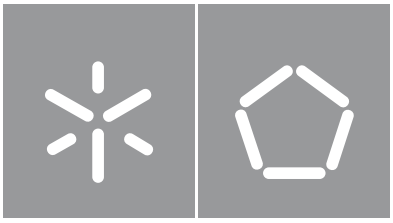


Universidade do Minho
Escola de Engenharia

Miguel Lopes Castro

Desenvolvimento de uma Toolbox de Soluções para Inspeção Visual Automática

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

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

dezembro de 2019

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

Agradecimentos

Esta pequena grande aventura teve início no ano 2014 e desde então foi uma montanha russa de aprendizagem, trabalhos, amizades. Como tudo na vida, existem sempre bons e maus momentos, mas é nos melhores que devemos celebrar e nos piores que devemos aprender, assim como eu celebrei e aprendi nestes últimos anos.

O meu primeiro agradecimento vai para o meu orientador e Professor Doutor Carlos Silva que me ajudou nesta importante etapa académica. Muito obrigado por todo o seu conhecimento e paciência.

A toda a minha família pelos conselhos e pelo carinho de sempre, mas em especial aos meus pais, Daniela e João, que sem eles nunca seria possível eu ter chegado até aqui. Obrigado por todo o esforço, suor e dedicação para que eu me tornasse no Homem que sou, apesar dos meus defeitos, nunca desistiram de mim e sempre me ajudaram, e ajudarão, a chegar cada vez mais longe. Muito obrigado!

À família Biomédica 2014/2015 e aos meus amigos do ramo de Eletrónica Médica, muito obrigado por todas as vivências partilhadas, mesmo até pelas longas horas de trabalho. Passei com vocês alguns dos momentos mais marcantes.

Aos meus Engenheiros, aos meus amigos, aos meus caloiros e às minhas Bestas agradeço pelos ensinamentos, brincadeiras e amizades. Foram uma parte importante na minha vida académica e pessoal. Vou sempre levar comigo alguns dos melhores momentos passados com vocês.

A todo o pessoal do #LabPessoasFixes pela rápida integração e pelo convívio passado neste último ano, rapidamente se tornou numa casa para mim.

Por último, o maior dos obrigados à minha namorada, melhor amiga e colega de trabalho, Alexandrine, por me mostrar que com algum jeitinho tudo se consegue. Por todas as horas passadas a chatear-me a cabeça, hoje agradeço por todas elas. Sempre presente quando mais precisei. Levo uma mulher para a vida!

De um modo geral, muito obrigado a todas as pessoas com quem me cruzei nestes últimos 5 anos, obrigado por todos os momentos passados, foram os melhores momentos da minha vida e, sem dúvida, que contribuíram para a pessoa que sou hoje.

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

Desenvolvimento de uma Toolbox de Soluções para Inspeção Visual Automática

No mundo industrial, as empresas pretendem que as suas linhas de produção apresentem elevadas taxas de produtividade e altos níveis de qualidade dos produtos. Para isso, é feita a inspeção visual por trabalhadores que, devido ao facto deste processo ser repetitivo e o nível de atenção do trabalhador tender a diminuir, podem levar a avaliações subjetivas e incorretas. A solução passa pela utilização de sistemas de inspeção visual automática, uma vez que estes permitem obter soluções reproduzíveis e ajudam a controlar a taxa de erro.

O objetivo deste trabalho passa pelo desenvolvimento de uma *toolbox* de soluções parametrizáveis, através de uma interface gráfica apropriada ao problema, de forma a poderem ser aplicadas em problemas similares. Esta dissertação foca-se na deteção de quatro problemas específicos, nomeadamente, a deteção de marcadores fiduciais, a deteção de pinos, a deteção do centro geométrico e verificação do posicionamento de ponteiros e deteção da ausência ou posicionamento defeituoso de parafusos.

Os métodos propostos para cada um dos problemas foram baseadas nos operadores fornecidos pelo *software Halcon* e validados em diferentes objetos. Os métodos desenvolvidos mostraram ser eficientes, sendo capazes de detetar corretamente os objetos. Assim, as metodologias propostas apresentam-se como promissoras para serem implementadas num sistema de inspeção visual automática, a nível industrial.

A aplicação desenvolvida centra-se na definição de micro-operações e no motor de execução. É a combinação do modo interativo da aplicação, com a *toolbox* parametrizável, que possibilita o utilizador de configurar as soluções. Relativamente aos resultados, estes vão de encontro aos objetivos estipulados, uma vez que a *toolbox* desenvolvida permite ao utilizador modificar as soluções, ajustando-as para diferentes problemas similares.

Palavras-chave: *Halcon*, inspeção visual automática, processamento de imagem, visão por computador.

Abstract

Development of an Automatic Visual Inspection Solutions Toolbox

In the industrial world, companies want their production lines to have high productivity rates and high product quality levels. For this, the visual inspection is made by workers, which may lead to subjective and incorrect evaluations because this process is repetitive, and the level of attention of the worker tends to decrease. The solution involves the use of automatic visual inspection systems as they provide reproducible solutions and help to control the error rate.

The objective of this work is the development of a solution toolbox and the implementation of parameterizable solutions so that they can be applied to similar problems, through a graphical interface appropriate to the problem. This dissertation focuses on the detection of four specific problems, namely, fiducial marker detection, pin detection, geometric center detection and pointer positioning verification, and the absence or defective positioning of screws.

The proposed methods for each problem were based on operators provided by Halcon software and validated on different objects. The developed methods proved to be efficient and robust, being able to detect the objects correctly. Thus, the proposed methodologies have shown to be promising for implementation in an automatic visual inspection system at the industrial level.

The application developed focuses on the definition of micro-operations and the execution engine. It is the combination of the interactive mode of the application with the parameterizable toolbox that enables the user to configure the solutions. Regarding the results, these meet the stated objectives, since the developed toolbox allows the user to modify the solutions, adjusting them to different similar problems.

Keywords: Halcon, automatic visual inspection, image processing, computer vision.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

3D *Três dimensões*. 5, 8, 11

EVT *Eye Vision Technology*. 6, 8, 11

GUI *Graphical User Interface*. viii, 5, 8, 10, 12, 46, 52

IDE *Integrate Development Environment*. 5–8, 10, 12

IVA Inspeção Visual Automática. xii, 1–6, 9, 11, 22, 24, 34, 41, 53–55

kB *Kilobyte*. 41, 42

LED *Light-Emitting Diode*. x, 35, 40–42, 55

MVTec *Machine Vision Technology*. 4, 5, 11, 14

NI *National Instruments*. 9–11

PCB *Printed Circuit Board*. 1, 29

ROI *Region Of Interest*. ix–xi, 5, 25–34, 36, 40, 45–48, 50–52, 54

XLD *eXtended Line Description*. vii, 20, 23, 31, 33

XML *Extensible Markup Language*. vii, 12, 21, 22, 45, 46, 48–50

Capítulo 1

Introdução

Neste capítulo apresenta-se a motivação e os principais objetivos associados ao desenvolvimento desta dissertação. Para além disto, são também indicadas as contribuições e a estrutura da dissertação.

1.1 Motivação e Enquadramento

Na era atual, existe uma grande rivalidade entre as empresas no que toca à comercialização dos seus produtos. Aumentos da qualidade dos produtos são algumas das caraterísticas mais importantes que permitem fazer a distinção empresarial. As linhas de manufaturação atuais apresentam taxas muito altas de produtividade, enquanto que, ao mesmo tempo, asseguram produtos com caraterísticas de alta qualidade. A substituição do homem pela máquina é algo recorrente nos dias de hoje e, até mesmo estas, acabam por ser trocadas por outras mais modernas e autónomas, de gerações superiores [1]. Tudo isto com o intuito de aumentar a produção e a qualidade das linhas de montagem. Porém, as máquinas ainda não são totalmente independentes, o que implica que o processo de produção esteja em constante supervisão por parte de pessoas qualificadas, uma vez que, mesmo nas linhas modernas de montagem ocorrem erros durante a manufaturação que levam à conceção de produtos não funcionais ou parcialmente funcionais e, consequentemente, a perdas económicas significativas [2].

A inspeção visual realizada por pessoas pode tornar-se numa tarefa extremamente repetitiva e exaustiva, tornando a inspeção subjetiva e errática. Outra das razões envolve o facto de, para este tipo de trabalho, o desempenho do humano tender a diminuir. Estes fatores associados aos elevados custos salariais, levam a que a inspeção visual automática (IVA) se revele uma solução vantajosa [1, 3].

Sistemas como a IVA consistem em processos de controlo de qualidade baseados em visão por computador, como demonstra a Figura 1, onde é representado um sistema básico. Hoje são várias as áreas que recorrem a este tipo de aplicações, por exemplo, no reconhecimento automático de alvos para diagnóstico médico, na interpretação de fotos aéreas para planeamento geográfico e em inúmeras atividades industrias. Aliás, estes sistemas, na sua maioria, são aplicados na indústria, mais propriamente na área automóvel e na inspeção automática de *printed circuit board* (PCB) [4, 5]. No mundo industrial, a inspeção é um aspeto com enorme relevância e é aqui que a IVA atua, ajudando na avaliação da qualidade dos produtos manufaturados recorrendo à informação visual, como por exemplo, uso de câmaras. Pode ser

utilizada para análise do produto final, bem como, durante todos os passos das linhas de montagem, visto que, quanto mais cedo uma falha for detetada, menor será a probabilidade desse produto ser rejeitado na avaliação final, contribuindo de forma económica para a empresa e mostrando a importância deste tipo de sistemas em diferentes tipos de manufaturação [4, 6, 7].

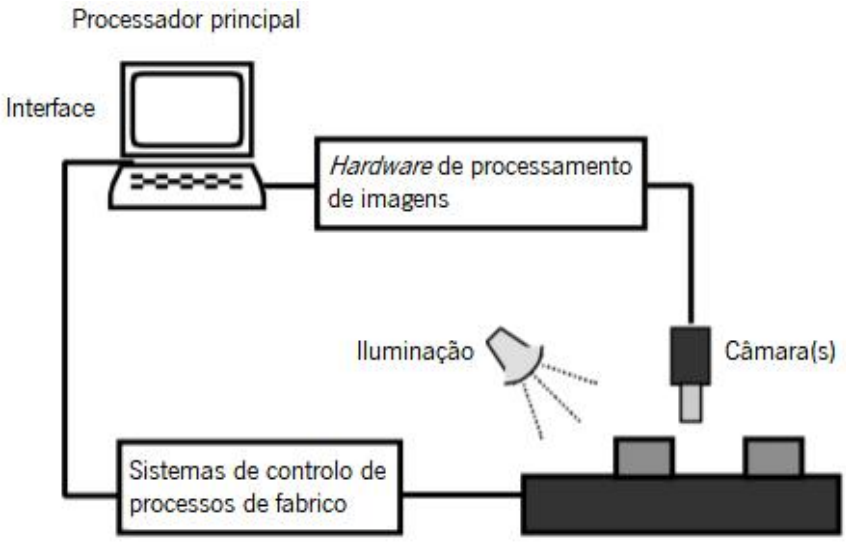


Figura 1: Sistema estrutural básico dos sistemas de inspeção visual. Adaptado de [8].

Um dos maiores problemas que afeta os sistemas de IVA é a variação de caraterísticas, tais como, os níveis de iluminação, o tipo de material de leitura utilizado e as caraterísticas dos produtos. Consequentemente, isto leva a falhas de deteção por parte dos sistemas de IVA na deteção de um desvio excessivo nos parâmetros de qualidade na produção, sendo necessário o reajuste às novas condições [4]. Assim como, um novo ajuste deverá ser realizado para novas peças que apresentarão, certamente, novos formatos, diferentes dimensões e diferentes localizações de componentes. Esta alteração deve ser preferencialmente feita na própria linha de montagem, minimizando o impacto na produção.

1.2 Objetivos

Neste trabalho pretende-se contribuir para a resolução de algumas das limitações detetadas nos sistemas de IVA atuais, permitindo um maior nível de qualidade dos produtos criados, através de uma *toolbox* de soluções parametrizável. Reconhecendo também a qualidade das soluções existentes, o principal foco passa pelo desenvolvimento de soluções configuráveis para problemas específicos, com recurso ao sistema *Halcon*, devido à sua abrangência e maturidade. Os principais objetivos para o projeto serão:

- Deteção e reconhecimento de diferentes marcadores fiduciais para, por exemplo, auxiliar a localização de outros componentes eletrónicos inseridos em placas;
- Deteção de pinos corretamente posicionados em fichas;
- Deteção do centro geométrico de objetos para a colocação de ponteiros e respetiva verificação do seu correto posicionamento;

- Deteção da ausência ou posicionamento defeituoso de parafusos.

As soluções desenvolvidas também serão configuráveis, de modo a poderem serem aplicadas noutros problemas com diferentes propriedades. Tudo isto será implementado numa interface gráfica adequada aos problemas acima transcritos.

1.3 Contribuições

Ao longo deste trabalho foram desenvolvidos métodos que podem ser considerados contribuições. Com isto, o intuito será contribuir no âmbito tecnológico e industrial através da resolução de alguns problemas atuais considerados críticos nas linhas de montagem, nomeadamente:

- O desenvolvimento de uma *toolbox* de soluções para a deteção de objetos específicos, baseadas num conjunto de problemas existentes na área industrial, de modo a ajudar os sistemas de IVA na validação de produtos;
- Uma aplicação interativa que permita a parametrização das soluções criadas, de forma a poderem ser utilizadas em casos semelhantes.

1.4 Estrutura da Dissertação

A presente dissertação encontra-se dividida em 6 capítulos. No capítulo 1 está a introdução do trabalho, nomeadamente a motivação, o enquadramento, objetivos e as contribuições do trabalho. O capítulo 2 descreve e compara diferentes ambientes de desenvolvimento de apoio à IVA, mostrando algumas das suas funcionalidades e caraterísticas. De seguida, o capítulo 3 contém o conhecimento sobre as ferramentas do *Halcon*, assim como alguns conceitos sobre alguns dos operados *Halcon* utilizados neste trabalho. Posteriormente, o capítulo 4 diz respeito à implementação da *toolbox* de soluções. No final, é feita a discussão dos resultados obtidos para cada uma das soluções desenvolvidas. Seguidamente, no capítulo 5 é descrita a estrutura da aplicação implementada e a forma ocorre a ligação da *toolbox* com a aplicação. São descritos os dois modos de funcionamento e, no fim do capítulo, são apresentados os resultados obtidos por estes. Por último, no capítulo 6 encontram-se as principais conclusões e possíveis sugestões de trabalho futuro.

Capítulo 2

Estado da Arte

A inspeção visual consiste num método de supervisionamento de todo o processo de produção de uma atividade, operação ou produto. Tudo isto acontece sem nunca interferir no seu meio, sendo uma tarefa totalmente visual, com a intenção de avaliar as condições ou qualidade de algo que está a ser inspecionado. Este procedimento sempre existiu no nosso dia-a-dia, pois ainda antes dele ser automatizado, era feito por operadores especializados que, através dos seus olhos, faziam a inspeção dos produtos durante todos os passos da sua criação e, especialmente, na sua avaliação final.

O presente capítulo faz uma comparação entre alguns dos ambientes de desenvolvimento de imagem mais avançadas no mercado atual e são discutidas vantagens e desvantagens de cada um deles. Também serão feitas referências a algumas das suas funcionalidades.

2.1 Ambientes de Desenvolvimento de Apoio à Inspeção Visual

Na atualidade, já existem ambientes de desenvolvimento e bibliotecas de visão por computador capazes de auxiliar a inspeção visual ao longo da linha de manufaturação. É possível encontrar diversos conjuntos de soluções para o desenvolvimento de sistemas de IVA que respondem a problemas de visão por computador, sendo da responsabilidade do utilizador escolher e sequenciar os procedimentos que serão mais favoráveis e adequados para a resolução do seu problema e, com isto, alcançar os melhores resultados [9].

2.1.1 Halcon

O *Halcon* [10] foi desenvolvido pela *Machine Vision Technology* (MVTec), sendo um *software* de visão por computador. É uma ferramenta profissional com uma biblioteca que inclui uma grande variedade de operadores, básicos e avançados, de processamento de imagem. Além disso, fornece métodos de paralelização para explorar um objeto com vários processadores, através da divisão da imagem, ou vários núcleos para acelerar processos, em paralelo com *threads*. O *Halcon* não só providencia interfaces para centenas de câmaras e *frame grabbers*, como também suporta os sistemas operativos mais comuns e linguagens de programação [11].

O desenvolvimento do *software* é suportado pelo seu próprio ambiente de desenvolvimento integrado (IDE[1]), composto pelo *HDevelop* e pelo *HDevEngine*. Possui cerca de dois mil operadores associados às áreas de pré-processamento, segmentação, morfologia, extração de caraterísticas, classificação, reconhecimento de objetos, reconhecimento e verificação de caracteres e visão a três dimensões (3D).

A biblioteca do *Halcon* é baseada numa arquitetura aberta, ou seja, o utilizador pode estender o *Halcon* ao integrar a sua própria funcionalidade de visão na forma de novos operadores. Também é possível usar dispositivos de aquisição de imagens que o *Halcon* não suporta, carregando e abrindo as imagens diretamente no ambiente ou criando uma interface de aquisição de imagens para o dispositivo [11, 12].

2.1.2 Merlic

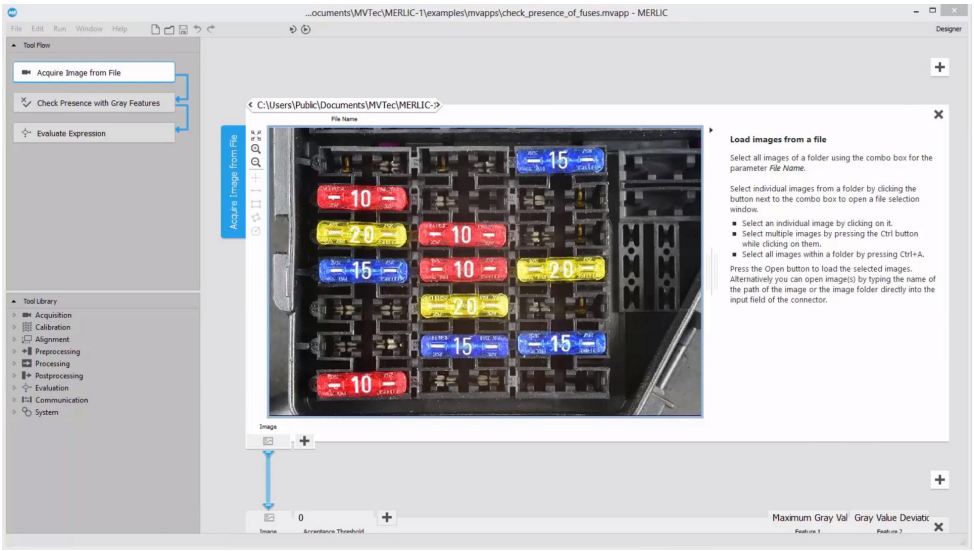
A MVTec além de ter lançado o *Halcon*, também lançou o *Merlic* [13]. Com recurso ao *Merlic* rapidamente se pode conceber aplicações de visão por computador, sem nenhuma linha de programação, ao contrário de outros ambientes de desenvolvimento que obriga o utilizador a ter conhecimentos sobre programação [9]. A combinação entre o desempenho juntamente com a facilidade do seu uso e desenvolvimento, possibilita o projeto de aplicações de visão por computador mesmo por parte de trabalhadores inexperientes, sem qualquer cultura em programação. Ao contrário do que sucede no *Halcon*, o *Merlic* recorre à linguagem gráfica para caraterizar o seu algoritmo, constituindo o *back-end* de uma aplicação. Além disso, também permite a construção do *front-end* gráfico através de uma palete de *widgets*. Os elementos da interface gráfica podem ser conetados a entradas e saídas do algoritmo, seguindo assim um fluxo condutor. O *Merlic*, também escolhe os melhores valores para a configuração dos métodos de visão por computador e assim minimizar o número de interações necessárias. Permite ainda acelerar o desenvolvimento de aplicações de IVA comparativamente ao *HDevelop*.

Na Figura 2 está representado um pequeno programa dividido nas suas componentes principais, a interface do *back-end* e a interface do *front-end*. Estas foram criadas de modo a serem muito claras e reduzidas, centrando-se na janela gráfica, e consequentemente na imagem carregada, para possibilitar um processamento diretamente na imagem, sem a necessidade de escrever código. Devido a esta organização e à funcionalidade *easyTouch*, é possível definir regiões de interesse (ROIs[2]) e os parâmetros das funções automaticamente. Ao passar o rato sobre a imagem, o *easyTouch* reconhece e marca objetos que identificou. Esta funcionalidade orienta o utilizador em direção a uma solução de uma forma interativa, oferecendo vários tipos de solução para uma mesma imagem. Na interface gráfica do utilizador (GUI[3]) representada na Figura 2b, também designado como *Merlic Designer*, com recurso às funções de arrastar e soltar *widgets*, disponíveis no agregado lateral, o *Merlic Designer* vincula os *widgets* aos parâmetros das ferramentas, sendo depois possível visualizar os valores desses parâmetros. Com esta simplicidade, os utilizadores são facilmente capazes de desenvolver aplicações [14].

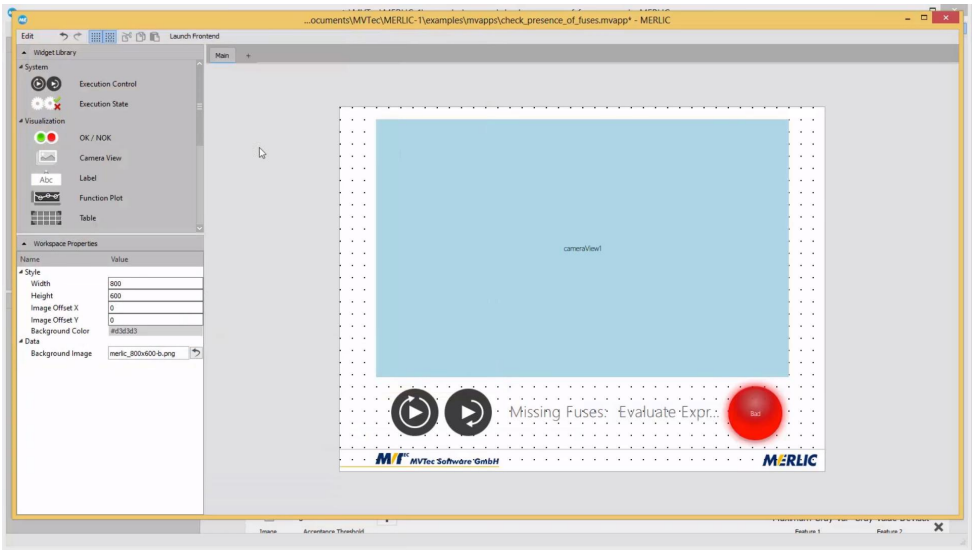
Relativamente à sua biblioteca integrada, todas as operações são baseadas nas mais recentes tecnologias de visão. Existem ferramentas que dão suporte a uma variedade de processos de inspeção e,

[1] Do inglês: *integrate development environment*
[2] Do inglês: *region of interest*
[3] Do inglês: *graphical user interface*

consequentemente, de avaliação, como por exemplo, a realização de cálculos ou deteções de defeitos de uma região específica. Depois da sua configuração, o *Merlic* fornece ferramentas de comunicação onde será possível enviar e receber ou ler e gravar dados. Adicionalmente, a biblioteca poderá ser expandida através de ferramentas personalizadas definidas pelo próprio utilizador [14].



(a)



(b)

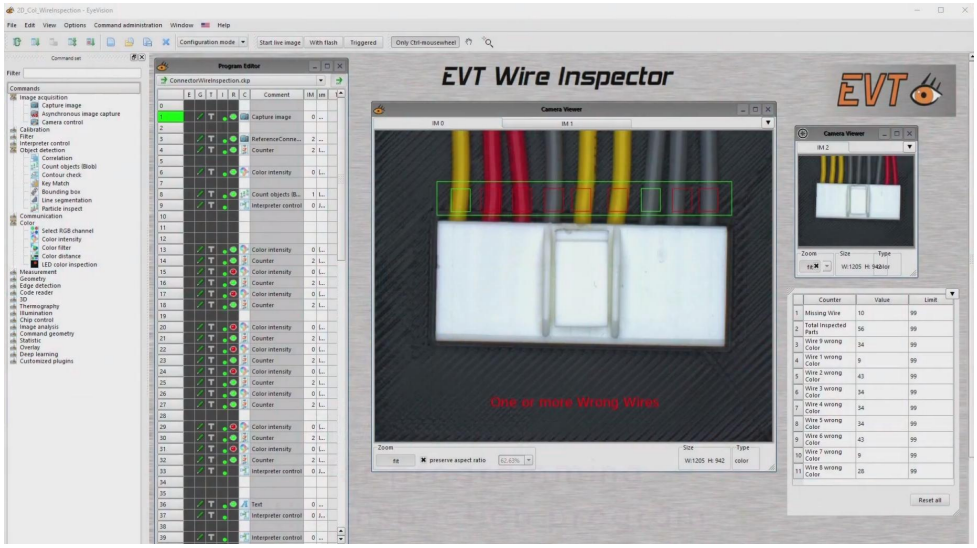
Figura 2: Exemplo de um programa produzido no *Merlic* que consiste na verificação da presença de fusíveis. Está divido nas suas interfaces gráficas: (a) *back-end*; (b) *front-end* [15].

ESTOU A AQUI A RELER!!!

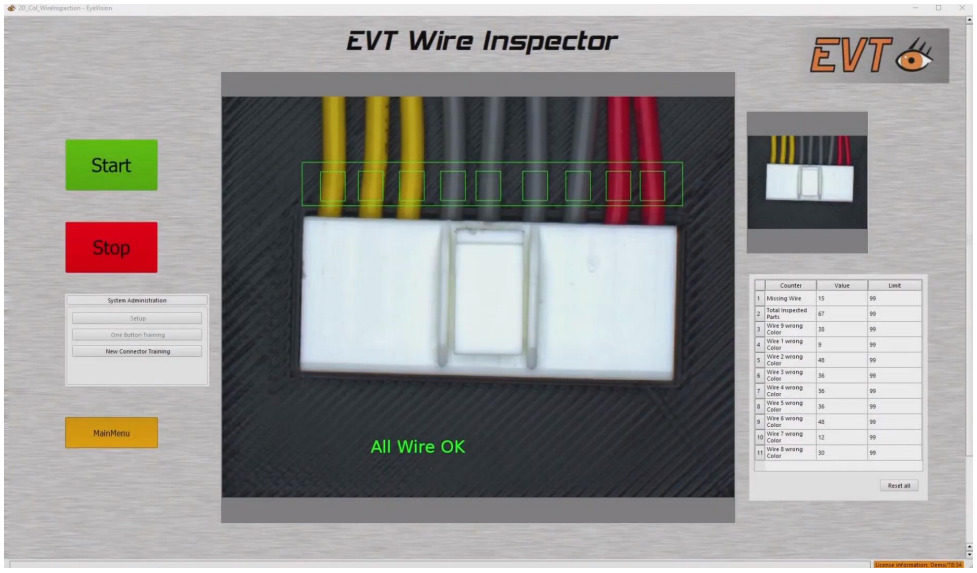
2.1.3 EyeVision

Analogamente ao *Halcon*, a empresa *Eye Vision Technology* (EVT) também desenvolveu um IDE para sistemas de IVA, o *EyeVision* [16]. Com várias parecenças ao *Merlic*, o *EyeVision* permite construir aplicações com recurso a palete de *widgets*, de uma forma totalmente gráfica, sem requerer programação.

Contudo, a organização do algoritmo é agrupada consoante a finalidade do método. O *front-end* da aplicação e o programa são realizadas em simultâneo, através um painel próprio. Igualmente ao *HDevelop*, o IDE possibilita executar o programa na sua totalidade ou passo a passo. Através deste último método referido, o utilizador tem acesso às variáveis resultantes, pois assim será capaz de definir qual a melhor série de valores para os diferentes procedimentos. Para além de fornecer um vasto leque de implementações de métodos, uma das maiores potencialidades do *EyeVision* é o facto de esta poder integrar diferentes bibliotecas de visão por computador, como por exemplo, incluir o *Halcon*. Também oferece uma gestão de todas as caraterísticas medidas e testadas para um banco de dados possibilitando o rastreamento de produção para futuras reavaliações [17].



(a)



(b)

Figura 3: Exemplo de um programa criado no *EyeVision* que consiste na verificação e posicionamento de fios elétricos. Apresenta: (a) o *back-end*, onde e como foi elaborada resolução do problema; (b) a aplicação a decorrer, também denominado como *front-end* [18].

A interface do *EyeVision* possui a funcionalidade de arrastar e soltar contribuindo para a flexibilidade

e facilidade de manipular uma aplicação, até mesmo para pessoas que não são especialistas em programação. Como exemplificado na Figura 3, a sua interface não é tão simplificada quanto a do *Merlic* e tanto o *back-end* como o *front-end* encontram-se na mesma GUI. Contudo existe uma separação entre os métodos selecionáveis e o fluxo de execução e a visualização da imagem e dos resultados, sendo já *widgets* que irão ser compilados para a aplicação final (Figura 3b) [17].

Dependendo do tipo de aplicação, a EVT disponibiliza pacotes de *software EyeVision* para expansão e são eles: *Basic*, *Standard*, *Professional*, 3D *Area or Profile* e *Thermo*. Cada um deles têm diferentes conjuntos de instruções, não existindo qualquer limitação para a sua capacidade de expansão, sendo possível adicionar qualquer comando a qualquer momento [17].

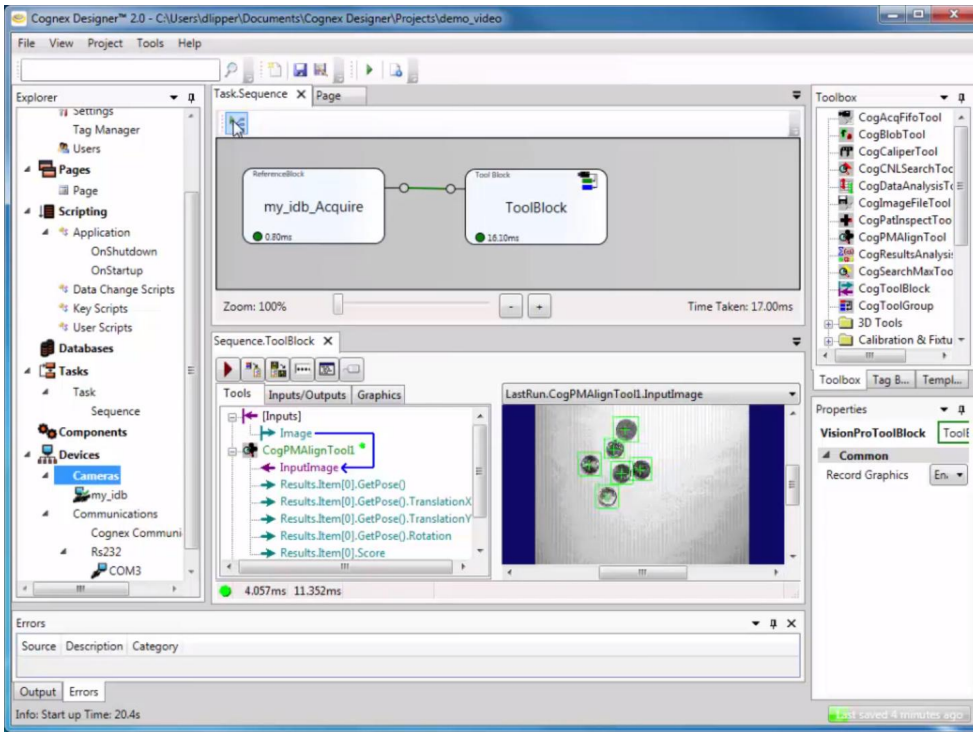
2.1.4 VisionPro

Um outro IDE gráfico para os sistemas de visão é o *VisionPro* [19], fabricado pela *Cognex*, projetado para configurar e formular aplicações de visão. Apresenta uma estratégia de resolução idêntica ao *Merlic* e ao *EyeVision*, no entanto é uma ferramenta mais elaborada. O *VisionPro* apresenta uma versatilidade da representação gráfica das entradas e saídas de um módulo, embora menos polida que o *Merlic* e o *EyeVision*. É ainda possível definir novos módulos através de *scripts* e integra-los na parte gráfica do programa desenvolvido, assim como, proporcionar uma interação direta com o código fonte, (*C#* ou *Visual Basic*), da aplicação e das bibliotecas [20]. Além disso, como o suporte na definição da GUI é mais avançado, permite definir *templates* para diferentes partes da GUI, podendo até ser reutilizadas noutras aplicações. Os modos de resolução, tais como, leitores de código de barras, deteção de caraterísticas da peça, verificação de montagens, de objetos e automação, desenvolvidos pelo *VisionPro* ajudam a melhorar a qualidade e desempenho das linhas de manufaturação, além disso, têm a funcionalidade de rastrear todas as etapas do processo de produção [9].

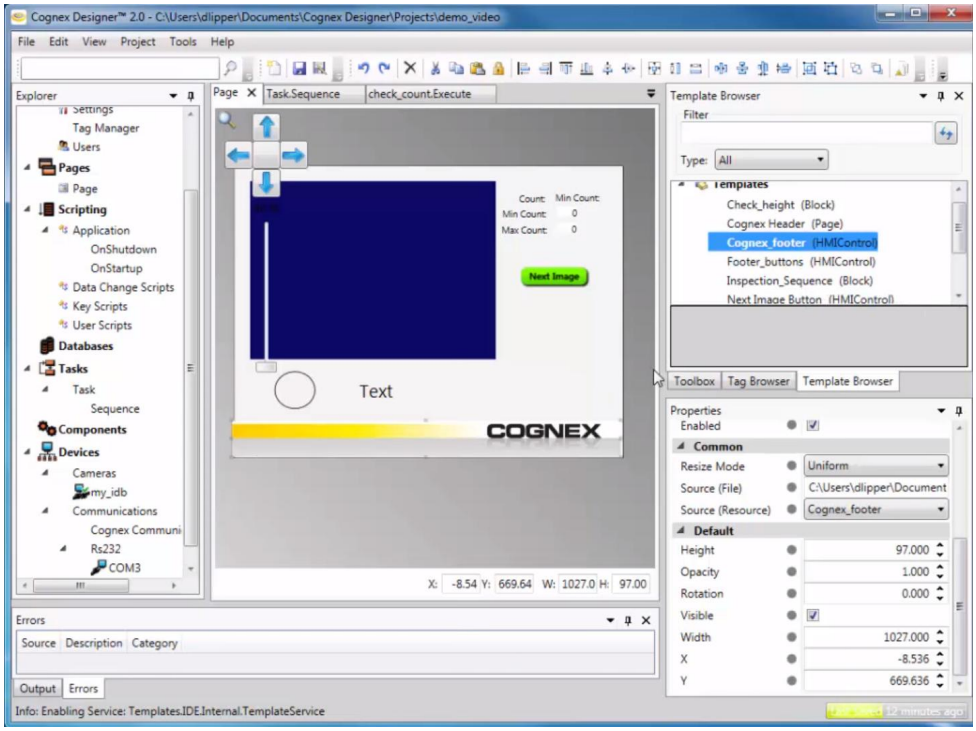
Além disso, a *Cognex* também criou uma vertente chamada de *VisionPro ViDi*. Este é um *software* de análise de imagem com base na tecnologia *deep learning* projetado especificamente para automação. Associando a inteligência artificial com as propriedades do *VisionPro* e do *Cognex Designer*, o *VisionPro ViDi* resolve problemas de uma forma mais rápida e eficaz. É considerado ideal para deteção de defeitos, classificação de texturas, verificação e localização de objetos e leitura de carateres[4] [21].

Tal como o *Merlic* e o *EyeVision*, também o *VisionPro* tem a opção de arrastar e soltar, facilitando a comunicação das ferramentas inseridas com os resultados obtidos. No entanto, também se pode optar por configurar manualmente as ferramentas por código. A parte gráfica recorre ao *Cognex Designer* que facilita o desenvolvimento, a implementação e a manutenção da aplicação, isto sem o uso de programação. A Figura 4 apresenta as interfaces gráficas facultadas pelo *VisionPro*. Assim como no *Merlic*, o *back-end* e o *front-end* são interfaces separadas. A programação é substituída pela função de arrastar e soltar blocos, como se vê na Figura 4a. Através das ferramentas do *Cognex Designer*, o utilizador pode idealizar a parte visual da aplicação, tendo que criar e ajustar todos os blocos para tal. No "Test Mode", ocorre a combinação de ambas as interfaces, resultando na aplicação final [21].

[4] Do inglês: *optical character recognition*



(a)



(b)

Figura 4: Exemplo de um programa criado no *VisionPro* que consiste na contagem e verificação moedas. Constituído pelas interfaces: (a) *back-end*; (b) *front-end* [22].

2.1.5 Vision Builder

Mais um concorrente para o apoio aos sistemas de IVA é o *Vision Builder* [23] criado pela *National Instruments* (NI). O *Vision Builder* é um *software* interativo que tem como função desenvolver e implementar

sistemas de visão. Este é dividido em cinco áreas principais, aperfeiçoamento de imagens, verificação, localização, medição e identificação de objetos [24]. Fornece um fluxo orientado por menus, que permitem auxiliar o operador na criação de sistemas automatizados. A NI desenvolveu um IDE que segue o mesmo fluxo de desenvolvimento encontrado noutras ferramentas desenvolvidas pela mesma, como por exemplo, o *LabView*. Uma das suas particulares é o desenvolvimento da aplicação utilizando linguagem gráfica com uma hierarquia de tipos de barramento para descrever a ligação entre os componentes gráficos da aplicação. O que diferencia o *Vision Builder* é o facto da validação e conversão de tipos ser feita de um modo muito simples. Outra ponto a favor é a facilidade na ligação de outras ferramentas ou módulos escritos noutras linguagens de programação para simular ou implementar algoritmos [25, 26].

O *Vision Builder* permite a configuração e inspeção de tarefas automatizadas que podem ser executadas independentemente. Por um lado, a configuração permite ajustar as etapas de inspeção que contêm as funções aplicadas à imagem a ser processada. Por outro lado, a inspeção indica os resultados e a estatística dessa mesma inspeção [24]. Relativamente à sua GUI, visualizada na Figura 5, esta também é focada na janela gráfica, onde contém a imagem, e resto da interface centra-se nos passos da resolução do problema, ou seja, nos procedimentos a que podem ser recorridos e a construção do programa efetuado. Verifica-se que a visualização do fluxo de execução assemelha-se a um diagrama de blocos, seguindo as mesmas diretrizes que o *LabView*, como anteriormente explicado. O *display* dos resultados pode ser configurado dentro da janela gráfica pois o *Vision Builder* não fornece um *front-end*, assim como o *Halcon* [25].

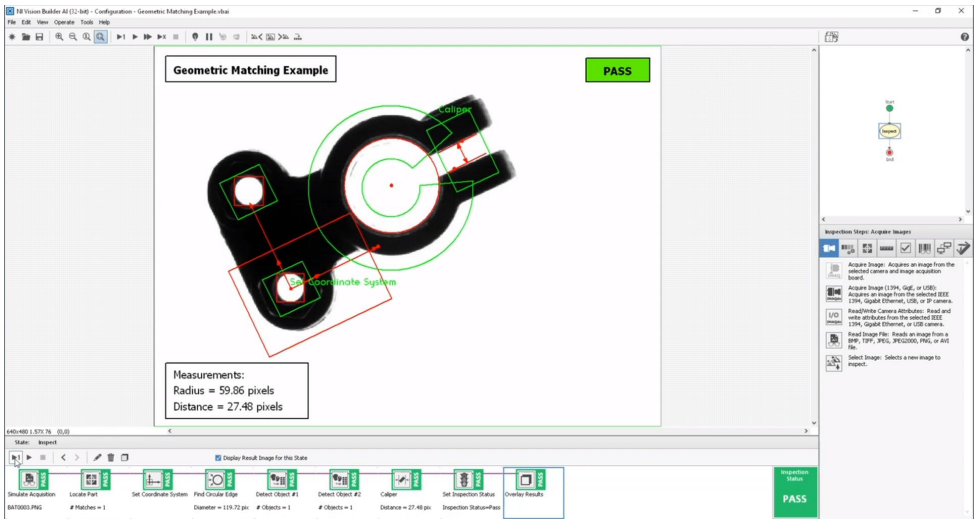


Figura 5: Exemplo de um programa desenvolvido na GUI do *Vision Builder* que consiste na correspondência geométrica de um objeto e medições de formas e distâncias [25].

2.2 Comparação dos Ambientes de Desenvolvimento

A Tabela 1 apresenta, de entre as funcionalidades mais utilizadas no mercado atual, quais os recursos fornecidos por cada um dos ambientes de desenvolvimento, de forma a fazer uma comparação sintetizada.

Tabela 1: Comparação entre as funcionalidades das ferramentas de apoio à IVA [9, 14, 17, 20, 21, 24–26].

	Halcon (MVTec)	**Merlic (MVTec)**	**EyeVision (EVT)**	**VisionPro (Cognex)**	**Vision Builder (NI)**
Calibração	X	X	X	X	X
Medições	X	X	X	X	X
Alinhamento	X	X	X	X	X
Verificação	X	X	X	X	X
Posicionamento	X	X	X	X	X
Leitura de caracteres, códigos *de barras e QR codes*	X	X	X	X	X
Deteção de defeitos	X	X	X	X	X
Processamento em visão 3D	X	X	X	X	X
Biblioteca expansível	X	X	X		
Paralelização/ *Multithreading*	X	X	X		
Múltiplos suportes de visão	X	X	X	X	
Front-end próprio		X	X	X	
Execução total ou parcial	X	X	X	X	X
Integração de diferentes bibliotecas			X		
Inspeção de cor	X	X	X	X	X
Deep learning	X	X	X	X	

2.3 Sumário

Na atualidade, já existem ambientes de desenvolvimento e bibliotecas de visão por computador com a capacidade e o poder de auxiliar a inspeção visual ao longo da linha de manufaturação, para diferentes tipos de avaliações. Cada ferramenta tem as suas caraterísticas, o que as torna únicas, trazendo vantagens e desvantagens dependendo do tipo de solução ou aplicação que se pretenda desenvolver. Da análise dos ambientes de desenvolvimento supramencionados verifica-se que estes são os mais completos e robustos para o mercado industrial. Atendendo às suas funcionalidades, constata-se que estes ambientes de desenvolvimento têm a capacidade de reduzir a complexidade do desenvolvimento de um sistema de IVA, como por exemplo, ao ajustar as variáveis de entrada dos operadores para a obtenção de melhores resultados. Porém, apesar de todas estas garantias oferecidas pelos atuais ambientes de desenvolvimento, ainda existem pontos críticos que não estão a ser abordados, tais como, a existência de *toolboxes* para determinados problemas industriais. Apesar destas desvantagens, estes ambientes proporcionam um vasto conjunto de métodos que respondem a problemas de visão por computador, sendo da responsabilidade do utilizador escolher e sequenciar os operadores que serão mais favoráveis e adequados para a resolução do seu problema e, com isto, alcançar os melhores resultados.

Capítulo 3

Halcon

Nesta secção, será feita uma apresentação das potencialidades utilizadas do *Halcon* e das suas ferramentas, nomeadamente do *HDevelop* e do *HDevEngine*. Além disso, aqui também serão explicados alguns conceitos necessários para este trabalho, tais como, a influência que alguns operados *Halcon* têm nos programas e a utilização dos ficheiros de procedimento *Halcon* como ficheiros *extensible markup language* (XML).

3.1 HDevelop

No *Halcon*, um algoritmo é desenvolvido usando uma linguagem *script* suportada pelo seu próprio IDE gráfico denominado de *HDevelop*. A Figura 6 apresenta a interface do *HDevelop* e a composição das suas quatro janelas principais:

- *Graphics window* - essencialmente é onde é feita a apresentação dos resultados gráficos dos programas desenvolvidos. Possibilita ainda a manipulação da mesma ou do seu conteúdo, tais como, modificação das dimensões da janela ou da imagem, movimentação, *zoom in* e *zoom out* da imagem;
- *Variable window* - dispõe as variáveis que irão ser processadas pelo algoritmo, sendo estas divididas em variáveis icónicas (imagens, regiões, contornos) e variáveis de controlo (números, *strings*);
- *Operator window* - procura operadores/funções existentes na biblioteca do *Halcon* ou criados(as) pelo utilizador. Identifica quais as variáveis icónicas e de controlo, de entrada e de saída e os seus respetivos nomes e valores;
- *Program window* - permite a construção e edição de código através de descrição textual. Podem ainda ser adicionados ou removidos *breakpoints*, que são pontos de interrupção no programa que param a execução do mesmo nos locais definidos pelo utilizador, para que se possa proceder à depuração do algoritmo por examinação dos resultados intermédios.

Esta é uma GUI interativa que permite um rápido progresso de tarefas de processamento de imagens. Com recurso a imagens adquiridas por câmaras, ou imagens carregadas a partir de ficheiros, o IDE pode

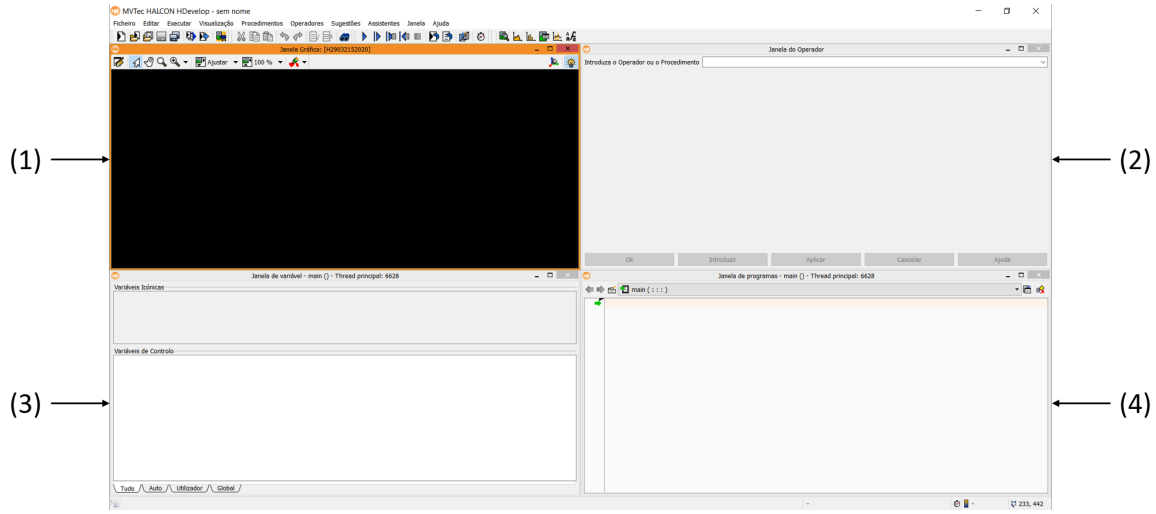


Figura 6: Interface do *HDevelop* composto por: (1) *graphics window*; (2) *operator window*; *variable window*; (4) *program window*.

executar o *script* na sua totalidade ou passo a passo. Este último traz a vantagem de se poder alterar os parâmetros de entrada de cada função e visualizar de que forma estes afetam a análise na imagem e os seus respetivos resultados. Como alternativa, o algoritmo desenvolvido poderá ser exportado para outras linguagens de programação, *C*, *C++*, *C#* ou *Visual Basic*, e inserida numa aplicação com uma interface gráfica, porém este é um método tradicional e não permite a sua depuração, além de, se for necessária alguma alteração no código é preciso voltar a exportar e recompilar a aplicação. [27, 28]. Na Figura 7 está ilustrado o fluxo de trabalho do *HDevelop*, onde o programa criado no *HDevelop* é exportado como código fonte *C*, *C++*, *C#* ou *Visual Basic* e integrado numa aplicação, como o *Visual Studio*. Depois, o programa é compilado e origina a aplicação final.

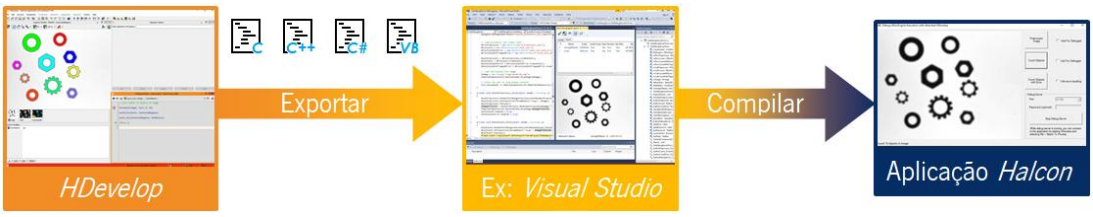


Figura 7: Código *HDevelop* a ser exportado e integrado numa aplicação. Adaptado de [29].

Na interface gráfica do *HDevelop*, os operadores, as variáveis icónicas e as variáveis de controlo, podem ser selecionadas diretamente no ambiente. Como a visualização dos resultados é imediata, é possível experimentar diferentes abordagens, operadores e parâmetros na janela gráfica. Também é possível a visualizar os resultados de um operador sem alterar um programa. Dentro dos operadores, estes são sugeridos para as tarefas específicas, além de existir informação sobre cada um dos operadores *Halcon* relativamente à sua funcionalidade, tais como, operadores sucessores e predecessores, variáveis de entrada, saída e o seu tipo, complexidade do operador e exemplos de aplicação [27]. Quando os programas estão finalizados, estes são guardados como programas de procedimentos, do tipo *.hdev*. Este tipo de ficheiro pode ser carregado e executado numa aplicação, sendo possível extrair variáveis icónicas e de controlo, no entanto, é impossível passar parâmetros de entrada.

3.2 HDevEngine

A partir do *Halcon* 13 surgiu uma nova forma de exportar e incluir código nas aplicações através do *HDevEngine*. O *HDevEngine* é um motor de execução de *scripts* do *Halcon*. Ele atua como um intérprete e permite carregar e executar procedimentos *Halcon* numa máquina virtual da MVTec, a partir de uma aplicação escrita noutra linguagem. Na Figura 8 é representada a forma como o *HDevEngine* interage com o *HDevelop* e a aplicação, em tempo real, e as vantagens que advêm com a inclusão desta funcionalidade. Esta ferramenta veio colmatar uma das desvantagens do *HDevelop*, passando a ser possível efetuar depuração remota do sistema de IVA na linha de montagem, isto é, o operador é capaz de modificar o programa ou procedimento *HDevelop*, alterando a parte da visão na aplicação, sem a necessidade de voltar a compilar, traduzindo diretamente em economia de tempo e custo. As linhas de produção podem ser rapidamente ajustadas para lidar com diferentes objetos e as unidades de inspeção podem ser equipadas rapidamente com o novo código [30].

Figura 8: Modo de funcionamento e vantagens do *HDevEngine* no desenvolvimento de uma aplicação. Adaptado de [31].

Outra das suas vantagens é o facto de poder passar valores como parâmetros de entrada. Este tipo de ficheiro é chamado de ficheiro de procedimento, do tipo *.hdvp*, podendo atuar como ficheiro *.hdev*, mas não o inverso, dado que o primeiro aceita entradas a partir de uma aplicação, ao contrário do segundo tipo. O carregamento, execução e extração de resultados acontece da mesma forma. Contudo existe uma limitação, visto que a depuração terá de ser realizada a partir da interface do *HDevelop*, pois o seu protocolo é privado [30].

A integração do *HDevEngine* para uma aplicação acontece devido à funcionalidade de "Exportar biblioteca" do *HDevelop*. Este recurso fornece um *wrapper C++* e *C#* com todo o código necessário para a programação do *HDevEngine*. Assim o utilizador apenas precisa de incluir as bibliotecas do *HDevelop* e *HDevEngine* no projeto da aplicação. Depois os procedimento *Halcon* podem ser chamados como qualquer outra função do *C++* ou *C#*.

3.3 Operadores Halcon

Os operadores utilizados para o processamento de imagem neste trabalho são todos fornecidos pelo *Halcon*. Todos estes dados são disponibilizados e descritos na documentação referente aos operadores.

3.3.1 Modelo de Forma

No *Halcon*, a identificação de objetos pode ser efetuada recorrendo a modelos de correspondência baseados em formas. A ideia adjacente a estes modelos consiste em primeiro lugar na especificação e criação de um modelo e, em segundo lugar, na utilização do modelo para encontrar e localizar um objeto noutra imagem. O resultado da fase de procura de uma correspondência pode ser otimizado, restringindo o espaço de pesquisa [32, 33].

Para criar um modelo para a correspondência baseada em forma, primeiro uma região de interesse que cubra o objeto na imagem de referência deve ser especificada. Os modelos de forma podem ser criados usando diversos operadores *Halcon*. O modelo de forma simples é o mais conveniente no entanto falha caso o modelo e o objeto a detetar se encontrem em escalas distintas. Neste caso, o modelo de forma com escala deve ser utilizado, uma vez que este permite que haja um dimensionamento do modelo [32, 33].

A correspondência baseada em formas permite localizar várias instâncias do mesmo modelo. Se o objeto for encontrado, as suas propriedades, como posição, rotação e escala, juntamente com uma pontuação, serão retornadas. A visualização dos resultados é feita através da exibição dos contornos do modelo sobrepostos na imagem na posição encontrada [32, 33].

3.3.1.1 Modelo de Forma Simples

Um modelo de forma simples permite identificar objetos que apresentem rotações e translações relativamente ao modelo de forma criado.

Este modelo permite ajustar vários parâmetros. Um deles é o número de níveis da pirâmide. Este deve ser o maior possível de modo a reduzir o tempo para a localização do objeto. Consequentemente, o número de pontos do modelo será menor, podendo levar a deteções menos exatas, devendo ser conjugado um valor de modo construir-se um modelo preciso e rápido. Para modelos particularmente grandes, poderá ser útil a redução de pontos do modelo, através da configuração do valor de otimização [32].

Outro parâmetro é o intervalo de ângulos e o comprimento do passo dos ângulos. Quanto maiores forem estes níveis, maior será o tempo de execução. O comprimento do passo deverá ser tanto maior quanto maior for o intervalo de ângulos, traduzindo-se num melhor rácio entre tempo e desempenho [32].

Relativamente ao contraste, este pode determinar o contraste dos pontos que o modelo deverá ter, visto que, o contraste mede as diferenças entre os locais de valor de cinzento entre o objeto e o *background*. Este deverá ser escolhido de forma a que torne o modelo mais vantajoso [32].

Por último, é escolhido o parâmetro das métricas que determina as condições sob as quais o modelo é reconhecido na imagem, exemplificado na Figura 10. Existem quatro opções:

- "Use_polarity" - o objeto e o modelo devem ter o mesmo contraste, por exemplo, deteções de um objetos brilhantes sobre um *background* escuro;
- "Ignore_global_polarity" - o objeto também será encontrado na imagem se o contraste da imagem se reverter globalmente, por exemplo, um objeto é detetado quer seja mais escuro que o *background* ou mais claro;
- "Ignore_local_polarity" - deteta o modelo mesmo que o contraste, para um mesmo objeto, altere localmente, ou seja, faz as deteções dos dois primeiros pontos e também deteta um objeto que, por exemplo, esteja colocado num fundo mais claro e escuro, simultaneamente. Porém, o tempo de execução aumenta significativamente;
- "Ignore_color_polarity" - o modelo é encontrado mesmo que o contraste mude de cor localmente, como por exemplo, se partes do objeto mudarem de cor. Esta particularidade é útil para imagens com múltiplos canais, porque, se a imagem for de canal único, este método terá o mesmo efeito que "ignore_local_polarity" [32].

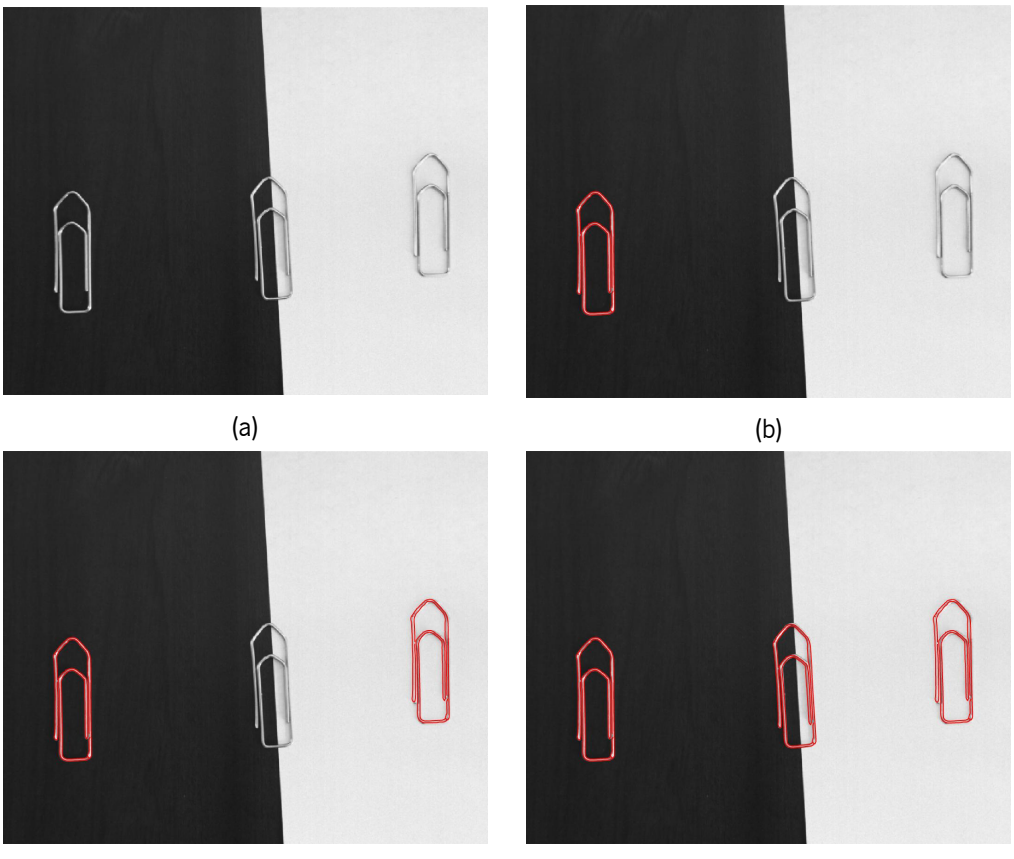


Figura 9: Reconhecimento de um modelo para diferentes parâmetros: (a) imagem inicial; (b) deteção com "use_polarity"; (c) deteção com "ignore_global_polarity; (d) deteção com "ignore_local_polarity".

Na identificação da correspondência do modelo de forma, o operador encontra as melhores instâncias de um modelo na imagem de entrada, retornando as suas localizações, os ângulos de rotação e as

pontuações[1] [32, 33].

Assim como na criação do modelo de forma, também na sua procura é possível definir o intervalo de ângulos, sendo aconselhável utilizar o mesmo intervalo, quer na criação, quer na procura. Estes determinam o intervalo de rotações no qual o modelo será pesquisado [32].

Um parâmetro muito importante é a pontuação mínima, pois é esta que estabelece um limite sobre quais as instâncias que devem ser consideradas. Só serão retornadas aquelas cuja pontuação seja superior à pontuação mínima. Quanto maior for o valor mínimo, menores serão os resultado obtidos, contribuindo para uma pesquisa mais categórica e diminuindo o tempo de pesquisa. Também é possível fixar o número de correspondências que se pretende determinar. Caso sejam encontradas mais instâncias do que o número máximo definido, somente as melhores serão retornadas [32].

O modelo formado poderá ser simétrico, como por exemplo, ter um formato circular, podendo resultar na sobreposição de resultados para um mesmo objeto. A sobreposição máxima determina por qual fração duas instâncias podem sobrepor-se de modo a serem consideradas instâncias diferentes. Num caso de ser superior, apenas a correspondência com maior pontuação é retornada. O cálculo da sobreposição é baseado no menor retângulo anexo da orientação arbitrária das instâncias encontradas [32].

A determinação das instâncias varia com a precisão que se pretenda a nível do *subpixel*. Este parâmetro pode tomar os seguintes valores:

- "None" - o modelo será determinado apenas com precisão de *pixel* e a resolução do ângulo definida na criação do modelo;
- "Interpolation" - a posição e a rotação são determinadas com precisão de *subpixel*. Este modo quase não gasta nenhum tempo de computação e atinge altos níveis de precisão para a maioria das aplicações;
- "Least_squares", "Least_squares_high", "Least_squares_very_high" - o modelo pode ser determinado através de um ajuste dos mínimos quadrados, minimizando as distâncias dos pontos do modelo aos pontos de imagem correspondentes, elevando a precisão e, consequentemente, o tempo computacional. Para este técnica existem três modos, variando distância mínima entre eles;
- "Max_deformation 0", "Max_deformation 1", ..., "Max_deformation 32" - permite encontrar objetos deformados em relação ao modelo, sendo a deformação especificada em *pixels*, podendo ir de um valor inteiro entre 0 e 32. Quanto maior for a deformação, maior será o tempo de execução e maior será a probabilidade de encontrar instâncias erradas. O cálculo da pontuação vai depender do valor escolhido para a extração do *subpixel* [32].

Para finalizar, a operação de procura do modelo, é preciso definir os níveis de pirâmide. Por norma são utilizados os mesmo níveis que os níveis da criação do modelo. Também é possível definir o intervalo onde se pretende que se ocorra a pesquisa, tendo como consequência uma menor precisão dos parâmetros extraídos, mas um algoritmo mais rápido. Normalmente as instâncias são encontradas no último nível

[1] Do inglês: *scores*

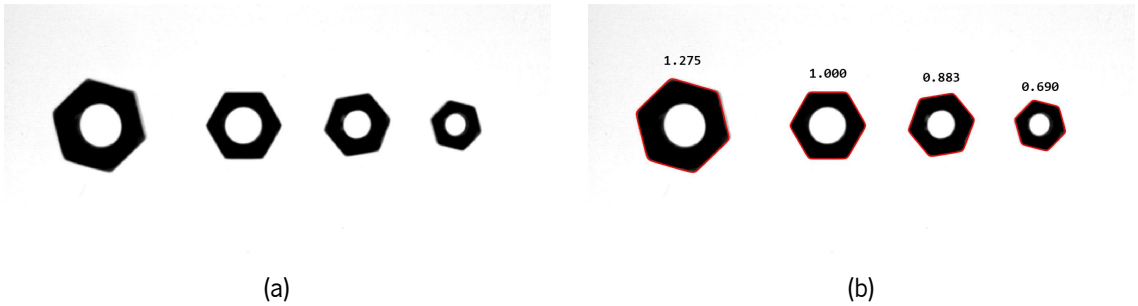
da pirâmide, porém, para imagens de baixa qualidade (desfocadas, deformadas ou com ruído) não é possível encontrar instâncias do modelo neste nível, pois as informações das bordas estão ausentes ou deformadas. A definição de novos níveis, nomeadamente níveis superiores, veio colmatar este problema da qualidade de imagem [32].

3.3.1.2 Modelo de Forma com Escala

Um modelo de forma com escala permite identificar objetos que apresentem rotações, translações e escalamento relativamente ao modelo. A criação do modelo de forma com escala é semelhante ao modelo de forma sem escala, seguindo exatamente os mesmos métodos, mas o modelo de forma criado é isotropicamente dimensionado. A utilização de um modelo de forma com escala isotrópica para correspondência permite que seja possível detetar objetos com diferentes dimensões [32].

Em relação ao modelo de forma sem escala, apenas são acrescentados os parâmetros que determinarão o intervalo de possíveis escalas do modelo. Para isso, são definidos os limites do intervalo e o valor com que o intervalo variará. Este último deverá ser definido de acordo com o tamanho do objeto, visto que quanto maior for este passo, maior será o tempo de computação [32].

Relativamente à pesquisa do modelo na imagem, nesta operação também é necessário definir o intervalo da escala que se pretende identificar, sendo que os dados retornados por esta são equivalente aos do modelo sem escala, adicionando as escalas das instâncias encontradas do modelo. Estes valores são retornados entre os valores mínimo e máximo previamente definidos, sendo que uma escala de 1 corresponderá ao tamanho original do modelo [32]. Na Figura 10 demonstra a funcionalidade da utilização do modelo de forma com escala em objetos com diferentes dimensões.



(a) (b)

Figura 10: Aplicação do modelo de forma com escala: (a) imagem inicial; (b) resultados da identificação dos objetos e os respetivos valores da escala.

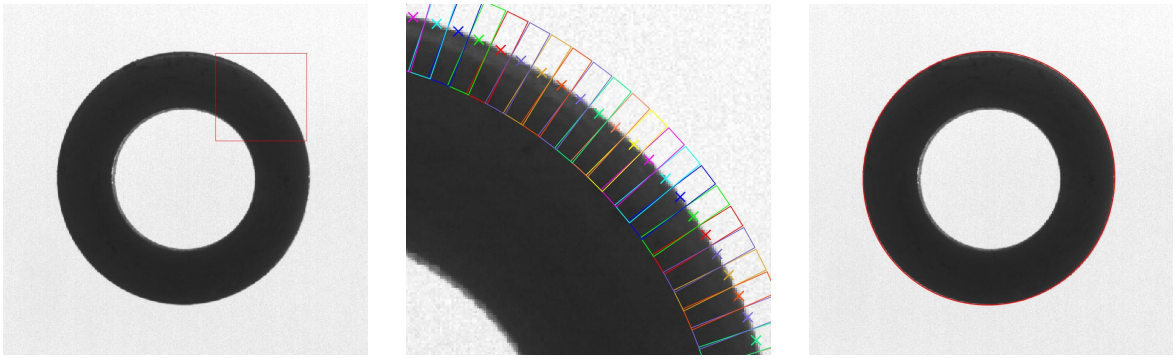
3.3.2 Modelo de Metrologia

A metrologia é um conjunto de técnicas que permite fazer medições de objetos geométricos com base em imagens. O modelo de metrologia[2] consiste numa estrutura de dados necessária para medir objetos com uma forma geométrica específica através de técnicas de metrologia. O modelo de metrologia é usado para armazenar informações como as formas geométricas dos objetos de metrologia e os resultados das

[2] Do *Halcon*: *metrology_model*

medições. Inicialmente começa-se por criar a estrutura de dados, que irá criar um modelo de metrologia em duas dimensões, ou seja, será um modelo capaz de medir objetos com formas geométricas específicas, retornando um identificador desse mesmo modelo. As medições de objetos pode ser em dimensões muito reduzidas, por exemplo, na ordem dos nanómetros, portanto é importante que as medições sejam muito eficientes. Para isso é adicionada as dimensões da imagem à metrologia criada [32, 34].

Uma vez que, o modelo de metrologia criado baseia-se em formas geométricas, é necessário informar qual a forma a ser pesquisada, assim como a informação relativa às dimensões dessa forma. Dentro das opções podem ser escolhidas os formatos: círculo, elipse, linha e retângulo. Poderá também optar-se pelo modo genérico, onde o formato será um parâmetro, aplicando-se a casos onde se deseje medir objetos com diferentes formas. Como, na maioria dos casos, se presente procurar mais do que um objeto, estes irão apresentar pontos de coordenadas centrais diferentes, mantendo a linha e coluna ambos a 0, para permitir que o modelo de metrologia se ajuste aos vários objetos. Contudo, é necessário definir os parâmetros das formas geométricas pelo qual o modelo deverá procurar, como por exemplo, raio para o círculo ou comprimento, largura e ângulo para o retângulo. Depois são ajustadas duas medidas de comprimento, uma é a metade do comprimento das regiões de medida perpendiculares ao limite, outra é a metade do comprimento das regiões de medida tangenciais ao limite [32]. Estas medidas estabelecem o comprimento e largura dos diferentes retângulos presentes na Figura 11b, porque é dentro delas que serão localizados os pontos de limite do objeto (Figura 11c). No final, são estes pontos que serão usados para extrair o contorno final do objeto, retornando o ponto central do objeto e os parâmetros necessários para definir o contorno [32].



(a) (b) (c)

Figura 11: Extração do contorno de um objeto através do modelo de metrologia: (a) imagem inicial; (b) parte da imagem (a) onde se visualiza as tolerâncias que o modelo de metrologia pode ter; (c) imagem (a) com o contorno final detetado.

No casos onde a procura de objetos é superior a um, a metrologia necessita de ser alinhada para diferentes possibilidades. Portanto é realizado o seu alinhamento, que consiste na translação e rotação do modelo de metrologia, em relação ao eixo de coordenadas da image, garantindo que o modelo seja adaptado aos objetos. Após a sua orientação, é então aplicado o modelo de metrologia de modo a efetuar a medição dos objetos de acordo com a sua forma geométrica. Assim são localizadas as arestas dentro das regiões de medida dos objetos, ajustando as formas geométricas correspondentes às posições de aresta resultantes. Por fim, através de outra operação, são obtidos os parâmetros resultantes da aplicação do

modelo de metrologia nos objetos pretendidos [32].

3.3.3 Ajuste do Contorno XLD do Círculo

XLD é a abreviação de *extended line description* e compreende todos os dados baseados em contornos e polígonos. O ajuste de contornos XLD pode ser utilizado para obter parâmetros caraterísticos de polígonos existentes nas imagens, nomeadamente, para identificar o centro geométrico de um círculo.

No caso do ajuste do contorno XLD do círculo[3], este operador considera os contornos XLD e aproxima-os por círculos, retornando o centro do círculo, o seu raio e o ângulo de início e fim do contorno, para casos em que o contorno não seja totalmente fechado. É importante referir que este procedimento não faz segmentação do contorno e é preciso garantir que cada contorno corresponda a um e apenas um círculo [32, 34].

Este algoritmo usado para o ajuste de círculos pode ser selecionado via:

- "Algebraic" - minimiza a distância algébrica entre os pontos do contorno e o círculo resultante;
- "Ahuber" - semelhante ao "algebraic", porém os pontos do contorno são ponderados para diminuir o impacto dos valores discrepantes com base na abordagem de Huber;
- "Atukey" - semelhante ao "algebraic", mas os pontos do contorno e os *outliers* ignorados com base na abordagem de Tukey;
- "Geometric" - minimiza a distância geométrica entre os pontos do contorno e o círculo resultante. É uma opção que leva mais tempo de computação, porém, a medida da distância é estatisticamente ideal, sendo ainda a mais aconselhável para contornos que estão distorcidos por ruído;
- "Geohuber" - semelhante ao "geometric", contudo os pontos do contorno são ponderados para diminuir o impacto dos valores discrepantes com base na abordagem da Huber;
- "Geotukey" - semelhante ao "geometric", todavia os pontos do contorno são ponderados e os *outliers* são ignorados com base na abordagem de Tukey [32].

As iterações da operação são apenas utilizadas para os algoritmos "algebraic", "ahuber" e "atukey" e ignorado pelos "geometric", "geohuber" e "geotukey". Estas servem para otimizar o círculo identificado, contribuindo para um maior tempo de execução. Um modo de jogar com esta consequência é diminuindo o número máximo de pontos. Se algum valor diferente de -1 for atribuído, apenas serão utilizados o número de pontos estabelecidos, distribuídos uniformemente. Para se determinar um círculo, é necessário um número mínimo de três pontos do contorno [32].

É possível detetar círculos e também arcos circulares. Para a segunda opção, o procedimento escolhe os pontos iniciais e finais dos contornos originais, sendo depois retorna-os. Além disso, deverá definir-se a distância máxima entre os pontos finais de um contorno, de modo a que este seja considerado fechado [32].

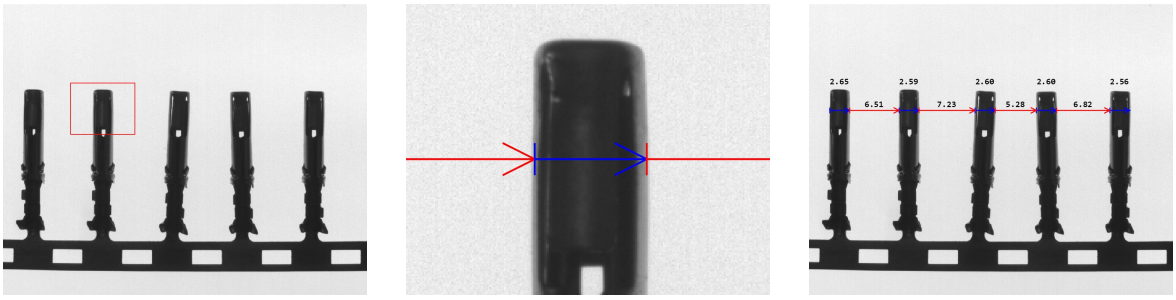
[3] Do *Halcon*: *fit_circle_contour_xld*

3.3.4 Medição de Pares

A medição de pares[4] permite extrair pares de retas perpendiculares ao eixo principal de um objeto e retorna as posições dos centros desses pares de retas, as suas amplitudes, as distâncias entre os pares de retas e as distâncias entre pares consecutivos de retas [32, 35].

Antes de ser aplicado esta operação, é necessário desenhar um arco que preparará a extração de retas perpendiculares, retornando uma estrutura de dados otimizada, também designada de metrologia. Depois, é preciso introduzir no procedimento o tipo de reta que se pretende investigar, ou seja, se for pretendido pontos das reta com uma transição claro-escuro-clara, uma transição claro-escuro-escuro ou todas as transições, optando por escolher "positive", "negative" ou "all", respetivamente. Se mais de uma consecutiva com a mesma transição for encontrada, a primeira será usada como par da segunda. Em alguns casos, este comportamento pode causar problemas, dado que o *threshold* não é alto o suficiente para suprimir retas consecutivas da mesma transição. Para estes casos, existem as opções "positive_strongest" e "negative_strongest" que apenas detetam as retas mais fortes de uma sequência de retas consecutivas de subida e descida [32].

No final, serão retornados oito listas contendo: as linhas e colunas dos centros das primeiras e segundas retas (pares respetivos), as amplitudes das bordas das primeiras e segundas bordas (ambas com sinal) e as distâncias entre os pares de retas e os pares de retas consecutivos. Alguns destes resultados são demonstrados na Figura 12. Os pontos iniciais das setas e os pontos finais das setas correspondem às coordenadas das retas detetadas, ou seja, ao início e fim dos objetos e ao início e fim do espaço entre objetos (Figura 12b), uma vez que o procedimento foi preparado para identificar todas as transições. Os valores em cima das setas equivalem aos comprimentos das setas (Figura 12b), cuja função é indicar as distâncias de cada um dos objetos e as distâncias entre eles [32].



(a) (b) (c)

Figura 12: Extração de pares de retas perpendiculares: (a) imagem inicial; (b) parte imagem (a) para melhor visualização dos resultados; (c) resultados da aplicação da operação medição de pares e respetivas distâncias calculadas.

3.4 XML

Um ficheiro *script Halcon* é armazenado no formato XML. Isto torna-se numa vantagem, dado que permite a leitura do ficheiro sem a necessidade de recorrer às bibliotecas do *Halcon*, possibilitando a

[4] Do *Halcon*: *measure_pairs*

recolha de informação vital para o desenvolvimento da aplicação, nomeadamente, o tipo de procedimentos que vão ser utilizados e o tipo de variáveis de entrada e de saída, icónicas e de controlo. A sua estrutura é apresentada na Figura 13.

```
<?xml version="1.0" encoding="UTF-8"?>
<hdevelop file_version="1.2" halcon_version="18.11.1.1">
<procedure name="orientation">
<interface>
<io>
<par name="Image" base_type="iconic" dimension="0"/>
</io>
<oo>
<par name="Regions" base_type="iconic" dimension="0"/>
<par name="ConnectedRegions" base_type="iconic" dimension="0"/>
<par name="SelectedRegions" base_type="iconic" dimension="0"/>
</oo>
<ic>
<par name="Minimum" base_type="ctrl" dimension="0"/>
<par name="Maximum" base_type="ctrl" dimension="0"/>
</ic>
<oc>
<par name="Phi" base_type="ctrl" dimension="0"/>
</oc>
</interface>
<body>
<l>threshold (Image, Regions, Minimum, Maximum)</l>
<l>connection (Regions, ConnectedRegions)</l>
<l>select_shape (ConnectedRegions, SelectedRegions, 'area', 'and', 5500, 7100)</l>
<l>orientation_region (SelectedRegions, Phi)</l>
<l>return ()</l>
</body>
<docu id="orientation">
<parameters>
<parameter id="ConnectedRegions"/>
<parameter id="Image"/>
<parameter id="Maximum"/>
<parameter id="Minimum"/>
<parameter id="Phi"/>
<parameter id="Regions"/>
<parameter id="SelectedRegions"/>
</parameters>
</docu>
</procedure>
</hdevelop>
```

Figura 13: Resultado de um ficheiro de procedimento aberto num editor de texto.

De forma a contribuir para este trabalho, ao ler o ficheiro de procedimento como um documento XML, este apresenta dois conjuntos principais de informação, nomeadamente, o grupo "<interface>" e o grupo "<body>". No primeiro grupo são representadas todas as variáveis pelo seu nome, tipo e dimensão. Estas são separadas pelo tipo de variável que representam, icónicas ("<_o>") ou controlo ("<_c>"), e se são variáveis de entrada ("<i_>") ou de saída ("<o_>"). Estas variáveis foram assim definidas no ficheiro de procedimento no *HDevelop*. No outro grupo, surge o corpo do ficheiro, onde cada linha representa uma função e as suas variáveis associadas.

3.5 Sumário

O *Halcon* é um *software* abrangente para a visão por computador, com uma grande variedade de funções para programar e idealizar as soluções ideias para sistemas de IVA. Permitindo assim uma redução dos custos e uma diminuição de tempo das linhas de manufaturação. Relativamente à sua arquitetura, esta permite aceder a estruturas de dados definidas e, assim, integrar o *Halcon* com outros componentes

de *software*. Este *software* oferece ainda várias interfaces para aceder aos seus operadores a partir de diferentes linguagens de programação, tais como, *C*, *C++*, *C#* e *Visual Basic*.

As principais ferramentas que o *Halcon* oferece são o *HDevelop* e o *HDevEngine*. A sua junção permite que os pedidos de mudança do cliente sejam mais rápidos e práticos, resultando numa linha de produção mais rapidamente ajustável para lidar com novos objetos.

Os modelos de forma são modelos usados para correspondência que identificam objetos similares ao objeto utilizado como referência, através de vários níveis de pirâmides da imagem. Com a metrologia, o utilizador pode retirar as dimensões dos objetos, desde que estes possam ser representados por formas geométricas específicas, tais como, círculos, elipses, retângulos e linhas. Por sua vez, quando as regiões são idênticas a círculos, os dados destes podem ser extraídos através do ajuste do contorno XLD do círculo. Na medição de pares, são extraídos os pares de retas perpendiculares a um retângulo ou arco anular, com base nas transições.

Capítulo 4

Toolbox de Soluções

No mercado industrial, o elevado padrão e índice de uniformidade são condições obrigatórias, por isso a concentração por parte dos trabalhadores é fundamental, principalmente no que toca à inspeção visual de produtos [6, 36]. Os sistemas de IVA são a combinação entre a experiência humana e a elevada velocidade de processamento dos computadores. Com estes, pretende-se que a análise de imagens seja o mais assertiva possível e no menor espaço de tempo, contribuindo assim para uma maior qualidade do produto [6].

Os sistemas de IVA podem ser usados para inspecionar a qualidade de todos os passos do processo de manufaturação. Este tipo de sistemas traz vantagens económicas visto que quanto mais cedo for detetada a falha, a probabilidade de um produto ser rejeitado é diminuída. Além disso, durante a produção, pode existir variações de características ambientais, tais como, a intensidade da iluminação que pode levar a falhas do sistemas de IVA, requerendo o seu reajuste para as novas condições. A indústria atual requer que a inspeção visual de tarefas seja cada vez mais precisa e rápida, possibilitando uma maior qualidade de produtos. Na atualidade, as soluções comerciais vigentes apenas apresentam implementações primitivas elementares de métodos de visão por computador. Uma vez que, a inspeção de objetos podem apresentar problemas semelhantes, por exemplo, diferir na alteração da cor ou na posição de um objeto, passa a ser desnecessário o desenvolvimento de novos algoritmos de cada vez que é apresentado um novo produto. A resolução para estes obstáculos passa pela criação de uma *toolbox* com *templates* de soluções completas parametrizáveis para sistemas de IVA. A *toolbox* de soluções pode ser rapidamente personalizada para novos problemas.

No presente capítulo é descrita a importância do desenvolvimento de uma *toolbox* de soluções, assim como uma definição dos problemas que serão abordados. De seguida, serão caraterizadas as soluções desenvolvidas para cada um dos problemas e, para cada uma delas, é explicado o tipo de sequenciação sucedido. Por fim, são apresentados e discutidos os resultados obtidos das diferentes soluções propostas para a *toolbox*.

4.1 Registo da Peça

O registo da peça é um passo comum a todas as soluções implementadas, sendo o primeiro procedimento de todas elas. Este é o passo mais importante para uma correta execução e análise de uma solução. Durante o processo de manufaturação, os materiais não são sempre colocados na mesma posição, podendo existir ligeiras alterações do posicionamento da peça que, por exemplo, podem alterar as localizações dos objetos a detetar. Porém, também não seria benéfico se o utilizador tivesse de desenhar novas ROIs por cada nova peça que surgisse.

Este procedimento recorre ao modelo de forma (detalhado na Secção 3.3.1.2) para obter o registo de uma peça. O modelo criado é baseado no formato da peça de referência e apenas traz vantagens enquanto que as peças de teste são iguais à peça de referência. O seu objetivo é reconhecer o registo da peça de referência nas peças de teste e alinhar a imagem de teste de modo a que ambos os registos sejam o mais coincidentes possíveis. Assim, o utilizador apenas precisa de definir uma vez as ROIs, na peça de referência, servindo de padrão para as novas imagens de teste.

Para esse fim, o utilizador deverá definir uma região que abranja a peça na globalidade, pois será apenas essa região que irá ser processada. A ROI desenhada pode tomar diferentes formatos, como por exemplo, um círculo, uma elipse, um retângulo paralelo ao eixo, um retângulo com qualquer orientação ou uma região livre, sendo aconselhado a utilizar o formato mais semelhante à forma do objeto, diminuindo a área de procura e, por conseguinte, o tempo de computação. É importante referir que para todas as ROIs utilizadas ao longo deste trabalho se seguiu este processo.

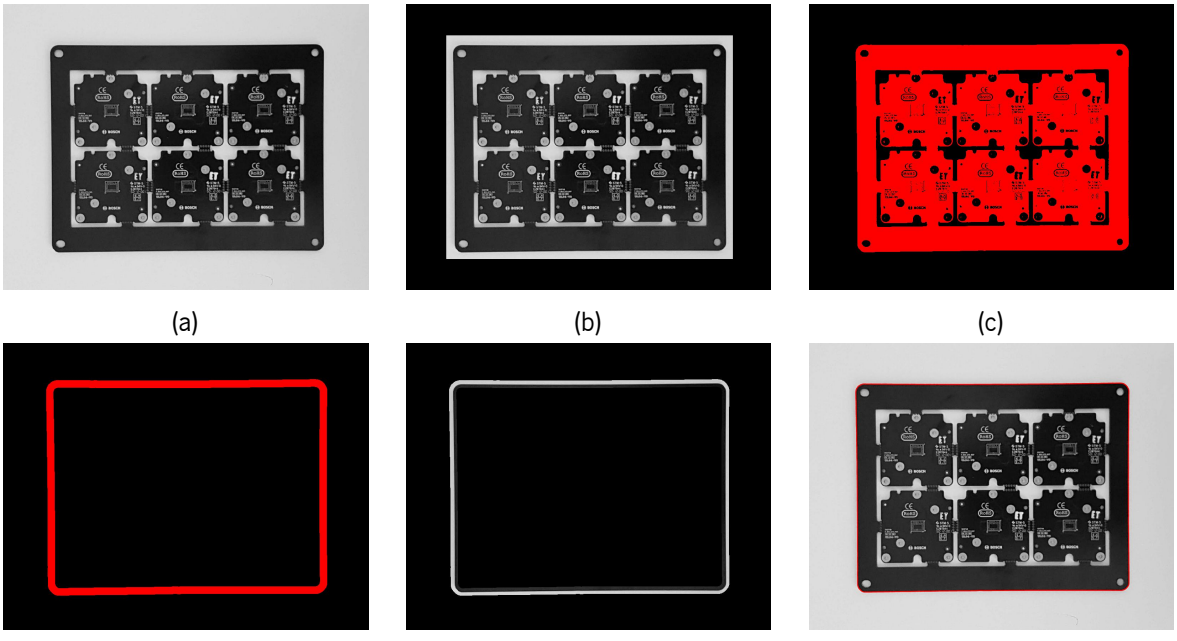


Figura 14: Passos executados para uma deteção de borda: (a) imagem inicial; (b) *threshold* da imagem (a); (c) abertura da região (b); (d) diferença das regiões resultantes da dilatação e da erosão; (e) redução da imagem (a) para a região (d); (f) imagem (a) com o contorno da borda.

Seguidamente, é aplicado um *threshold* (Figura 14c) e uma abertura com um elemento estruturante circular de raio 3.5, de forma a apenas restringir a peça e eliminar possíveis pontos isolados existentes

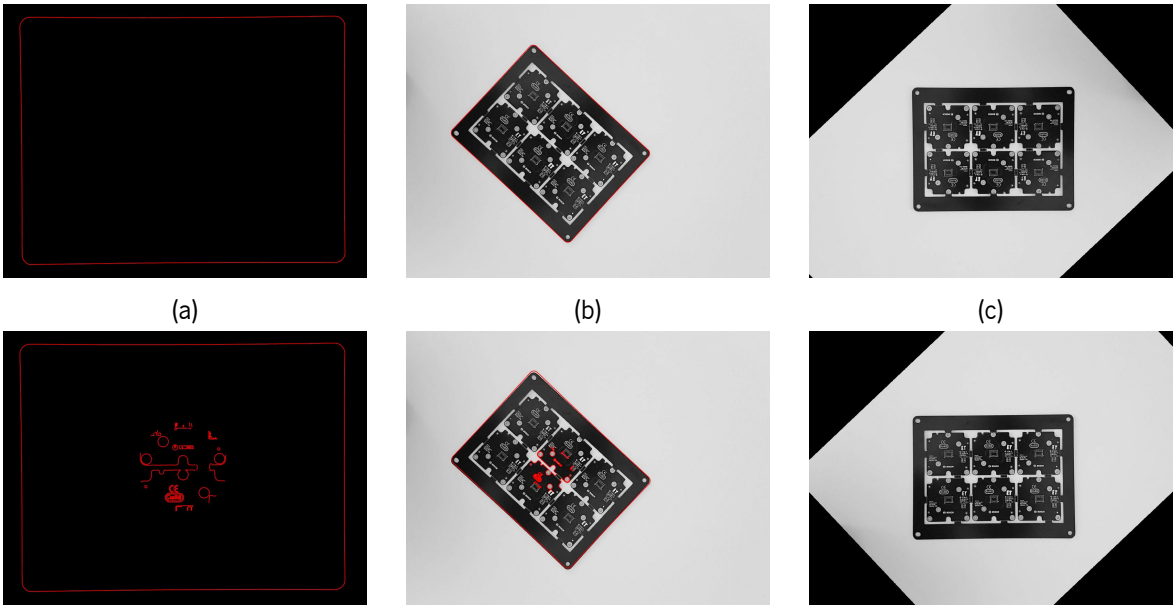
na imagem, respetivamente, para não interferirem aquando a criação do modelo de forma da peça. A seguir, aplica-se um preenchimento para preencher todo o interior da região que irá corresponder à peça a ser inspecionado. Depois, é efetuada uma dilatação e uma erosão, ambas na região resultante do preenchimento, e calculada a diferença entre as suas áreas resultantes (Figura 14d), com o objetivo de obter a região que inclui contorno da peça. O novo domínio de imagem é calculado como a interseção da região anterior (Figura 14d) com a imagem inicial (Figura 14a) originando uma redução da imagem inicial (Figura 14e). Depois, o modelo de forma será criado baseado nesta imagem reduzida, isto porque caso o modelo se baseasse na peça total, apesar ter um melhor reconhecimento das peças nas imagens de teste e ser menos suscetível a falhas, o seu tempo de computação seria significativamente maior, não havendo assim um equilíbrio entre a performance e o tempo de execução do algoritmo. A Figura 14 ilustra as etapas mais importantes da criação do registo da peça, equivalente à deteção do seu contorno.

Ao aplicar um modelo de forma, além de este definir na imagem o registo da peça, também retorna a posição e rotação do registo efetuado na imagem. Dessa maneira, o modelo de forma criado é ainda aplicado na imagem de referência com o intuito de extrair a posição e a rotação do registo da peça. Por fim, o modelo e os valores do posicionamento da peça são guardados para, posteriormente, servirem como diretrizes para as novas peças em estudo.

No ponto de vista dos novos testes, inicialmente é feita a leitura e aplicação do modelo de forma na nova imagem da qual resultará a localização da peça, as suas coordenadas e ângulo de rotação, e a sua pontuação, isto é, o grau de certeza na deteção de uma peça que obedece ao modelo. Através destes parâmetros, e dos pontos da peça de referência, é possível calcular uma matriz de transformação, que consiste numa translação e rotação. A matriz de transformação é fundamental, dado que as ROIs criadas na peça de referência são fixas, pois fará as alterações necessárias na imagem de teste, para que o registo da peça de teste coincida com o registo da peça de referência. Consequentemente, a imagem transformada apresentará as áreas de interesse nas ROIs definidas, na imagem de referência. Uma vez que, todo o processamento e resultados estão em conformidade com a imagem transformada, é necessário alterá-los para a imagem inicial, antes da transformação auxiliada pelo modelo de forma. Para isso, é calculada a matriz inversa, que consiste na translação e rotação inversa da matriz de transformação. Esta permitirá alinhar os resultados obtidos na imagem transformada para a imagem inicial. Esta operação será necessária para quando estão a ser analisadas novas peças, ou seja, dado que existe uma transformação da imagem para a deteção de objetos, depois também será preciso a transformação inversa dos resultados das deteções para a imagem de teste original.

Contudo, por vezes o limite das peças pode ser simétrico (Figura 14a), o que poderá fazer com que o reconhecimento da peça seja igual, quer para uma posição, quer para uma rotação de 180º. A Figura 15 pretende mostrar a diferença do alinhamento de uma imagem com recurso a diferentes modelos de forma, quando aplicados à mesma imagem. Aqui são apresentados os registos de forma de dois modelos (Figuras 15a e 15d), baseados na imagem da Figura 14a, que serviu de referência. Depois do cálculo da matriz e respetiva transformação da imagem, foram obtidas as imagens apresentadas nas Figuras 15c e 15f. Apesar do primeiro modelo ter feito o reconhecimento da peça com sucesso (Figura 15b), a transformação correspondente não foi efetuada corretamente (Figura 15c), uma vez que a peça está rodada 180º comparativamente à peça de referência, podendo apresentar os objetos a detetar fora das

ROIs definidas. Para colmatar esta falha, após a definição da região do contorno da peça, é incluído uma nova região circular, no ponto central da peça, com um raio ajustável (Figura 15d). Esta adição contribui para um modelo de forma mais completo e evita que ocorra um reconhecimento da peça erradamente, quando aplicado a uma nova imagem de teste. Na Figura 15f, a transformação da imagem pelo modelo ilustrado na Figura 15d é equivalente à peça de referência, mostrando assim a importância da adição de uma outra região para o regista de forma das peças e o respetivo reconhecimento nas peças de teste.



(a) (b) (c)

(d) (e) (f)

Figura 15: Diferença da aplicação de diferentes modelos de forma. (a) Modelo de forma sem região circular; (b) contorno identificado pelo modelo (a); (c) transformação da imagem (b) pelo modelo (a); (d) modelo de forma com região circular de raio 200; (e) contorno identificado pelo modelo (d); (f) transformação da imagem (e) pelo modelo (d).

4.2 Definição das Soluções

Cada solução é implementada de acordo com o melhor um conjuntos de passos para a identificação de objetos correspondentes aos problemas em questão. Cada uma das soluções implementadas serão explicadas de seguida.

4.2.1 Deteção de Diferentes Marcadores Fiduciais

Na área industrial, o marcador fiducial, ou fiducial, consiste numa marcação que serve como referência para o reconhecimento de padrões do circuito, permitindo identificar a posição de uma placa. Por exemplo, um marcador fiducial circular, geralmente, é constituído por uma máscara aberta de solda e revestido com cobre no seu centro, de diâmetro inferior, aumentando o seu contraste e a facilitando o seu reconhecimento [37]. Porém, existem várias formatos de fiduciais. Neste trabalho apenas foram abordados dois tipos, círculo e cruz. Ambas as deteções seguem o mesmo raciocínio, apenas diferenciando no

método utilizado para a sua identificação. Todo este processo é apresentado na Figura 16, nos dois tipos de fiduciais estudados.

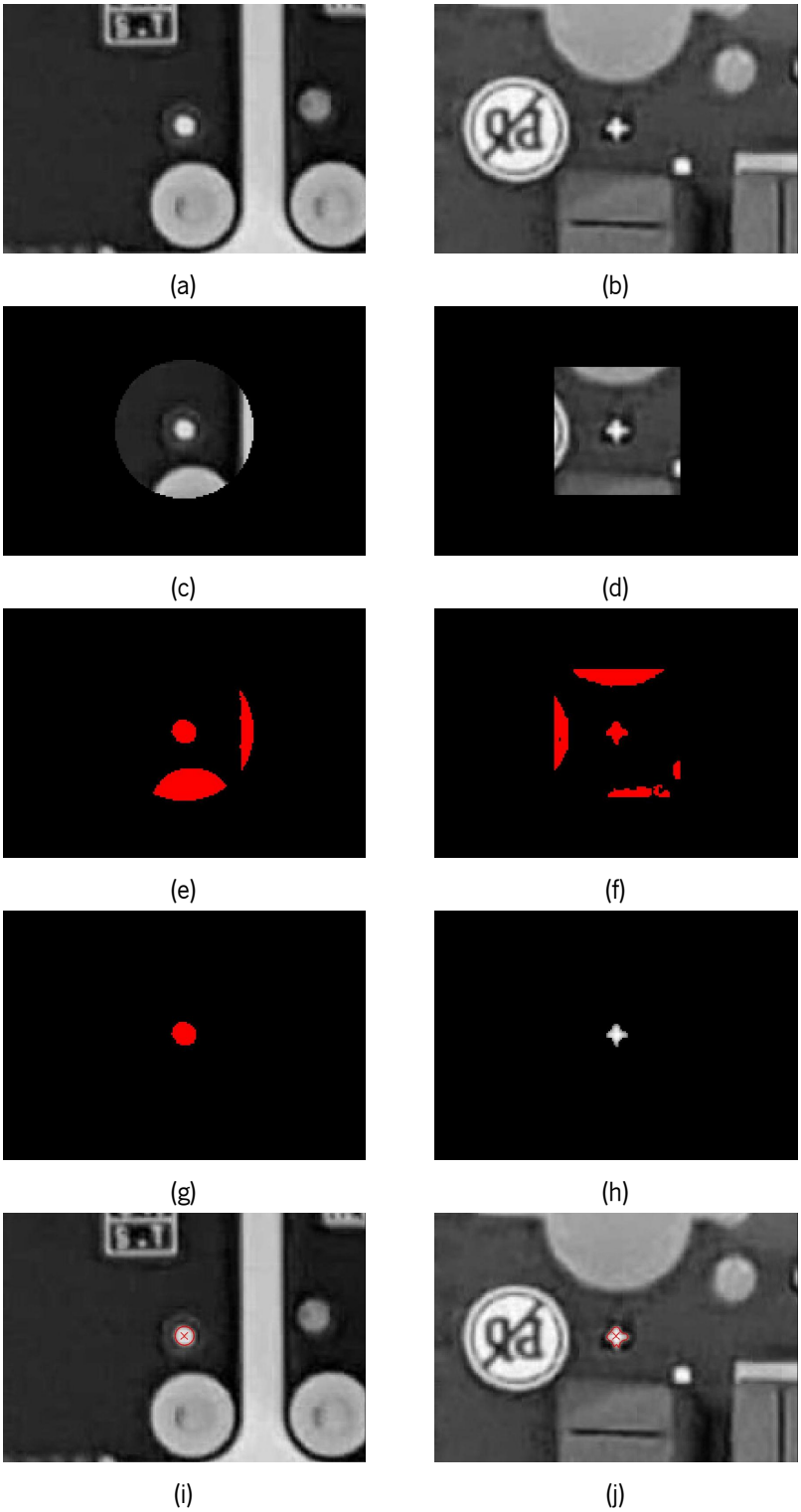


Figura 16: Dois exemplos das diferentes etapas da deteção de um fiducial: (a) imagem inicial para o fiducial circular; (b) imagem inicial para o fiducial em cruz; (c) seleção da ROI da imagem (a); (d) seleção da ROI da imagem (b); (e) *threshold* da imagem (c); (f) *threshold* da imagem (d); (g) seleção de caraterísticas da região (e); (h) interseção da imagem (b) com a região resultante da seleção de caraterísticas da região (f); (i) resultado da aplicação do modelo de metrologia; (j) resultado da aplicação do modelo de forma, sobreposto na imagem (b).

Primeiramente, para o fiducial circular, é gerado um modelo de metrologia que consiste numa estrutura para medir formas geométricas. Depois, é necessário que o modelo de metrologia seja ajustado para as mesmas dimensões da imagem, de modo a diminuir o tempo de procura quando aplicado a uma imagem. Posteriormente, é adicionada informação relativamente aos parâmetros de pesquisa da metrologia, nomeadamente o raio do fiducial e o valor de *threshold* a ser aplicado. Por outro lado, o fiducial em cruz, é detetado recorrendo ao modelo de forma, que irá identificar as melhores correspondências deste modelo numa imagem. Para este problema, apenas é preciso desenhar a ROI para a forma desejada, ou seja, neste caso, a ROI deverá ser o mais pequena possível e apenas deverá conter o fiducial, sendo criado um novo modelo de forma para a parte da imagem selecionada. Este será o fiducial padrão para todos os outros, dado que todos os fiduciais numa placa apresentam todos as mesmas caraterísticas. Em ambos os casos, a informação sobre os parâmetros de entrada é guardada para, futuramente, serem lidos e aplicados a todos os fiduciais existentes nas PCBs de teste.

Depois, é efetuada uma ROI geral que irá englobar todos as áreas que se pretendem investigar, criando uma área por cada fiducial (Figuras 16c e 16d). Ao mesmo tempo, é contabilizado o número de regiões criadas de forma a indicar o algoritmo quantas vezes será preciso averiguar e aplicar o modelo criado, que varia com o tipo de fiducial que se esteja a inspecionar. Nesta parte, apenas as ROIs irão ser processadas, contribuindo para a eficiência e diminuição do tempo de execução do programa. Aqui também é aplicado um *threshold* (Figuras 16e e 16f), de modo a que contenha pelo menos as regiões coincidentes com os fiduciais. Posteriormente, a região derivada do *threshold* é individualizada, ou seja, são determinados os componentes conectados das regiões para diferenciar cada uma das regiões, caso contrário o *Halcon* considera que apenas existe uma região. Esta individualização de regiões procede-se sempre antes da aplicação da seleção de caraterísticas, ao longo de todo o trabalho. As regiões resultantes são depois aplicadas na função de seleção de caraterísticas. A função de seleção de caraterísticas consiste na extração de caraterísticas de regiões, especificadas num determinado intervalo. Os seus parâmetros caraterísticos, valor mínimo e máximo, são variáveis ajustáveis pelo utilizador (Figuras 16e e 16g).

Para o fiducial circular, é calculado o centro de cada uma das regiões extraídas na etapa anterior, pois será nesta região e neste ponto de coordenadas que será aplicado o modelo de metrologia. Após a sua aplicação são retiradas três propriedades, a posição no eixo das ordenadas, a posição no eixo das abcissas e o raio do círculo, resultando na identificação de um fiducial (Figura 16i). Esta técnica é aplicada a cada uma das áreas, resultando na deteção de todos os fiduciais. Em contrapartida, para o fiducial em cruz, uma vez que a pesquisa é através do modelo de forma, a entrada terá de ser uma imagem, logo é feita a redução de domínio (redução da imagem) através da a imagem inicial e das regiões selecionadas anteriormente resultando na Figura 16h. O modelo aplicado irá restringir a sua busca a um limite máximo equivalente ao número total de ROIs criadas e apenas serão mantidas as deteções com maior pontuação. Como resultado final, este fornecerá as coordenadas do centro da deteção, o ângulo comparativamente ao modelo inicial e a pontuação do reconhecimento do fiducial.

Para finalizar, de modo a visualizar os resultados obtidos, foi gerada uma imagem vazia, com as mesmas dimensões da imagem de teste, e, através do ponto de coordenadas do centro das identificações, foi desenhada uma cruz para auxiliar a perceção destas deteções.

4.2.2 Deteção de Pinos Corretamente Posicionados

De todas as soluções desenvolvidas neste trabalho, a deteção de pinos é a solução mais geral, devido aos diferentes pinos existentes e suas possíveis variações. Esta resolução deverá ser capaz de interpretar e avaliar corretamente pinos de diferentes tipos e formatos, como por exemplo, pinos em fichas e pinos de circuitos integrados.

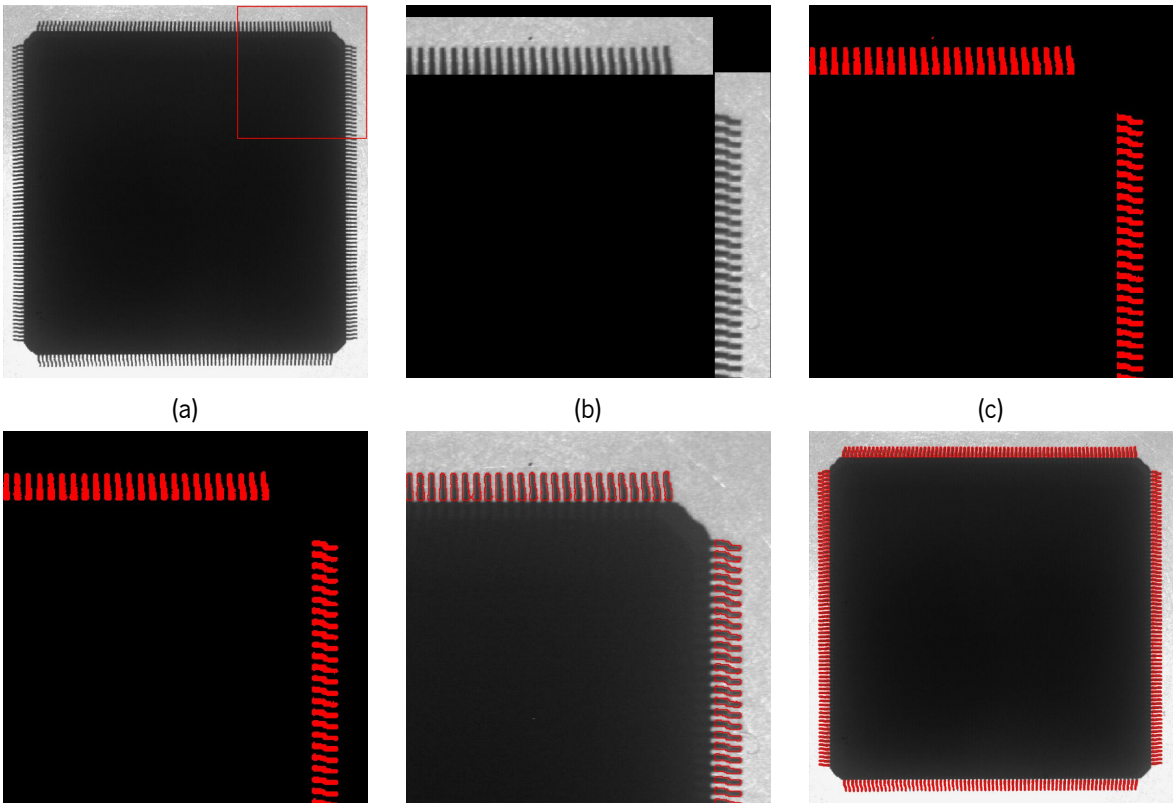


Figura 17: Diferentes passos da identificação de pinos numa dada região: (a) imagem inicial com uma área de interesse definida; (b) definição das ROIs para os pinos; (c) aplicação do *threshold* na imagem (b); (d) região resultante de toda as operações morfológicas; (e) parte da imagem (a) com os resultados obtidos da deteção de pinos; (f) resultados obtidos da deteção de pinos.

Começa-se por selecionar as ROIs (Figura 17b) referentes às zonas com pinos. Posteriormente, visto existirem variações nos níveis de cinzento quer ao longo de cada pino quer entre pinos devido, por exemplo, à iluminação ambiente, serão efetuados cinco processos relevantes, abertura, preenchimento, fecho, preenchimento e novamente abertura. É aplicado um *threshold* (Figura 17b), garantindo que os pinos ficam incluídos. Depois, a primeira abertura (Figura 17b) aplicada à região resultante do *threshold* anterior, serve para eliminar pontos isolados ou pequenas regiões que saem dos limites das regiões. Depois, é seguido de um preenchimento que tem o objetivo selar as áreas que contenham buracos no seu interior, compactando a identificação, isto acontece porque para uma mesma imagem, pinos em diferentes posições podem apresentar diferentes tons de cinzento ou, devido às sombras dos objetos, um pino pode apresentar diferentes tons, no entanto, como apenas é definido um valor de *threshold* para toda a imagem, é preciso existir um equilíbrio na escolha do valor de *threshold*. Posteriormente, o fecho tem a funcionalidade de fechar pequenos buracos e conetar componentes, é utilizado para reforçar a estrutura

de cada pino, porque na maioria dos casos os pinos apresentação pequenas dimensões e diferentes tons de cinzento ao longo do seu objeto, logo as regiões derivadas do *threshold*, para um mesmo pino, podem não estar todas conectadas. É novamente aplicado o preenchimento para consolidar ainda mais as regiões dos pinos, preenchendo alguma buraco interno existente nas regiões. Para finalizar estes passos, aplica-se outra abertura para auxiliar na suavização dos contornos e separação de possíveis uniões entre regiões. Tanto as aberturas como o fecho fazem uso de um elemento estruturante circular, cujo raio é uma entrada configurável, com a finalidade de poder ser ajustado a outros problemas semelhantes.

Para concluir, nem sempre as regiões adquiridas até este processo representam só pinos. Partindo do princípio que, para uma mesma peça, os pinos vão apresentar as mesmas caraterísticas, é feita uma seleção baseada na forma das regiões anteriormente obtidas. A escolha das regiões é com ajuda de recursos de forma, tais como, área, rácio, orientação. A seleção deve ser feita de acordo com os recursos mais favoráveis para a identificação dos pinos (Figura 17d). Esta seleção pode ser adaptável para os diferentes casos em estudo. Deste procedimento resultam as regiões correspondentes aos pinos que, depois serão reduzidas para seus limites, de modo a facilitar a visualização dos resultados obtidos (Figura 17e). Este método proposto encontra-se ilustrado na Figura 17, onde é representada a identificação de pinos de um componente eletrónico.

4.2.3 Deteção do Centro Geométrico e Verificação do Posicionamento de Ponteiros

Neste trabalho, a deteção do centro geométrico e posicionamento de ponteiros está associado aos conta-rotações, porém pode servir de solução para casos similares. Esta resolução está dividida em duas partes. A primeira serve para calcular o ponto central da peça para inserir um ponteiro nessa coordenada, ou seja, esta primeira análise será efetuada sem ponteiro. A segunda consiste na verificação do posicionamento do ponteiro e se este está corretamente direcionado, neste caso, para a posição 0.

Inicialmente, é desenha uma ROI e aplicado um *threshold* (Figura 18b), ambos definidos pelo utilizador, de modo a englobar todo o objeto. Depois, toda a região formada pelo *threshold* vai ser preenchida, com o intuito de que a região abranja todo o objeto (Figura 18c). Seguidamente, será efetuada uma abertura que, como explicado anteriormente, tem a função de suavizar os limites do contorno e também eliminar possíveis pontos isolados existentes na região. O principal objetivo desta abertura será polir toda a região, pois quanto mais a região se aproximar de um círculo (Figura 18d), melhor será a deteção do seu centro, sendo que esta abertura também se baseia num elemento estruturante circular configurável. Depois de extraída a região resultante da abertura, é calculada a borda desta região (Figura 18e), isto porque, mesmo com raios elevados, essa região pode apresentar ligeiras deformações o que vai influenciar a correspondência do ponto central, caso se queira determinar diretamente o centro desta região. Por essa razão, recorre-se a uma função do *Halcon* que permite aproximar os contornos *extended line description* (XLD) (na qual todos os dados são baseados em contornos e polígonos) por círculos (descrito na Secção 3.3.3). Deste método resultará, entre outras saídas, o centro geométrico do objeto. Esta coordenada indicará o local onde deverá ser posicionado o ponteiro. Esta metodologia está apresentada na Figura 18.

Adicionalmente, também é identificada a barra correspondente ao 0, método ilustrado na Figura 19.

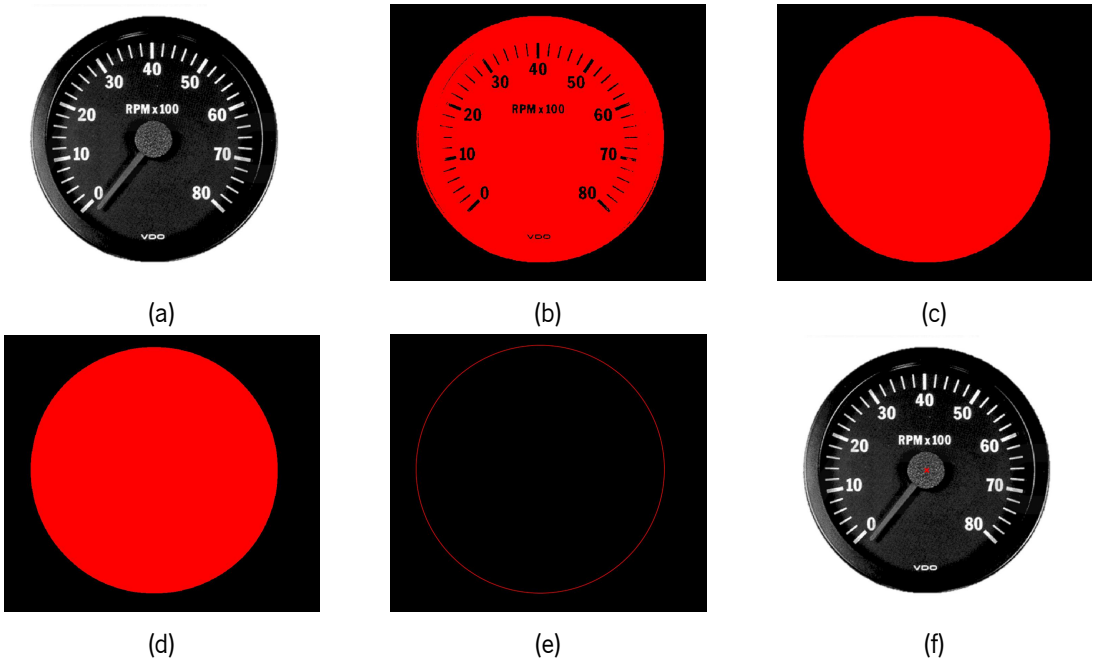
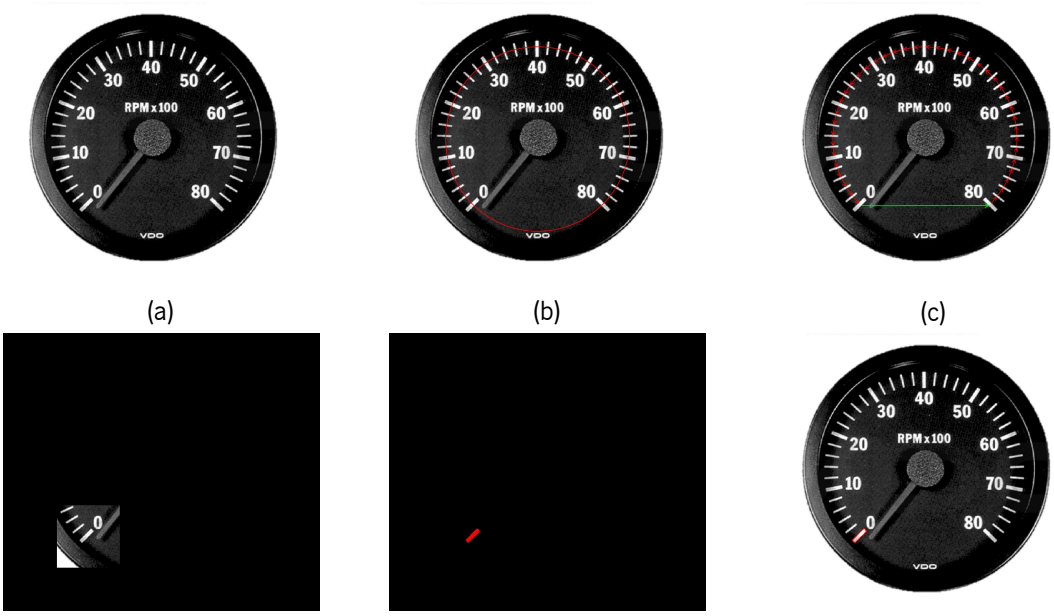


Figura 18: Diferentes fases da localização do centro geométrico de um objeto: (a) imagem inicial; (b) *threshold* da imagem (a); (c) preenchimento da região (b); (d) abertura da região (c); (e) contorno da região (d); (f) deteção do centro.

Começa-se por definir uma ROI que intercete as barras principais (associadas aos números) do conta-rotações. Esta definição origina uma região, pelo que esta precisa de ser processada modo a obter-se apenas o seu raio e as suas posições no eixo das ordenadas e no eixo das abcissas (Figura 19b). Isto é necessário porque, com a informação do ponto central e do raio, o programa elabora um arco circular preparado para extrair arestas perpendiculares aos locais onde foi definido, originando uma metrologia para medição de distâncias entre deteções. Depois, esta estrutura será uma entrada do operador responsável por determinar as medições de pares[1] (descrito na Secção 3.3.4). Isto consiste na extração do conjunto de todos os pontos correspondentes ao cruzamento do arco anular com as arestas perpendiculares detetadas, equivalentes aos pontos de coordenadas de início e fim de cada barra, principal e intermédia. A distância entre duas barras consecutivas será relativamente semelhante, todavia existe um caso em que a distância é relativamente superior a todas as outras (Figura 19c), que corresponde à distância entre a barra do 0 e a barra da rotação máxima. Posto isto, com a disposição, por ordem crescente, da lista com as distâncias entre barras é possível retirar o índice do valor máximo obtido, pois este corresponderá ao ponto de coordenadas final da barra do 0. Depois, será criado um quadrado, cujo seu ponto central será igual ao ponto anteriormente detetado e lado igual 100, de forma a envolver a barra do 0 (Figura 19d). É, então, aplicado um *threshold* e o *select shape* para determinar a barra do 0 (Figura 19e). Por último, é calculada a orientação da área determinada. Dado que, tanto a região como a orientação determinados são referências para o posicionamento do ponteiro, serão automaticamente guardados, para futura comparação com o ponteiro.

Relativamente à segunda parte, associada à identificação do ponteiro, primeiramente é aplicado um *threshold* (Figura 20b), seguido de uma abertura (Figura 20c), uma seleção da região correspondente ao

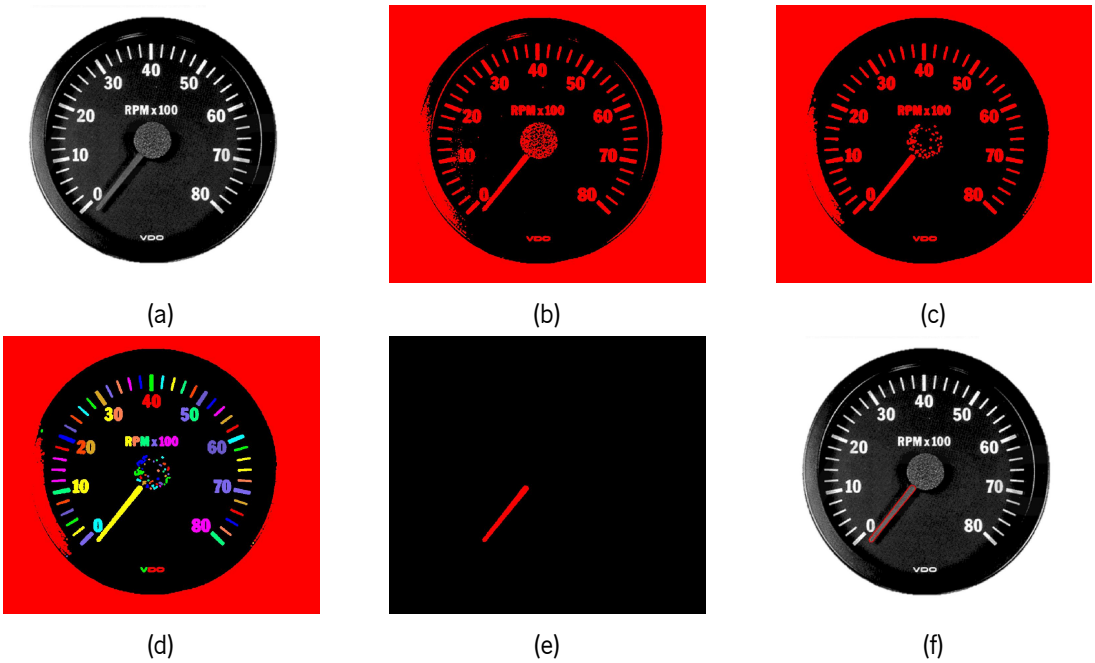
[1] Do *Halcon*: *measure_pairs*



(a) (b) (c)

(d) (e) (f)

Figura 19: Diferentes etapas localização de uma barra específica: (a) imagem inicial; (b) delineamento da ROI desenhada; (c) identificação da distância entre duas barras consecutivas, sendo a maior delas representado noutra cor; (d) redução da imagem (a) para a barra detetada (c); (e) caraterística selecionada após o *threshold* da imagem (d); (f) contorno da barra.



(a) (b) (c)

(d) (e) (f)

Figura 20: Passos executados para a deteção de um ponteiro: (a) imagem inicial; (b) *threshold* aplicado na imagem (a); (c) abertura da região (b); (d) individualização da região (c); (e) caraterísticas específicas selecionadas da região (e); (f) contorno do ponteiro detetado.

ponteiro e um preenchimento (Figura 20e) para, deste modo, identificar a presença do ponteiro na peça. Depois o algoritmo baseia-se no mesmo procedimento de deteção efetuado para barra do 0, ou seja, para a região determinada, equivalente ao ponteiro, é calculada a sua orientação, para posteriormente ser comparado com a orientação da barra do 0. No final, é determinado o seu contorno XLD, para efeito

de melhor visualização (Figura 20f).

4.2.4 Deteção da Ausência ou Posicionamento Defeituoso de Parafusos

Assim como fiduciais, na indústria existem vários tipos de parafusos. Estes podem apresentar uma série de cabeças diferentes e, dentro da cabeça do parafuso, podem apresentar fendas diferentes. O método proposto permite detetar a ausência ou defeito da posição de parafusos, como apresenta o exemplo da Figura 21.

(a) (b) (c)

Figura 21: Diferentes passos para a deteção do parafuso: (a) imagem inicial; (b) ROI; (c) localização do parafuso, após aplicação do modelo de forma com escala.

Esta solução segue um princípio muito semelhante à deteção da borda, sobre a deteção de objetos. É selecionada uma pequena ROI e realizada a interceção da região resultante com a imagem inicial de modo a apenas obtermos parte da imagem, referente ao parafuso que se pretende examinar. Depois é criado um modelo de forma com escala (detalhado na Secção 3.3.1.2). Este procedimento têm exatamente as mesmas caraterísticas que o modelo de forma só que também retorna uma escala. Esta opção permite que o modelo se ajuste, mudando o seu rácio, de forma a obter uma melhor correspondência. Esta operação é utilizada porque, para um determinado objeto, quanto maior for a sua aproximação da câmara, maior será a sua área. Logo, recorrendo a uma referência, se o parafuso da imagem de teste tiver um rácio ligeiramente superior a 1, quer dizer que o posicionamento deste parafuso está incorreto.

Seguidamente, é novamente desenhada uma ROI, ou várias, ligeiramente maior que a anterior, pois será esta que servirá de zona crítica para as imagens de estudo. Também aqui é feita a contagem do número de regiões desenhadas, definindo assim um limite máximo, para o programa saber quantas deteções devem ser efetuadas. Por fim, existe a redução da imagem para as partes da imagem escolhidas, ocorre a aplicação do modelo de forma com escala na imagem de teste e, consequentemente, são obtidos os resultados da identificação do parafuso.

4.3 Resultados e Discussão

A configuração de uma sistema de IVA é um fator extremamente importante, visto que tem grande influência na aquisição de imagens. Estas imagens recolhidas são inseridas na aplicação para o seu

processamento. Por isso, quanto mais adequado ao problema for o *setup*, melhores serão as imagens adquiridas e, consequentemente, mais fácil será o seu processamento.

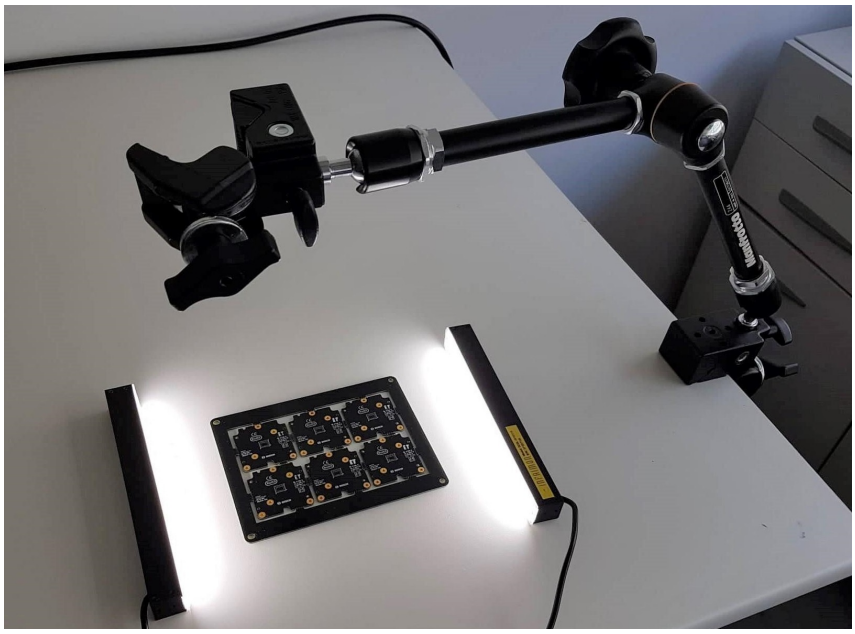


Figura 22: *Setup* utilizado para a aquisição de imagens.

Na Figura 22 está presente o *setup* desenvolvido para a aquisição de imagem. Para tal, esta recolha foi realizada numa superfície branca, aumentando o contraste entre o fundo e peça processada. Usou-se um suporte constituído por um braço articulado e por duas braçadeiras, sendo que estas últimas encontram-se nas pontas do braço articulado, estando suportadas pela superfície e auxiliam na estabilidade e fixação da câmara. O suporte auxilia a aquisição de imagem, tendo como principal objetivo estabilizar a câmara, fixando a sua posição e contribuindo para uma aquisição uniforme. Foram ainda utilizados dois díodos emissores de luz (LEDs[2]), posicionados manualmente. A iluminação é essencial este tipo de aplicações, pois como uma maior quantidade de luz incide no objeto, haverá um maior contraste entre elementos, sendo assim mais fácil distingui-los [38]. Relativamente à aquisição de imagens, foi utilizada uma câmara *dual pixel* de 12 *megapixels* com auto focagem e uma abertura f/1,7.

Na registo de forma de uma peça, a inclusão de uma região central circular no modelo de forma elabora um modelo mais completo, ao mesmo tempo que permite que a identificação de peças seja mais eficiente. A adição desta região leva a um aumento do tempo de execução e, na maioria dos casos, à diminuição do grau de certeza na deteção das peças identificados, isto porque, aumenta a área de procura de correspondência, pelo que, a percentagem do número de pontos coincidentes do registo da peça na imagem de teste com a imagem inicial diminua. Apesar desta ocorrência, a transformação aplicada à imagem é mais assertiva, evitando falsas deteções ou que estas sejam mal concretizadas, como por exemplo, no caso de peças com bordas simétricos. Se a peça de teste tiver uma discrepância elevada comparativamente à imagem de referência, nomeadamente devido a translações elevadas, a pontuação da instância será menor, uma vez que a perspetiva da câmara em relação ao objeto será diferente comparativamente com a peça de referência.

[2]Do inglês: *light-emitting diodes*

Relativamente às soluções, estas foram divididas em quatro casos distintos. No primeiro caso, o problema incide sobre a deteção de marcadores fiduciais. Foram realizadas duas abordagens para os fiduciais, pelo facto de adquirirem diferentes formatos. Para o fiducial circular o maior desafio reside na segmentação dos fiduciais. Estes são fabricados de modo a apresentarem bom contraste para facilitar o seu reconhecimento. No entanto, a avaliação efetuada pelo modelo de metrologia baseia-se em regiões, logo uma boa segmentação dos fiduciais levará a uma melhor deteção dos mesmos. Para o outro formato, o fiducial em cruz, a extração dos fiduciais não é tão rigorosa. Dado que o reconhecimento é através do modelo de forma, as instâncias não precisam de estar completamente na ROI, sendo apenas suficiente o centro do modelo original estar dentro da ROI. Uma vez que este método se baseia em contornos de formas, torna-o muito robusto contra fatores como a iluminação, no caso de se apresentarem imagens com brilho elevado ou muito escuras.

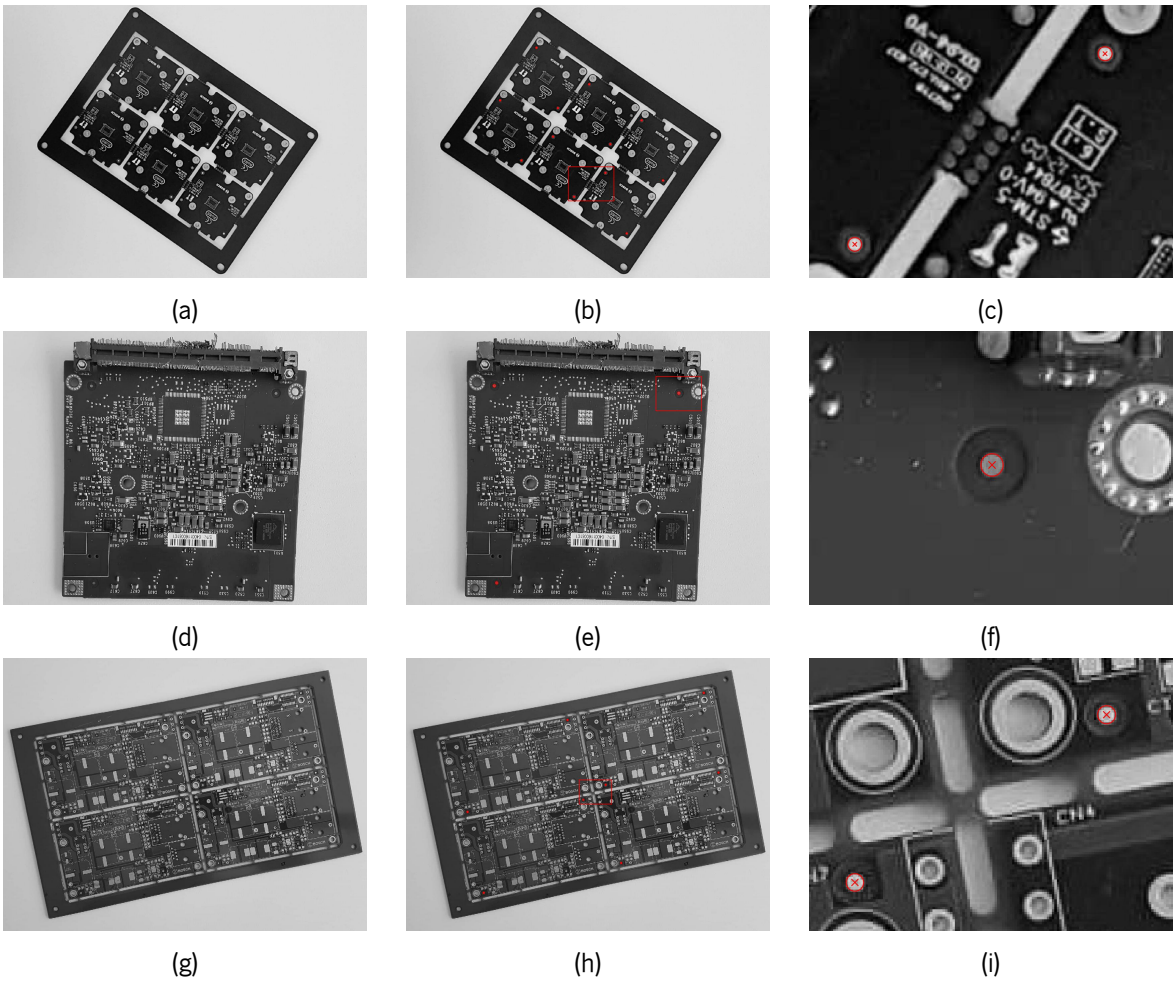
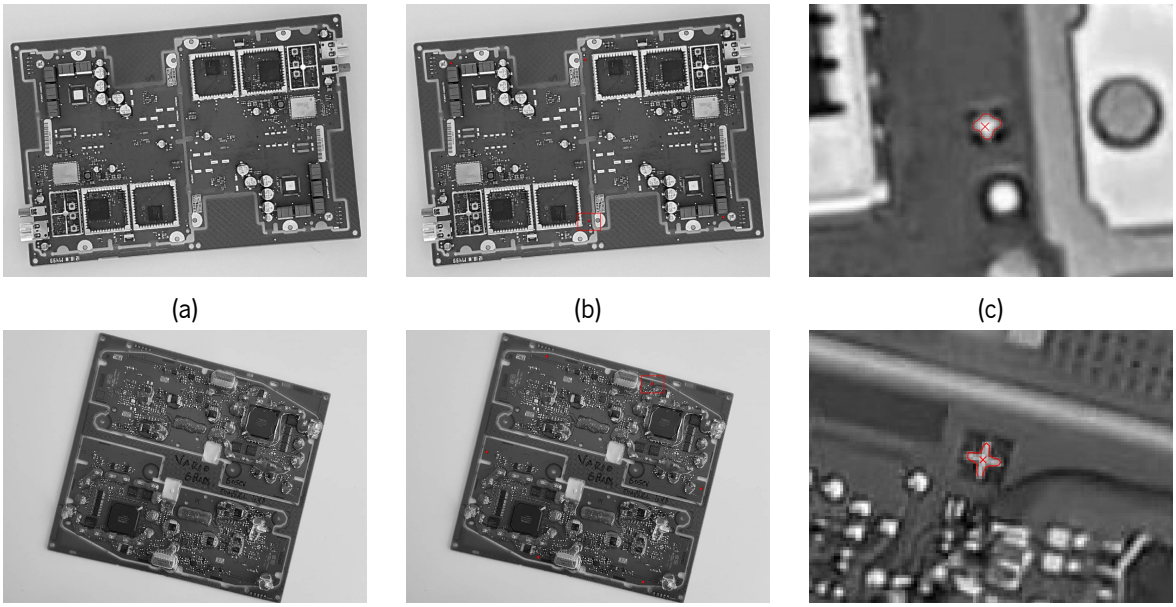


Figura 23: Diferentes casos estudados para a deteção de fiduciais, em forma de círculo: (a), (d) e (g) imagem inicial; (b), (e) e (h) resultado do algoritmo para a deteção de fiduciais; (c), (f) e (i) parte da imagem (b), (e) e (h), respetivamente.

As Figuras 23 e 24 apresentam a capacidade do algoritmo em identificar diferentes fiduciais em diferentes objetos. Sabendo-se o tipo de fiducial a identificar, ao configurar-se novas ROIs e novos parâmetros, o algoritmo é ajustado. Para este trabalho, o algoritmo foi capaz de detetar os fiduciais, independentemente da sua localização na imagem e da localização da peça, desde que devidamente identificado pelo

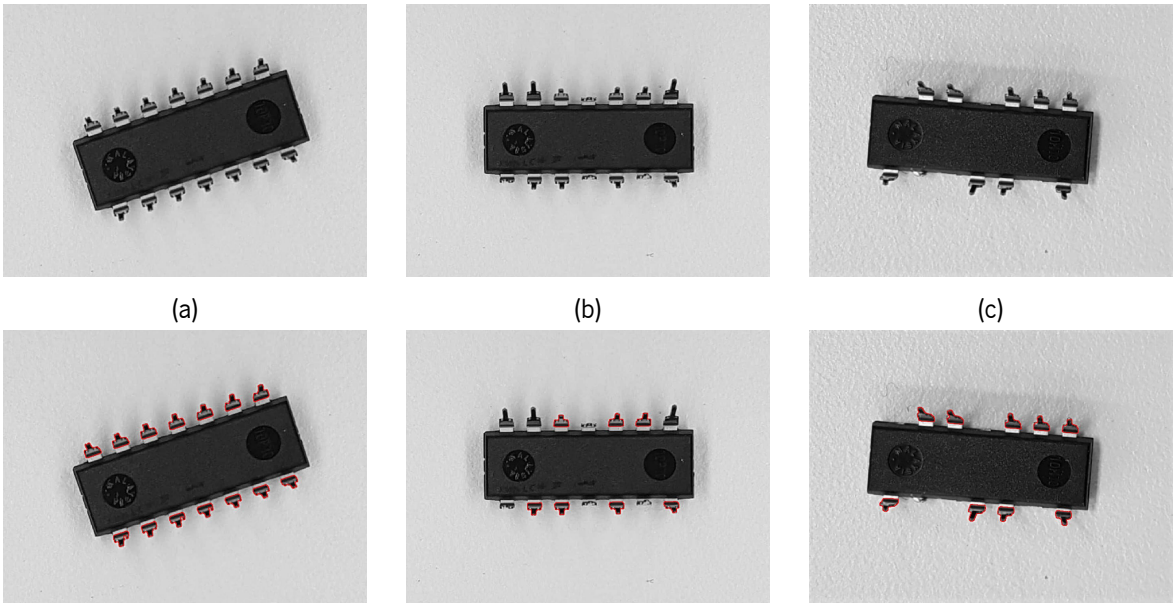
utilizador.



(a) (b) (c)

(d) (e) (f)

Figura 24: Diferentes casos estudados para a deteção de fiduciais, em forma de cruz: (a) e (d) imagem inicial; (b) e (e) resultado do algoritmo para a deteção de fiduciais; (c) e (f) parte da imagem (b) e (e), respetivamente.



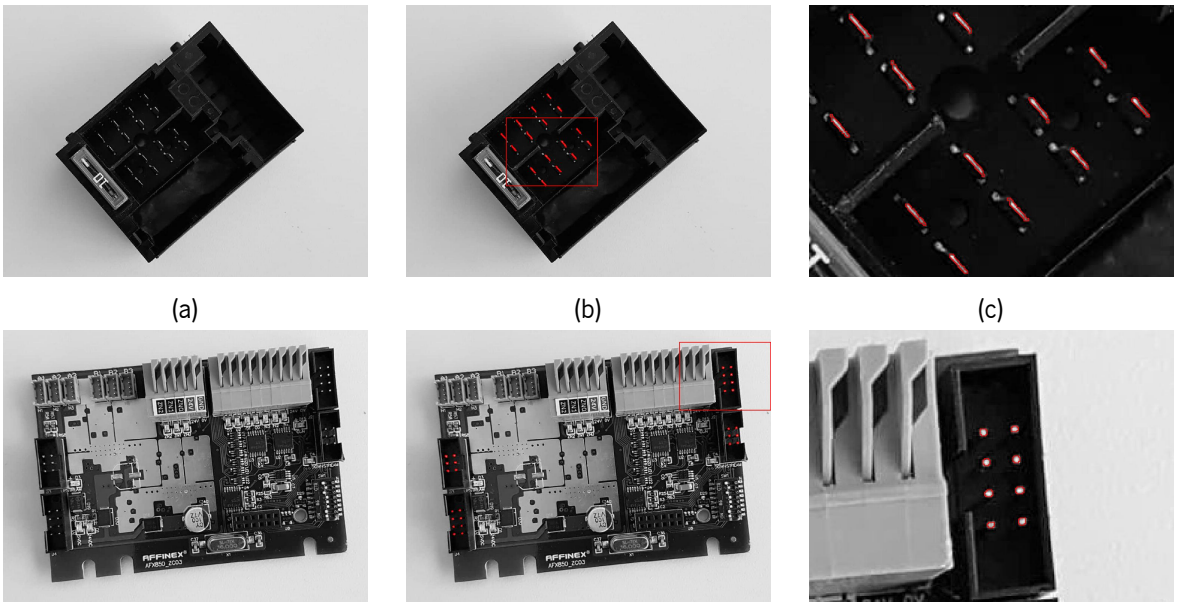
(a) (b) (c)

(d) (e) (f)

Figura 25: Diferentes casos estudados para a deteção de pinos: (a) objeto com todos os pinos corretamente orientados; (b) objeto com alguns pinos dobrados; (c) objeto com alguns pinos ausentes; (d), (e) e (f) resultado do algoritmo para a deteção de pinos.

Para a segunda solução, a identificação de pinos, o algoritmo desenvolvido é capaz de detetar pinos corretamente quando estes se apresentem bem posicionados. Devido à variedade de pinos, a solução baseia-se, essencialmente, na seleção das caraterísticas dos pinos para detetar regiões associados aos

pinos. Parte-se do princípio, que os pinos que estão a ser avaliados têm todos as mesmas caraterísticas, por isso, o utilizador só precisa de escolher as caraterísticas e os intervalos que melhor se adequem à situação. Na Figura 25 observa-se que a deteção acontece para os pinos posicionados corretamente (Figura 25d) e no caso de existirem pinos dobrados (Figura 25e), ou ausentes (Figura 25f), estes não são detetados pelo algoritmo. Além disso, a Figura 26 apresenta outros resultados obtidos noutras peças onde também existem pinos, exemplificando o desempenho do método proposto.



(a) (b) (c)

(d) (e) (f)

Figura 26: Outros casos analisados para a deteção de pinos: (a) e (d) imagem inicial; (b) e (e) resultado do algoritmo para a deteção de pinos; (c) e (f) parte da imagem (b) e (e), respetivamente.

Para terceiro caso, os resultados da solução desenvolvida para a deteção do centro geométrico podem ser divididos em três partes. Estas partes são ilustradas na Figura 27. Na primeira parte, representada nas Figuras 27a, 27b e 27c, observa-se a deteção do ponto central, que apesar de corresponder a um ponto pertencente ao encaixe para o ponteiro, verifica-se que o ponto de deteção não é ótimo. Este ligeiro desvio pode ser explicado pelo facto do objeto não estar totalmente paralelo com a câmara, devido aos apoios irregulares que este apresenta. Na segunda parte, correspondente à deteção da barra do 0, onde a maior dificuldade reside no desenho do arco que incluirá todas as barras a ser detetadas e, posteriormente, calculadas para a deteção da barra pretendida. Este arco deverá englobar todas as barras presentes no objeto para que o algoritmo seja bem sucedido (Figura 27e). A última etapa, é a segmentação do ponteiro (Figuras 27h e 27k) e verificação do seu correto posicionamento. Para o objeto estudado, a solução desenvolvida funcionou para todos os testes realizados, até mesmo em pequenas variações (Figura 27g). Nas Figuras 27i e 27l são apresentados os resultados das segmentações efetuados, a deteção da barra do 0, a contorno verde, e a identificação do ponteiro, a contorno vermelho. Na Figura 27i verifica-se que o posicionamento do ponteiro foi identificado como errado, apesar do ponteiro estar ligeiramente desviado da barra do 0, o algoritmo tem a capacidade de indicar que o ponteiro não se encontra no seu devido lugar. Enquanto que na Figura 27l mostra um exemplo onde o ponteiro está a apontar para o local correto, mostrando assim a performance do algoritmo proposto.

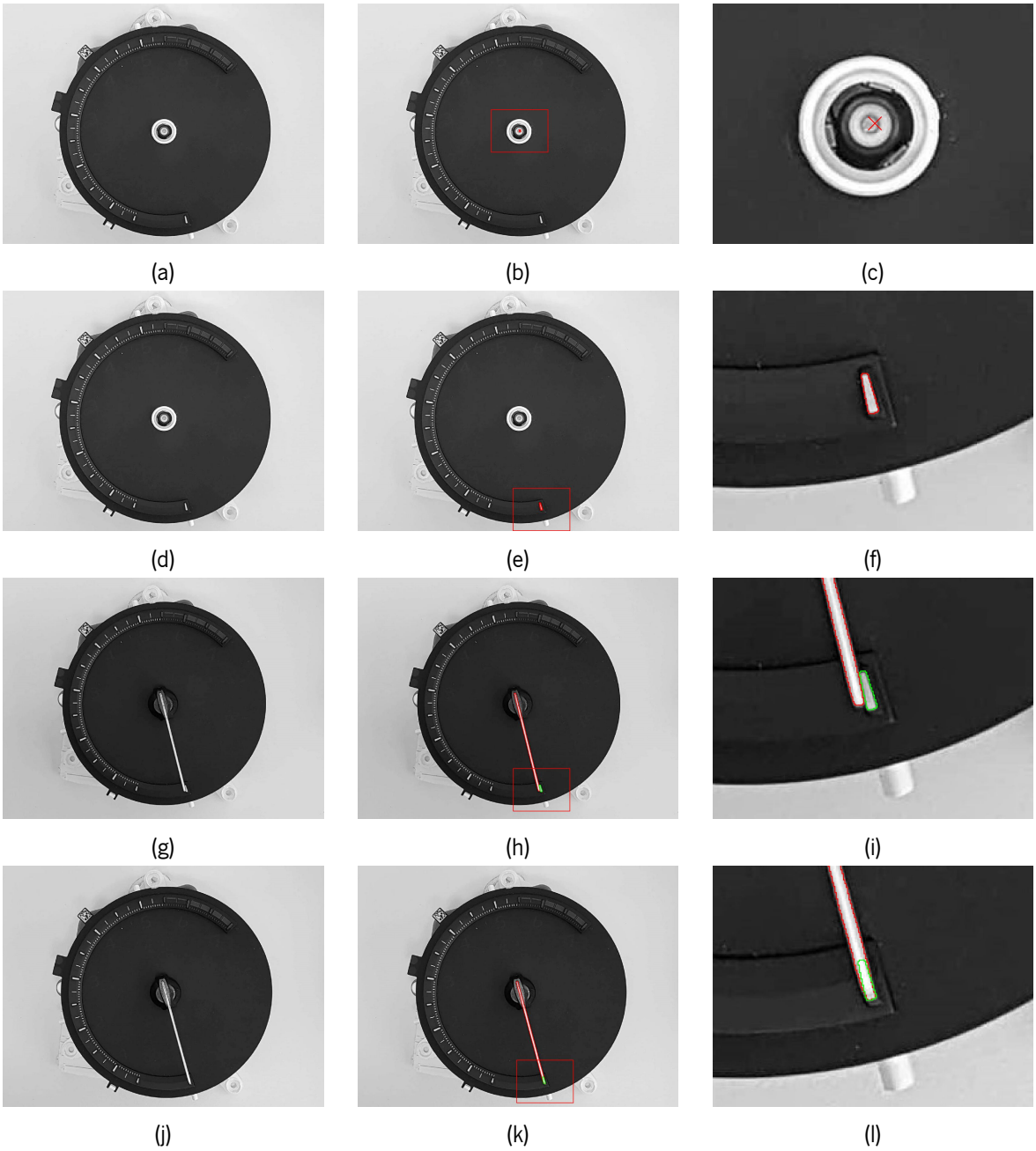


Figura 27: Resultados obtidos após a aplicação do algoritmo para a deteção do ponteiro: (a) imagem inicial sem ponteiro; (b) resultado da deteção do ponto central do objeto; (c) parte da imagem; (d) imagem inicial sem ponteiro; (e) resultado da identificação da barra do 0; (f) parte da imagem (d); (g) imagem inicial com ponteiro; (h) resultado da deteção do ponteiro e comparação com a barra do 0; (i) parte da imagem (h); (g) imagem inicial com ponteiro; (j) resultado da deteção do ponteiro e comparação com a barra do 0; (l) parte da imagem (k).

No quarto e último caso, a verificação da existência ou defeito dos parafusos foi a única solução que não cumpriu os seus objetivos na totalidade. Com este algoritmo obteve-se os resultados presentes na Figura 28. Na maioria dos casos, o algoritmo detetou a presença do parafuso e o seu correto posicionamento, ou seja, se estava bem aparafusado (Figura 28b) ou mal (Figura 28e), baseando-se na pontuação atribuída na sua escala, pelo modelo de forma com escala. Nos casos em que a o valor da escala distancia de 1, significa que o parafuso encontra-se mal apertado e por isso, é considerado que

tem um posicionamento defeituoso. O único resultado onde obteve sempre sucesso foi na verificação da existência de parafuso, isto é, caso não existisse parafuso (Figura 28h) na ROI, o algoritmo informa que não existe nenhum parafuso, visto que o modelo não deteta nenhuma forma na ROI desenhada. Os resultados menos favoráveis são relativos ao correto posicionamento do parafuso, onde o algoritmo, por vezes, deteta que o parafuso está bem posicionado quando este não está (Figura 28j) e vice-versa (Figura 28m). Estas falhas podem ser justificadas pelo posicionamento dos LEDs e também pelo *setup* construído. Como não existia suportes para os LEDs, era complicado manter a luz constante nas zonas de interesse, fazendo com que a luz refletida pelo parafuso variasse à medida que se movia o objeto ou os LEDs.

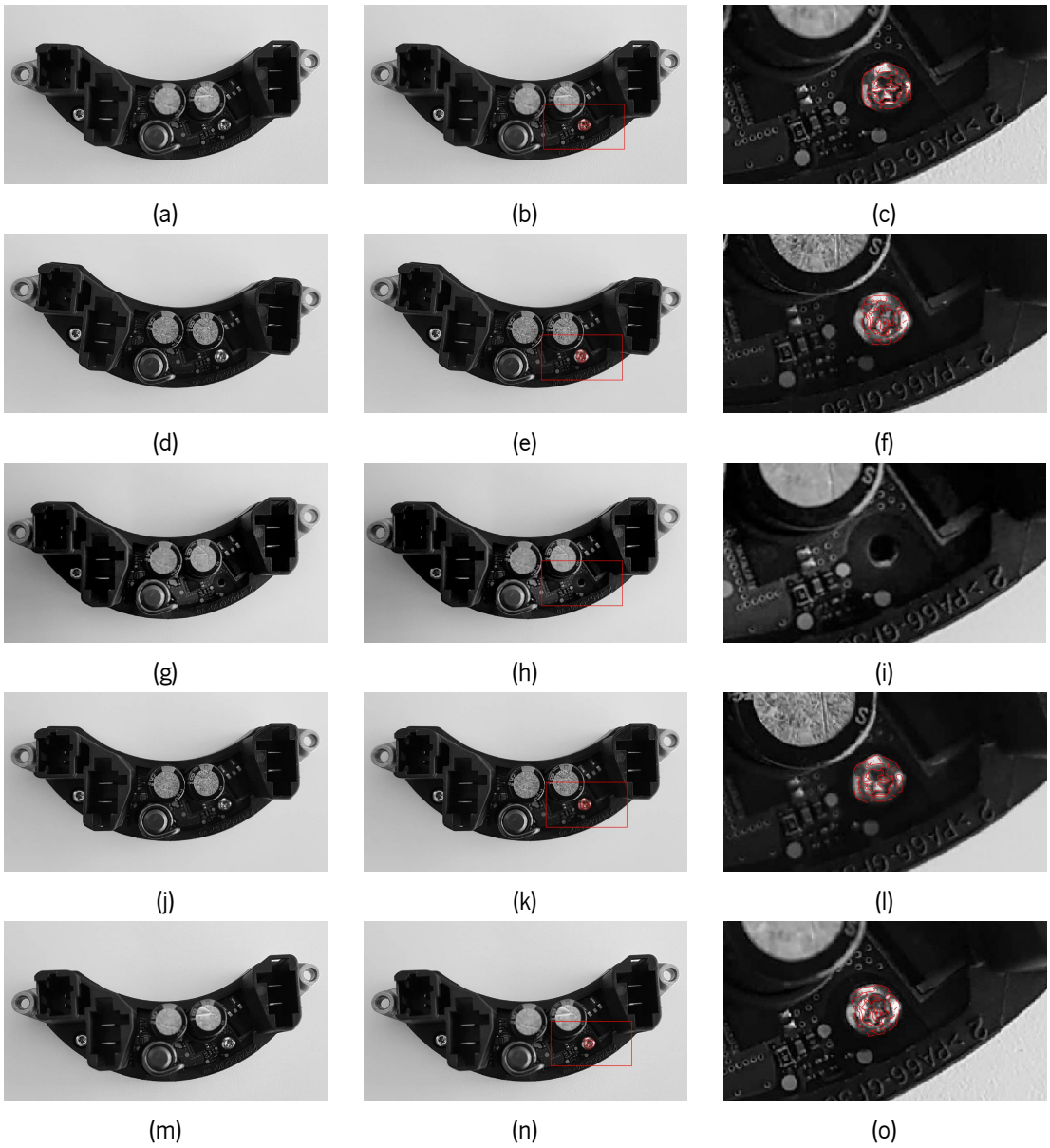
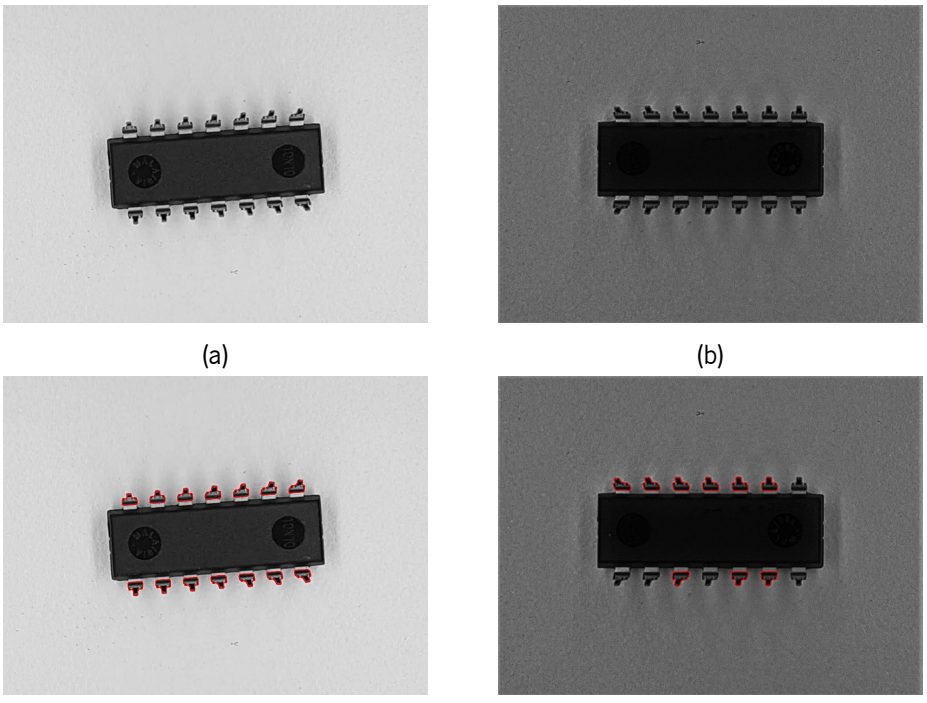


Figura 28: Diferentes resultados obtidos através da solução associada aos parafusos: (a), (d) (g), (i) e (m) imagem inicial; (b), (e), (h), (k) e (n) resultado do algoritmo para a deteção de parafusos; (c), (f), (i), (l) e (o) parte da imagem (b), (e), (h), (k) e (n), respetivamente.

As imagens recolhidas pelo sistema de aquisição implementado são referentes aos objetos fornecidos

para os diferentes problemas estipulados deste projeto. A qualidade da imagem é um reflexo da qualidade da câmara. Nos sistemas de IVA é muito importante a existência de câmaras de alta resolução, pois quanto maior a qualidade das imagens, mais rápido e minucioso será a execução do algoritmo. Associado à aquisição de imagens está a iluminação. Uma boa iluminação e seu correto posicionamento configura uma grande vantagem na hora da aquisição de imagem, principalmente no que toca a superfícies metálicas. As superfícies metálicas refletem grande parte da luz incidente sobre elas, logo com auxílio de luz, os objetos metálicos presentes na imagem ficarão mais brilhantes, contribuindo para uma melhor eficiência do algoritmo. Na Figura 29 é apresentada o efeito que a iluminação tem na aquisição de imagens e, consequentemente, no sistema desenvolvido. Na aquisição de imagens, uma das imagens foi suportada por iluminação extra, por via de LEDs (Figura 29a), enquanto que a outra apenas tem luz natural a incidir (Figura 29b). Verificou-se que a deteção de pinos toma resultados diferentes. Embora seja a mesma peça e o mesmo algoritmo, ocorreu a deteção de todos os pinos na Figura 29c, porém na Figura 29d existem pinos que não foram identificados, apesar do seu correto posicionamento.



(a) (b)

(c) (d)

Figura 29: Influência do recurso à iluminação para o mesmo algoritmo na deteção de pinos: (a) objeto iluminado por LEDs; (b) objeto com pouco iluminação; (c) resultado do algoritmo para a deteção de pinos na imagem (a); (d) resultado do algoritmo para a deteção de pinos na imagem (b).

Outro aspeto importante é o tempo de computação. Pretendia-se que as soluções implementadas funcionassem para as peças estudadas, ao mesmo tempo que executassem rapidamente. Para as peças estudadas e para as diferentes soluções, foram avaliados os tempos de computação. Os tempos médios medidos podem ser consultados na Tabela 2. Denota-se que, dentro do mesmo problema, o tempo de computação não aumenta de forma diretamente proporcional com o aumento do número de objetos a ser identificados, mas sim com a quantidade de memória (quilobyte[3] (kB)) gasta pelo modelo de forma

[3]Do inglês: *kilobyte*

criado para reconhecer a peça. Claramente que, quantas mais regiões estiverem definidas, mais vezes os métodos terão de ser aplicados, contudo o aumento do tempo de computação não é significativo, dado que o número de objetos a procurar não aumenta consideravelmente nos casos estudados. De um modo geral, para todas as peças de teste estudadas, os tempos de execução apresentados são inferiores a 70 milissegundos, exceto para a solução dos parafusos. Este tempo de computação pode ser explicado devido ao facto de esta solução ser a única que trabalha com dois modelos de forma e destes modelos representarem grande carga computacional.

Tabela 2: Avaliação dos tempos médios de execução e da memória ocupada pelo modelo de forma da peça, para cada uma das soluções e para diferentes peças.

	Peça nº1		**Peça nº2**		**Peça nº3**	
	Tempo médio	**Memória ocupada**	**Tempo médio**	**Memória ocupada**	**Tempo médio**	**Memória ocupada**
Fiducial circular	28 ms (8 fiduciais)	328 kB	37 ms (12 fiduciais)	449 kB	68 ms (3 fiduciais)	894 kB
Fiducial em cruz	42 ms (6 fiduciais)	1276 kB	44 ms (4 fiduciais)	1539 kB		
Pinos	29 ms (16 pinos)	145 kB	35 ms (14 pinos)	239 kB	42 ms (28 pinos)	1027 kB
Conta-rotações	16 ms	263 kB				
Parafusos	285 ms	238 kB				

4.4 Sumário

As soluções foram desenvolvidas recorrendo às propriedades do *Halcon*. Os seus métodos foram criados de modo a poderem ser implementadas noutros casos semelhantes. Estas foram idealizadas recorrendo aos requisitos de deteções pedidos e baseadas nas imagens recolhidas pelo sistema de aquisição. Foram desenvolvidas quatro soluções para diferentes deteções, nomeadamente, de tipos de fiduciais (formato circular e em cruz), de pinos, do centro geométrico de objetos e posicionamento de ponteiros e da ausência ou posicionamento defeituoso de parafusos.

As soluções implementadas foram baseadas nas peças fornecidas e no tipo de problema a ser identificado. Através do ambiente de desenvolvimento do *Halcon* foi possível desenvolver um conjunto de soluções para serem implementadas numa *toolbox*. As soluções criadas, na sua maioria, funcionaram para os diferentes casos estudados. Apenas uma das soluções não funcionou sempre corretamente, nomeadamente, a deteção da ausência ou posicionamento defeituoso de parafusos, podendo ser explicado pelo facto de a iluminação a mesma durante a aquisição de imagens

Além disso, a utilização de iluminação, por via de LEDs, na aquisição de imagens auxilia a análise do algoritmo, no sentido em que, principalmente quando se trabalha com superfícies metálicas, quando incididas por mais luz, estas também refletem mais luz, tornando-as mais brilhantes, aumentando seu o contraste.

Por último, a quantidade de memória utilizada para o modelo de forma interfere com os tempos de computação, na medida em que o tempo é tanto mais longo, quanto maior for a memória do modelo de forma.

Capítulo 5

Aplicação

O *Halcon* não apresenta um *front-end* próprio, pelo que foi necessário recorrer a outra plataforma para o desenvolvimento da aplicação. A plataforma utilizada foi o *Qt*. O *Qt* é uma *framework* multi-plataforma, escrito em *C++*, usada no desenvolvimento de vários tipos de aplicações. Contém uma vasta gama de classes e ferramentas, permitindo a interatividade de um utilizador com uma ou mais interfaces gráficas. Além das aplicações, também é possível desenvolver bibliotecas e compilá-las para outras plataformas [39].

Um dos objetivos do trabalho consiste na implementação de uma *toolbox* de soluções parametrizáveis, por isso a aplicação foi dividida em dois modos de ação. Uma é relacionada com as micro-operações e o modo como as variáveis de entrada destas podem ser manipuladas de forma a tornar as soluções configuráveis. Estas micro-operações são baseadas nas funções definidas como configuráveis e são secções das soluções da *toolbox* implementada. A outra parte é referente ao motor de execução da aplicação, onde as soluções customizadas são aplicadas às novas imagens para avaliação.

No seguinte capítulo é explicado como foi construída a aplicação que fará a ligação da *toolbox* com o utilizador. Depois será descrita a sua estrutura e os seus modos de funcionamento, repartidos pela definição das micro-operações e o motor de execução. No final serão demonstrados os resultados das interfaces implementadas e a forma como estas interagem com as soluções de modo a originar uma aplicação interativa e parametrizável.

5.1 Modos de Funcionamento

A aplicação apresenta dois modos de execução, um serve para definir as variáveis de entrada das micro-operações que, por sua vez, irão definir as variáveis de entrada da solução, possibilitando a sua configuração. O outro modo é relativo ao motor de execução da aplicação. Este executa as soluções parametrizadas nas imagens de teste. Em ambos os casos, o *display* das imagens é efetuado pela janela do *Halcon*, seguidamente explicado.

5.1.1 Micro-operações

A definição das micro-operações é muito importante para um bom funcionamento da *toolbox*. Uma micro-operação consiste numa fração da solução, ou seja, uma micro-operação pode ser constituída por apenas um ou um conjunto de operadores *Halcon*. A parametrização da solução apenas acontece quando surgem etapas específicas do algoritmo da solução, sendo que estas etapas são definidas em função dos operadores do *Halcon*, ativando assim uma micro-operação. É nas micro-operações que os parâmetros de entrada podem ser modificados e, por conseguinte, alteram a configuração de uma solução.

5.1.1.1 Estrutura do Algoritmo

Primeiramente, é selecionada e verificada uma imagem de referência que servirá como padrão para as imagens de teste. O ficheiro selecionado é lido e retornado, caso não se trate de uma imagem, uma imagem nula é retornada. Depois existe um *checkpoint* que caso o ficheiro selecionado não seja uma imagem, será exibida uma informação a notificar esse erro e o utilizador terá de escolher outro ficheiro. Caso a verificação seja bem sucedida, o algoritmo retorna a diretoria e nome do ficheiro para depois serem lidos com um operador *Halcon* e seguidamente visualizados na janela do *Halcon*. Enquanto que o objeto em estudo for o mesmo, também a imagem de referência deverá manter-se, visto que toda a inspeção foi realizada em torno dela e, assim, evitar a definição, por exemplo, de novas ROIs à medida que aparecem novas imagens para inspeção. A imagem de referência apenas deve ser alterada quando se muda de peça ou de solução. Quando uma nova imagem de referência é escolhida toda a solução é reinicializada, começando uma nova análise à peça de referência, possibilitando o utilizador de definir os novos parâmetros para a solução, através das micro-operações.

Seguidamente, após a seleção da imagem de referência, é escolhida e verificada a solução que se pretende aplicar. Assim como na seleção da imagem de referência, é retornado retorna a diretoria e nome do ficheiro. Relativamente ao nome, este é constituído pelo nome dado ao ficheiro e o tipo de ficheiro que este representa. Uma vez que as soluções são desenvolvidas em ficheiros de procedimento (*.hdvp*), apenas os ficheiros do mesmo tipo são aceites, caso contrário, a aplicação informa o utilizador que o ficheiro selecionado não é apropriado, caso o ficheiro seja considerado uma solução, uma breve descrição da sua funcionalidade será exposta na interface gráfica, através de uma *QLabel*. Por último, é enviado um sinal da interface das definições para a interface do motor de execução com o nome da solução selecionada, assim quando o motor de execução estiver a ser executado, sabe qual a solução e o ficheiro XML de dados que devem ser chamados.

O *Halcon* armazena os dados em formato XML, não sendo um formato proprietário, podendo ser lido por outra biblioteca, diferente do *Halcon*. Esta operação tem o intuito de retirar todos os passos protagonizados pela solução indicada, informando os tipos de operadores *Halcon* que serão executados pela solução. Este procedimento é extremamente importante porque, após a extração das linhas de código da solução, estas linhas são processadas de modo a apenas reterem o nome do operador e adiciona-lo a uma lista. Posteriormente, cada elemento da lista será lido e avaliado sendo que, caso seja um procedimento configurável, é apresentado um menu na interface, associado ao tipo de procedimento, que permitirá, por exemplo, definir novas ROIs ou ajustar valores, para uma otimização da solução para

o problema em causa. Caso não seja configurável, avança para a leitura do procedimento seguinte, até surgir outro procedimento configurável.

5.1.1.2 Interface

A GUI desenvolvida para esta fase encontra-se na Figura 30.

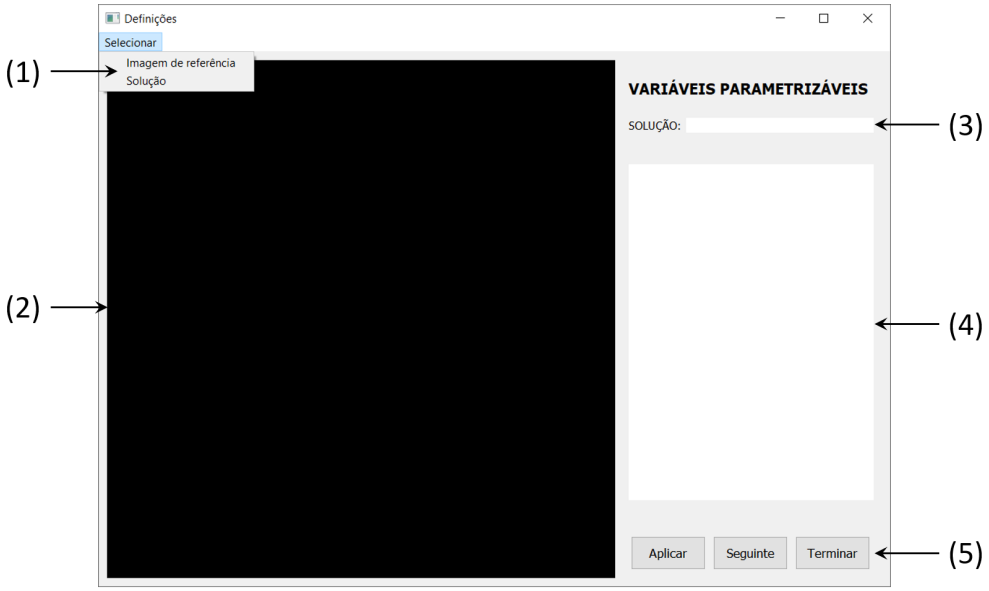


Figura 30: Interface das definições implementada, constituída por: (1) seleção da imagem de referência e da solução; (2) janela do *Halcon*; (3) breve designação da solução escolhida em (1); (4) *QWidget* dinâmico que apresentará as variáveis das micro-operações configuráveis; (5) botões interativos.

Relativamente ao botão "Aplicar", este permite a visualização, em tempo real, das alterações realizadas nas micro-operações. Quando o utilizador tem de configurar uma solução e o procedimento em causa resulta numa imagem, esta será apresentada na janela do *Halcon*. Este processo auxilia o operador ao exibir os resultados gráficos das micro-operações, de acordo com as suas modificações efetuadas. Porém, este botão não avança para o procedimento seguinte, apenas permite executar uma micro-operação as vezes desejadas. Contudo, se se tratar da criação da ROI, esta será guardada diretamente.

Ao longo do processo de parametrização, as variáveis podem sofrer ajustes, sendo necessário a sua constante atualização. Deste modo, no decorrer de uma configuração de solução, sempre que o operador pressionado o botão "Seguinte", os valores presentes no menu serão extraídos e guardados no ficheiro XML, além de avançar para o procedimento seguinte. Sempre que uma micro-operação é modificada, os resultados da solução, a partir desse ponto, poderão ser diferentes, por isso é que também é realizada uma atualização das imagens de saída. Porque, ao longo das etapas, algumas das imagens apresentadas na janela do *Halcon* são as imagens que serão utilizadas como entrada para um procedimento configurável, logo sempre que o utilizador fizer uma alteração, também o processamento futuro será diferente.

Por último, o botão "Terminar" permite encerrar e concluir as definições, voltando à interface inicial.

5.1.1.3 Tipos de Micro-operações

As micro-operações permitem a parametrização de um solução. Para tal, foi feita uma análise dos operadores, ou conjunto destes, que deviam ser implementados como uma micro-operação, dada a sua relevância na solução e permitindo que a sua alteração facilite na aplicação da solução noutros problemas semelhantes. Sendo assim foram definidas as seguintes micro-operações:

- Registo da peça de referência - reduz o domínio da imagem inicial para a imagem que será a entrada do modelo de forma, retornando a localização da peça de referência e o registo obtido;
- ROI - permite ao utilizador escolher o tipo de ROI que pretende desenhar, sendo as opções existentes: círculo, elipse, retângulo paralelo ao eixo, retângulo com qualquer orientação e região livre. Também é possível remover a última ROI desenha no caso o utilizador se engane. Caso as ROIs desenhadas já tenham sido aplicadas, é também possível reiniciar o processo;
- *Threshold* - segmenta uma imagem utilizando um *threshold* global, no entanto, este operador *Halcon* exige que sejam definidos um valor mínimo e máximo, ou seja, um intervalo de valores;
- Modelo de metrologia - não apresenta resultados gráficos, apenas serve para definir os limites das medições dos objetos;
- Seleção de uma caraterística - escolhe regiões com a ajuda de recursos de forma. É apresenta uma lista com estas caraterísticas (recurso de forma) que podem ser analisadas e, após a caraterística e seus valores mínimo e máximo serem definidos, pode ser visualizado o seu impacto nas regiões. Cada conjunto destes dados, caraterística e valores, são adicionados a uma tabela;
- Seleção de caraterísticas - todas as caraterísticas presentes na tabela são aplicadas à região de entrada. Também a última informação inserida na tabela pode ser removida e sucessivamente eliminada;
- Abertura/Fecho - têm funções diferentes, mas o seu formato é semelhante. Um aplica uma abertura numa região, enquanto que outro aplica um fecho. Em ambos os procedimentos é preciso definir raio do elemento estrutural;
- *Display* de resultados - é considerado uma micro-operação, mas não possui nenhuma variável que possa configurável. Apenas serve para apresentar os resultados obtidos das deteções.

Para cada algumas das micro-operações foi desenvolvido um menu interativo. Para implementar os diferentes menus recorreu-se a *widgets* do *Qt*, tais como, a *QLabel*s para informar o utilizador a que variável corresponde, aos *QDoubleSpinBox* para definir valores, até duas casa decimais, aos *QRadioButton* para escolher o tipo de formato da ROI, aos *QComboBox*, aos *QPushButton* e ao *QTableWidget* para escolher uma caraterística, aplicá-la ou remover a última e ser visualizada numa tabela. Na Figura 31 estão exemplificados os menus interativos que poderão surgir na aplicação, onde o utilizador poderá alterar os parâmetros das micro-operações e assim configurar diferentes soluções. Cada menu mostra as variáveis

que podem ser parametrizáveis para cada micro-operação. Contudo é importante referir que as micro-operações são uma parte das soluções, dado que existe outras partes constituídas por operadores *Halcon* mas não configuráveis.

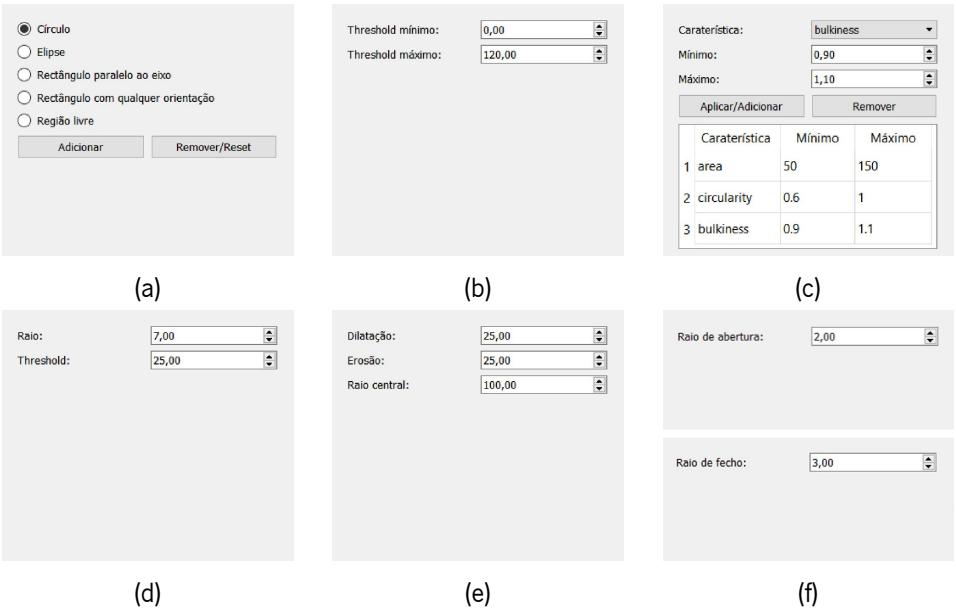


Figura 31: Menus interativos criados: (a) menu para a definição de ROIs; (b) menu do *threshold*; (c) menu da seleção de caraterísticas; (d) menu do modelo de metrologia; (e) menu para o registo da peça; (f) menu da abertura (cima) e do fecho (baixo).

5.1.1.4 Seleção e Armazenamento de Dados

Para este trabalho foi criada uma diretoria, denominada "DATA". Visto que a estrutura das diretorias é comum às soluções, foi preciso criar sub-diretorias. Estas são divididas pelo nome das soluções e no seu conteúdo estão os parâmetros de entrada necessários para a solução, nomeadamente, as variáveis icónicas e de controlo. As variáveis icónicas são gravadas diretamente na diretoria como variáveis do *Halcon* enquanto que as variáveis de controlo são adicionadas a um ficheiro XML de modo a facilitar a organização dos dados. Este ficheiro XML guarda a localização do objeto de referência, o valor das variáveis de entrada dos procedimentos configuráveis, o nome das ROIs e o nome dos modelos de forma, caso existam. Este ficheiro XML é importante uma vez que sempre que uma solução é selecionada, é necessário que as variáveis de entrada estejam definidas para que a execução da solução aconteça. Outra das razões é porque, todos os parâmetros de entrada definidos nas micro-operações serão, mais tarde, aplicados nas imagens de teste, através do motor de execução.

Os valores apresentados nas micro-operações advêm de um ficheiro XML, associado a cada solução, com os dados relativos às variáveis de entrada. Sempre que o utilizador avança para a micro-operação seguinte, os valores indicados nas micro-operações substituem os dados anteriores.

Dentro das listas, por vezes pode existir informação referente à mesma operação, quando essa operação acontece mais do que uma vez na solução correspondente. De forma a selecionar os dados corretamente, é criado um contador que avançará à medida que a operação surge. Assim, os valores são

corretamente inseridos nos respetivos menus. No momento de efetuar o armazenamento, são contabilizados os elementos das listas e escritos no ficheiro XML pela ordem apresentada na lista.

No que toca às variáveis icónicas, nomeadamente, imagens, modelos de forma e modelos de metrologia, o *Halcon* apresenta funções associadas à sua leitura e ao seu armazenamento. Assim, ao configurar a solução, estas variáveis são carregadas e armazenadas utilizando funções do *Halcon*, de modo a manter a congruência da informação.

5.1.2 Motor de Execução

Quando se inicia a aplicação, a sua primeira visão é a interface do motor de execução, visualizada na Figura 32. Esta funciona como a interface principal da aplicação, uma vez que a interface das definições só pode ser aberta através desta. Sempre que se pretender aceder às definições, independentemente do último passo intermédio que o utilizador tenha ficado na interface das definições, ao abrir uma nova interface, esta irá apresentar a imagem de referência inicialmente selecionada (caso a imagem tenha sido inicializada) e recomeçará a analisar a imagem desde o início da solução (caso a solução já tenha sido escolhida).

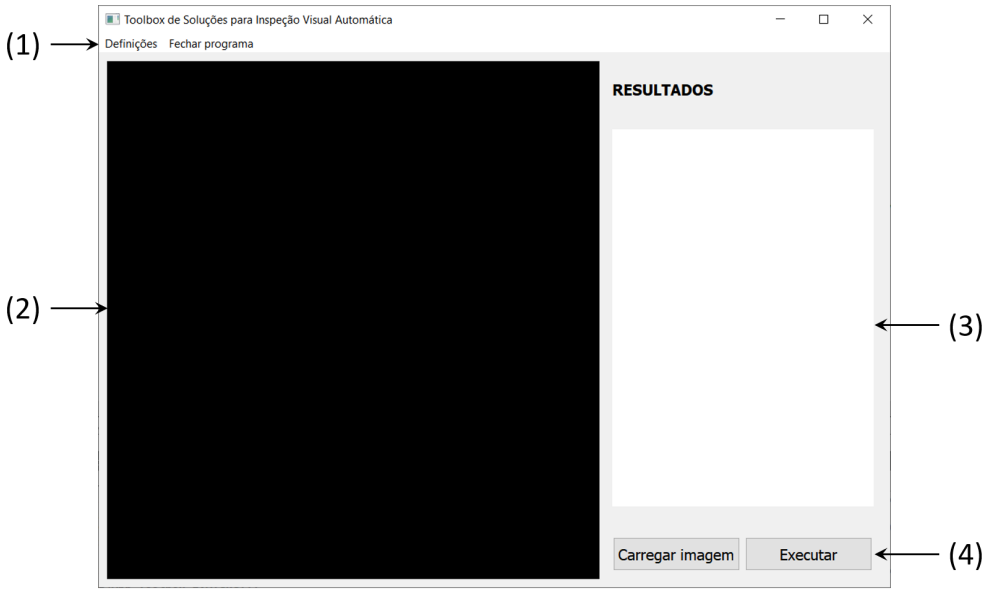


Figura 32: Interface do motor de execução desenvolvida, sendo constituída por: (1) barra de menu interativa, permitindo aceder às definições ou fechar a aplicação; (2) janela do *Halcon*; (3) *QWidget* dinâmico com os resultados obtidos; (4) botões interativos.

Assim como a interface das definições, a interface de execução apresenta a janela do *Halcon* (Figura 32 (2)) para observação de resultados gráficos e uma zona apropriada onde serão apresentados os resultados numéricos (Figura 32 (3)).

A funcionalidade mais importante desta interface encontra-se na área dos botões interativos. O botão, "Executar" (Figura 32 (4)), começa por ler o ficheiro XML, existente na diretoria "DATA". Depois, executa uma solução reduzida, daquela escolhida na interface das definições. Isto acontece porque existem certos métodos que não são necessários ser novamente aplicados. Sendo assim esta solução reduzida

é semelhante à carregada nas definições, contudo, as variáveis de entrada como as ROIs, os modelos de forma e os modelos de metrologia não precisam de ser novamente formados, pois eles já foram criados e guardados durante o processo das micro-operações. Nesta fase, estas variáveis são diretamente carregadas na solução, enquanto que as restantes variáveis são lidas e retornados através do ficheiro XML e inseridas na solução. Por fim, são extraídas todas as deteções efetuadas, assim como algumas das variáveis de controlo de saída, especificamente escolhidas. Estas últimas servirão de complemento à informação visual apresentada na janela do *Halcon*, variando o tipo informação com o tipo de solução escolhida.

5.2 Janela do Halcon

Na inspeção visual, a visualização de imagens é um requisito obrigatório, logo é necessário a implementação de uma janela gráfica que facultará estes dados, tais como, imagens e regiões. A janela gráfica do *Halcon* apresenta propriedades únicas. Contudo, como anteriormente explicado, este não oferece *front-end*. Assim, a incorporação da janela do *Halcon* dentro da aplicação é uma mais valia devido às suas caraterísticas, como por exemplo, desenho de ROIs interativas, essenciais para os algoritmos, pois permite que o utilizador não precise de estar sempre a criar ROIs à medida que trabalha com novas imagens.

Inicialmente, é adicionado um *QWidget* à interface, que será responsável pela janela do *Halcon*, e é retornado o seu identificador do sistema de janelas. Seguidamente, é criada uma janela através de um operador *Halcon*. O identificador é definido como janela fonte, de forma a que a janela do *Halcon* produzida seja contida no seu interior, assim mantendo as propriedades das janelas do *Halcon*, possibilitando a utilização das propriedades do *Halcon* dentro da interface da aplicação. Isto permite, por exemplo, desenhar ROIs na interface desenvolvida como se fosse a janela existente no *HDevelop*.

Apesar da janela ser criada pelas propriedades do *Halcon*, esta não apresenta nenhuma funcionalidade de manipulação das imagens, só é possível a sua visualização. Desta forma, foi necessário criar funções para permitir a manipulação da imagem pelo utilizador, de forma análoga ao que acontece no *Halcon*, adicionando novas funcionalidades, sendo elas:

- Possibilidade de movimentação da imagem através de clique contínuo na janela e arrastamento do cursor;
- Possibilidade de ampliação e diminuição da imagem (*zoom in* e *zoom out*) através do botão *scroll* do rato;
- Possibilidade de ajuste automático da imagem às dimensões da janela através de duplo clique na janela;
- Capacidade de redimensionamento. Permite que a janela possua sempre o mesmo tamanho do *QWidget*.

5.3 Resultados e Discussão

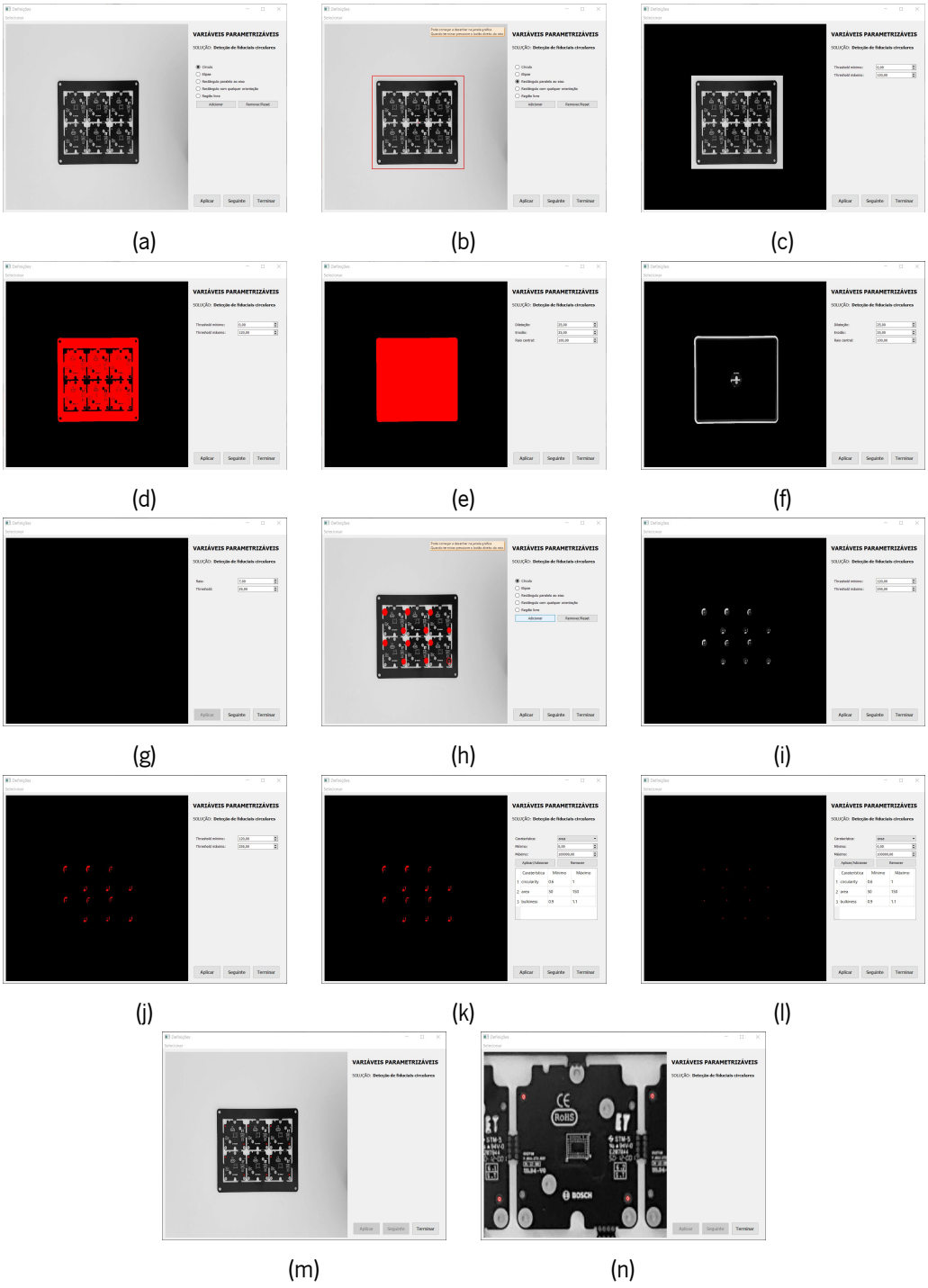


Figura 33: Exemplo de todos os passos de uma solução na interface das definições: (a) imagem de referência e solução escolhidas; (b) definição da ROI para o objeto; (c) imagem de entrada para a micro-operação de *threshold*; (d) aplicação da micro-operação de *threshold*; (e) região de entrada para a micro-operação de dilatação, erosão e raio central; (f) aplicação da micro-operação de dilatação, erosão e raio central; (g) definição dos parâmetros para o modelo de metrologia; (h) definição das ROIs para os fiduciais; (i) imagem de entrada para a micro-operação de *threshold*; (j) aplicação da micro-operação de *threshold*; (k) aplicação da micro-operação de seleção de caraterísticas; (l) aplicação da micro-operação de seleção de caraterísticas; (m) resultados obtidos; (n) *zoom in* dos resultados através da janela gráfica.

Na interface das definições, os parâmetros de entrada definidos irão servir de variáveis padrão para as imagens de teste na interface do motor de execução. Toda a parametrização deve ser cuidadosamente selecionada, desde a seleção das ROIs até aos valores de entrada para as diferentes micro-operações. Cada solução segue uma metodologia própria, portanto, a sequência de menus que aparecerão no *QWidget* também será diferente. A Figura 33 ilustra um exemplo da sequência de uma solução para a deteção de fiduciais circulares na interface das definições, desde a seleção da imagem de referência e da solução, até ao resultado final detetado pela configuração da solução. Na janela gráfica é possível visualizar as imagens e regiões que servem de parâmetro de entrada para as micro-operações, as imagens e regiões resultantes das micro-operações e os resultados obtidos da parametrização da solução. Ainda, pode-se verificar os menus a que se recorreu e as variáveis que foram utilizados ao longo da sua configuração. Nas Figuras 33b, 33h e 33n é também possível visualizar algumas das propriedades da janela gráfica, nomeadamente, desenhar a ROI para o objeto, para os fiduciais e fazer *zoom in* do resultado final.

A GUI das definições permite o utilizador redefinir a solução, passo a passo, para as necessidades futuras, como por exemplo, a alteração das condições externas, afetando a iluminação que incide no objeto ou uma troca do tipo de objeto, para o mesmo problema. Quando o utilizador tem intenções de alterar apenas um parâmetro, ele não precisa de refazer toda a solução, uma vez que o programa guarda os últimos parâmetros utilizados.

Relativamente à interface do motor de execução, esta só se torna executável quando o utilizador define a solução, na interface das definições, e carrega uma imagem para ser validada. Juntamente com os parâmetros utilizados na Figura 33, a Figura 34 demonstra a funcionalidade da interface de execução. Neste caso, para a deteção de fiduciais circulares os resultados são ilustrados na janela gráfica, por cima da imagem original, e apresentados na *QWidget* lateral, onde é possível averiguar a pontuação do objeto, o número de fiduciais que foram encontrados no objeto e o tempo de execução (em milissegundos). Estes resultados variam de solução para solução, de uma forma dinâmica, e as suas possibilidades encontram-se na Figura 35, devidamente identificadas.

(a) (b) (c)

Figura 34: Etapas processadas pela interface do motor de execução: (a) imagem de teste; (b) resultados da execução da solução; (c) *zoom in* dos resultados através da janela gráfica.

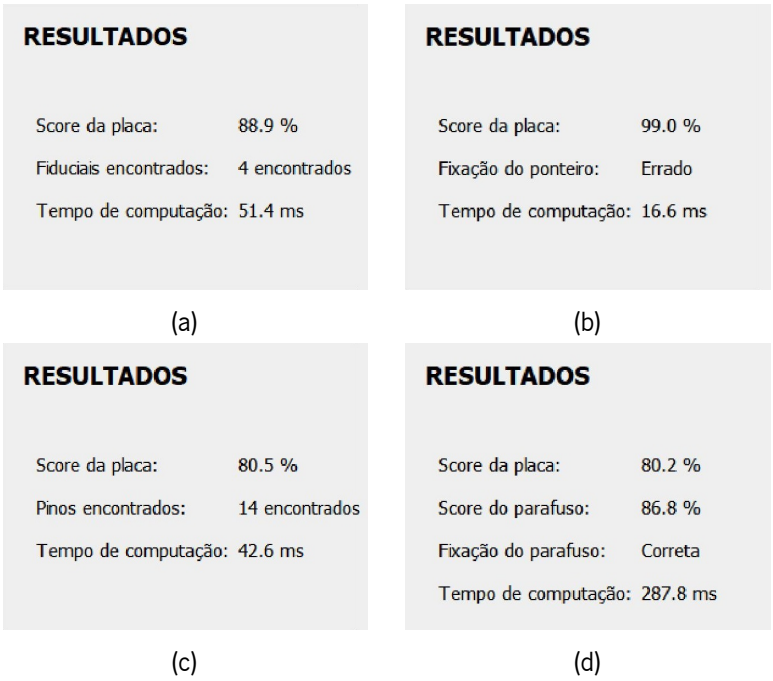


Figura 35: Exemplos dos diferentes resultados que podem ser apresentados de acordo com a solução: (a) tipos de fiduciais; (b) pinos corretamente posicionados; (c) centro geométrico e posicionamento de ponteiros; (d) ausência ou posicionamento defeituoso de parafusos.

5.4 Sumário

A aplicação, desenvolvida no *Qt*, permite a resolução dos casos estudados através da incorporação da *toolbox* implementada. É graças à aplicação que a configuração dos parâmetros de entrada é possível, para depois serem inseridos nas micro-operações e nas soluções reduzidas. Dado que existem fatores externos que poderão interferir com o normal decorrer dos sistemas de IVA, a implementação de uma *toolbox* parametrizável é um bom método proposto. Para tal, a resolução dividiu-se em duas fases. Uma que permite ler a imagem de referência, selecionar a solução e definir todos os parâmetros para a deteção de objetos, através de menus interativos. A outra fase, fará uso dos dados adquiridos na fase anterior, e os aplicará nas imagens de teste, retornando os resultados e a informação fornecida em cada uma das soluções. Os diversos procedimentos podem ser consultados, visualmente, na janela do *Halcon* implementada, presente em ambas as interfaces.

Capítulo 6

Conclusões e Perspetivas Futuras

Neste capítulo final, é realizada uma síntese do trabalho desenvolvido ao longo desta dissertação e são apresentadas as conclusões mais relevantes. Por último, são apontados alguns aspetos que poderão ser alvo de trabalho futuro.

6.1 Conclusão

A finalidade da presente dissertação consistiu no desenvolvimento de uma *toolbox* de soluções para a IVA. Ao longo dos anos, os sistemas de IVA têm vindo a sofrer melhorias de modo a contribuir para um maior nível de qualidade dos produtos. Estes sistemas são auxiliados por ferramentas com diversas capacidades, tais como, *Halcon*, *EyeVision* e *VisionPro*. Devido às funcionalidades e o facto de poderem ser aplicados em diversas áreas, os sistemas de IVA trazem inúmeras vantagens para auxiliar os operadores em muitas aplicações, nomeadamente na área industrial. Contudo, ainda não são utilizadas *toolboxes* de soluções para problemas específicos, bem como algoritmos parametrizáveis, tendo sido estas as motivações fundamentais para a realização deste trabalho.

O trabalho realizado foi dividido em duas fases distintas. A primeira incidiu no desenvolvimento de soluções para quatro problemas, nomeadamente na sugestão de métodos para a deteção de fiduciais, de pinos ausentes ou dobrados, do centro geométrico de objetos e posicionamento de ponteiros, e da ausência ou posicionamento defeituoso de parafusos. Na segunda, foi criada uma aplicação de modo a permitir parametrização das soluções, com o objetivo de poderem ser aplicados em problemas semelhantes ou devido a possíveis alterações das caraterísticas ambientais.

As soluções implementadas baseiam-se nos operadores fornecidos pelo *Halcon* e foram adaptadas para os diferentes problemas. Dadas as necessidades deste trabalho, o *Halcon* provou ser uma mais valia e totalmente adequado para a resolução dos problemas facultados. No primeiro caso, a solução para a deteção de fiduciais, em forma de círculo e cruz, funcionou para os objetos estudados, tendo sido devidamente identificados todos os fiduciais presentes nas ROI definidas. No segundo caso, o algoritmo foi capaz de reconhecer quais os pinos que se encontram corretamente colocados. Para os restantes pinos, pinos ausentes e dobrados, a solução não faz qualquer tipo de identificação ou divisão. No terceiro caso, a localização do ponto geométrico de um objeto é efetuada com sucesso, porém, a sua precisão

pode não ser a mais acertada, devido ao facto do objeto não estar totalmente paralelo com a câmara. Depois, tanto a deteção da barra como do ponteiro são bem sucedidas, permitindo à solução avaliar se o ponteiro se encontra direcionado para a barra correspondente. No quarto e último caso, a solução criada para a deteção de parafusos exibe algumas falhas no que toca à decisão do correto posicionamento do parafuso, pelo que se conclui que a solução não é robusta para casos onde a iluminação não seja apropriada.

Relativamente à aplicação desenvolvida no *Qt*, o seu principal objetivo é parametrizar as soluções, de forma a ser interativa e intuitiva. Efetivamente, os resultados apresentados na aplicação vão de encontro aos objetivos traçados, ao permitir o ajuste dos algoritmos às particularidades dos problemas. Para um determinado problema, após a definição de um conjunto de parâmetros de entrada que melhor se adequam, através de micro-operações, a aplicação possibilita o processamento automático de um número infinito de imagens, para o mesmo objeto. Os tempos de computação apresentados na aplicação são abaixo dos 70 milissegundos e apenas uma solução registou tempos médios na ordem dos 285 milissegundos.

Em resumo, as soluções propostas para os problemas expostos realmente funcionaram, além de também funcionarem em todas as peças analisadas e facultadas para este trabalho, permitindo validar o potencial da *toolbox* de soluções e da aplicação desenvolvidas para a IVA.

6.2 Perspetivas Futuras

Ao longo deste trabalho foram verificados alguns aspetos que podem ser alvos de melhorias. Nomeadamente no que diz respeito ao *setup*. A aquisição de imagens poderia beneficiar da colocação do *setup* numa caixa negra. Esta medida faria com que todos os fatores externos passariam a ser eliminados, deixando assim de interferir com a aquisição de imagens. Outra opção seria a adição de LEDs, de modo a manter a luminosidade fixa e uniforme. Relativamente à câmara, a utilização de uma câmara com maior resolução também poderia representar uma vantagem contribuindo para a obtenção de imagens de maior qualidade. Uma configuração do *setup* capaz de minimizar perturbações externas e aumentar a qualidade das imagens adquiridas é essencial para potencias o bom do funcionamento do sistema.

No que toca à aplicação, a apresentação de textos na interface para informar o utilizador que procedimento está a ser efetuado e qual a sua finalidade.

Outro ponto pertinente passa por o sistema implementado precisar de ser alvo de mais estudos. Especificamente, este deveria ser testado com mais casos e, em outros problemas. Por fim, deverá ser efetuada a implementação e teste do sistema numa linha de montagem.

Bibliografia

[1] Andrew DH Thomas, Michael G Rodd, John D Holt, e CJ Neill. Real-time industrial visual inspection: A review. *Real-Time Imaging*, 1(2):139–158, 1995.

[2] J Simão. Inspeção visual automática de mamões. Master's thesis, Universidade Federal do Espírito Santo, 2003.

[3] James W Schoonard e John D Gould. Field of view and target uncertainty in visual search and inspection. *Human Factors*, 15(1):33–42, 1973.

[4] M Bueno, M Stemmer, e PSDS Borges. Inspecção visual automática de peças cerâmicas via inteligência artificial. *Cerâmica Industrial, 5 (5), Brasil, Setembro/Outubro 2000*, 2000.

[5] Eider Lúcio de Oliveira. Inspeção automática de cerâmicas lisas via lógica difusa. Master's thesis, Universidade Federal de Santa Catarina, 1997.

[6] Luis Miguel Morais Martins. Inspeção visual automática em problemas industriais. Master's thesis, Instituto Superior Técnico, 2013.

[7] Ernest W Kent e Michael O Shneier. Eyes for automatons: Faster, more accurate, and more consistent than humans can hope to be, machine vision systems are on their way to broad application. *IEEE Spectrum*, 23(3):37–47, 1986.

[8] Elias N Malamas, Euripides GM Petrakis, Michalis Zervakis, Laurent Petit, e Jean-Didier Legat. A survey on industrial vision systems, applications and tools. *Image and vision computing*, 21(2): 171–188, 2003.

[9] Simon Reich, Florian Teich, Minija Tamosiunaite, Florentin Wörgötter, e Tatyana Ivanovska. A data-driven approach for general visual quality control in a robotic workcell. In *Journal of Physics: Conference Series*, volume 1335, page 012013. IOP Publishing, 2019.

[10] MVTec Software GmbH. Halcon – the power of machine vision, Outubro 2019. URL `https://www.mvtec.com/products/halcon/`.

[11] MVTec Software GmbH. *Quick Guide*, 2018.

[12] Hui-lan Luo, Yu Long, Xiao-Bing Xie, e Jin-Cheng Huang. Realization of vehicle license plate character recognition based on halcon. In *2011 4th International Congress on Image and Signal Processing*, volume 2, pages 936–939. IEEE, 2011.

[13] MVTec Software GmbH. Merlic - a new generation of machine vision software, Outubro 2019. URL `https://www.mvtec.com/products/merlic/`.

[14] MVTec Software GmbH. Tools and features, Outubro 2019. URL `https://www.mvtec.com/products/merlic/product-information/tools-and-features/`.

[15] MVTec Software GmbH. Merlic – create an application in five minutes, Outubro 2019. URL `https://www.mvtec.com/news-press/video/detail/merlic-create-an-application-in-five-minutes/?no_cache=1&cHash=acc566a4bab9eb237dc0b4b144b735da`.

[16] Eye Vision Technology GmbH. Bildverarbeitungssoftware eyevision, Outubro 2019. URL `http://evt.eyevision-web.com/produkte/eyevision-software/`.

[17] Eye Vision Technology GmbH. Eyevision image processing software, Outubro 2019. URL `https://www.cognex.com/products/machine-vision/vision-software/visionpro-software`.

[18] Eye Vision Technology GmbH. Evt wire inspector fÜr korrekte kabelbelegung, Outubro 2019. URL `http://evt.eyevision-web.com/evt-wireinspector-fuer-korrekte-kabelbelegung/`.

[19] Cognex Corporation. Software visionpro, Outubro 2019. URL `https://www.cognex.com/pt-pt/products/machine-vision/vision-software/visionpro-software`.

[20] Oskar Bremer. 3d-modeling with model vision pro 9.0. 2010.

[21] Cognex Corporation. Visionpro software, Outubro 2019. URL `http://evt.eyevision-web.com/produkte/eyevision-software/`.

[22] Cognex Corporation. Cognex visionpro training video, Outubro 2019. URL `https://www.cognex.com/pt-pt/videos`.

[23] National Instruments Corporation. O que é o vision builder for automated inspection?, Outubro 2019. URL `https://www.ni.com/pt-pt/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-vision-builder-for-automated-inspection.html`.

[24] Ismail Al Kamal e Mohamad Adnan Al-Alaoui. Online machine vision inspection system for detecting coating defects in metal lids. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2. Citeseer, 2008.

[25] National Instruments Corporation. What is vision builder for automated inspection?, Outubro 2019. URL https://www.ni.com/pt-pt/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-vision-builder-for-automated-inspection.html.

[26] Phansak Nerakae, Pichitra Uangpairoj, e Kontorn Chamniprasart. Using machine vision for flexible automatic assembly system. *Procedia Computer Science*, 96:428–435, 2016.

[27] MVTec Software GmbH. *HDevelop User's Guide*, 2018.

[28] Wolfgang Eckstein e Carsten Steger. The halcon vision system: an example for flexible software architecture. In *Proceedings of 3rd Japanese Conference on Practical Applications of Real-Time Image Processing*, pages 18–23. Citeseer, 1999.

[29] MVTec Software GmbH. Integrated development environment for machine vision, Outubro 2019. URL https://www.mvtec.com/products/halcon/work-with-halcon/hdevelop/.

[30] MVTec Software GmbH. *Programmer's Guide*, 2018.

[31] MVTec Software GmbH. Hdevengine, Outubro 2019. URL https://www.mvtec.com/products/halcon/work-with-halcon/hdevengine/.

[32] MVTec Software GmbH. *HALCON/HDevelop - Operator Reference*, 2018.

[33] MVTec Software GmbH. *Solution Guide II B - Matching*, 2018.

[34] MVTec Software GmbH. *Solution Guide III B - 2D Measuring*, 2018.

[35] MVTec Software GmbH. *Solution Guide III A - 1D Measuring*, 2018.

[36] David Vernon. Machine vision-automated visual inspection and robot vision. *NASA STI/Recon Technical Report A*, 92, 1991.

[37] Leonid Naimark e Eric Foxlin. Fiducial detection system, June 12 2007. US Patent 7,231,063.

[38] Zheng Liu, Hiroyuki Ukida, Pradeep Ramuhalli, e Kurt Niel. *Integrated Imaging and Vision Techniques for Industrial Inspection*. Springer, 2015.

[39] Ray Rischpater. *Application development with qt creator*. Packt Publishing Ltd, 2013.