INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Aplicações de Visão em C++

João Filipe de Almeida Caetano Firmino Freire

Dissertação para obtenção do Grau de Mestre em

Mestrado Integrado em Engenharia Mecânica

Júri

Presidente: Prof. Hélder Carrico Rodrigues

Orientador: Prof. João Caldas Pinto

Vogais: Prof. Mário António Neves Ramalho

Prof. José Luís Carrilho Sequeira

Novembro 2009

AGRADECIMENTOS

Este trabalho não poderia ter sido feito sem o essencial apoio de várias pessoas que, voluntária ou involuntariamente incutiram motivação e inspiração mesmo nos momentos mais difíceis.

À minha família, em especial ao meu Pai e à minha Mãe, que sem eles nada disto seria possível. Local de trabalho, tecto acolhedor e sossegado para repousar a mente, condições mais que suficientes para assegurar um bem-estar e uma calmia que me permitiram momentos longos de reflexão e pensamento.

Ao Daniel e ao Bruno por terem acompanhado as minhas exaustivas explicações sobre os procedimentos que me propus a implementar, quando estes ameaçavam só fazer sentido na minha cabeça.

Ao Prof. Caldas Pinto que me guiou e orientou durante o processo de amadurecimento da ideia até à fase de conclusão, nunca se abstendo de criticar quando era preciso, nem que para isso me fizesse duvidar do sucesso da minha demanda.

Ao Pedro Morgado por ter discutido aspectos técnicos que poucos perceberiam.

Ao Gonçalo Vinagre por me ter acompanhado nos anos em que o gosto pela programação e automação cresceram em mim.

Ao Gaspar, Hélder, Sara, Pedro Mona e Hélia por terem ouvido vezes sem conta as minhas lamúrias e queixas da tese, por no Verão terem dado uso às minhas facilidade de divertimento enquanto eu fechado em casa lutava por me concentrar, por todos os fins-de-semana que me permitiram descarregar frustrações e carregar baterias.

Ao Marlo, por todos os momentos passados e vários *brainstorming* partilhados, mesmo quando a única resposta que obtinha era um longo silêncio de alguém que não percebia nada do assunto, mas que depois dizia “vá força com isso então”.

RESUMO

Este trabalho foca-se no desenvolvimento de aplicações de processamento de imagem, baseado numa montagem pré-existente no laboratório de Automação e Robótica do Instituto Superior Técnico. Foi desenvolvido um programa em C++, compacto e robusto, com facilidades no sentido de futuros desenvolvimentos. Foi criada uma base de dados com quatro cores, vermelho, verde, azul e amarelo, e outra com cinco formas distintas. Através da captura de imagens através de uma webcam, o reconhecimento é feito detectando a classe de cor ou de forma a que um objecto pertence. Foram pesquisados métodos de extracção das características desses objectos de modo a desenvolver um sistema de decisão automática. Com o avançar do programa optou-se por um sistema simples de classificação pelo método do Vizinho Mais Próximo.

Para a descrição da cor usou-se o parâmetro *h* do espaço de cor LCH. Para a forma fez-se uma triangulação de Delaunay, sendo criado um histograma de ângulos. Todas as decisões são posteriormente transferidas para um autómato que actua de acordo com a situação detectada.

O *software* é fácil de utilizar e de modificar. O sistema de classificação quer para a cor quer para a forma apresentou-se bastante eficaz, mesmo nas condições mais adversas.

Palavras-chave: detecção de cor, detecção de forma, contorno, reconhecimento de objectos, triangulação de Delaunay.

ABSTRACT

This work focus on the development of applications of image processing based on a preexisting assembly in the laboratory of Automation and Robotics of the Instituto Superior Técnico. A program in C++, compact and robust, was developed, taking into account future developments. A database with four colors, red, green, blue and yellow, was created and another one with five distinct shapes. Through the capture of images by a webcam, the recognition is made matching an object to the correspondent shape or color class. A research regarding features extraction was made, in order to develop a system of automatic decision. In the end was chosen a simple method for classification, based on the nearest neighbor. For the description of the color it was chosen the parameter *h* from the LCH space color. For the shape it was used a Delaunay triangulation, creating an angle histogram. All the decisions were later transferred to a programmable logic controller that actuates in accordance with the detected situation.

The *software* is easy to use and to modify. The classification system used for color and shape was presented sufficiently efficient, even in the most adverse conditions.

Keywords: color detection, shape detection, contour, objects recognition, Delaunay triangulation.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

O campo da visão computacional pode ser caracterizado como complexo e diverso [1]. Foi somente no final da década de 70 que começaram estudos aprofundados, quando os computadores já podiam processar grandes conjuntos de dados como imagens. Contudo a investigação baseava-se em problemas concretos que iam surgindo, não existindo uma formulação padrão para o problema da visão computacional, assim como não existe uma formulação padrão de como os problemas de visão computacionais devem ser resolvidos [2]. O que existe actualmente são diversos métodos para resolver várias tarefas bem definidas, no qual os métodos são bastante especializados e raramente podem ser generalizados para várias aplicações. Na maioria das aplicações de visão computacional, os computadores são pré-programados para resolver uma tarefa particular, mas métodos baseados em aprendizagem estão-se a tornar cada vez mais comuns.

Uma parte significativa da inteligência artificial lida com sistemas que realizam acções mecânicas como movimentar um robô num determinado ambiente. Esse tipo de processamento necessita de um sistema de visão computacional, um sensor de visão, fornecendo informações de alto nível sobre o ambiente ao robô. Outras áreas da inteligência artificial relacionadas com a visão computacional são o reconhecimento de padrões e a aprendizagem da máquina. Como consequência, a visão computacional é por vezes vista como parte da inteligência artificial ou da ciência da computação de modo geral.

A Física é outro campo relacionado. Grande parte da visão computacional lida com métodos que requerem o entendimento dos processos no qual a radiação electromagnética, tipicamente na camada visível ou infravermelha, é reflectida pelas superfícies dos objectos e capturada pelos sensores para formar a imagem. Esse processo é baseado na óptica e na física dos estados sólidos. Sensores de imagem mais sofisticados requerem também a física quântica para fornecer uma compreensão completa do processo de formação da imagem [3]. Também alguns dos problemas de movimento de fluidos podem ser resolvidos com a visão computacional.

Um terceiro campo relacionado é a neurobiologia, especificamente o estudo de sistemas biológicos de visão. Durante o século XX houve vários estudos extensos sobre as estruturas do olho, neurónios e do cérebro, devotados ao processo de estimulação visual tanto nos

humanos quanto em vários outros animais. Isso levou a uma descrição bastante rica e complexa de como a visão biológica opera para resolver algumas tarefas de visão. Os resultados foram usados para o enriquecimento dos estudos sobre a imitação dos sistemas biológicos, em diferentes níveis de complexidade.

Outro campo relacionado com a visão computacional é o processamento de sinais. Vários métodos de processamento de sinais uni-variáveis, geralmente relacionados com o tempo, podem ser estendidos ao processamento de sinais com duas ou várias variáveis.

Além dos campos de pesquisa anteriores, vários outros tópicos também são estudados na visão computacional, tais como estatística, optimização e geometria computacional.

Reconhecimento

Um problema clássico da visão computacional e do processamento de imagens é determinar se uma imagem contém ou não um dado objecto, uma dada característica ou uma dada actividade. Tal tarefa pode ser resolvida de forma robusta e sem esforço humano, mas ainda não foi resolvida satisfatoriamente para o caso geral, objectos arbitrários em situações arbitrárias. Os métodos actuais conseguem, no máximo, resolver para objectos específicos, como poliedros, faces humanas, letras escritas à mão ou veículos; também em situações específicas, como iluminação bem definida, fundo fixo e pose dos objectos bem definida.

Diferentes variedades de problemas de reconhecimento são descritas na literatura:

> **Reconhecimento:** uma ou várias classes pré-definidas ou aprendidas de objectos podem ser reconhecidas, geralmente em conjunto com sua posição em imagens bidimensionais ou com sua pose em imagens tridimensionais.
>
> **Identificação:** uma instância individual de um objecto pode ser reconhecida, como a identificação de uma face ou de impressão digital, ou até mesmo a identificação de um veículo.
>
> **Detecção:** a imagem é digitalizada para uma condição específica, como a detecção de células ou tecidos anormais.

A organização de um sistema de visão computacional é dependente da aplicação. A implementação específica de tal sistema depende também se a sua funcionalidade é pré-

especificada ou se existe alguma parte de aprendizagem durante a operação. Existem, entretanto, funções típicas encontradas em vários sistemas de visão computacional:

Aquisição de imagem: uma imagem digital é produzida por um ou vários sensores. Dependendo do tipo do sensor, o resultado pode variar entre uma imagem bidimensional, uma cena tridimensional ou ainda uma sequência de imagens. Os valores dos pixéis geralmente indicam a intensidade da luz em uma ou várias faixas de cor (o que forma imagens em tom de cinza ou coloridas), mas também podem indicar valores físicos como profundidade e absorção ou reflexão das ondas electromagnéticas.

Pré-processamento: antes de um método de visão computacional ser aplicado numa imagem para extrair informação, é geralmente necessário processar a imagem para se assegurar que ela satisfaz as condições do método. Exemplos incluem remapeamento (para assegurar o sistema de coordenadas), redução de ruídos (para assegurar que as informações são verdadeiras) e aumento de contraste (para assegurar que as informações relevantes serão detectadas).

Extracção de características: características matemáticas da imagem em vários níveis de complexidade são extraídas. Exemplos básicos incluem detecção de bordas, cantos ou pontos. Exemplo sofisticados incluem detecção de textura, formato e movimento.

Detecção e segmentação: em algum ponto do processo uma decisão é feita sobre a relevância de regiões da imagem para processamento posterior. Exemplos incluem a selecção de regiões de interesse específicos e segmentação de uma ou mais regiões que contém um objecto de interesse.

Processamento de alto nível: neste ponto a entrada é geralmente um conjunto pequeno de dados. O processo posterior inclui a verificação da satisfação dos dados, a estimativa de parâmetros sobre a imagem e a classificação dos objectos detectados em diferentes categorias.

Neste trabalho focam-se dois pontos essenciais na caracterização de uma peça: a sua cor e a sua forma. Estes dois permitem uma correcta e inequívoca caracterização da peça em análise. Contudo a maioria da literatura aponta para métodos exigentes, pesados e complexos.

Pôr os computadores a ver não é tarefa complicada. Existem no mercado um sem número de câmaras de maior ou menor resolução, mais ou menos elaboradas. O problema reside na correcta interpretação desses dados. Um cérebro portanto. É neste ponto que se tem que

minimizar os custos de processamento, para assegurar uma tomada de decisão rápida e eficaz. Com uma imagem de baixa resolução, com muita sombra, pouco controlo do ambiente, é requerido um método altamente fiável e robusto. Quando se trata processamento em tempo real, então os problemas aumentam substancialmente. Nomeadamente na programação dos métodos correctos para lidar com estes problemas, assegurar uma minimização dos recursos computacionais utilizados e tempos de resposta compatíveis com uma análise fiável. Foi introduzido um conjunto de bibliotecas *open source*, desenvolvidas pela Intel, o OpenCV (*Open Source Computer Vision*), que contém inúmeros algoritmos e funções.

1.1. Instalação existente

No laboratório de robótica do IST existe um laboratório de automação contendo, entre outros equipamentos, um tapete rolante equipado com uma webcam da *LabTec*, autómatos e um conjunto de luzes e cilindros que podem ser actuados por este (ver Fig. 1). Existem também várias versões de softwares diversos (algoritmo para contagem de pintas em peças de dominó, reconhecimento de voz, detecção de cor usando outros espaços de cor, detecção de movimento, entre outros). Contudo, todos os programas foram desenvolvidos por alunos que estavam a aprender programação, muitas vezes com algoritmos desenvolvidos por eles mesmos, não tendo em atenção a minimização de recursos computacionais, resultado numa programação complicada de entender, com dificuldades na adição de novos módulos e muitas vezes com falta de robustez, resultante também do pouco conhecimento das ferramentas mais avançadas de programação.

(a) (b)

Fig. 1: (a) Bancada da instalação (b) Exemplo de uma imagem captada pela webcam

1.2. Objectivos

Nesta tese o objectivo foi unificar todas as aplicações num só programa, tendo sempre em atenção a rapidez de processamento e a integração de futuros projectos. Ou seja privilegiar não só a interface com o utilizador mas também tornar o programa o mais limpo possível para o programador futuro.

Um dos grandes problemas que condicionam a robustez e a eficácia dos programas em desenvolvimento tem a ver com a escassez dos recursos ao dispor, nomeadamente a fraca qualidade das câmaras usadas e uma iluminação deficiente. Face a estes factores era necessário encontrar soluções robustas mas rápidas.

Com esta instalação pretende-se simular um ambiente fabril, com todas as suas limitações e problemas já descritos, mas assegurando a eficiência e a fiabilidade das análises feitas, recorrendo às funções existentes no OpenCV.

Desenvolver então uma aplicação para detecção de cor, que permita uma correcta identificação desta em tempo real, e uma aplicação para detecção de forma dentro de cinco tipos de peças, sendo robusta a variações de rotação, translação, escala e ruído. Note-se que, para este trabalho, apenas foram usadas formas poligonais, ou seja, objectos com arestas rectas. Todos os métodos descritos só podem ser usados nestas condições, uma vez que foi considerado como descritor de forma os cantos desses objectos.

1.3. ESTRUTURA DA TESE

No capítulo 2 é apresentada brevemente a biblioteca de funções *OpenCV*, com alguns exemplos de aplicações que se podem desenvolver. As funções usadas são explicadas no Anexo 2.

No capítulo 3 é feita uma introdução aos espaços de cor, com especial enfoque para o usado neste programa, o LCH (lightness, crohma and hue).

No capítulo 4 é feita uma introdução à problemática da descrição de formas e mostra-se como se pode caracterizar uma forma recorrendo aos algoritmos de *Douglas-Peucker* e ao método de Triangulação de *Delaunay*.

No capítulo 5 é feita uma breve explicação aos métodos de classificação mais usados, redes neuronais e sistemas fuzzy, apesar de estes não serem aplicados no programa. É também

explicado o método do vizinho mais próximo, este sim o sistema de classificação usado, assim como o porquê da sua escolha em detrimento de outros.

No capítulo 6 é feita uma explicação do programa, identificando todos os passos assim como são apresentados exemplos do funcionamento do programa.

Finalmente no capítulo 7 são apresentados os resultados. Foram realizados testes com várias variáveis: três velocidades distintas do tapete, três tipos de complexidade de peças e diversas condições de iluminação. São apresentadas as taxas de sucesso da aplicação nas variadas condições, pretendendo explorar ao máximo as potencialidade e eventuais limitações do programa.

Em anexo está presente o manual do operador, onde são explicados todos os passos para cada uma das funcionalidades do programa e um breve manual do programador.

2. Open Cv[1]

OpenCV (significa *Open Source Computer Vision Library*. Originalmente, desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma, totalmente livre ao uso académico e comercial, para o desenvolvimento de aplicações na área de Visão Computacional. O *OpenCV* possui módulos de Processamento de Imagens e Vídeo I/O, Estrutura de dados, Álgebra Linear, *GUI* (Interface Gráfica do Utilizador) básica com sistema de janelas independentes, Controlo de rato e teclado, além de mais de 350 algoritmos de Visão. Entre as várias aplicações surgem:

-Interacção humano-computador;

-Identificação de objectos;

-Segmentação e reconhecimento;

-Detecção de caras;

-Detecção de movimentos;

-Seguimento de trajectórias;

-Previsão de trajectórias, entre outras.

Ou seja, muitos dos métodos usados para visão já se encontram recriados em C ou C++, assegurando a robustez e a rapidez de processamento. A resolução de um problema torna-se então em encontrar os métodos exactos que permitam uma maior fiabilidade. O programador pode-se concentrar em pesquisar o método mais correcto para a resolução desse problema ao invés de concentrar esforços na implementação de algoritmos, centrando o esforço numa análise do seu desempenho e não na programação desses mesmos métodos.

[1] Versão 2.0 em http://opencv.willowgarage.com/wiki/

3. Espaços de Cor

A cor é uma propriedade visual que é percepcionada pelo homem. Deriva do espectro da luz (distribuição da energia luminosa versus comprimento de onda) que interage no olho com a sensibilidade espectral dos receptores de luz. Categoria e características físicas da cor são também associadas a objectos, materiais, fontes de luz, etc., com base nas suas propriedades, tais como absorção, reflexão ou emissão. O olho humano detecta a cor num espectro de 380nm a 740nm de comprimento de onda.

3.1. Espaço de cor RGB

Neste espaço a cor é descrita pelas percentagens de cada cor primária (vermelho, verde e azul) que a constitui. Este espaço é baseado num referencial cartesiano cujo sub espaço de interesse é um cubo de aresta unitário, ou seja, os valores de cor encontram-se normalizados. Na Figura 2 pode-se visualizar o sub-espaço descrito anteriormente.

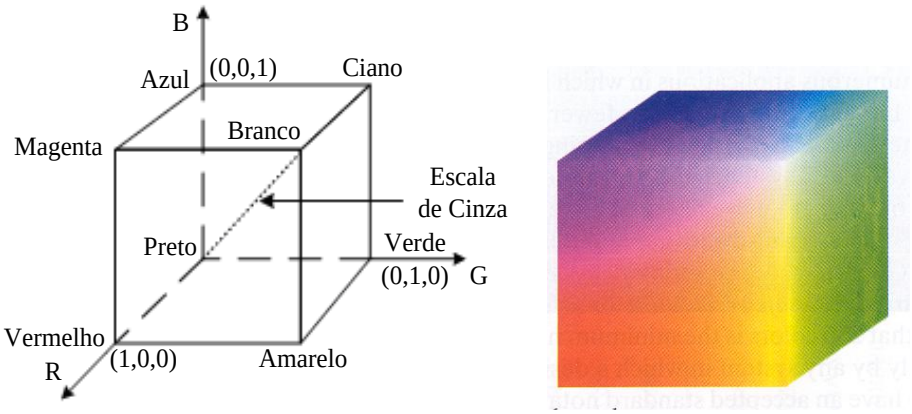


Fig. 2: Cubo do espaço de cor RGB

Nos vértices encontram-se as cores primárias (vermelho, verde e azul), as cores secundárias (magenta, amarelo e ciano, que se obtém através das cores primárias), o branco e o preto. É ainda de salientar que a diagonal que une os vértices correspondentes ao preto e ao branco representa a escala de cinza.

O espaço de cor RGB é utilizado em diversos dispositivos, sendo os mais usuais os monitores. O número de bits utilizado para representar cada pixel no espaço RGB é designado por profundidade do pixel. Usualmente recorre-se a uma profundidade de 8 bits para descrição de cada componente. Como tal para a descrição da cor são necessários 24 bits. Cada componente de cor (R, G e B) pode ser vista como uma imagem monocromática (escala de cinza) cujos pixéis apresentam valores de intensidades contidas no intervalo [0, …, 255], sendo que a união destas três componentes origina a imagem a cores.

3.2. Espaço de cor LCH

O espaço de cor LCH é composto por um disco, contendo todas as cores, com a particularidade da descrição da cor neste espaço ser efectuada através de coordenadas polares e não cartesianas.

A coordenada L mede a luminosidade do objecto. Esta coordenada pode apresentar valores entre 0 (preto) e 100 (branco).

Por sua vez a coordenada C mede a saturação, sendo o valor dado pela distância a que o ponto se encontra do eixo da luminosidade.

Finalmente a coordenada H mede o valor de tonalidade. Esta variável pode apresentar valores entre os 0º e 360º. Este intervalo contém 4 zonas distintas, caracterizadas pelas cores contidas em cada uma das zonas.

[0º, 90º]	-	Cores vermelha, laranja e amarelo.
[90º, 180º]	-	Cores amarela, amarelo-verde e verde.
[180º, 270º]	-	Cores verde, ciano (azul-verde) e azul.
[270º, 360º]	-	Cores azul, púrpura, magenta e vermelha.

Na figura seguinte apresenta-se o espaço de cor LCH:

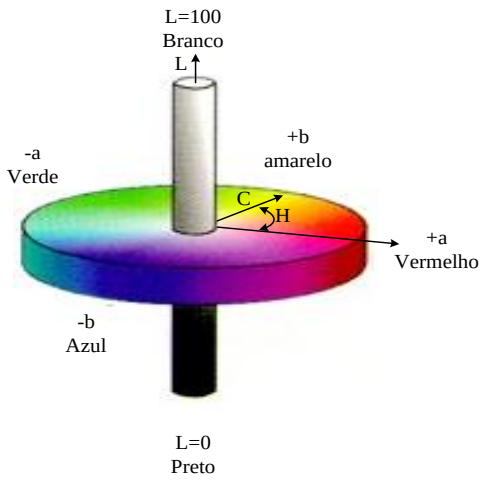


Fig. 3: Disco do espaço de cor LCH

A conversão de uma cor no espaço CIE(*Commission Internationale de l'Eclairage*) L* a* b* para o espaço CIE LCH é efectuada através das seguintes equações.

$H_{aux} = \tan^{-1}\ b^*, a^*$, Consoante a zona em questão.

$$\begin{cases} H_{aux} = \dfrac{H_{aux}}{\pi} \times 180 & \text{Se } H_{aux} > 0 \\ H_{aux} = 360 - \dfrac{|H_{aux}|}{\pi} \times 180 & \text{Se } H_{aux} \leq 0 \end{cases} \quad (3.1)$$

$$\begin{cases} L = L^* \\ C = \sqrt{a^{*2} + b^{*2}} \\ H = H_{aux} \end{cases} \quad (3.2)$$

Este espaço de cor é bastante útil porque, para distinguir as cores, apenas é necessária a variável H, já que as outras prendem-se com intensidade e tipo de cor, ou seja, mais clara ou mais escura, mais intensa ou mais esbatida, mas não distingue cores diferentes.

4. Análise de Forma

A descrição computacional da forma, assim como a sua manipulação, é complexa. Porque uma forma em si é complexa. Num mundo limitado por formas Euclidianas essa complexidade não existiria. Porém, com a inclusão de formas biológicas, árvores e as suas folhas, peixes, animais, flores, plantas, assim como formas naturais, linha costeira, pedras, cristais, aumenta a complexidade e a dificuldade de generalizar essas mesmas formas.

A motivação original da geometria era descrever e medir terrenos e edifícios (a palavra "geometria" deriva do Grego, e significava "*earth-measuring*"). Porém, nos tempos modernos, a descrição de formas tornou-se um dos maiores campos de estudo em disciplinas como visão computadorizada, robótica, análise e reconhecimento de padrões, computação gráfica, processamento de imagem, design e manufactura assistida por computador. Matérias como modelação geométrica, geometria computacional e modelação de sólidos, todas estão relacionadas com descrição de formas. Contudo, na maioria destas, as metodologias usadas são altamente intuitivas e tratadas de uma forma específica.

Deverá o computador digital e a programação influenciar o modo como se tenta a descrição de uma forma? Por um lado sim, na medida em que a intenção primária é especificar uma forma para análise, manipulação, renderização, etc., por computador. Contudo não podemos esquecer a generalidade matemática da análise e a necessidade de abstracção do real para a constituição de um método válido e aceite para a maior parte dos casos, mesmo que estes tenham origem em problemas concretos do "real".

O interesse na detecção de contornos tendo em vista a descrição de formas advém da redução da quantidade de informação a processar, e por poderem corresponder a limites físicos dos objectos observados, defeitos em peças, zonas de igual cor ou intensidade numa imagem. Estes revelam nas imagens variações bruscas dos níveis de brilho, correspondendo muitas vezes a descontinuidades na orientação ou na estrutura de uma superfície, oclusões em objectos, etc., sendo importantes na percepção da imagem como um todo.

Um contorno pode ser considerado como uma linha que delimita a fronteira de um determinado objecto ou de uma determinada área numa imagem. É caracterizado como uma variação rápida dos níveis de brilho. Contudo, numa imagem captada por uma câmara, ao invés de existir uma linha definida e demarcada, há uma zona onde gradualmente esta variação se dá, como se pode observar na figura 4. É necessário então recorrer a ferramentas matemáticas, como por exemplo o cálculo da segunda derivada, para detectar estas variações.

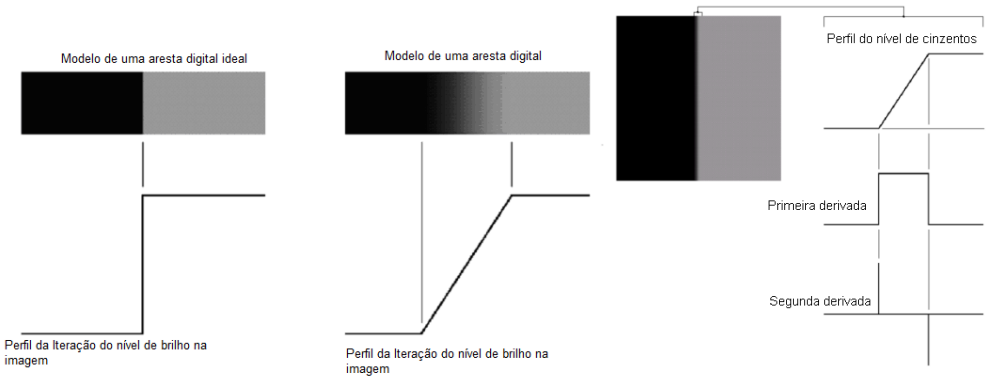


Fig. 4: a) Exemplos de descontinuidades em linhas de contorno; b) 1ª e 2ª derivada de uma linha de contorno

4.1. Descritores de contornos

Os descritores da forma podem ser classificados pela sua invariância no que diz respeito às transformações permitidas na definição associada da forma. A maioria é invariante no que diz respeito à sua congruência, resultando nos mesmos descritores independentemente da sua orientação (formas que podem sofrer translação, rotação ou espelho). Exemplo destes descritores são os baseados no cálculo dos momentos.

Outra classe de descritores são os chamados intrínsecos. Estes são invariantes à isometria. Não se alteram com diferentes isometrias próprias da forma. Têm como vantagem poderem ser aplicados a formas deformadas, uma vez que estas deformações são quase isométricas. São baseadas em medidas geodésicas ao longo de uma superfície de um objecto ou noutras características isométricas invariantes.

Existem outros, baseados por exemplo em análise gráfica, como a linha média, ou o *Reeb graph*, que captam informação geométrica ou topológica e simplificam a representação, mas não podem ser facilmente analisadas como os que descrevem uma forma como um vector de números.

Outro tipo de descritores muito usados são os chamados descritores geométricos. Porém, são rudimentares, muitas vezes variantes com escala, rotação ou translação e não permitem uma inequívoca distinção entre formas distintas. Como exemplo destes:

-Diâmetro efectivo: Diâmetro de um círculo, com a mesma área do objecto;

-Circularidade: Razão entre a área de sobreposição do objecto ao círculo equivalente e a área total do objecto;

-Irregularidade: complemento de um da razão entre os perímetros do círculo equivalente e do objecto;

-Compacticidade: Razão entre o quadrado do perímetro e a área do objecto

Como se pode ver existem várias abordagens a este problema. Deve-se analisar o pretendido, o tipo de complexidade da forma, as ferramentas que se pretende usar, para se conseguir extrair os descritores que permitam uma boa caracterização.

É apresentado de seguida as ferramentas que permitiram usar como característica de uma forma os seus cantos.

4.2. Simplificação de Linhas Poligonais

Muitas entidades do mundo real podem ser modeladas como linhas ou, mais genericamente, polígonos. Essas entidades são frequentes em bases de dados geográficas, onde correspondem tipicamente a cerca de 80% do volume de dados vectoriais [10]. Por isso, o problema de simplificação de linhas é particularmente importante, sendo estudado intensivamente desde os anos 60, quando ocorreram as primeiras experiências com o uso de instrumentos de transcrição de mapas para o computador, como o scanner.

No processo de digitalização de linhas são frequentemente introduzidos vértices em excesso, vértices que, se descartados, não provocariam uma alteração visual perceptível na poligonal. Assim, um primeiro objectivo para algoritmos de simplificação de linhas é "limpar" a poligonal de pontos claramente desnecessários, do ponto de vista de sua visualização [5], mantendo a qualidade da aparência gráfica [4] [6].

Outro objectivo é o de gerar uma nova versão da linha, mais adequada para a representação do mesmo fenómeno geográfico numa outra escala, menor que a original. Neste caso, obtém-se uma generalização da linha [10]. Numa extensão deste pormenor, existe o interesse em organizar os vértices da poligonal de tal forma que seja possível produzir, dinamicamente, versões generalizadas adequadas para uma escala definida no momento da visualização [7],

conseguindo portanto gerar múltiplas representações geométricas para o mesmo fenómeno sem introduzir dados redundantes. No entanto, a utilização de métodos e algoritmos desenvolvidos originalmente apenas a pensar na redução do número de vértices da linha podem não ser adequados para alcançar o objectivo de generalização [8], em geral por não conseguirem uma boa representação geométrica, e portanto devem ser analisados cuidadosamente quanto a este aspecto.

Assim, o problema de simplificação de linhas consiste em obter uma representação mais grosseira (formada por menos vértices, e portanto mais compacta) de uma poligonal a partir de uma representação mais refinada, atendendo a alguma restrição de aproximação entre as duas. Essa restrição pode ser definida de várias maneiras [9], mas é em geral alguma medida da proximidade geométrica entre as poligonais, tais como o máximo deslocamento perpendicular permitido (ver Fig 5(a)) ou o mínimo deslocamento angular permitido (ver Fig 5(b)). Na Figura 5(a), o vértice 2 será mantido, uma vez que a distância entre ele e a recta que passa pelos vértices 1 e 3 é superior à permitida. Na Figura 5(b), o vértice 3 será eliminado, uma vez que o ângulo $\widehat{234}$ é menor que o mínimo tolerável. Uma alternativa mais rara é a área entre as poligonais 5(c), onde se estabelece um limite para ao deslocamento de área.

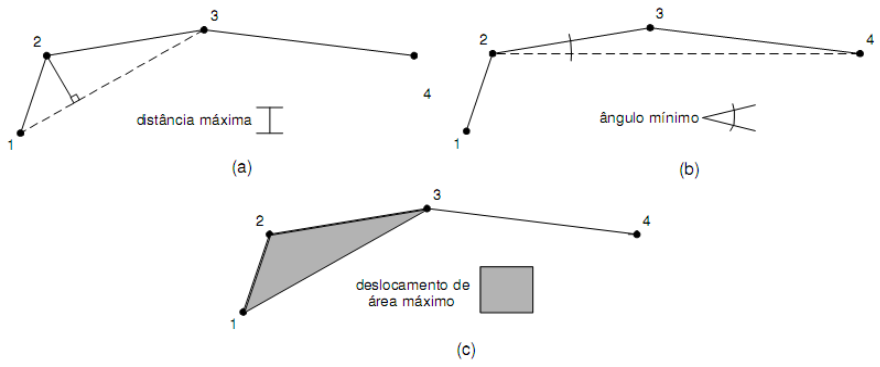


Fig. 5: Diferentes medidas usadas em simplificação de linhas

De todas as medidas possíveis, a mais utilizada é a distância perpendicular. O conceito de gama de tolerância, apoiado no cálculo de distâncias perpendiculares, é utilizado em grande parte nos algoritmos de simplificação que serão apresentados a seguir. Um problema eventualmente abordado na literatura é a escolha do parâmetro de tolerância (ε), e a sua correlação com a escala da representação simplificada.

Um critério interessante para a determinação da tolerância é o que usa o tamanho do menor objecto visível numa determinada escala [9]. Este tamanho pode ser dado em termos de uma

distância medida no espaço de coordenadas do mapa plotado, ou seja, em milímetros do papel, independente da escala utilizada. Assim, é definida uma correspondência linear entre a escala e a tolerância adoptada.

Grande parte dos algoritmos de simplificação de poligonais necessita realizar de maneira eficiente cálculos de distância entre um ponto dado e uma recta definida por outros dois pontos. A maneira mais interessante de calcular essa distância é utilizar o produto vectorial, para determinar a área S do triângulo formado por um ponto A e uma recta definida por outros dois (B e C). Assim, a distância do ponto A à recta definida pelos pontos B e C pode ser calculada como:

$$d = \frac{2\,S}{dist(B,C)} \qquad (4.1)$$

Onde dist (B, C) é a distância euclidiana entre os pontos B e C.

Embora existam muitos algoritmos de simplificação de linhas, envolvendo variados critérios de aproximação e estratégias de processamento, um deles se destaca pela ampla aceitação. Foi proposto em 1973 por *Douglas* e *Peucker*, e é reconhecidamente o melhor em termos de preservação das características da poligonal original [10] [11].

4.3. Algoritmo de Douglas-Peucker.

O algoritmo é recursivo, e a cada passo processa o intervalo de pontos contido entre um vértice inicial (denominado âncora) e um vértice final (denominado flutuante). É estabelecido um corredor de largura igual ao dobro da tolerância, formando duas faixas paralelas ao segmento entre o âncora e o flutante. A seguir, são calculadas as distâncias de todos os pontos intermediários ao segmento básico, ou seja, contidos entre a âncora e o flutuante. Caso nenhuma das distâncias calculadas ultrapasse a tolerância, ou seja, nenhum vértice fica fora do corredor, então todos os vértices intermediários são descartados. Caso alguma distância seja maior que a tolerância, o vértice mais distante é preservado, e o algoritmo é reiniciado em duas partes: entre o âncora e o vértice mais distante (novo flutuante), e entre o vértice mais distante (novo âncora) e o flutuante. De acordo com este processo, os pontos tidos como críticos para a geometria da linha, a cada passo, são mantidos, enquanto os demais são descartados. Tome-se como exemplo a figura 6, para explicar o funcionamento do algoritmo.

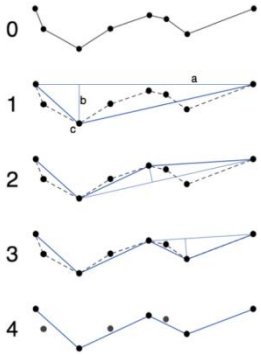


Fig. 6: Exemplo da aplicação do algoritmo a uma curva

Seja uma curva composta por segmentos de recta. Tendo em vista a minimização do número de pontos constituintes da curva, une-se o 1º com o último ponto. Mede-se então a distância entre cada um dos pontos à nova recta. Neste caso a distância máxima é da recta ao ponto *c*. Une-se então o 1º ponto ao ponto *c* e esse mesmo ponto ao último ponto. Volta-se a medir as distâncias, seguindo o mesmo método. Todos os pontos que se encontrarem a uma distância inferior a um valor imposto ξ podem-se excluir da curva, dando origem à curva em 4, em que todos os pontos se encontram dentro da dita distância ξ e por consequência a curva é caracterizada por um número inferior de pontos.

O resultado deste algoritmo é aclamado pela literatura como sendo o que mais respeita as características (ou, como no título do artigo de *Douglas* e *Peucker*, a "caricatura") da linha cartográfica [11] [13].

Assim, este algoritmo veio a ser a escolha dos programadores de software comercial na implementação de funções de simplificação de linhas para processamento pós-digitalização [9], ou seja, para limpeza de vértices desnecessários. O uso do algoritmo *Douglas-Peucker*, no entanto, é comprometido pelo seu comportamento em situações de generalização mais radical [12], ou seja, com tolerâncias maiores. Conforme a situação, o algoritmo pode ser levado a escolher vértices que terminam por deixar a linha com uma aparência pouco natural, com tendência a apresentar "picos" (como no exemplo da Figura 7, entre os vértices 17, 24 e 29), com ângulos agudos e mudanças bruscas de direcção.

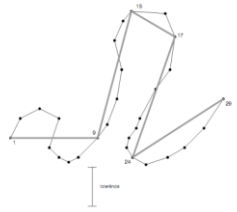


Fig. 7: Exemplo da aplicação do algoritmo a uma curva

4.4. Método de triangulação de Delaunay

Como descrever então uma forma com base nos seus cantos? As relações entre lados, distância dos vértices ao centro geométrico, razão entre a área e o perímetro, todos estes são métodos usados para a caracterização de formas geométricas. Contudo não são muito robustos. No artigo[20], é descrito um método usado para comparação de formas complexas como folhas ou peixes, usando a triangulação de *Delaunay* e criando um histograma dos ângulos dos triângulos gerados. Este algoritmo é bastante robusto, não só porque elimina potenciais discrepâncias a nível do posicionamento dos cantos como também se mostra invariante à rotação, translação e escala.

Este método, criado pelo matemático russo *Boris Delaunay*, consiste numa triangulação em que nenhum vértice de outro triângulo pertença a um círculo traçado com base nos vértices.

Sejam p_i, p_j e $p_k \in S$, sendo S um espaço bidimensional. p_i, p_j e p_k são vértices de um triângulo de Delaunay se e só se o círculo que passa por p_i, p_j e p_k é vazio (isto é não contém outro ponto de S). O algoritmo consiste numa análise de força bruta aos possíveis triângulos que se podem gerar, eliminando todos os que não constituem um triângulo de *Delaunay*.

De seguida faz-se um levantamento aos ângulos criados pelo método, arrumando-os em *bins* de 10º. Obtém-se então um histograma de gamas de ângulos. Isto permite eliminar os tais erros de posicionamento dos cantos, uma vez que a diferença dos ângulos criada é absorvida pelo histograma.

Surge então a necessidade de comparar esse histograma criado com os presentes na base de dados. O método usado chama-se a distância Qui Quadrado.

Sejam dois histogramas, S e M, e n o número de bins no histograma. A distância *Qui Quadrado* é então definida por [21]:

$$\chi^2(S,M) = \sum_{i=1}^{n} \frac{(S_i - M_i)^2}{(S_i + M_i)} \qquad (4.2)$$

Ou seja, passámos de *n* pontos de contorno para o número de cantos da peça, estabelecemos uma triangulação e reduzimo-la primeiramente a um histograma e posteriormente a um valor. Sendo S o histograma da peça a analisar e M o número de histogramas de ensinamento

estamos em condições de usar o método do vizinho mais próximo para classificação de peças, um método computacionalmente mais leve que qualquer dos sistemas de classificação mencionados no capítulo 5.

4.5. Importância destas aplicações na indústria

Durante décadas o olho humano foi usado para monitorizar a saída de sistemas de montagem contínuos. Os seres humanos são "sistemas de inspecção" flexíveis, robustos, e fáceis de treinar, mas não são rápidos nem exactos. A tecnologia da visão assistida por computador foi introduzida nos anos 80 para superar as limitações da visão humana. Os descendentes de hoje destes sistemas adiantados (câmaras com autómatos programáveis, com softwares próprios, recurso a sistemas de laser, raio-X entre outros) são muito mais fáceis de programar e usar, mais poderosos, baratos e de dimensões mais reduzidas.

A visão industrial transformou-se numa parte integrante de muitos processos de produção. Os procedimentos automatizados no controlo óptico de qualidade tornam-se cada vez mais importantes a fim de conseguir consistentemente e por um preço razoável, padrões de alta qualidade. Um sistema de visão tem de cumprir exigências rigorosas a respeito da precisão, da velocidade, da repetitividade, da facilidade em ser validado, da facilidade da integração e de operação. Há ainda um potencial enorme em utilizar a tecnologia da visão por computador a fim de tornar os processos de produção mais eficientes e os seus produtos mais rentáveis.

Além deste campo, os sistemas de escolha automáticos também se encontram equipados com sistemas avançados de visão, tornando o processo mais fiável e rápido. A robótica de manipulação também sofreu grandes avanços, na fiabilidade, repetitividade e rapidez de execução, mais uma vez devido à incorporação deste tipo de sistemas.

Outras áreas como por exemplo a segurança rodoviária e em particular variados campos de intervenção médica beneficiam grandemente do interesse e desenvolvimento de ferramentas cada vez mais robustas e fiáveis.

5. Sistemas de classificação

Decidir é um dos actos mais nobres que realizamos. Decidimos quando vemos, falamos, quando reconhecemos pessoas ou objectos, quando interpretamos o meio que nos rodeia e actuamos sobre ele. Serão as máquinas capazes de decidir? E de apreender a decidir? São perguntas fascinantes. Ambas têm resposta positiva, desde que as interpretemos em sentidos precisos, mais limitados do que quando nos referimos à aprendizagem e capacidade de decisão humanas.

5.1. Redes neuronais [17] [18]

O sistema nervoso central dos animais recebe do exterior, armazena, processa e transmite informação ao exterior. A observação do desempenho que este apresenta tem revelado uma extraordinária capacidade para executar rápida e eficientemente tarefas de grande complexidade tais como o processamento em paralelo da informação, a memória associativa e a capacidade para classificar e generalizar conceitos. Estes factos têm servido de motivação quer para o estudo detalhado da constituição do cérebro quer para a sua mimetização na concepção de sistemas com as capacidades atrás referidas e designados por redes neuronais artificiais (*Artificial Neural Nets*). As redes neuronais artificiais (abreviadamente, NN) têm sido utilizadas na modelação de memória associativa, reconhecimento de padrões, representação de funções booleanas, representação de funções contínuas, previsão de séries temporais, optimização etc. Refiram-se algumas das áreas em que são aplicadas:

(a) Física de alta energia;

(b) Processos industriais, robótica, indústria de defesa, indústria aeroespacial, indústria de telecomunicações, indústria electrónica, indústria automóvel, indústria de transportes, indústria de prospecção petrolífera;

(c) Medicina e biomedicina;

(d) Reconhecimento de texto, imagem e voz.

(e) Economia e gestão, análise financeira e banca;

(f) Seguros;

(g) Entretenimento.

O córtex cerebral é constituído por unidades celulares independentes designadas por neurónios que podem comunicar entre si através das chamadas ligações sinápticas ou sinapses.

O primeiro modelo lógico-matemático de um neurónio foi desenvolvido por *McCulloch* e *Pitts* em 1943 [15] tendo sido designado por TLU (*Thresh-Logic Unit*). Consiste numa unidade computacional que opera com sinais binários (0/1).

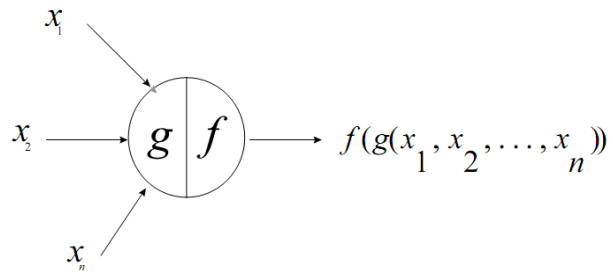


Fig. 8: Exemplo de um neurónio artificial

No entanto a pretensão de modelar uma unidade computacional verdadeiramente elementar aconselha a escolha de uma função g mais simples (campo escalar que traduza uma aplicação linear).

Os nós e as arestas designam-se, respectivamente, neurónios (ou unidades computacionais) e sinapses. Neurónios de entrada são aqueles que não têm sinapses dirigidas para os mesmos. Neurónios de saída são aqueles que não possuem sinapses de saída.

As redes neuronais podem ser classificadas, em termos de arquitectura, em:

Redes planares – consiste numa estrutura bidimensional de neurónios escondidos e periféricos.

Redes constituídas por camadas - As redes neuronais artificiais planares podem ser constituídas por diferentes camadas dispostas paralelamente. A primeira é designada por camada de entrada e a última por camada de saída. As camadas intermédias são designadas por camadas escondidas. As redes constituídas por camadas constituem casos particulares das redes planares.

Em termos de processo de aprendizagem:

- Aprendizagem supervisionada
- Aprendizagem não supervisionada

A aprendizagem sem supervisão é aplicada em sistemas de memória associativa e de reconhecimento de padrões, essencialmente. Nestas redes a aprendizagem é realizada sem se conhecer antecipadamente as respostas consideradas correctas.

5.2. Sistemas de decisão fuzzy

O conceito "fuzzy" pode ser entendido como uma situação onde não podemos responder simplesmente "Sim" ou "Não". Mesmo conhecendo as informações necessárias sobre a situação, dizer algo entre "sim" e "não" como por exemplo "talvez", "quase", torna-se mais apropriado.

As primeiras noções da lógica dos conceitos "vagos" foi desenvolvida por um lógico da Polónia, Jan Lukasiewicz (1878-1956), em 1920 que introduziu conjuntos com graus de pertença sendo 0, ½ e 1 e, mais tarde, expandiu para um número infinito de valores entre 0 e 1.

A primeira publicação sobre lógica "fuzzy" data de 1965, quando recebeu este nome. O autor foi Lotfi Asker Zadeh (ZAH-da), professor em Berkeley, Universidade da Califórnia.

Zadeh criou a lógica "fuzzy" combinando os conceitos da lógica clássica e os conjuntos de Lukasiewicz, definindo graus de pertença.

Entre 1970 e 1980 as aplicações industriais da lógica "fuzzy" aconteceram com maior importância na Europa e após 1980, o Japão iniciou o seu uso com aplicações na indústria. Algumas das primeiras aplicações foram num tratamento de água feito pela Fuji Electric em 1983 e pela Hitachi num sistema de metro inaugurado em 1987. Por volta de 1990 é que a lógica "fuzzy" despertou um maior interesse em empresas dos Estados Unidos.

Devido às inúmeras possibilidades práticas dos sistemas "fuzzy" e ao grande sucesso comercial de suas aplicações, a lógica "fuzzy" é considerada hoje uma técnica "standard" e tem uma ampla aceitação na área de controlo de processos industriais.

Na teoria clássica, os conjuntos são denominados **"crisp"** e um dado elemento do universo em discurso (domínio) **pertence** ou **não pertence** ao referido conjunto.

Na teoria dos conjuntos **"fuzzy"** existe um grau de pertença de cada elemento a um determinado conjunto. Por exemplo considere os conjuntos abaixo:

· Conjunto das pessoas que pagam uma renda elevada.

· Conjunto das pessoas altas.

Podemos verificar que não existe uma fronteira bem definida para decidirmos quando um elemento pertence ou não ao respectivo conjunto nos exemplos acima.

Com os conjuntos "fuzzy" podemos definir critérios e graus de pertença para tais situações.

A função característica (crisp sets) pode ser generalizada de modo que os valores designados aos elementos do conjunto universo U pertençam ao intervalo de números reais de 0 a 1 inclusive, isto é [0,1].

Estes valores indicam o grau de pertença dos elementos do conjunto U em relação ao conjunto A, isto é, *quanto é possível* para um elemento x de U pertencer ao conjunto A.

Tal função é chamada de função de pertença e o conjunto A é definido como *"conjunto Fuzzy"*.

A teoria dos conjuntos fuzzy providencia um método de cálculo para lidar com essa informação linguisticamente, e efectua uma computação numérica usando etiquetas linguísticas estipuladas por funções pertença [20] e regras if-then. Usualmente o sistema fuzzy não possui a adaptabilidade de lidar com mudanças nos ambientes externos apesar de ser estruturado com regras if-then.

Os conjuntos fuzzy e os operadores fuzzy são os sujeitos e verbos da lógica fuzzy. As regras if-then são usadas para formular os condicionamentos que compreendem a lógica fuzzy. Uma regra if-then tem a seguinte forma

Se *x* é A então *y* é B.

Onde A e B são os valores linguísticos definidos por conjuntos fuzzy no universo de discurso X e Y, respectivamente. A parte “x é A” é chamada de antecedente e a parte “y é B” de

consequente. Sendo o valor de x representado por um número qualquer o antecedente é a interpretação que retorna um valor entre 0 e 1. Sendo B representado como um conjunto fuzzy, o consequente é, então, uma atribuição que atribui B à variável de saída y. Em geral a entrada de uma regra if-then é o valor da variável x sendo a saída composta pelo conjunto fuzzy.

A Interpretação de uma regra if-then é constituída por duas partes. Primeiro avaliar os antecedentes, que envolve "fuzzificar" a entrada e aplicar as operações necessárias. A segunda parte consiste em aplicar o resultado ao consequente. O antecedente de uma regra pode conter múltiplas partes que terão que ser calculadas em simultâneo e resolvidas entre si de modo a produzirem um único valor.

É necessário, normalmente, o uso de duas ou mais regras que terão, para cada uma, uma saída correspondente a um conjunto fuzzy. Esses conjuntos fuzzy são depois agregados para produzirem apenas uma saída. Para agregar as regras é usado um sistema de inferência fuzzy (ou FIS).

Um sistema de inferência fuzzy é um sistema que agrega as entradas e calcula a saída usando os conceitos da teoria dos conjuntos fuzzy, regras if-then e o raciocínio fuzzy. Essa saída é depois a base das decisões a serem tomadas. Em geral um FIS implementa um sistema não linear entre as entradas e as saídas. Este sistema é conseguido através de regras if-then que descrevem o seu comportamento para uma determinada situação

5.3. Método do vizinho mais próximo

É um sistema de classificação mais simples que os outros referidos (redes neuronais ou redes fuzzy). Neste método, a classificação de uma observação *x* consiste em fazer uma busca exaustiva para todos os padrões de treino como vista a determinar o padrão cuja distância a x é menor. A classe desse padrão é a classe em que *x* é classificado. A figura 9 ilustra a utilização do método do vizinho mais próximo na classificação de um padrão *x*. São identificadas na figura algumas das distâncias que é necessário calcular, assim como o padrão da observação *x*. O padrão *x* e classificado na classe 1.

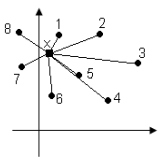


Fig. 9: Exemplo do método do vizinho mais próximo.

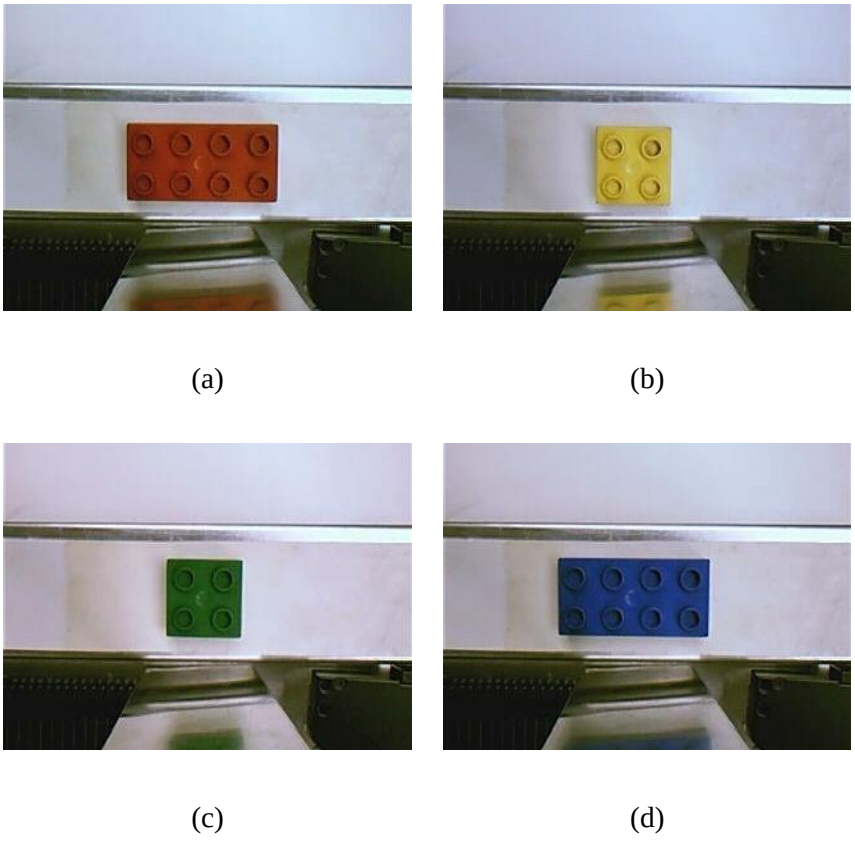
6. Explicação do programa

6.1. CONSTRUÇÃO DA TABELA DE REFERÊNCIA

O treino do programa não é mais que a construção de uma base de dados, com os valores de treino (ou os histogramas) de cada uma das classes. É necessário criá-la para a cor e para a forma.

6.1.1. COR

Usando as imagens da Fig 10 é calculado e armazenado o valor de *h* para cada cor, vermelho, azul, amarelo e verde. Isto foi feito offline e gravado num ficheiro txt, que é lido cada vez que se inicia o programa.



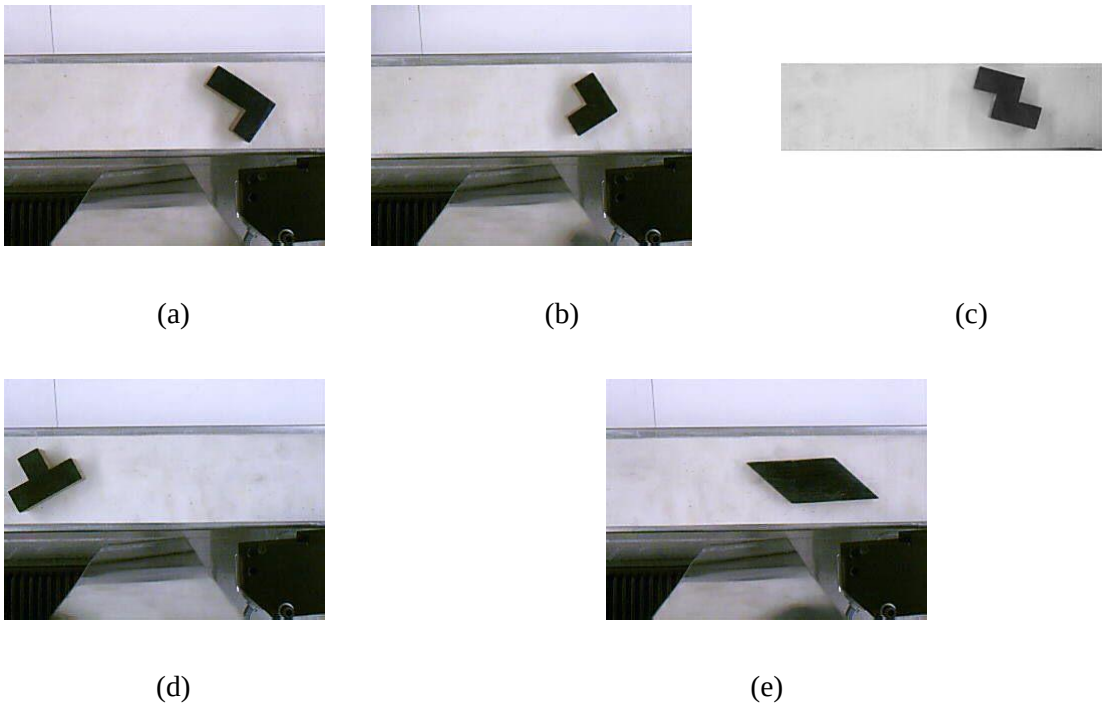
(a) (b)

(c) (d)

Fig. 10: Exemplo de peças usadas na caracterização por cor. (a) vermelho (b) amarelo (c) verde (d) azul

6.1.2. FORMA

O processo é semelhante. Foram aplicados os mesmos métodos usados na análise em tempo real, descritos na secção 4, e os histogramas de cada peça (ver figura 11) calculados e armazenados num ficheiro txt também, que é lido cada vez que se inicia o programa.



(a) (b) (c)

(d) (e)

Fig. 11: Peça da classe 1(a), 2(b), 3(c), 4(d) e 5(e)

Este programa pretende ser fácil de utilizar e de incluir novas aplicações. Mas também permite adicionar peças e cores à base de dados, através da introdução de uma fotografia, assegurando porém que esta seja tirada sob um fundo branco, já que o programa não tem a capacidade de eliminar qualquer tipo de fundo. Este processo é explicado na secção "Manual de Utilizador".

6.2. Descrição do programa

A interface do programa é composta por uma janela de visualização (ver Fig. 12), que permite visualizar a imagem captada pela câmara, quatro *sliders* para afinação da área de interesse, uma caixa de texto contendo a decisão quer da cor quer da forma e um slider para ajuste do *threshold*.

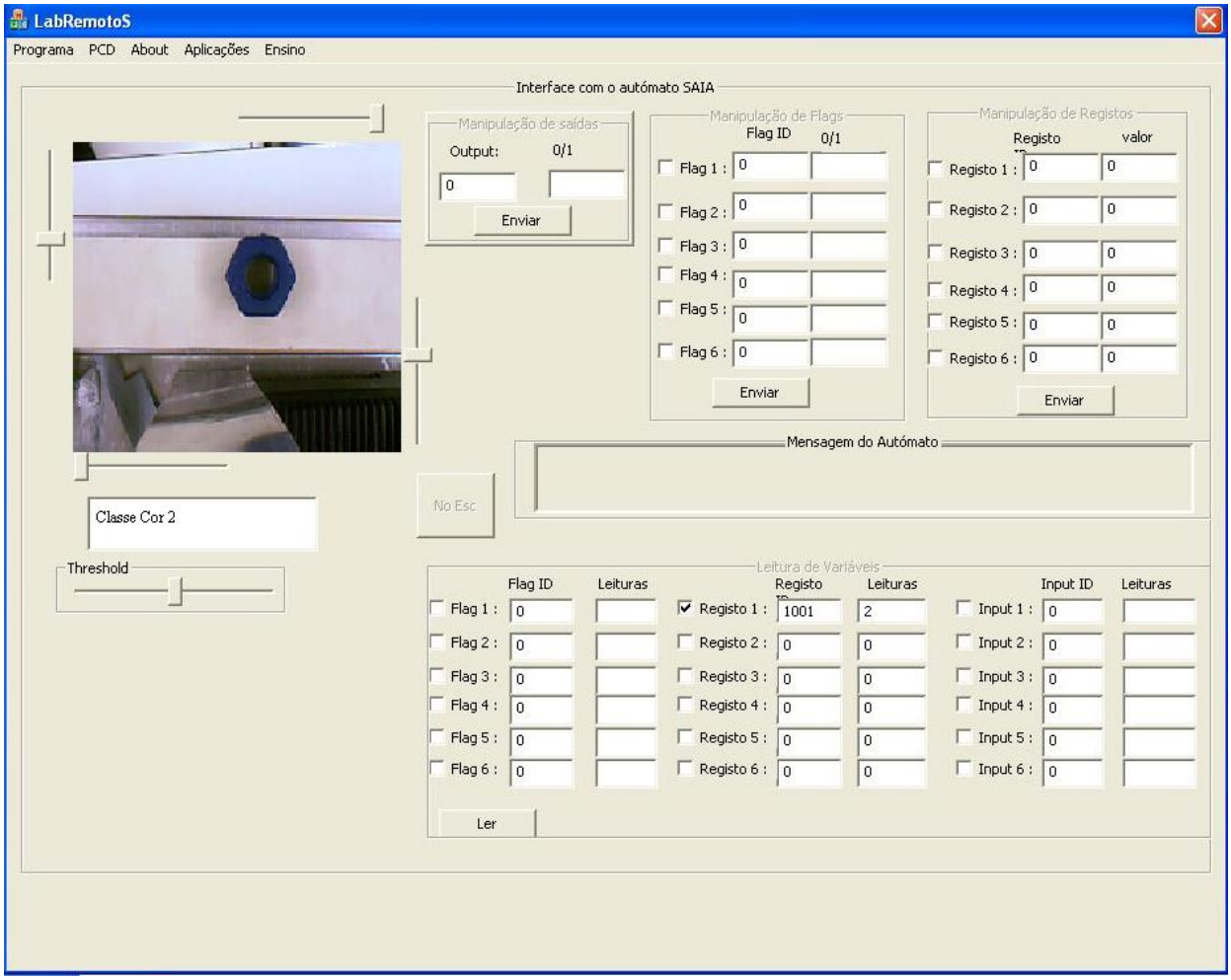


Fig. 12: Interface do programa

Aquando da inicialização do programa, é feito um ajustamento da janela de visualização, via *sliders*, para definir uma região de interesse (ROI), ou seja, todos os tratamentos de imagem serão efectuados somente nessa zona.

Quer para a aplicação de cor, quer para a forma, é feita uma cópia da imagem adquirida da câmara para um vector do tipo IplImage (formato aceite pelo OpenCV), eliminando já a informação respeitante à área fora da definida. Fica-se então com a informação RGB da ROI. É feita uma passagem para preto e branco e uma binarização, com um *threshold* pré-definido no programa (este valor pode ser alterado *in loco* no programa, para salvaguardar possíveis

alterações de iluminação). Na imagem binarizada é efectuado um varrimento para detectar a localização da peça, assegurando que o campo de visão da câmara abrange a totalidade da peça. Isto é obtido através do cálculo do contorno e com a aplicação do método de Douglas-Peucker a esses pontos.

É neste ponto que o procedimento difere consoante a aplicação.

6.2.1. COR

-É feita uma média do valor *H* de todos os pixéis pertencentes à peça em análise;

-Esse valor é comparado com os de ensino, usando o método do vizinho mais próximo para a tomada de decisão;

-Essa decisão é enviada para o autómato.

6.2.2. FORMA

-As coordenadas dos cantos são enviadas para o método de triangulação de Delaunay;

-Cada ângulo de cada triângulo é guardado em *bins* de 10º, gerando um histograma desses ângulos;

-Esse histograma é comparado com os de treino, através do método de Chi-Square. Este método gera cinco valores, sendo a classe da peça correspondente ao menor valor;

-Essa decisão é enviada para o autómato.

Em ambos os casos, aquando da tomada de uma decisão, esta permanece visível para o utilizador até outra peça entrar no campo visível da câmara.

7. Apresentação e análise de resultados

7.1. Introdução

Para a obtenção destes resultados foram efectuados, para o estudo da caracterização da forma, 350 ensaios para cada peça, 100 para velocidade do tapete lenta, 100 para velocidade do tapete média, 100 para velocidade do tapete rápida e 50 para análise de fotografias, perfazendo um total de 1750 análises. Para a análise de cor foram efectuados 100 ensaios para velocidade do tapete lenta, 100 para média e 100 para rápida, perfazendo um total de 1200 ensaios. Foi registada cada uma das decisões tomadas e condensadas em matrizes de confusão.

7.2. Treino

Na tabela 1 estão descritos os histogramas de ensino para cada classe. Na tabela 2 os quatro parâmetros *h* de cada cor. Na Fig. 13 encontra-se o contorno encontrado pelo programa assim como a sua triangulação para cada peça usada no treino.

Tabela 1: Histogramas de ensino para cada classe

Bins	C 1	C 2	C 3	C 4	C 5
[0 -10 [º	0	0	0	0	0
[10 -20 [º	0	0	0	1	0
[20 -30 [º	3	0	0	2	0
[30 -40 [º	0	2	2	1	0
[40 -50 [º	5	8	14	12	4
[50 -60 [º	0	0	0	0	0
[60 -70 [º	2	0	0	0	0
[70 -80 [º	0	0	0	0	0
[80 -90 [º	1	2	2	3	0
[90 -110 [º	3	3	6	3	2
[110 -120 [º	0	0	0	1	0
[120 -130 [º	1	0	0	0	0
[130 -180 [º	0	0	0	0	0

Tabela 2: Valor de *h* para cada uma das cores

Cor	Valor do parâmetro h
Amarelo	23.64
Azul	109.52
Verde	55.57
Vermelho	8.61

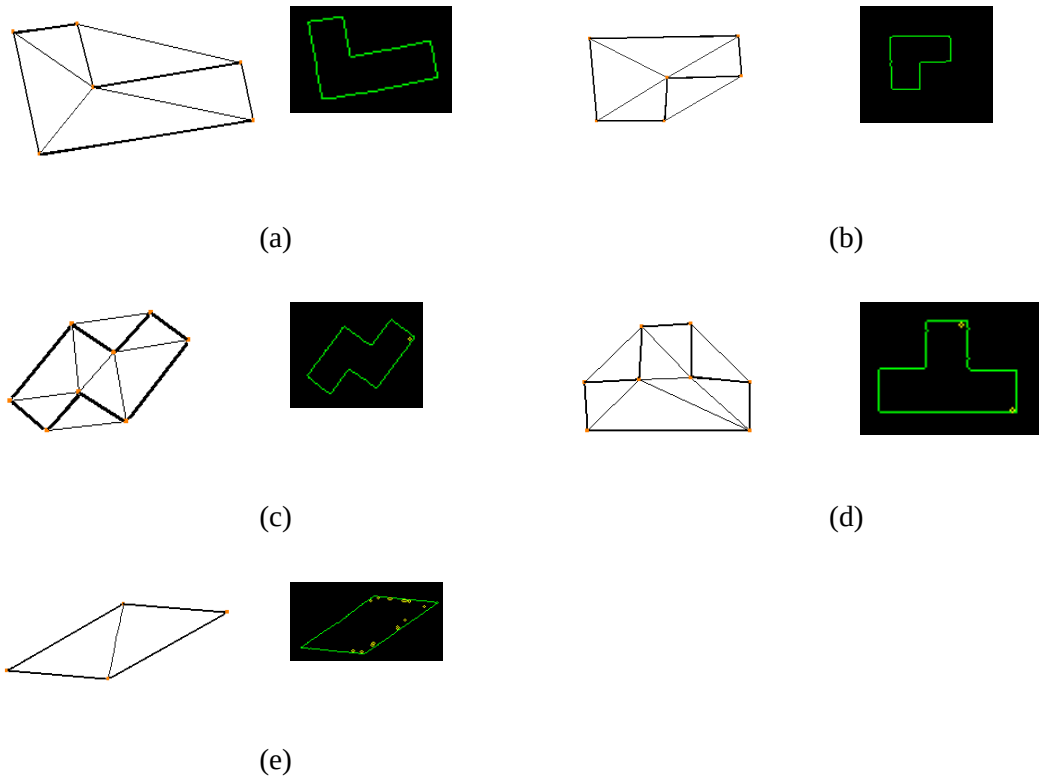


Fig. 13: Exemplo de uma triangulação e de um *plot* dos pontos de contorno encontrados para a classe 1(a), 2(b), 3(c), 4(d) e 5(e)

7.3. Resultados obtidos na caracterização de forma

Na tabela 3, 4, 5 e 6 encontram-se sumarizadas as classificações para cada classe, para diferentes velocidades do tapete e para a análise via fotografia.

Tabela 3: Classificações para velocidade de tapete lenta

Classe	1	2	3	4	5
1	87	11	0	0	2
2	5	89	1	0	5
3	5	6	84	0	5
4	0	7	8	78	7
5	0	0	0	0	100

Tabela 4: Classificações para velocidade de tapete média

Classe	1	2	3	4	5
1	83	11	0	0	6
2	4	85	2	0	9
3	6	6	81	0	7
4	0	6	9	75	10
5	0	0	0	0	100

Tabela 5: Classificações para velocidade de tapete rápida

Classe	1	2	3	4	5
1	80	10	1	0	9
2	4	84	2	0	10
3	6	6	77	0	11
4	0	7	8	75	10
5	0	0	0	0	100

Tabela 6: Classificações para análise de fotografias

Classe	1	2	3	4	5
1	41	0	0	1	8
2	1	44	0	0	5
3	0	2	43	4	1
4	0	4	0	39	7
5	0	0	0	0	50

7.4. Resultados obtidos na caracterização de cor

Nas tabelas 7, 8 e 9 encontram-se sumarizadas as classificações de cada cor para velocidades diferentes do tapete.

Tabela 7: Classificações para velocidade de tapete lenta

Classe	Verde	Vermelho	Azul	Amarelo
Verde	95	2	3	0
Vermelho	2	94	0	4
Azul	3	3	94	0
Amarelo	3	1	0	96

Tabela 8: Classificações para velocidade de tapete média

Classe	Verde	Vermelho	Azul	Amarelo
Verde	92	5	3	0
Vermelho	2	95	0	3
Azul	0	4	96	0
Amarelo	3	0	4	93

Tabela 9: Classificações para velocidade de tapete rápida

Classe	Verde	Vermelho	Azul	Amarelo
Verde	90	6	4	0
Vermelho	2	95	3	0
Azul	0	5	93	2
Amarelo	3	4	0	93

7.5. Matrizes de confusão

Para a apresentação de resultados foi feita uma análise das matrizes de confusão para cada tipo de peça. Segue uma breve descrição dos parâmetros calculados para melhor percepção destes:

Tabela 10: Exemplo de uma matriz de confusão para uma peça

	Positivo	Negativo
Positivo	a	B
Negativo	c	D

A *exactidão* (AC) é a proporção do número total de previsões (positivo - positivo e negativo - negativo) que estavam correctas. É determinada usando a equação:

$$AC = \frac{a + d}{a + b + c + d} \qquad (7.1)$$

A *taxa de verdadeiros positivos* (TP) é a proporção de casos positivos que foram identificados correctamente, como calculado usando a equação:

$$TP = \frac{a}{b + a} \qquad (7.2)$$

A *taxa de falsos positivos* (FP) é a proporção de casos negativos que foram classificados incorrectamente como positivo, calculada por:

$$FP = \frac{c}{d + c} \qquad (7.3)$$

A *taxa de verdadeiros negativos* (TN) é definida como a proporção de casos negativos que foram classificados correctamente, usando a equação:

$$TN = \frac{d}{c + d} \qquad (7.4)$$

A *taxa de falsos negativos* (FN) é a proporção de casos positivos classificados incorrectamente como negativos, através da equação:

$$FN = \frac{b}{a+b} \qquad (7.5)$$

A *precisão* (P) é a proporção de casos positivos (note-se aqui a diferença com exactidão, que contemplava também os casos negativos e classificados dessa forma) correctamente classificados, de acordo com a equação:

$$P = \frac{a}{a+c} \qquad (7.6)$$

7.5.1. Forma

Nas tabelas 11, 12, 13, 14 e 15 encontram-se as tabelas de confusão para as três diferentes velocidades do tapete assim como a análise por fotografia.

Tabela 11: Tabelas de confusão para velocidade de tapete lenta para a peça 1(a), 2(b), 3(c), 4(d) e 5(e)

1	P	N
P	87	13
N	10	390

(a)

2	P	N
P	89	11
N	24	376

(b)

3	P	N
P	84	16
N	9	391

(c)

4	P	N
P	78	22
N	0	400

(d)

5	P	N
P	100	0
N	19	381

(e)

Tabela 12: Tabelas de confusão para velocidade de tapete média para a peça 1(a), 2(b), 3(c), 4(d) e 5(e)

1	P	N
P	83	17
N	10	390

(a)

2	P	N
P	85	15
N	23	377

(b)

3	P	N
P	81	19
N	11	389

(c)

4	P	N
P	75	25
N	0	400

(d)

5	P	N
P	100	0
N	32	368

(e)

Tabela 13: Tabelas de confusão para velocidade de tapete rápida para a peça 1(a), 2(b), 3(c), 4(d) e 5(e)

1	P	N
P	80	20
N	10	390

(a)

2	P	N
P	84	16
N	23	377

(b)

3	P	N
P	77	23
N	11	389

(c)

4	P	N
P	75	25
N	0	400

(d)

5	P	N
P	100	0
N	40	360

(e)

Tabela 14: Tabelas de confusão para análise de fotografias para a peça 1(a), 2(b), 3(c), 4(d) e 5(e)

1	P	N
P	41	9
N	1	199

(a)

2	P	N
P	44	6
N	6	194

(b)

3	P	N
P	43	7
N	0	200

(c)

4	P	N
P	39	11
N	5	195

(d)

5	P	N
P	50	0
N	21	179

(e)

7.5.2. Cor

Nas tabelas 15, 16 e 17 encontram-se as tabelas de confusão para as três diferentes velocidades do tapete.

Tabela 15: Tabelas de confusão para velocidade de tapete lenta para a peça verde(a), vermelha(b), azul(c), amarela(d)

verde	P	N
P	95	5
N	8	292

(a)

vermelho	P	N
P	94	6
N	6	294

(b)

azul	P	N
P	94	6
N	3	297

(c)

amarelo	P	N
P	96	4
N	4	296

(d)

Tabela 16: Tabelas de confusão para velocidade de tapete média para a peça verde(a), vermelha(b), azul(c), amarela(d)

verde	P	N
P	92	8
N	5	295

(a)

vermelho	P	N
P	95	5
N	9	291

(b)

azul	P	N
P	96	4
N	7	293

(c)

amarelo	P	N
P	93	7
N	3	297

(d)

Tabela 17: Tabelas de confusão para velocidade de tapete lenta para a peça verde(a), vermelha(b), azul(c), amarela(d)

verde	P	N
P	90	10
N	5	295

(a)

vermelho	P	N
P	95	5
N	15	285

(b)

azul	P	N
P	93	7
N	7	293

(c)

amarelo	P	N
P	93	7
N	2	298

(d)

7.5.3. Influência da velocidade na classificação das peças quanto à sua forma

Tabela 18: Influência da velocidade na classificação da peça 1

Peça 1	Total	Vel. lenta	Vel. média	Vel. rápida	Análise foto
AC	0,95	0,95	0,95	0,94	0,96
TP	0,83	0,87	0,83	0,80	0,82
FP	0,02	0,03	0,03	0,03	0,01
TN	0,98	0,98	0,98	0,98	1,00
FN	0,17	0,13	0,17	0,20	0,18
P	0,90	0,90	0,89	0,89	0,98

Tabela 19: Influência da velocidade na classificação da peça 2

Peça 2	Total	Vel. lenta	Vel. média	Vel. rápida	Análise foto
AC	0,93	0,93	0,92	0,92	0,95
TP	0,86	0,89	0,85	0,84	0,88
FP	0,05	0,06	0,06	0,06	0,03
TN	0,95	0,94	0,94	0,94	0,97
FN	0,14	0,11	0,15	0,16	0,12
P	0,80	0,79	0,79	0,79	0,88

Tabela 20: Influência da velocidade na classificação da peça 3

Peça 3	Total	Vel. lenta	Vel. média	Vel. rápida	Análise foto
AC	0,95	0,95	0,94	0,93	0,97
TP	0,81	0,84	0,81	0,77	0,86
FP	0,02	0,02	0,03	0,03	0,00
TN	0,98	0,98	0,97	0,97	1,00
FN	0,19	0,16	0,19	0,23	0,14
P	0,90	0,90	0,88	0,88	1,00

Tabela 21: Influência da velocidade na classificação da peça 4

Peça 4	Total	Vel. lenta	Vel. média	Vel. rápida	Análise foto
AC	0,95	0,96	0,95	0,95	0,94
TP	0,76	0,78	0,75	0,75	0,78
FP	0,00	0,00	0,00	0,00	0,03
TN	1,00	1,00	1,00	1,00	0,98
FN	0,24	0,22	0,25	0,25	0,22
P	0,98	1,00	1,00	1,00	0,89

Tabela 22: Influência da velocidade na classificação da peça 5

Peça 5	Total	Vel. lenta	Vel. média	Vel. rápida	Análise foto
AC	0,94	0,96	0,94	0,92	0,92
TP	1,00	1,00	1,00	1,00	1,00
FP	0,08	0,05	0,08	0,10	0,11
TN	0,92	0,95	0,92	0,90	0,90
FN	0,00	0,00	0,00	0,00	0,00
P	0,76	0,84	0,76	0,71	0,70

Destas tabelas podemos observar que não há uma alteração significativa nos valores com o aumento da velocidade, não sendo este portanto um factor limitativo do rendimento do programa.

7.5.4. Estudo da classificação de formas

Tabela 23: Influência da forma nos coeficientes das matrizes de confusão para valores totais

Total	Peça 1	Peça 2	Peça 3	Peça 4	Peça 5
AC	0,95	0,93	0,95	0,95	0,94
TP	0,83	0,86	0,81	0,76	1,00
FP	0,02	0,05	0,02	0,00	0,08
TN	0,98	0,95	0,98	1,00	0,92
FN	0,17	0,14	0,19	0,24	0,00
P	0,90	0,80	0,90	0,98	0,76

Para todas as peças o valor da exactidão é extremamente satisfatório. A precisão é inferior, uma vez que estão nela contabilizados os casos de classificações incorrectas.

A classe 4 é a que apresenta piores resultados a nível de verdadeiros positivos, sendo a única que fica abaixo dos 80%, sendo isto explicado pelo facto de as sombras criadas devido à forma da peça serem complicadas de eliminar. De facto estas sombras dão um aspecto arredondado à forma, eliminando os ângulos mais fechados, e como já foi referido, o algoritmo ignora as formas curvas, resultando numa incorrecta detecção de cantos e consequentemente uma incorrecta classificação. Porém é a única peça que apresenta uma taxa de falsos positivos nula, resultando na maior precisão de entre todas as classes.

Observa-se que a classe 5 tem o maior número de falsos positivos. Isto deve-se ao facto de a maioria dos casos de falsos positivos ocorrer associado a uma incorrecta detecção dos cantos. Normalmente com um valor inferior a 4 cantos. Como a única peça que tem quatro cantos é a 5, isto leva a que a triangulação seja mais aproximada à desta classe. Isto faz com que seja esta peça a que apresenta piores resultados na precisão, apesar de ter sido de todas as vezes correctamente identificada. Este factor é revelado pela taxa de verdadeiros positivos (TP), que tem o valor de 1.

7.5.5. Influência da velocidade na classificação das peças quanto à sua cor

Tabela 24: Influência da velocidade na classificação da cor verde

Verde	Lento	Médio	Rápido
AC	0,97	0,97	0,96
TP	0,95	0,92	0,90
FP	0,03	0,02	0,02
TN	0,97	0,98	0,98
FN	0,05	0,08	0,10
P	0,92	0,95	0,23

Tabela 25: Influência da velocidade na classificação da cor vermelha

Vermelho	Lento	Médio	Rápido
AC	0,97	0,97	0,95
TP	0,94	0,95	0,95
FP	0,02	0,03	0,05
TN	0,98	0,97	0,95
FN	0,06	0,05	0,05
P	0,94	0,91	0,25

Tabela 26: Influência da velocidade na classificação da cor azul

Azul	Lento	Médio	Rápido
AC	0,98	0,97	0,97
TP	0,94	0,96	0,93
FP	0,01	0,02	0,02
TN	0,99	0,98	0,98
FN	0,06	0,04	0,07
P	0,97	0,93	0,24

Tabela 27: Influência da velocidade na classificação da cor amarela

Amarelo	Lento	Médio	Rápido
AC	0,98	0,98	0,98
TP	0,96	0,93	0,93
FP	0,01	0,01	0,01
TN	0,99	0,99	0,99
FN	0,04	0,07	0,07
P	0,96	0,97	0,24

Observa-se que o factor velocidade não é factor limitativo da qualidade do programa.

7.5.6. Estudo da classificação da cor

Tabela 28: Influência da cor nos coeficientes das matrizes de confusão para valores totais

Total	Verde	Vermelho	Azul	Amarelo
AC	0,97	0,96	0,97	0,98
TP	0,92	0,95	0,94	0,94
FP	0,02	0,03	0,02	0,01
TN	0,98	0,97	0,98	0,99
FN	0,08	0,05	0,06	0,06
P	0,94	0,90	0,94	0,97

Por esta tabela pode-se comprovar os excelentes resultados obtidos com este método.

8. Conclusões e Trabalho Futuro

8.1. Conclusões

As principais dificuldades deste trabalho prenderam-se com a qualidade da imagem obtida. De facto as condições de iluminação no laboratório são muito fracas, criando sombras complicadas de eliminar. Aliado a este facto nota-se um decréscimo na taxa de *refresh* da câmara, à medida que o programa fica a correr por longos períodos de tempo. Este problema tanto pode ser devido às limitações do computador onde os testes foram efectuados, não tendo um processador muito rápido, ou uma incorrecta programação de alocação de memória, tornando o programa lento com o tempo.

A principal limitação do programa é a detecção de cantos. É necessária uma correcta afinação dos parâmetros de *threshold* da binarização e dos parâmetros de brilho para assegurar uma correcta detecção. Foi este o ponto mais complicado do ensino, encontrar os valores perfeitos naquelas condições para ensinar as formas com a maior precisão possível.

A decisão de estabelecer uma região do tapete onde seria feita a identificação prendeu-se com o assegurar que a totalidade da peça estivesse na área de visão da câmara. Apesar de as cinco peças tratadas neste projecto serem de dimensões reduzidas, a possibilidade de ensino de novas peças, possivelmente de dimensões maiores, não está limitada. É completamente aberto a peças de várias dimensões, formas, complexidade, desde que todos os seus lados sejam rectos.

8.2. Trabalho futuro

A solução encontrada para a problemática das sombras foi a aplicação de vários filtros que aumentaram o brilho total da imagem, permitindo um melhor reconhecimento, ainda que estes tenham sido embebidos no código. Porém, existem várias soluções que permitem resolver este problema. Entre eles, um foco dirigido à imagem, perpendicular à peça na zona de análise, um anel de leds à volta da câmara para permitir uma iluminação uniforme ou uma dispersão da luz natural do laboratório para evitar sombras tão fortes.

Existe a necessidade de implementar outro método que englobe peças curvas. Outra limitação é o facto de, mesmo que uma peça em análise não esteja na base de dados, haver sempre uma decisão, neste caso à forma mais aproximada. Poderia haver um segundo método, baseado

num sistema de decisão mais avançado, por exemplo, podia ser construído um sistema de decisão hierárquico, em que primeiramente seria avaliado se uma dada forma pertenceria ou não à base de dados, e em caso afirmativo qual era então essa forma. O mesmo raciocínio se aplica para a cor.

9. Referências

[1] BALLARD, D. H.; BROWN, C. M.; . *Computer Vision,***1982**.

[2] ROY DAVIES, E.; . *Machine Vision : Theory, Algorithms, Practicalities,***2005**.

[3] HUANG F.; KLETTE R.; SCHEIBE K. *Panoramic Imaging - Sensor-Line Cameras and Laser Range-Finders,***2008**.

[4] PEUCKER, T. K.; *A theory of the cartographic line. In: International yearbook of cartography*. cap. 16, p. 134-143,1975.

[5] WEIBEL, R.; (1995). *Map generalization in the context of digital systems. Cartography and Geographic Information Systems,***1995**.

[6] BEARD, K.; . *Theory of the cartographic line revisited: implications for automated generalization*. In: Cartographica. 1991. v. 28, cap. 4, p. 32-58,1991.

[7] VAN OOSTEROM, P.; SCHENKELAARS, V.;. *The development of an interactive multi-scale GIS*. In: *International Journal of Geographical Information Systems*, v. 9, n. 5, p. 489-507, 1995.

[8] LAURINI, R.; THOMPSON, D.;. *Fundamentals of spatial information systems.*In: Academic Pressm,1992.

[9] LI, Z.; OPENSHAW, S. *Algorithms for automated line generalization based on a natural principle of objective generalization*.In: International Journal of Geographic Information Systems, v. 6, n. 5, p. 373-389,1992.

[10] MCMASTER, R. B.; SHEA, K. S. .*Generalization in digital cartography*. In: Asociation of American Geographers,1992.

[11] MARINO, J. S.;. *Identification of characteristic points along naturally occurring lines: an empirical study*. In: The Canadian Cartographer, v. 16, n. , p. 70-80,1979.

[12] RAMER,U.;. *An iterative procedure for the polygonal approximation of plane curves*. In: Computer Graphics and Image Processing,1972.

[13] David Douglas & Thomas Peucker, *"Algorithms for the reduction of the number of points required to represent a digitized line or its caricature"*, The Canadian Cartographer 10(2), 112-122, 1973.

[14] John Hershberger & Jack Snoeyink, *"Speeding Up the Douglas-Peucker Line-Simplification Algorithm"*, Proc 5th Symp on Data Handling, 134-143,1992.

[15] McCulloch, W. W. and Pitts, W., *A logical calculus of the ideas imminent in nervous activity*, Bull. Math. Biophys., no. 5, pp. 115-133,1943.

[16] Hebb, D., *The Organization of Behaviour*, New York: John Wiley, 1949.

[17] Grossenberg, S., *Studies of the Mind and Brain*, Drodrecht, Holland: Reidel Press, 1982.

[18] Kohonen, T., *Self-Organization and Associative Memory*, 2nd Edition,Springer, 1987.

[19] Jang, J. S., Sun, C. T. e Mizutani, E., *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, New Jersey, 1997.

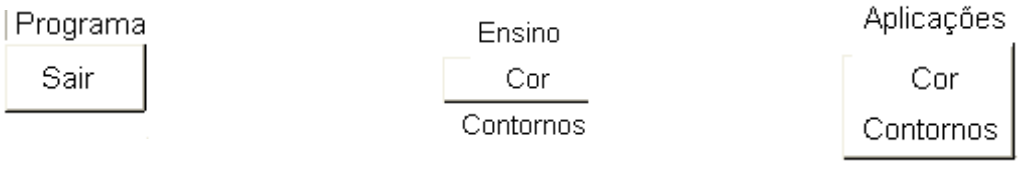
[20] Greenwood, P.E., Nikulin, M.S. *A guide to chi-squared testing*. Wiley, New York,1996

[21]Tao, Y.,Grosky, W., I.; "*Delaunay triangulation for image object indexing: a novel method for shape Representation*", Department of Computer Science, Wayne State University

10. Anexos

10.1. Manual do utilizador

Aquando da inicialização do programa, o operador tem que, recorrendo aos sliders posicionados na janela de visualização, ajustar a ROI, assegurando que apenas o tapete fica nessa área.



(a) (b) (c)

Fig. 14: Menus do programa

De seguida tem que seleccionar a aplicação pretendida. A opção *"Programa"* permite sair do programa (ver Fig. 14.a). A opção *"Ensino"* permite o ensino via fotografia para a cor e contorno (ver Fig. 14.b e consultar secção 10.2). Por fim a opção *"Aplicações"* permite correr o programa no modo de decisão de cor ou de forma(ver Fig. 14.c).

10.1.1. COR

O tapete é posto em movimento, cabendo ao operador alimentá-lo com a peça pretendida para análise. A análise dá-se de forma automática, sendo indicado na *box* "Resultado da análise" a cor correspondente à decisão, assim como o envio para o autómato do registo.

10.1.2. FORMA

O tapete é posto em movimento, cabendo ao utilizador alimentá-la com a peça pretendida para análise. Na janela aberta aquando da selecção da aplicação surge a binarização da imagem, sendo o tapete a branco e a peça a preto. A análise dá-se de forma automática, sendo indicado na *box* "Resultado da análise" a classe correspondente à decisão, assim como o envio para o autómato do registo.

10.2. ENSINO

O ensino das cores base (vermelho, azul, verde e amarelo), assim como das classes base (classe 1, 2, 3, 4 e 5) é feito de forma automática.

Caso o operador deseje incluir mais cores ou formas pode proceder da seguinte maneira:

10.2.1. ENSINO VIA FOTOGRAFIA

Surge uma caixa de diálogo onde o operador tem que seleccionar uma fotografia contendo a nova cor ou a nova forma (note-se que para ambos os casos a fotografia tem que conter apenas a peça em análise, em fundo branco). O número de peças ensinadas é actualizado, sendo também adicionado ao txt já existente o parâmetro de h para a nova cor ou o histograma da nova forma. Caso se pretenda eliminar alguma cor ou forma deve-se ir ao ficheiro txt em questão, apagar o valor de h ou as entradas do histograma da peça em questão, actualizando manualmente o número de peças ensinadas.

10.3. Ligações ao Autómato

Este programa foi desenvolvido com o intuito de haver uma ligação ao autómato. Para isso foram reservadas *flags* e registos, como *outputs*, permitindo uma programação do autómato.

Tabela 29: Valor das *flags* e dos registos para programação do autómato

Tipo	Valor	Significado
Flag	600	Detecção de peça
Registo	1000	Forma
Registo	1001	Cor