05

Nestes dois casos, o modelo "vê" a peça, mas com pouca convicção. Não se recomenda baixar o limiar global de confiança (CONF) para os recuperar — o valor de 0,05 usado no gerador de legendas só é aceitável nesse contexto, em que a deteção serve apenas para classificar o tipo de peça (secção 3.3) —, porque, ao mesmo nível (0,05-0,10), surge também um volume considerável de deteções fracas de outras classes em praticamente todos os desenhos do lote — baixar o limiar globalmente arriscaria introduzir mais ruído do que ganho. Uma alternativa mais cirúrgica seria reforçar o conjunto de treino com mais exemplos deste estilo específico de desenho, ou aplicar um segundo passo de deteção com limiar reduzido apenas para a classe *peca_c* quando a primeira passagem não encontra nenhuma peça.

5.6 Síntese e recomendações

Em suma, este diagnóstico substitui uma leitura inicial e pouco específica ("2 PDFs em 5 ficam sem cotas, provavelmente por falha do gate") por uma caracterização quantificada e acionável: 86,4% do lote (38 dos 44 PDFs) reflete o comportamento pretendido do pipeline — PDFs processados normalmente ou corretamente excluídos por serem peça não-cilíndrica, estes últimos 9 dos 15 casos sem cotas —, e apenas 9,1% do lote (4 PDFs) corresponde a uma falha de deteção que justifica investigação e, possivelmente, reforço do conjunto de treino. Como próximos passos concretos: (1) inspecionar visualmente os 4 PDFs sem qualquer deteção de peça; (2) esclarecer o significado da classe peca genérica e decidir se deve ser incluída no gate; (3) considerar um segundo passo de deteção a limiar reduzido, restrito à classe peca_c, para os 2 casos-limite identificados. Vários destes pontos — em particular os 4 PDFs sem qualquer deteção e os 2 casos-limite — apontam para uma cobertura insuficiente do conjunto de treino, pelo que o reforço do dataset com mais exemplos é a medida transversal mais importante (cf. secção 8.1).

6. Validação Empírica: Extração Vetorial vs. OCR

As secções anteriores caracterizaram, com base na literatura, a distinção entre extração vetorial de PDF e OCR (secção 2.2) e os resultados reportados por trabalhos próximos deste projeto (secção 2.4). Este capítulo acrescenta a essa revisão uma validação empírica, própria deste trabalho, que aplica três motores de OCR de uso geral (*EasyOCR*, *PaddleOCR* e *docTR*) à mesma tarefa de extração de cotas em desenhos técnicos de peças cilíndricas, e compara os resultados obtidos com a extração vetorial utilizada no pipeline principal do projeto e com os valores reportados na literatura revista.

6.1 Metodologia de comparação

Foi desenvolvido um *script* de comparação que corre, sobre o mesmo lote de desenhos técnicos, cinco fontes de texto distintas: a extração vetorial de referência (lógica equivalente ao script *extrair_cotas_c.py*), a mesma extração acrescida do filtro de sanidade *plausivel_dimensao()* presente no pipeline de produção (*extrair_cotas_tab.py*), e os três motores de OCR, aplicados sobre uma imagem da página renderizada a partir do mesmo PDF. Em todos os casos, a deteção da peça cilíndrica (classe *peca_c*) e das zonas de exclusão da legenda foi feita com o mesmo modelo YOLOv8, garantindo que as diferenças observadas refletem apenas a fonte do texto, não a deteção geométrica.

A extração vetorial foi utilizada como referência (*ground truth*): como os desenhos são exportados diretamente de CAD e não digitalizados, o texto vetorial do PDF corresponde exatamente ao que foi desenhado, sem ambiguidade de leitura. Importa, no entanto, notar que a referência é a saída da extração vetorial sem filtro, e não uma lista de cotas verificada de forma independente: o texto é lido sem erro, mas a classificação de um span como cota pode estar errada. Foi o que aconteceu com um código SAP classificado como cota linear (secção 6.6), que ficou incluído nas 14 cotas de referência; face às 13 cotas reais, a extração de produção recupera todas. As métricas da extração vetorial medem, por isso, a concordância com a própria referência e não uma exatidão independente; o resultado informativo desta comparação são as métricas dos motores de OCR. Por servir de referência, a extração vetorial sem filtro não consta da Tabela 6.1. Cada cota extraída por cada motor foi reduzida ao seu valor nominal (o primeiro número presente na cadeia de caracteres, ignorando tolerâncias e sufixos — por exemplo, "Ø20 j6 (+0,009/-0,004)" → 20) e comparada com o conjunto de valores nominais da referência para o mesmo PDF. Foram calculadas três métricas por motor: recall (percentagem de cotas reais da referência que o motor conseguiu recuperar), precisão (percentagem das cotas devolvidas pelo motor que correspondem a uma cota real) e tempo total de execução. A precisão e o recall são definidos, respetivamente, pelas Equações (6.1) e (6.2) [11]:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{6.1}$$

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{6.2}$$

em que VP (verdadeiros positivos) é o número de cotas devolvidas pelo motor que correspondem a uma cota real da referência, FP (falsos positivos) o número de cotas devolvidas sem correspondência na referência — as leituras espúrias — e FN (falsos negativos) o número de cotas reais da referência que o motor não conseguiu recuperar. A

precisão mede, assim, a fiabilidade do que o motor devolve, enquanto o recall mede a sua capacidade de encontrar todas as cotas existentes no desenho.

6.2 Resultados

A Tabela 6.1 resume os resultados obtidos sobre um lote de teste de 5 PDFs, dos quais 3 continham cotas de peça cilíndrica detetáveis pelo modelo YOLO (num total de 14 cotas de referência).

Tabela 6.1 — Precisão, recall e tempo de execução por motor de extração (5 PDFs de teste, 14 cotas de referência).

Motor	Recall	Precisão	Tempo total (s)	Fator de lentidão (vs. vetorial)
Extração vetorial (produção, com filtro)	92,9%	100%	6,1	1×
docTR	21,4%	33,3%	51,6	≈8×
EasyOCR	28,6%	25,0%	294,0	≈48×
PaddleOCR	28,6%	40,0%	468,4	≈77×

O caso mais ilustrativo é o do desenho 06-0-05838-00-001-02.pdf, no qual a extração vetorial recupera corretamente as duas cotas presentes (um comprimento de 17 mm e um diâmetro de Ø50 mm, ambos com tolerância ±0,1 mm), enquanto nenhum dos três motores de OCR testados acerta em nenhuma delas, devolvendo antes valores sem correspondência no desenho (por exemplo, Ø3, 0,05 ou 62). No desenho mais denso do lote (06-0-06280-00-000.pdf, com sete diâmetros distintos), o EasyOCR e o PaddleOCR recuperam corretamente apenas 2 dos 7 valores reais e o docTR apenas 1, introduzindo cada motor, em simultâneo, entre 4 e 6 leituras espúrias — diâmetros que não existem no desenho. A Figura 6.1 representa graficamente os valores da Tabela 6.1, permitindo comparar lado a lado o recall, a precisão e o tempo total de execução de cada fonte de texto; o tempo é apresentado em escala logarítmica, dada a diferença de quase duas ordens de grandeza entre a extração vetorial e o motor de OCR mais lento.

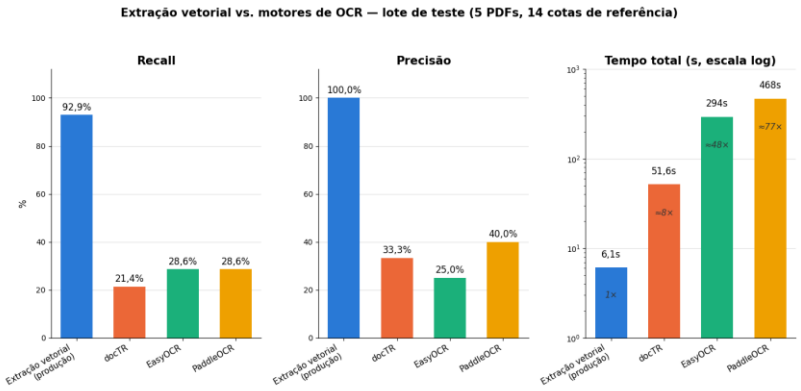


Figura 6.1 — Precisão, recall e tempo total de execução por fonte de texto, sobre o lote de teste de 5 PDFs e 14 cotas de referência (dados da Tabela 6.1, cf. secção 6.2). O eixo do tempo está em escala logarítmica; os fatores junto às barras indicam a lentidão relativa face à extração vetorial.

6.3 Análise granular: tipo de cota e orientação do texto

Os valores agregados da Tabela 6.1 escondem uma assimetria importante. Nas convenções de cotagem seguidas nestes desenhos, os comprimentos (tipo linear) surgem sempre em texto horizontal e os diâmetros (tipo diametro) surgem sempre em texto rodado 90° (vertical) junto à vista onde são cotados — ou seja, neste dataset, tipo de cota e orientação do texto coincidem univocamente. A Tabela 6.2 desagrega o *recall* de cada motor por esta distinção.

Tabela 6.2 — Recall por tipo de cota / orientação do texto (linear=horizontal, n=3; diâmetro=vertical, n=11).

Motor	Recall — linear / horizontal	Recall — diâmetro / vertical
EasyOCR	66,7%	18,2%
PaddleOCR	66,7%	18,2%
docTR	66,7%	9,1%

Os três motores recuperam corretamente dois dos três comprimentos horizontais (66,7%), mas caem para 9%-18% de recall nos onze diâmetros em texto vertical — uma queda de recall entre 3,7× e 7,3× consoante o motor. Este resultado confirma diretamente, com dados, a limitação já identificada na documentação dos scripts de OCR deste projeto: apesar das melhorias implementadas para lidar com texto rodado (rotation_info no *EasyOCR*, deteção de polígonos rodados com assume_straight_pages=False no *docTR*, cálculo de orientação pelo ângulo real do polígono de deteção no *PaddleOCR* e no *docTR*), o texto vertical continua a ser significativamente mais difícil de reconhecer corretamente do que o texto horizontal para estes três motores de uso geral — provavelmente por serem pré-treinados maioritariamente sobre texto de leitura natural (documentos, cenas), onde a orientação vertical é rara.

6.4 Fiabilidade da confiança do OCR como filtro de qualidade

Uma vez que os três motores devolvem um valor de confiança por leitura, colocou-se a questão de saber se esse valor poderia ser usado como filtro adicional para descartar leituras incorretas antes de as integrar no pipeline — de forma análoga ao papel que o filtro *plausivel_dimensao()* desempenha para a extração vetorial. A Tabela 6.3 compara a confiança média e mediana das leituras corretas com a das leituras espúrias, por motor.

Tabela 6.3 — Confiança OCR (média / mediana) em leituras corretas vs. espúrias, por motor.

Motor	Confiança — corretas	Confiança — espúrias	n (corretas / espúrias)
EasyOCR	0,97 / 0,99	0,76 / 0,79	4 / 12
PaddleOCR	0,99 / 1,00	0,91 / 0,96	4 / 6
docTR	1,00 / 1,00	0,80 / 0,96	3 / 6

A diferença entre as duas colunas é demasiado pequena para sustentar um filtro fiável: a confiança mediana das leituras espúrias do *PaddleOCR* (0,96) e do *docTR* (0,96) está muito próxima da das leituras corretas (1,00), e no caso do *PaddleOCR* as três leituras mais espúrias observadas neste teste (Ø9, Ø8 e Ø4, nenhuma correspondente a um diâmetro real do desenho) foram todas devolvidas com confiança máxima (1,00) — indistinguíveis, do ponto de vista do motor, de uma leitura correta. Isto sugere que a confiança reportada pelos motores de OCR reflete a certeza do reconhecimento dos carateres (o motor "leu" claramente algo), não a certeza de que esse algo é a cota correta — uma distinção relevante para qualquer tentativa futura de usar OCR em produção: o filtro de sanidade baseado em plausibilidade geométrica do valor (como o *plausivel_dimensao()* já usado na extração vetorial) parece ser, com base nestes dados, mais eficaz do que um limiar sobre a confiança do OCR para descartar leituras incorretas.

6.5 Taxonomia dos erros observados

Classificando manualmente as 24 leituras espúrias observadas nos três motores, distinguem-se dois tipos de erro com implicações práticas distintas: erros de arredondamento/truncamento, em que o motor lê corretamente a grandeza mas perde a última casa decimal (por exemplo, Ø33,188 → Ø33,18 no *EasyOCR*, → Ø30,09 no *PaddleOCR* quando a referência é Ø30,098, ou → Ø33 no *docTR*) — próximos do valor real e potencialmente úteis com uma tolerância de comparação menos estrita —, e erros sem correspondência plausível, em que o valor lido não tem relação identificável com nenhuma cota real do desenho (por exemplo, Ø512,5, Ø920, Ø825 ou Ø382, valores muito acima da escala física de peças deste tipo). A Tabela 6.4 resume esta distribuição.

Tabela 6.4 — Taxonomia das leituras espúrias por motor: quase-certas (truncamento) vs. sem correspondência plausível.

Motor	Quase-certas (truncamento)	Sem correspondência plausível	Exemplo mais extremo
EasyOCR	1	11	Ø920 (real mais próximo: Ø144 mm — dif. 539%)
PaddleOCR	1	5	Ø4 (real mais próximo: Ø12,5 mm — dif. 68%)
docTR	1	5	Ø188 (real mais próximo: Ø33,188 mm — dif. 466%)

É particularmente relevante notar que valores como Ø920, Ø825 e Ø382 mm (*EasyOCR*, no desenho 06-0-06600-00-000-03.pdf, cujas cotas reais não ultrapassam os 144 mm) excedem claramente a escala física plausível para este tipo de peça, mas ficariam abaixo do limiar de 1000 mm usado pelo filtro *plausivel_dimensao()* do pipeline de produção — pelo que esse filtro, tal como está definido atualmente, não os rejeitaria se fossem produzidos por um motor de OCR integrado no pipeline. Um refinamento possível, sugerido por esta análise, seria limitar a plausibilidade de cada cota não a um limiar absoluto único, mas a um múltiplo da maior cota já confirmada na mesma peça/página (por exemplo, rejeitar diâmetros superiores a 3-5× o comprimento principal detetado) — uma direção concreta para trabalho futuro caso a integração de OCR no pipeline de produção venha a ser considerada. A Figura 6.2 ilustra um destes casos no desenho 06-0-06600-00-000-03.pdf, colocando lado a lado a página processada, a cota real do desenho e a leitura espúria de Ø920 mm devolvida pelo EasyOCR para essa mesma cota.

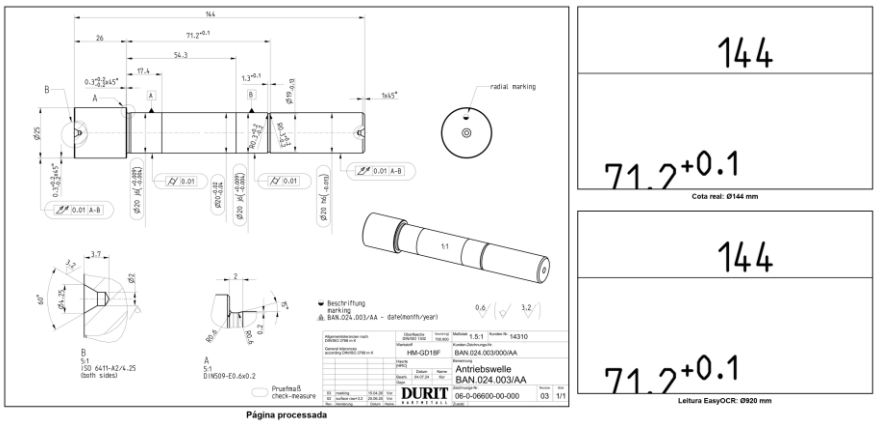


Figura 6.2 — Exemplo de erro de OCR (secção 6.5): a imagem da página processada, a cota real do desenho e a leitura espúria devolvida pelo motor, lado a lado.

6.6 Efeito do filtro de sanidade do pipeline de produção

A extração vetorial de referência e a lógica de cotas do pipeline de produção (*extrair_cotas_tab.py*) partilham exatamente o mesmo código de deteção, montagem de tolerâncias e filtro de envelope exterior; a única diferença é o filtro adicional plausivel_dimensao(), que rejeita valores acima de 1000 mm e sequências numéricas longas (seis ou mais dígitos) sem separador decimal — tipicamente um código de cliente ou SAP colado junto de uma cota. No lote de teste, este filtro foi acionado uma única vez (Tabela 6.5), removendo corretamente o valor "0208175000" (um código SAP) que a extração vetorial sem filtro classificava incorretamente como uma cota linear. Em todas as restantes 13 cotas do lote, os dois métodos produziram resultados idênticos.

Tabela 6.5 — Efeito do filtro plausivel_dimensao() na remoção de falsos positivos.

Fonte	Cotas totais	Falsos positivos (códigos mal classificados)
Extração vetorial (sem filtro)	14	1
Extração vetorial + filtro (produção)	13	0

6.7 Discussão e comparação com a literatura

Os resultados obtidos confirmam empiricamente, sobre este dataset, o posicionamento teórico apresentado na secção 2.2: uma vez que o texto do PDF é lido diretamente da camada vetorial, sem qualquer etapa de reconhecimento de padrões visuais, não existe, por definição, erro de reconhecimento de carateres, o que explica — em conjunto com o facto de a referência ser a própria extração vetorial (secção 6.1) — a precisão de 100% obtida neste teste, associada a um tempo de execução entre 8 e 77 vezes inferior ao dos motores de OCR — mesmo depois de estes terem sido otimizados (maior DPI de renderização, pré-processamento de imagem, deteção de texto rodado e reconhecedores mais precisos, como o parseq no *docTR*).

Vale a pena colocar estes valores lado a lado com os reportados na literatura revista na secção 2.4. Lin, Ting, Huang, Cheng, & Jong [4], num pipeline em camadas com três modelos YOLOv7 e reconhecimento de texto via *Tesseract*-OCR sobre desenhos técnicos 2D alinhados com a norma ASME Y14.5-2018, reportam entre 70% e 85% de precisão nas diferentes fases do seu pipeline — valores muito superiores aos 25%-40% de precisão obtidos neste teste pelos motores *EasyOCR*, *PaddleOCR* e *docTR*. A diferença é expectável e não invalida os motores aqui testados: o pipeline de Lin, Ting, Huang, Cheng, & Jong [4] foi treinado e afinado especificamente para aquele domínio e norma, enquanto os motores usados nesta comparação foram aplicados de forma genérica, sem re-treino do reconhecedor. Ainda assim, a comparação ilustra a distância prática entre uma solução de OCR bem afinada ao domínio e o uso genérico de motores de OCR, e reforça que mesmo o melhor cenário reportado na literatura para OCR aplicado a desenhos técnicos (85%) fica aquém dos 100% de precisão obtidos pela extração vetorial neste projeto.

Uma segunda comparação relevante é com o protótipo DigiEDraw de Scheibel, Mangler, & Rinderle-Ma [6], que recorre a clustering geométrico (DBSCAN) para agrupar informação dimensional sem qualquer componente de OCR ou *deep learning*, atingindo um recall superior a 88% num processo de manufatura real. O recall da extração vetorial

deste projeto não é diretamente comparável com esse valor, por ser medido face a uma referência derivada da própria extração (secção 6.1); a comparação informativa é a dos motores de OCR, cujo recall (21%-29%) fica muito abaixo desse patamar — um indício de que, tal como o DigiEDraw, abordagens que evitam o reconhecimento visual de carateres (seja por regras geométricas, seja por leitura direta da camada vetorial do PDF) tendem a ser mais robustas do que OCR genérico para a extração de informação dimensional densa e simbólica típica de desenhos técnicos.

Por fim, a elevada proporção de leituras espúrias produzidas pelos motores de OCR neste teste — entre 60% e 75% das cotas devolvidas não correspondem a nenhuma cota real do desenho — é qualitativamente análoga, ainda que não diretamente comparável em termos metodológicos, à taxa de alucinação reportada por Khan, Yong, Chen, Tan, Feng, & Moon [1] e Khan, Chen, Yong, Tan, Feng, & Moon [2] para os pipelines baseados em modelos de visão-linguagem: 5,23% [1] e 11,5% [2], em ambos os casos com o Donut, o melhor dos modelos avaliados. A diferença de cerca de uma ordem de grandeza entre estes valores não deve ser lida como uma desvantagem do OCR genérico face aos modelos de visão-linguagem — os dois tipos de modelo falham de formas distintas e sobre tarefas nem sempre equivalentes (reconhecimento de carateres isolados versus interpretação semântica de uma anotação completa) — mas confirma que a produção de valores plausíveis mas incorretos é um risco transversal a qualquer abordagem que dependa do reconhecimento visual de padrões, seja um motor de OCR clássico ou um modelo generativo de grande dimensão, e que a extração direta da camada vetorial do PDF elimina esse risco por construção — corroborando o posicionamento assumido na síntese da secção 2.5.

6.8 Limitações do estudo

Devem ser tidas em conta quatro limitações desta comparação antes de generalizar as conclusões. Em primeiro lugar, a amostra de teste é pequena (5 PDFs, dos quais apenas 3 continham cotas detetáveis pelo modelo YOLO, totalizando 14 cotas de referência), pelo que os valores percentuais apresentados devem ser lidos como indicativos e não como estimativas estatisticamente robustas — ao contrário, por exemplo, do dataset de 1367 desenhos anotados usado por Khan, Yong, Chen, Tan, Feng, & Moon [1]; a ampliação do lote de teste para 20 a 30 desenhos, com uma referência verificada manualmente, fica como trabalho futuro (secção 8.1). Em segundo lugar, a análise granular por tipo/orientação e por taxonomia de erro (Tabelas 6.2 a 6.4) parte de subconjuntos ainda mais pequenos (apenas 3 cotas lineares e 11 diâmetros de referência, e entre 6 e 12 leituras espúrias por motor), pelo que deve ser lida como um levantamento qualitativo de padrões de erro a confirmar num lote maior, não como uma estimativa fiável das proporções exatas. Em terceiro lugar, embora se pudesse esperar que o símbolo de diâmetro (Ø) fosse mal reconhecido, por ser raro nos textos de uso geral com que estes motores são treinados, este foi corretamente recuperado em várias leituras pelos três motores testados, pelo que essa limitação não deve ser assumida como universal sem verificação adicional em datasets maiores. Em quarto lugar, como referido na secção 6.1, a referência é a saída da própria extração vetorial e não uma verificação manual independente, o que torna as métricas da extração vetorial otimistas por construção.

Estas limitações têm uma origem comum: a escassez de dados. O aumento do dataset — do conjunto de treino do detetor, do lote de teste e do número de cotas de referência — é a condição para transformar os valores indicativos deste capítulo em estimativas robustas (cf. secção 8.1).

7. Resultados do Pipeline Final

Os capítulos 5 e 6 avaliaram, respetivamente, a cobertura da deteção YOLO e a fiabilidade da extração vetorial de texto, ambas componentes isoladas do pipeline. Este capítulo avalia o pipeline unificado como um todo (secção 3.9), a partir de uma corrida real sobre um lote de desenhos da Durit, com o resultado escrito diretamente no template Excel de produção (*dados_preenchidos.xlsx*). Note-se que esta corrida corresponde à versão 1.11 do script, imediatamente anterior à versão 1.12 aqui documentada; os resultados devem por isso ser lidos como um retrato do comportamento do pipeline nesta fase de consolidação, não como uma validação final da versão mais recente.

7.1 Metodologia de avaliação

O pipeline processou os PDFs disponíveis na pasta *dataset_desenhos*, um lote que não coincide necessariamente com o conjunto de 44 desenhos utilizado nos capítulos 3, 5 e 6: nesta corrida foram geradas linhas a partir de 40 PDFs, e as contagens por categoria (por exemplo, 16 linhas sem deteção e 4 peças variadas, face aos 4 e 9 PDFs das categorias equivalentes do capítulo 5) não são diretamente comparáveis com as do diagnóstico, também porque aqui se contam linhas do template — uma por página ou por linha de tabela — e não PDFs. Cada linha escrita no template foi classificada, para efeitos desta análise, em quatro categorias: cota real extraída (um valor numérico de diâmetro ou comprimento), sem cota (tabela) (linha originada numa tabela de componentes cujo PDF correspondente não foi encontrado, ou cuja segunda passagem de resolução não foi executada nesta corrida), não detetado (nenhuma classe de peça, *peca_c* ou *peca_v*, identificada na página) e peça variada (*peca_v* detetado, mas sem lógica de extração de cotas integrada, cf. secção 4.4).

7.2 Resultados globais

O pipeline gerou 107 linhas no template, a partir de 40 desenhos (PDFs) de origem distintos. A Tabela 7.1 e a Figura 7.1 resumem a distribuição por categoria. Das 49 linhas originadas em páginas individuais (não provenientes de uma tabela de componentes), 28 (57,1%) obtiveram uma cota real — das quais 26 incluem também o diâmetro menor (coluna MD-Diam Int., cf. secção 3.9) e 24 o comprimento —, 16 (32,7%) ficaram sem deteção de qualquer classe de peça, 4 (8,2%) corresponderam a peças não-cilíndricas ainda sem extração automática de cotas e 1 (2,0%) ficou sem categoria atribuída. As restantes 58 linhas, todas originadas em tabelas de componentes (*Stückliste*), ficaram marcadas como “Sem cota (tabela)” nesta corrida — nenhuma foi resolvida pela segunda passagem de correspondência com o PDF do componente. Por classificação de material, das 107 linhas, 62 (57,9%) correspondem a materiais com prefixo HM- (metal duro, MD) e 44 (41,1%) a aço, segundo a regra descrita na secção 3.9; a linha restante ficou sem material atribuído. No ficheiro gerado por esta corrida, as duas etiquetas surgem trocadas (62 “Aço” e 44 “MD”), devido ao erro da versão 1.11 corrigido na versão 1.12. A Figura 7.1 apresenta graficamente esta distribuição para as 106 linhas com categoria atribuída, evidenciando que as linhas “Sem cota (tabela)” constituem, isoladamente, a maior categoria (58 linhas).

Tabela 7.1 — Distribuição das linhas geradas pelo pipeline unificado, por categoria (107 linhas, 40 PDFs de origem, corrida com a versão 1.11).

Categoria	Nº de linhas	%	Observação
Cota real extraída (página individual)	28	26,2%	26 casos também com diâmetro menor (MD-Diam Int.); 24 também com comprimento
Sem cota (tabela)	58	54,2%	Linha de Stückliste sem cota resolvida nesta corrida (cf. secção 7.3)
Não detetado	16	15,0%	Nenhuma classe peca_c/peca_v detetada na página
Peça variada (peca_v)	4	3,7%	Extração de cotas não implementada (cf. secção 4.4)
Sem classificação	1	0,9%	Linha sem categoria atribuída

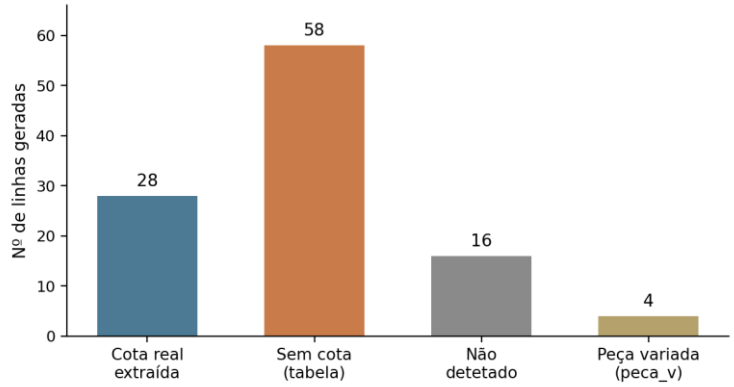


Figura 7.1 — Distribuição das 106 linhas com categoria atribuída (excluída a linha sem classificação) geradas pelo pipeline unificado extrair_cotas_tab.py (versão 1.11), por categoria de resultado (cf. Tabela 7.1, secção 7.2).

Exemplos ilustrativos de desenhos processados

Para complementar os valores agregados, as Figuras 7.2 a 7.5 apresentam exemplos representativos de desenhos processados pelo pipeline, escolhidos de forma a cobrir os principais tipos de resultado da Tabela 7.1: uma peça cilíndrica com cotas extraídas com sucesso, um desenho com tabela de componentes (Stückliste), um caso-limite em que a deteção falha ou fica abaixo do limiar e uma peça não-cilíndrica. Em cada exemplo, a imagem da página surge com as deteções do YOLOv8 sobrepostas, acompanhada do respetivo resultado no template Excel.

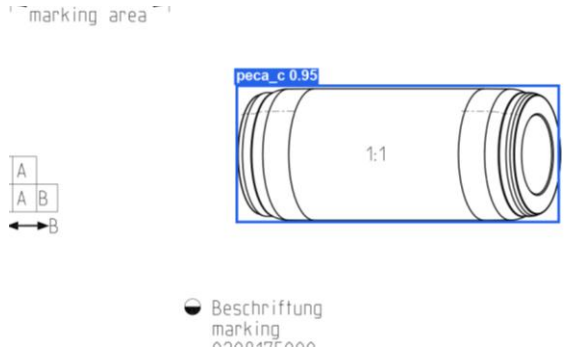
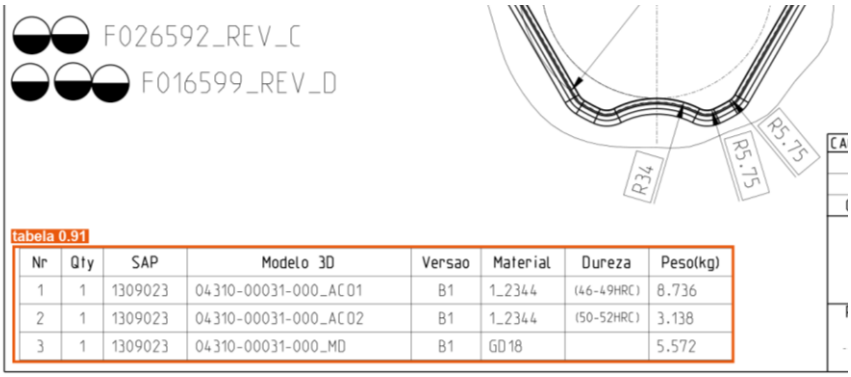


Figura 7.2 — *Peça cilíndrica com extração bem-sucedida: página com as caixas delimitadoras do YOLOv8 (peca_c, tabela, legenda) e as cotas reconstruídas (diâmetro exterior, diâmetro menor, comprimento e tolerâncias), cf. secção 3.5.*



Nr	Qty	SAP	Modelo 3D	Versao	Material	Dureza	Peso(kg)
1	1	1309023	04310-00031-000_AC01	B1	1_2344	(46-49HRC)	8.736
2	1	1309023	04310-00031-000_AC02	B1	1_2344	(50-52HRC)	3.138
3	1	1309023	04310-00031-000_MD	B1	GD18		5.572

Figura 7.3 — *Desenho com tabela de componentes (Stückliste): tabela detetada, linhas extraídas e correspondentes linhas no template Excel (cf. secções 3.6 e 7.3).*

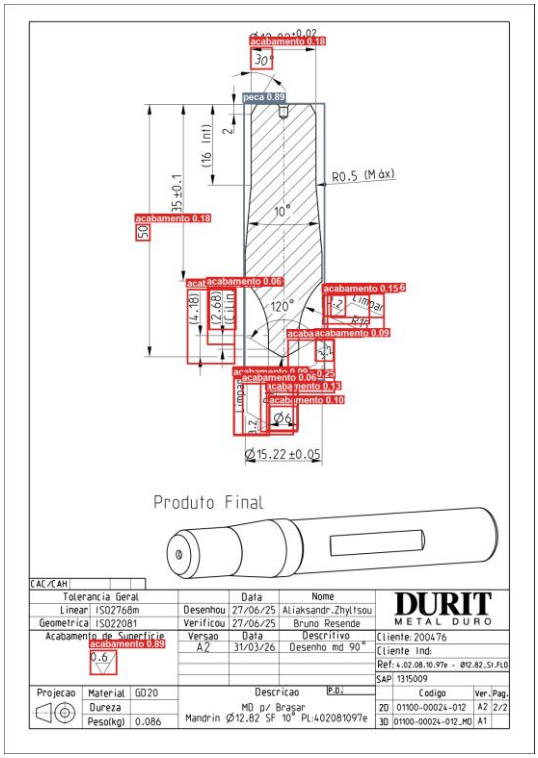


Figura 7.4 — *Caso-limite de deteção: página em que peca_c fica abaixo do limiar de 0,25 ou não é detetada, e resultado "Não detetado" no template (cf. secções 5.4 e 5.5).*

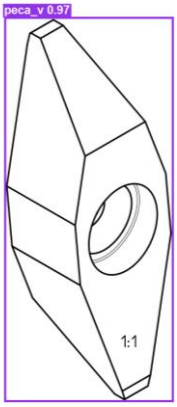


Figura 7.5 — *Peça não-cilíndrica (peca_v): página com a deteção e resultado "Peça variada" no template, sem extração automática de cotas (cf. secção 4.4).*

7.3 As 58 linhas de tabela por resolver: uma limitação a investigar

O facto de nenhuma das 58 linhas provenientes de tabelas de componentes ter sido resolvida com uma cota real nesta corrida é o resultado mais acionável deste capítulo. Não é possível, apenas a partir do ficheiro Excel de saída, distinguir entre duas causas distintas: (i) a segunda passagem de correspondência de PDF (secção 3.9) simplesmente não encontrou, para nenhuma destas 58 designações, um ficheiro correspondente no lote de PDFs processado — plausível se o lote não incluir os desenhos de detalhe de todos os componentes listados nas tabelas —, ou (ii) esta corrida em concreto foi feita com uma versão do script anterior à introdução dessa segunda passagem. O ficheiro de avisos (*_avisos.txt*), gerado automaticamente pelo *script* mas não incluído nesta análise, regista exatamente esta distinção caso a caso e deve ser consultado para uma resposta definitiva.

7.4 Discussão e próximos passos

Lidos em conjunto com os capítulos 5 e 6, estes resultados situam o pipeline unificado numa fase de consolidação sólida mas ainda incompleta: a deteção YOLO e a extração vetorial de texto — as duas componentes mais tecnicamente exigentes do projeto — foram avaliadas quantitativamente (capítulos 5 e 6), ainda que sobre amostras reduzidas (secção 6.8), enquanto a integração de ponta a ponta no template de produção ainda tem duas lacunas concretas e bem delimitadas: as 58 linhas de tabela por resolver (secção 7.3) e a extração de cotas para peças não-cilíndricas, ainda não implementada (secção 4.4). Como próximos passos diretos: (1) repetir esta corrida com a versão 1.12 do script sobre o lote de 44 desenhos confirmado dos capítulos 3, 5 e 6, para isolar o efeito da segunda passagem de resolução de tabelas; (2) analisar o ficheiro *_avisos.txt* gerado nessa corrida para separar, dentro das linhas de tabela por resolver, os casos de PDF em falta dos casos de erro de correspondência; e (3) considerar a integração da lógica de extração de cotas para *peca_v*, atualmente a maior lacuna funcional do pipeline face ao descrito no capítulo 3. O principal constrangimento a ultrapassar é a dimensão do dataset, e os trabalhos futuros — incluindo a passagem da extração de informação para a estruturação de roteiros de fabrico — são desenvolvidos no capítulo 8.

8. Constrangimentos e Trabalho Futuro

Este capítulo identifica o principal constrangimento que condiciona os resultados apresentados nos capítulos anteriores — a dimensão do dataset — e propõe direções de trabalho futuro. Estas direções organizam-se em torno de um objetivo mais amplo, que enquadra o projeto e justifica o título deste trabalho: passar da extração automática de informação dos desenhos técnicos para a estruturação de roteiros de fabrico otimizados, isto é, para a definição assistida da sequência de operações, máquinas e ferramentas necessárias para produzir cada componente em metal duro.

8.1 O constrangimento do dataset

O constrangimento mais importante deste trabalho é a quantidade de dados disponível. O modelo YOLOv8 foi afinado sobre 101 imagens anotadas, obtidas a partir de 44 desenhos técnicos (secção 3.2), e as comparações do capítulo 6 assentam em apenas 5 PDFs de teste, dos quais 3 continham cotas detetáveis, num total de 14 cotas de referência (secção 6.8). Estes números permitem demonstrar a viabilidade da abordagem, mas não sustentam conclusões estatisticamente robustas nem garantem que o modelo generaliza a desenhos novos. Vários resultados do diagnóstico do capítulo 5 — os 4 desenhos sem qualquer deteção de peça, os 2 casos-limite em que peca_c fica abaixo do limiar de confiança e a ambiguidade em torno da classe genérica peca — são coerentes com uma cobertura insuficiente da variedade de layouts, famílias de peças e estilos de cotagem presentes na produção.

São por isso necessários muito mais dados. Para efeitos de comparação, o dataset usado por Khan et al. [1], [2] contém 1367 desenhos anotados, cerca de trinta vezes mais do que o disponível neste projeto. Aumentar o dataset é um requisito transversal a todas as direções descritas nas secções seguintes: (i) para treinar o detetor com exemplos suficientes de cada classe, incluindo as que dispõem de menos exemplos, como as peças não-cilíndricas (peca_v), as tabelas de componentes e os campos da legenda; (ii) para constituir conjuntos de validação e de teste independentes do treino — o lote de teste de 20 a 30 desenhos referido na secção 6.8 é um mínimo, não um objetivo final; e, sobretudo, (iii) para que, numa fase posterior, existam dados suficientes para treinar modelos que relacionem as características de uma peça com o respetivo roteiro de fabrico, um problema que tende a exigir ainda mais exemplos do que a deteção de objetos.

O crescimento do *dataset* pode ser feito de forma incremental e com custo de anotação controlado. O próprio pipeline pode pré-anotar novos desenhos, cabendo ao anotador apenas corrigir as caixas delimitadoras no Label Studio (anotação assistida por modelo); os casos em que o modelo tem menor confiança, como os identificados no capítulo 5, devem ser priorizados para anotação (aprendizagem ativa); e a recolha deve ser estratificada por família de peça, de modo a evitar que o dataset continue dominado por poucos layouts. Técnicas de aumento de dados (*data augmentation*) podem complementar, mas não substituem, a recolha de desenhos reais, uma vez que a diversidade relevante é a de layouts e de convenções de cotagem.

8.2 Consolidar e alargar a extração de informação

Antes de avançar para a estruturação de roteiros, é necessário fechar as lacunas já identificadas na extração: (1) integrar no pipeline unificado a extração de cotas de peças não-cilíndricas (*peca_v*), atualmente uma função placeholder (secção 4.4); (2) resolver as 58 linhas de tabelas de componentes (*Stückliste*) por resolver, repetindo a corrida com a versão 1.12 (secção 7.3); (3) esclarecer o significado da classe genérica peca e considerar um segundo passo de deteção a limiar reduzido para *peca_c* (secção 5.6); e (4) tornar o filtro de plausibilidade adaptativo, ligando-o à maior cota confirmada na peça e não a um limiar absoluto (secção 6.5).

Para os roteiros, será ainda relevante alargar a extração a informação que hoje não é capturada de forma estruturada — por exemplo, tolerâncias geométricas (GD&T), rugosidades, material e dureza e as restantes indicações do bloco de título —, porque são estas características que condicionam as operações e os meios de fabrico necessários. Por fim, dado que o pipeline se apoia na camada vetorial do PDF, a robustez a desenhos digitalizados ou de menor qualidade exigirá um mecanismo de recurso, por exemplo um OCR aplicado apenas quando a camada vetorial estiver ausente.

8.3 Da extração de informação a roteiros de fabrico otimizados

O objetivo de longo prazo é usar a informação extraída dos desenhos como ponto de partida para propor roteiros de fabrico otimizados — a sequência de operações, com as respetivas máquinas, ferramentas e tempos, através da qual cada componente é produzido. Hoje, a definição de um roteiro depende da leitura do desenho por um engenheiro de processo e da sua experiência com peças semelhantes; a extração automática de cotas, tolerâncias, material e dados da legenda é o primeiro passo para tornar essa informação legível por máquina. Propõem-se as direções seguintes, que assumem a existência de dados históricos de roteiros e que devem ser definidas e validadas em conjunto com a Durit.

(1) Base de dados que liga desenhos a roteiros. O primeiro passo é construir um conjunto de dados que associe, para cada componente, os atributos extraídos do desenho (geometria, cotas e tolerâncias, material e dureza, peso) ao roteiro efetivamente executado — operações, sequência, máquinas, ferramentas, tempos de preparação e de ciclo —, a obter junto da empresa a partir dos seus sistemas de gestão. A ligação pode apoiar-se nos códigos de referência que já surgem nos desenhos, como o código SAP referido na secção 6.6. A qualidade e a cobertura deste conjunto são determinantes e reforçam a necessidade de aumentar o dataset (secção 8.1).

(2) Famílias de peças e recomendação por semelhança. Uma primeira abordagem, de baixo risco, consiste em representar cada peça por um vetor de atributos e recomendar, para uma peça nova, o roteiro das peças mais semelhantes (por exemplo, com métodos de vizinhos mais próximos ou de agrupamento), segundo a lógica da tecnologia de grupo. Esta abordagem é natural para as peças cilíndricas, que são a classe mais bem coberta pelo pipeline atual, e fornece uma linha de base com que comparar métodos mais complexos.

(3) Previsão de operações e de tempos. Modelos de aprendizagem supervisionada, como árvores de decisão e métodos de reforço de gradiente, podem prever, a partir dos atributos da peça, que operações são necessárias e quais os tempos de preparação e de

ciclo esperados, o que permitiria estimar custos e prazos logo na fase de orçamentação. A sequenciação das operações pode ser tratada como um problema de previsão de sequências, com modelos sequenciais ou baseados em grafos de precedência, respeitando restrições tecnológicas conhecidas (por exemplo, que uma operação de acabamento só pode ocorrer depois da respetiva operação de desbaste).

(4) Otimização do roteiro. Dispondo de previsões de operações e de tempos, a escolha do roteiro pode ser formulada como um problema de otimização — minimizar o tempo total de fabrico, o custo ou o número de preparações, sujeito às capacidades das máquinas, à disponibilidade de ferramentas e às restrições de precedência —, resolvido com programação por restrições, programação linear inteira ou heurísticas. O critério de "otimizado" a adotar (tempo, custo, utilização de máquina ou um compromisso entre estes) deve ser definido com a empresa.

(5) Validação com humano no ciclo e integração. O sistema deverá propor roteiros para revisão pelo engenheiro de processo, e não substituí-lo: as correções feitas devem ser registadas e reutilizadas como novos dados de treino. A avaliação deve recorrer a métricas concretas, como a distância de edição entre o roteiro proposto e o executado, o erro na estimativa dos tempos e o tempo poupado na elaboração de cada roteiro. A integração natural faz-se através do template Excel de produção já usado pelo pipeline (secção 3.9), evoluindo posteriormente para uma ligação direta ao ERP.

8.4 Síntese

Em síntese, a principal condição para avançar é o aumento e a diversificação do dataset, tanto de desenhos anotados como de roteiros associados. O trabalho aqui apresentado constitui a primeira etapa — a extração fiável e estruturada da informação dos desenhos — sobre a qual as etapas seguintes podem ser construídas de forma incremental, começando pelas peças cilíndricas.

9. Conclusões

Este trabalho desenvolveu um pipeline para a extração automática de cotas, tolerâncias e metadados de desenhos técnicos em PDF exportados de CAD, combinando um detetor YOLOv8, afinado sobre 101 imagens anotadas a partir de 44 desenhos da Durit, com a leitura direta da camada vetorial do PDF através do PyMuPDF e com um conjunto de regras geométricas e expressões regulares. Os três scripts desenvolvidos para cotas de peças cilíndricas, tabelas de metadados e legendas foram consolidados num único programa, *extrair_cotas_tab.py*, que escreve os resultados diretamente no template Excel usado pela equipa de produção.

Os resultados respondem, no essencial, aos objetivos definidos na secção 1.2. O diagnóstico do capítulo 5 mostrou que 86,4% do lote tem o comportamento pretendido no gate de deteção e que apenas 4 dos 44 desenhos correspondem a uma falha de deteção genuína. A comparação do capítulo 6 confirmou a vantagem da extração vetorial face a três motores de OCR de uso geral: estes recuperaram apenas 21%–29% das cotas de referência, com 60%–75% de leituras espúrias e tempos de execução 8 a 77 vezes superiores, e a confiança devolvida pelos próprios motores revelou-se insuficiente para filtrar essas leituras. A corrida do pipeline unificado (capítulo 7) produziu 107 linhas no template, com cota real extraída em 57,1% das linhas provenientes de páginas individuais.

Estes resultados devem ser lidos à luz das limitações identificadas: o *dataset* é pequeno; a comparação com OCR assenta em 5 PDFs e 14 cotas de referência derivadas da própria extração vetorial; as 58 linhas de tabelas de componentes ficaram por resolver na corrida avaliada; e a extração de cotas de peças não-cilíndricas ainda não está integrada. Acresce que a abordagem depende da existência de uma camada de texto vetorial, pelo que não se aplica diretamente a desenhos digitalizados.

Em conclusão, para desenhos exportados de CAD, a combinação de deteção de objetos com extração vetorial de texto revelou-se uma alternativa viável, leve e interpretável às abordagens baseadas em OCR ou em modelos de visão-linguagem. Constitui a primeira etapa — tornar a informação dos desenhos legível por máquina — da estruturação de roteiros de fabrico que dá título a este trabalho, cujas etapas seguintes, bem como o aumento do dataset de que dependem, foram propostas no capítulo 8.

Referências

[1] Khan, M. T., Yong, Z., Chen, L., Tan, J. M., Feng, W., & Moon, S. K. (2025). Automated parsing of engineering drawings for structured information extraction using a fine-tuned document understanding transformer. arXiv:2505.01530.

[2] Khan, M. T., Chen, L., Yong, Z., Tan, J. M., Feng, W., & Moon, S. K. (2025). From drawings to decisions: A hybrid vision-language framework for parsing 2D engineering drawings into structured manufacturing knowledge. arXiv:2506.17374.

[3] IntuitionLabs. (n.d.). Technical analysis of modern non-LLM OCR engines. https://intuitionlabs.ai/articles/non-llm-ocr-technologies

[4] Lin, Y.-H., Ting, Y.-H., Huang, Y.-C., Cheng, K.-L., & Jong, W.-R. (2023). Integration of deep learning for automatic recognition of 2D engineering drawings. Machines, 11(8), 802. https://doi.org/10.3390/machines11080802

[5] Kasem, M., Abdallah, A., Berendeyev, A., Elkady, E., Mahmoud, M., Abdalla, M., Hamada, M., Vascon, S., Nurseitov, D., & Taj-Eddin, I. (2024). Deep learning for table detection and structure recognition: A survey. ACM Computing Surveys. https://doi.org/10.1145/3657281

[6] Scheibel, B., Mangler, J., & Rinderle-Ma, S. (2021). Extraction of dimension requirements from engineering drawings for supporting quality control in production processes. Computers in Industry, 129, 103449.

[7] Xu, Y., Zhang, C., Xu, Z., Kong, C., Tang, D., Deng, X., Li, T., & Jin, J. (2024). Tolerance information extraction for mechanical engineering drawings – A digital image processing and deep learning-based model. CIRP Journal of Manufacturing Science and Technology, 50, 55–64.

[8] HumanSignal. (n.d.). Label Studio: Data labeling software [Software]. https://github.com/HumanSignal/label-studio

[9] Jocher, G., Qiu, J., & Chaurasia, A. (2023). Ultralytics YOLOv8 (Versão 8.0.0) [Software]. GitHub. https://github.com/ultralytics/ultralytics

[10] Artifex Software. (n.d.). PyMuPDF [Software]. https://github.com/pymupdf/PyMuPDF

[11] Google for Developers. (n.d.). Classificação: acurácia, recall, precisão e métricas relacionadas. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=pt-br