Universidade do Minho
Escola de Engenharia

Daniel Vilaça da Costa

Soluções Configuráveis para Inspeção Visual Automática

dezembro de 2021

Soluções Configuráveis para
Inspeção Visual Automática

Daniel Vilaça da Costa

UMinho | 2021

Universidade do Minho
Escola de Engenharia

Daniel Vilaça da Costa

Soluções Configuráveis para Inspeção Visual Automática

Dissertação de Mestrado
Mestrado Integrado em Engenharia Biomédica
Ramo de Eletrónica Médica

Trabalho efetuado sob a orientação do
Professor Doutor Carlos Alberto Batista Silva

dezembro de 2021

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Agradecimentos

Ao meu orientador Professor Doutor Carlos Silva, estou muito grato por todo o seu suporte e partilha de conhecimento. A sua disponibilidade ao longo de todo o trabalho foi fundamental para o desenvolvimento da dissertação.

Quero também agradecer aos meus colegas de projeto, que me ofereceram apoio e críticas em diversos aspetos do trabalho.

Por último, expresso a minha enorme gratidão a toda a minha família, amigos e colegas por sempre me suportarem incondicionalmente em todo o meu percurso académico, quer nos bons quer maus momentos.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Soluções Configuráveis para Inspeção Visual Automática

Atualmente, na área da indústria, as empresas requerem elevados níveis de qualidade e, se possível, aumentar a produtividade. Para tal, é necessário a realização de inspeção em linhas de montagem, geralmente visual, por trabalhadores qualificados. Este processo revela-se muitas vezes exaustivo e monótono, assim como subjetivo ao trabalhador que executa o processo de inspeção. Uma alternativa consiste na automatização deste tipo de tarefas, com o emprego de sistemas de inspeção visual automática, visto permitirem a realização de inspeção sempre de forma objetiva e independente de muitos dos fatores que afetam os humanos em ambiente de trabalho.

O presente trabalho tem como finalidade o desenvolvimento de uma *toolbox* de soluções configuráveis, apenas com base nos operadores disponíveis no software *HALCON*, de forma a responder a alguns problemas de inspeção visual utilizando processamento de imagem. A dissertação foca-se nos problemas de inspeção da dispensação de cola, a deteção da ausência ou do posicionamento defeituoso de parafusos, deteção de peças como tubos, casquilhos e conectores para a realização de tarefas de *object picking*, inspeção do posicionamento de cabos fita e do posicionamento e dimensionamento de peças.

As metodologias desenvolvidas para resolução dos problemas abordados demonstraram ser eficientes e capazes de cumprir os objetivos impostos em cada tipo de inspeção. Deste modo, as soluções propostas mostraram-se propícias a serem implementadas como apoio à inspeção visual automática de problemas industriais.

Palavras chave: Inspeção Visual Automática, *HALCON*, Processamento de Imagem, Indústria 4.0

Abstract

Configurable Solutions for Automated Visual Inspection

Nowadays, companies in the industrial world require high levels of product quality and, if possible, increase productivity. To achieve this, it is necessary to carry out inspection tasks along the assembly line, usually visual, by qualified workers. This process is often exhausting, monotonous as well as subjective to the worker performing the inspection task. An alternative to this is to automate these tasks, with the employment of automated visual inspection systems, as these always allow objective inspections to be carried out and are independent of many of the factors affecting human workers in the work environment.

This works aims to develop a toolbox of configurable solutions, solely based on the available operatores of the *HALCON* software, in order to answer some of the existing visual inspection problems by applying image processing. The dissertation focuses on problems such as inspection of glue dispensing, detection of the absence or defective positioning of screws, detection of objects such as tubes, bushings and connectors for carrying out object picking tasks, inspection of the positioning of ribbon cables and the positioning and dimensioning of objects.

The developed methodologies proved to be efficient and capable of meeting the imposed requirements for each of the problems addressed. Thus, the proposed solutions proved to be suitable for implementation as support for automatic visual inspection of industrial problems.

Keywords: Automated Visual Inspection, *HALCON*, Image Processing, Industry 4.0

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

IDE Integrated Development Environment. 5, 7

IVA Inspeção Visual Automática. 2, 3, 8, 9, 21, 22, 25, 28, 42, 57, 58

OCR Optical Character Recognition. 6–9

OCV Optical Character Verification. 6, 8

ROI Region Of Interest. viii, ix, 22–25, 28, 30–34, 36, 37, 39, 41–47

Capítulo 1

Introdução

Este capítulo expõe a motivação e principais objetivos para o desenvolvimento da presente dissertação. Ainda, são referidas as contribuições resultantes do trabalho, bem como se elucida a estrutura da dissertação.

1.1 Motivação

A área da indústria sofreu três grandes revoluções industriais, que podem ser resumidas na Figura 1.1 [1]. A Primeira Revolução Industrial começou no fim do século XVIII, incorporando máquinas movidas a vapor e água para substituir métodos de produção manuais, criando as fábricas. Com a ajuda da energia elétrica veio a Segunda Revolução Industrial, entre meados do século XIX e meados do século XX, que introduziu a produção em massa na indústria. A expansão acelerada nas áreas tecnológicas ajudou as empresas a implementar tecnologias digitais, alcançando maiores níveis de produção, considerando-se esta a Terceira Revolução Industrial. A iminente Quarta Revolução Industrial (Indústria 4.0), representa o desenvolvimento adicional das tecnologias de automação na indústria, com o objetivo de criar processos automáticos capazes de tomar decisões e de monitorização em tempo real [2, 3].

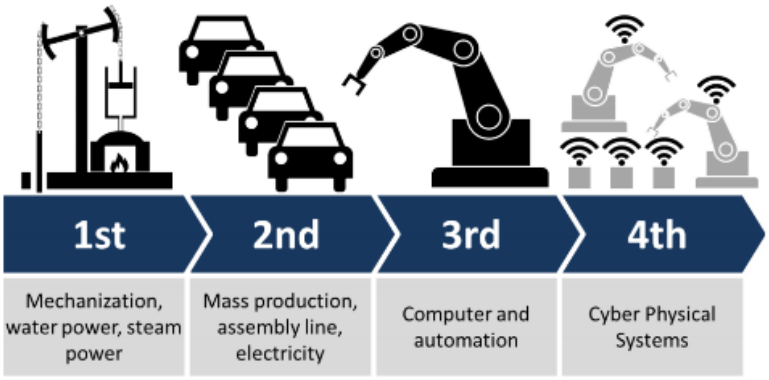


Figura 1.1: Revoluções industriais. Retirado de [1].

Associado ao avanço rápido nas tecnologias e modernização, a Indústria 4.0 traz benefícios em questões de tempo, custos e flexibilidade [3]. Isto cria um aumento na competitividade dentro da indústria de

manufaturação. Com o apoio de sistemas de informação e comunicação, as empresas tentam superar a competição, estudando novas formas para aumentar o design, desenvolvimento, fabrico e uso de produtos, nomeadamente utilizando tecnologias de computação para automatizar o maior número de passos de manufaturação possível, no sentido de atingir os requisitos exigidos pelos clientes, ao mesmo tempo que se diminuem custos [1].

A maior disponibilidade de produtos implica o aumento das expectativas de qualidade por parte dos clientes, levando a que as empresas invistam cada vez mais em processos de controlo de qualidade, de forma a assegurar os standards pretendidos [4, 5]. Com o aumento na exigência de qualidade por parte do consumidor, a inspeção na manufaturação de produtos é um passo cada vez mais fundamental para assegurar os standards de qualidade pretendidos, que são cada vez mais apertados [4].

A inspeção visual é um dos tipos de inspeção mais comuns na manufaturação de produtos no âmbito da indústria. Devido ao processo repetitivo e exaustivo ligado à inspeção visual realizada por pessoas, esta tarefa torna-se muitas vezes subjetiva, monótona e errática, mesmo quando feita por operadores experientes e qualificados. Utilizar operadores humanos para o processo de inspeção implica ter de gerir entre tempo e exatidão de inspeção, consequência da complexidade de algumas tarefas [4, 6]. Adicionalmente, é necessário ter em conta fatores humanos, sociais, ambientais e organizacionais que também influenciam os resultados obtidos. Fatores humanos individuais como idade, género, estado de fatiga, experiência e problemas de visão, assim como condições de ruído e iluminação associadas ao ambiente de trabalho, entre outros, trazem efeitos negativos para o rigor elevado que é exigido em tarefas de inspeção visual [6, 7] . A deteção tardia de produtos defeituosos pode acarretar perdas significativas, principalmente após a fase de produção. É desta forma imperativa a deteção de defeitos o quanto antes na linha de montagem.

Sistemas de Inspeção Visual Automática (IVA), pretendem resolver estes tipos de problemas, por não serem influenciados por fatores humanos como exaustão e fatiga, sendo também independentes de fatores externos, por terem, por exemplo, ambientes controlados em termos de iluminação e ruído. Além disso, a automação da inspeção proporciona a deteção de defeitos em produtos mais pequenos, onde para uma pessoa, os defeitos não seriam percetíveis, a execução de tarefas mais complexas e o uso de luz fora do espetro visível ao olho humano [4, 5] . O objetivo principal de IVA consiste em realizar uma inspeção mais rápida, ter precisão nos resultados esperados, sempre de forma objetiva, permitindo ainda guardar os dados, para documentação e rastreabilidade [5].

O emprego de sistemas de IVA deve ter em consideração alguns problemas associados. Um problema nestes sistemas fundamenta-se no facto de ser difícil reproduzir a adaptabilidade inerente a uma pessoa. Em muitos casos, um humano consegue avaliar um objeto a ser inspecionado, apenas tendo um caso de referência, enquanto que uma máquina frequentemente necessita de vários casos para que seja possível otimizar os parâmetros do algoritmo usado para a inspeção [5]. Pequenas variações na iluminação podem ter efeitos significativos nos resultados da inspeção obtidos. Por vezes, os produtos sofrem alterações de cor ou pequenos detalhes, precisando do ajuste de alguns parâmetros dos algoritmos de visão, obrigatoriamente por pessoas especializadas.

1.2 Objetivos

Os passos de inspeção na área industrial são atualmente indispensáveis como meio para a obtenção de produtos de elevada qualidade. Neste trabalho, são explorados alguns dos obstáculos nos sistemas de IVA. É desenvolvida uma *toolbox* de soluções configuráveis, capaz de ultrapassar problemas de controlo de qualidade específicos. Os objetivos para o trabalho são os seguintes:

- Inspeção da dispensação de cola;
- Inspeção do posicionamento de *foils*/cabos fita;
- Deteção da ausência ou posicionamento defeituoso de parafusos;
- Inspeção de posicionamento e montagem defeituosa a partir do cálculo de distâncias;
- Deteção de peças para *object picking* – casquilho, conector, tubo.

As soluções devem ser desenvolvidas com recurso exclusivo ao *software HALCON* e devem poder ser configuráveis de forma a que o utilizador consiga adaptar as soluções a diferentes situações.

1.3 Contribuições

Ao longo desta dissertação foram estudadas metodologias que podem ser consideradas como contribuições, com o objetivo de colaboração na área tecnológica e industrial.

Pretende-se contribuir a partir da resolução de alguns problemas atuais existentes em linhas de montagem nas indústrias, nomeadamente o desenvolvimento de uma *toolbox* de soluções de apoio a sistemas de IVA no controlo de qualidade de produtos, através da deteção e medição de objetos específicos, com base num conjunto de problemas existentes na área industrial.

1.4 Estrutura da Dissertação

Esta dissertação está estruturada em 6 Capítulos, organizados como enunciado a seguir. Este primeiro Capítulo 1, comunica a motivação do trabalho, assim como o seu contexto, e objetivos. O Capítulo 2 passa a apresentar alguns dos *softwares* que existem atualmente no mercado capazes de criar sistemas de IVA, referindo algumas das caraterísticas deles e também funcionalidades que estes integram, assim como a fundamentação do desenvolvimento de uma *toolbox* de soluções configuráveis. O Capítulo 3 descreve e carateriza os problemas abordados ao longo do trabalho, relativamente ao que se pretende detetar e inspecionar em cada um dos conjuntos de imagens, clarificam-se os requisitos que cada solução deve ter e a distinção entre situações de sucesso e situações de falha, no ato da inspeção visual. De seguida, no Capítulo 4 é explicada a utilização de registo de imagens em IVA e são expostas as soluções que fazem parte da *toolbox* e que foram desenvolvidas como forma de resposta aos problemas previamente apresentados. São mostrados e discutidos os resultados obtidos no Capítulo 5, esclarecendo os aspetos em que

cada uma das abordagens aos problemas desenvolvidas conseguiu cumprir os objetivos e aspetos onde falharam. A dissertação finaliza no Capítulo 6, com algumas notas conclusivas e perspetivas futuras para o seguimento deste trabalho.

Capítulo 2

Estado da Arte

Atualmente, com o rápido avanço de tecnologia, vem a oportunidade da automatização de processos de inspeção visual na manufaturação de produtos. Associada ao desenvolvimento de soluções para resolver problemas de inspeção, está a criação de mais bibliotecas para programação de soluções, assim como *softwares* para facilitar o desenvolvimento das ditas soluções, principalmente para utilizadores com menor conhecimento na área de programação de fluxos de operações.

O capítulo presente cobre alguns dos softwares que possibilitam a criação de fluxos de operações de inspeção visual automática atualmente existentes, sendo apresentadas algumas caraterísticas e funcionalidades sobre cada um deles. São abordados os softwares *HALCON*, *Sapera Vision* e *Adaptive Vision Library*, com bibliotecas de mais alto nível com operações prontas a serem utilizadas, mais dedicadas para a resolução de problemas visuais na área da indústria, com ambientes gráficos de desenvolvimento de soluções associados, assim como o *OpenCV*, uma biblioteca *open-source* com variadas funções para todos os tipos de problemas de imagem, tendo uma estrutura de nível mais baixo comparativamente às outras bibliotecas referidas. É por fim analisada uma grande limitação comum a todos os *softwares* que fundamenta o desenvolvimento de uma *toolbox* de soluções parametrizáveis.

2.1 Softwares de Desenvolvimento de Apoio a IVA

2.1.1 HALCON

HALCON é um software de visão por computador criado pela *Machine Vision Technology* (MVTec), oferecendo uma biblioteca de processamento de imagem com capacidade de aceleração GPU para a otimização da performance dos algoritmos. Tem como principal objetivo o desenvolvimento de algoritmos para a resolução de problemas em linhas de produção, montagem e inspeção na indústria. O *HALCON* pode ser utilizado para áreas como a indústria automóvel, área médica, produção de componentes elétricos e de semicondutores [8].

O *HALCON* inclui um ambiente de desenvolvimento integrado (IDE), designado por *HDevelop*, permitindo o desenvolvimento de soluções da biblioteca com maior suporte, fornecendo documentação para os operadores utilizados e uma interface gráfica para maior facilidade na manipulação e visualização de

variáveis e de resultados da execução de cada função [9]. Os algoritmos desenvolvidos dentro do *HDevelop* podem posteriormente ser executados em aplicações escritas em linguagens de programação como *C*, *C++*, *C#* e *Python*, através do interpretador *HDevEngine* sem necessidade de compilação, exportados e compilados nessas linguagens, ou então diretamente desenvolvidos nas linguagens mencionadas [8].

Esta biblioteca contém operadores e *data types* próprios dos quais o programador pode usufruir desde funcionalidades de baixo nível como medição, operadores morfológicos e aquisição de imagem, até algoritmos mais complexos como metrologia, *matching*, classificação, segmentação, [10]. Relativamente a operadores de *matching*, estes são divididos em 2 tipos: ortogonal e perspetiva [10]. Aqueles dedicados para vista ortogonal são modelos baseados em formas, correlações, componentes ou de deformação local. Relativamente aos de vista perspetiva, existem os modelos deformável de perspetiva ou baseado em descritores [11]. Os operadores de *matching*, juntamente com operadores de criação de matrizes de transformação e operadores para a transformação de imagens ou regiões consoante estas matrizes, possibilitam o registo de imagens. Assim, é possível resolver problemas de inspeção de placas e *wafers*, inspeção de superfícies e texturas, posicionamento e alinhamento, OCR e OCV, leitura de códigos (barras, data e QR), inspeção de impressão, deteção e classificação de comprimidos e processamento de imagens de angiografia [8, 12].

Adicionalmente, é disponibilizado um módulo de *deep learning*, que permite a resolução de problemas de classficação, deteção de anomalias e de objetos e segmentação, providenciando algoritmos de exemplos para cada um deles [10]. Oferece a possibilidade de aplicar modelos de redes neuronais previamente treinadas como *alexnet*, *mobilenet* e *resnet*, assim como redes criadas pela empresa, uma que é mais compacta, ou seja, tem menos camadas, de forma a ser mais eficiente em termos de memória e tempo de execução e uma outra rede mais complexa, para tarefas de classificação mais complicadas [10].

2.1.2 Sapera Vision

Sapera Vision Software, criado pela *Teledyne DALSA*, é um conjunto de bibliotecas de desenvolvimento de aplicações para visão industrial, tanto em *C++* como *C#*. Este *software* encontra-se dividido em 3 partes: *Sapera LT*, *Sapera Processing* e *Astrocyte* [13].

O *Sapera Processing* é um biblioteca para processamento de imagem tanto de baixo nível como de alto nível. Inclui funções básicas de processamento de imagem, como vários tipos de filtros, operadores morfológicos e geométricos e transformações entre sistemas de coordenadas. Algoritmos mais avançados estão também presentes, para leitura de códigos, medição, ferramentas de calibração para compensar distorções nas imagens causadas pela câmara. A biblioteca permite ainda o desenvolvimento de soluções para segmentação, OCR, procura pelo uso de áreas ou cantos e *matching* de padrões, baseado nos contornos de objetos ou baseado na correlação cruzada normalizada (NCC) [14]. No entanto, a biblioteca *Sapera Processing* não tem suporte para aquisição de imagem e configuração de câmaras, sendo complementado nestes aspetos pelo *Sapera LT* [13, 14]. O *Sapera Processing* inclui algoritmos dedicados a inteligência artificial para realizar a inferência em imagens, sendo que necessita do pacote *Astrocyte* para o treino de modelos de *deep learning*, otimizados para execução em CPU ou em GPU. Este pacote

contém algoritmos capazes de resolver problemas de deteção de anomalias, de objetos, classificação, segmentação e de redução de ruído, com o uso de redes neuronais [15].

2.1.3 Adaptive Vision

Adaptive Vision Library (AVL) é uma biblioteca para desenvolvimento de soluções de visão por computador em *C++* e *C#*. Possui um IDE, *Adaptive Vision Studio* (AVS), onde as soluções são desenvolvidas com base em fluxos de operações. As aplicações criadas no AVS podem ser depois exportadas para *C++* ou *C#* [16]. A construção do fluxo de uma aplicação a partir do IDE consiste apenas na utilização dos operadores disponíveis de forma gráfica a partir da interação com *widgets*, de forma a permitir o seu uso por pessoas com pouco conhecimento de programação de baixo nível [17]. O AVS permite ainda substituir a AVL pelo *OpenCV*, sendo possível elaborar fluxos de operações com esta biblioteca. A biblioteca inclui um conjunto de funções para processamento de imagem e algoritmos de visão por computador mais gerais, assim como um conjunto de algoritmos para o desenvolvimento de sistemas de inspeção industrial, prontas a serem adaptadas e utilizadas. A AVL dispõe de operadores de medição, morfológicos, filtros e transformada de Hough. Inclui ainda algoritmos para resolução de problemas de leitura de códigos de barras e de dados, *matching* por bordas ou correlação, deteção de bordas, calibração, segmentação, *shape fitting* e OCR [16–18].

O *Adaptive Vision Studio* possui também um *add-on* de *deep learning* que permite criar modelos para aplicações como deteção de anomalias de formas e superfícies, *feature detection*, classificação de objetos, segmentação e localização de pontos e objetos [17, 19]. Em comparação com bibliotecas para *deep learning*, como *PyTorch* e *Tensorflow*, com estruturas de baixo nível, este *add-on* oferece ainda algoritmos de nível mais elevado, prontas para implementação em problemas mais simples. Os operadores disponíveis fazem uso do *WEAVER* um motor de inferência de *deep learning* para *machine vision* em *C* e *C++* [20]. É capaz de executar modelos com a biblioteca *Keras*, para modelos funcionais, mas com limitações relativamente às camadas que podem ser utilizadas, e restrições dentro de algumas destas [21]. Tem a possibilidade de utilizar GPU compatíveis com CUDA, de modo a acelerar o treino das redes neuronais.

2.1.4 OpenCV

O *software OpenCV* (*Open Source Computer Vision Library*) é uma biblioteca *open-source* para visão por computador em tempo real, para resolução de problemas de imagem, tendo como grande objetivo proporcionar a elaboração de aplicações consideravelmente complexas a partir da sua infraestrutura simples [22, 23]. Possui milhares de algoritmos otimizados para imagem 2D, estruturas 3D e vídeo prontos a usar, tanto numa situação académica como em condições industriais. As funções disponíveis encontram-se divididas em vários módulos, em que cada um está dedicado a um tipo de problemas de visão por computador. Esta biblioteca tem a finalidade de desenvolvimento de soluções associadas a problemas como a deteção de objetos, segmentação, imagem médica, inspeção de produtos e sistemas de reconhecimento facial, de gestos e objetos, a partir de operações simples como operações morfológicas e manipulação de histogramas até operações mais complexas como detetores de *features*, análise de vídeo

[23, 24]. Contém também uma biblioteca de *machine learning*, para complementar os algoritmos de modo a aprenderem sem interação humana e ainda um módulo de *deep learning*, para que o *OpenCV* possa ser utilizado em conjunto com bibliotecas como *Pytorch* e *Tensorflow* para aplicar modelos de redes neuronais.

Softwares open-source trazem grandes vantagens no desenvolvimento de aplicações de visão, dado o aumento da acessibilidade de câmaras, permitindo uma expansão muito mais célere desta biblioteca, podendo ser utilizado para o desenvolvimento de aplicações de inspeção livremente. Comparativamente a outros outros softwares, por ter maior nível de suporte e documentação, a biblioteca é expandida facilmente, tanto em novas funções, como na otimização das funções já existentes [22].

Tal como as outras bibliotecas exploradas, os operadores do *OpenCV* podem ser integrados em algumas linguagens de programação como *C++*, *Python*, *Java* e *Matlab* e alguns algoritmos aproveitam de interfaces aceleradas por GPU, através do CUDA, com o objetivo de usufruir as vantagens da computação de imagem em GPU relativamente a CPU, aumentando bastante a performance das aplicações, enquanto as funcionalidades destas se mantêm as mesmas [25]. Inclui ainda suporte para *multithreading*.

2.2 Soluções Configuráveis

Em grande parte do mundo industrial, principalmente em indústrias de produção em massa, um objetivo principal é assegurar a qualidade dos produtos que são criadas, a par da redução de custos [2]. Por esta razão, tarefas de inspeção são um passo importante no processo de produção, para garantir o alcance dos requisitos dos produtos. Muitas vezes, a inspeção é uma tarefa complicada, pelo que se torna exaustiva e demorada quando realizada por operadores humanos, mesmo que qualificados. Ainda, a experiência e treino de cada pessoa resulta em resultados subjetivos, podendo não ser possível obter os níveis de exatidão necessários [6].

Os *softwares* explorados possuem funcionalidades de processamento de imagem de baixo nível, com operadores morfológicos e medição. Existem também operações de nível mais elevado, como a leitura de códigos de barras, OCR, OCV, classificação e segmentação, conseguindo responder a algumas tarefas de IVA mais genéricas e simples. No entanto, não dispõem de soluções de IVA prontas a usar e capazes de se adaptarem a problemas mais complexos como os que são estudados neste trabalho, pelo que resulta na pertinência da criação de uma *toolbox* que integra soluções completas de IVA e adaptáveis para vários problemas específicos na inspeção visual que não são resolúveis apenas com os algoritmos atuais presentes nestes tipos de *softwares*.

A possibilidade de parametrização das soluções da *toolbox* é essencial, com o intuito de alteração dos fluxos existentes quando existem falhas nos sistemas de IVA. Deste modo, a *toolbox* deve ser facilmente personalizada a novas situações dos problemas existentes, assim como a novos problemas de inspeção [26]. No entanto, continua a ser necessária a interação humana como suporte, tanto para a implementação e execução da *toolbox* nas linhas de montagem, como para a adaptação das soluções face a mudança dos problemas de IVA.

2.3 Sumário

No mercado encontramos vários ambientes e bibliotecas para o desenvolvimento de sistemas IVA. Neste capítulo foram estudados os *softwares HALCON*, *Sapera Vision*, *Adaptive Vision* e *OpenCV*, que contêm ferramentas mais básicas, como por exemplo para captura de imagem ou processamento de imagem, como operações morfológicas e filtros. Dispõem também de operadores mais avançados para registo de imagens, deteção de bordas e cantos, entre outros. Oferecem ainda exemplos de fluxos de operações prontos a utilizar em problemas de IVA, como leitores de códigos, ferramentas de *matching*, classificação, OCR, segmentação e inspeção de texturas e superfícies. Apesar da existência de soluções de IVA nos *softwares*, estas apenas conseguem resolver os problemas simples de deteção e inspeção de objetos mencionados. Não existem fluxos de operações capazes de solucionar problemas críticos mais complexos e específicos. Verificou-se que é essencial o desenvolvimento de uma *toolbox* parametrizável com soluções que sejam generalizadas mas que consigam resolver vários problemas críticos de IVA, a partir da modificação de alguns parâmetros por parte do utilizador, para poder adaptar as soluções a novas situações dos problemas que pretende resolver, assim como implementar as soluções a novos problemas de inspeção nas linhas de montagem.

Capítulo 3

Caraterização das Soluções

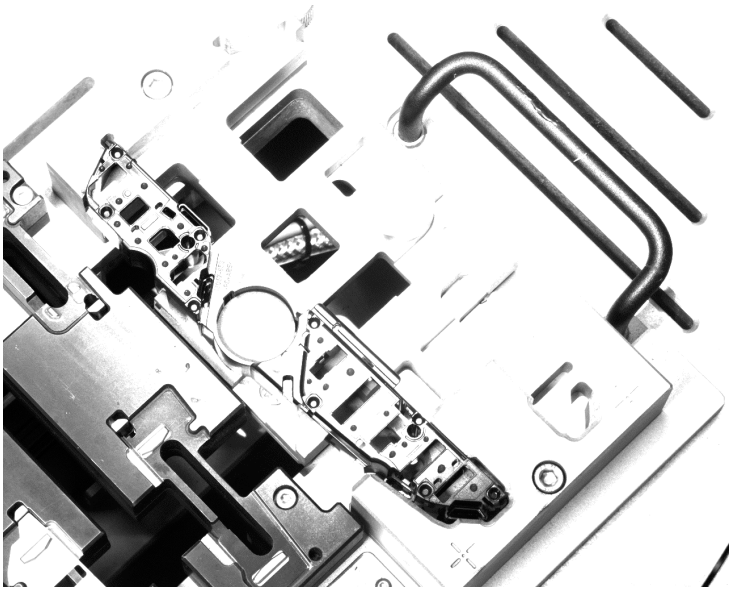
O presente capítulo é feita uma descrição dos problemas que devem ser resolvidos, juntamente com uma caracterização dos requisitos que devem ser cumpridos, a explicitação da base de dados disponível para cada um dos problemas e desenvolvida uma solução no sentido de atingir os objetivos existentes na inspeção.

3.1 Inspeção de Dispensação de Cola

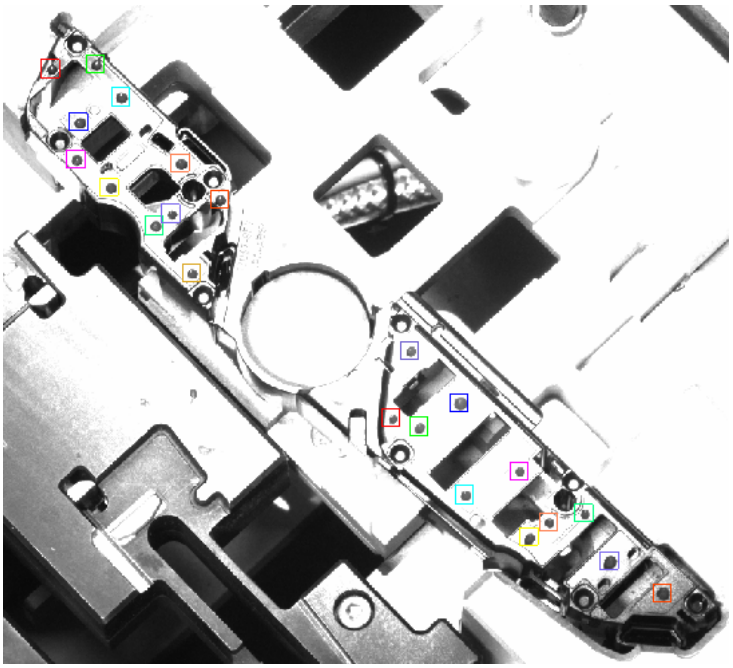
A inspeção dos pontos de dispensação de cola numa peça tem o propósito de verificar se a colocação de cola foi bem ou mal executada, em todos os locais necessários. Um exemplo de imagem prevista para esta inspeção encontra-se na Figura 3.1a, onde se observa uma estrutura com múltiplos objetos arredondados. Estes objetos correspondem aos pontos de cola a serem inspecionados, numerando um total de 22 instâncias de cola a analisar por imagem.

A inspeção da dispensação de cola é realizada para cada um dos 22 locais, respetivamente marcados na Figura 3.1b. Considera-se uma dispensação de cola num único ponto bem sucedida, se a cola dispensada se encontra dentro de limites pré-definidos pelo utilizador. Para que a dispensação seja reconhecida como sucesso na inspeção como um todo, é fundamental que todas as instâncias de cola cumpram as especificações de quantidade de cola pretendidas. Os limites máximos e mínimos de cola devem poder ser definidos individualmente para cada ponto. A Figura 3.1b serve também de exemplo de uma inspeção bem sucedida.

Diz-se que a inspeção falha quando qualquer um dos pontos de dispensação na corrente imagem não cumpre o critério de sucesso referido. Considera-se a existência de duas possíveis situações de falha na inspeção relativamente à quantidade de cola dispensada. Quanto às falhas relativas a quantidades de cola, num primeiro caso, exemplificado pela Figura 3.2a, a cola foi dispensada em quantidade inferior ao valor mínimo requerido, podendo a quantidade de cola dispensada não ter sido suficiente, o que acontece nas regiões marcadas pelos quadrados vermelhos ou então quando não foi dispensada qualquer cola, como ocorre no quadrado laranja. A outra situação de falha encontra-se presente na Figura 3.2b, em que a quantidade de cola dispensada ultrapassa o limite máximo permitido.

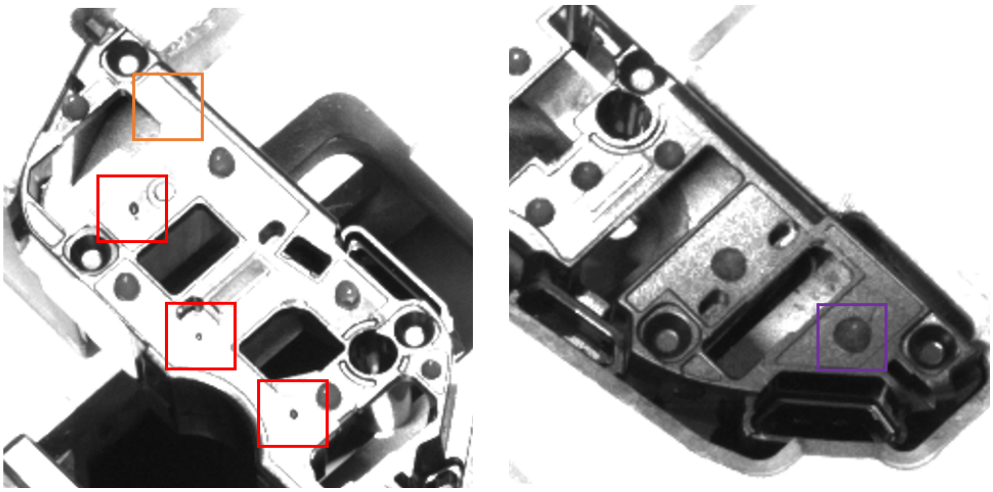


(a)



(b)

Figura 3.1: (a) Exemplo de imagem onde é realizada a dispensação de cola.(b) Marcação dos locais onde deve ser feita a dispensação da cola.

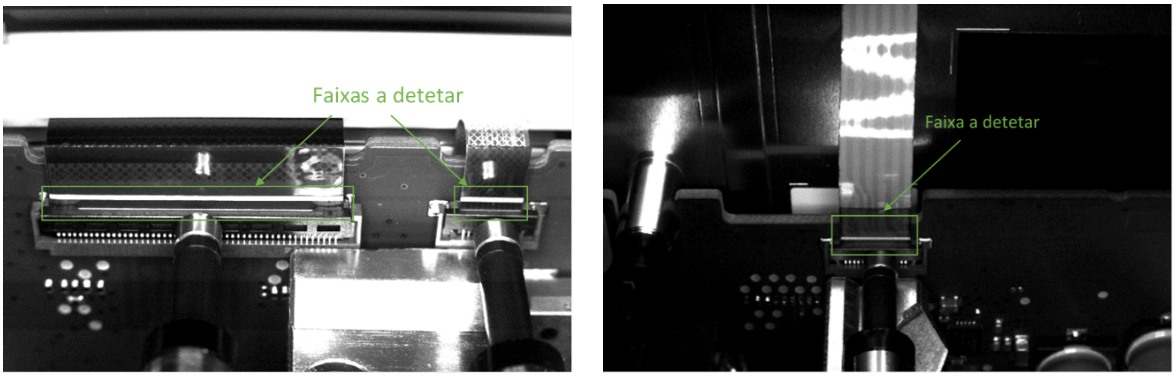


(a) (b)

Figura 3.2: Exemplos de situações de falha na inspeção de dispensação de cola: (a) quantidade de cola dispensada insuficiente nas áreas marcadas; (b) quantidade de cola dispensada excessiva na área marcada.

3.2 Inspeção de Cabos Fita

A inspeção de cabos fita ou *foils* consiste na análise do posicionamento correto de cabos fita em dois tipos de imagens, o Tipo A (Figura 3.3a) e o Tipo B (Figura 3.3b). Estes cabos possuem 2 faixas brancas que servem como guias para a inspeção. O objetivo da inspeção resume-se na verificação da existência de apenas uma ou ambas as faixas, sendo que para o caso de que as 2 faixas se encontram visíveis, realizar a medição da largura desta, de modo a concluir acerca do sucesso na colocação do *foil*.



(a) (b)

Figura 3.3: Inspeção da colocação de cabos fita e respetivas faixas a serem detetadas em (a) imagens tipo A e (b) imagens tipo B.

O critério de avaliação na inspeção dos cabos consiste em verificar a visibilidade de apenas uma faixa do respetivo cabo fita. É ainda necessário ter em conta um cenário em que a segunda faixa se encontra também presente. Neste caso, a largura visível da segunda faixa deve ser inferior a uma percentagem máxima da largura total da faixa de referência, valor este sendo pré-definido pelo operador. Este tipo de situações são também considerados como sendo eventos de uma inspeção bem sucedida. Exemplos de sucesso na inspeção apresentam-se explicitados na Figura 3.4. A Figura 3.4a representa uma situação ideal, em que apenas uma faixa é visível. Na Figura 3.4b verifica-se que além da primeira faixa existe também uma pequena porção da segunda faixa, evidenciada pela marcação na imagem, no entanto considera-se que o cabo está devidamente colocado visto a porção da faixa que se encontra visível não ser suficientemente para ser considerada como falha, tal como previamente referido.

Contrariamente aos casos supra referidos, a falha na colocação do cabo fita acontece quando existe a presença da segunda faixa referida e a percentagem da segunda faixa branca foi determinada estar acima do valor *threshold* máximo definido ou até mesmo na sua totalidade. Tal é demonstrado na Figura 3.4c, onde é claramente perceptível de que ambas as faixas possuem uma largura bastante similar, até que idêntica, estando o *foil* evidentemente mal colocado, sendo imperativo que casos como este sejam classificados como falha no momento de inspecionamento.

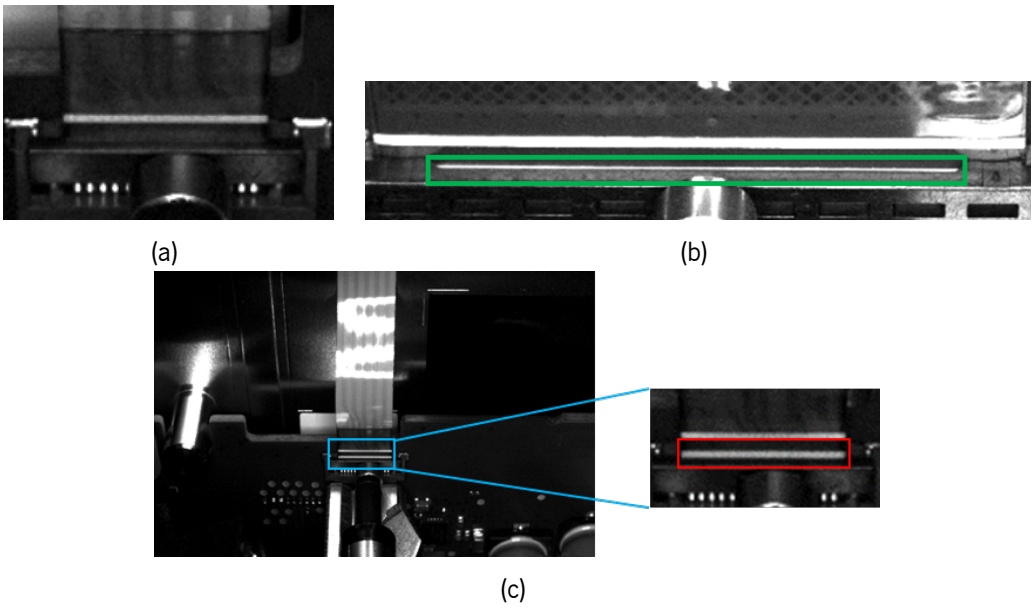
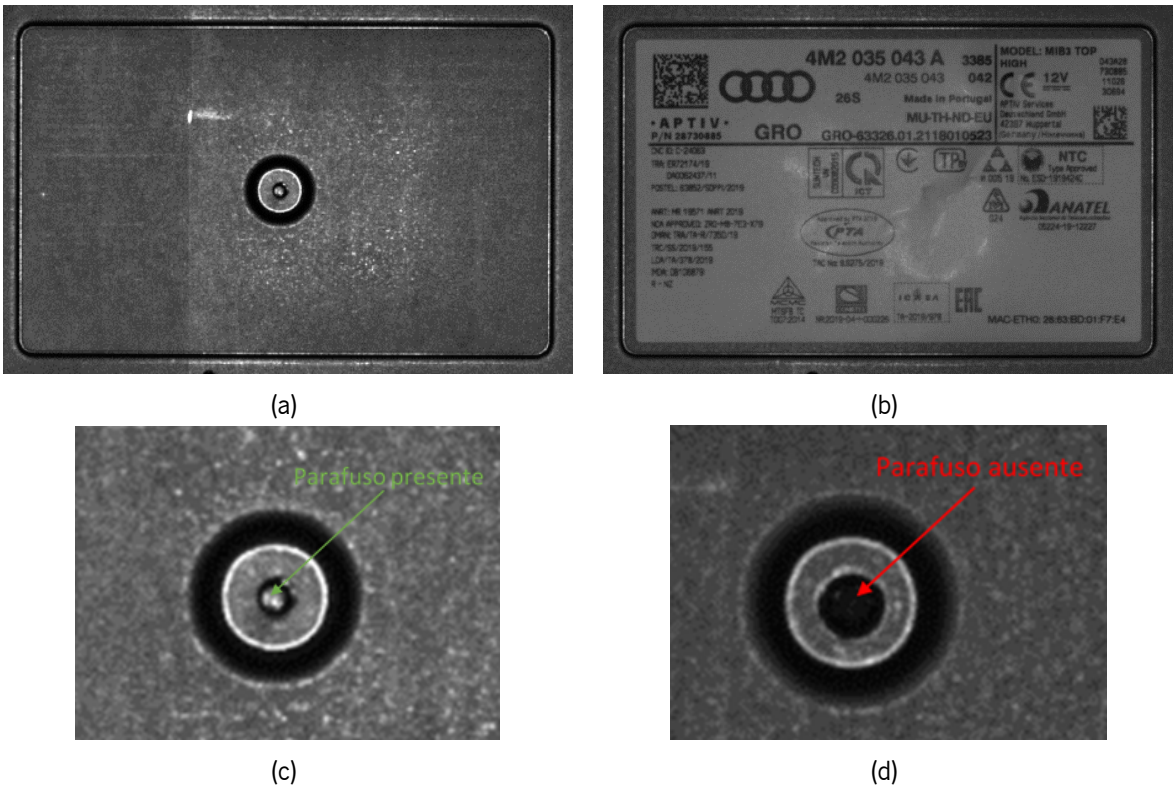


Figura 3.4: Exemplos de casos de: (a) e (b) sucesso na inspeção da colocação de *foils*; (c) falha na inspeção da colocação de *foils*.

3.3 Deteção da Presença de Parafusos

O problema que se segue consiste na deteção da presença ou ausência de parafusos nas imagens, como apresentado na Figura 3.5a. Nestas imagens, pretende-se detetar não a cabeça do parafuso, mas antes a ponta deste. Considera-se a inspeção como sucesso quando é verificada a presença do parafuso, partindo da comparação entre a intensidade média do local onde o parafuso é suposto se encontrar e a intensidade média na vizinhança exterior ao buraco do parafuso, tal como demonstrado na Figura 3.5c. Uma falha na inspeção pode ser consequência de duas situações: o parafuso está ausente ou o parafuso está presente mas mal colocado. Como ambas as situações resultam no escurecimento do local do parafuso, demonstrado pela imagem da Figura 3.5d, estas são tratadas da mesma forma e portanto não distinguidas entre si.

É também necessário ter em consideração que, dentro do conjunto de imagens, existem algumas imagens onde toda a região de análise se encontra tapada por um autocolante pertencente a um passo posterior ao desta inspeção, no processo de manufaturação, como por exemplo na imagem da Figura 3.5b. Tais imagens devem ser assinaladas e invalidadas para o processo de inspeção.



(a) (b)

(c) (d)

Figura 3.5: Exemplo de (a) imagem para identificação de presença/ausência de parafuso; (b) obstrução do parafuso por autocolante; (c) presença do parafuso; (d) ausência do parafuso.

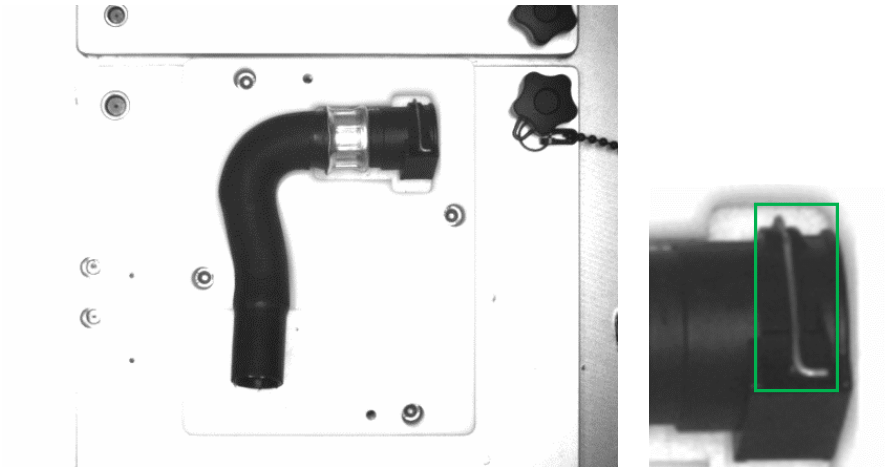
3.4 Inspeção do Posicionamento e Dimensões de Peças

O problema que se segue está associado à inspeção da peça apresentada na Figura 3.6a. Pretende-se que sejam examinadas algumas dimensões associadas à peça. É requerida a verificação inicial de que a peça foi montada na posição correta, partindo da deteção do pino de referência existente no conector marcado na Figura 3.6b. No evento de que o pino está ausente, a respetiva peça deve ser considerada como falha na tarefa de inspeção.

A peça pode ser dividida em três partes: o conector, a anilha ou casquilho e o tubo. A respeito das dimensões pretendidas, são feitas várias medições, estando estas definidas na Figura 3.7. É realizado o cálculo das seguintes distâncias:

1. Distância do início do conector até ao início da anilha;
2. Largura da anilha;
3. Comprimento do tubo;
4. Distância entre a borda inferior do conector e o pino de referência;
5. Comprimento da peça no eixo horizontal.

Em relação às distâncias 1 e 2, estas requerem a medição ao longo do ponto médio do conector e da anilha, respetivamente. A distância 3 deve ser medida também ao longo do ponto médio, no entanto é necessário ter em consideração que existe uma curvatura em grande parte do tubo. O comprimento do tubo deve portanto ser medido paralelamente às paredes da parte inferior do tubo, visto estas serem



(a) (b)

Figura 3.6: Inspeção de posicionamento e dimensões: (a) Objeto de inspeção de distâncias; (b) Marcação do pino de referência da peça.

retilíneas, ao contrário da restante porção do tubo que apresenta curvatura, que poderia interferir com o processo de medição para o cálculo da distância 3. A distância 4 considera o comprimento mínimo existente entre o pino utilizado como referência de orientação correta e a borda inferior da forma do conector. Por fim, a distância 5 é referente a todo o comprimento da peça, no eixo horizontal.

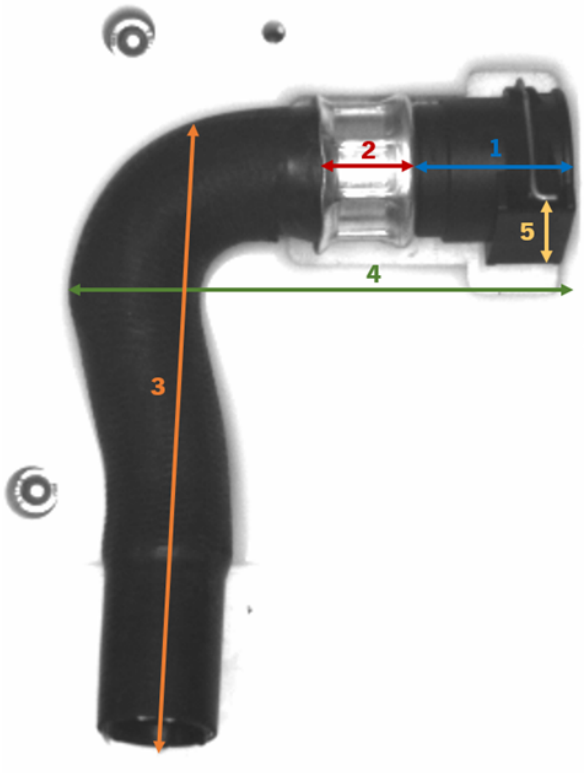


Figura 3.7: Indicação das distâncias a medir na inspeção de posicionamento e dimensões.

3.5 Object Picking

O problema de *object picking* consiste na inspeção de peças que passam num tapete, onde existe um *robot* com uma garra que pega nas peças que seguem os requisitos de posição e orientação correta. Em cada imagem é requerida a deteção de todas as peças que possuem a posição e orientação apropriadas para poderem ser agarradas pelo *robot*, sendo que, para essas peças, se pretende fornecer as coordenadas necessárias para o *picking*. Neste problema de *picking*, são abordados 3 tipos de peças diferentes, casquilhos/anilhas, conectores e tubos, apresentados na Figura 3.10.

(a) (b) (c)

Figura 3.8: Peças para *object picking*: a) anilha; (b) conector; (c) tubo.

É desejado primeiramente que haja a deteção de todas peças presentes na respetiva imagem. Para todas as ocorrências resultantes da deteção, deve existir a verificação de que estas se encontram devidamente orientadas, excluindo todas as que não seguem o respetivo critério de orientação. Ainda, para os tubos e conectores, é adicionalmente requerida a averiguação de que a zona onde o *picking* é realizado carece de quaisquer objetos potenciais de obstrução à tarefa de *picking*. Esta zona é uma área retangular, pré-estabelecida pelo operador, que consiga abranger toda região do tubo e vizinhança que o *robot* necessita de ocupar. No caso das anilhas, o *robot* agarra na peça a partir do interior desta, tornando-se portanto irrelevante a execução deste passo. Nota-se que, para este problema em específico, uma imagem não é considerada como sucesso ou falha na validação de elegibilidade para *picking*, mas as peças individualmente.

3.5.1 Casquilhos

No caso de *picking* de anilhas/casquilhos, apresentado na Figura 3.9, devido ao facto de o casquilho ser um objeto cilíndrico em que as bases possuem forma diferente, é necessário distingui-las para ser realizado o *picking*, pois em apenas um dos casos é pretendido que o robot agarre a peça. É de notar ainda que a posição de cada anilha relativamente ao centro da imagem afeta como a respetiva anilha é percecionada. Esta característica é um fator relevante a ter em conta, visto não ser possível descrever as anilhas mais afastadas do centro da imagem, a partir a criação de um modelo de *matching* utilizando apenas uma anilha de referência centrada espacialmente. Em casquilhos longe do centro da imagem, a iluminação provoca uma reflexão bastante pronunciada na parte correspondente ao interior da superfície lateral do objeto, enquanto que, quanto mais perto este está do centro do foco da câmara, menos se visualiza o efeito referido.

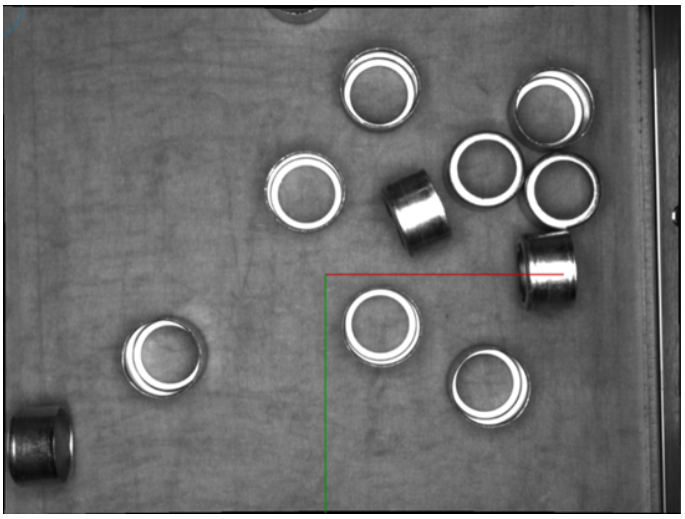
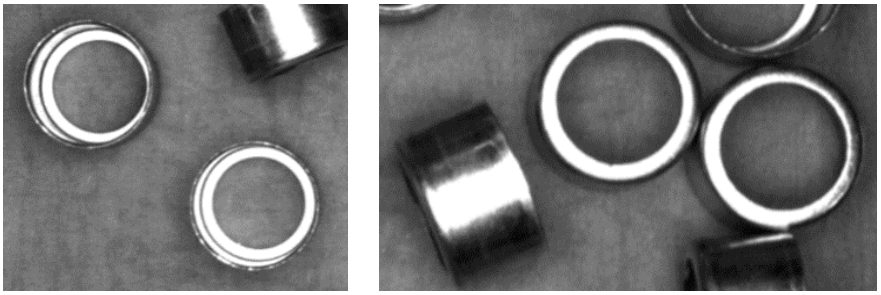


Figura 3.9: Exemplo de deteção de anilhas para picking.

Relativamente ao cenário em que o *robot* deve agarrar o casquilho, este decorre quando o casquilho se encontra da forma exemplificada pela Figura 3.10a. Observa-se que a anilha tem apenas uma base anelar, enquanto que do lado oposto existe apenas a circunferência da superfície lateral. A distinção entre as possíveis orientações das anilhas é realizada a partir da face que contacta com o tapete. Casquilhos que estejam corretamente orientados, como na Figura 3.10a, possuem a base anelar assentada no tapete, enquanto que casquilhos impropriamente orientados, tal como a Figura 3.10b apresenta, têm a base anelar voltada para a câmara ou então apresentam-se apoiadas com a superfície lateral no tapete.



(a) (b)

Figura 3.10: Exemplos das possíveis disposições dos casquilhos: (a) na orientação desejada; (b) nas orientações não desejadas.

3.5.2 Conectores

Para o *picking* de conectores, exemplificado pela Figura 3.11, pretende-se que sejam identificados todos os conectores presentes dentro da região branca. A deteção dos conectores deve assegurar que o pino de referência do respetivo conector se encontra vísivel na imagem e que não exista nenhum objeto dentro da área onde o *robot* agarra a peça que possa obstruir a tarefa.

Primeiramente, a presença de um pino de referência visível para cada um dos conectores encontrados na imagem é necessária. O pino apenas é visível quando a orientação do conector é a desejada, como

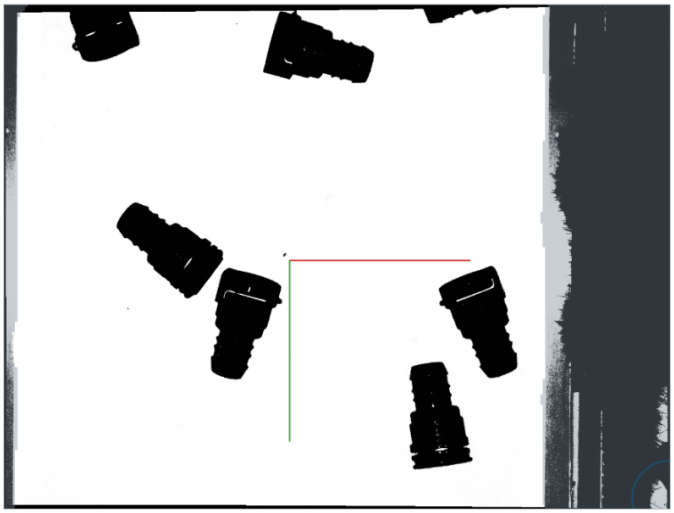


Figura 3.11: Imagem para *picking* de conectores.

demonstrado na Figura 3.12a. Quando o oposto acontece, ou seja, a orientação do conector é inadequada, o conector não apresenta o pino visível, tal como se sucede na Figura 3.12b. Esta condição de visibilidade do pino de referência de orientação deve servir como primeiro passo de exclusão dos conectores que não devem ser elegíveis para o *picking*.

(a) (b)

Figura 3.12: Identificação do pino de referência do conector: (a) pino presente; (b) pino ausente.

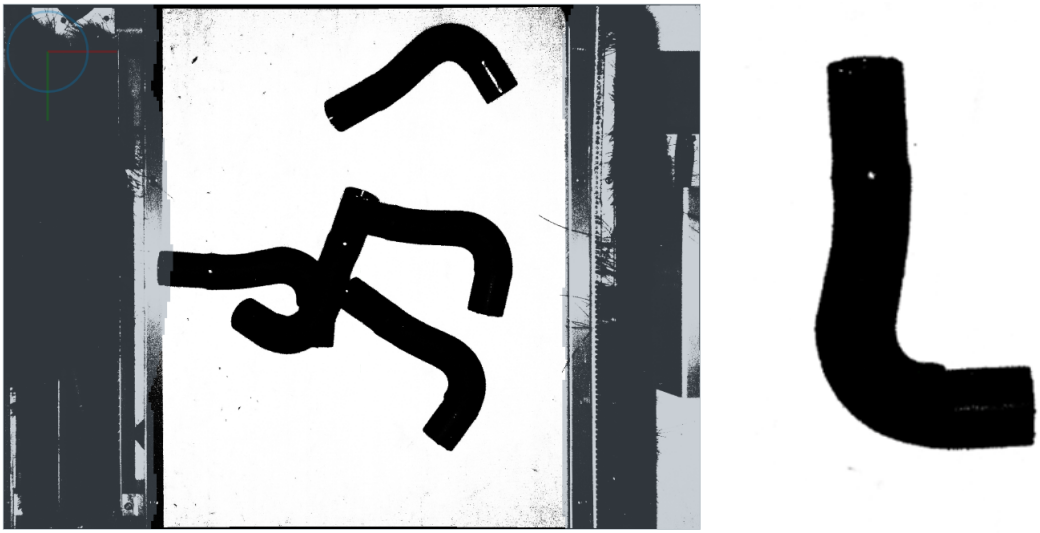
Estando a primeira condição verificada, existe ainda outro passo de exclusão de conectores, sendo este a verificação da violação da área de *picking*. É preciso confirmar que o *robot* consegue agarrar o conector sem qualquer problema. Isto significa que, na região onde a garra do *robot* irá pegar no conector, não podem existir quaisquer obstruções, exemplificado na Figura 3.13a. Em situações similares à representada pela Figura 3.13b, o conector deve ser descartado da tarefa de *picking*, devido à obstrução por parte de outra peça.

(a) (b)

Figura 3.13: Verificação da violação da área de picking: (a) não violada; (b) com obstruções.

3.5.3 Tubos

A deteção de tubos (Figura 3.14a), semelhantemente aos conectores, necessita inicialmente da verificação da disposição destes e, para as peças bem orientadas, a confirmação de que a zona de *picking* não tem nenhuma obstrução. A orientação dos tubos é mais simples, relativamente aos conectores, visto ser apenas preciso verificar qual é a forma do tubo, se é com a mesma que a apresentada na Figura 3.14b, exibindo a orientação correta ou invertida horizontalmente, sendo desprezadas no ato de *picking*.



(a) (b)

Figura 3.14: (a) Exemplo de imagem para *picking* de tubos; (b) exemplo de tubo corretamente orientado.

Relativamente à área de *picking*, representado pelo retângulo na Figura 3.15a, esta deve ser definida numa das extremidades do tubo, da mesma forma que acontece para os conectores. Tubos que não tenham essa região vazia, como na Figura 3.15b, onde existe outro tubo a ocupar o retângulo marcado, além do tubo em análise, devem ser excluídos como candidatos para o *picking*. Para todos os tubos que cumpram ambas as condições, orientação e não violação da área de *picking*, devem ser extraídas as respetivas coordenadas necessárias, de modo a indicar onde pode ser realizado o picking.

(a) (b)

Figura 3.15: Exemplos de zona de *picking* em tubos: (a) não obstruída e (b) com objetos não desejados.

3.6 Sumário

Neste capítulo foram abordados 5 problemas críticos de IVA. A inspeção de dispensação de cola consiste na verificação da quantidade de cola que foi dispensada num total de 22 locais, sendo que uma imagem deve ser considerada como falha quando pelo menos um ponto de cola não cumpre os limites mínimos e máximos de quantidade de cola exigidos. Para a inspeção de cabos fita, é pretendida a averiguação da existência de 2 faixas brancas existentes em cada cabo, em 2 tipos de imagens, sendo que quando é detetada a segunda faixa e esta se encontra parcialmente visível acima de uma limite máximo de largura ou totalmente visível, o respetivo cabo deve ser marcado como tendo sido indevidamente colocado na linha de montagem. Passando para a deteção da presença de parafusos, devem ser distinguidos os casos de sucesso de casos de falha partindo do escurecimento da área do parafuso relativamente à sua vizinhança. Devem também ser descartadas da tarefa de inpeção todas as imagens que possuem um autocolante que tapa a área de análise. Com o problema de inspeção do posicionamento e dimensões de peças, é inicialmente pretendida a verificação de que a peça foi colocada na posição correta e posteriormente a medição de vários comprimentos, para confirmar que a peça foi corretamente montada. Por fim, para o problema de *object picking*, pretende-se a deteção de 3 tipos de peças, casquilhos, conectores e tubos, a verificação de que estas peças se encontram corretamente orientadas e para aquelas que cumprem os requisitos de orientação, a extração das coordenadas relevante para a realização da tarefa de *picking*. Para os casos dos conectores e tubos, é também necessário ser confirmado que a zona onde a garra irá pegar nas peças se encontra livre de quaisquer objetos que possam interferir com o *picking*.

Capítulo 4

Implementação das Soluções

O presente capítulo começa por descrever o que é o registo de imagens, a importância deste passo em sistemas de IVA e como este se realiza. De seguida, são descritas as soluções implementadas como modo de resposta aos problemas de inspeção definidos no Capítulo 3 e é explicado o raciocínio para cada passo dos fluxos de operações.

4.1 Registo de Imagens

Durante o processo de inspeção visual, nas imagens capturadas poderão existir pequenas ou grandes diferenças na localização das estruturas nelas presentes. Adicionalmente, em problemas em que existem vários componentes iguais a detetar e inspecionar, estes podem estar naturalmente espalhados pela imagem. Estes dois exemplos de situações geram problemas na reprodutibilidade de processos de inspeção visual automáticos. O registo de imagens permite efetuar o alinhamento entre duas imagens, tornando-se num passo importante quando os objetos das imagens não se encontram espacialmente fixos [27, 28]. Com o registo, pretende-se alinhar as estruturas de interesse de um conjunto de imagens através da alteração dos objetos por meio de transformações não lineares, de forma a minimizar a diferença entre estes relativamente ao *template* [29]. A necessidade de as imagens estarem alinhadas entre si baseia-se em permitir a deteção de diferenças entre estas, assim como a simplificação da resolução dos problemas. O registo pode ser utilizado como um passo primário no fluxo de operações de um problema, permitindo que seja possível isolar os objetos de interesse partindo de ROIs, eliminando interferências de partes irrelevantes das imagens para o processo de alinhamento. Facilita também a generalização da definição de regiões necessárias para execução de operadores, independentemente da imagem em estudo.

O registo de imagens consiste em 3 passos principais: a localização de pontos de referência na imagem em análise, a criação de uma matriz de transformação espacial e a realização do alinhamento da imagem a partir da matriz de transformação.

No primeiro passo, é realizado uma procura de *shape-based matching*. A operação de *matching* localiza o objeto utilizado como referência na imagem. Este objeto deve ser tal que se encontre presente e facilmente identificável em todas as imagens. A localização do objeto nas imagens a serem analisadas fornece a informação indicativa de como a imagem em análise se apresenta, em termos de translação,

rotação e escala, apontando para o que deve ser feito de forma a que o posicionamento da imagem de teste esteja em concordância com a imagem de referência. Como forma de clarificação, foi aplicado registo de imagens para imagens em que é necessário localizar mais do que um objeto e alinhar a imagem para cada objeto encontrado. Para a criação de um modelo de *matching*, o espaço de trabalho da imagem é reduzido com uma ROI, na Figura 4.1a, ou conjunto de ROIs que contêm os pontos a serem utilizados como referência para a procura por *matching* nas imagens. Dentro destas ROIs, são selecionados os contornos dos objetos e é criado um modelo de *matching*, resultando na Figura 4.1b.

Após a execução da procura do modelo de *matching* numa nova imagem, são encontrados os objetos pretendidos, como demonstrado na Figura 4.1c e na Figura 4.1e, e partindo das coordenadas e ângulo de rotação do modelo encontrado, assim como das coordenadas e ângulo de referência, é criada uma matriz de transformação espacial para mapear a imagem em análise, alinhando-a com a imagem de referência. Com a transformação afim, o objeto que foi encontrado pelo passo de *matching* nas Figuras 4.1c e 4.1e pode ser alinhado com o objeto de *template* como demonstrado pela Figura 4.1d e Figura 4.1f, respetivamente.

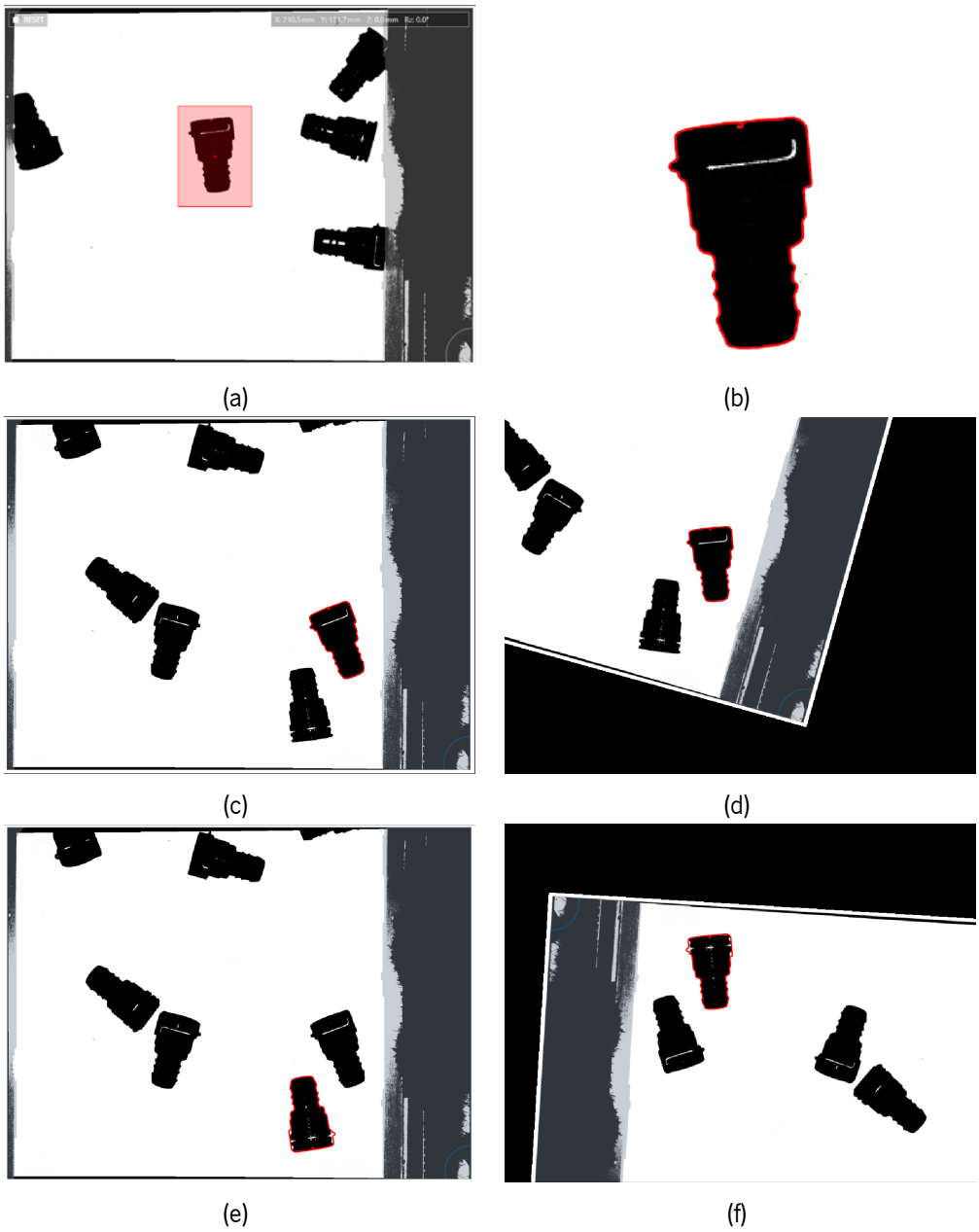


Figura 4.1: Passos do registo de imagens: (a) ROI de seleção de objeto de *template*; (b) forma do modelo criado; (c) e (e) localização de objetos encontrados pelo modelo de *matching*; (d) e (f) alinhamento da imagem (c) e (e), respetivamente.

4.1.1 Template Matching

Template matching consiste na localização de objetos em imagens, com a criação de um modelo a partir de um template e a posterior procura desse modelo noutras imagens. O modelo é obtido a partir de uma imagem de referência, extraindo pontos chave, nomeadamente bordas e cantos, dadas as diferenças de contraste mais acentuadas nestes casos.

O processo de *matching* começa com a redução do espaço de trabalho de uma imagem de referência a partir de ROIs adequadas que encapsulam os objetos, para se derivar o *template* utilizado pelo modelo desta operação, como verificado pela Figura 4.1a. A escolha de uma ROI adequada influencia a criação do modelo e como este se comporta, se tem uma performance ótima ou não. Por esta razão, é fundamental

que a ROI que será utilizada para o modelo de *matching* permita identificar as estruturas corretas para a procura. É de notar que, consoante a diminuição das dimensões de uma ROI, mais rápido é o operador que a aplica, devido a reduzir o domínio da imagem ou região, reduzindo consequentemente o espaço em que um operador é executado. Para os operadores de *matching*, uma ROI mais pequena implica um espaço de procura por correspondências do *template* também menor. Continua-se com um passo de isolamento dos pontos chave utilizados como *template* para a criação do modelo, importante para a remoção de estruturas que não sejam relevantes ou constantes em todas as imagens do problema em análise. Com a remoção de pontos irrelevantes a partir do apoio de algumas operações, como por exemplo o uso de transformações morfológicas, o objeto de referência pode ficar reduzido às bordas da sua forma, como na Figura 4.1b.

Segue-se a criação de modelos e a otimização dos parâmetros de procura. Numa operação de *matching*, é utilizado pelo menos um objeto, que serve como *template* para a criação de um modelo que o represente. Este objeto de referência deve existir e ser facilmente localizado em todas as imagens onde se pretende efetuar o processo de alinhamento. A procura do modelo de *matching* vai localizar todas as correspondências encontradas na imagem em estudo, obtendo-se as respetivas coordenadas centrais e ângulo de rotação de cada uma das correspondências. Adicionalmente, é indicado ainda um *score* de semelhança entre o objeto encontrado e a referência. Consoante a necessidade do uso de um modelo capaz de detetar objetos em escalas diferentes, existe ainda a possibilidade da explicitação das escalas.

4.1.2 Operadores de Matching Disponíveis no HALCON

Devido à importância do registo de imagens no desenvolvimento de soluções de IVA e a utilização de *matching* para a localização inicial dos objetos utilizados como *templates* para se poder alinhar as imagens, é relevante explorar os vários operadores disponíveis no *HALCON* para o *matching*, descritos em [11] e [30]. O processo de *matching* pode ser abordado de diferentes maneiras, de acordo com as caraterísticas que cada operador possui. A escolha do operador de *matching* depende da imagem e das caraterísticas dos objetos que se pretende encontrar. São possíveis distinguir as imagens em 2 grupos primários, se as imagens foram capturas a partir de uma vista ortogonal ou de uma vista perspetiva. Para imagens capturadas com vista ortogonal, podem ser considerados operadores de *shape-based matching*, *correlation-based matching*, *component-based matching* e *local deformable matching*, ou seja:

- O operador de *shape-based matching* descreve um objeto a partir dos contornos da sua forma, sem ter em conta a intensidade dos píxeis ou da vizinhança. A forma é extraída selecionando todos os pontos cujo contraste da vizinhança excede um certo limite mínimo, visto que normalmente esses pontos fazem parte do objeto. As formas dos objetos são utilizadas tanto para a geração do modelo como para a realização da operação de *matching*. Este tipo de *matching* permite a utilização simultânea de múltiplos modelos na execução da procura. Da execução deste operador podem ser encontradas várias instâncias que tenham partes em falta ou ligeiramente deformadas, desfocadas e afetadas por ruído, sendo que os objetos são encontrados também para casos em que estes se encontram rodados ou com escala diferente da escala do *template* usado. Esta abordagem pode ser aplicada em imagens a cores ou em níveis de cinzento.

- O operador de *correlation-based matching* utiliza correlação cruzada normalizada de forma a avaliar, a partir dos valores de cinzentos dos píxeis, a correspondência entre a imagem de análise e o modelo previamente criado. Permite compensar variações tanto aditivas como multiplicativas na iluminação e, ao contrário do *shape-based matching*, *matching* por correlação encontra também objetos com pequenas alterações na forma ou textura e ainda objetos com contornos mais desfocado. No entanto, tem como desvantagens não conseguir localizar objetos com escala diferente da original e apenas pode ser aplicado a imagens em níveis de cinzento.

- O *component-based matching* aborda o problema de correspondência, de forma análoga ao *shape-based matching*, no sentido em que extrai os contornos dos objetos para a criação de modelos de *matching*. Tem como vantagem considerar que um objeto é constituído por várias partes que se podem transladar e rodar entre si. Um objeto composto por partes é abordado num único passo de procura, aumentando a robustez do modelo e da procura, quando comparado com uma abordagem que resolva cada uma das partes do objeto individualmente, com modelos diferentes. Contrariamente ao operador de *matching* por formas, este operador não deve ser aplicado quando as imagens se encontram desfocadas ou quando existem deformações nos objetos.

- *Local deformable matching* baseia-se também em *shape-based matching*, sendo que os objetos encontrados não estão restringidos a possuir a mesma forma que os objetos de referência aplicados para a criação dos modelos, permitindo deformações mais acentuadas. Os objetos podem ser encontrados mesmo estando ligeiramente deformados, onde a orientação e escala destes são consideradas como aspetos que fazem parte da deformação existente. É calculado um campo vetorial que descreve as deformações, assim como uma versão retificada da imagem de procura.

Relativamente ao segundo tipo, o *HALCON* contém 2 operadores para realizar o *matching*, *perspective deformable* e *descriptor-based*. Em ambos os operadores pode ser utilizada uma câmara calibrada ou não calibrada. Estes operadores devem ser capazes de encontrar objetos quando existe a possibilidade de que estes estejam deformados relativamente à sua forma apresentada na imagem capturada. Os algoritmos de *matching* para vista perspetiva são:

- *Perspective deformable matching* extrai os contornos da região da imagem utilizada e corresponde os contornos com as formas dos modelos previamente criados, como no *matching* baseado em formas. Tal como acontece no *local deformable matching*, deformações acentuadas são resolvidas, sendo que em vez de uma pose, é calculada uma matriz de transformação projetiva. O operador permite também obter uma versão calibrada.

- *Descriptor-based matching* difere do *matching* deformável no aspeto em que o *template* do modelo criado não se baseia em formas, mas num conjunto de pontos de interesse, que são primeiramente extraídos por um detetor e de seguida descritos por um descritor, sendo classificados de acordo com a sua localização e níveis de cinza da vizinhança. Este operador revela-se mais indicado para ser aplicado em objetos com muitas texturas, não devendo ser utilizado em objetos com poucas texturas e cantos arredondados. Relativamente à outra opção de *matching* para vista em

perspetiva, é mais rápido para espaços de procura maiores, tendo a desvantagem de ser menos preciso.

4.1.3 Representação em Pirâmide

O *matching* pode usufruir de uma representação em pirâmide, de forma a diminuir o tempo de procura. A pirâmide, retratada na Figura 4.2, é uma representação multi-escala de uma imagem, em que cada nível desta corresponde ao resultado do *downsampling* da imagem do nível anterior, partindo da imagem de procura na sua resolução original e sendo esta considerada como o nível 0 [11].

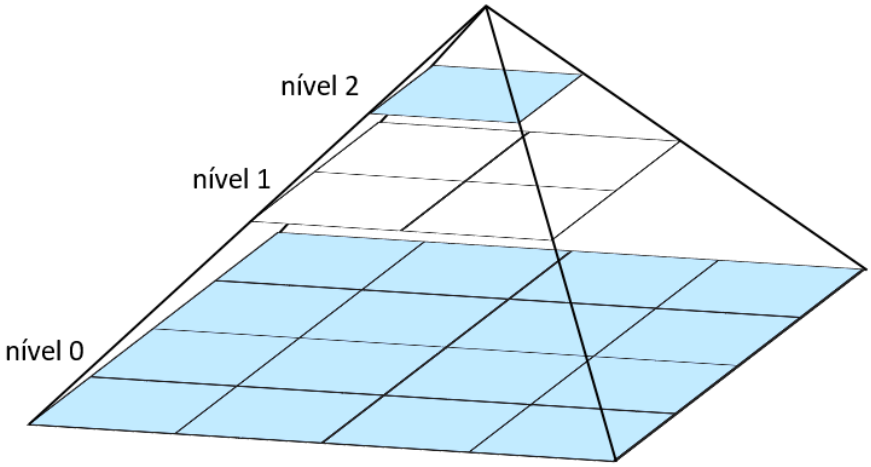


Figura 4.2: Representação em pirâmide.

A procura de um modelo começa no nível mais elevado da pirâmide, correspondente à imagem com menor resolução. Os modelos utilizados em níveis mais altos da pirâmide contêm menos pontos relativamente a níveis mais baixos. Isto acontece porque o *downsampling* de uma imagem provoca uma redução do contraste das suas estruturas, desfocando as bordas das estruturas. Assim, para a utilização de representação em pirâmide, modelos derivados de estruturas maiores, com bordas bem definidas e elevado contraste, tem geralmente maior sucesso [11, 31]. Para que seja possível identificar as estruturas relevantes, é necessário que estas ainda tenham contraste suficiente para serem distinguidas do plano de fundo da imagem [32]. Objetos mais pequenos e mais finos desaparecem em níveis mais baixos da pirâmide, em comparação com objetos grandes e mais robustos [32, 33]. Em problemas onde se pretende identificar estruturas que já são pequenas na imagem original, a procura em vários níveis da pirâmide não é tão benéfica, pois os objetos apenas serão visíveis nos níveis mais baixos.

4.2 Resolução dos problemas

São de seguida apresentadas as soluções desenvolvidas para os problemas de IVA enunciados, considerando as bases de dados disponíveis, de forma a ir de encontro aos objetivos definidos. São utilizadas figuras para ilustrar o que é pretendido obter nos passos ao longo de cada fluxo de operações para cada um dos problemas abordados.

4.2.1 Inspeção de Dispensação de Cola

De forma a executar o fluxo de operações para solucionar o problema de inspeção de dispensação de cola, o utilizador deve previamente definir a região ou regiões a utilizar como referência para o alinhamento da imagem e as ROIs para cada um dos respetivos 22 locais de dispensação que se pretende inspecionar (como apresentado na Figura 4.3a). No sentido de maior simplicidade, devem ser definidas regiões quadradas, apenas localizando as coordenadas do ponto central de cada uma e um tamanho das regiões apropriado, grandes o suficiente de forma a que os pontos pudessem ser identificados no passo seguinte de *matching*, mas que não abrangessem demasiadas estruturas irrelevantes para a solução, para evitar interferências na execução das operações morfológicas e de escolha de regiões. Como se deve considerar que a cola pode ser dispensada de forma diferente em cada ponto, deve-se poder escolher o valores de limite inferior e superior de quantidade de cola dispensada. Ainda, o utilizador deve também ter a escolha do uso de binarização por *threshold* variável (de forma automática) ou introduzir valores de *threshold* específicos para cada ponto, no caso do uso da binarização por *threshold* tradicional.

É primeiramente realizado o registo da imagem, alinhando-a com a referência de forma a que o conjunto das 22 regiões esteja devidamente centrado nos respetivos pontos de cola. O registo da imagem nesta solução é particularmente importante, porque mesmo com um alinhamento ideal, os pontos de cola não se encontram exatamente no mesmo local de imagem para imagem. Alinhamentos menos ótimos podem ter consequências para o cálculo da da quantidade de cola, por haver a possibilidade de as regiões de cola ficarem cortadas pelas ROIs usadas. Os passos seguintes são executados para efetuar o cálculo da quantidade de cola, em cada uma das regiões onde a cola foi dispensada. Começa-se por procurar a instância de cola dentro da ROI e retiradas as coordenadas desta, a partir de *template matching* com escala. O modelo com escala é necessário para que se possa encontrar correspondências com diferentes tamanhos, devido às diferentes quantidades de cola dispensada. No caso de não ser encontrada nenhuma correspondência dentro da ROI em análise, o local é considerado como situação de falha, porque não foi dispensada cola ou foi dispensada em quantidade demasiado reduzida para ser detetada.

Como as ROIs não podem ser demasiado amplas, e tendo em conta que a cola pode não ser dispensada sempre no mesmo local com precisão, causando desvios nos pontos de cola, existe a possibilidade de escolha da criação de uma nova ROI que se vai adaptar ao ponto em análise. Esta nova ROI é definida com base nas coordenadas resultantes do processo de *matching*, de forma a centrar a região no ponto. Com a região centrada, pretende-se abranger a totalidade do ponto, evitando que este fique incompleto. Assim, evitam-se erros de cálculo do número de píxeis da região de cola e da comparação com os limites definidos para ser classificado como sucesso na inspeção.

Para cada ponto de cola detetado é utilizado um filtro *bottom-hat*, como na Figura 4.3b, para enfatizar os pontos de dispensação de cola, sendo escolhido um elemento estruturante circular de raio apropriado. Ao resultado da operação de *bottom-hat*, é aplicada uma binarização por *threshold* variável em cada uma das regiões respetivas aos pontos de cola, apresentada na Figura 4.3c. A binarização pode ser realizada com um operador de *threshold* variável, calculado automaticamente, para adaptação à região atual a ser inspecionada ou então pode ser feita com valores mínimos e máximos de *threshold* individuais para cada ponto, manualmente introduzidos pelo utilizador da solução. A razão pela possibilidade de binarização com *threshold* variável consiste no facto de que os intervalos de intensidades associados a cada um dos pontos de cola não são suficientemente semelhante entre regiões, não havendo a possibilidade da generalização da operação de binarização com valores mínimos e máximos pré-definidos pelo utilizador. Segue-se uma operação de preenchimento, ilustrado pela Figura 4.3d, para completar falhas dentro das regiões de modo a que a operação de *opening*, que segue o preenchimento, não altere as regiões que se quer analisar ao mesmo tempo que exclui as estruturas irrelevantes. Como a operação de *opening* por vezes não é suficiente para eliminar todos os elementos além dos correspondentes à cola, torna-se necessária uma escolha de regiões, por meio de comparação de coordenadas entre as regiões resultantes da abertura e as coordenadas resultantes do processo de *matching* do ponto em análise. O resultado obtido após as operações de *opening* e da escolha da região do ponto deve ser algo como a Figura 4.3e. Após a realização deste algoritmo de *bottom-hat*, binarização, preenchimento, *opening* e seleção de regiões para todos os pontos, obtém-se algo como a Figura 4.3f. Nota-se que a Figura 4.3f é um apenas um excerto do resultado total obtido, apresentando apenas 11 das 22 regiões de cola que são obtidas.

Para verificação das condições impostas como limites mínimo e máximo do tamanho de cada ponto de cola, é feita a contagem do número de píxeis que compõem cada região e a comparação com os limites impostos, sendo a imagem sinalizada como estando em incumprimento na inspeção caso algum dos pontos não esteja dentro dos valores definidos.

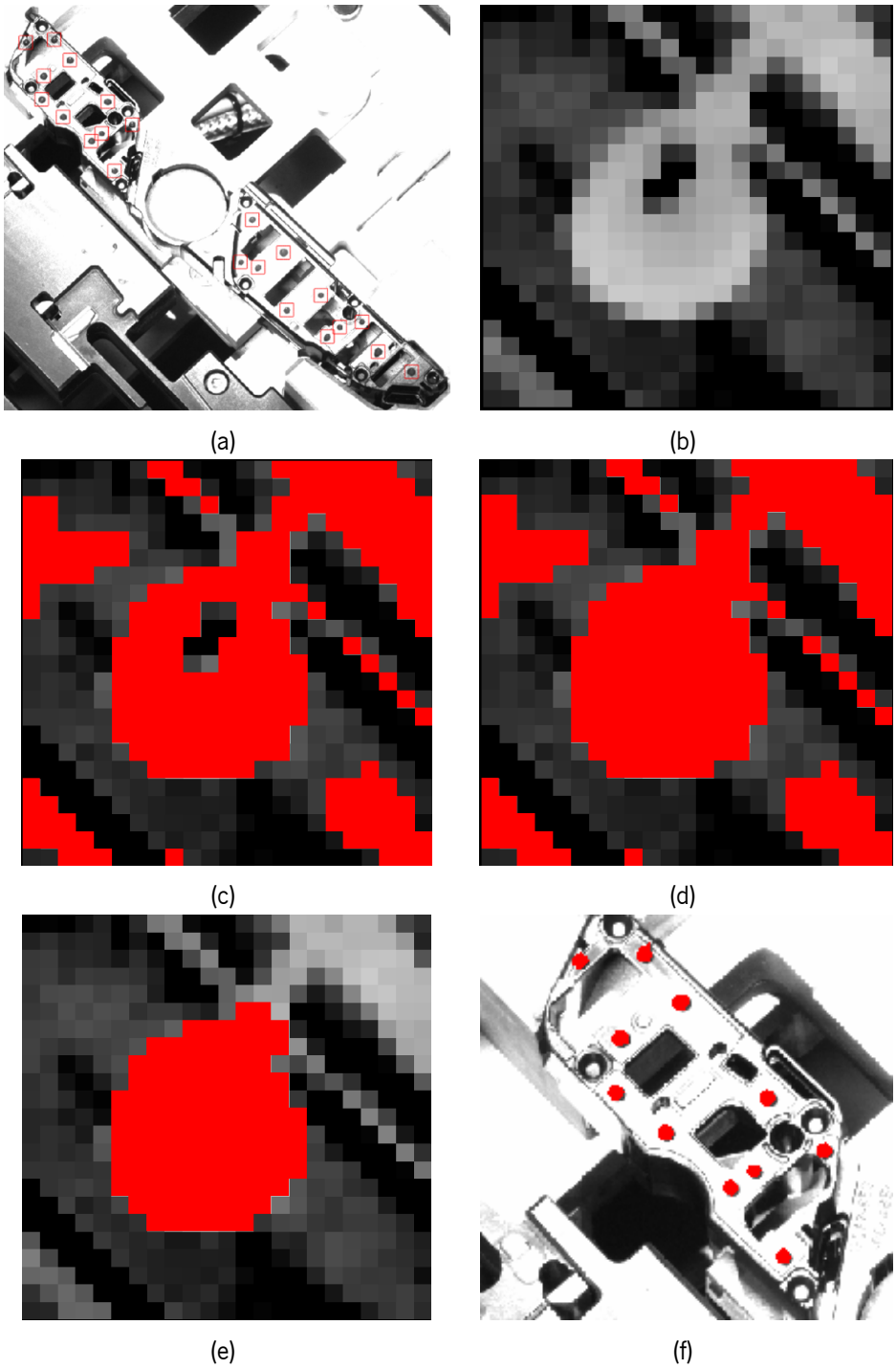


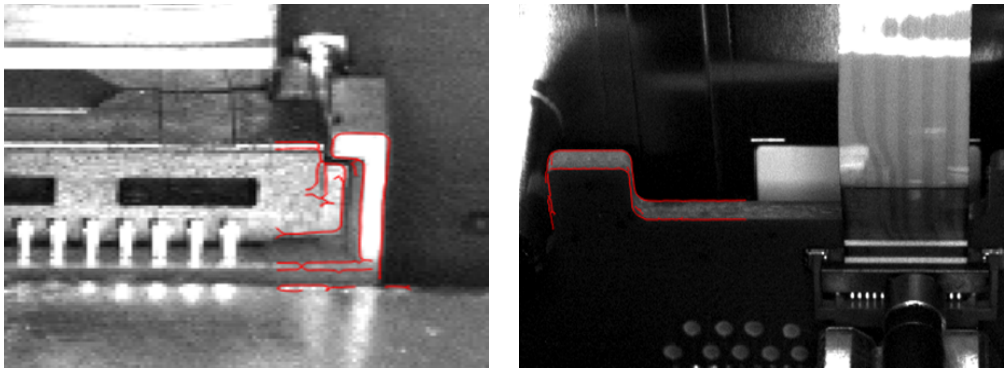
Figura 4.3: Imagens ilustrativas do fluxo da solução de dispensação de cola: (a) ROIs a inspecionar; (b) resultado do *bottom-hat*; (c) resultado após binarização de (b); (d) preenchimento das regiões resultantes de (c); (e) resultado da operação de *opening*; (d) conjunto de 11 pontos de cola segmentados com a solução.

4.2.2 Inspeção de Cabos Fita

Para a resolução deste problema, é previamente necessário treinar um modelo de *matching* para ser realizado o alinhamento das imagens, de forma a facilitar o processo de inspeção. É ainda essencial permitir que o operador arbitre uma percentagem máxima permitida da largura da segunda faixa branca

para que o respetivo cabo fita ainda passe na inspeção, relativamente à primeira faixa. No sentido de simplificação da inspeção, é requerida uma ROI que será utilizada para a redução da imagem para a região onde é previsto que cada um dos *foils* apresente as faixas brancas de referência.

Para se poder facilmente localizar as faixas, foi utilizado um objeto com bordas facilmente distinguíveis e que fosse constante ao longo de todo o conjunto de imagens, de modo a realizar o processo inicial de alinhamento. Para tal foi utilizado o objeto marcado na Figura 4.4a para as imagens do Tipo A e a estrutura marcada na Figura 4.4b para as imagens do tipo B.



(a) (b)

Figura 4.4: Objeto de referência para o *matching* da imagens da cabos fita do: (a) tipo A; (b) tipo B.

É aplicada a ROI para a redução do domínio de trabalho a ser utilizado nos restantes passos, previamente definida de acordo com a imagem de referência e marcada na Figura 4.5, garantindo a visibilidade total de ambas as barras, caso estejam presentes.

(a) (b)

Figura 4.5: (a) Marcação da ROI para a inspeção de *foils*; (b) ampliação da ROI.

Trabalhando apenas dentro da ROI, é primeiramente executada uma binarização por *threshold*, devido ao facto de as barras serem facilmente distinguíveis na ROI por possuírem em geral níveis de cinzento muito superiores às restantes estruturas da vizinhança. O resultado desta operação (Figura 4.6a) isola os píxeis mais claros da região. Como é desejado obter os píxeis mais claros, possíveis reflexões existentes no cabo causadas pela forma de iluminação para a captura da imagem, demonstradas na Figura 4.6b, tornam-se um problema necessário de resolver, visto que apesar da binarização resultar em elementos ligados correspondentes maioritariamente às barras. Por análise da Figura 4.6c, correspondente à ampliação do resultado da binarização, focada no problema de reflexão, observa-se que os conjuntos de píxeis que não são pretendidos se encontram ligados ao elemento da faixa, havendo a necessidade de os eliminar. Por meio de uma operação de *opening* com *kernel* retangular, com altura de apenas 1 píxel e uma

largura adequada, de forma a eliminar os píxeis não desejados, contudo sem modificar significativamente as regiões pretendidas, as regiões da barra e da reflexão são separadas. Seguidamente, é realizada uma escolha das regiões, por meio de comparação entre o comprimento de cada uma das regiões com limites previamente definidos, de forma a excluir quaisquer elementos ligados irrelevantes para a solução, após o *opening*, como por exemplo regiões resultantes de problemas de reflexão. Com os limites adequados, são eliminadas todas as regiões exceto as pertencentes às barras. O resultado esperado encontra-se exemplificado na Figura 4.6d.

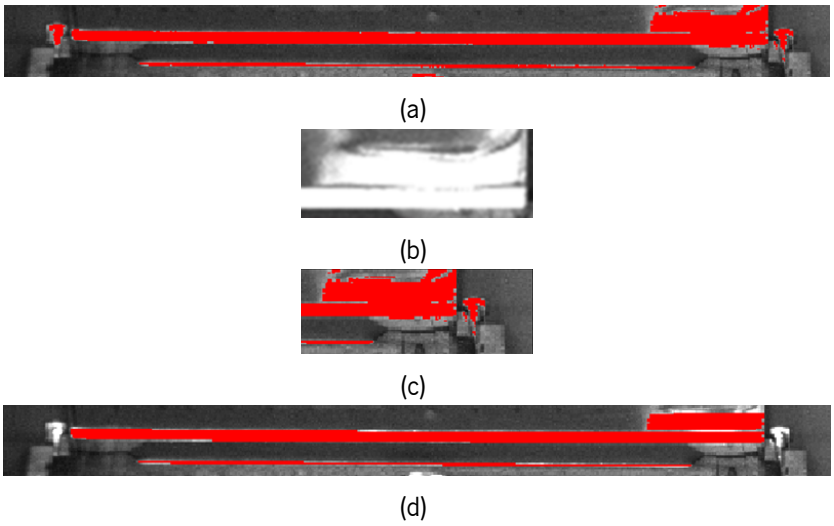


Figura 4.6: Passos da inspeção de cabos fita: (a) resultado da binarização por *threshold*; (b) e (c) identificação de problemas de reflexão; (d) resultado da operação de *opening*.

Após o isolamento das regiões pertencentes às faixas de análise, com a exclusão de todos os outros elementos inicialmente obtidos com a binarização, resta retirar informações acerca da presença da segunda faixa, no sentido de verificar se a largura desta se encontra abaixo ou acima do limite definido, indicativo de um bom ou mau encaixe do cabo no respetivo local. Caso o número de objetos resultantes da escolha seja superior a 1, assume-se que o cabo apresenta ambas as faixas de forma visível e é calculada a altura da faixa que determina a classificação da inspeção. No caso de a altura ser inferior ao *threshold* estabelecido, o cabo é considerado como bem colocado, passando na inspeção. Caso a altura seja igual ou superior, o contrário acontece, sendo que ocorreu um encaixe defeituoso.

Para a resolução deste problema, o método explicado é semelhante para ambos os tipos de imagens A e B, diferenciando-se apenas nos valores de *threshold* utilizados para a operação de binarização, no comprimento do elemento estruturante do *opening* e as comprimentos mínimo e máximo da escolha de regiões após a operação de *opening*. O mesmo acontece para os dois cabos de cada imagem do tipo A, adicionalmente com a criação de uma ROI para a análise de cada um dos cabos. Estes valores, tal como o valor de percentagem máxima da segunda faixa, podem ser alterados pelo utilizador.

4.2.3 Deteção da Presença de Parafusos

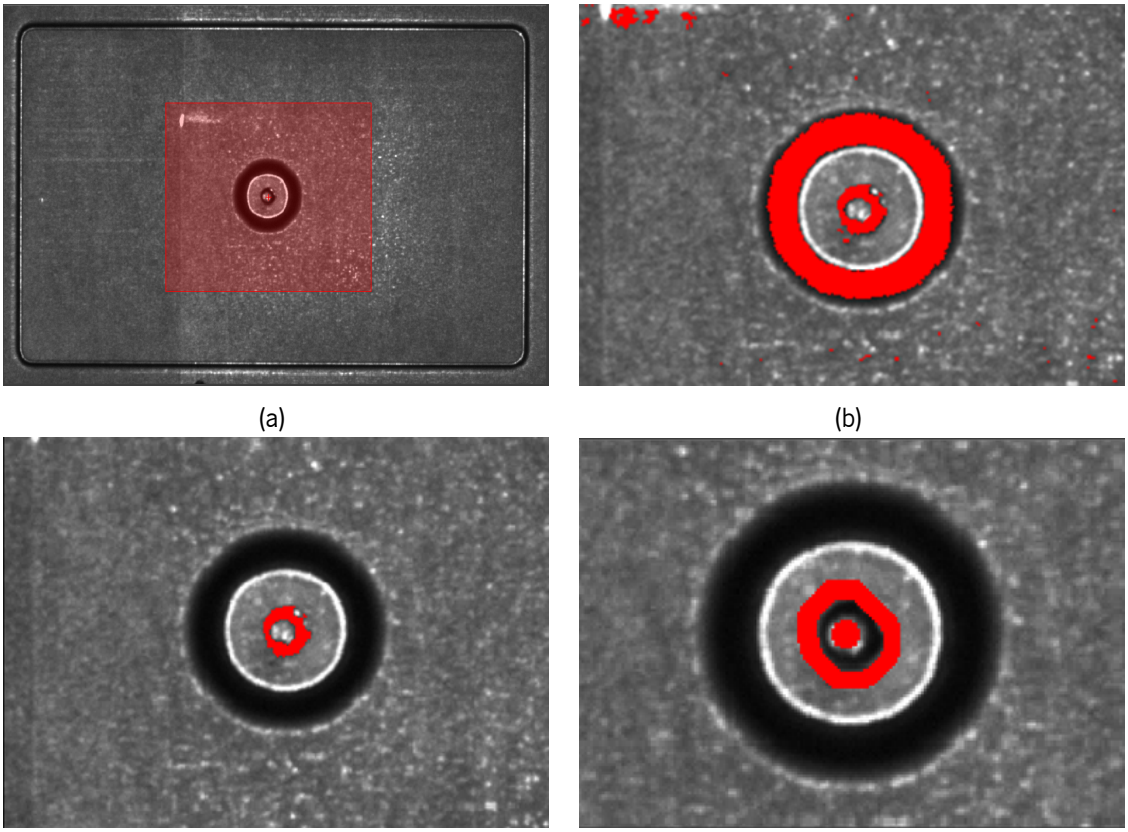
Para o desenvolvimento da solução para o problema de inspeção da presença de parafusos, não se revelou necessária a utilização de alinhamento e consequentemente *shape-based matching*, apesar de as imagens a inspecionar possuírem translações e pequenas rotações dos objetos presentes. Esta decisão baseia-se no facto de ser possível isolar o objeto utilizado como referência para localizar a região onde é efetuada a verificação da inspeção do parafuso. Além disso, as imagens associadas a este problema em específico revelam que todas as estruturas importantes para a execução deste fluxo apresentam forma circular e o parafuso encontrar-se-á no centro destas regiões, não havendo relevância acerca da rotação das imagens para igualar uma referência. Tendo isto em conta, é apenas a criação prévia de uma ROI capaz de incluir o local do parafuso e a sua vizinhança, como exemplificado pela Figura 4.7a.

Como os objetos de análise na região central da imagem, é utilizada uma ROI (Figura 4.7a) suficientemente larga para abranger todas os objetos para limitar o espaço de procura, mas que permita possíveis translações do objeto de estudo de imagem entre imagens. Trabalhando apenas na ROI definida, o próximo passo consiste numa binarização. Visto o conjunto de imagens apresentar variações significativas nos níveis de cinzento entre imagens, uma binarização por *threshold* com valores fixos não é uma abordagem ideal para este problema (algumas imagens podem ser significativamente mais claras ou mais escuras, outras apresentam problemas de reflexão, provocando regiões de píxeis com elevadas intensidades). Como alternativa, é utilizada a binarização por *threshold* variável para permitir a adaptação dos valores de *threshold* especificamente a cada imagem, adaptando os parâmetros de tamanho da máscara de modo a tentar eliminar o máximo de ruído associado ao *background* da imagem e escolher apenas as regiões escuras da região envolvente ao encaixe do parafuso. O resultado pretendido da binarização apresenta-se na Figura 4.7b.

Devido ao facto da binarização resultar em vários elementos ligados, em que alguns estão associados às irregularidades visíveis na superfície do objeto capturado, torna-se relevante realizar a seleção da região desejada. A seleção é feita tendo em conta o comprimento e a altura das regiões resultantes da binarização, para a elaboração de passos futuros no fluxo do algoritmo. Deve ser escolhida a região associada ao anel escuro que representa o espaço vazio à volta do parafuso, dentro do encaixe, obtendo-se algo como a região representada na Figura 4.7c. É de notar que, apenas com a binarização e a seleção dos elementos ligados, é possível discriminar as imagens que têm um autocolante a tapar o que se pretende analisar. Todas as imagens que caiam nesta situação são descartadas dos restantes passos do fluxo de operações. Trabalhando com a região da Figura 4.7c, é realizado um preenchimento com o objetivo de não se perder a forma da região na execução do passo seguinte. O preenchimento segue-se de uma abertura com *kernel* circular para ser obtida uma região melhor definida. É ainda realizada uma dilatação, também com *kernel* circular, seguida de uma subtração entre a região resultante da dilatação e a região resultante da abertura. Este conjunto de operações tem como produto uma região em anel como a definida na Figura 4.7d.

Para verificar se o parafuso está ou não presente e bem colocado, foi considerado que, para um caso de sucesso, a superfície do parafuso tem em geral píxeis com níveis de cinzento semelhantes à região envolvente, onde o parafuso encaixa. Para o parafuso ser considerado como presente, é realizada

comparação entre a média das intensidades dos píxeis da região anelar na superfície do encaixe do parafuso, assim como a média das intensidades dentro de um pequeno *kernel* circular centrado dentro desse anel. De forma a se obter a intensidade média do parafuso, é primeiramente calculado o centro da região em anel para alinhar este *kernel* circular no local onde é suposto o parafuso estar presente. Ambas as regiões para o cálculo de cada uma das médias estão clarificadas na Figura 4.7d, sendo a região mais pequena e circular, o local do parafuso e a região maior, parte do encaixe do parafuso. As intensidades médias são comparadas de forma a comentar acerca da presença ou ausência do parafuso, podendo também ser utilizado um valor de desvio para permitir uma maior margem de diferença entre as médias das intensidades dos píxeis do parafuso e os da vizinhança, visto existir a possibilidade de o parafuso ter média de intensidades ligeiramente inferior à da vizinhança, devido a questões de iluminação. Quando a diferença entre a média dos níveis de cinza do anel e o desvio é menor ou igual à média de intensidades do *kernel* central, o parafuso é considerado como presente para a respetiva imagem. Em caso contrário, o parafuso está ausente, sendo considerado este tipo de casos como falha na colocação do parafuso na inspeção.



(a) (b)

(c) (d)

Figura 4.7: Passos da inspeção de presença de parafusos: (a) ROI onde o fluxo é focado; (b) resultado da binarização; (c) seleção da região pretendida da binarização (b); (d) Regiões utilizadas na comparação de intensidade média entre o local parafuso e a vizinhança.

Nesta solução, é necessária a prévia definição da ROI para a redução do espaço de trabalho e alguns parâmetros podem ser ajustados conforme o julgamento do utilizador. Tais correspondem aos valores *threshold* da binarização por *threshold* variável, como a seleção das regiões após a binarização é feita,

os tamanhos dos elementos estruturantes das operações de abertura e dilatação, o raio do círculo para cálculo da média de intensidades do local do parafuso e ainda o valor de desvio que permite alterar o rigor com que a comparação entre as duas regiões é feita.

4.2.4 Inspeção do Posicionamento e Dimensões de Peças

Para o primeiro passo, é necessário identificar onde se localiza a peça e que se encontra bem posicionada para facilitar a inspeção. Inicialmente é utilizado um modelo de *matching* para procurar a peça na imagem, de modo a ser realizado o registo da imagem. Com a imagem alinhada obtida, verifica-se a existência do pino de referência, excluindo todas as imagens que não o apresentam dos restantes passos.

Abordando as imagens cujo pino de referência se encontra visível, é realizada uma binarização por *threshold*, seguido de uma operação de preenchimento e de *opening*. Estas operações permitem a obtenção das regiões pertencentes às partes do tubo e do conector da peça, representadas na Figura 4.8, a verde e vermelho respetivamente. A definição destas duas regiões é fundamental para o cálculo das distâncias requeridas.

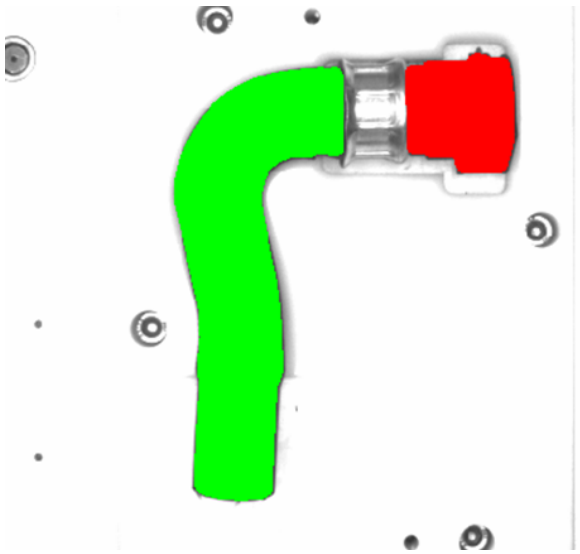
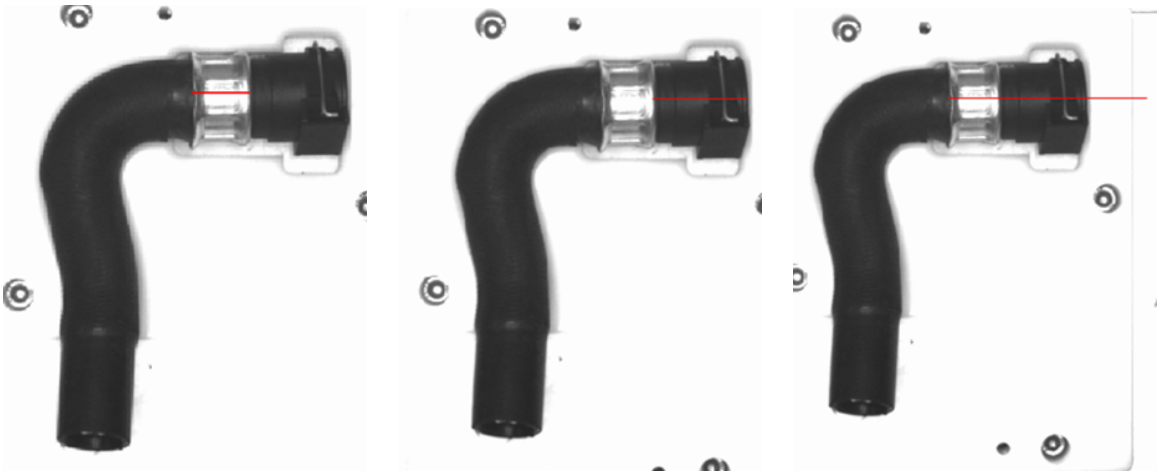


Figura 4.8: Identificação da região do tubo (a verde) e do conector (a vermelho).

Apenas com estas duas regiões formadas já é possível o cálculo da distância 2, o comprimento da anilha, através do cálculo da distância mínima entre as duas regiões. Partindo das coordenadas de ambos os extremos da linha por onde é calculada a distância 2 (Figura 4.9a), é feita a extensão dessa linha, obtendo-se o segmento de reta mostrado na Figura 4.9b e intersetada com a região do conector da Figura 4.8, obtendo-se um segmento de reta que permite o cálculo da distância 1 (Figura 4.9c), o comprimento do conector.

Continuando para o cálculo do comprimento do tubo, distância 3, pretende-se contar o número de píxeis pertencentes à região do tubo ao longo de um segmento de reta. Este segmento deve ser paralelo às bordas da parte inferior do tubo, marcado pelo retângulo na Figura 4.10a, visto estas seguirem uma forma retilínea grande o suficiente que permita a definição do segmento de reta. Para tal, a partir de um algoritmo de deteção de bordas, obtém-se o esqueleto da respetiva parte da peça. Com interação



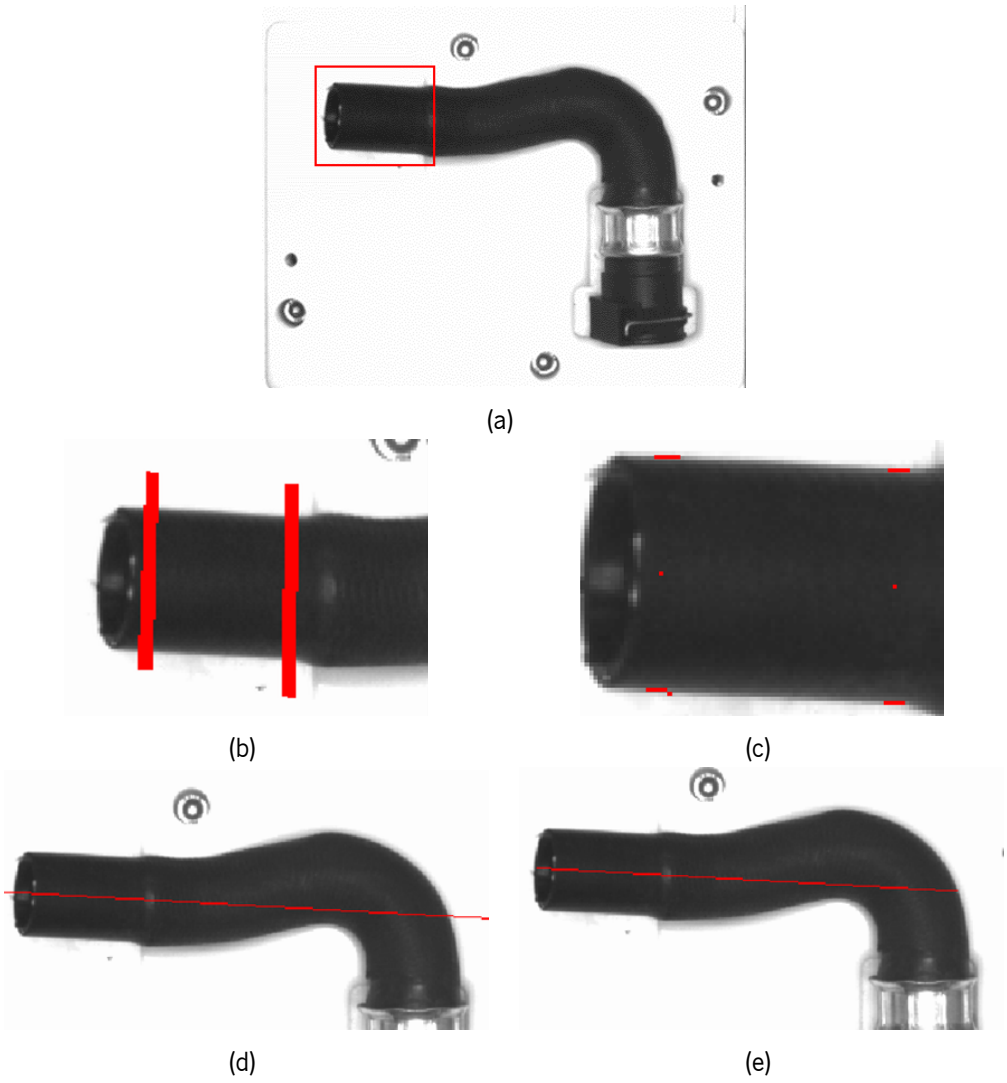
(a) (b) (c)

Figura 4.9: Passos para medição das distâncias 1 e 2: (a) Linha para medição de distância 2; (b) extensão do segmento de reta (a); (c) linha para medição da distância 1.

prévia do utilizador são definidas duas regiões (Figura 4.10b), que são intersetadas com a região do tubo inicialmente definida, para a extração dos ponto médios em cada uma delas (Figura 4.10c), resultando na extração das coordenadas necessárias para o desenho de um segmento de reta que permita calcular a distância 3 (Figura 4.10d). Por fim, é realizada a interseção de regiões (Figura 4.10e), agora entre a região do tubo e o novo segmento de reta. Obtém-se assim a contagem no número de píxeis associados ao comprimento do tubo.

Para a distância 4, o cálculo da distância entre a borda inferior do conector e o pino, foram calculadas as regiões associadas à borda e ao pino. Relativamente ao elemento da borda, é previamente necessária a definição de uma ROI focada na borda inferior do conector, exemplificada na Figura 4.11a, de modo a apenas efetuar as operações necessárias nessa região. A partir da ROI da borda, foi utilizado um operador de deteção de bordas seguido de um *opening* com elemento estruturante apropriado para eliminar as bordas não horizontais. Para a região do pino, aplicou-se uma erosão na região do conector, de modo a eliminar algumas estruturas irrelevantes para a resolução do problema. A esta operação segue-se uma binarização por *threshold*, para escolher as estruturas mais claras, de forma a incluir todo o pino, resultando em algo similar à Figura 4.11b. Como, após a binarização, existem elementos que não são pertinentes para o cálculo da distância, foi realizada uma seleção da região do pino a partir da compactidade dos elementos. A região do pino apresenta um valor de compactidade muito superior aos valores das restantes estruturas. Com as duas regiões determinadas (Figura 4.11c), foi calculada a distância mínima entre elas para ser obtida a distância 4.

Por fim, para o cálculo da distância 5, referente ao comprimento horizontal da peça entre o extremo esquerdo da parte do tubo e o extremo direito da parte do conector, basta ter em vista os números das colunas dos píxeis das regiões do tubo e do conector. Foi apenas realizada a subtração entre o número da primeira coluna da região do tubo e a última coluna da região do conector.



(a) (b) (c) (d) (e)

Figura 4.10: Passos para medição da distância 3: (a) região de trabalho; (b) regiões definidas pelo utilizador; (c) extração dos pontos médios das regiões de (a); (d) segmento de reta obtido com os pontos de (c); (e) interseção de (d) com a região do tubo.

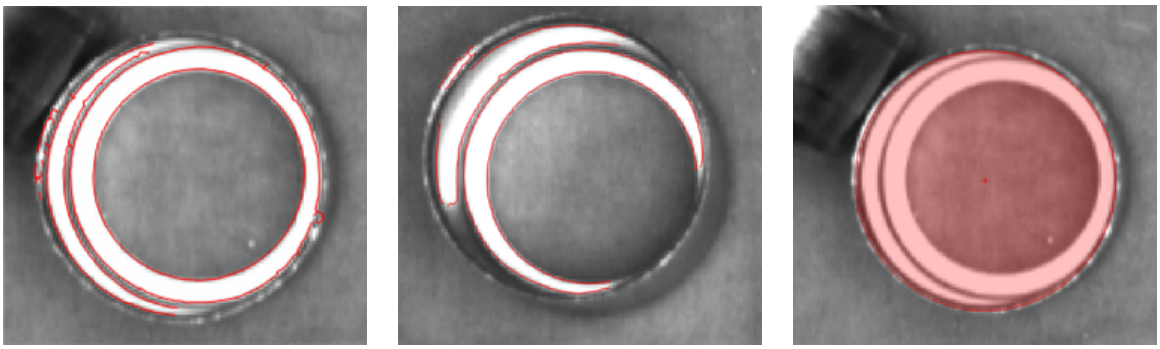
(a) (b) (c)

Figura 4.11: Passos para o cálculo da distância 4: (a) ROI para o cálculo da borda; (b) resultado da operação de binarização; (c) Estruturas para o cálculo da distância mínima entre o pino e a borda inferior do conector.

4.2.5 Object Picking

4.2.5.1 Casquilhos

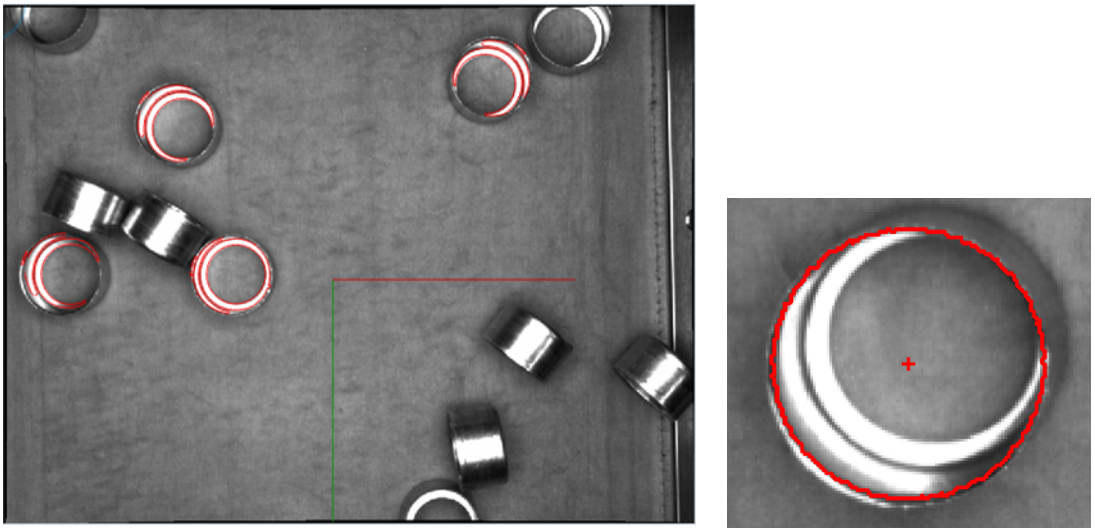
A procura das anilhas por *matching* consiste no primeiro passo, sendo previamente definidos 2 modelos de casquilhos (Figura 4.12a e 4.12b), assim como uma região circular representante da área de picking Figura 4.12c, para cada um dos casquilhos usados como *template* para o *matching*. Para o *picking* destas peças, é aplicado mais do que um modelo, porque a posição do casquilho relativamente ao centro da imagem influencia a forma como a iluminação é refletida neste. Utiliza-se mais do que um modelo para melhor representar as possíveis formas que este pode tomar na imagem e melhorar os resultados do algoritmo de procura.



(a) (b) (c)

Figura 4.12: (a), (b) Referências para o modelo de *matching*; (c) área de *picking*.

É inicialmente executada a procura de anilhas por meio de *template matching*, de modo a se encontrarem todas as peças presentes na imagem. Como anteriormente mencionado, a forma que a peça exibe difere bastante consoante como está pousada na superfície, com a base anelar para cima ou para baixo ou então com a superfície lateral em contacto com o tapete, permitindo a exclusão de todas as peças que não se dispõem corretamente orientadas. Tal se verifica no resultado do *matching*, na Figura 4.13a, apenas as anilhas com a orientação correta devem ser localizadas. Devido ao facto de o *picking*, para este caso em específico, ser realizado pela parte interior da anilha, não é necessária a verificação de obstruções para essa tarefa. Assim, partindo do resultado do *matching*, consegue-se retirar as coordenadas necessárias para o *picking* para cada um dos casquilhos corretamente orientados. Para se saber as coordenadas, não é suficiente utilizar as coordenadas encontradas pela procura do *matching*, porque estas correspondem ao centro do modelo *template* aplicado na peça atual. É realizada a criação de uma matriz de transformação entre as coordenadas de referência do *template* utilizado e as coordenadas do objeto encontrado pelo *matching*. A aplicação da matriz no círculo correspondente à área de *picking* previamente definido e referente ao mesmo modelo de *matching* que encontrou o casquilho, tem o intuito de centrar o círculo na peça. Por sua vez, as coordenadas do círculo são significativas do centro do casquilho, como exemplificado na Figura 4.13b, sendo estas utilizadas posteriormente para a tarefa de *picking*.



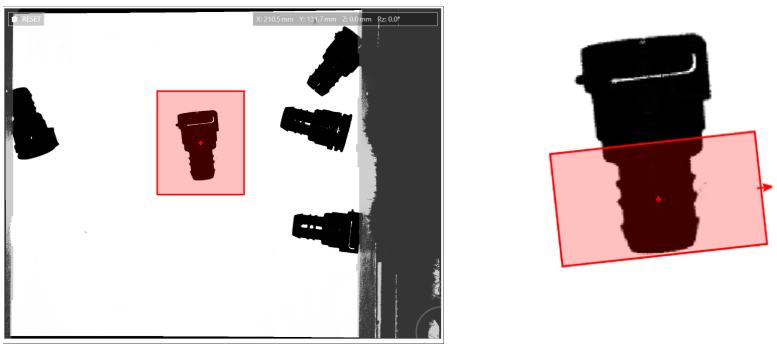
(a) (b)

Figura 4.13: Exemplos de (a) *matching* de casquilhos; (b) região circular centrada na anilha.

4.2.5.2 Conectores

Na imagem de referência desta aplicação são previamente definidas 2 regiões de referência e utilizados 2 modelos de *matching* baseado em formas. Uma das regiões, marcada na Figura 4.14a, serve para delimitar o espaço para apenas conter o conector que servirá como referência. A outra região, delineada na Figura 4.14b, é definida para marcar onde é realizado o *picking*. Dentro desta segunda região é necessária a verificação de que esse espaço não contém mais nenhum objeto, de modo a que não haja interferências no *picking*.

Relativamente aos modelos de forma, é primeiramente desenvolvido um modelo de procura da forma exterior do conector, para encontrar os conectores presentes na imagem. Para que seja possível encontrar conector candidatos para ser efetuado o *picking*, é utilizada a forma do conector, devidamente orientado, como base para treino do modelo de *matching*. O segundo modelo de forma serve apenas para encontrar o pino de referência nas regiões referentes aos conectores encontrados pelo primeiro modelo e baseia-se no pino do conector utilizado para o primeiro modelo de procura como *template*.



(a) (b)

Figura 4.14: ROIs necessárias para a solução de *picking* de conectores: (a) para a criação do modelo de matching para procura de conectores; (b) da zona de violação.

Procura-se primeiramente todos os conectores presentes na imagem por meio de *matching* (Figura 4.15a), conseguindo excluir algumas peças que estão mal orientadas, dependendo da percentagem mínima de semelhança entre o objeto encontrado e a referência, visto que, apesar da forma da peça variar pouco entre as situações de boa e má orientação, consegue ser suficiente para tal. É de seguida realizada uma segunda procura, apenas dentro de cada uma das regiões das peças que o primeiro modelo encontrou, com o objetivo de detetar a presença do pino de referência (Figura 4.15b). Caso a procura retorne algum resultado, considera-se que o pino está visível no respetivo conector, permitindo a exclusão dos restantes conectores que não têm pino visível.

(a) (b)

Figura 4.15: Resultados esperados das procuras por *matching* de: (a) conectores; (b) pino de referência do conector.

Segue-se por último a averiguação da exclusividade da zona de *picking* pelo *robot*, em que, como já foi referido, se averigua se existem possíveis obstruções para a tarefa em questão. Para tal, é realizada uma subtração entre a região correspondente à zona de verificação de violação e a forma do conector de referência, sofrida de uma pequena dilatação para incluir algumas diferenças entre a forma de referência e a do conector atual. Na região resultante, aplica-se uma binarização por *threshold* para identificar todas as estruturas alheias presentes dentro da zona de violação. No caso em que o *threshold* não retorna elementos ligados, ou seja, não existem outros objetos além do conector de análise, o conector avança para o processo de *picking*, sendo retornadas as coordenadas do centro da região de *picking* definida. Nas situações em que o conector está orientado corretamente mas a zona de *picking* encontra-se violada, como por exemplo na imagem da Figura 4.16, a peça a ser inspecionada deve ser identificada como não sendo viável para a realização do *picking*.

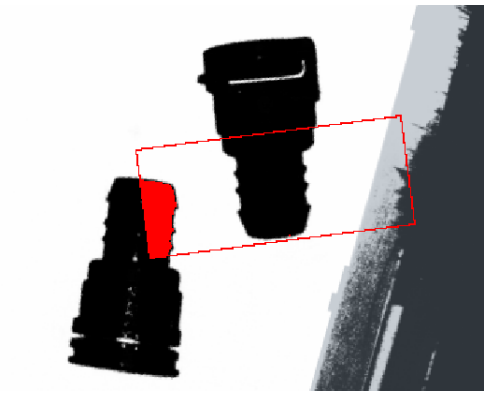


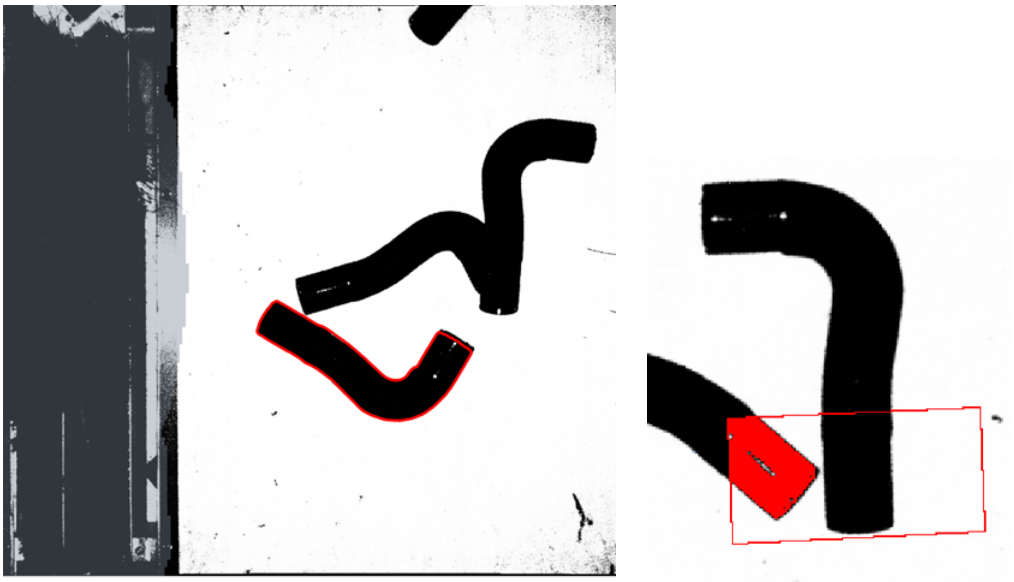
Figura 4.16: Violação da zona de *picking* para conectores.

4.2.5.3 Tubos

A solução desenvolvida para *picking* de tubos segue uma estrutura similar ao *picking* de conectores, diferenciando apenas na verificação da orientação, visto os tubos não possuírem um pino que serve como referência.

São procuradas as peças partindo da definição prévia de um modelo de forma, baseado numa peça de referência, cuja região foi calculada com o preenchimento do resultado da deteção de bordas numa ROI. São também definidas, anteriormente à execução desta solução, uma região equivalente à forma do tubo utilizado como *template* e uma região para verificação da curvatura do tubo.

Para cada tubo com correta orientação, ou seja, as correspondências obtidas pela procura por *template matching* (como por exemplo na imagem da Figura 4.17a), resta verificar se este possui obstruções dentro da região exclusiva para *picking*. O mesmo método utilizado na solução de *picking* de tubos, para este passo, é aplicável no *picking* de tubos. É primeiramente efetuada uma subtração entre a região da zona de violação e a forma dilatada do tubo de referência. À região resultante, aplica-se uma binarização por *threshold* para serem detetados os objetos que poderão se encontrar dentro da zona de *picking*. A Figura 4.17b apresenta um resultado esperado após as operações previamente explicadas, no caso de existirem objetos adicionais ao tubo de análise dentro da zona de *picking* (marcada pelo retângulo). Nesta situação, o tubo é eliminado da ação de *picking* pela garra. Porém, se a binarização não evidenciar nenhum objeto, diz-se que o tubo é selecionável para o processo de *picking*, sendo retornadas as coordenadas do centro da região de *picking* definida.



(a) (b)

Figura 4.17: Resultados esperados da solução de *picking* de tubos: (a) de procura de tubos por *matching*; (b) verificação de violação da zona de *picking*.

4.3 Sumário

O registo de imagens é uma etapa importante como passo inicial, anterior ao processamento de imagem, de forma a garantir que todas as imagens que são alimentadas aos fluxos de operações para IVA se encontram alinhadas com uma imagem de referência. Assim, as estruturas presentes na imagem encontrar-se-ão sempre na mesma localização espacial, promovendo o desenvolvimento de soluções para a tarefa de inspeção visual, conseguindo-se identificar os objetos de análise mais facilmente.

Recorrendo exclusivamente aos operadores disponíveis no *HALCON*, foram implementados fluxos de operações para resolver os problemas de inspeção de dispensação de cola, inspeção do posicionamento de cabos fita, deteção da ausência ou posicionamento defeituoso de parafusos, inspeção de posicionamento e montagem defeituosa a partir do cálculo de distâncias e a deteção de peças de casquilhos, conectores e tubos para a tarefa de *object picking*, de acordo com os objetivos requeridos. Todas as soluções, à exceção da deteção de parafusos, passam pelo registo de imagens, tanto como forma de alinhamento de imagens, como de localização de peças (como no caso de deteção de peças para *picking*). Em todas as soluções, é solicitada a definição prévia de parâmetros por parte do utilizador, podendo estes serem modelos para o registo de imagens, ROIs para limitação do espaço de trabalho e identificação de estruturas da relevantes ou simples valores para a execução de certas operações, assim como valores que servem para verificação das condições de sucesso de inspeção.

Capítulo 5

Resultados e Discussão

Neste capítulo, são explicados os testes realizados a cada uma das soluções da *toolbox* para os problemas abordados. São comentados os resultados obtidos e é feita uma avaliação das soluções desenvolvidas, se estas conseguem ou não resolver com fiabilidade cada um dos problemas explorados, consoante os requisitos definidos.

5.1 Inspeção de Dispensação de Cola

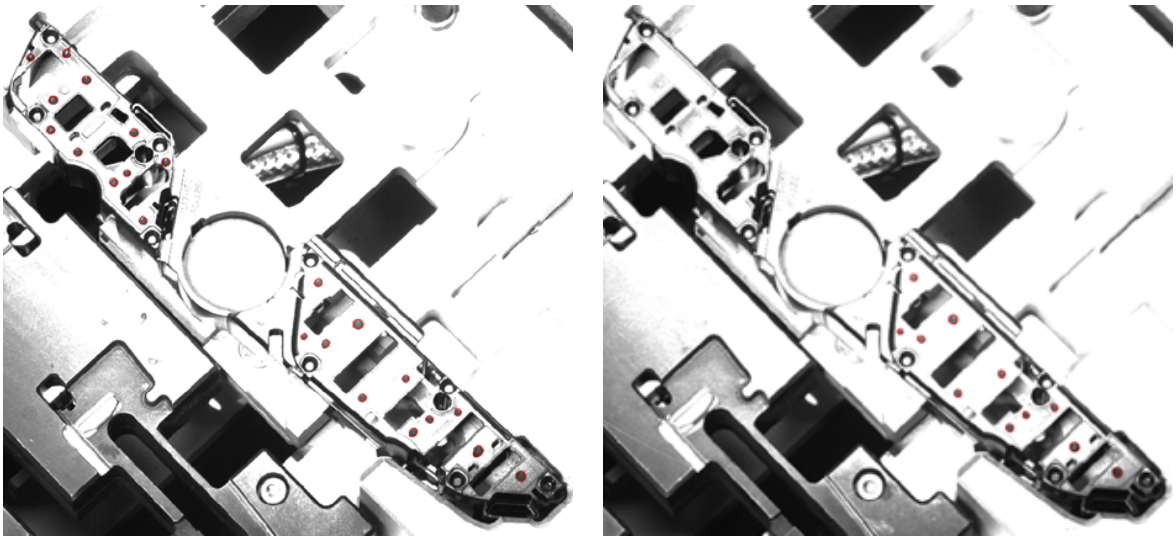
A solução desenvolvida para a inspeção da dispensação de cola foi testada utilizando ROIs fixas para os pontos de cola, assim como com a criação da ROI adaptada à localização de cada ponto e ficar centrada nos pontos. Para ambos os casos, as regiões foram criadas com um comprimento de 23 píxeis.

Os testes devem ter em conta uma base de dados com um total de 60 imagens, em níveis de cinzento e com resolução de 1280 x 1024 píxeis, divididas em 30 imagens de sucesso e 30 imagens representativas de falha na inspeção. É de notar que, como as quantidades mínimas e máximas de cola podem ser diferentes para cada ponto, e não são fornecidas juntamente com a base de dados, não é evidente em alguns casos a razão para a falha na inspeção em variadas imagens. O mesmo acontece na situação oposta, onde a imagem é considerada como sucesso, existindo no entanto pontos que aparentam não estar em cumprimento com a cláusula de sucesso. O resultado obtido pela inspeção com a solução é considerado como positivo quando a imagem deve passar na inspeção e o resultado coincidiu com a *label* associada à imagem. Do mesmo modo, considera-se como negativo quando a *label* diz que a imagem deve falhar e a solução a classificou como tal.

A solução foi testada de forma a verificar as diferenças nos resultados obtidos ao utilizar ROIs fixas e na situação da criação de nova *ROI*. Para tal, foram também testados diferentes tamanhos de ROIs para melhor clarificar as diferenças existentes.

Relativamente à escolha da binarização, foi apenas aplicada binarização com *threshold* variável com *kernel* de *17 x 17*, visto não ser possível escolher acertadamente valores mínimos e máximos de *threshold* de forma a se obter resultados satisfatórios. Uma situação semelhante acontece na escolha de valores limite de quantidade de cola, onde foram apenas usados valores universais, um mínimo de 40 píxeis e máximo de 220 píxeis, tendo em conta a média do número de píxeis de todos os pontos de cola na base

de dados rondar os 110 píxeis. Exemplos do funcionamento da solução encontram-se expostos na Figura 5.1, onde a Figura 5.1a representa um caso de sucesso na inspeção, pelo que todos os pontos de cola se encontram identifcados, sinalizando que se encontram dentro os limites, e a Figura 5.1b representa um caso de falha na inspeção, sendo que não foi identificado nenhum ponto de cola na parte superior da peça, visto não ter sido dispensada qualquer cola. Ambas as imagens apresentadas se encontram em concordância com os respetivos *ground truth*.

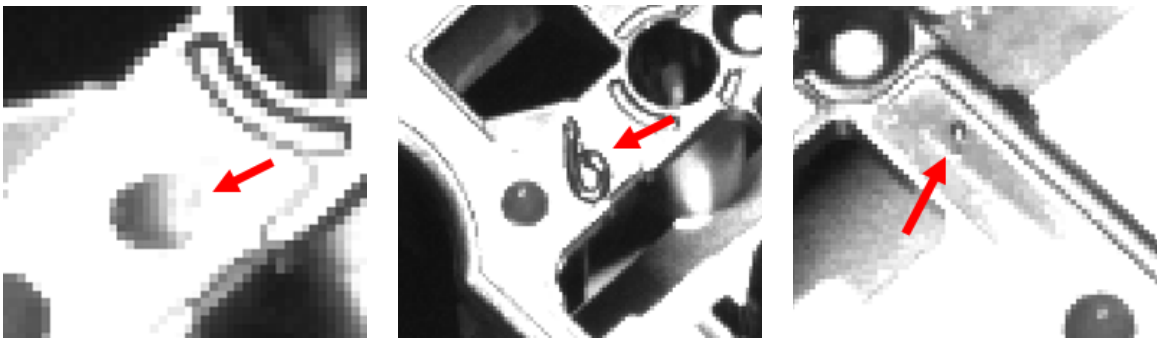


(a) (b)

Figura 5.1: Exemplos do funcionamento da solução para inspeção de dispensação de cola: (a) caso de sucesso; (b) caso de falha.

Quanto ao uso da criação de uma ROI nova centrada em cada ponto de cola em análise, não houve nenhum falso negativo para ambas condições de teste, ou seja, a solução considerou como inspeção bem sucedida todas as imagens que deveriam passar no teste. Houve no entanto vários falsos positivos, 9 quando não foi definida nova ROI e 8 quando a solução foi testada com a criação de nova ROI. A Figura 5.2 explicita alguns pontos de cola que causaram as respetivas imagens a serem falsos positivos, consequentes de problemas de iluminação na captura da imagem. Nas Figuras 5.2a e 5.2b, a contagem dos píxeis foi suficiente para serem considerados como dispensação correta e na 5.2c, o *background* foi identificado como fazendo parte do ponto de cola, aumentando o número de píxeis para dentro dos limites de contagem definidos. Outros falsos positivos foram devidos aos limites universais considerados para a contagem de píxeis.

Apesar de as duas situações de teste, utilização de ROIs fixas e criação de ROI centrada no ponto de cola, aparentarem obter resultados similares, a diferença entre elas passa a ser explicada com as Figuras 5.3 e 5.4. Na imagem de teste da Figura 5.3, esta é corretamente sinalizada como falha na tarefa de inspeção, no entanto nem todos os pontos que deveriam ser considerados fora dos limites de quantidade de cola o foram. Verifica-se que, por exemplo, o ponto não foi assinalado na Figura 5.3b falha na inspeção por não passar estar em concordância com o limite máximo de contagem de píxeis definido, porque possui cola a mais. Já para o ponto marcado na Figura 5.3c, apesar de ser suposto que o mesmo aconteça, tal não foi verificado. Considerando a ROI marcada na 5.3c, parte da cola não foi utilizada nas

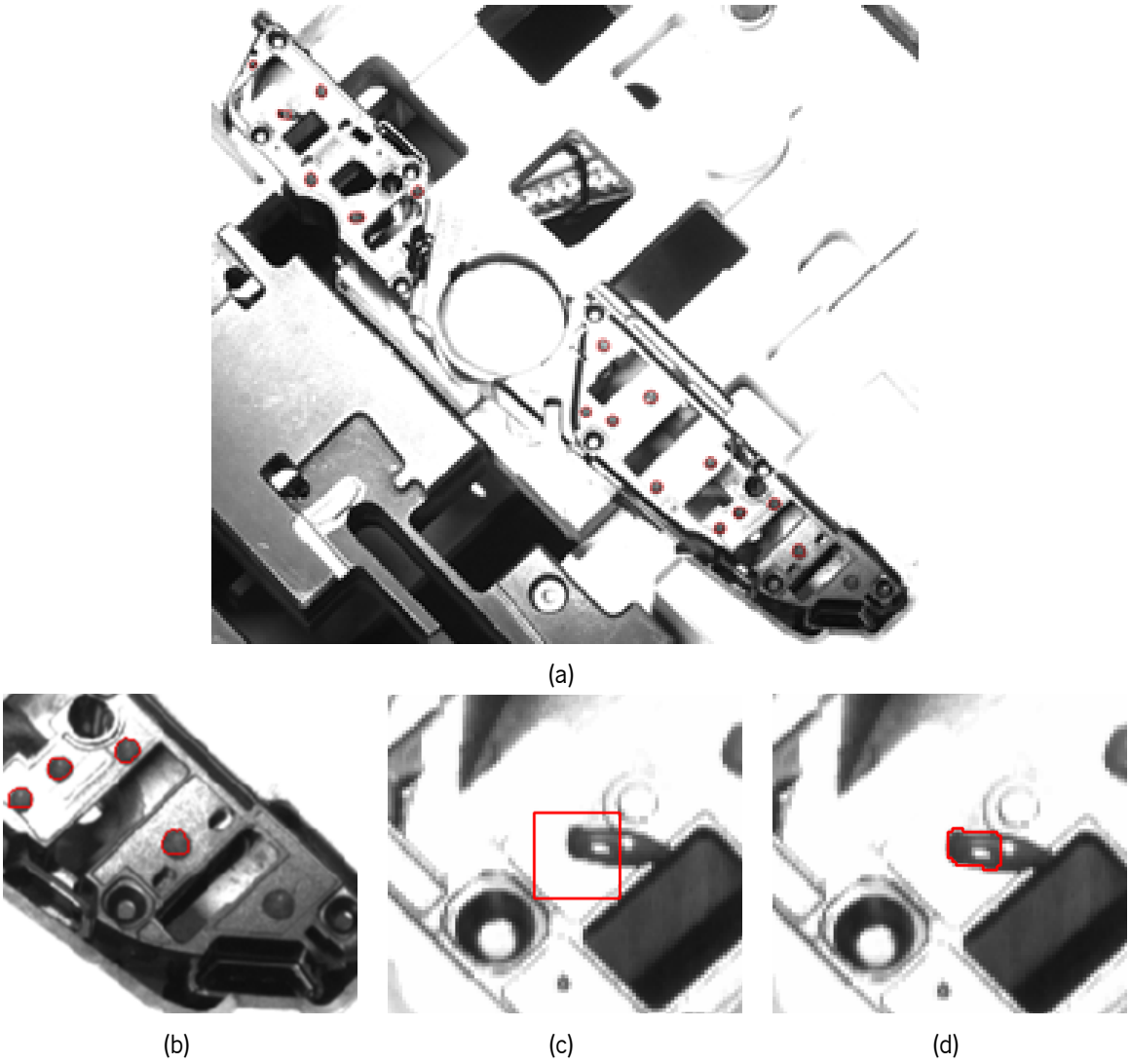


(a) (b) (c)

Figura 5.2: Exemplos de pontos falsos positivos na inspeção de dispensação de cola devido a problemas de iluminação: (a) e (b) os pontos estão parcialmente mais claros; (c) solução confunde o *background* como sendo cola.

operações que levam à definição da região para a contagem de píxeis, porque a ROI, ao reduzir o espaço de trabalho, cortou parte da região da cola. Assim, a região considerada para a contagem foi apenas a representada na Figura 5.3d.

Isto poderia ser resolvido com o aumento do tamanho da ROI, pois seriam detetados mais píxeis pertencentes ao ponto de cola. Porém, o aumento do tamanho das ROIs tem como consequência situações como a representada na Figura 5.4a, onde se aumentou o tamanho da ROI para 31 píxeis de largura. Verifica-se que a região associada ao ponto de cola, na Figura 5.4b, inclui uma porção do *background*, pois este tem intensidades semelhantes ao ponto e não é possível a separação de ambos os objetos. Nestes casos, a criação de uma ROI centrada mostra-se benéfica, porque ao centrar o espaço de trabalho nas coordenadas centrais do ponto encontrado, é possível que a nova ROI seja mais pequena, Figura 5.4c. Na Figura 5.4d, é demonstrado o resultado da utilização da criação de uma ROI mais pequena após a deteção do ponto com a ROI inicial. Tendo em consideração as situações referidas, gera-se um problema. Por um lado, a contagem dos pontos de cola pode ser mais acertada com o uso de uma ROI com tamanho superior pois a região do ponto não fica parcialmente cortada pela ROI, porém existe a possibilidade de acontecerem casos como a Figura 5.4b. Por outro lado, estes casos como a Figura 5.4b podem ser mitigados com a definição de uma ROI nova mais pequena e centrada na instância detetada, criando no entanto de novo o problema da Figura 5.3c, em que, apesar de o ponto na totalidade possuir contagem superior ao limite definido, a inspeção é considerada como sucesso para a instância em específico porque a região ficou cortada pela ROI e a contagem de píxeis é portanto menor.



(a)

(b) (c) (d)

Figura 5.3: Exemplo de problema na solução da inspeção de dispensação de cola: (a) imagem de teste; (b) exemplo de ponto de cola acima da quantidade máxima de cola; (c) ROI de ponto mal detetado; (d) região detetada do ponto da ROI em (c).

5.2 Inspeção de Cabos Fita

O funcionamento da solução desenvolvida para o problema de verificação da colocação de cabos fita foi testado com a alteração de um único parâmetro, a percentagem máxima na qual a largura da segunda faixa que poderia estar vísivel, enquanto considerando-se o *foil* como corretamente colocado, visto este ser o único valor que o utilizador deve ser capaz de escolher.

Para este problema, é utilizada uma base de dados com um total de 22 imagens em RGB, com uma resolução de 831 x 538 píxeis. Para o primeiro tipo de imagens, que será denominado de tipo A, composto por um total de 11 imagens, exemplificado pela Figura 3.3a, é pretendido que sejam inspecionados 2 cabos fita em cada imagem analisada. Este conjunto de imagens não tem *labels* associadas para nenhum dos tipos de imagens. No entanto foi requerido que cabos como o da Figura 3.4c devem ser considerados como falha, devido à significativa exposição da segunda faixa, indicativa de falha de inspeção. Deste modo, do conjunto de imagens do tipo A, apenas 3 imagens correspondem a falha,

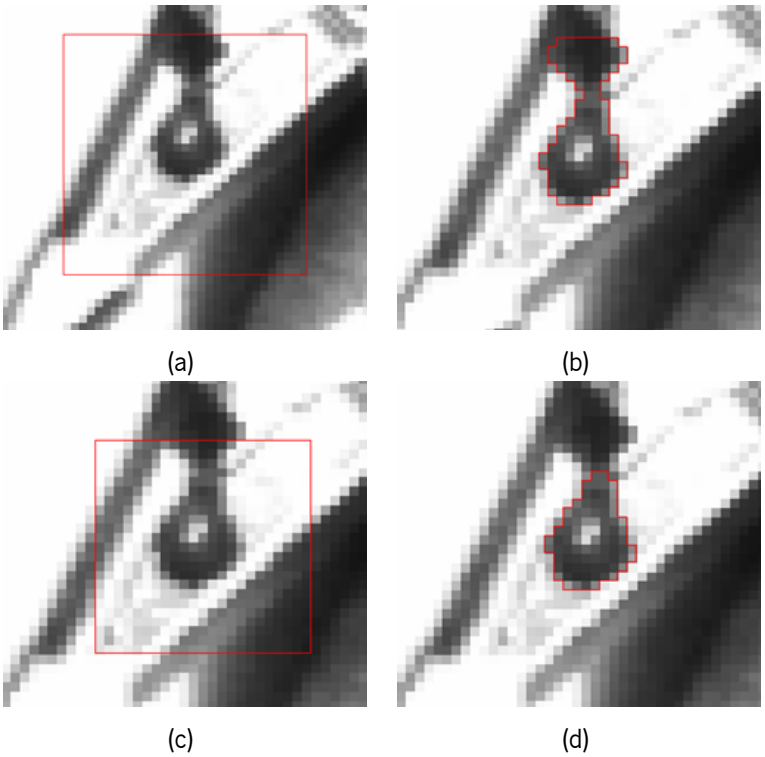


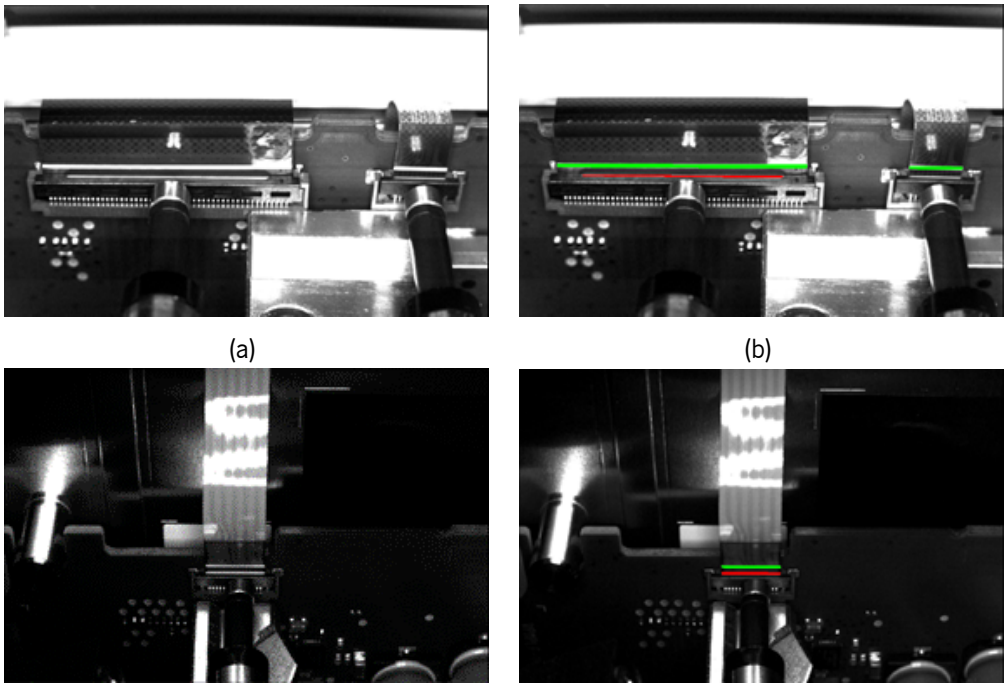
Figura 5.4: Diferenças entre uso de ROIs fixas e criação de nova ROI para deteção de pontos de cola: (a) utilização de ROI larga; (b) resultado da solução trabalhando na ROI (a); (c) criação de nova ROI mais pequena; (d) resultado da solução trabalhando na ROI (c).

onde o cabo maior aparenta estar mal colocado. Em todas as imagens o cabo mais estreito deve passar sempre na inspeção. Relativamente ao segundo tipo de imagens, tipo B, representado pela Figura 3.3b, este é também composto por um conjunto de 11 imagens. No entanto, existe apenas uma inspeção da colocação do cabo a ser feita, em cada imagem. Das 11 imagens do Tipo B, 4 devem correspondem a falha no momento de inspeção. Os resultados da solução são classificados como positivos quando estes concordam com as respetivas *labels* no sucesso da colocação do cabo. Assim, um falso positivo acontece quando a solução detetou sucesso e na verdade houve falha de colocação. O contrário acontece para os falsos negativos.

Nas imagens do tipo A foram estudadas 3 valores de percentagem diferentes, 30%, 40% e 50%. Estes valores foram os mesmos para ambas as faixas. Com a verificação de que a segunda faixa não é visível em mais que 30% da sua largura total, houve 1 imagem considerada como tendo o cabo mal colocado no total de 11 imagens, quando na realidade tal não acontece, sendo um falso negativo. O contrário aconteceu com a utilização de valor de 50%, em que houve 1 falso positivo. No entanto, para a testagem com 40%, todas as imagens foram corretamente classificadas. Na Figura 5.5a é demonstrado um exemplo da deteção da má colocação do cabo maior e na Figura 5.5b verifica-se o resultado obtido pela solução desenvolvida. Consta-se que a solução identificou corretamente a segunda faixa e indicou que esta não cumpre o limite definido.

Relativamente às imagens do tipo B, o teste foi feito de maneira similar, mas utilizando percentagens de 40%, 50% e 60%. Com o valor mais baixo testado, surgiram 2 falsos negativos. Com o valor de 60%, os testes erraram em 2 imagens, em que a inspeção deveria ter resultado em falha e não aconteceu, ou

seja, ocorreram 2 falsos positivos. O teste com o valor de 50% conseguiu detetar corretamente todas as imagens de falha e de colocação bem sucedida. A Figura 5.5d corresponde a um exemplo de execução da solução numa imagem de tipo B (Figura 5.5c) que deve ser considerada como falha. Verificou-se que a segunda faixa foi detetada como pretendido.



(a) (b)

(c) (d)

Figura 5.5: Resultados da solução para inspeção de cabos fita: (a) Imagem de teste do tipo A; (b) resultado da execução da solução para inspeção de foils com a imagem (a); (c) imagem de teste do tipo B; (d) resultado da execução da solução para inspeção de foils com a imagem (c).

De acordo com os resultados obtidos, determinou-se que as percentagens ideais rondam os 40% para as imagens do tipo A e os 50% para as imagens do tipo B. Contudo, como na base de dados para as imagens do tipo A não existem casos de falha para a inspeção do cabo mais pequeno, é impossível verificar qual a percentagem mais acertada a utilizar.

5.3 Deteção da Presença de Parafusos

A base de dados para a resolução do problema contém 108 imagens RGB, com uma resolução de 831 x 538 píxeis. Destas, 42 correspondem a caso positivo da presença de parafuso, 19 sem parafuso e ainda 47 com obstrução pelo autocolante. Para efeitos de teste, foram considerados os resultados obtidos como verdadeiros positivos quando as imagens contêm parafuso e a solução também considerou como o parafuso estando presente. Desta forma, falsos negativos definem-se como sendo imagens que contêm parafuso mas a solução desenvolvida considerou que o parafuso não foi colocado como desejado. Os falsos positivos são todos os casos em que as imagens não contêm parafuso mas a solução considerou que o parafuso estava presente.

A solução foi testada para vários tamanhos de *kernel* na operação de binarização (21 x 21; 25 x 25; 35 x 35), tendo em conta que diferentes tamanhos para a deteção do objeto de localização da área de trabalho é também utilizado nas operações posteriores a esse passo. Como a média das intensidades da vizinhança do parafuso é um fator determinante do resultado da inspeção, testaram-se raios de 5, 7 e 9 píxeis na operação dilatação, pois o raio determina a quantidade de píxeis que são contabilizados para o cálculo da média do anel. Foi também estudada a influência da alteração do valor de desvio utilizado na comparação entre as médias das intensidades do parafuso e da vizinhança, tendo sido testados valores de 0, 20 e 40. É de notar que as imagens com autocolante foram corretamente sinalizadas, em todos os testes realizados.

Os resultados obtidos relativamente ao número de falsos negativos encontram-se na Tabela 5.1, e em relação ao número de falsos positivos, estes são apresentados na Tabela 5.2. Relativamente ao tamanho do *kernel* usado na binarização, o melhor *kernel* para a operação de binarização foi de tamanho 25 x 25. De acordo com a Tabela 5.1, o *kernel* de 21 x 21 foi o que errou mais, tendo sempre mais falsos negativos que os restantes *kernels*. Os *kernels* de 25 x 25 e 35 x 35 obtiveram números de falsos negativos similares, para todos os valores de desvio, sendo que o último supera o tamanho 25 x 25 para um valor de desvio de 40, com apenas 1 falsos negativos. No entanto, o número de falsos positivos foi sempre superior com o maior tamanho de *kernel* testado. Pela análise das tabelas pode-se concluir que um desvio de 20 intensidades na média dos níveis de cinza do anel demonstrou ser o melhor valor de forma a maximizar o número de verdadeiros negativos e positivos, relativamente às *labels* de *ground truth*.

Tabela 5.1: Número de falsos negativos para diferentes tamanhos de *kernels* de binarização, raio de dilatação e valores de desvio.

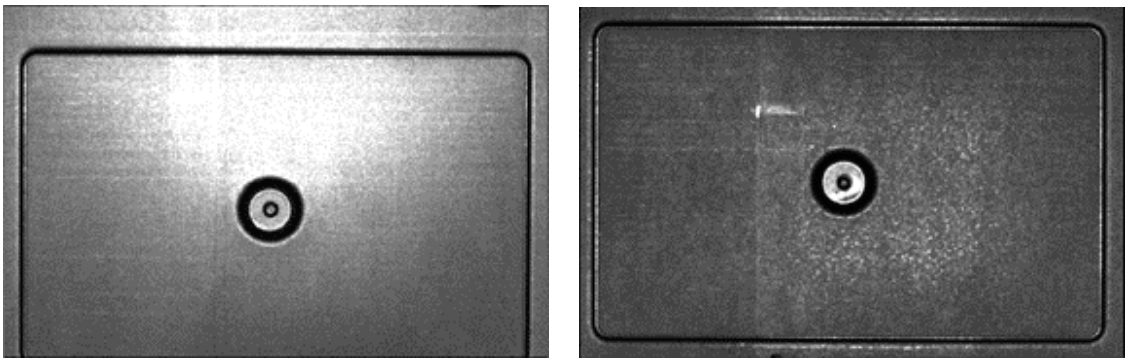
		Valor de desvio na comparação entre regiões								
		0			**20**			**40**		
		Raio de dilatação			**Raio de dilatação**			**Raio de dilatação**		
		5	**7**	**9**	**5**	**7**	**9**	**5**	**7**	**9**
Kernel de binarização	**21 x 21**	9	9	10	8	8	8	8	7	8
	25 x 25	5	4	6	2	2	3	2	2	3
	35 x 35	5	5	6	2	2	3	2	1	2

Deve ser explicitado que as mesmas 2 imagens de teste são comuns a todos os falsos negativos

Tabela 5.2: Número de falsos positivos para diferentes tamanhos de *kernels* de binarização, raio de dilatação e valores de desvio.

		Valor de desvio na comparação entre regiões								
		0			**20**			**40**		
		Raio de dilatação			**Raio de dilatação**			**Raio de dilatação**		
		5	**7**	**9**	**5**	**7**	**9**	**5**	**7**	**9**
Kernel de binarização	**21 x 21**	0	0	0	2	0	0	2	2	2
	25 x 25	0	0	0	2	0	0	2	2	2
	35 x 35	1	1	1	1	1	1	3	3	3

obtidos. Estas duas imagens foram mal classificadas graças a problemas de iluminação. Para a primeira imagem, (Figura 5.6a), a má iluminação na imagem tem como consequência o aumento dos níveis de cinza numa grande porção da imagem, o que implica que a vizinhança do parafuso tem média de intensidades acima do parafuso. Na imagem da Figura 5.6b, acontece uma situação similar, mas o problema foca-se apenas na vizinhança. Para ser possível retificar estes dois casos, seria necessário alterar o valor de desvio para 125. Esta modificação compromete o rigor da solução e não é particularmente relevante, porque o erro tem origem na captura da imagem.



(a) (b)

Figura 5.6: Imagens para inspeção de parafuso com problemas de iluminação.

Tendo em mente tudo o que foi referido, a melhor combinação de parâmetros revelou ser binarização com *threshold* variável com *kernel* de 25 x 25, uma operação de dilatação de raio 7 e a comparação entre os valores médios de intensidades entre o anel representativo da vizinhança e o local do parafuso desviada de 20 níveis de cinzento.

5.4 Inspeção do Posicionamento e Dimensões de Peças

Como a base de dados para o desenvolvimento desta solução apenas contém 1 imagem, não foram realizados quaisquer tipos de testes, porque a solução foi criada a partir da imagem disponível. Deste modo, não é possível analisar o fluxo de operações desenvolvido.

5.5 Object Picking

5.5.1 Casquilhos

A solução para este problema deve ser testada com base em 3 imagens fornecidas. Nestas imagens, os casquilhos podem estar colocados no tapete em 3 posições diferentes, em cada uma das suas bases ou então apoiados na parede lateral. Destes, apenas se pretende que sejam escolhidos os casquilhos apoiados na base com superfície anelar para a tarefa de *picking*. De todas peças apoiadas nas suas bases, 13 casquilhos estão apoiados da forma pretendida e 5 da forma oposta. Todas as deteções de casquilhos selecionáveis para *picking* pela solução e que de facto o são, são consideradas como resultados positivos.

A solução foi testada primeiro com a definição de apenas um modelo de *matching* e depois adicionando um segundo modelo. Foi também verificada a influência do *score* mínimo da procura por *matching* na deteção dos casquilhos corretamente orientados.

Para a localização de casquilhos para *picking*, a operação de *template matching* foi testada com *score* mínimo de semelhança entre a instância encontrada e a referência de 70%. Isto quer dizer que, para existir correspondência, o casquilho tem de partilhar pelo menos 70% da sua forma com um dos modelos de *matching*. A utilização de apenas um modelo, tal como esperado não foi suficiente para encontrar todos os casquilhos presentes nas imagens, como demonstrado na Figura 5.7b, resultante da procura por correspondências na imagem da Figura 5.7a. No entanto, com a criação de um segundo modelo, isto já foi possível, permitindo retirar as coordenadas do centro de cada casquilho corretamente orientados, representado pela Figura 5.7c.

Com a utilização dos dois modelos e realizando a procura com um *score* de 70%, todos as anilhas pretendidas para *picking* foram corretamente localizadas e retiradas as respetivas coordenadas, a partir da região de *picking* circular, centrada em cada anilha e todos os objetos mal orientados foram excluídos do *picking*. Já com *score* de 50%, as anilhas indevidamente orientadas foram também detetadas e seriam sinalizadas para picking, validado pela Figura 5.7d, onde foram marcadas com a região da garra do *robot*. Desta forma, verificou-se que diminuir o score de modo a que o modelo fosse mais adaptável às formas que o casquilho pode tomar não seria aconselhado. Não se revelou ser possível aumentar o *score* mínimo acima de 70%, porque comprometeria a localização de todos os casquilhos, possivelmente devido ao facto da forma detetada do casquilho variar com a sua localização espacial, ou seja, algumas peças não eram detetadas. Uma solução seria utilizar mais um casquilho como *template* para criar outro modelo que pudesse complementar ainda melhor os modelos usados. Desta maneira, as formas que o casquilho pode adoptar na imagem estariam mais bem representadas, permitindo assim a que o *score* utilizado fosse mais rigoroso.

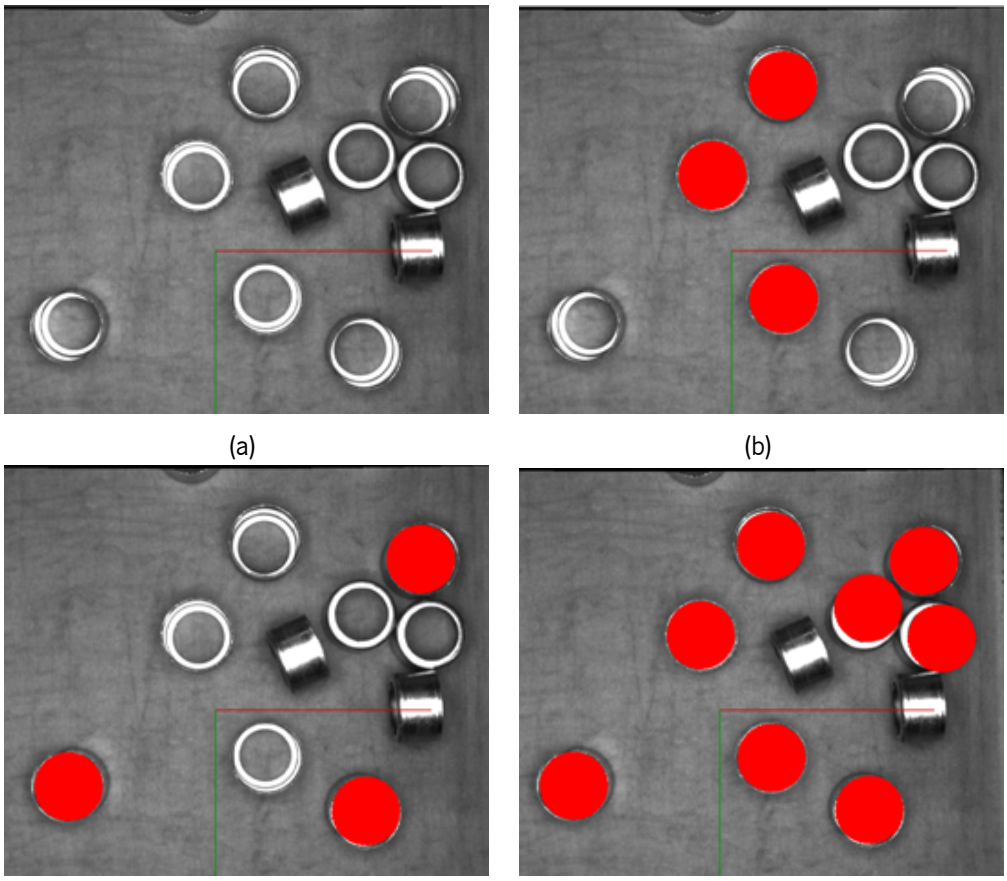


Figura 5.7: Resultados da solução para *picking* de casquilhos: (a) imagem de teste; (b) resultado do *matching* com o primeiro modelo com *score* de 70%; (c) resultado do *matching* com o segundo modelo com *score* de 70%; (d) resultado do *matching* com utilizando 2 modelos com *score* de 50%.

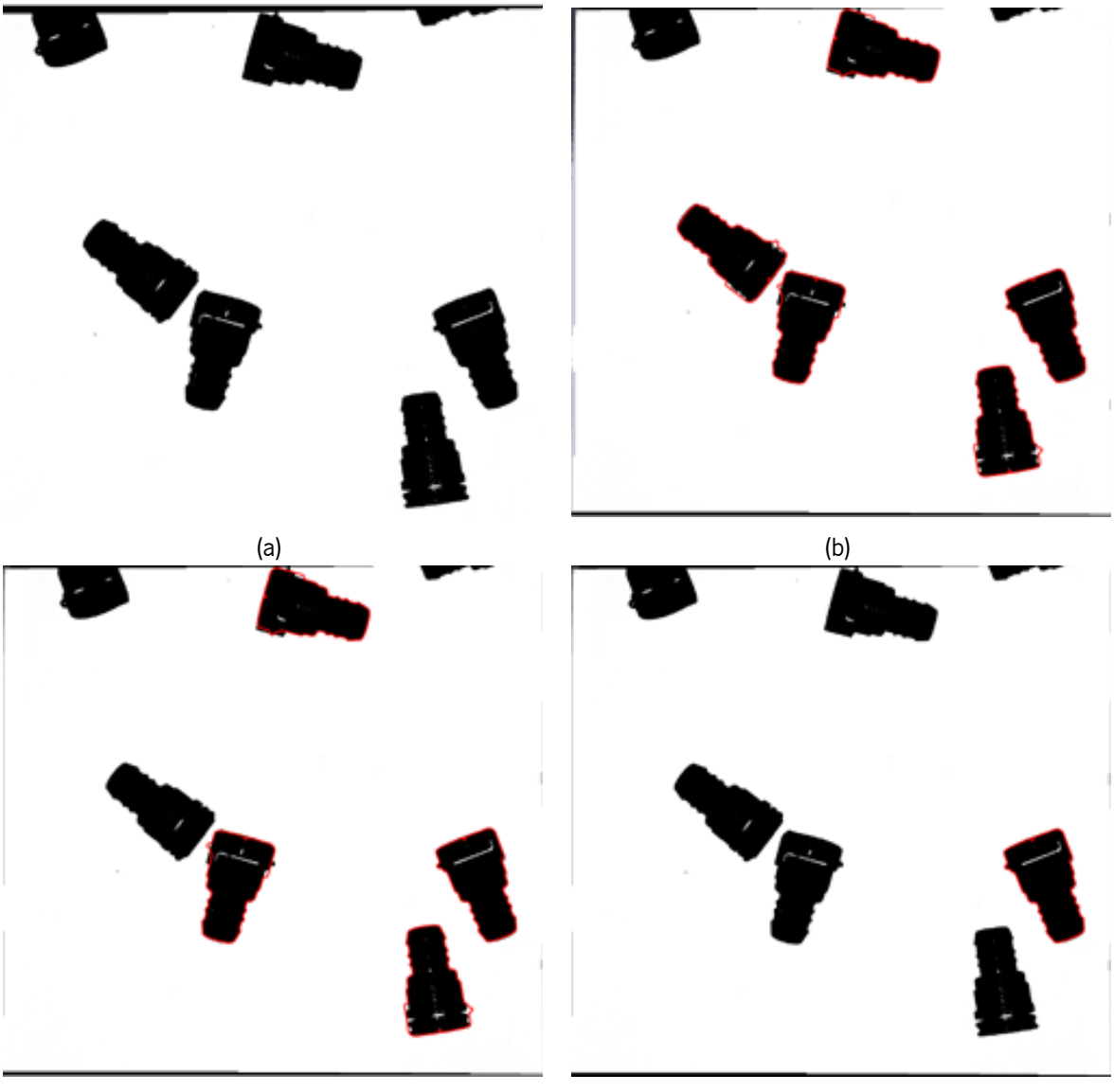
5.5.2 Conectores

A solução para inspeção de *picking* de conectores deve ter em conta uma base de dados com apenas 3 imagens, equivalendo no total a 7 conectores com orientação correta e 7 conectores com orientação errada. Dos que possuem orientação correta, 6 ou 7 devem ser marcados como peças para *picking*, dependendo das dimensões da área de violação utilizada, representativa do espaço que a garra do *robot* ocupa. Obtém-se um resultado positivo quando a solução detetou um conector corretamente posicionado, a partir da visibilidade do pino de referência, e com a sua zona de *picking* livre de objetos alheios, tal como a sua respetiva *label* o indica. Num caso negativo, um conector que deveria não deveria ser considerado para *picking*, foi descartado pela solução.

No sentido de validar a solução do problema de deteção de conectores, foram utilizados vários valores de *score* mínimo para *matching*, para se confirmar como o *score* afeta a deteção dos conectores bem orientados, assim como a deteção dos conectores que devem ser descartados. Adicionalmente, foi aumentada a zona de *picking* para se averiguar que a parte do fluxo de operações correspondente à verificação de violação dessa zona funcionou como esperado.

inicialmente testados três *scores* mínimos de semelhança para a procura de peças por *template matching*, com valores de 70%, 80% e 90%. Os resultados estão representados na Figura 5.10. Por análise

da Figura 5.8a, devem ser detetados no máximo 5 conectores, independentemente da sua orientação. Tal acontece quando é aplicado *score* de 70%, como visto pela Figura 5.8b. Foi no entanto possível excluir conectores com orientação errada ao aumentar o *score* mínimo. Com *score* de 80% (Figura 5.8c), um dos conectores mal orientados não foi detetado e com *score* de 90% (Figura 5.8d), nenhum dos conectores não desejados foi detetado, sendo que houve uma peça que deveria ter sido detetada e não o foi. Ainda, a procura com *score* de 80% conseguiu detetar todas as 7 instâncias que era necessário detetar dentro do conjunto de imagens, enquanto que com *score* de 90% apenas foram identificados 3 conectores. Tendo estes resultados em consideração, pode-se concluir que a procura de conectores de *matching* é mais eficiente quando é aplicado *score* mínimo de semelhança entre a referência e a instância encontrada de 80%.



(a) (b)

(c) (d)

Figura 5.8: Resultados da solução de *picking* de conectores no passo de *matching*: (a) imagem de teste de *picking* de conectores; (b) com *score* de 70%; (c) com *score* de 80%; (d) com *score* de 90%.

Quanto à verificação da violação da zona de *picking* por objetos que não o conector em análise, a solução comportou-se como esperado, descartando os conectores em que a garra não poderia pegar

com segurança. Para efeitos de testagem, foi definida uma área de *picking* excessivamente grande para garantir que a peça que poderia ser descartada do processo de *picking*, consoante o tamanho da área ocupada pela garra, seria corretamente identificada como não sendo elegível. Como exemplo foi estudada a imagem da Figura 5.9a. Por análise da Figura 5.9b, a estrutura adicional ao conector de análise foi marcada por violar a região definida, resultando na deteção de apenas 1 conector para ser agarrado na imagem em questão, como visto pela Figura 5.9c.

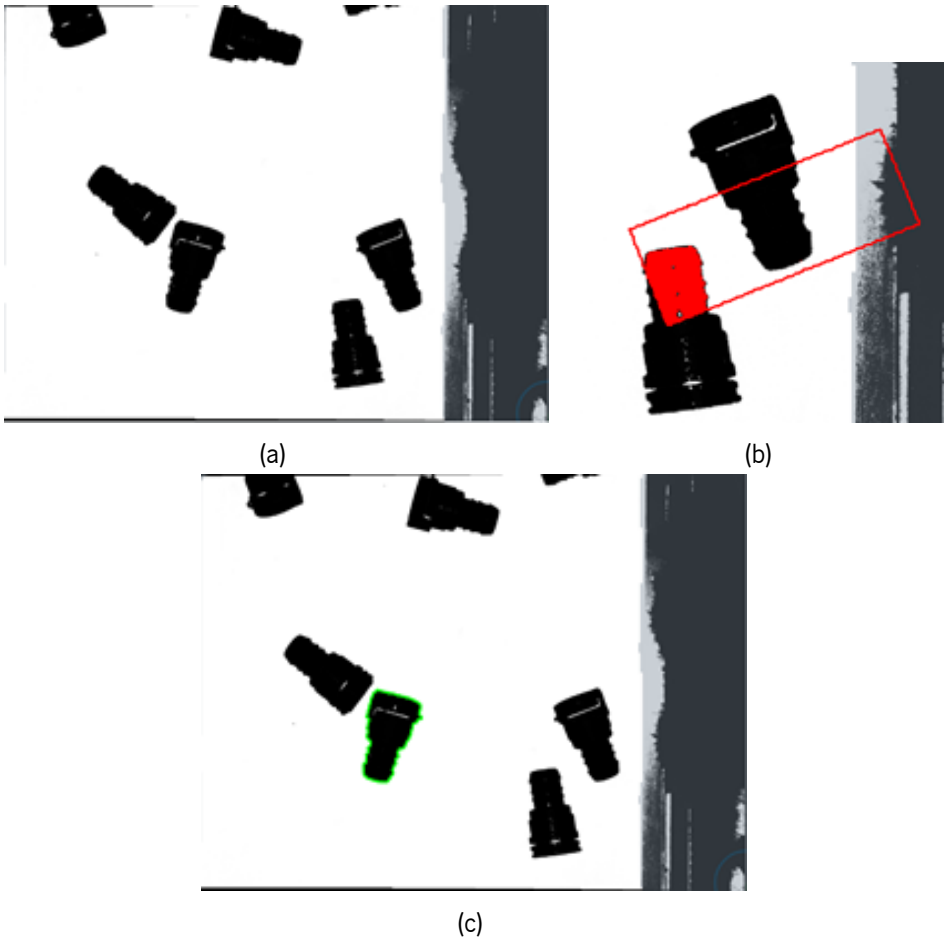


Figura 5.9: Resultados da solução de *picking* de conectores no passo de verificação da zona de *picking*: (a) imagem de teste; (b) violação da zona de *picking*; (c) resultado da solução.

5.5.3 Tubos

De forma a investigar o funcionamento da solução do problema de *picking* de tubos, existem 4 imagens disponíveis, sendo que nesse conjunto de imagens, 3 tubos possuem a orientação correta para a realização de *picking*, enquanto que 9 têm de ser descartados. Apenas 2 dos tubos orientados corretamente devem ser considerados como peças a serem escolhidas pelo *robot*. As condições de resultados positivos e negativos são similares à situação dos conectores, com exceção de como a orientação deve ser verificada. Para os tubos, basta analisar a forma que estes apresentam.

Durante os testes, foram alterados os valores de *score* mínimo de *matching* para verificar o *score* máximo que podia ser aplicado para atuar como percentagem mínima de semelhança entre o tubo en-

contrado e o tubo de referência.

A solução para *picking* de tubos comportou-se de forma similar à solução para conectores. No passo de *matching*, não foi possível encontrar todos os tubos com orientação correta com *score* mínimo superior a 70%, devido ao facto de os tubos não possuírem forma fixa, tendo como consequência pequenas alterações na forma consoante a posição que eles se dispõem nas imagens. Um exemplo de deteção correta encontra-se exposta na Figura 5.10b, obtida pela execução da solução desenvolvida com a imagem da Figura 5.10a. em que existe 1 tubo com orientação correta e não existem estruturas adicionais na zona de *picking* definida. É ainda de notar que a verificação da violação da zona designada para a garra agarrar nas peças identificou corretamente o tubo que apesar de estar corretamente orientado, não deveria ser escolhido para o *picking*, pois a área encontrava-se com estruturas adicionais, como se pode verificar na Figura 5.10d.

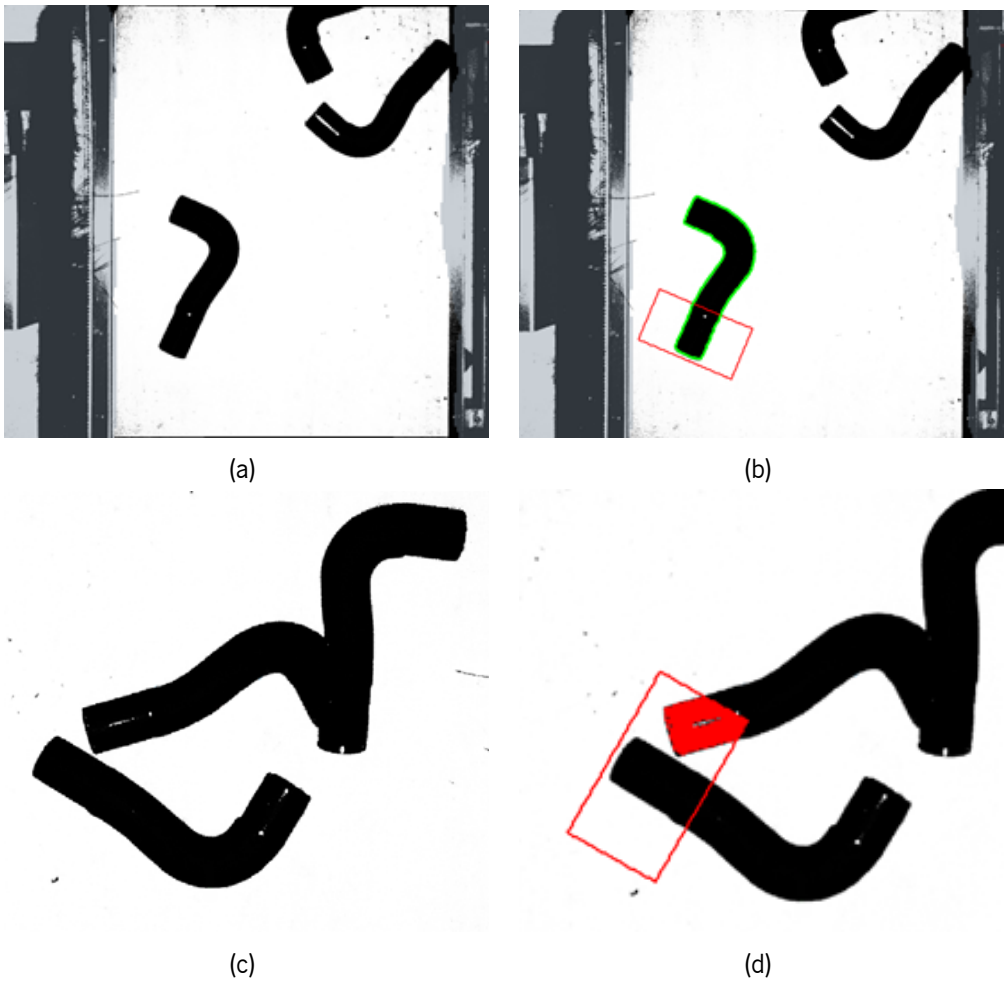


Figura 5.10: Resultados da solução de *picking* de tubos: (a) imagem de teste; (b) resultado obtido na execução da solução de inspeção para picking de tubos na imagem (a); (c) exemplo de imagem teste para verificação da zona de *picking*; (d) resultado de verificação da zona de *picking* de (c).

5.6 Sumário

De forma geral, as soluções criadas conseguem responder aos problemas de inspeção visual que são investigados neste trabalho, conseguindo identificar se cada peça se encontra devidamente concebida ou montada, detetando o incumprimento das condições de sucesso definidas.

Na inspeção de dispensação de cola, surgiram alguns problemas na deteção de situações de falha, maioritariamente relacionadas com os limites de quantidade de cola, por estes serem universais para todos os pontos de cola e definidos apenas para o processo de desenvolvimento do fluxo de operações. A solução para a inspeção de cabos fita conseguiu responder corretamente em todos os casos tanto de sucesso como de falha, se o cabo foi bem ou mal colocado, de acordo com a existência da segunda faixa de referência. Para a deteção de parafusos, houve a distinção entre presença de parafuso, ausência de parafuso e existência de autocolante, apenas falhando em imagens com complicações de iluminação. A elaboração de uma solução para a inspeção do posicionamento e dimensões de peças carece da realização de testes devido a haver uma única imagem disponível para estudo. Por fim, para a execução de tarefas de *picking*, a utilização de dois modelos de *matching* para a localização de casquilhos mostrou resultados promissores na identificação de casquilhos em diferentes posições na imagem. Nas situações de *picking* de tubos e conectores, as soluções identificaram acertadamente todas as peças elegíveis, tanto nos passos de averiguação de orientação como na verificação da presença de objetos adicionais na zona onde a garra pega na peça.

Capítulo 6

Conclusões e Perspetivas Futuras

No presente capítulo final, o trabalho desenvolvido ao longo desta dissertação é sintetizado e são expostas as conclusões associadas às abordagens desenvolvidas, após a obtenção de resultados e a discussão das observações feitas. Para finalizar, são identificadas perspetivas futuras para melhoria das soluções propostas.

6.1 Conclusão

A presente dissertação focou no desenvolvimento de uma *toolbox* de soluções de resposta a problemas de inspeção e de apoio a sistemas de IVA. Nos últimos anos, o controlo de qualidade de produtos na área industrial tem vindo a ser cada vez prevalente, provocando o aparecimento de ferrramentas capazes de criar métodos de resposta a problemas de inspeção visual, tais como *HALCON*, *Adaptive Vision*, *Sapera Vision* e *OpenCV*. Este tipo de ferramentas traz operadores de auxílio a IVA, contudo não existem toolboxes de ferramentas configuráveis para o solucionamento de problemas de inspeção específicos, sendo esta o objetivo principal desta dissertação.

O trabalho realizado incidiu no estudo de cinco problemas de inspeção visual, nomeadamente a inspeção de dispensação de cola, a deteção da ausência ou do posicionamento defeituoso de parafuso, deteção de peças como tubos, casquilhos e conectores para a realização de tarefas de *object picking*, inspeção do posicionamento de cabos fita e do posicionamento e dimensionamento de peças. A partir da utilização exclusiva dos operadores disponíveis no *HALCON*, foram desenvolvidas soluções capazes de responder a estes problemas, podendo estas serem configuráveis pelo utilizador na eventualidade de existirem alterações nos objetivos de inspeção, de forma a se adaptarem a diferentes situações. O *HALCON* demonstrou-se qualificado para responder a todas as necessidades que foram surgindo na criação das soluções.

Relativamente às soluções implementadas, o trabalho começou por focar na inspeção de quantidade de cola dispensada num conjunto de locais, tendo a solução, em geral, identificado as imagens em que não tinha sido dispensada cola dentro dos limites de quantidade exigidos. Porém, a metodologia de testagem não foi ideal devido à possibilidade da quantidade de cola necessária a ser dispensada ser, em média, diferente para cada ponto de cola e portanto não seria indicado arbitrar valores limite

de contagem de píxeis para tal. Para o segundo problema, a inspeção da colocação de cabos fita, o algoritmo conseguiu reconhecer quando a segunda faixa de referência se encontrava visível acima de uma fração máxima pré-definida, indicativo de uma má colocação. Na inspeção da presença ou ausência de parafusos, a ferramenta demonstrou ser capaz de distinguir corretamente quase todas as imagens com e sem parafuso, assim como indicar todas as imagens cujo o local de análise se encontrava tapado por um autocolante. A solução apenas falhou em dois casos, ambos associados a problemas de iluminação na captura das imagens. O problema de posicionamento e dimensionamento careceu de imagens para teste, não tendo sido possível a realização de testes com o intuito de validar a solução desenvolvida. Por fim, foi abordado o problema da deteção de tubos, conectores e anilhas para a tarefa de *picking*. As soluções desenvolvidas, para cada uma das peças, obtiveram os resultados esperados, localizando todas as peças com orientação correta e cuja zona de *picking* ser encontrava vazia além do objeto de análise e descartando as peças que não cumpriam os requisitos definidos.

Em suma, os algoritmos propostos para os problemas expostos funcionaram como desejado, permitindo a criação de uma *toolbox* de soluções para apoio a IVA de alguns problemas.

6.2 Perspetivas Futuras

No que concerne a este trabalho, existem alguns aspetos que podem ser alvos de melhoria futura. De forma geral, os problemas careceram de uma vasta base de dados com imagens representativas de todas as situações possíveis para cada problema, principalmente o problema de inspeção de posicionamento e dimensionamento. Com o intuito de colmatar as eventuais falhas presentes em cada solução, a utilização de um banco de dados mais extenso seria essencial, permitindo a otimização das soluções através de uma testagem mais exaustiva.

Outro aspeto a melhorar consiste na futura generalização de alguns processos para maior adaptabilidade a diferentes tipos de imagens do mesmo problema de controlo de qualidade. Exemplificando, a solução do problema de *picking* de casquilhos poderia ser generalizada de forma permitir a reconfiguração de como o *picking* é realizado. Ao invés de o *picking* ser realizado pela parte interior da peça, esta tarefa poderia passar a ser feita pelo exterior, sendo que se teria de verificar a zona de *picking*, tal como nos casos dos conectores e dos tubos. Ainda, a solução poderia ser adaptável para peças com formas diferentes dos casquilhos, como por exemplo objetos retangulares.

Bibliografia

[1] Alin Stncioiu. The fourth industrial revolution „industry 4.0". *Fiabilitate Şi Durabilitate*, 1(19):74–78, 2017.

[2] Li Da Xu, Eric L Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.

[3] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014.

[4] Szu-Hao Huang and Ying-Cheng Pan. Automated visual inspection in the semiconductor industry: A survey. *Computers in industry*, 66:1–10, 2015.

[5] Jürgen Beyerer, Fernando Puente León, and Christian Frese. *Machine vision: Automated visual inspection: Theory, practice and applications*. Springer, 2015.

[6] William Taylor, Brian Melloy, Pallavi Dharwada, Anand Gramopadhye, and Joe Toler. The effects of static multiple sources of noise on the visual search component of human inspection. *International journal of industrial ergonomics*, 34(3):195–207, 2004.

[7] Agnieszka Kujawinska, Katarzyna Vogt, Magdalena Diering, Michal Rogalewicz, and Sachin D Waigaonkar. Organization of visual inspection and its impact on the effectiveness of inspection. In *Advances in Manufacturing*, pages 899–909. Springer, 2018.

[8] MVTec Software GmbH. *Quick Guide HALCON 20.11 Progress*. 2020.

[9] MVTec Software GmbH. *HDevelop User's Guide - HALCON 20.11 Progress*. 2020.

[10] MVTec Software GmbH. *HALCON/HDevelop Operator Reference - HALCON 20.11 Progress*. 2020.

[11] MVTec Software GmbH. *Solution Guide II B - Matching*. 2020.

[12] MVTec Software GmbH. MVTec HALCON. URL `https://www.mvtec.com/products/halcon`. acedido em janeiro 2021.

[13] Teledyne DALSA. Sapera Vision Software Suite. URL `https://www.teledynedalsa.com/en/products/imaging/vision-software/sapera-software-suite/`. acedido em maio 2021.

[14] Teledyne DALSA. *Sapera Processing™ 9.20 C++ Programmer's Manual*. 2021.

[15] Teledyne DALSA. *Astrocyte AI Trainer User Manual*. 2021.

[16] Adaptive Vision Sp. z o.o. Adaptive Vision Library Documentation, . URL `https://docs.adaptive-vision.com/avl/`. acedido em fevereiro 2021.

[17] Adaptive Vision Sp. z o.o. Adaptive Vision Brochure 5, 2021.

[18] Krystian Radlak, Mariusz Frackiewicz, Marek Szczepanski, Michal Kawulok, and Michal Czardybon. Adaptive Vision Studio - Educational tool for image processing learning. 2015.

[19] Adaptive Vision Sp. z o.o. Adaptive Vision Deep Learning Documentation, . URL `https://docs.adaptive-vision.com/current/deep_learning/`. acedido em fevereiro 2021.

[20] Adaptive Vision Sp. z o.o. Deep Learning Inference Engine for Machine Vision, . URL `https://www.adaptive-vision.com/en/software/weaver/`. acedido em fevereiro 2021.

[21] Adaptive Vision Sp. z o.o. WEAVER 5.0 Documentation, 2020.

[22] OpenCV About. URL `https://opencv.org/about/`. acedido em maio 2021.

[23] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. "O'Reilly Media, Inc.", 2016.

[24] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. A brief introduction to OpenCV. In *MIPRO 2012 - 35th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings*, 2012.

[25] Gloria Bueno García, Oscar Deniz Suarez, José Luis Espinosa Aranda, Jesus Salido Tercero, Ismael Serrano Gracia, and Noelia Vállez Enano. *Learning image processing with OpenCV*. Packt Publishing Birmingham, 2015.

[26] Miguel Castro. *Desenvolvimento de uma Toolbox de Soluções para Inspeção Visual Automática*. Masters dissertation, Universidade do Minho, 2019.

[27] Francisco P M Oliveira, João Manuel R S Tavares, Francisco P M Oliveira, João Manuel, and R S Tavares Medical. Medical image registration : a review. *Computer Methods in Biomechanics and Biomedical Engineering*, 17(2), 2016.

[28] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2001.

[29] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys (CSUR)*, 24(4), 1992.

[30] MVTec Software GmbH. *Solution Guide I - Basics*. 2020.

[31] Yanwei Pang, Tiancai Wang, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Ling Shao. Efficient featurized image pyramid network for single shot detector. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June.

[32] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2006.

[33] Yi Yang and Shawn Newsam. Spatial pyramid co-occurrence for image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.