FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sistema de Detecção de Impactos

Peter Edward Salgado Cebola

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Prof. Dr. Armando Luís Sousa Araújo

29 de Junho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Sistema de Detecção de Impactos"

foi aprovada em provas realizadas em 24-07-2015

o júri

Presidente Professor Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Paulo Jorge Campos Costa
Professor Adjunto do Departamento de Ciências Básicas e da Computação da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo

Professor Doutor Armando Luís Sousa Araújo
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Peter Edward Salgado Cebola

Faculdade de Engenharia da Universidade do Porto

Resumo

Desde o seu aparecimento na China por volta de 1250 AC [1], a arma de fogo tem vindo a ser primordial em termos bélicos. No entanto, a mesma tem vindo ao longo dos tempos a ser também utilizada para a caça, bem como para fins lúdicos em tiro ao alvo. De facto, o tiro ao alvo como modalidade desportiva de excelência surge desde logo nos primeiros Jogos Olímpicos em 1896. Quer as armas, quer as competições têm vindo a evoluir, sendo comum nos tempos modernos a utilização de meios eletrónicos associados à gestão da organização das provas, com sistemas mais ou menos sofisticados para a contagem e visualização, por todos os intervenientes, atletas, juízes, imprensa e público em tempo real das pontuações.

Deste modo, esta dissertação tem como objetivo o projeto de um protótipo funcional de um sistema de classificação automática para competições de tiro ao alvo, usando aquisição e processamento de imagem. As especificações do mesmo foram definidas com base num estudo da tecnologia atual e das regras impostas pela ISSF (*International Shooting Sport Federation*), organismo que gere o tiro de precisão a nível internacional. Tal conduziu a que o *hardware* utilizado fosse uma placa de computador único associada a uma câmara de vídeo, nomeadamente a Raspberry Pi 2 com o seu módulo de câmara. A arquitetura de *software* usa aquisição de imagens, com recurso à câmara mencionada, seguidas de um processamento da mesma, composto por três partes fundamentais: definição da região de interesse, obtenção dos anéis do alvo e obtenção das posições dos furos, tudo isto para o objetivo principal de cálculo da pontuação. No final apresenta-se a todos os intervenientes, a partir de uma interface gráfica, uma imagem virtual do alvo com a informação relevante retirada. No desenvolvimento do código, foram testadas várias abordagens, recorrendo a técnicas como a transformada de Hough circular, deteção de contornos de objetos e uso da respetiva informação de defeitos de convexidade e à segmentação a partir da cor no espaço HSV, *Hue Saturation Value*. Recorreu-se ainda ao uso de métodos numéricos de máxima verosimilhança, tais como o método de mínimos quadrados e estimador-M para a aproximação de uma linha, para estimar o melhor valor de centro e raio dos anéis do alvo e furos causados por impactos.

Foram realizados testes de maneira a averiguar a eficácia dos algoritmos. Foram ainda efetuadas comparações de tempos de execução, entre diferentes plataformas, e entre funções para o mesmo fim, para além do cálculo do erro entre a distância ao centro.

Abstract

Since its beginnings on China till its expansion as the weapon of excellence on contemporary war, the firearm was one of the inventions that changed the world in socio-cultural, economic and political ways. Despite the warlike nature of the device, it is also used as a tool for the practice of the shooting sport, introduced in the Olympic Games in 1896. To help the management of shooting events and speed up processes such as scoring, electronic systems have been developed for posterior commercialization, resorting to various technologies available.

This document's objective is to produce a functional prototype of a ranking system for shooting target competitions, using image processing. The restrictions defined were based on research of state-of-art technologies already employed and the rules imposed by the ISSF. The material used was a Raspberry Pi 2 with the camera module. The software architecture is composed by image acquisition by means of the beforementiond camera, followed by image processing that is composed by three modules: The first, the region of interest definition, followed by the retrieval of the target rings and finally the retrieval of the holes in the cardboard. These procedures allows the calculation of the final score. In the end, the score is shown to the user through an interface, showing the target with relevant information retrieved from the previous operations. While developing the final code, several approaches were experimented, using techniques like Circular Hough Transform, contour detection and convexity defect analisis, and HSV color segmentation. Numerical methods of maximum likelihood were also used, such as least-square fit and M-estimator for line approximation to estimate the best value for the center, the rings radius and the impact holes.

Tests were conducted in order to ascertain the effectiveness of the algorithms. Comparisons of runtimes between different platforms and between functions for the same purpose were made, in addition to the calculation of the distance for the score value.

Agradecimentos

A realização deste trabalho só foi possível graças a todas as pessoas e entidades que, direta ou indiretamente, contribuíram para o meu percurso pessoal e académico, tendo-me ajudado a superar este desafio. Às seguintes pessoas gostaria de deixar um agradecimento especial: Ao meu orientador Professor Dr. Armando Sousa Araújo pelo seu indispensável apoio, pela simpatia demonstrada, pelo conhecimento transmitido e pelo incentivo prestado, sobretudo nas alturas de maior frustração;

No ambiente de faculdade, quero deixar um agradecimento especial a certos professores: ao Professor Dr. Adriano Carvalho, por desafiar-me, transmitir o seu vasto conhecimento e ensinar-me a ter método de trabalho; ao Professor Dr. Armando Jorge de Sousa, por orientar-me durante este percurso de cinco anos e proporcionar-me vários desafios a nível extra-curricular e ao Professor Dr. José Carlos Alves, por ser uma pessoa excecional e paciente com os seus alunos e disponibilizar o laboratório I001 para trabalhar no âmbito da tese. Também quero agradecer a um grupo de pessoas que me ajudam e suportam em diferentes situações da vida: Tiago Cunha, por ser meu amigo de longa data e por ser o meu parceiro de trabalhos de grupo; Ricardo Carvalho, por inspirar-me a nunca desistir e mostrar-me o que é uma pessoa verdadeiramente trabalhadora; Valter Costa, por ser uma pessoa bestial e meter-me juízo na cabeça; Hugo Costa, por ser o modelo de engenheiro ideal e direcionar-me nas tarefas de carácter manual, Susana Neves, por fazer o papel de "mãe"e orientar os "meninos do laboratório"; Filipe Lopes, por lembrar-me que sou uma pessoa decente; Pedro Tavares, por apoiar sempre que fiquei em baixo; Diana Neves, por ser a pessoa mais compreensiva e humana do laboratório, apesar de ser constantemente irritante e ao resto das pessoas do I001, que são uma segunda família.

Quero agradecer aos meus pais pelo suporte básico de vida fornecido, tanto a nível emocional como a nível de saúde: a minha mãe pela paciência e trabalho constante dos meus pedidos; o meu pai por suportar a minha família. Os meus dois irmãos também: o meu irmão John pela compreensão da dificuldade do percurso académico, por direcionar-me para o caminho certo noutros percursos pessoais e por estar sempre pronto a posar quando os irmãos Pose são chamados ao serviço; a minha irmã Melissa por lembrar-me que há lazer para além do trabalho e ter a bondade de aguentar os meus disparates constantes. E finalmente, agradecimentos especiais aos meus avós paternos e avó materna, que sempre me apoiaram.

Peter Edward Salgado Cebola

"When life gives you lemons? Don't make lemonade.
Make life take the lemons back! Get mad!
I don't want your damn lemons! What am I supposed to do with these?"

Cave Johnson

Conteúdo

Lista de Figuras

Abreviaturas e Símbolos

API	*Application Programming Interface*
CPU	*Central Processing Unit*
CSI	*Computer System Interface*
GPIO	*General-Purpose Input/Output*
GPU	*Graphics Processing Unit*
GUI	*Graphical User Interface*
HDMI	*High Definition Media Interface*
HSV	*Hue-Saturation-Value*
IDE	*Integrated Development Environment*
ISSF	*International Shooting Sport Federation*
LCD	*Liquid-Crystal Display*
LED	*Light Emmiting Diode*
LoG	*Laplacian of Gaussian*
LUT	*Look-Up Table*
MMC	*MultiMedia Card*
MMQ	Método de Mínimos Quadrados
MTP	*Media Transfer Protocol*
NICTA	*National Information and Communications Technology Australia*
PC	*Personal Computer*
PTP	*Picture Transfer Protocol*
RAM	*Random Access Memory*
RGB	*Red-Green-Blue*
ROI	*Region of Interest*
RPi	*Raspberry Pi*
SD	*Secure Digital*
SVD	*Singular Value Decomposition*
USB	*Universal Serial Bus*

g	Grama
Hz	Hertz
lx	Lux
m	Metro
P	*Pixel*
s	Segundo

G	giga
M	mega
m	mili

Capítulo 1

Introdução

Este capítulo introdutório apresenta a motivação associada a este trabalho, um breve contexto histórico das armas de fogo e o desporto de tiro ao alvo, os objetivos a alcançar no desenvolvimento do projeto de dissertação e um guia da estrutura deste relatório.

1.1 Contexto

O conceito de armas de fogo é originário da China, sendo as primeiras construídas no final do século XIII [2]. Estas não eram mais que simples canhões de bronze, versões rudimentares do que se iria observar no futuro. Nos próximos séculos o seu uso iria ser difundido para outras partes do mundo, como a Europa (que iria aperfeiçoar vários mecanismos como o fecho de mecha) e a Ásia. A arma de fogo só volta a receber grandes desenvolvimentos no século XVIII, no norte da América, a partir de pessoas como Samuel Colt, inventor do primeiro revólver (Colt) e Eliphalet Remington, inventor da espingarda Remington [3].

Apesar do fim bélico para que as armas de fogo foram criadas, elas também são usadas para a caça bem como, desportivamente, no tiro ao alvo. Diferentes tipos de eventos como pistola de velocidade, pistola a 25 metros, pistola a 50 metros, fosso olímpico e *skeet* foram introduzidos como modalidades nos jogos Olímpicos em 1896 [4], sendo estandardizada em 1907 pela ISSF (Federação Internacional de Tiro Desportivo).

Com a evolução rápida da eletrónica, nestes últimos 50 anos, e a sua introdução em vários sectores da indústria, de maneira a agilizar e simplificar o trabalho e produção realizados pela manufatura, algumas tecnologias foram já utilizadas para o desporto de tiro ao alvo, de maneira a simplificar tarefas como a visualização de alvos e contagem de pontuação.

1.2 Motivação

A classificação dos alvos de tiro ao alvo de competição, antes dos classificadores eletrónicos, era exclusivamente efetuada manualmente. Normalmente, apenas por inspeção visual consegue-se realizar a pontuação. Nos casos que podem gerar dúvidas utilizam-se os chamados calibradores,

ferramentas que garantem que o alvo seja bem pontuado. Todo este processo envolve alguma logística e recursos humanos, o que atrasa as competições de tiro. Este facto ameaçou o estatuto de modalidade nos Jogos Olímpicos, já que obrigava, normalmente, a elevados tempos de atraso entre o final das provas e a proclamação dos vencedores. Tal tornava este tipo de competições de pouco interesse para os espetadores [5].

A partir dos anos 80 começam a ser usados os primeiros sistemas eletrónicos de pontuação. Este sistema é primeiramente utilizado nas finais olímpicas de 1990 [6]. Este sistema foi um sucesso, já que permitiu que os espetadores tivessem acesso, em tempo real, às pontuações obtidas, removendo o problema do atraso temporal, aproximando assim o tiro ao alvo às outras modalidades. Hoje em dia, existem vários sistemas de pontuação automática no mercado, com recurso a variadas tecnologias. Podem ser para grandes recintos ou aplicações de *software* para ajudar no treino pessoal.

Este tipo de sistemas são normalmente fechados, maioritariamente usados em carreiras de tiro, são soluções caras, com preços a rondar os milhares de euros, pouco flexíveis, desenvolvidas apenas para alguns tipos específicos de alvo e oferecidos apenas por poucas empresas.

Assim, o desenvolvimento de uma solução barata, flexível, fiável e de fácil instalação, que consiga ser competitiva com a oferta existentes no mercado, será um produto com sucesso certo. Para este objetivo, o estado atual da Engenharia possui um leque variado de soluções recorrendo a diferentes tecnologias, como, por exemplo, o processamento de som, luz e imagem.

Deste modo surgiu a ideia de tentar a implementação de um sistema de classificação automática de alvos, baseado em visão, com as características atrás enunciadas.

1.3 Objetivos

O sistema a desenvolver tem como objetivo fulcral a identificação automática dos impactos nos alvos devido aos disparos efetuados pelo atirador. Tal identificação deverá ser realizada a partir de um sistema baseado em visão por computador. O sistema irá permitir a visualização, em tempo real, de vários dados, tais como a pontuação atribuída a cada impacto, a pontuação total e o ponto médio dos disparos efetuados. Esta informação também pode ser mantida num histórico armazenado pelo sistema.

De modo a conseguir um sistema capaz de tais objetivos apresenta-se, de seguida, uma lista dos requisitos que para tal são necessários.

- **Aquisição de imagem** — Com a aquisição da imagem pretende-se obter uma fotografia do alvo com a resolução suficiente para permitir a classificação correta. Tal irá levar uma resolução mínima do *hardware* utilizado.

- **Processamento de imagem** — A partir da imagem obtida, o tratamento da mesma deverá permitir encontrar o centro do alvo, bem como os círculos correspondentes a cada disparo e posteriormente o centro de cada um. Tal implica o tratamento dos dados, tanto a nível de ruído, como em informação útil e análise de características associadas aos cálculos. Para tal,

é necessária a utilização de métodos numéricos, para o encontro do valor real do centro do alvo e cálculo da pontuação e do ponto médio de disparos, tais como métodos de mínimos quadrados e máxima verosimilhança;

- **Visualização gráfica da informação** — Realizados os passos anteriores, a visualização da informação deverá ser feita, graficamente, num alvo virtual, num ecrã de computador ou de um *tablet*.

Para além destes requisitos, está sempre presente como base que a solução a desenvolver seja de baixo custo comparativamente às existentes atualmente no mercado. Também é relevante ser de fácil utilização, não sendo necessário o uso de tecnologias avançadas ou de difícil compreensão para o utilizador comum.

1.4 Estrutura do Relatório

Uma breve descrição da estrutura deste relatório é apresentada de seguida.

A totalidade do documento engloba seis capítulos, sendo este o primeiro. Assim, neste capítulo foi apresentado o contexto e motivação do projeto de dissertação e os objetivos fulcrais a alcançar.

No capítulo 2, Revisão Bibliográfica, é realizada uma revisão bibliográfica, descrevendo métodos de processamento e análise de imagem digital bem como algoritmos matemáticos relevantes para a execução do projeto. Também é realizado um levantamento dos sistemas de deteção já existentes no mercado e soluções académicas propostas.

No capítulo 3, Descrição do Sistema, apresentam-se os fundamentos e regras básicas do desporto de tiro ao alvo de competição, definindo algumas das restrições e necessidades a tomar, de maneira a escolher os componentes de *hardware* necessários para a construção do protótipo pretendido, justificando assim as opções tomadas.

No capítulo 4, Proposta de *Software*, é apresentado o código desenvolvido, explicando as técnicas usadas com toda a informação disponível do problema e apontando para limitações dos algoritmos criados.

No capítulo 5, Resultados Obtidos, são apresentados os resultados obtidos com o sistema desenvolvido, concretamente os resultados dos testes realizados quer na deteção dos impactos, quer na precisão conseguida.

Finalmente, no capítulo 6, Conclusões e Trabalho Futuro, apresentam-se não só as principais conclusão do trabalho efetuado, como também possíveis melhoramentos a serem futuramente implementados.

Capítulo 2

Revisão Bibliográfica

Neste capítulo é apresentada a revisão bibliográfica referente às tecnologias e matérias científicas relevantes à execução do projeto.

2.1 Processamento digital de imagem

Esta secção ilustra os principais métodos para o tratamento e análise de imagem direcionados para a deteção de tiro ao alvo, apresentando uma breve explicação do seu funcionamento.

2.1.1 Pré-processamento

O pré-processamento de uma imagem é qualquer operação com o intuito de modificar uma imagem digital de maneira a facilitar a sua análise, suprimindo degradações indesejadas e/ou realçando caraterísticas relevantes. O tratamento é realizado com recurso a matrizes ou funções, as quais permitem controlar o contraste e realce da imagem ou efetuar transformações geométricas. As operações são realizadas ao nível pontual (do *pixel*), local (conjunto de pixeis) ou global (a totalidade da imagem). De seguida são apresentadas as principais técnicas que existem.

2.1.1.1 Operações globais

As operações globais são realizadas aplicando matrizes que transformam a totalidade da imagem. Os dois tipos de operações principais são as transformações geométricas ou cálculo a partir de histograma.

Transformações geométricas

- **Transformação afim** (2.1), a transformação afim é definida de acordo com a equação matricial seguinte:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

em que x e y são as coordenadas da imagem original, x' e y' são as coordenadas da imagem transformada e as variáveis a a f são os parâmetros da transformação. Dependendo dos parâmetros escolhidos, é possível efetuar transformações rígidas como translações, rotações em torno de um *pixel*, mudança de escala em volta de um *pixel* invariante e transformações de perspetiva;

- **Transformação quadrática** (2.2), a transformação quadrática é dada pela seguinte equação:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{bmatrix} \begin{bmatrix} x & y & 1 & x^2 & xy & y^2 \end{bmatrix}^T \tag{2.2}$$

onde x e y são as coordenadas da imagem original, x' e y' são as coordenadas da imagem transformada e as variáveis a_{11} a a_{26} são os parâmetros da transformação. Esta transformação permite distorções quadráticas na imagem em relação a um *pixel*.

Devido ao facto de o uso destas matrizes de transformação resultarem em coordenadas com valor fracionário, o que contradiz o mapeamento discreto de uma imagem, recorre-se à transformada inversa ilustrada na figura 2.1 (retirada de [8]), obtendo-se o valor dos *pixeis* a partir de interpolação do brilho dos *pixeis* vizinhos. Alguns dos métodos mais conhecidos são:

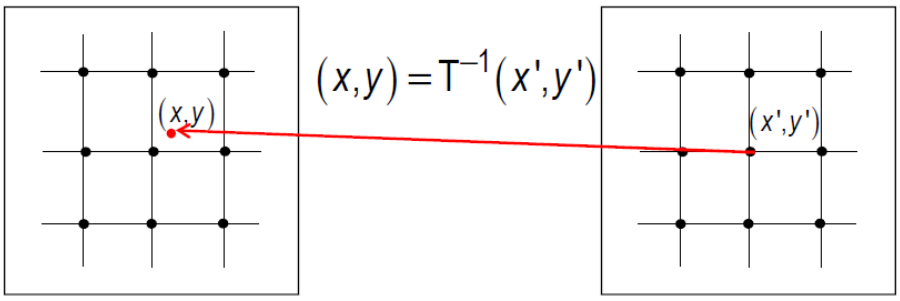


Figura 2.1: Método da transformação inversa

- **Vizinho mais próximo** (2.3), esta transformação é dada pela seguinte equação:

$$I\left(x',y'\right) = I\left(round\left(x\right), round\left(y\right)\right) \tag{2.3}$$

onde $I\left(x,y\right)$ é o nível de brilho do *pixel* na posição x,y e $round\left(x\right)$ o arredondamento de valor de x (podendo ser visualizado na figura 2.2, retirada de [8]);

- **Bilinear** (2.4), esta transformação é dada pela seguinte equação:

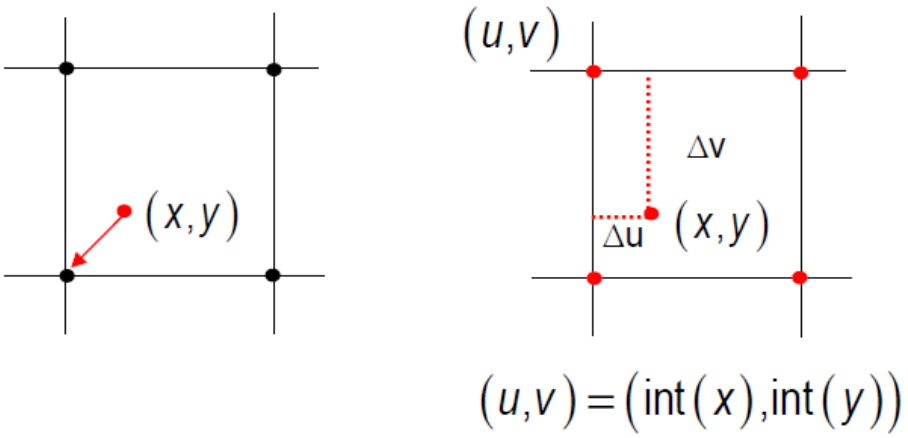
$$I(x',y') = (1-\Delta u)(1-\Delta v)I(u,v) + (\Delta u)(1-\Delta v)I(u+1,v) + (1-\Delta u)(\Delta v)I(u,v+1) + (\Delta u)(\Delta v)I(u+1,v+1) \tag{2.4}$$

onde u e v são as coordenadas da imagem original, Δu e Δv são a diferença entre as coordenadas da imagem original e transformada as e $I(x,y)$ é o nível de brilho do *pixel* na posição x,y (mostrado na figura 2.2, retirada de [8]);

- **Bicúbica** (2.5), esta transformação é dada pela seguinte equação:

$$I(x',y') = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j \tag{2.5}$$

onde x e y são as coordenadas da imagem original, x' e y' são as coordenadas da imagem transformada, $I(x,y)$ é o nível de brilho do *pixel* na posição x,y, i é a posição da coordenada referente à vizinhança de x' e y' e a_{ij} é o peso da intensidade das coordenadas da vizinhança dependente da distância.



(a) Vizinho mais próximo (b) Bilinear

Figura 2.2: Ilustração de interpolações

Histograma O cálculo de histograma baseia-se no gráfico de distribuição de intensidades dos *pixeis* (normalmente dos níveis de cinzento). A partir deste é possível retirar indicadores como o nível global de intensidade, o contraste e a gama de cinzentos usada para além de informação estatística como média, desvio padrão, moda, *skew* (assimetria em torno da média) e entropia, que podem ser úteis para a segmentação da imagem.

2.1.1.2 Operações pontuais

Estas operações têm como objetivo o melhoramento do aspeto da imagem, efetuando correções de brilho, contraste e diferenças de iluminação. Algumas das principais operações são mostradas nesta secção.

Binarização A operação de binarização, calculada na equação 2.6:

$$T(g) = \begin{cases} 0 & g < t \\ K & g \geq t \end{cases} \tag{2.6}$$

onde t é o limiar escolhido, K é o valor máximo da gama da imagem (equivale a branco) e g é o valor de intensidade do *pixel*. Esta operação torna a escala de intensidades de nível de cinzento para preto e branco, sendo essa escolha definida por um limiar. A binarização é um passo que na generalidade dos casos é sempre realizado para retirar informação de características relevantes. Neste trabalho é usada para detetar o círculo negro presente em todos os alvos, bem como para deteção do contorno dos impactos [7, 9, 10, 11]. A binarização pode ser uma transformação multi-nível, aplicando vários limiares em vez de um único.

Negação A operação de negação é calculada com recurso à equação 2.7:

$$T(g) = K - g \tag{2.7}$$

Esta operação inverte os níveis de intensidade dos *pixeis*, sendo invertível (g e K têm o significado definido anteriormente).

Expansão linear de intensidades A expansão linear mapeia um intervalo de valores noutro intervalo escolhido, modificando assim a gama de intensidades da imagem original. A equação de uma expansão genérica é definida em 2.8:

$$g = \begin{cases} g_{min} & f < f_{min} \\ \frac{g_{max} - g_{min}}{f_{max} - f_{min}} (f - f_{min}) + g_{min} & f_{min} \leq f \leq f_{max} \\ g_{max} & f > f_{max} \end{cases} \tag{2.8}$$

em que f é o valor de intensidade do *pixel*, f_{min} e f_{max} são os valores máximo e mínimo da gama atual e g_{min} e g_{max} são os valores máximo e mínimo da nova gama.

Equalização de histograma Com a equalização de histograma [12] uniformiza-se o histograma da imagem original. A equação associada é a apresentada em 2.9:

$$T(i) = round\left(\frac{f_{max}}{p_{num}} hc(i)\right) \tag{2.9}$$

em que i é o valor do nível de intensidade, p_{num} é o número total de *pixeis* na imagem e $hc(i)$ é a função de histograma cumulativo até ao valor de intensidade i. Isto cria uma maior diferença entre os níveis de intensidade, oferecendo um melhor contraste na imagem transformada.

Tabelas de transformação As operações envolvendo o cálculo de operadores pontuais podem ser computacionalmente dispendiosas, demorando mais tempo do que o disponibilizado. O recurso a uma tabela de transformação (LUT), onde se encontram já pré-calculados os resultados das operações, reduz o tempo necessário de processamento, removendo o cálculo.

2.1.1.3 Operações locais

Estes operadores envolvem uma vizinhança definida à volta do *pixel* (denominada de máscara) em que se vai aplicar uma determinada operação. São utilizados para remoção de ruído, correção de perturbações na aquisição e realce ou deteção de estruturas como orlas, cantos, linhas, etc. Há três tipos de operadores: soma, diferença e de ordem.

Operadores de soma Os operadores de soma removem ruído e suavizam a imagem, eliminando variações acentuadas. Filtros de média aritmética e gaussiano são normalmente usados para a remoção de ruído, como visto em [9].

Operadores de ordem Os operadores de ordem ordenam os valores da vizinhança para escolher um deles segundo um critério definido. Exemplos desses operadores são os filtros de máximo, mínimo ou mediana, com o último usado em [7, 10, 13].

Operadores de diferença O uso dos operadores de diferença é principalmente para a deteção de orlas, sendo sensíveis a mudanças de intensidade na imagem. São muito usados na segmentação, com recurso às derivadas, como o operador de gradiente e laplaciano.

2.1.1.4 Segmentação

Como descrito em [14], a segmentação é a subdivisão de uma imagem nos seus constituintes (regiões com atributos semelhantes, também denominado de objetos), sendo a definição desses constituintes dependente da aplicação. Os métodos de segmentação podem ser divididos em segmentação por semelhança (baseada em características) ou por descontinuidade (baseada em imagem). Esta segmentação habitualmente não é perfeita, sendo necessário recorrer a um algoritmo de ligação de orlas.

2.1.1.5 Segmentação por semelhança

Para a segmentação por semelhança existem vários algoritmos de acordo com as diferentes aplicações (evidenciado na figura 2.3, retirada de [15]). Apenas os mais relevantes vão ser tratados de seguida.

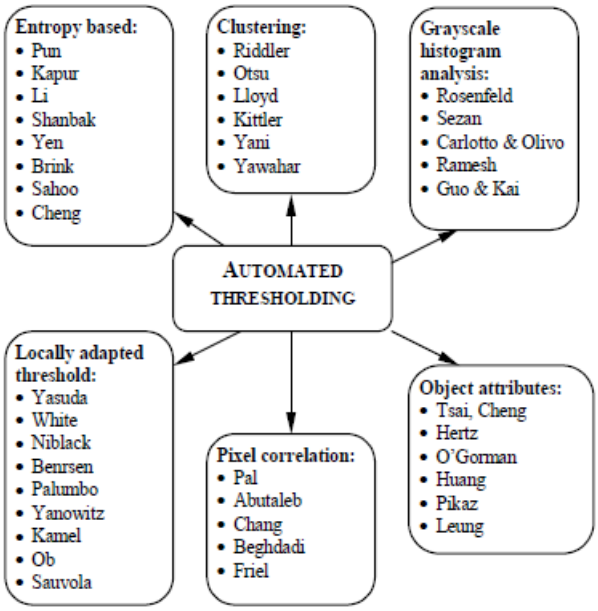


Figura 2.3: Alguns métodos de segmentação por semelhança

Método de Otsu O método de Otsu [16] tenta encontrar um limiar ótimo k^* para a binarização (ver 2.1.1.2) a partir da maximização da variância inter-classe, sendo essas classes dependentes do histograma de intensidades. O valor de k^* é calculado de acordo com a equação 2.10:

$$k^* = argmax\left\{\sigma_B^2(k)\right\} \tag{2.10}$$

A fórmula de cálculo da variância inter-classe é dada pela equação 2.11:

$$\sigma_B^2(k) = \frac{[\mu_T\,\omega(k) - \mu(k)]^2}{\omega(k)\,[1-\omega(k)]} \tag{2.11}$$

onde k é o nível de intensidade, $\sigma_B^2(k)$ é a variância inter-classe, μ_T é a média de intensidades da imagem e $\omega(k)$ e $\mu(k)$ são, respetivamente, os momentos cumulativos do histograma de ordem zero e um.

Limiar dinâmico Uma possível abordagem para evitar ruído como luminosidade disforme é utilizar um cálculo de limiar ao nível da vizinhança do *pixel* ou da subdivisão da imagem em várias de tamanho reduzido, aplicando um limiar para cada sub-imagem. Existem muitos algoritmos disponíveis [17], como, por exemplo, o de Bernsen e Niblack.

2.1.1.6 Segmentação por descontinuidade

Esta segmentação utiliza propriedades como a variação de intensidades das orlas e dos cantos das imagens para identificar os objetos relevantes. De seguida são descritas as técnicas mais relevantes.

Deteção de orlas por gradiente Esta deteção usa o princípio descrito em 2.1.1.3 combinado com o de 2.1.1.2, realizando assim um acentuamento de orlas para depois efetuar uma binarização, separando-as do resto da informação da imagem.

Método de Marr-Hildreth Este método baseia-se em derivadas, usando o facto que uma variação abrupta de brilho dá origem a uma resposta intensa da primeira derivada e uma variação do sinal (passagem por zero ou *zero crossing*) da segunda derivada. Ele recorre ao operador LoG (figura 2.4 retirada de [18]), que é a junção de um operador gaussiano com um laplaciano.

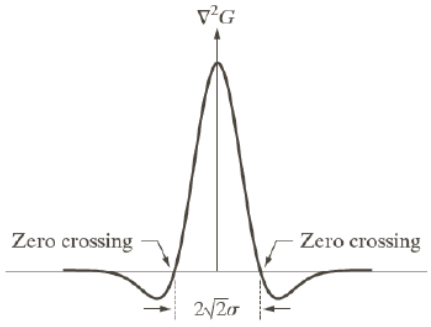


Figura 2.4: Operador LoG

Método de Canny O método de Canny tem como fundamento que a melhor deteção de orlas é com recurso à primeira derivada de um filtro gaussiano. As etapas do método, descritas em [18], são as seguintes:

1. Suavização da imagem com um filtro gaussiano;
2. Realce de orlas usando um filtro de gradiente (amplitude e ângulo);
3. Aplicar supressão de não-máximos (na direção do gradiente) à imagem de amplitude do gradiente;
4. Usar "duplo-limiar" e análise de conetividade para detetar e ligar as orlas.

2.1.1.7 Ligação de orlas

Os métodos de deteção de orlas costumam formar contornos algo fragmentados, não sendo muito úteis para a extração de informação. Como tal, estes algoritmos reparam estas imperfeições, obtendo assim as orlas ligadas.

Transformada de Hough A transformada de Hough [19, 20] procura curvas paramétricas que ocorrem na imagem, recorrendo à transformação de cada *pixel* na imagem num espaço de parâmetros (ou espaço de Hough). Um ponto no espaço de Hough que possua um valor elevado de

contagens, ou seja, que possua várias intersecções de curvas evidencia que essa equação paramétrica tem grande probabilidade de existir na imagem.

Esta transformada é interessante porque uma das curvas possíveis é a da equação referente ao círculo ($x_c^2 + y_c^2 = r^2$), que é bastante útil na deteção dos círculos do alvo, retirando informação do centro do círculo e do raio.

2.2 Ferramentas algébricas

2.2.1 Método de mínimos quadrados

O método dos mínimos quadrados (MMQ) refere-se a uma família de algoritmos de regressão para sistemas "sobre-determinados"(onde as equações conhecidas excedem o número de variáveis). Estes métodos procuram minimizar a soma dos quadrados dos erros resultantes da solução aproximada de cada equação.

Na sua aplicação ao ajuste de círculos e elipses [21], minimiza-se o quadrado das distâncias a um conjunto específico de pontos de interesse. Lembrando que tanto círculos como elipses podem ser representados algebricamente na forma de $F(x) = 0$ e posteriormente parametrizadas, a aplicação do método segue de forma imediata. O sistema resultante, não-linear, é resolvido iterativamente, abordando sucessivos problemas de resolução de mínimos quadrados lineares.

2.2.2 Estimadores-M

Estimadores-M são uma classe extensa de estimadores obtidos como o mínimo da soma das funções da variável ou variáveis relevantes [22].

Estes estimadores podem ser definidos como zeros de uma função estimadora, função esta frequentemente uma derivada de uma outra função estatística (a título de exemplo, esta situação verifica-se com os estimadores de máxima verosimilhança, um dos estimadores de tipo M mais comuns, que representam pontos críticos da função derivada duma função de verosimilhança). Em termos práticos, estes estimadores são utilizados para estimar características frequentes de uma população.

É possível que não exista uma solução fechada do problema de minimização apresentado na determinação de um estimador-M; na prática, computacionalmente, recorre-se frequentemente a soluções iterativas para aproximar esse valor.

2.3 Sistemas de deteção de impactos em tiro ao alvo

Nesta secção apresentam-se os principais sistemas de deteção de tiro ao alvo disponíveis no mercado bem como o *software* de pontuação normalmente utilizado.

2.3.1 Breve descrição

Um sistema de deteção de tiro ao alvo tem como função principal informar sobre a pontuação que um determinado atleta obteve no alvo. Tais sistemas podem dividir-se em dois tipos: os que processam a informação em tempo real, ou seja, que informam quase imediatamente o valor dos impactos e os que classificam os impactos à *posteriori*, ficando a informação disponível apenas no fim da prova.

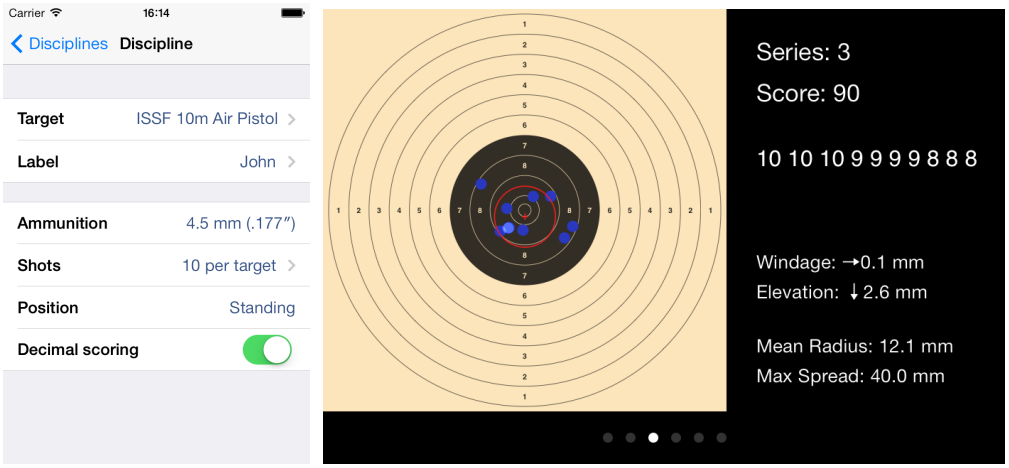
A identificação, em tempo real, dos impactos no alvo normalmente é feita com recurso a triangulação acústica [10], tendo muito recentemente surgido, no mercado, alguns sistemas que utilizam varrimento laser [23]. A identificação à *posteriori* dos valores dos impactos usa normalmente um sistema digitalizador [24, 25], câmara fotográfica ou vídeo [7, 9].

2.3.2 Sistemas comerciais para classificação à *posteriori*

2.3.2.1 TargetScan

O TargetScan, criado por Thomas Gabrowski [26], é uma aplicação para iPhone/iPad que utiliza a câmara integrada para classificar vários tipos de alvos. Para além de pontuar, guarda registos das sessões de tiro realizadas e monitoriza o progresso do atirador com o cálculo da elevação da mira, média geométrica dos disparos e dispersão dos disparos.

A figura 2.5 ilustra as funcionalidades descritas. Para adicionar mais alvos é necessário pagar um custo extra [27].



(a) Exemplo da interface (b) Pontuação do alvo

Figura 2.5: *Software* TargetScan

2.3.2.2 EasyScore 220

Este sistema, criado pela RIKA Sport GmbH & Co. KG [24], é um conjunto de digitalizador e *software* para deteção de furos (LISA [28]), conseguindo detetar alvos oficiais da ISSF de tamanho

até 220 mm, sendo necessário usar um programa separado para alvos de 25 e 50 metros. O sistema de guia para o digitalizador consegue arranjar o alvo de maneira a fixar a posição para facilitar a aquisição de imagem. Informa o utilizador da pontuação obtida e do alvo com melhor pontuação da sessão. Este sistema é exemplificado na figura 2.6.

Figura 2.6: Sistema EasyScore 220

2.3.2.3 RM III/IV

Os sistemas RM III Universal e RM IV desenvolvidos pela DISAG GmbH & Co KG [25], representado na figura 2.7, são digitalizadores que utilizam o software Match Manager da ABVisie [29] para classificação de alvos. Não possui algoritmo de processamento de imagem, sendo feita a localização de furos pelo utilizador no computador. Possui um visor LCD com botões e comunica com o PC a partir de uma interface RS-232.

Figura 2.7: Sistema RM IV

2.3.2.4 Orion Scoring System

O Orion Scoring System da Orion [30] é um conjunto de ferramentas que engloba algoritmos de deteção de impactos e pontuação, gestão de sessões de tiro e histórico de resultados. É compatível com digitalizadores à venda no mercado e corre em computadores com o sistema operativo Windows. Existem duas versões disponíveis, *Orion for Clubs* e *Orion at Home*, sendo a primeira uma versão mais simplificada para uso pessoal. O conjunto de ferramentas disponíveis por este sistema é representado na figura 2.8.

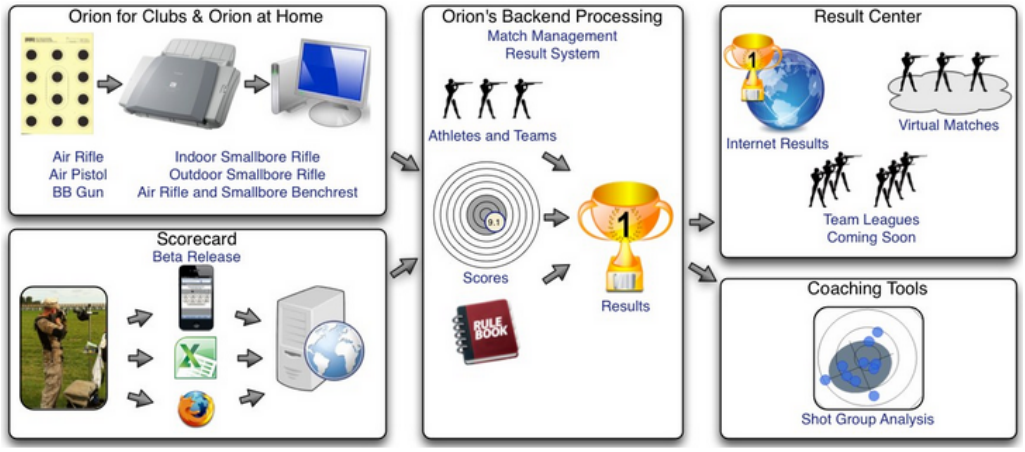


Figura 2.8: Ilustração das funcionalidades do *software* Orion Scoring System

2.3.3 Sistemas comerciais para classificação em tempo real

2.3.3.1 Sistemas Sius

A empresa Sius AG [31] é uma empresa suíça especializada na produção de sistemas de deteção de tiro ao alvo eletrónicos. É uma das poucas empresas que tem os seus sistemas aprovados pela ISSF [4]. Todos estes sistemas necessitam de uma unidade de controlo SA para o processamento da informação. Vai ser apresentado os diferentes modelos que são disponibilizados:

Série S Os sistemas da série S, exemplificado na figura 2.9a, utilizam triangulação acústica, com recurso a microfones colocados no suporte do alvo, detetando a posição do impactos no alvo. Para assegurar a eficiência do sistema, recomenda-se a instalação de placas entre os espaços dos atiradores e microfones no local do atleta, identificando assim a origem do tiro para o sistema.

Série LS Os sistemas da série LS, como mostrado na figura 2.9b, em vez de usarem a deteção baseada em som, usam sensores fotoelétricos em conjunção com feixes infravermelho para detetar o local de impacto no alvo.

Série H Esta série de sistemas utilizam em conjunto as tecnologias da série S com as da LS. É criado assim um sistema mais preciso, com resolução até às centésimas de milímetro. Um exemplar destes sistemas pode ser visto na figura 2.9c.

Série RT Os sistemas RT são sistemas de treino e pontuação de alvos feito especificamente para alvos móveis, representado na figura 2.9d. Possibilitam o atirador treinar individualmente a partir de um sensor fotoelétrico que deteta se a arma é levantada, retirando a necessidade de carregar em botões. A deteção também é feita por barramentos laser. Possui um painel para o controlo de velocidade do motor que movimenta o alvo.

(a) Sistema S110 (b) Sistema LS10 (c) HS10

(d) Sistema RT10

Figura 2.9: Exemplos dos diferentes sistemas vendidos pela Sius

2.3.3.2 ESA

Criado pela Häring, o ESA é um sistema de treino e pontuação eletrónico de alvos [32], instalável em carreiras de tiro ou para uso pessoal, como pode ser visto na figura 2.10. Existe a escolha de entre dois modelos (ESA10 e ESA50) que podem ser utilizados em alvos de 10 m, 15 m, 25 m, 50 m, 100 m e 300 m.

Figura 2.10: Sistema ESA10

As características que possui são a indicação do impacto detalhada através de um ecrã LCD, um procedimento de medição por triangulação acústica com a resolução 1/1000 mm e é possível

combinar vários sistemas de ESA para criar uma rede através de um servidor. O *software* de controle funciona apenas para sistemas operativos Windows.

2.3.3.3 ML2000 Scoring System

O ML2000, vendido pela Megalink [33], é um sistema de classificação de alvos e gestão de competições. É modular, podendo-se adicionar ou remover componentes, dependendo das necessidades do utilizador. Um exemplo de instalação deste sistema é mostrado na figura 2.11.

Possui monitores para a visualização do alvo para o atirador e diferentes versões do *software* a instalar, como o MLRange para controlo de competições a decorrer, o MLView para mostrar aos atiradores variadas informações sobre parâmetros da competição e o MLShoot para o treino individual.

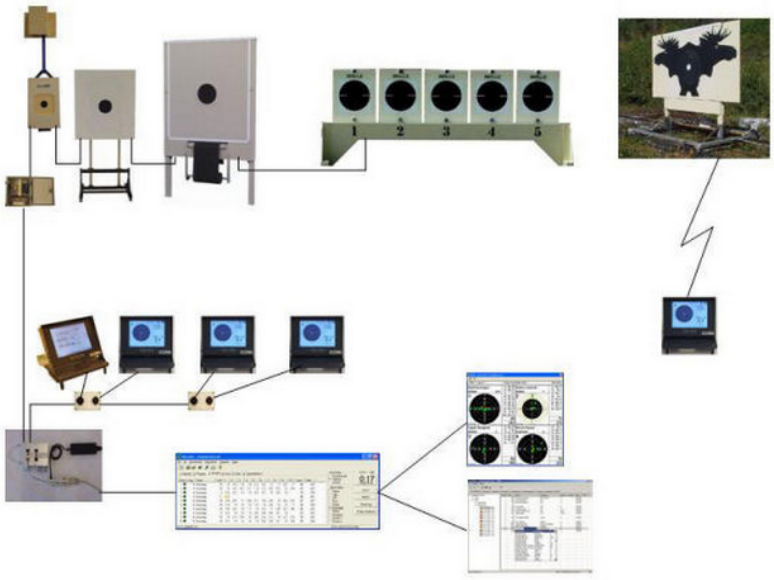


Figura 2.11: Exemplo de uma configuração possível do sistema ML2000

2.3.3.4 BLACK MAGIC

O sistema BLACK MAGIC, da Meyton [34], é um sistema de medição da posição dos furos de alvos. É compatível com alvos de pistola de ar e rápida a 10 m, alvo móvel a 10 m e carabina a 50 m. Existe a escolha de dois modelos, o BLACK MAGIC original ou o BLACK MAGIC XL, que é uma versão melhorada que possibilita mais tipos de alvos por possuir uma caixa de maiores dimensões. O sistema está representado na figura 2.12.

Este produto utiliza barreiras de LEDs para detetar a localização do furo, possibilitando também saber a velocidade a que a bala trespassou o alvo. Possui um PC de placa única, a MF5R1, com Linux embarcado no sistema, que efetua os cálculos das variáveis de interesse, enviando depois por RS232 ou Ethernet para um monitor.

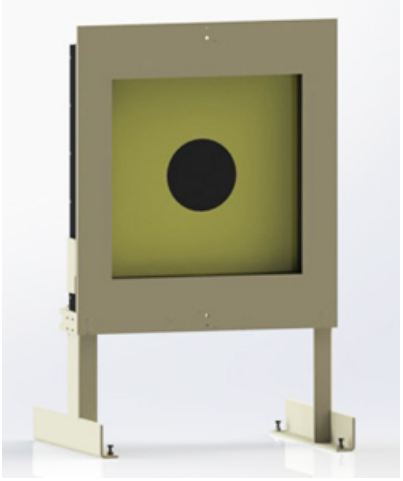


Figura 2.12: Sistema BLACK MAGIC

2.3.4 Sistemas académicos

Alguns trabalhos académicos nesta área foram já realizados e descritos em artigos científicos. O sistema descrito em [13] utiliza uma câmara com posterior processamento de imagem recorrendo a binarização por limiar de histerese. O limiar de histerese utiliza um limiar superior e um inferior para binarizar os *pixeis* da imagem. A escolha do valor dos que possuem um nível de intensidade entre os dois limiares é efetuada a partir do seguinte algoritmo: esses *pixeis* ficam com o valor 1 se esse *pixel* estiver ligado a qualquer pixel com valor de intensidade maior que o limiar superior, senão é 0.

Já o sistema apresentado em [35] utiliza a componente vermelha da representação RGB para efetuar uma segmentação sem grande interferência de ruído e computacionalmente rápida.

2.4 Bibliotecas de *software*

2.4.1 OpenCV

O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de *software* de uso livre para visão por computador e *machine learning* [36]. Desenvolvido pela Intel Russia e actualmente suportado pela Itseez e a Willow's Garage, foi construída para fornecer uma infraestrutura comum para aplicações de processamento de imagem em tempo real e visão por computador.

Esta biblioteca, escrita inicialmente em C, possui agora funções em C, C++ e Python, incluindo também *wrappers* para linguagens como Java, Ruby e MATLAB, entre outras. O OpenCV corre praticamente em todos os sistemas operativos, tais como, por exemplo, Windows, Linux, Android, entre outros.

Para além de ser uma biblioteca poderosa em termos do tratamento de imagem, possui adicionalmente a capacidade de tirar partido de sistemas *multi-core*, recorrendo à utilização do OpenCL

(*Open Computing Language*), o qual lhe oferece a capacidade usar várias partes do *hardware* do PC (CPU, GPU, etc.) de forma a acelerar o processamento de imagem.

2.4.2 MATLAB

O MATLAB (MATrix LABoratory), desenvolvido pela MathWorks, é um ambiente de computação numérica [37]. É proprietário, sendo necessário pagar uma licença para o utilizar.

O programa permite manipulações de matrizes, desenho de gráficos de funções e dados, implementação de algoritmos, criação de interfaces e programas escritos em outras línguas, como C, C++, Java, Python e Fortran.

Possui uma *toolbox* (biblioteca de funções) de processamento de imagem [38] que fornece vários métodos e algoritmos de análise, segmentação e melhoramento de imagem. Boa parte das funções suportam processadores *multi-core* e GPUs.

2.4.3 Armadillo

O Armadillo é uma biblioteca de álgebra linear de alta qualidade, visando atingir um bom equilíbrio entre velocidade e facilidade de uso [39]; a sua sintaxe é deliberadamente semelhante à utilizada no programa MATLAB.

Escrita em C++, é principalmente desenvolvida no NICTA (National Information and Communications Technology Australia) por Conrad Sanderson, com contribuições de todo o mundo. É *opensource*, distribuído sob uma licença livre tanto em contextos proprietários como para projetos individuais.

Útil para o desenvolvimento de algoritmos de *machine learning*, reconhecimento de padrões, processamento de sinal, estatística, etc, fornece classes eficientes para vetores, matrizes e cubos. Suporta variáveis do tipo inteiro, ponto flutuante e números complexos, bem como um subconjunto de funções trigonométricas e de estatísticas.

Várias decomposições matriciais são fornecidos através da integração com LAPACK (Linear Algebra PACKage), ou um dos seus substitutos (como a multi-threaded Intel MKL, ou AMD ACML, ou bibliotecas OpenBLAS).

2.5 Discussão

Os sistemas existentes no mercado apresentam soluções bastante flexíveis e com tecnologia eficiente, mas possuem preço muito elevado e instalação pouco prática. A partir dos exemplos vistos em sistemas como [13] e [35], pode-se criar uma solução mais rápida e genérica, adaptando alguns algoritmos para os requisitos definidos no capítulo 1.

Capítulo 3

Descrição do *hardware* do sistema

Neste capítulo apresenta-se o *hardware* utilizado neste projeto. O capítulo inicia com uma breve descrição dos alvos utilizados nas competições de tiro, bem como o método de pontuação, iluminação e a resolução necessária. Depois de definida a informação fornecida pelo problema, é exposto o material escolhido.

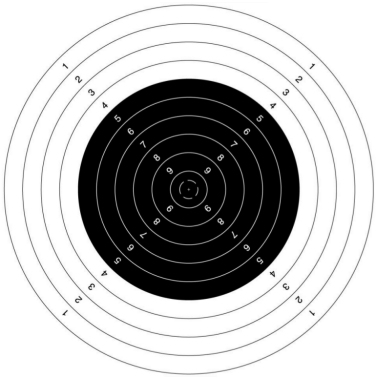
3.1 Alvo

3.1.1 Constituição do alvo

Em competições ISSF de tiro ao alvo, são usados alvos compostos por circunferências concêntricas de cor preta, com regiões pontuáveis de 1 a 10, e nas finais, de 1.0 a 10.9. Estes alvos são nas versões não eletrónicas feitos de papel, de acordo com normas definidas pela ISSF. Uma lista dos alvos oficiais existentes é apresentada de seguida.

- Alvo para carabina a 300 m;
- Alvo para carabina a 50 m;
- Alvo para carabina de ar comprimido a 10 m;
- Alvo para pistola de velocidade a 25 m;
- Alvo de precisão para pistola a 25 m e 50 m;
- Alvo para pistola de ar comprimido a 10 m;
- Alvo móvel a 50 m;
- Alvo móvel a 10 m.

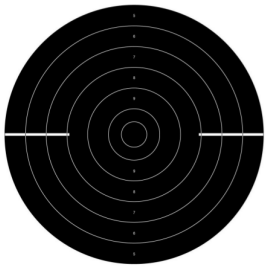
A figura 3.1 ilustra o exposto anteriormente, retiradas de [40].



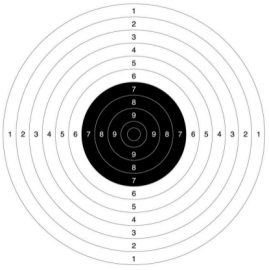
(a) Alvo para carabina a 300 m

(b) Alvo para carabina a 50 m

(c) Alvo para carabina de ar comprimido a 10 m

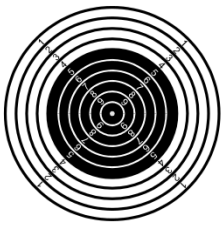


(d) Alvo para pistola de velocidade a 25 m

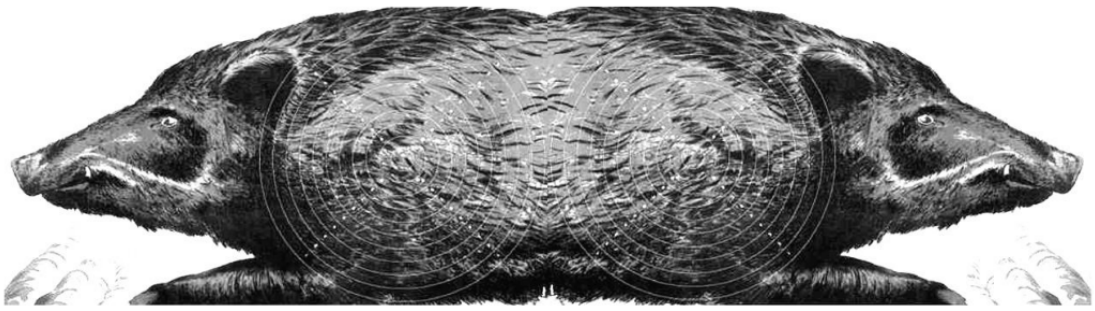


(e) Alvo de precisão para pistola a 25 m e 50 m

(f) Alvo para pistola de ar comprimido a 10 m



(g) Alvo móvel a 10 m



(h) Alvo móvel a 50 m

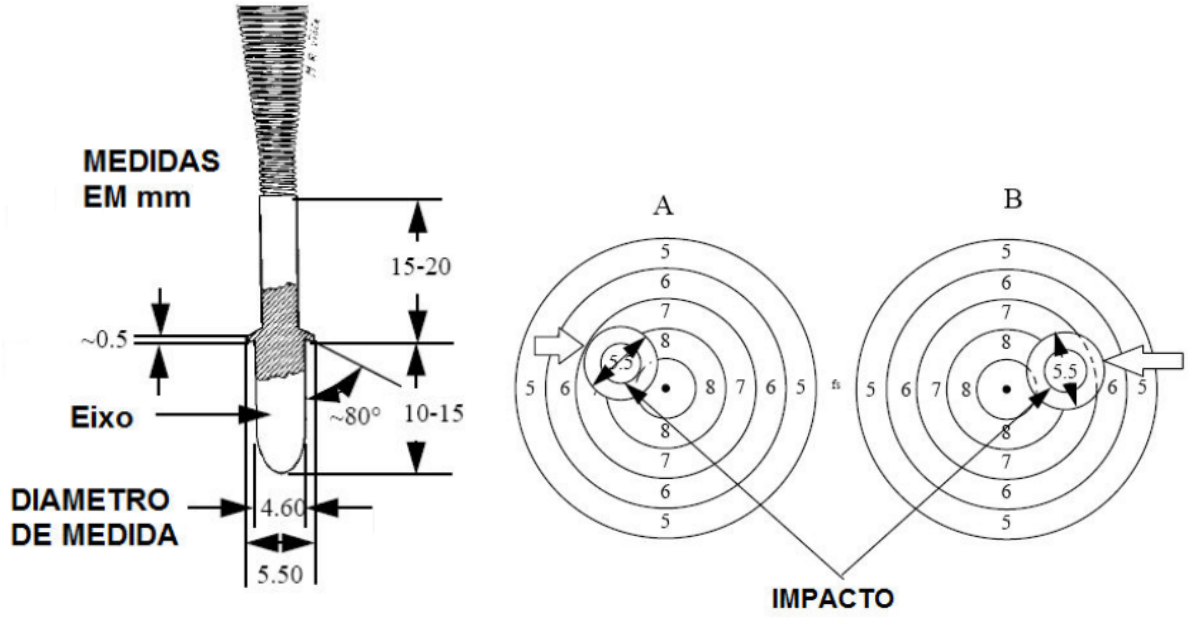
Figura 3.1: Representação dos diferente tipos de alvos das competições da ISSF

As dimensões dos anéis dos alvos são apresentadas na tabela A.1 do anexo A. Neste trabalho apenas se utilizaram os alvos de pistola e carabina de ar comprimido a 10 m e os alvos de precisão para pistola a 25 m.

3.1.2 Pontuação do alvo

A pontuação de alvos segue regras específicas, bem definidas pela ISSF [40]. Todos os impactos são classificados de acordo com o valor mais alto da zona ou anel em que o alvo for atingido.

Se qualquer parte do anel (linha de demarcação entre duas zonas) for tocada pelo projétil, creditar-se-á o valor mais alto das duas zonas. Tal valor é verificado pela simples observação do orifício provocado pelo projétil ou pela utilização de um calibrador (exemplo na figura 3.2a) que, depois de inserido nesse orifício, toque no bordo exterior do anel.



(a) Calibrador de alvo de carabina a 10m (b) Diferentes situações da classificação dum alvo de carabina a 10m

Figura 3.2: Ferramenta e classificação de casos duvidosos de pontuação de alvos

Como exemplo de classificação, o caso A da figura 3.2b representa um impacto de valor duvidoso. O calibrador exterior mostra que o bordo exterior da flange se situa dentro do círculo do 7, fazendo assim o tiro ser classificado com 9. Em contrapartida, o caso B da figura 3.2b também representa um impacto de valor duvidoso. O calibrador exterior mostra que o bordo da flange se situa para além do círculo do 7 e dentro do 6, logo, o tiro é classificado com 8. Nas finais dos torneios de tiro ao alvo, os alvos são pontuados às décimas, o que implica a utilização dos alvos eletrónicos.

3.1.3 Iluminação do alvo

Nas carreiras de tiro interior, é necessário garantir um mínimo de luminosidade para não haver problemas de visualização do alvo para o atirador. Os limites estipulados pela ISSF estão apresentados na secção 6.4.14 de [41], sendo o mínimo garantido para os alvos 1000 lux. A carreira é iluminada artificialmente como representado na figura 3.3, assegurando que não haja reflexos e sombras nos alvos. O plano de fundo dos alvos possui cor baça, suave e neutra.

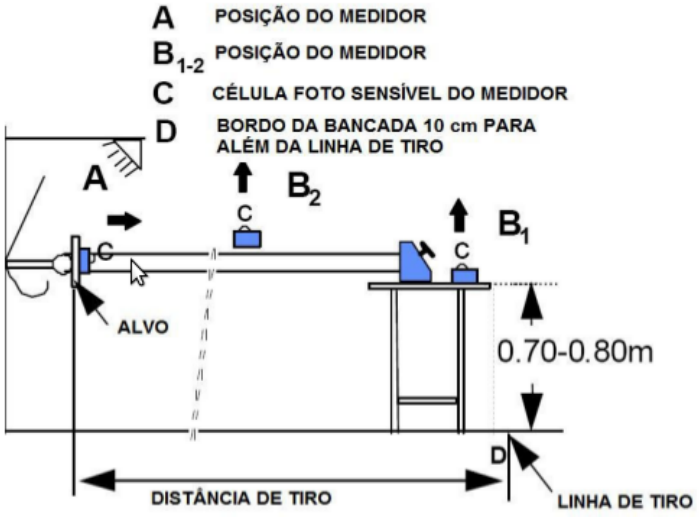


Figura 3.3: *Setup* da iluminação do suporte do alvo

3.1.4 Resolução

O sistema de aquisição de imagem tem de ser escolhido de maneira a salvaguardar que toda a informação retirada seja útil. A resolução da câmara é um parâmetro que tem que ser bem escolhido, de modo a garantir uma aquisição que permita uma correta pontuação. Os cálculos a seguir apresentados admitem que o alvo se encontra completamente inscrito na imagem, com a maior área possível a ser ocupada.

Inicialmente, propôs-se a deteção de todas as linhas dos anéis 1 a 10 do alvo. Assim, é essencial que a espessura da linha de cada anel da imagem seja pelo menos de 1 pixel. O cálculo da resolução associada a tal proposta é efetuado segundo a equação 3.1:

$$res = \frac{d_{alvo}}{e_{anel}} \times 2 \tag{3.1}$$

onde res é a resolução mínima necessária, d_{alvo} é o tamanho do cartão do alvo em mm e e_{anel} é a espessura das linhas dos anéis, também em mm. Para cumprir o teorema de Nyquist, o termo constante é necessário estar presente na equação. A partir da tabela A.1 do anexo A que apresenta a informação das dimensões dos alvos, construiu-se a tabela 3.1 que mostra os resultados de resolução obtidos para cada alvo.

A coluna "Resolução"da tabela 3.1 mostra que o pior caso é 5500 P, já que são necessários, no mínimo, 5500 pixeis num dos lados, o que resulta numa imagem com 30 MP (isto supondo que a janela é quadrada, o que não apresenta ser a normalidade dos casos e que leva a resoluções maiores). Câmaras com esta características são de valor monetário muito elevado, sendo habitualmente desenvolvidas para aplicações específicas.

Assim, e dado que é apenas necessário conseguir pontuar à décima optou-se por dividir o raio do círculo de valor um do alvo em 110 partes. Deste modo, conseguiremos pontuar desde 1.0 até

10.9. Para calcular o centro do alvo, o qual é necessário para calcular as distâncias dos impactos ao centro, é suficiente detetar apenas a área escura do alvo.

Tabela 3.1: Parâmetros dos alvos em mm e respetivas resoluções mínimas

Alvo	Tamanho do cartão	Espessura da linha		Resolução (eq. 3.1)
		Máxima	Mínima	
Carabina 300 m	1300x1300	0.5	1.0	5200.0
Carabina 50 m	250x250	0.2	0.3	2500.0
Carabina de ar 10 m	80x80	0.1	0.2	**5500.0**
Pistola precisão 50 m e 25 m	550x550 ou 550x530	0.2	0.5	2500.0
Pistola de ar 10 m	170x170	0.1	0.2	3400.0
Pistola de velocidade 25 m	550x550 ou 550x530	0.5	1.0	2200.0

O cálculo para a resolução mínima, para este caso, com a suposição de o alvo ocupar a máxima área possível da imagem sem estar cortado, é de 440 P. Então, a escolha da câmara com um valor de resolução deste calibre fica mais acessível. Caso se queira colocar a câmara a uma certa distância do alvo, a resolução necessária vai aumentar.

3.2 Material escolhido

Tendo em conta todos as restrições referidas na secção anterior, foi realizada uma escolha de material para a construção do protótipo.

3.2.1 Computador

Para o desenvolvimento do código e realização dos testes, o PC utilizado foi o Insys Gameforce, com um processador Intel Pentium Dual Core T4200 – 2,00 GHz, uma placa gráfica nVidia GeForce G105M com 512MB memória dedicada e com memória 8 GB (4GB RAM+4GB MMC Turbo Boost).

A escolha para a construção do protótipo tem de ser cuidadosa, pois tem ser economicamente competitiva e possuir as especificações que cumpram as restrições pedidas. Escolheu-se um computador de placa única, a Raspberry Pi, modelo 2 [42]. Possui um CPU *quad-core* ARM Cortex-A7 de 900MHz, 1 GB de RAM e um *core* de gráficos 3D VideoCore IV. A figura 3.4 mostra o componente descrito. Inclui também quatro portas USB, quarenta pinos GPIO, porta HDMI, Ethernet e entrada de áudio de 3,5 mm e para cartões Micro SD e interface para a câmara e ecrã proprietário. Por causa de possuir o processador ARMv7, pode ter como sistema operativo todas

as distribuições GNU/Linux de ARM, para além de Windows 10, o que fornece a compatibilidade com muito *software* já existente.

Estas características asseguram que a identificação e pontuação seja cumprida dentro do tempo esperado, pois sendo uma solução *off the shelf*, proporciona um bom compromisso entre o esforço a desenvolver a aplicação, a construção do *hardware* e o tempo para tal necessário. Também o preço é baixo em relação à competição, possuindo o valor de 35$ até à data deste documento [43].

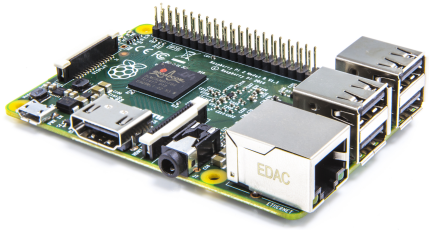


Figura 3.4: Raspberry Pi Modelo 2

3.2.2 Câmara

Como a escolha do PC é uma Raspberry Pi, para adquirir a imagem decidiu-se utilizar o módulo câmara da Raspberry Pi [44], ilustrada na figura 3.5.

A placa possui dimensões pequenas, em torno de $25 \times 20 \times 9mm^3$, e pesa $3g$. A ligação com a Raspberry Pi é feita por meio de um cabo de fita curta. A câmara está ligada ao processador BCM2835 na RPi através do barramento CSI, que proporciona maior largura de banda no transporte dos dados de pixel da câmara para o processador. O sensor em si tem uma resolução nativa de 5 megapixels, e tem uma lente de foco fixo a bordo. Em termos de imagens fixas, a câmara é capaz de imagens estáticas 2592×1944 pixels, e também suporta 1080P30, 720P60 e 640x480P60.

Esta construção centrada desta câmara para a RPi possibilita uma transmissão rápida do ficheiro de imagem, diminuindo o tempo total do processo. Também possui um preço menor que as câmaras digitais vendidas do mercado, comparável às *webcams* disponíveis [45].

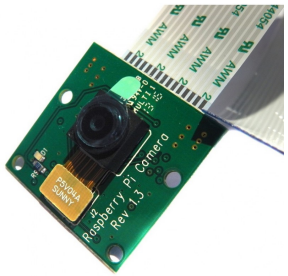


Figura 3.5: Módulo câmara da Raspberry Pi

3.3 Discussão

O estudo das características do alvo, das regras da pontuação e do seu condicionamento na carreira de tiro proporcionou ao levantamento de toda a informação útil do problema, para depois facilitar o desenvolvimento de algoritmos para os processos de deteção e pontuação do alvo.

As condicionantes definidas no capítulo 1 levaram à escolha da Raspberry Pi 2 e o módulo da câmara como os componentes a constituírem o protótipo. É de notar que provavelmente existem outras opções que também conseguem cumprir os requisitos pedidos, mas na altura da realização do projeto, estas escolhas provaram ser promissoras.

Capítulo 4

Proposta de *software*

Uma descrição detalhada do *software* desenvolvido bem como as plataformas de desenvolvimento escolhidas para tal vão ser apresentados neste capítulo.

4.1 Ferramentas utilizadas

Para o desenvolvimento e aplicação dos diferentes algoritmos, usou-se o OpenCV, uma biblioteca de funções de processamento e análise de imagem. Podendo escolher-se entre Python e C++, decidiu-se como linguagem de programação a última principalmente por não ser uma linguagem interpretada, como o Python, o que conduz a menores tempos de execução. Todo o código do projeto foi desenvolvido recorrendo a um ambiente de desenvolvimento integrado. Para cálculo algébrico com matrizes, foi usada a biblioteca de funções Armadillo, sendo uma das opções mais eficientes disponíveis em C++, como descrito em [39].

A IDE escolhida foi o Qt Creator [46], desenvolvido pela Qt Company, subsidiária da Digia Plc. Este IDE funciona em Linux, OS X e Windows e oferece terminação inteligente de código, destaque de sintaxe, um sistema de ajuda integrado, integração de perfil, depurador e também integração para todos os principais sistemas de controlo de versões (git e Bazaar, por exemplo).

4.2 Visão global

A arquitetura global do *software* pode ser vista na figura 4.1. Detalhando as diversas partes da arquitetura proposta, obtemos:

- **Aquisição de imagem** — As informações necessárias para calcular a pontuação dos alvos de tiro são retiradas das imagens enviadas para o programa;
- **Processamento de imagem**:

  Definição da região de interesse (ROI) — A imagem recebida no programa contém alguma informação não importante para a obtenção de pontuação, sendo apenas relevante a área associada ao alvo;

Obtenção dos anéis dos alvos — Com a parte relevante da imagem encontrada, é possível calcular os anéis do alvo de maneira a obter o centro comum necessário para pontuar os disparos efetuados;

Obtenção dos das posições dos furos — Tendo a informação espacial do alvo, para conseguir pontuar é essencial ter a informação dos impactos criados pelos disparos do atirador;

- **Pontuação dos alvos** — Depois de retirada toda a informação relevante do alvo, reúnem-se as condições necessárias para pontuar o mesmo corretamente;
- **Visualização do resultado** — Com a pontuação, é criada uma imagem virtual para o utilizador com informação sobre a sua performance.

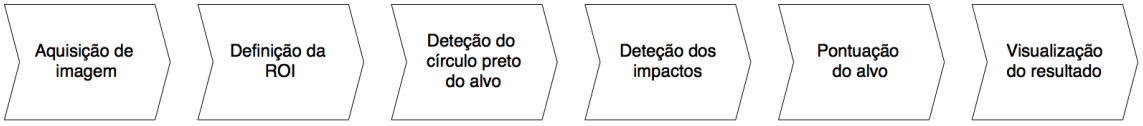


Figura 4.1: Visão geral do algoritmo a implementar

4.3 Aquisição de imagem

Para adquirir a imagem, existe uma variada escolha na tecnologia para essa propósito. Como tal é necessário escolher tanto a máquina a nível de características como ao nível de facilidade de implementação. Vão ser apresentados as duas rotinas experimentadas para cumprir esse objetivo.

4.3.1 Recurso a câmara digital

Numa fase inicial, testou-se a obtenção da imagem com recurso a uma câmara digital ligada por um cabo USB. O modelo utilizado para os testes foi uma Nikon Coolpix S3300, a qual possui uma resolução de 16 MP.

Para conseguir a comunicação entre a aplicação e a câmara, instalou-se a biblioteca gPhoto2 [47]. Esta biblioteca é *opensource*, usada em aplicações de software de câmara digital para sistemas Unix, suportando mais de 1800 câmaras. As versões mais recentes libgphoto2 também suportam o Media Transfer Protocol (MTP) que é baseado no protocolo de comunicação de câmaras Picture Transfer Protocol (PTP), criado pela Kodak.

As funções do gPhoto2 baseiam-se nos comandos de PTP, enviando-os diretamente para a câmara. O problema desta abordagem é que não existe garantias que as câmaras tenham esses comandos implementados, podendo ter um protocolo de comunicação proprietário. Na Coolpix, poucas funções de PTP estão presentes, usando apenas a de fotografar. A biblioteca funciona com instruções de linha de comandos (Bash). Com isto é possível comandar a câmara a partir de

código, chamando a função *system* de C++. A linha que executa a invocação da função de gPhoto2 é apresentada de seguida em 4.1.

Excerto 4.1: Invocação de função capturar imagem e descarregar

```
system("gphoto2 --capture-image-and-download");
```

Este trecho manda um comando para fotografar e descarregar a imagem no diretório raiz do programa. Apesar de funcionar relativamente bem, é preciso ter o cuidado de libertar a porta USB de outros processos do sistema operativo. Isto, aliado ao facto de a função *system* chamar a linha de comandos para esta execução, faz esta captura ser demorada, gastando cerca de cinco segundos desde a ordem até receber a imagem.

4.3.2 Recurso ao módulo da câmara da RPi

Devido aos problemas apresentados com o proposto em 4.3.1 e com escolha feita na subsecção 3.2.2, utilizou-se a câmara da RPi para adquirir a imagem. O controlo da câmara é feito com recurso à biblioteca Raspicam [48], uma adaptação das bibliotecas específicas de fotografia e vídeo da câmara da RPi (raspistill, raspivid e raspiyuv) para OpenCV.

A classe *RaspiCam_Cv* é usada para aceder à câmara, podendo depois capturar uma imagem e guardá-la com os métodos *grab()* e *retrieve()*.

Em comparação à solução apresentada em 4.3.1, o tempo de execução é muito menor, demorando apenas 770 *ms*. A integração com o OpenCV também facilita a importação da imagem, já tendo a informação acessível para ser usada sem ser necessário retirá-la de um ficheiro.

4.4 Processamento de imagem

As informações necessárias para calcular a pontuação dos alvos de tiro são retiradas das imagens enviadas para o programa. Seguidamente, nesta secção, vão ser descritas as técnicas de análise e processamento de imagem usadas para essa tarefa .

4.4.1 Definição da região de interesse (ROI)

A imagem recebida no programa contém alguma informação não relevante para a obtenção de pontuação, sendo apenas relevante a área associada ao círculo da zona 1 do alvo. Exemplos desta situação são exemplificados nas figuras 4.2a e 4.2b. É necessário então retirar a ROI, descartando o resto da imagem fornecida.

Uma abordagem testada foi a procura de circunferências com a transformada de Hough circular implementada em OpenCV, *HoughCircles*. Sabendo que é conhecido o tipo de alvo antes da execução do algoritmo e a resolução da imagem, é possível procurar por circunferências a partir de um tamanho mínimo definido (o qual garanta a deteção de pelo menos um dos anéis do alvo) até um tamanho máximo (o menor lado da imagem). Assim, podemos aplicar o algoritmo 1. Se for

encontrada alguma circunferência (no caso de várias escolhe-se a de maior raio para garantir que se engloba todo o alvo), recorta-se a imagem a partir do raio e centro da circunferência encontrada. Esta solução proposta possui alguns problemas. Na aquisição do ambiente tem que se garantir que não existem objetos ou contornos redondos, de maneira a não haver falsos positivos na deteção de circunferências. Também é necessário haver condições de iluminação minimamente uniformes, para garantir uma binarização correta. O tempo de execução médio deste troço de código para as figuras 4.2 é 3,8 *s*, o que apresenta ser uma solução custosa em termos temporais.

Para colmatar alguns dos problemas referidos, foi implementado um método diferente, recorrendo ao algoritmo 2. Em vez de se procurar circunferências com a transformada de Hough, utiliza-se a informação de cor, passando da representação RGB para a HSV. Como parte do alvo está a preto, é possível binarizar escolhendo limiares para cada componente da representação HSV. A cor preta normalmente apresenta na componente *Value* um valor baixo em comparação às cores restantes. A escolha desse limiar pelo método de Otsu revelou ser efetiva, separando a componente preta do resto da imagem. Os limiares das restantes componentes foram escolhidos empiricamente. Tendo a região de preto determinada, são retirados os contornos dos objetos presentes com a função *findContours*. A escolha do contorno da região preta do alvo baseia-se em dois critérios: em primeiro lugar, possuir uma área entre um valor máximo e mínimo, calculados pela medida do menor lado da imagem; caso cumpra essa condição, aproxima-se os pontos do contorno a uma elipse, na qual se compara a área da elipse com a do contorno. Se ela dista com uma variação de 0,5 %, o contorno é rejeitado. Depois de encontrado o contorno correto, recorta-se de maneira análoga ao código anterior. Este algoritmo, apesar de ser menos restrito nas condições de aquisição de imagem, continua a precisar da garantia que não existam círculos de cor preta no ambiente a ser fotografado.

(a) Foto dum alvo de carabina 10 m

(b) Foto dum alvo de pistola 10 m

Figura 4.2: Exemplos de fotografias de *setups* em casos reais

Comparando ambos os algoritmos, o algoritmo 2 tem um tempo de execução muito menor que 1, possuindo um tempo médio de 800 *ms*. Outra vantagem é de se garantir que a circunferência encontrada é sempre a mesma, independentemente da resolução da imagem, o que permite um escalamento estático e definido para todas as imagens no caso de 2. Nas figuras 4.3b e 4.3d podemos verificar o problema de escala mencionado, onde em 4.3b parte do alvo foi perdido devido a uma escolha errada do rácio de escalamento.

Algoritmo 1 Encontrar a região de interesse a partir de *HoughCircles*

função FINDROI_HOUGH

pré-processamento:
$img_{gray} \leftarrow$ converter *img* (em RGB) para escala de cinzentos
$mask_{Gauss} \leftarrow$ definir máscara quadrada gaussiana de 9x9
$img_{grayFilt} \leftarrow$ aplicar a máscara $mask_{Gauss}$ na imagem img_{gray}
$img_{edges} \leftarrow$ obter orlas de $img_{grayFilt}$ com detector de Canny

procura de circunferências:
para $i \leftarrow 0$ **até** 10 **execute**
Usar transformada de Hough circular com $minRadius + size * i$ e
$maxRadius + size * i$ em img_{edges}
se circunferência encontrada **então**
se $circle_{radius}$ maior que guardado **então**
$circle_{radius}, circle_{center} \leftarrow$ guardar raio e centro de *HoughCircles*
fim se
fim se
fim para
se nenhuma circunferência encontrada **então**:
sair
senão
continuar para *escolher região de interesse*
fim se

escolha da região de interesse:
$marginCorrector \leftarrow$ escolher margem de recorte dependendo do cartão escolhido e $circle_{radius}$
$ROI \leftarrow$ definir região de interesse com rectângulo definido por $circle_{center}$ e $marginCorrector$.
$img_{cropped} \leftarrow$ Selecionar apenas região de interesse em *img*
devolve $img_{cropped}$
fim função

Algoritmo 2 Encontrar a região de interesse a partir de HSV e deteção de contornos

função FINDROI_HSV

pré-processamento:
$img_{HSV} \leftarrow$ converter *img* (em RGB) para escala HSV
$img_V \leftarrow$ retirar componente *Value* de img_{HSV}
$threshold_V \leftarrow$ retirar limiar de binarização em img_V com o método de Otsu
$img_{BW} \leftarrow$ binarizar img_{HSV} com limiares pré-definidos para H, S e com $threshold_V$
$mask_{Open} \leftarrow$ definir máscara circular de 3x3
$img_{BWFilt} \leftarrow$ aplicar a operação de abertura na imagem img_{BW} com a máscara $mask_{Open}$

extração de contornos:
Usar função *findContours* em img_{BWFilt} e guardar contornos

filtragem de contornos:
minArea, *maxArea* $\leftarrow$ definir limites de escolha de área usando os lados da imagem
para $i \leftarrow 0$ **até** número de contornos total **execute**
se Área de contorno $>$ *minArea* **e** Área de ontorno $<$ *maxArea* **então**:
$ellipse \leftarrow$ Aproximação de contorno por uma elipse
se Área de $ellipse$ aproximado de $\pi \times ellipse_{height} \times ellipse_{width}$ **então**:
$ellipse_{center} \leftarrow$ guardar centro de $ellipse$
$ellipse_{radius} \leftarrow \frac{ellipse_{height}+ellipse_{width}}{4}$
fim se
fim se
fim para
se nenhum contorno escolhido **então**:
sair
senão
continuar para *escolha da região de interesse*
fim se

escolha da região de interesse:
$marginCorrector \leftarrow$ escolher margem de recorte dependendo do cartão escolhido e $ellipse_{radius}$
$ROI \leftarrow$ definir região de interesse com rectângulo com recurso a $ellipse_{center}$ e $marginCorrector$.
$img_{cropped} \leftarrow$ Selecionar apenas região de interesse em *img*
devolve $img_{cropped}$
fim função

(a) Resultado do algoritmo 1 em 4.2a

(b) Resultado do algoritmo 1 em 4.2b

(c) Resultado do algoritmo 2 em 4.2a

(d) Resultado do algoritmo 2 em 4.2b

Figura 4.3: Resultados da aplicação dos algoritmos 1 e 2 nas fotografias da figura 4.2

4.4.2 Obtenção dos anéis dos alvos

Com a parte relevante da imagem encontrada, é possível calcular os anéis do alvo de maneira a obter o centro comum necessário para pontuar os disparos efetuados.

Numa abordagem inicial foi utilizada a transformada de Hough, usada de maneira análoga à descrita na secção 4.4.1. Tendo conhecimento das medidas do alvo a ser utilizado, é possível escolher os parâmetros adequados para obter um anel específico. Apenas é necessário saber a localização de um, pois é possível calcular os outros a partir da informação retirada desse. Um caso problemático que pode ocorrer é os furos dos disparos coincidirem com o anel a detetar, o que gera erros na obtenção do centro do alvo. Na figura 4.4b foi aplicada a função *HoughCircles*, pode-se verificar que a circunferência encontrada não é coincidente com a real, havendo um desvio de alguns *pixeis* (nesta imagem, 6 *pixeis*) entre centro real e o detetado. Para corrigir esse erro, é

necessário retirar os pontos da orla do anel criados pelo disparo. Um algoritmo com recurso a uma aproximação da curva de circunferência a partir do método de mínimos quadrados (discutido na secção 2.2.1) é apresentado em 3 para obter melhor precisão. Para garantir que não é induzido erro pelo desvio do contorno mencionado anteriormente, é corrido o algoritmo de mínimos quadrados duas vezes, retirando pontos do contorno distantes da primeira estimativa do raio e centro calculados. O resultado desta rotina aplicado na imagem 4.4a é apresentado em 4.4c. Pode-se verificar o desvio encontrado no caso anterior é mitigado, tanto na posição do centro como da circunferência detetada.

4.4.3 Obtenção dos centros dos impactos

Tendo a informação espacial do alvo, para conseguir pontuar é essencial ter a informação dos furos criados pelos disparos do atirador. As diferentes abordagens testadas são exibidas nesta subsecção.

Para esta deteção, usou-se um fundo colorido de maneira a atribuir uma característica única na área dos furos, permitindo assim uma maneira rápida de deteção dos mesmos. Na figura 4.5a pode-se visualizar o fundo azul usado para o efeito descrito. A binarização é feita de maneira semelhante à secção de pré-processamento do algoritmo 2. A diferença em relação aos algoritmos é que os filtros nas três componentes são todos escolhido *offline*, sendo o mais relevante a escolha da componente *Hue*, que define a cor (exceto para branco e preto). Um resultado desta estratégia é visível em 4.5b.

Assim, a obtenção de contornos usa a função *findContours*. A procura do raio e do centro de cada furo utiliza o algoritmo 3, com duas condições: ele é aplicado para cada contorno existente e realiza-se uma verificação do tamanho do contorno a partir da sua área. A segunda medida separa assim contornos de furos sobrepostos e rasgões irregulares dos que representam de facto furos singulares.

Este método apresenta bons resultados detetando todos os furos simples do alvo, sendo visível como as circunferências vermelhas na figura 4.5c.

Apesar disso, ainda é necessário tratar do caso em que dois ou mais furos estão sobrepostos, como se pode ver com os furos juntos na figura 4.5a. Dois algoritmos que permitem detetar esta situação vão ser descritos na seguinte sub-subsecção.

4.4.3.1 Divisão por defeitos de convexidade

Na primeira tentativa,é proposto o algoritmo 4 , o qual se baseia em defeitos de convexidade do contorno. No caso ideal, as intersecções de círculos sobrepostos são as reentrâncias no invólucro de convexidade, representadas pelo traço interrompido entre os pontos cinzentos e o invólucro vermelho na figura 4.6, com número igual ao total de círculos sobrepostos. Esses pontos servem como marco para a divisão dos pontos pertencentes a cada círculo. Assim, é possível separar os pontos do contorno correspondentes às reentrâncias, sendo possível definir o conjunto de pontos

Algoritmo 3 Encontrar raio e centro do anel do alvo com recurso ao MMQ

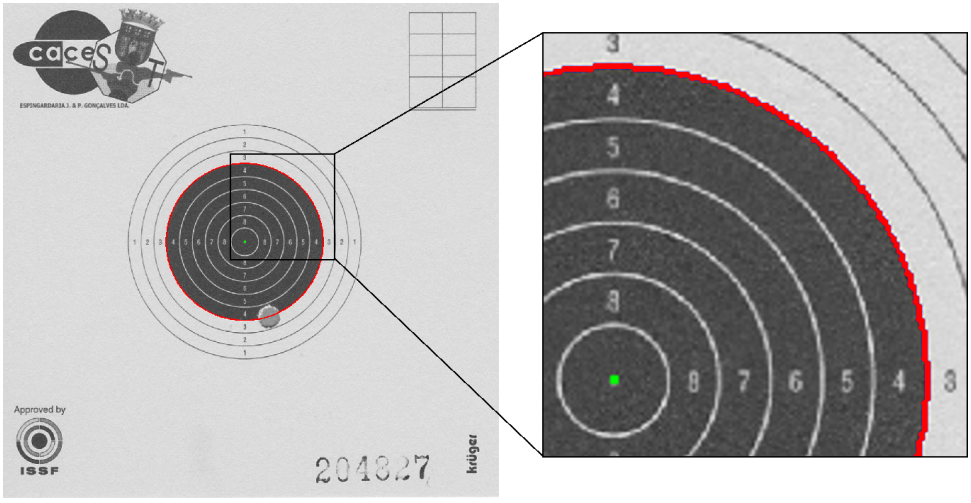
função FINDTARGETRINGDATA
preparação de iteração inicial:
$contour_{target} \leftarrow$ obter contorno adquirido em *findROI_HSV*
$x1, x2 \leftarrow$ passar das coordenadas x,y dos pontos do contorno para vetores de *Armadillo*
$B \leftarrow$ definir da matriz de coeficientes
$U, S, V \leftarrow$ aplicar SVD em B
$v \leftarrow$ obter do vetor correspondente ao menor valor singular de V
$a, b, c \leftarrow$ dividir v
$z_0, r_0 \leftarrow$ realizar estimativa inicial do raio e centro com a, b e c

aplicação inicial do MMQ:
$u \leftarrow$ juntar z_0 e r_0
para $nIts \leftarrow 0$ **até** número de iterações máximo **execute**
$f, J \leftarrow$ calcular da função objectiva e do Jacobiano a partir de u
$h \leftarrow$ resolver $^{-J}/_{f}$
$u \leftarrow$ atualizar u com h
$\Delta \leftarrow$ calcular diferença entre h e u com a norma
se $\Delta <$ tolerância **então**:
$z, r \leftarrow$ retirar centro e raio de u
sair ciclo para
senão
voltar para *aplicação inicial do MMQ*
fim se
fim para
se nIts = número de iterações máximo **então**
sair
fim se

aplicação do MMQ sem valores atípicos:
para $i \leftarrow 0$ **até** número de pontos no contorno inicial **execute**
$point \leftarrow$ ponto do contorno da iteração i
$dist \leftarrow$ distância de $point$ a z
se $dist > 1,03 \times r$ **ou** $dist < 0,97 \times r$ **então**
$numPoints \leftarrow$ retirar $point$ do contorno e guardar o novo valor
senão
continuar
fim se
fim para
se número de pontos do contorno inicial $< numPoints$ **ou** $filterFlag = 0$ **então**
$filterFlag \leftarrow 1$
voltar para *aplicação inicial do MMQ*
senão
devolve r, z
fim se
fim função

(a) Foto de exemplo dum alvo de carabina 10 m

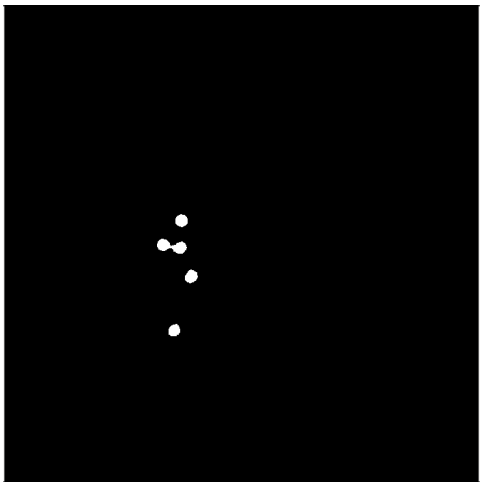
(b) Resultado da transformada de Hough circular em (a)



(c) Resultado do algoritmo 3 em (a)

Figura 4.4: Resultados da aplicação da transformada de Hough circular e do algoritmo 3 numa fotografia dum caso real

(a) Foto de exemplo dum alvo de pistola 10 m



(b) Resultado da binarização com HSV em (a)

(c) Resultado do algoritmo 3 e 4 em (b)

(d) Informação retirada dos defeitos de convexidade do contorno dos furos sobrepostos

(e) Resultado do algoritmo 5 em (b)

Figura 4.5: Resultados da aplicação da binarização com HSV, do algoritmo 4 e 5 numa fotografia dum caso real

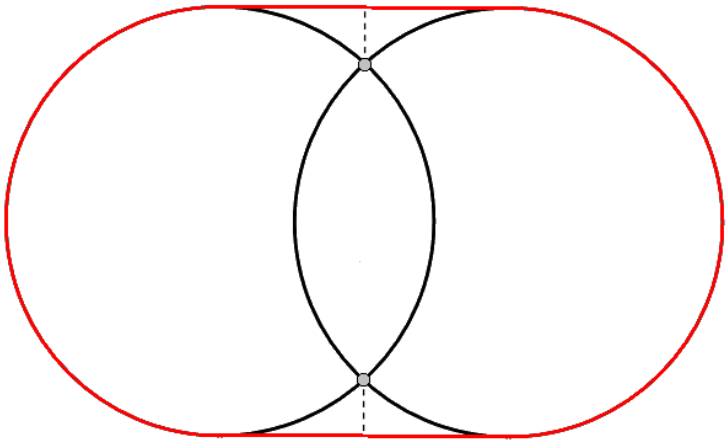


Figura 4.6: Exemplo ilustrativo dos defeitos de convexidade de círculos sobrepostos

pertencentes a cada circunferência. Mesmo assim, é preciso ter em conta que os contornos retirados possuem várias imperfeições, o que torna a tarefa de encontrar os pontos correspondentes mais complexa que o caso apresentado na figura 4.6. A figura 4.5d apresenta as informações retiradas de convexidade do contorno dos furos sobrepostos, com os pontos de profundidade a vermelho. A escolha dos pontos foi feita com o valor da profundidade do defeito, guardando apenas os pontos com maior valor. A rotina definida no algoritmo 4 é usada para a determinação dos parâmetros da circunferência do furo, com a divisão dos pontos do contorno feita com os pontos de defeito guardados.

Na maior parte dos casos testados, este algoritmo apresenta bons resultados, conseguindo encontrar os dois furos juntos na figura 4.5c.

4.4.3.2 Divisão por secções

Outra abordagem que foi testada é aproximar os pontos do contorno a uma linha a partir da função *fitLine*, que utiliza um estimador-M para realizar essa aproximação nesses pontos. A partir dessa linha, é definida uma linha perpendicular que divide o contorno, separando assim o conjunto de pontos a usar para a deteção dos furos. O algoritmo 5 demonstra com mais pormenor o processo descrito. Devido ao formato do contorno dos furos sobrepostos, a aproximação divide sempre longitudinalmente o objeto e com o auxílio da linha perpendicular é garantido uma divisão razoável da distribuição dos pontos do contorno. Este algoritmo só pode ser usado em casos que estão sobrepostos apenas dois furos (que é o caso mais comum). O resultado da aplicação está representado na figura 4.5d, onde se pode observar que foi feita uma correta deteção dos furos.

4.4.4 Pontuação dos alvos

Depois de retirada toda a informação relevante do alvo, reúnem-se as condições necessárias para pontuar o mesmo corretamente e informar o utilizador.

Algoritmo 4 Escolher pontos de furos sobrepostos com recurso a defeitos de convexidade

função FINDOVERLAPPEDHOLESDATA_DEFECT

escolha de contornos:
$contours_{hole} \leftarrow$ obter contornos adquiridos dos furos
minArea, *maxArea* $\leftarrow$ definir limites de escolha de área usando o calibre da munição
para $i \leftarrow 0$ **até** número de contornos total **execute**
se Área de contorno $>$ *minArea* **e** Área de contorno $<$ *maxArea* **então**:
manter contorno
senão
retirar contorno
fim se
fim para
se nenhum contorno escolhido **então**:
sair
senão
ir para *escolha da região de interesse*
fim se

separação de pontos de sobreposição a partir dos defeitos de convexidade:
$hull \leftarrow$ obter invólucro de convexidade do contorno com *convexHull*
$defects \leftarrow$ obter defeitos de convexidade a partir de $hull$ com *convexityDefects*
$defects_{depth} \leftarrow$ retirar profundidade dos defeitos de convexidade
$defects_{points} \leftarrow$ criar vector de pontos de tamanho igual ao número de disparos restantes
para $i \leftarrow 0$ **até** tamanho de $defects_{depth}$ **execute**
para $j \leftarrow 0$ **até** tamanho de $defects_{points}$ **execute**
se $defects_{depth}[i] > defects_{points}[j]$ **então**:
guardar ponto de contorno do novo defeito em $defects_{points}$
senão
continuar
fim se
fim para
fim para

deteção dos pontos com o contorno dividido:
para $num \leftarrow 0$ **até** tamanho de $defects_{points}$ **execute**
para $k \leftarrow 0$ **até** tamanho de $contours_{hole}$ **execute**
percorrer pontos desde $defects_{points}[num]$ até ao próximo $defects_{points}$
$contour_{overlapped} \leftarrow$ guardar os pontos do contorno
fim para
$r_{hole}, z_{hole} \leftarrow$ guardar centro e raio detetado da função *findTargetRingData* com $contours_{overlapped}$
fim para
devolve r_{hole}, z_{hole}
fim função

Algoritmo 5 Escolher pontos de furos sobrepostos com divisão geométrica

função FINDOVERLAPPEDHOLESDATA_DIVIDE

cálculo da linha divisória:

$v_x, v_y, p_x, p_y \leftarrow$ Usar função *fitLine* em $contours_{hole}$ e guardar vector e ponto que definem a linha estimada

$pointLeft_{fitLine}(x) \leftarrow$ definir coordenada x do ponto esquerdo da linha estimada com 0

$pointLeft_{fitLine}(y) \leftarrow -p_x \times \frac{v_y}{v_x} + p_y$

$pointRight_{fitLine}(x) \leftarrow$ definir coordenada x do ponto direito da linha estimada com $width(img_{cropped})$

$pointRight_{fitLine}(y) \leftarrow (width(img_{cropped}) - p_x) \times \frac{v_y}{v_x} + p_y$

$normal \leftarrow -\frac{v_x}{v_y}$ ▷ declive da linha divisória

$magnitude \leftarrow \sqrt{1^2 + normal^2}$ ▷ magnitude da linha divisória

$v2_x \leftarrow \frac{normal}{magnitude}$

$v2_y \leftarrow \frac{1}{magnitude}$

$pointLeft_{dividerLine}(x) \leftarrow$ definir coordenada x do ponto esquerdo da linha divisória com 0

$pointLeft_{dividerLine}(y) \leftarrow -p_x \times \frac{v2_y}{v2_x} + p_y$

$pointRight_{dividerLine}(x) \leftarrow$ definir coordenada x do ponto direito da linha divisória com $width(img_{cropped})$

$pointRight_{dividerLine}(y) \leftarrow (width(img_{cropped}) - p_x) \times \frac{v2_y}{v2_x} + p_y$

separação dos pontos do contornos:

para $i \leftarrow 0$ **até** tamanho de $contours_{hole}$ **execute**

$position \leftarrow (pointRight_{dividerLine}(x) - pointLeft_{dividerLine}(x)) * (contours_{hole}[i](y) - pointLeft_{dividerLine}(y)) - (pointRight_{dividerLine}(y) - pointLeft_{dividerLine}(y)) * (contours_{hole}[i](x) - pointLeft_{dividerLine}(x))$

se position > 0 **então** ▷ ponto à esquerda de *dividerLine*

$contourLeft_{overlapped} \leftarrow$ guardar pontos de $contours_{hole}$

senão se position < 0 **então** ▷ ponto à direita de *dividerLine*

$contourRight_{overlapped} \leftarrow$ guardar pontos de $contours_{hole}$

senão

$contourLeft_{overlapped}, contourRight_{overlapped} \leftarrow$ guardar pontos de $contours_{hole}$ em ambos

fim se

fim para

deteção dos furos com o contorno dividido:

para $j \leftarrow 0$ **até** 1 **execute**

se j == 0 **então**

$contours_{hole} \leftarrow contourLeft_{overlapped}$

senão

$contours_{hole} \leftarrow contourRight_{overlapped}$

fim se

$r_{hole}, z_{hole} \leftarrow$ guardar centro e raio detetado da função *findTargetRingData* com $contours_{hole}$

fim para

devolve r_{hole}, z_{hole}

fim função

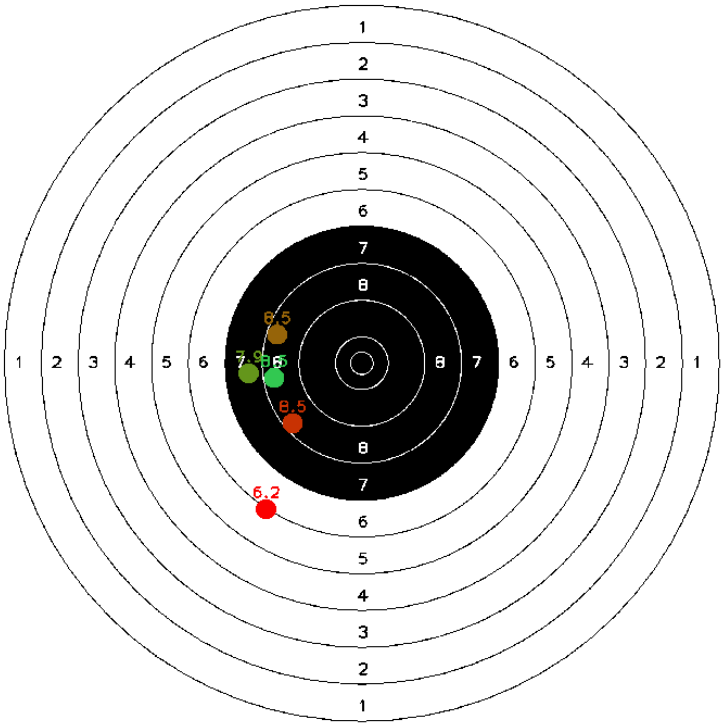


Figura 4.7: Resultado da pontuação dum alvo de pistola 10 m

Um alvo virtual é desenhado com os furos nas suas localizações respetivas, nos quais é escrita a pontuação em décimas por cima da área do furo. Esta operação pode ser vista na figura 4.7. Os furos são desenhados em diferentes cores para serem mais facilmente visualizados. A informação de pontuação também é impressa na janela de texto da consola caso os furos estejam sobrepostos.

A pontuação é calculada de acordo com a explicação descrita na secção 3.1.2 do capítulo 3, que matematicamente é o cálculo da norma de um vetor.

4.4.5 Interface gráfica do utilizador (GUI)

Com o intuito de facilitar os testes realizados e fornecer um ambiente gráfico para o utilizador manipular o programa, criaram-se GUIs com recursos às funções disponibilizadas pelo Qt Creator.

A UI de testes, representada na figura 4.8a, é constituída por uma janela principal onde se encontram todos os botões e caixas de texto que iniciam a execução dos diferentes códigos a testar. Sempre que se queira experimentar um novo trecho de código separadamente, adiciona-se um botão referente a esse processo. Para a edição rápida de parâmetros relevantes às funções que se encontram em teste, foram colocadas caixas de texto que permitem escrever valores para as variáveis associadas e também *sliders* para o mesmo efeito. A interface possui uma caixa na parte direita que mostra a imagem carregada pelo utilizador ou tirada pela câmara e desenha todas as edições feitas pelos diferentes métodos quando chamados. Além das estruturas descritas, há uma

caixa de texto que imprime informação sobre a execução da aplicação e dois botões tipo lista, um para a escolha do tipo de alvo e outro para a modo/representação de imagem.

Adaptando a interface de testes, realizou-se a versão final da interface, ilustrada na figura 4.8b. Esta GUI é mais simplificada em relação à anterior, tendo apenas os botões necessários para as operações do utilizador final. As funcionalidades implementadas são:

- ***Load Image*** — Carrega para o programa uma imagem do utilizador escolhida;
- ***Start Scoring*** — Inicia o processo de pontuação do alvo, mas só se estiver escolhido anteriormente o tipo de alvo. Antes da execução dos algoritmos, pede ao utilizador para inserir o número de disparos efetuados, para facilitar o processo de deteção. Imprime no ecrã uma imagem semelhante à figura 4.7. Caso esteja um a correr, reinicia o processo de pontuação da ronda;
- ***Next Target*** — Continua o processo de pontuação da ronda, onde se espera que o alvo seja diferente;
- ***Show Picture*** — Mostra a imagem original ou tirada pela câmara ao utilizador apenas quando o botão está a ser pressionado;
- ***Input User Info*** — Cria uma janela em que pede ao utilizador para escrever o nome do utilizador e o seu número de série.
- ***Export Image*** — Exporta a imagem de pontuação gerada por ***Start Scoring*** para um diretório à escolha do utilizador;
- ***Export Score*** — Gera um ficheiro de texto com a informação da ronda e alvos pontuados e grava para um diretório à escolha do utilizador.

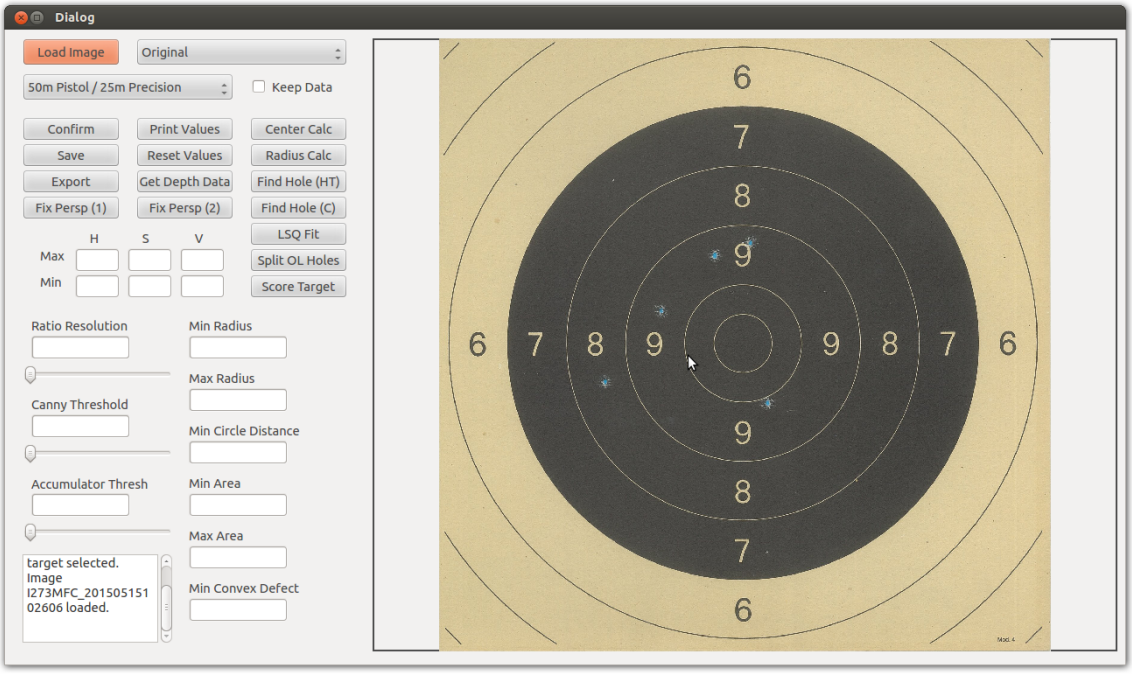
Para além do descrito, existe uma caixa de texto que imprime as pontuações obtidas dos alvos classificados e também tem no canto inferior direito a informação do utilizador atual.

4.5 Discussão

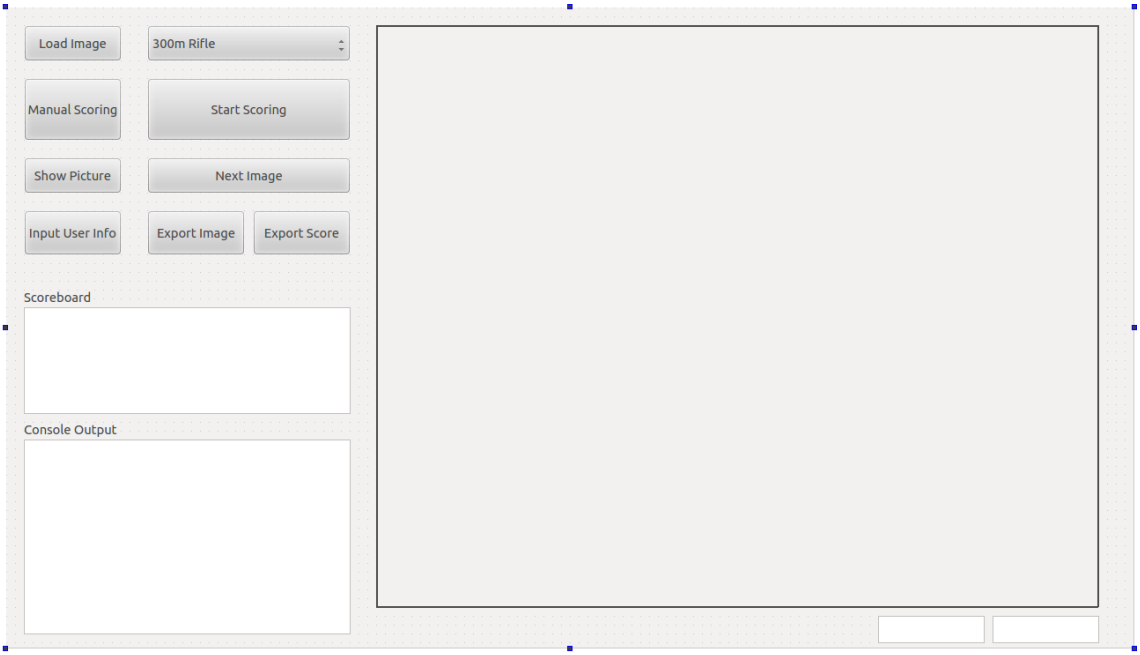
Neste capítulo, algumas abordagens foram testadas para averiguar a sua funcionalidade e eficácia na resolução dos problemas propostos.

Na aquisição de imagem, dois testes a tecnologias diferentes foram realizados, fazendo uma comparação entre elas: o módulo da câmara da RPi com a biblioteca Raspicam e a câmara digital Nikon CoolPix com a biblioteca gPhoto2.

Para o processamento da imagem, foram desenvolvidos algoritmos para cada etapa, os quais foram testados. Na definição da região de interesse, testou-se um algoritmo com recurso à transformada de Hough e outro com recurso à segmentação por cor e deteção de contornos. Na obtenção dos anéis dos alvos, foi usada a informação de raio e centro do contorno da região de interesse



(a) Interface gráfica para a realização de testes



(b) Interface gráfica para o protótipo

Figura 4.8: Exemplo das UIs criadas

detetada anteriormente para estimar os verdadeiros valores a partir do método de mínimos quadrados. Já as abordagens para a obtenção dos das posições dos furos, foram feitas de maneira semelhante às descritas anteriormente, com recurso a segmentação de cor, contornos e MMQ. Foi necessário desenvolver algoritmos para a situação em que os impactos se encontram sobrepostos, em que um se baseia em defeitos de convexidade de contornos e o outro com um estimador-M para a equação da linha de maneira a separar o contorno em dois.

A pontuação dos alvos é conseguida com o cálculo da distância entre os a região dos impactos mais perto do centro alvo e o centro do alvo.

No final, é desenhado uma imagem virtual do alvo com os furos para o utilizador que fornece informações como a pontuação.

Para a realização dos testes do código gerado como também para criar a interação entre o utilizador e o programa, interfaces gráficas foram criadas com recurso ao Qt.

Capítulo 5

Resultados

Neste capítulo são apresentado os resultados obtidos dos algoritmos descritos anteriormente, tanto ao nível da eficácia da deteção do raio e centro retirados do alvo e dos furos provocados pelos impactos como na precisão da pontuação efetuada. O tempo de execução entre trechos de código de algoritmos diferentes e entre o computador de teste e a placa também são inspecionados.

5.1 Erros

5.1.1 Erros de pontuação do alvo

A distância entre o centro do alvo e a zona de impacto foi medida com recurso a um paquímetro, com um erro de $\pm 0,05$ mm. A tabela 5.1 mostra a média e o desvio-padrão do erro absoluto entre a distância real medida e a calculada pelo programa desenvolvido. A amostra utilizada é de 50 impactos.

Tabela 5.1: Média e Desvio Padrão do erro absoluto entre a medição da distância de pontuação e a calculada pelo programa em mm

	Média (mm)	Desvio Padrão (mm)	Erro Máximo (mm)
Sem correção	1.760	0.335	1.790
Com correção	0.254	0.214	0.570

O erro absoluto médio, como visto na tabela 5.1 é grande para a resolução pedida, induzindo em erros de pontuação até duas décimas. Em contrapartida, todas as medidas calculadas pelo programa são maiores que as reais, fazendo o erro ser sistemático. Assim, é possível corrigi-lo, obtendo o valor máximo de 0.570 mm.

5.2 Tempos de execução de código

Foi temporizado a execução de todo o código criado. Para medir o tempo, recorreu-se à classe *QElapsedTimer*, que fornece temporizações até ao nanossegundo [49]. Os resultados obtidos estão representados da tabela 5.2 até à tabela 5.7, com uma resolução de milissegundos.

Tabela 5.2: Média e Desvio Padrão do tempo de execução da rotinas da definição do ROI em ms

		Média (ms)	Desvio Padrão (ms)
Algoritmo 1	Insys	3747.264	92.613
	RPi 2	23763.050	185.007
Algoritmo 2	Insys	466.319	4.266
	RPi 2	820.525	6.063

Tabela 5.3: Média e Desvio Padrão do tempo de execução da função *HoughCircles* em ms

	Média (ms)	Desvio Padrão (ms)
Insys	93.589	8.716
RPi 2	431.700	3.541

Tabela 5.4: Média e Desvio Padrão do tempo de execução da rotina de deteção de contornos em ms

	Média (ms)	Desvio Padrão (ms)
Insys	210.534	10.751
RPi 2	2521.370	16.893

Tabela 5.5: Média e Desvio Padrão do tempo de execução da rotinas deteção de impactos sobrepostos em ms

		Média (ms)	Desvio Padrão (ms)
Algoritmo 4	Insys	4.115	5.796
	RPi 2	60.460	2.728
Algoritmo 5	Insys	39.684	10.944
	RPi 2	210.830	6.635

Tabela 5.6: Média e Desvio Padrão do tempo de execução da rotinas de mínimos quadrados em ms

		Média (ms)	Desvio Padrão (ms)
Algoritmo MMQ no contorno do alg. 3	Insys	15.411	8.265
	RPi 2	135.241	7.851
Algoritmo MMQ no contorno do alg. 4 e 5	Insys	4.747	5.478
	RPi 2	212.440	7.490

Tabela 5.7: Média e Desvio Padrão do tempo de execução da rotina de desenho e pontuação do alvo em ms

	Média (ms)	Desvio Padrão (ms)
Insys	72.345	8.215
RPi 2	260.660	8.399

Algumas observações gerais podem ser tiradas destes valores. Os valores de desvio-padrão são relativamente semelhantes entre plataformas e nunca muito elevados. Estes pequenos desvios devem-se ao facto de existirem tarefas de maior prioridade nos sistemas operativos, que interrompem durante o tempo de processamento, adicionando atrasos. Quanto mais demorado for o código a executar, mais tarefas de prioridade superior irão intervir, aumentando o atraso.

Outra observação óbvia é os tempos de processamento na RPi são superiores aos do Insys. Tal facto é expectável, pois o Insys possui características superiores às da RPi. Além disso, foi possível usar o OpenMP no Insys, uma interface programável de aplicação (API) que possibilita a execução de código em paralelo, utilizando todos os processadores existentes no PC [50].

Em relação à comparação de algoritmos implementados, há diferenças relevantes no tempo de execução. Na tabela 5.2, o algoritmo 2 de segmentação HSV com contornos é mais rápido que o algoritmo 1, que utiliza a transformada de Hough circular. Não tendo informação sobre o tamanho ou posição da circunferência e sendo esta função computacionalmente pesada, é esperado que o desempenho seja pior. Na tabela 5.5, o algoritmo 4 de defeitos de convexidade é mais rápido que o 5.

5.3 Discussão

O resultado de testes efetuados para avaliação da performance do sistema desenvolvido foi apresentado neste capítulo.

O erro de medição resultante do cálculo da distância pelo programa expresso pela média aritmética e o desvio-padrão foi apresentado, o que mostrou ser um desvio sistemático, o que faz possível uma correção.

Além destas medições, também se mediu tempos de execução das diferentes funções implementadas, onde se chegou às conclusões de a Raspberry Pi 2 ser mais lenta em processamento que o PC de testes, devido ao processador mais potente do Insys e do uso do OpenMP, que agiliza a execução do código. Comparações entre algoritmos para o mesmo fim também foram realizadas, onde se observou que o algoritmo 2 é mais rápido que o 1, devido ao uso da transformada de Hough com pouco conhecimento do problema, e que o algoritmo 4 é mais ágil que o 5.

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo final é apresentada um levantamento dos pontos mais relevantes do projeto desenvolvido, deliberando sobre as escolhas efetuadas no desenvolvimento do mesmo. O documento é finalizado com sugestões de possíveis medidas que podem ser aplicadas no protótipo, de maneira a melhorar a qualidade do trabalho.

6.1 Conclusões

Este projeto de Dissertação foi realizado um levantamento do estado atual dos sistemas de pontuação existentes no mercado, que em conjunto com um estudo realizado das várias tecnologias disponíveis para uso e da informação do problema em questão, definiu o plano para a construção de um protótipo e a codificação do relativo *software*.

A escolha de material, tendo em conta que se apontava para uma boa relação custo/qualidade, inicialmente, apresentou ser correta. Observando os tempos de execução do código desenvolvido, a RPi 2 é bastante lenta em comparação ao PC de testes (Insys Gameforce).

Numa fase inicial foram testadas várias técnicas de processamento de imagem. Das duas abordagens de aquisição da imagem, a biblioteca Raspicam com o uso da câmara escolhida, o módulo de câmara da Raspberry Pi, provou ser uma solução muito mais eficiente que a biblioteca gPhoto2 com uma câmara digital comercial de baixo custo, permitindo tempos mais curtos.

Na extração de características do alvo, a transformada de Hough apresentou ser computacionalmente lenta em relação a opções como deteção de contornos aliado à segmentação por cor num espaço HSV, em ambas as situações de encontrar o alvo na imagem fotografada para o corte da ROI, como na deteção de circunferências. Também apresentou alguns erros na deteção, onde mais uma vez o uso de contornos aliado com o método de mínimos quadrados é mais preciso no cálculo dos parâmetros certos dos círculos.

Os dois algoritmos usados no caso em que existe sobreposição dos furos, tanto o baseado em defeitos por convexidades como o do estimador-M da linha, funcionam para a maior parte dos casos. Os defeitos de convexidade conseguem cobrir as situações em que vários furos se encontram sobrepostos, mas a divisão correta do contorno está dependente da maneira como o

rasgão derivado dos impactos se apresenta. Formas muito irregulares levam a falsos positivos, fazendo que sejam detetados mais furos dos que de facto existem. Já no caso da divisão feita pela linha perpendicular à estimada pelo estimador-M, é garantido que a deteção seja sempre feita, mas está limitado a dois impactos sobrepostos, devido à natureza do algoritmo.

No cálculo na distância para a pontuação, o erro é aceitável desde que se garanta que se veja o alvo na totalidade, aumentando a relação $pixel/mm$, o que aumenta a precisão do sistema.

6.2 Trabalho Futuro

Tendo em conta que o trabalho apresentado é apenas um protótipo construído num espaço de seis meses como prova de conceito, várias funcionalidades e melhorias podem ser exploradas. Uma lista de sugestões para trabalho futuro de forma a acelerar melhorar o sistema e acelerar os algoritmos descritos é então apresentada:

- **Criação de uma base de dados** — a integração de uma base de dados para guardar várias informações da ronda em vez de exportar para um ficheiro de texto. Assim, é possível guardar toda a informação num servidor que depois pode ser acedido por o utilizador, caso tenha conta válida. Também permitiria realizar estatísticas em maior escala que as permitidas neste momento;

- **Rede *wireless*** — Outra funcionalidade que se pode acrescentar é a possibilidade de integração com uma rede *wireless*, tornando o sistema mais cómodo. Em casos que a distância à zona de tiros seja grande, esta alternativa é melhor que uma solução cablada e mais barata. Seria mais fácil visualizar em vários dispositivos ao mesmo tempo (acesso ao *browser*) em vez de partilhar uma ligação por cabos ou sem a necessidade de ter um módulo central;

- **Melhoramento na deteção** — Analisando os algoritmos, ainda se podem detetar falhas nas deteções de furos. Seria interessante experimentar outros algoritmos (por exemplo, *blob detector* [51]) para comparar com os previamente implementados.

- **Uso de *OpenMP*** — Apesar de não sido possível de implementar, o uso de bibliotecas para a utilização de processadores *multicore* de maneira eficiente, como o paralelismo de programas, apresenta ser uma medida viável para colmatar esse problema. Um exemplo referido no capítulo anterior de uma API é o OpenMP , que é compatível com OpenCV.

Anexo A

Anexos

A.1 Informação dos alvos

Tabela A.1: Dimensões dos diferentes alvos

		Anel (mm)									
		Dez Interior		10		9		8		7	
Alvo	**Cartão**	Medida	Desvio	Medida	Desvio	Medida	Desvio	Medida	Desvio	Medida	Desvio
Carabina 300m	1300x1300	50	0,5	100	0,5	200	1,0	300	1,0	400	3,0
Carabina 50m	250x250	5	0,1	10,4	0,1	26,4	0,1	42,4	0,2	58,4	0,2
Carabina de ar 10m	80x80	N/A		0,5	0,1	5,5	0,1	10,5	0,2	15,5	0,2
Pistola precisão 50m/25m	550x550 (-30)	25	0,2	50	0,2	100	0,4	150	0,5	200	1,0
Pistola de ar 10m	170x170	5,0	0,1	11,5	0,1	27,5	0,1	43,5	0,2	59,5	0,5
Pistola de velocidade 25m	550x550 (-30)	50	0,2	100	0,4	180	0,6	260	1,0	340	1,0

Anel (mm)											
6		5		4		3		2		1	
Medida	Desvio	Medida	Desvio	Medida	Desvio	Medida	Desvio	Medida	Desvio	Medida	Desvio
500	3,0	600	3,0	700	3,0	800	3,0	900	3,0	1000	3,0
74,4	0,5	90,4	0,5	106,4	0,5	122,4	0,5	138,4	0,5	154,4	0,5
20,5	0,5	25,5	0,5	30,5	0,5	35,5	0,5	40,5	0,5	45,5	0,5
250	1,0	300	1,0	350	1,0	400	2,0	450	2,0	500	2,0
75,5	0,5	91,5	0,5	107,5	0,5	123,5	0,5	139,5	0,5	155,5	0,5
420	2,0	500	2,0	N/A							

Bibliografia

[1] W. H. B. Smith and J. E. Smith, *Small arms of the world: a basic manual of small arms*. Stackpole Books, 1973.

[2] W. A. Mandelbaum, "Gunpowder: Alchemy Bombards & Pyrotechnics: The History of the Explosive That Changed the World," *Military History*, vol. 21, no. 3, pp. 68–69, 2004.

[3] C. Kyle and W. Doyle, "American Gun: A History of the U.S. in Ten Firearms," *William Morrow Paperbacks*, no. 1st edition, 2013.

[4] I. Langhag, "Brochure - Sius."

[5] G. Anderson, "Information about Finals and Their Use in Competition Conducting Finals The New ISSF 50m," 2014.

[6] "The ISSF History - International Shooting Sport Federation ." [Online]. Available: http://www.issf-sports.org/theissf/history.ashx [Acedido: 2015-06-28]

[7] X. Fan, Q. Cheng, P. Ding, and X. Zhang, "Design of automatic target-scoring system of shooting game based on computer vision," *Proceedings of the 2009 IEEE International Conference on Automation and Logistics, ICAL 2009*, no. August, pp. 825–830, 2009.

[8] A. M. Mendonça, "Segmentação - Sistemas Baseados em Visão," pp. 1–100, 2014.

[9] P. R. Aryan, "Mobile Shooting Range," pp. 978–979, 2012.

[10] B. G. Mobasseri, "Automatic target scoring system using machine vision," *Machine Vision and Applications*, vol. 8, no. 1995, pp. 20–30, 1995.

[11] C. Ye and H. Mi, "The technology of image processing used in Automatic Target-Scoring System," *Proceedings - 4th International Joint Conference on Computational Sciences and Optimization, CSO 2011*, pp. 349–352, 2011.

[12] P. Sumathy and M. Aarthy, "A COMPARISON OF HISTOGRAM EQUALIZATION METHOD AND HISTOGRAM EXPANSION," *International Journal of Computer Science and Mobile Applications*, vol. 2, no. 3, pp. 25–34, 2014.

[13] F. Ali, "Shooting Targets," pp. 515–519, 2008.

[14] R. C. Gonzalez, R. E. Woods, and B. R. Masters, "Digital Image Processing, Third Edition." *Journal of biomedical optics*, vol. 14, no. 2, pp. 1–976, 2009.

[15] W. Bieniecki and S. Grabowski, "Multi-pass approach to adaptive thresholding based image segmentation," *Computer Engineering Dpt., TU Lodz, Poland, CADSM*, pp. 4–8, 2005. [Online]. Available: http://wbieniec.kis.p.lodz.pl/research/files/05_CADSM_bernsen.pdf

[16] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. January, pp. 62–66, 1979.

[17] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. January, p. 220, 2004.

[18] A. M. Mendonça, "Pré-Processamento - Sistemas Baseados em Visão," pp. 1–77, 2014.

[19] G. D. Duarte, "Uso da Transformada de Hough na Detecção de Círculos em Imagens Digitais."

[20] R. O. Duda and P. E. Hart, "Use of the Hough Transformation To Detect Lines and Curves in Pictures," vol. 15, no. 1, 1972.

[21] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses," *Bit*, vol. 34, no. 4, pp. 558–578, 1994.

[22] V. P. Godambe, *Estimating functions*. Clarendon Press Oxford, 1991.

[23] A. Soetedjo, A. Mahmudi, M. I. Ashari, and Y. I. Nakhoda, "DETECTING LASER SPOT IN SHOOTING SIMULATOR USING AN EMBEDDED CAMERA," vol. 7, no. 1, pp. 423–441, 2014.

[24] "Easyscore 220." [Online]. Available: http://www.hellotrade.com/rika-sport-gmbh-cokg/target-transport-systems-easyscore-220.html [Acedido: 2015-02-12]

[25] "RM-IV - DISAG." [Online]. Available: http://www.disag.de/produkte/rm-iv/ [Acedido: 2015-02-13]

[26] "TargetScan App - Paper Target Scoring System for iPhone." [Online]. Available: http://www.targetshootingapp.com/ [Acedido: 2015-06-27]

[27] "TargetScan - Pistol & Rifle Target Scoring on the App Store on iTunes." [Online]. Available: https://itunes.apple.com/us/app/targetscan-pistol-rifle-target/id448045769?mt=8 [Acedido: 2015-06-27]

[28] "LISA - Auswertesoftware." [Online]. Available: http://ssv-dorsten.de/50022996a01312225/ [Acedido: 2014-12-15]

[29] "Sport shooting score registration software." [Online]. Available: http://www.abvisie.nl/en/mm.html [Acedido: 2014-12-15]

[30] "Orion Scoring System Bundles." [Online]. Available: http://www.orionscoringsystem.com/orion/BundleList.aspx [Acedido: 2014-12-15]

[31] "SIUS AG." [Online]. Available: http://www.sius.com/en/03.12_Produkte_Sport_Scheiben.html [Acedido: 2015-06-27]

[32] "Electronic Target Systems (ESA)." [Online]. Available: http://www.haeringtargets.com/index.php/en/elektr-scheibenanlagen/electronic-target-system [Acedido: 2015-06-27]

[33] "Megalink." [Online]. Available: http://www.megalink.no/ [Acedido: 2015-06-27]

[34] "Meyton Elektronik: BLACK MAGIC." [Online]. Available: http://www.meyton.info/en/products/black-magic/index.html [Acedido: 2015-06-28]

[35] P. A. d. C. Avelino, "Reconhecimento óptico de furos em alvos de tiro desportivo na ISR," p. 36, 2006.

[36] "OpenCV." [Online]. Available: http://opencv.org/ [Acedido: 2015-01-05]

[37] "MATLAB - The Language of Technical Computing." [Online]. Available: http://www.mathworks.com/products/matlab/?s_tid=hp_fp_ml [Acedido: 2015-06-27]

[38] "Image Processing Toolbox - MATLAB." [Online]. Available: http://www.mathworks.com/products/image/ [Acedido: 2015-06-27]

[39] C. Sanderson, "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments," *Technical Report*, vol. NICTA, pp. 1–16, 2010. [Online]. Available: http://it.uq.edu.au/~conrad/pdfs/armadillo_nicta_2010.pdf$\delimiter"026E30F$nhttps://fossies.org/linux/privat/armadillo-3.900.5.tar.gz:a/armadillo-3.900.5/armadillo_nicta_2010.pdf

[40] I. S. S. Federation, *International Shooting Sport Federation Official Statutes Rules and Regulations*, 3rd ed., vol. 2013, no. January 2013.

[41] ISSF; Federação Portuguesa de Tiro, "Regulamento técnico para todas as disciplinas de tiro," pp. 1–44, 2005.

[42] "Raspberry Pi 2 Model B." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [Acedido: 2015-06-27]

[43] "Raspberry Pi 2 on sale now at $35 - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/raspberry-pi-2-on-sale/ [Acedido: 2015-06-27]

[44] "Camera Module - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/camera-module/ [Acedido: 2015-06-27]

[45] "Raspberry Pi Camera Board: Adafruit Industries, Unique & fun DIY electronics and kits." [Online]. Available: http://www.adafruit.com/products/1367 [Acedido: 2015-06-27]

[46] “Qt Project.”

[47] “gPhoto2 - Digital Camera Software.” [Online]. Available: http://www.gphoto.org/ [Acedido: 2015-06-27]

[48] “RaspiCam: C++ API for using Raspberry camera with/without OpenCv | Aplicaciones de la Visión Artificial.” [Online]. Available: http://www.uco.es/investiga/grupos/ava/node/40 [Acedido: 2015-06-27]

[49] “QElapsedTimer Class | Qt 4.8.” [Online]. Available: http://doc.qt.io/qt-4.8/qelapsedtimer.html [Acedido: 2015-06-29]

[50] “OpenMP.org.” [Online]. Available: http://openmp.org/wp/ [Acedido: 2015-06-29]

[51] “Blob Detection Using OpenCV (Python, C++) | Learn OpenCV.” [Online]. Available: http://www.learnopencv.com/blob-detection-using-opencv-python-c/ [Acedido: 2015-06-29]