INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Combate à fraude em jogos de casino, assistida por computador

DIOGO FILIPE BAPTISTA DE PINHO
Outubro de 2014

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Combate à fraude em jogos de casino, assistida por computador

Diogo Filipe Baptista de Pinho

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Pedro Oliveira

Co-orientador: Nuno Bettencourt

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro, 2014

Aos meus pais...

Resumo

No decorrer dos últimos anos tem-se verificado um acréscimo do número de sistemas de videovigilância presentes nos mais diversos ambientes, sendo que estes se encontram cada vez mais sofisticados.

Os casinos são um exemplo bastante popular da utilização destes sistemas sofisticados, sendo que vários casinos, hoje em dia, utilizam câmeras para controlo automático das suas operações de jogo.

No entanto, atualmente existem vários tipos de jogos em que o controlo automático ainda não se encontra disponível, sendo um destes, o jogo *Banca Francesa*.

A presente dissertação tem como objetivo propor um conjunto de algoritmos idealizados para um sistema de controlo e gestão do jogo de casino *Banca Francesa* através do auxílio de componentes pertencentes à área da computação visual, tendo em conta os contributos mais relevantes e existentes na área, elaborados por investigadores e entidades relacionadas.

No decorrer desta dissertação são apresentados quatro módulos distintos, sendo um destes módulos serve de apoio aos restantes os quais têm como objetivo auxiliar os casinos a prevenir o acontecimento de fraudes durante o decorrer das suas operações, assim como auxiliar na recolha automática de resultados de jogo. Os quatro módulos apresentados são os seguintes:

Dice Sample Generator – Módulo proposto para criação de casos de teste em grande escala;
Dice Sample Analyzer – Módulo proposto para a deteção de resultados de jogo;
Dice Calibration – Módulo proposto para calibração automática do sistema;
Motion Detection – Módulo proposto para a deteção de fraude no jogo.

Por fim, para cada um dos módulos, é apresentado um conjunto de testes e análises de modo a verificar se é possível provar o conceito para cada uma das propostas apresentadas.

Palavras-chave: Banca Francesa, Jogo de casino, Computação visual, Reconhecimento de dados, Reconhecimento em tempo real, Deteção de fraude

Abstract

Over the last few years there has been an increase in the number of video surveillance systems present in multiple environments, which are getting more sophisticated as time goes by.

Casinos are a popular example of the use of these sophisticated systems. Several casinos, nowadays, use cameras for automatic control of their gambling operations.

However, currently there are some games where automatic control is not yet available, one of those games is the *Banca Francesa*.

Thus, this thesis focus on proposing a set of algorithms devised for a system that controls and manages one table of the casino game *Banca Francesa* through the aid of components belonging to the field of visual computing, taking into account existing contributions within the area from researchers and related entities.

Four distinct modules are presented throughout this dissertation, one of these modules is a support to the other which aim to assist casinos in preventing the occurrence of fraud during the course of its operations, as well as assisting in the automatic collection of game results. The four modules proposed are the following:

Dice Sample Generator - Module proposed for the creation of test cases on a large scale;
Dice Sample Analyzer - Module proposed for the detection of game results;
Calibration Dice - Module proposed for the automatic calibration system;
Motion Detection - Module proposed for the detection of fraud in the game.

Finally, for each of the modules, a set of testing and consequent analysis is presented in order to verify whether it is possible to prove the concept for each of the proposals.

Keywords: Casino gaming, Computer vision, Dice recognition, Real time recognition, fraud detection

Agradecimentos

Gostava de agradecer em primeiro lugar aos meus familiares, em especial aos meus pais, pois é com o apoio deles que consegui chegar onde cheguei e que sempre acreditaram em mim independentemente das opções que tomei durante o decorrer de todos estes anos.

Quero agradecer também aos meus colegas que partilharam comigo o local de trabalho tornando este num ambiente descontraído e ao mesmo tempo mais produtivo devido à afinidade criada entre todos, e aquando da necessidade, estarem disponíveis para me ajudar no que quer que fosse necessário.

Um especial obrigado é também dirigido aos professores Pedro Oliveira e Nuno Bettencourt pela oportunidade oferecida, pelo tempo despendido e por me apoiarem em concretizar esta dissertação da forma mais dinâmica que possível apresentando-me inúmeras oportunidades, como a iniciativa ao *Passaporte do Empreendedorismo*, a visita à exposição de material de casino *ICE 2014*, presença na *Mostra Tecnológica – Engenharia na Fábrica* e a realização de documentos de carácter científico para submissão em conferências, tudo isto de forma a enriquecer o meu argumento para esta dissertação.

Não posso também deixar de agradecer também aos responsáveis pela iniciativa do *Passaporte do Empreendedorismo* por acreditarem que o trabalho realizado até ao momento é capaz de ser aplicado como ideia de negócio, e, à *Unidade de Inovação e Transferência Tecnológica* do INESC, os quais apesar do pouco tempo de interação demonstraram uma total disponibilidade para ajudar no crescimento deste projeto.

A todos, um sincero obrigado.

Índice

1 Introdução 1

1.1 Contribuição 3

1.2 Organização do documento 3

2 Levantamento do estado de arte 5

2.1 Conceito de computação visual 5

2.2 Fraudes recorrentes em jogos de casino 6

2.3 Soluções propostas/existentes 7

2.3.1 Introducing Computers to Blackjack: Implementation of a Card Recognition System using Computer Vision Techniques 8

2.3.2 Playing Card Recognition Using Rotational Invariant Template Matching 9

2.3.3 Who's Counting? Real-Time Blackjack Monitoring for Card Counting Detection 10

2.3.4 Machine Vision in Casino Game Monitoring 11

2.3.5 Image Identification Scheme for Dice Game 12

2.3.6 Color and Illumination Invariant Dice Recognition 12

2.3.7 An Auto-Recognizing System for Dice Games Using a Modified Unsupervised Grey Clustering Algorithm 13

2.3.8 Automated Detection and Classification of Dice 14

2.3.9 Tangam Gaming 15

2.3.10 Cammegh 15

2.3.11 SET-Production 18

2.3.12 BHS Böhm 19

2.3.13 Stereo Vision Based Dice Recognition System and Stereo Vision Based Dice Recognition Method for Uncontrolled Environments 20

2.3.14 Dice-O-Matic mark II – GamersByEmail.com 21

3 Conceitos fundamentais 23

3.1 Representação digital de imagem 23

3.2 Espaço de cores HSV 24

3.3 Conceito de *Kernel* 27

3.4 Operações de *Thresholding* 28

3.5 Operações morfológicas básicas 30

3.5.1 Dilatação 31

3.5.2 Erosão 31

3.5.3 Abertura 32

3.5.4 Fecho 32

3.6 Redução polinomial de regiões 33

3.7 Algoritmo k-means clustering 34

3.8 Algoritmo background subtraction 35

3.9 Jogo Banca Francesa 36

4 Proposta de algoritmos de controlo e deteção de jogadas no jogo *Banca Francesa* 39

4.1 Dice Sample Generator (DSG) 41
4.1.1 Modelo XML: exemplo de entrada 41
4.1.2 Método de produção de amostras 43

4.2 Dice Sample Analyzer (DSA) 45
4.2.1 *Workflow* do processo 46
4.2.2 Pré-processamento da imagem 46
4.2.3 Segmentação da imagem 50

4.3 Dice Calibration (DC) 64
4.3.1 Definição de razão entre tamanhos (*spatial multiplier*) 64
4.3.2 Razão entre distâncias significativas e seleção de face utilizada 66
4.3.3 Cálculo automático de distâncias significativas 67
4.3.4 Cálculo automático do tamanho do dado 69

4.4 Motion Detection (MD) 74
4.4.1 Separação das áreas de jogo 74
4.4.2 Área de deteção de jogo 75
4.4.3 Área de deteção de fraude 77

5 Testes e análise de resultados 81

5.1 Configuração do ambiente e especificação do material utilizado 81

5.2 Dice Sample Generator (DSG) – Testes e Resultados 84

5.3 Dice Sample Analyzer (DSA) – Testes e Resultados 86
5.3.1 Abordagem preliminar efetuada e testes elaborados 86
5.3.2 Testes de implementação do DSA (amostras do DSG) 90
5.3.3 Testes de implementação do DSA (casos reais) 93

5.4 Dice Calibration (DC) – Testes e Resultados 95

5.5 Motion Detection (MD) – Testes e Resultados 98
5.5.1 Área de deteção de jogo 98
5.5.2 Área de deteção de fraude 99

5.6 Integração com dispositivo *Raspberry Pi* 104

6 Conclusões e trabalho futuro 107

6.1 Trabalho futuro 108

Lista de Figuras

Figura 1 – Cartas sobre plano de fundo pretol 8
Figura 2 – Representação do resultado do algoritmo *contour detection* 9
Figura 3 – Sistema de monitorização de *Blackjack* proposto por Krists Zutis e Jesse Hoey 10
Figura 4 – Resultado do algoritmo de classificação de cartas parcialmente oclusas 11
Figura 5 – Imagem de entrada inicial 12
Figura 6 – Características entre imagens segmentadas usando o algoritmo *Harris-Hessian affine region detector*, para diferentes cores de dados 13
Figura 7 – Resultado visual do *output* do algoritmo 13
Figura 8 – Imagem recolhida pela câmera CCD 14
Figura 9 – Imagem ampliada dos dados com face 1, 2 e 3 14
Figura 10 – Solução apresentada pela *Tangam Gaming* 15
Figura 11 – Sistema *EyeBall* da *Cammegh* 16
Figura 12 – Sistema *EyeCard*/*EyeCardPlus* da *Cammegh* 17
Figura 13 – Sistema *Cyclops-Junior* da *SET-Production* 18
Figura 14 – Modelo do sistema *PEye: Table Managment System* da *BHS Böhm* 19
Figura 15 – Modelo do componente *dice robot ´shaker´* do sistema *WildLizzy* 19
Figura 16 – Modelo conceptual do sistema proposto por Gee-Sern Hsu e Xun-Jia Zhang 20
Figura 17 – Pormenor da máquina *Dice-O-Matic mark II* 21
Figura 18 – Exemplo de uma amostra do *Dice-O-Matic mark II* 21
Figura 19 – Representação digital do retrovisor do carro 23
Figura 20 – Geometria cilíndrica do espaço de cores HSV 25
Figura 21 – Aplicação de um filtro (uso de *kernel*) para aumento de contraste 27
Figura 22 – Imagem a ser aplicado o *threshold* 28
Figura 23 – *Threshold* binário 28
Figura 24 – *Threshold* binário, invertido 29
Figura 25 – *Threshold* truncado 29
Figura 26 – *Threshold* para zero 29
Figura 27 - *Threshold* para zero, invertido 30
Figura 28 – Imagem modelo 30
Figura 29 – Resultado da operação de dilatação 31
Figura 30 – Resultado da operação de erosão 31
Figura 31 – Resultado da operação de abertura 32
Figura 32 – Resultado da operação de fecho 33
Figura 33 – Representação visual do algoritmo Douglas-Pecker 33
Figura 34 – Conjunto de informação modelo 34
Figura 35 – Escolha dos pontos $C1$ e $C2$ 34
Figura 36 – Novos pontos calculados $C1$ e $C2$ 35
Figura 37 – Resultado final do algoritmo *k-mean clustering* 35
Figura 38 – Algoritmo de *background subtraction* genérico 36
Figura 39 – Mesa típica do jogo *Banca Francesa* 36

Figura 40 – Sistema proposto ao *Passaporte do Empreendedorismo*..... 39
Figura 41 – Algoritmos/Responsabilidades do *Middleware* (algoritmos de deteção)..... 40
Figura 42 – Faces de um dado utilizadas como recursos do DSG..... 42
Figura 43 – Base de jogo utilizada como recurso do DSG..... 42
Figura 44 – Fluxograma de produção de uma amostra do DSG..... 43
Figura 45 – Exemplos de casos possíveis durante a colocação de dados..... 44
Figura 46 – Amostra aleatória gerada pelo DSG (*image95.jpg*)..... 45
Figura 47 – Diagrama do *workflow* do processo..... 46
Figura 48 – Imagem original fx e imagem resultante gx (16)..... 47
Figura 49 - Imagem original $RGB(x)$ e imagem resultante Ix (17)..... 48
Figura 50 – Imagem original fx e imagem resultante gx (18)..... 49
Figura 51 – Operação de erosão, imagem original (esquerda) e imagem resultado (direita)..... 50
Figura 52 – Resultado visual da utilização do algoritmo de deteção de contornos..... 51
Figura 53 – Resultado dos centros dado pelo algoritmo *Minimum Enclosing Circles*..... 51
Figura 54 – Modelo generalizado de criação de faces de um dado..... 52
Figura 55 – Representação visual das distâncias significativas..... 52
Figura 56 – Exemplo teórico para algoritmo de catalogação..... 53
Figura 57 – Distância entre $N1$ e todos os outros pontos..... 53
Figura 58 – Análise entre os pontos $N2$ e $N3$..... 54
Figura 59 – Grafo resultado com as distâncias filtradas..... 54
Figura 60 – Visualização da distância aferida entre $N3$ e $N4$..... 55
Figura 61 – Grupos de pontos formados com base no exemplo teórico..... 56
Figura 62 – Visualização do caso especial de grupos de 2 pontos..... 56
Figura 63 – Processo de descobrimento de padrão para cada face de um dado..... 57
Figura 64 – Possível falso positivo teórico (*errado*) e a solução correta (*correto*)..... 59
Figura 65 – Pontos de origem do padrão (*verdes*) e falsos positivos (*vermelhos*)..... 60
Figura 66 – Exemplo teórico, com distâncias significativas, para algoritmo de catalogação..... 60
Figura 67 - Exemplo teórico, separação de pontos por grupos..... 61
Figura 68 – Exemplo teórico, ordem dos pontos no grupo..... 61
Figura 69 – Dado e nomenclatura genéricos..... 62
Figura 70 – Imagem axadrezada 10x7..... 65
Figura 71 – Representação gráfica dos cantos interiores de uma imagem axadrezada..... 65
Figura 72 – Representação visual das distâncias significativas em múltiplos de A..... 67
Figura 73 – Representação dos pontos do dado com base no algoritmo de deteção de dados 68
Figura 74 – Área de jogo e subseção de análise do algoritmo de calibração..... 68
Figura 75 – Complexidade de cor do corpo de um dado (vários “tipos” de vermelho)..... 70
Figura 76 – Resultado do algoritmo *K-means* para partição em 6 grupos..... 70
Figura 77 – Representação do corte de secção num cilindro com limites *HSV* genéricos..... 71
Figura 78 – Resultados da aplicação do algoritmo de filtragem de cor..... 72
Figura 79 - Resultado visual da utilização do algoritmo de delimitador mínimo circular..... 73
Figura 80 – Relação entre diâmetro (2r) do círculo e a diagonal do corpo do dado..... 73
Figura 81 – Áreas de deteção existentes no jogo *Banca Francesa*..... 74
Figura 82 – Representação genérica das áreas constituintes do jogo..... 75

Figura 83 – Aplicação das mascaras para separação das zonas de jogo 75
Figura 84 – Três instâncias da máscara do algoritmo de *background subtraction* MOG 76
Figura 85 – Resultado do algoritmo *background subtraction* MOG2 77
Figura 86 – Resultado da operação de abertura 78
Figura 87 – Recorte das localizações anterior e atual respetivamente na imagem final 79
Figura 88 – Fluxograma de deteção de fraude 80
Figura 89 – Vista em 3ª pessoa e vista a partir da câmera da "mesa de jogo" 82
Figura 90 – Vista em perspetiva (esquerda) e resultado da recolha de imagens (direita) 83
Figura 91 – Múltiplas instâncias da configuração utilizada para a realização de testes 83
Figura 92 – Conjunto de dados brancos utilizados inicialmente 84
Figura 93 – Amostra gerada pelo DSG (5 dados) 85
Figura 94 – Tempo de processamento por número de amostras 85
Figura 95 – Pré-processamento da análise de proporção de *pixels* 87
Figura 96 – Exemplo teórico de mudança de perspetiva baseada em 4 pontos 87
Figura 97 – Resultado do recorte e mudança de perspetiva 87
Figura 98 – Imagem exemplo da bateria de testes utilizada 88
Figura 99 – Exemplo de imagens analisadas com resultados díspares 89
Figura 100 – Resultado em que os dados conectados não são identificados 89
Figura 101 – Resultado da contagem de regiões de *pixels* 0 (pretos) 90
Figura 102 – Exceção ocorrida no algoritmo do DSA proposto para amostras do DSG 92
Figura 103 – Fatores responsáveis pela ambiguidade no resultado do DSA 92
Figura 104 – Análise da envolvente dos candidatos a faces de valor 1 93
Figura 105 – Exceção ocorrida no algoritmo do DSA proposto para amostras reais 93
Figura 106 – Falsos positivos gerados devido às condições de iluminação 94
Figura 107 – Solução apresentada na mostra realizada na Fábrica de Santo Thyrso 94
Figura 108 – Lente polarizada aplica na câmera utilizada na proposta do DSA 95
Figura 109 – Resultado da aplicação do filtro polarizado 95
Figura 110 – Amostra usada para teste de cálculo da razão de tamanhos 96
Figura 111 – *Frame* de uma amostra utilizada para o algoritmo de calibração 96
Figura 112 – Configuração para cálculo das distâncias significativas 97
Figura 113 – Três *frames* distintos amostra de uma amostra de vídeo recolhida 98
Figura 114 – Máscara do algoritmo MOG e resultado da operação de abertura 99
Figura 115 – Resultados do teste de deteção de movimento para uma amostra de vídeo. 99
Figura 116 – Imagem com ajuste automático ligado 100
Figura 117 – Imagem com ajuste automático desligado 102
Figura 118 – Máscara resultante da aplicação do algoritmo MOG2 102
Figura 119 – Máscara MOG2 após filtragem de ruído 102
Figura 120 – Imagem modelo a ser procurada pelo algoritmo SURF 103
Figura 121 – Resultado da aplicação do algoritmo SURF 103
Figura 122 – Aplicação do algoritmo SURF a uma imagem de baixa complexidade 103
Figura 123 – Computador *Raspberri Pi*, desmontado e montado 104

Lista de Tabelas

Tabela 1 – Tipos de aposta no jogo *Banca Francesa* 37
Tabela 2 – Valores utilizados para *gain* (α) e *bias* (β) 47
Tabela 3 – Valor utilizado para o limiar do *threshold* 49
Tabela 4 – Valor utilizado para o índice de tamanho (k) do *kernel* 50
Tabela 5 – Resultado da relação dado/ponto/distância. 55
Tabela 6 – Distâncias significativas presentes nas faces dos dados 67
Tabela 7 – Valores utilizados para os limites de HSV 72
Tabela 8 – Valor utilizado para o índice de tamanho (k) do *kernel* 78
Tabela 9 – Percentagem de *pixels* 0 (preto) padrão para cada face de um dado 88
Tabela 10 – Resultado no qual é demonstrada disparidade de valores 89
Tabela 11 – Distâncias significativas utilizadas para o DSA 91
Tabela 12 – Valores utilizados para o cálculo da razão entre tamanhos do DSA 91
Tabela 13 – Pontos resultantes da aplicação do algoritmo de calibração 96
Tabela 14 – Amostras utilizadas para o teste do cálculo do tamanho do dado 97
Tabela 15 – Distâncias significativas aferidas 98
Tabela 16 – Teste de performance do *Raspberry Pi* (captura de imagem simples) 105
Tabela 17 – Teste de performance do *Raspberry Pi* (captura de imagem e utilização do algoritmo MOG) 106
Tabela 18 – Teste de performance do *Raspberry Pi* (captura de imagem e utilização do DSA) 106

Acrónimos e Símbolos

Lista de Acrónimos

SIFT	*Scale-Invariant Feature Transform*
LDC	*Least Distance Criterion*
MSER	*Maximally Stable Extremal Regions*
GCCS	*Gradient-Conditioned Color Segmentation*
INETI	Instituto Nacional de Engenharia, Tecnologia e Inovação
IGJ	Inspeção Geral de Jogos
CCD	*Charge-Coupled Device*
LCD	*Liquid-Crystal Display*
USB	*Universal Serial Bus*
IP	*Internet Protocol*
DSG	*Dice Sample Generator*
XML	*eXtensible Markup Language*
DSA	*Dice Sample Analyzer*
JPG/JPEG	*Joint Photographic Experts Group*
RGB	*Red-Green-Blue*
DC	*Dice Calibration*
HSV	*Hue-Saturation-Value*
IDE	*Integrated Development Environment*
ComVics	*COMputer Vision and Intelligent Computer Systems*
MOG	*Mixture Of Gaussians*
API	*Application Programming Interface*
V4L	*Video4Linux version 1*
V4L2	*Video4Linux version 2*
FPS	*Frames Per Second*

1 Introdução

Até que seja possível atingir um reconhecimento de imagens, ao nível da capacidade humana, ainda existe um grande caminho a percorrer na área de deteção e reconhecimento de objetos auxiliada por computador. Embora ainda estejamos longe de atingir esse nível, nos últimos anos tem-se assistido à disseminação de sistemas de videovigilância nos mais diversos ambientes [AVSS, 2014], colocando desafios aos investigadores no sentido de automatizar a monitorização do conteúdo recebido, através das câmeras de vídeo, utilizando técnicas de processamento de imagem e visão por computador.

Ao contrário da capacidade humana, um computador pode processar uma ou mais imagens, com uma maior rapidez e com formas diferentes de análise, o que por vezes permite detetar padrões de comportamento irregulares que passariam despercebidos a um humano, como por exemplo, a deteção do ritmo cardíaco com base em diferenças subtis de mudanças de cor na imagem [Wu *et al.*, 2012] que podem ser exploradas recorrendo a algoritmos de computação visual.

A indústria dos jogos, particularmente de casino, tem demostrando grande interesse na aquisição e utilização destes sistemas pois além da tradicional monitorização [Cooper and Dawson-Howe, 2004], estes sistemas permitem o registo automático dos jogos, prevenindo perdas por enganos, irregularidades e incumprimento de regras, bem como para fins estatísticos.

Os jogos de casino não baseados em máquinas de jogo, vulgarmente denominados por jogos bancados, são tradicionalmente alvos de tentativa de fraude por parte dos seus jogadores. Dependendo do tipo de jogo, com roleta, cartas ou dados, sempre existiram esquemas criados com intuito de defraudar o jogo [Boss and Zajic, 2010], como manipulação mecânica dos dispositivos, deteção de velocidade da bola na roleta com o intuito de determinar a probabilidade de saída de números, contagem e troca de cartas, deteção de paragem do primeiro dado na mesa [Goding, 2012].

No decorrer da última década têm sido apresentadas soluções, com base em sistemas de computação visual, tendo em vista solucionar a problemática levantada pelas fraudes ocorrentes em jogos de casino do tipo bancado. Estas soluções encontram-se normalmente segmentadas em relação ao tipo de jogo abrangido (cartas, roletas e dados), sendo que, por

norma, cada trabalho apresentado utiliza como base o seu antecessor de modo a melhorar os resultados obtidos com a sua contribuição.

Em 2004, Hollinger e Ward [Hollinger and Ward, 2004] propõem uma solução capaz de detetar cartas de jogo com base na segmentação das mesmas, seguido do uso de algoritmos de *template matching*. Sendo este o ponto de partida para outras soluções, como por exemplo a solução proposta por João Pimentel [Pimentel, 2010] onde este contribuiu com um método para lidar com situações onde as cartas se encontram parcialmente oclusas.

Uma vez que existem diferentes jogos de casino que recorrem a baralhos de cartas para que estes sejam jogados, e os jogos de cartas serem reconhecidos como um jogo típico de casino, tanto o interesse acadêmico como o interesse industrial tem recaído maioritariamente sobre estes. Desta forma é possível encontrar soluções comerciais que fazem uso de computação visual, como o caso dos produtos da *Cammegh* [Cammegh, 2014b] e da *Tangam Gaming* [Zemanta, 2007].

No que toca a jogos relacionados com roletas é evidenciada a utilização de computação visual para detetar em tempo real o estado da roleta como é apresentado pelos produtos, *EyeBall* da *Cammegh* [Cammegh, 2014a] e *Cyclops* da *SET-Productions* [SET-Production, 2014]. No entanto, estas soluções são consideradas como menos eficazes e mais intrusivas do que as soluções baseadas em leitura por infravermelhos, as quais servem o mesmo propósito, segundo os representantes das mesmas. Os pontos fortes que os sistemas por computação visual possuem face aos sistemas de infravermelhos são a capacidade destes serem montados por cima de aparelhos já existentes, não sendo necessária a aquisição de novas roletas, assim como o custo do material (aquando da necessidade da compra de roletas para ambos os casos).

Em relação a jogos baseados em dados encontram-se disponíveis estudos feitos, com base na utilização de computação visual. A solução proposta por Chung [Chung et. al., 2009] apresenta um método capaz de detetar os valores de dados do tipo asiático com base em distâncias de pontos num ambiente restrito.

Em 2012 e tendo como base o trabalho realizado por Chung, um grupo de investigadores do *Artificial Vision Lab.* da *National Taiwan University of Science and Technology* apresentou uma solução [Hsu *et al.* 2012] baseada em várias câmeras de vídeo, de forma a atenuar o efeito de reflexão que acontece nos dados em determinados casos impossibilitando desta forma a deteção dos mesmos.

Uma solução apresentada por um grupo de investigadores portugueses [Bento *et. al*. 1995], no que toca a reconhecimento de dados, chegou a protótipo de modo a serem efetuados testes em ambientes reais de casino. Esta solução faz uso de uma câmera de vídeo CCD monocromática de forma a segmentar as pintas dos dados presentes na mesa, as quais, em seguida, por meio de *template matching* são catalogadas de forma a aferir os valores dos dados presentes na mesa.

Todas estas soluções têm em vista minimizar a ocorrência de ações fraudulentas em jogos de casino. Em Portugal a introdução de fiscais de jogo, face a uma obrigação legal atualmente em vigor [Lei do Jogo, 2014], tende a minimizar este tipo de tendências de manipulação de jogo e tentativas de fraude. No entanto, apesar de já serem utilizadas várias tecnologias informáticas na contabilidade de jogo e monitoração de jogadas, por exemplo o sistema *PEYE – Table*

Management System [BHS Böhm, 2014], ainda não existe, pelo menos em território nacional, a adoção de tecnologias que auxiliem os fiscais de jogo a identificar tentativas de fraude nos jogos bancados, em especial num tipo de jogo que é a Banca Francesa, cuja predominância de jogo é feita em casinos portugueses, espanhóis e asiáticos e o qual serviu de objeto de estudo para a dissertação.

O trabalho sobre o qual assenta esta dissertação apresenta um sistema capaz de verificar a ocorrência de irregularidades no jogo Banca Francesa, assim como o reconhecimento do estado atual da mesa de jogo, com base no uso de algoritmos de computação visual.

1.1 Contribuição

O trabalho descrito nesta dissertação apresenta as seguintes contribuições:

- Estado de arte referente ao uso de sistemas de computação visual no âmbito de jogos de casino;
- Proposta de um sistema de criação de amostras para elaboração de testes em grande escala para a análise de dados de jogo;
- Especificação de uma solução generalizada de deteção de dados de jogo baseado em distâncias de pintas da face superior;
- Especificação de uma solução para atenuar os efeitos de reflexão dos dados apenas recorrendo a uma câmera de vídeo;
- Especificação de uma solução para a calibração automática do sistema de deteção de dados;
- Especificação de uma solução capaz de notificar o operador de casino com base numa atuação ilícita no jogo Banca Francesa;
- Proposta de uma arquitetura de um sistema capaz de ser integrado num ambiente de casino.

1.2 Organização do documento

Esta dissertação encontra-se organizada em 6 capítulos.

O capítulo 1 referente à introdução.

O capítulo 2 engloba um breve resumo dos conceitos de computação visual, fraudes de casino assim como a compilação do estado de arte atual referente a soluções no âmbito de jogos de casino que recorrem a computação visual.

O capítulo 3 foca-se na explicação de conceitos necessários para a compreensão posterior da dissertação, como por exemplo a explicação de como se processa uma instância do jogo

Banca Francesa, assim como a explicação de conceitos referentes a algoritmos de computação visual.

O capítulo 4 apresenta a solução proposta no âmbito da dissertação, assim como o processo utilizado para elaboração da mesma.

O capítulo 5 tem como base a análise dos resultados provenientes da aplicação da solução proposta.

O capítulo 6 refere-se às conclusões obtidas durante este processo e o trabalho futuro a ser realizado com base nos resultados obtidos.

2 Levantamento do estado de arte

Soluções baseadas em algoritmos de computação visual são bastante dependentes da informação que lhes é fornecida. Desta forma, para um determinado problema generalizado, pode existir um leque de abordagens distintas que são fabricadas em função das especificações que cada problema possuí. Deste modo, a título de exemplo, temos a deteção facial que pode variar consoante um número de restrições existentes, como a existência ou ausência de óculos, a pessoa se encontrar ou não de perfil, entre outras. As soluções de controlo de jogo e similares que sejam desenvolvidas recorrendo a componentes de computação visual, encontram-se sujeitas a abordagens distintas, apesar do tema ser similar.

O levantamento do estado de arte efetuado encontra-se dividido em três partes, sendo a primeira uma breve introdução do que é a computação visual e de que forma esta se enquadra no dia-a-dia através das suas áreas abrangentes.

A segunda parte refere-se a um breve levantamento do tipo de fraudes existentes em ambientes de casino, e os pontos que as caracterizam.

A terceira parte é dedicada ao conteúdo científico de teor acadêmico, publicações, dissertações e trabalho desenvolvido por membros da comunidade académica, assim como, serviços e produtos de teor industrial que foram encontrados e os quais recorrem a componentes de computação visual nas suas soluções, também estes aplicados em ambientes de casino e similares.

2.1 Conceito de computação visual

Computação visual pode ser compreendida como a ciência que proporciona visão a computadores e outras máquinas.

Este conceito de visão não se refere só a processos de deteção e gravação de raios de luz de forma a reproduzir numa instância futura, como funcionam as câmeras de vídeo convencionais, mas também à interpretação dos mesmos de forma a gerar ações por meio tomadas de decisão com base na informação adquirida [Learned-Miller, 2011].

De forma a possibilitar a criação das decisões a tomar, o campo da computação visual inclui métodos para aquisição, processamento, análise e compreensão de imagens.

Outra forma de definir o conceito de computação visual é através das áreas em que esta se encontra aplicada. Um dos mais proeminentes campos da sua aplicação encontra-se na análise e processamento de imagens médicas. Esta área caracteriza-se pela extração da informação de uma imagem com o propósito de formular o diagnóstico do paciente.

Como segunda área de peso na utilização da computação visual temos aplicações com vertente militar, apesar de maior parte destes projetos não se encontrarem abertos ao público, existem exemplos óbvios da sua utilização, como a deteção de soldados ou veículos inimigos ou então a programação de mísseis teleguiados.

Uma terceira área de aplicação da computação visual é na indústria. Neste contexto a informação extraída tem como propósito o suporte ao processo de fabrico, como por exemplo no controlo de qualidade de um elemento ou produto inspecionado (forma automática), ou então ao auxílio da componente humana do processo de forma a ajudar este na tomada de decisões.

Mais recentemente uma quarta área tem vindo a ganhar força, sendo esta a automatização de veículos como por exemplo o carro *Google Driverless Car* ou então o projeto desenvolvido pela NASA, *NASA's Mars Exploration Rover*. A aplicação de tecnologias de computação visual já se pode também encontrar presente em menor escala em diversos veículos, como métodos de auxílio ao condutor, tendo por exemplo a deteção de condutores com indícios de sonolência ao volante.

A tabela presente no Anexo A [Lowe, 2013] apresenta uma compilação dos sectores da indústria e as empresas que incorporam produtos baseados parcialmente ou totalmente em métodos de computação visual.

2.2 Fraudes recorrentes em jogos de casino

Foram detetados vários métodos que, através de ações fraudulentas em jogos de casino, criam vantagens para o jogador. No entanto, os métodos mais recorrentes utilizados pelos jogadores podem ser divididos em categorias sendo [Goding, 2012] [Boss and Alan, 2010]:

Conhecimento de eventos futuros – Em jogos de cartas estes métodos traduzem-se pelo conhecimento das próximas cartas a serem jogadas, as cartas que o *dealer* possuiu, ou o conhecimento das cartas dos outros jogadores. Sendo o método mais conhecido o da contagem de cartas em *blackjack* de forma a jogar consoante as probabilidades associadas ao aparecimento das cartas no jogo.

Apostas tardias – Estas apostas tardias normalmente são efetuadas após uma decisão ter sido tomada no jogo em questão. As ações que figuram nesta categoria são por norma as seguintes:

- Aumento da aposta após ter sido feita uma decisão sendo este ato denominado de *capping* ou *pressing;*

- Decréscimo da aposta após uma decisão ter sido tomada sendo esta ato denominado de *pinching*;

- Trocar uma aposta de maior valor por uma de menor valor após o conhecimento da decisão;

- Movimentação da aposta efetuada de uma posição perdedora para uma posição vencedora, exemplos de jogos são o *baccarat*, roleta, *craps*, banca francesa, etc..

Manipulação do resultado do jogo – Neste caso o jogador que comente a ação fraudulenta manipula as componentes do jogo (dados, cartas, etc.) de forma a fazer com que a sua aposta, ou uma aposta feita por um cúmplice ganhe. Um exemplo simples seria a troca de cartas não autorizado.

Conluio – Neste caso a ação fraudulenta é efetuada não por um jogador mas por um grupo de dois ou mais jogadores que por algum meio comunicam informação referente ao jogo que não deveria ser acessível aos outros jogadores, um exemplo seria a identificação das cartas do parceiro por base de sinalização gestual.

Tendo isto, conclui-se que todas as fraudes efetuadas em casino seguem um princípio comum, *as pessoas veem aquilo que esperam ver e falham em observar os detalhes*, sendo que tal acontece devido a três fatores, sendo eles:

- Complacência;

- Falta de habilidade em observar;

- Falta de treino. Não saber que pormenores observar.

Todos estes fatores, no entanto, podem ser atenuados ou tornarem-se inexistentes com o auxílio de máquinas que utilizem aplicações baseadas em métodos de computação visual pois estas têm como principal função conseguir perceber uma situação com base nas imagens que lhe são fornecidas, sendo assim possível averiguar, em determinado momento, se certas ações ocorreram sem que para tal seja necessário a intervenção de um humano.

2.3 Soluções propostas/existentes

Os projetos apresentados neste subcapítulo segmentam-se nas três vertentes onde são encontradas componentes de computação visual. A primeira vertente recai sobre a identificação de cartas de jogo, sendo esta a vertente mais proeminente nos projetos baseados na área da computação visual aplicada a material de casino.

A segunda vertente abordada é referente a jogos de roleta, apesar destes não apresentarem uma grande adesão em comparação com as outras vertentes (devido ao facto de ser possível solucionar as mesmas questões de outras formas mais adequadas através da utilização de infravermelhos). No entanto, é visível a sua aplicação de forma bem-sucedida em alguns produtos comerciais.

A terceira vertente aborda soluções na área do reconhecimento de dados para jogos de casino, o qual é o foco principal deste projeto. Apesar dos projetos apresentados possuírem um grau de similaridade elevado no que toca aos resultados pretendidos, a forma como estes são abordados cria uma diferença significativa entre eles.

Devido à natureza fechada de certos produtos apresentados, a informação referente ao funcionamento dos mesmos é escassa, sendo que a informação referente a certas empresas e os seus produtos só foi possível de ser constatada na visita efetuada à exposição de material de jogos de casino e similares, *ICE Totally Gaming 2014*.

2.3.1 Introducing Computers to Blackjack: Implementation of a Card Recognition System using Computer Vision Techniques

Este projeto [Hollinger and Ward, 2004] baseia-se num programa de computador capaz de reconhecer todos os membros de um grupo de um baralho de cartas padrão de 52 cartas, também conhecido como baralho francês. O sistema utilizado faz uso de uma câmera que se encontra apontada para um plano de fundo preto, no qual são dispostas as cartas de forma a serem reconhecidas [Figura 1].

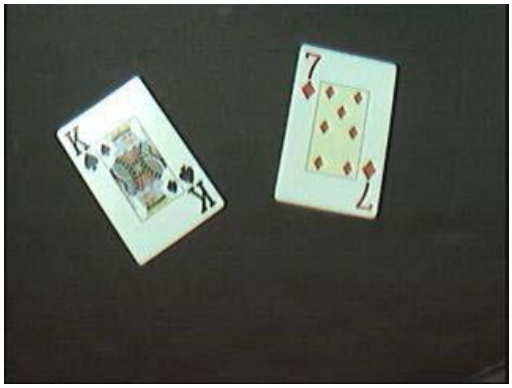


Figura 1 – Cartas sobre plano de fundo pretol[1]

O programa encontra-se primeiramente configurado para realizar decisões referentes ao jogo *Blackjack*, no entanto os autores afirmam que outras regras podem ser adicionadas de forma a abranger outro tipo de jogos carteados. É também referido que o sistema funciona corretamente na maior parte dos testes. No entanto, existem casos pontuais onde a identificação das cartas é efetuada de forma incorreta devido à orientação das mesmas no plano ou a problemas de luminosidade (reflexos e falta de luz ambiente). Infelizmente não é apresentado qualquer tipo de informação referente a taxas de sucesso/insucesso ou qualquer tipo de casos de teste utilizados.

[1] http://www.ultranurd.net/engin/e27/final/images/raw.jpeg

2.3.2 Playing Card Recognition Using Rotational Invariant Template Matching

O projeto [Zheng and Green, 2007] é bastante similar ao projeto [Hollinger and Ward, 2004], anteriormente mencionado. No que toca aos resultados pretendidos, este trabalho pretende identificar as cartas presentes numa mesa de jogo de um baralho do tipo francês.

Este projeto, contrasta com o projeto [Hollinger and Ward, 2004] em determinados aspetos, sendo um deles, a independência do plano de fundo, sendo que este apenas necessita de ser um plano com ruído reduzido, ou seja um plano aparentemente uniforme. Em contrapartida o plano de fundo do projeto [Hollinger and Ward, 2004] teria de ser um plano de fundo aproximadamente preto.

Outro aspeto que diferencia os projetos é a forma como estes se encontram estruturados. Apesar de produzirem resultados similares, o processo utilizado por ambos difere em certos pontos, sendo de notar o uso do algoritmo de *contour detection*, apresentado na Figura 2, no projeto [Zheng and Green, 2007], de forma a encontrar os contornos das cartas presentes na imagem de entrada, sendo que é desta forma que o projeto consegue fazer com que o plano de fundo não necessite de ser estritamente preto.

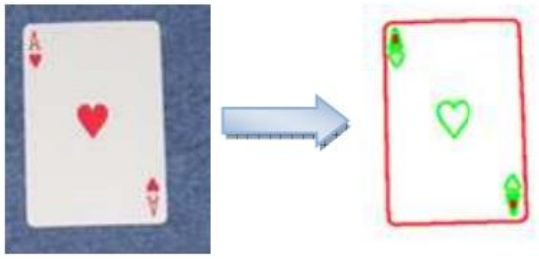


Figura 2 – Representação do resultado do algoritmo *contour detection*[2]

Os autores afirmam ter obtido uma taxa de 99.79% de precisão com o método proposto para o reconhecimento do valor da carta e uma taxa de 81.06% para o reconhecimento do naipe.

Os casos de teste utilizados consistiram na utilização das 52 cartas de um baralho francês sendo que cada uma delas foi disposta num ângulo entre 0° e 180°, com incrementos de 10°, com uma resolução de imagem 1024x768. Uma vez segmentadas as cartas do fundo estas foram avaliadas para as escalas de 100% (tamanho original) até 40%, com decrementos de 10% em cada iteração. É mencionado que a precisão do algoritmo proposto é ideal até uma redução de 60% da resolução da carta.

[2]http://www.researchgate.net/publication/241922902_Playing_Card_Recognition_Using_Rotational_Invariant_Template_Matching

2.3.3 Who's Counting? Real-Time Blackjack Monitoring for Card Counting Detection

O projeto [Zutis and Hoey, 2009] segue mais uma vez a linha de pensamento dos projetos anteriormente mencionados [Hollinger and Ward, 2004] [Zheng and Green, 2007], ou seja, a identificação de cartas de jogo. No entanto, este não é apenas o foco principal do mesmo mas apenas uma componente do sistema proposto, sendo que este conta também com a deteção das fichas de jogo presentes na mesa.

O sistema apresentado por Krists Zutis e Jesse Hoey pretende identificar possíveis contadores de cartas no jogo *Blackjack*. De forma a realizar este objetivo as apostas dos jogadores são também rastreadas em paralelo, com as cartas de forma a identificar padrões de jogo que apontem que o jogador se encontra de facto a fazer contagem de cartas [Figura 3].

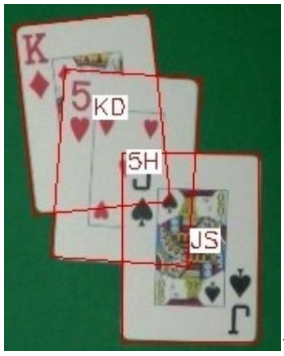
Figura 3 – Sistema de monitorização de *Blackjack* proposto por Krists Zutis e Jesse Hoey[3]

Este trabalho destaca-se dos restantes devido ao uso do algoritmo de *feature extraction* e *feature classification,* SIFT, para o reconhecimento das cartas de jogo em comparação com os algoritmos de *template matching* utilizados nos projetos anteriormente mencionados [Hollinger and Ward, 2004] [Zheng and Green, 2007], assim como, a utilização de uma câmera estéreo (*Point Grey Bumblebee*) de forma a capturar a cena e aferir a quantidade de fichas que o jogador possuiu num determinado momento. A relação entre a quantidade de fichas e a sequência de cartas ao longo do tempo serve para verificar se as jogadas do jogador caiem no padrão de contagem de cartas.

[3] https://cs.uwaterloo.ca/~jhoey/research/blackjack/SystemExample-CardCounter.jpg

2.3.4 Machine Vision in Casino Game Monitoring

O projeto [Pimentel, 2010] pretende expandir a versatilidade dos projetos anteriormente mencionados [Hollinger and Ward, 2004] [Zheng and Green, 2007] [Zutis and Hoey, 2009]. O trabalho realizado por João Pimentel, como objeto da sua tese de mestrado, em colaboração com a *Observit* propõe um método de deteção e reconhecimento de cartas de jogo tendo em conta oclusões parciais das mesmas [Figura 4]. Este projeto destaca-se dos restantes ao introduzir o algoritmo *Hough Transform* de forma a auxiliar na aferição dos retângulos candidatos a serem considerados como contornos das cartas de jogo.



[4]

Figura 4 – Resultado do algoritmo de classificação de cartas parcialmente oclusas

O trabalho elaborado apresenta também três métodos para o reconhecimento de cartas, sendo eles "*Classificação de Cartas com um Modelo Probabilístico*", "*Classificação com um Modelo Deformável*" e "*Classificação por Template Matching*". Dos métodos expostos, o autor afirma que os primeiros dois, apesar de serem mais robustos no que toca a mudanças de luminosidade, apresentam um tempo computacional muito longo, sendo que o autor qualifica o método tradicional de *template matching* como adequado para a classificação das cartas.

[4] https://fenix.tecnico.ulisboa.pt/downloadFile/395142223319/resumo.pdf

2.3.5 Image Identification Scheme for Dice Game

O seguinte projeto [Chung et. al., 2009] propõe uma solução para identificar dados baseado no método de distância mínima de pontos, LDC. Segundo o autor, a solução apresenta uma robustez elevada, no entanto faz uso de determinadas restrições que eliminam possíveis erros provenientes dos métodos utilizados.

 [5]

Figura 5 – Imagem de entrada inicial

Uma das restrições utilizadas refere-se ao uso de um plano de fundo de tonalidade clara similar à base dos dados utilizados, fazendo com que o processo de segmentação das pintas dos dados, por meio da utilização do método de *thresholding O'stu* [Otsu, 1979] devolva resultados claros e sem ruído associado. Outra restrição prende-se com a própria configuração dos dados utilizados, uma vez que estes são modelos asiáticos, o que faz com que as pintas de valor 1 dos dados possam ser automaticamente reconhecidas, uma vez que estas têm um tamanho superior às outras pintas diferentemente do que acontece com os dados ocidentais.

Os autores afirmam ter uma taxa de precisão de 100% para um conjunto de testes de 250 amostras.

2.3.6 Color and Illumination Invariant Dice Recognition

O projeto [Hsu *et al.*, 2012], realizado pelo grupo de investigadores do *Artificial Vision Lab.* da *National Taiwan University of Science and Technology,* tem em vista propor um sistema de leitura automática do número de pintas de um dado para jogo de dados genérico. Este trabalho diferencia-se dos demais devido à sua versatilidade, pois foi concebido para detetar dados com diferentes cores em diferentes ambientes de iluminação. O trabalho faz uso de três câmeras posicionadas em diferentes localizações, de forma a recolher três imagens para cada determinado momento de três perspetivas diferentes. Uma vez recolhidas as imagens, os dados são segmentados usando um algoritmo criado pelos autores, GCCS. O resultante das segmentações é então utilizado em conjugação com o algoritmo *Harris-Hessian affine region detector* [Mikolajczyk and Schmid, 2002], de forma a encontrar as características comuns

[5] http://www.inf.cyut.edu.tw/AIT2009/Paper/ft_144.pdf

entre as 3 imagens de modo a criar um modelo resultante da interceção entre as 3 imagens, sendo este modelo uma representação de topo do dado a ser analisado.

Esta foi a forma que os autores encontraram para atenuar os efeitos de reflexão, que pode acontecer em determinados ângulos, impossibilitando o reconhecimento dos dados.

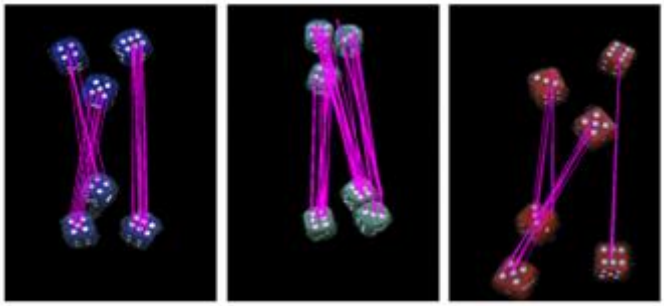


Figura 6 – Características entre imagens segmentadas usando o algoritmo *Harris-Hessian affine region detector*, para diferentes cores de dados[6]

Uma vez gerado o modelo resultante das 3 imagens é aplicado o algoritmo MSER [Matas *et al.* 2002] de forma a serem detetadas as pintas para cada um dos dados que em seguida são agrupadas utilizando um algoritmo semelhante ao proposto por [Chung et. al., 2009].

2.3.7 An Auto-Recognizing System for Dice Games Using a Modified Unsupervised Grey Clustering Algorithm

O projeto [Huang, 2007], realizado por Kuo Yi Huang do departamento de *Mechatronic Engineering* da *Huafan Univeristy,* propõe uma solução com base em algoritmos de computação visual de forma a fazer a segmentação da imagem. Em seguida, utiliza um algoritmo *unsupervised grey clustering* modificado, de forma a encontrar *clusters* de cinzentos na imagem, de modo a aferir os valores numéricos dos dados.

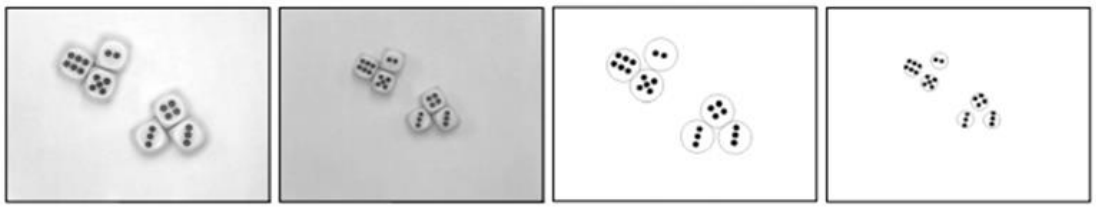


Figura 7 – Resultado visual do *output* do algoritmo[7]

Este projeto apesar de conseguir resultados com taxas de sucesso de 100%, segundo o documento, está limitado por várias restrições semelhantes ao projeto [Chung et. al., 2009] apresentado no subcapítulo [2.3.5].

[6] http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=6377835
[7] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3927534/

2.3.8 Automated Detection and Classification of Dice

Este projeto [Bento et. al. 1995] foi elaborado por um grupo português do INETI em 1995 sob o contrato da IGJ de forma a analisar os jogos de Banca Francesa e retornar os valores dos dados em jogo. O sistema intitulado de SORTE faz uso de uma câmera CCD colocada numa vista de topo em relação à área de jogo [Figura 8].

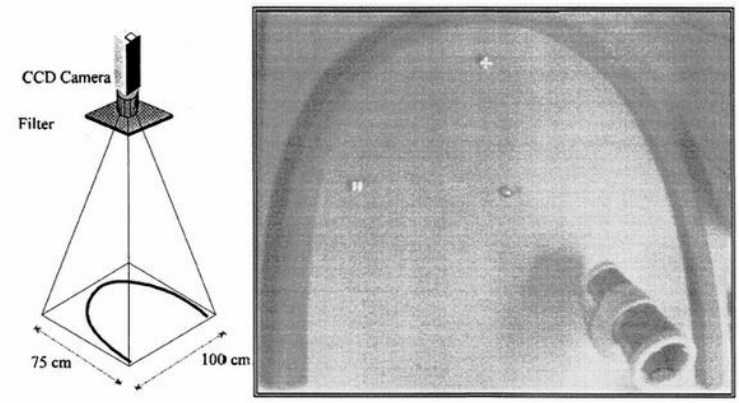


Figura 8 – Imagem recolhida pela câmera CCD[8]

Uma vez que a qualidade de imagem captada era precária à data de desenvolvimento do projeto, quando comparada com os padrões da atualidade [Figura 9], o método utilizado para segmentar as pintas dos dados consiste numa varredura da imagem.

A varredura da imagem era efetuada na horizontal e na vertical de forma a definir os centros das pintas com base nos máximos na horizontal e na vertical de segmentos com mais intensidade na imagem, sendo que estes segmentos possuem aproximadamente o mesmo tamanho do diâmetro das pintas dos dados.

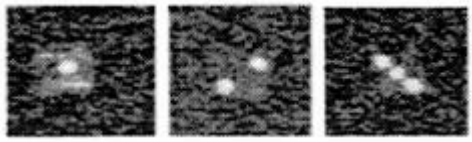


Figura 9 – Imagem ampliada dos dados com face 1, 2 e 3[9]

No entanto, nenhuma informação foi encontrada em relação ao sistema e a falta de informação do mesmo faz com que seja possível supor que este foi descontinuado e não se encontra a ser utilizado na data corrente.

[8] http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=992363
[9] http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=992363

2.3.9 Tangam Gaming

A pesquisa de aplicações de *software* direcionadas exclusivamente à monitorização de jogos de casino, revelou existir até à data apenas uma empresa denominada de *Tangam Gaming* [Zemanta, 2007]. A empresa fornece uma solução para monitorização e registo de jogadas, para posterior análise estatística [Figura 10].

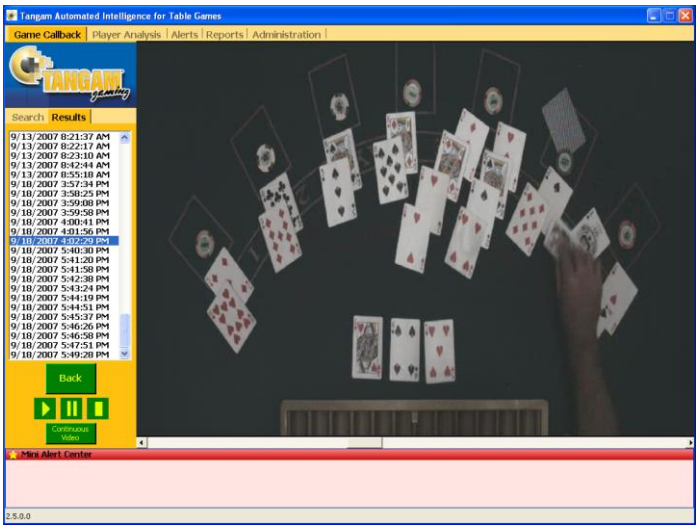


Figura 10 – Solução apresentada pela *Tangam Gaming*[10]

Apesar da solução apresentada pela empresa aparentar possuir um desempenho satisfatório [Youtube, Paulof, 2013] este é apenas dedicado ao jogo de cartas *Baccarat.*

2.3.10 Cammegh

A empresa *Cammegh* foi dada a conhecer durante a visita à exposição *ICE Totally Gaming 2014*. Esta empresa é um dos maiores fabricantes de roletas, a nível mundial, dispondo de vários modelos, sendo que duas das soluções apresentadas pela empresa encontravam-se baseadas em componentes de computação visual.

[10] http://www.csml.ucl.ac.uk/courses/msc_ml/?q=node/83

2.3.10.1 Eyeball

A primeira solução apresentada pela empresa recai sobre a sua especialidade, ou seja, roletas. Esta solução, denominada de *EyeBall* [Cammegh, 2014a], que pode ser visualizada na Figura 11, baseia-se no reconhecimento dos valores obtidos numa roleta por meio de uma câmera.

Figura 11 – Sistema *EyeBall* da *Cammegh*[11]

A solução apresentada na exposição, ICE 2014, possuía 3 estados, sendo eles '*Place Your Bets*' referindo-se ao período no qual os jogadores efetuam as apostas, *'No More Bets'* referindo-se ao período onde a bola se encontra em movimento e onde não podem ser feitas apostas, e finalmente o estado *'Reveal'* onde o número obtido na jogada é apresentado aos jogadores da mesa por meio do *LCD* do sistema *EyeBall*.

As principais características do sistema *EyeBall* segundo a *Cammegh* são as seguintes:

- Trabalhar com qualquer roleta padrão de casino;
- Enviar informação para o *LCD* de forma a apresentar o resultado aos jogadores;
- Possibilidade de integração com aplicações externas devido ao *output* padrão de informação utilizando o protocolo de comunicações do *Mercury 360* (outro modelo de roleta);
- Sistema pequeno e discreto;
- Múltiplas opções de montagem.

É importante enunciar que apesar do sistema *Eyeball* ser bastante robusto, os representantes da empresa afirmam que, a preferência de produtos de reconhecimento automático de resultados em roletas é dominado por soluções baseadas em sensores infravermelhos como o caso do modelo *Mercury 360*, sendo o este no caso da *Cammegh* o sistema mais rápido na aquisição do resultado.

[11] http://www.cammegh.com/files/Cammegh_Brochure.pdf

2.3.10.2 EyeCard e EyeCardPlus

A segunda solução apresentada pela *Cammegh*, apesar de esta ser maioritariamente um fabricante de roletas e relacionados, baseia-se em reconhecimento de cartas de jogo e tem por nome *EyeCard* [Cammegh, 2014b], e a versão com mais funcionalidades *EyeCardPlus*. A diferença entre o sistema *EyeCard* e o *EyeCardPlus* é a capacidade do *EyeCardPlus* ser capaz de reconhecer a localização das caixas (posição) onde as cartas se encontram a ser jogadas [Figura 12].

Figura 12 – Sistema *EyeCard*/*EyeCardPlus* da *Cammegh*[12]

As principais características dos sistemas *EyeCard* e *EyeCardPlus* segundo a *Cammegh* são as seguintes:

- Reconhecimento de cartas na zona do *dealer* (para o *EyeCard*) ou reconhecimento completo da mesa (para o *EyeCardPlus*) para os jogos *Midi*/ *Mini Baccara*t/ *Punto Banco, Three Card Poker* e *BlackJack*;
- Possibilidade de mostrar as cartas no *LCD* para confirmação de resultados;
- Possibilidade de exportação de informação para registo de jogadas, histórico, vigilância, aplicações externas;
- Não se encontra restrito a uma marca específica de cartas;
- Atualmente suporta os modelos de cartas *Jumbo-index Founier*, *Copag*, *Carta Mundi*, *Aristocrat*, *Bee*, *Paulson,* existindo a possibilidade de ser treinado para outros modelos;
- Funciona com câmera com ligação *USB* ou *IP*.

[12] http://www.cammegh.com/files/Cammegh_Brochure.pdf

2.3.11 SET-Production

A empresa *SET-Production* foi outra das empresas que participou na *ICE Totally Gaming 2014* e que recorre a componentes de computação visual nos seus produtos. A empresa apresenta soluções com um grau de similaridade elevado em relação à *Cammegh* no que toca a produtos relacionados com roletas, sendo estes a gama *Cyclops* [SET-Production, 2014] que conta com os modelos *Simula*, *Classic*, *Slim*, *Slideshow*, *Renaissance* e *Junior* [Figura 13].

Figura 13 – Sistema *Cyclops-Junior* da *SET-Production*[13]

As funcionalidades do produto apresentado pela *SET-Production* apesar de similares com os produtos da *Cammegh* apresentam uma peculiaridade entre eles. Na *Cammegh* o produto de reconhecimento com base em computação visual era substancialmente mais lento que o produto de infravermelhos, sendo que, na *SET-Production* o produto baseado em computação visual era bastante mais rápido em relação ao produto baseado no método de infravermelhos.

[13] http://www.dinaris.cz/foto%20elektro/3model2.png

2.3.12 BHS Böhm

A empresa alemã *BHS Böhm* esteve presente durante a *ICE Totally Gaming 2014* a apresentar o seu sistema de monitorização de jogos de casino *PEye: Table Managment System* [BHS Böhm, 2014]. A empresa apresenta uma solução completa dividida por módulos dentro dos quais se encontra incorporado um módulo similar aos produtos das empresas *Cammegh* e *SET-Production* no que toca ao reconhecimento de bolas numa roleta por meio de computação visual.

Figura 14 – Modelo do sistema *PEye: Table Managment System* da *BHS Böhm*[14]

Durante o decorrer da exposição foi revelado que a empresa estaria a criar um sistema de reconhecimento automático de dados, que no entanto não estava em exposição, uma vez que se encontrava num estágio inicial de elaboração denominado *WildLizzy* [WildLizzy, 2013], tendo disponibilizado apenas a informação necessária para visitar a página do projeto.

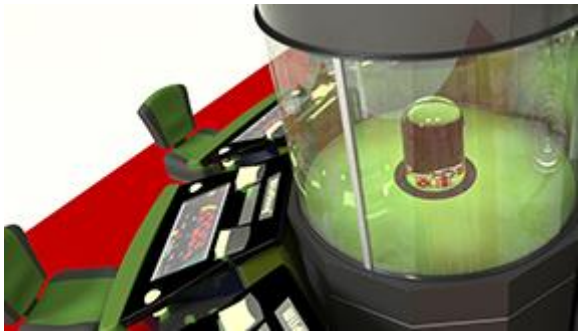


Figura 15 – Modelo do componente *dice robot ´shaker´* do sistema *WildLizzy*[15]

A informação relativa ao *WildLizzy* é bastante escassa sendo apenas referido que este incorpora um lançador de dados automático que uma vez lançados são apresentados aos jogadores por meio de uma câmera de vídeo.

[14] http://www.p-eye.eu/images/TMS/peyetms.jpg
[15] http://wildlizzy.de/images/System_Img/sys_complete_s.png

2.3.13 Stereo Vision Based Dice Recognition System and Stereo Vision Based Dice Recognition Method for Uncontrolled Environments

Esta patente [Zhang amd Hsu, 2012] apresentada por Gee-Sern Hsu e Xun-Jia Zhang demonstra similaridades com o projeto [Hsu *et al.,* 2012] devido ao uso de múltiplas câmeras de forma a criar um modelo do dado que será posteriormente analisado recorrendo a um algoritmo de *clustering* de forma a aferir o valor numérico de cada dado [Figura 16].

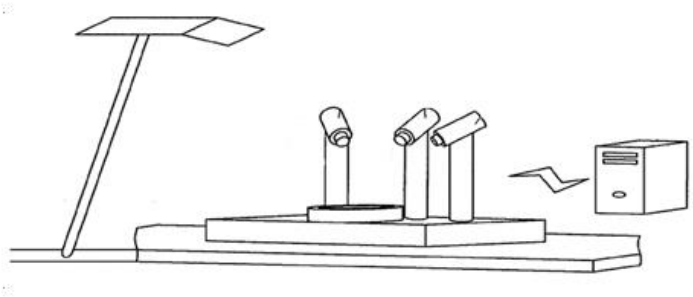


Figura 16 – Modelo conceptual do sistema proposto por Gee-Sern Hsu e Xun-Jia Zhang[16]

Uma vez analisado o procedimento apresentado pela documentação da patente, é possível constatar que esta pode ser considerado como uma fusão dos projetos de Kuo-Yi Huang [Huang, 2007], no que toca ao reconhecimento por meio de *greyscale clustering,* e o projeto realizado por Gee-Sern Hsu (este sendo também um dos coautores da patente), Hsiao-Chia Peng e Shang-Min Yeh [Hsu *et al.*, 2012], no que toca à utilização de múltiplas câmeras para criação de um modelo a analisar.

[16] http://patentimages.storage.googleapis.com/thumbnails/US20120106831A1/US20120106831A1-20120503-D00000.png

2.3.14 Dice-O-Matic mark II – GamersByEmail.com

O *Dice-O-Matic mark II* [GamersByEmail.com, 2013] é uma máquina desenvolvida de forma a gerar combinações aleatórias de dados para o *website GamesByEmail.com* uma vez que os algoritmos utilizados não eram considerados totalmente aleatórios [Hamilton, 2014].

Esta máquina de aproximadamente 2m de altura, base quadrangular de 20,25m2 e 47kg é capaz de gerar 1.3 milhões de combinações de dados por dia [Figura 17].

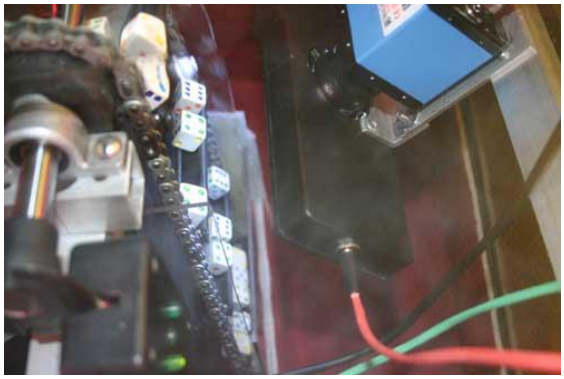


Figura 17 – Pormenor da máquina *Dice-O-Matic mark II*[17]

O reconhecimento dos dados baseia-se numa característica peculiar dos mesmos, criada de forma a facilitar e diminuir a taxa de erro de deteção. Esta característica consiste na diferença de cores de pintas consoante o valor das faces dos dados [Figura 18]. Desta forma caso as pintas dos dados não sejam segmentadas corretamente existe a possibilidade de aferir o número do dado com base na cor das pintas.

Figura 18 – Exemplo de uma amostra do *Dice-O-Matic mark II*[18]

[17] http://static.gamesbyemail.com/News/DiceOMatic/Timing.jpg?633789305620000000
[18] http://static.gamesbyemail.com/News/DiceOMatic/Processed.gif?633793325720000000

3 Conceitos fundamentais

O presente capítulo explica os conceitos considerados fundamentais para a compreensão da dissertação. No entanto, este capítulo não é considerado de leitura integral obrigatória, sendo a sua leitura aconselhável aos leitores que não possuam qualquer nível de conhecimento sobre os conceitos básicos de computação visual, algoritmos de manipulação e análise de imagem por computador ou então que não se encontrem familiarizados com o jogo de casino *Banca Francesa*.

3.1 Representação digital de imagem

Existe um variado número de métodos para adquirir imagens digitais do tipo *raster* (matricial) do mundo real, quer seja pelo uso de câmeras digitais, *scanners*, raios X, entre outros métodos. Em todos os casos enunciados o que nós, humanos, vemos são apenas e só imagens, no entanto, quando transformado para uma representação digital a informação guardada consiste apenas em valores numéricos para cada ponto da imagem recolhida. Na Figura 19 pode-se ver que o espelho do carro, não é nada mais que uma matriz cujo os valores apresentados representam a intensidade em cada ponto (*pixel*).

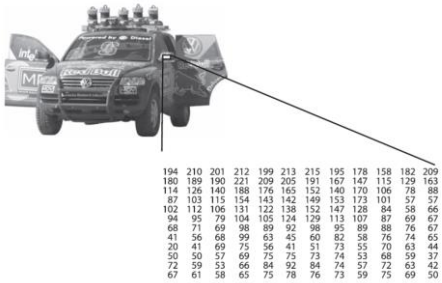


Figura 19 – Representação digital do retrovisor do carro[19]

[19] http://luks.fe.uni-lj.si/sl/studij/KCS/vaje/projekti/dereggi/photos/computer-see.jpeg

A forma como é armazenada a informação pode variar de acordo com o método mais adequado (GRAYSCALE, RGB, HSV entre outros), no entanto no final todas as imagens *raster* dentro do mundo computacional podem ser representadas por uma matriz numérica.

Sendo assim, esta dissertação abordará toda e qualquer imagem utilizada como uma matriz numérica manipulada consoante o espaço de cores referido durante o processo a ser explicado. Os espaços de cores que se encontram mencionados no decorrer da dissertação são os seguintes:

RGB – Formato mais popular maioritariamente devido ao facto de ser assim que o olho humano constrói as cores que visualiza. Sendo as componentes básicas o vermelho (*R*), verde (*G*) e azul (*B*). Uma imagem em formato RGB é normalmente representada por uma matriz tridimensional na qual a terceira coordenada representa a componente do *pixel* definido pelas duas primeiras coordenadas.

GRAYSCALE – O formato *Grayscale* é representado apenas por uma matriz bidimensional, a qual guarda a intensidade de cada ponto para cada *pixel*. Este formato é o mais utilizado para a aplicação de algoritmos de computação visual uma vez a maior parte destes faz apenas uso das diferenças de intensidade entre regiões da imagem. Sendo o seu uso bastante popular, também devido ao facto da complexidade dos algoritmos ser bastante reduzida, pois não se tem de aplicar o método a cada uma das componentes em separado.

HSV – Este formato é constituído pelas componentes, tonalidade (*H*), saturação (*S*) e valor (*V*), sendo esta uma forma mais natural para descrever cores. Este sistema enumera vantagens como por exemplo: a omissão da última componente faz com que os algoritmos sejam menos sensíveis às condições de luz da imagem original. A matriz representativa do formato HSV é homóloga à matriz de representação RGB sendo a terceira coordenada utilizada para a representação da componente do formato.

3.2 Espaço de cores HSV

HSV é uma geometria cilíndrica, com a tonalidade (*hue*) sendo a sua dimensão angular, começando pelo vermelho primário aos 0°, passando para verde primário aos 120° e o azul primário aos 240°, voltando em seguida ao vermelho primário aos 360°.

As cores aditivas primárias e secundárias (vermelho, amarelo, verde, turquesa, azul e magenta) e as misturas lineares entre pares adjacentes das mesmas, às vezes chamadas de cores puras, são organizadas em torno da borda exterior do cilindro com a saturação (*saturation*) a tomar o valor 1.

O eixo vertical central compreende as cores neutras, acromáticas, ou cinzentas, variando de preto com o valor (*value*) a 0, até branco com o valor (*value*) a 1.

A Figura 20 ilustra a representação da geometria cilíndrica do espaço de cores HSV.

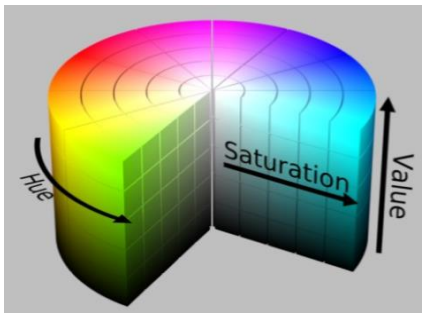


Figura 20 – Geometria cilíndrica do espaço de cores HSV[20]

A maior parte dos algoritmos de computação visual utilizados em imagens coloridas são extensões diretas de algoritmos projetados para imagens a uma escala a preto e branco. Na sua forma mais simples, a utilização destes algoritmos passa cada componente do espaço de cores RGB pelo algoritmo separadamente. Devido ao facto das componentes vermelho (R), verde (G) e azul (B) de uma imagem digital estarem correlacionadas com a quantidade de luz que incide sobre a cena da imagem torna-se bastante complicado fazer a segmentação de um objeto da cena com base na informação das componentes RGB. Desta forma é comummente utilizada a representação HSV de uma imagem de forma a ser possível segmentar regiões da cena com base na sua cor, independentemente se este se encontra muito ou pouco iluminado. A transformação de RGB para HSV é efetuada segundo as seguintes equações para componentes RGB a variar de 0 a 1,

$$H = \begin{cases} 60 \times \dfrac{G-B}{MAX-MIN} + 0, & se\ MAX = R \\ & \wedge\ G \geq B \\ 60 \times \dfrac{G-B}{MAX-MIN} + 360, & se\ MAX = R \\ & \wedge\ G < B \\ 60 \times \dfrac{B-R}{MAX-MIN} + 120, & se\ MAX = G \\ 60 \times \dfrac{R-G}{MAX-MIN} + 240, & se\ MAX = B \end{cases} \quad (1)$$

$$S = \frac{MAX - MIN}{MAX} \quad (2)$$

$$V = MAX \quad (3)$$

Em que,

H – Componente da tonalidade (*hue*) da representação HSV;

S – Componente da saturação (*saturation*) da representação HSV;

V – Componente do valor (*value*) da representação HSV;

R – Componente vermelha (*Red*) da representação RGB;

G – Componente verde (*Green*) da representação RGB;

B – Componente azul (*Blue*) da representação RGB

MAX – Máximo valor entre as componentes RGB para um dado *pixel*;

MIN – Mínimo valor entre as componentes RGB para um dado *pixel*.

[20]http://upload.wikimedia.org/wikipedia/commons/thumb/0/0d/HSV_color_solid_cylinder_alpha_lowgamma.png/197px-HSV_color_solid_cylinder_alpha_lowgamma.png

Sendo que a conversão inversa (de RGB para HSV) é dada pelas seguintes equações,

$$se\ S = 0 \longrightarrow R = G = B = V \tag{4}$$

Senão,

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor mod\ 6$$

$$f = \frac{H}{60} - H_i$$

$$p = V(1 - S)$$

$$q = V(1 - fS) \tag{5}$$

$$t = V(1 - (1 - f)S)$$

$$se\ H_i = 0 \rightarrow R = V, G = t, B = p$$

$$se\ H_i = 1 \rightarrow R = q, G = V, B = p$$

$$se\ H_i = 2 \rightarrow R = p, G = V, B = t$$

$$se\ H_i = 3 \rightarrow R = p, G = q, B = V$$

$$se\ H_i = 4 \rightarrow R = t, G = p, B = V$$

$$se\ H_i = 5 \rightarrow R = V, G = +, B = q$$

Em que,
H – Componente da tonalidade (*hue*) da representação HSV;
S – Componente da saturação (*saturation*) da representação HSV;
V – Componente do valor (*value*) da representação HSV;
R – Componente vermelha (*Red*) da representação RGB;
G – Componente verde (*Green*) da representação RGB;
B – Componente azul (*Blue*) da representação RGB;
H_i, f, p, q, t – Variáveis auxiliares.

Os valores resultantes da conversão de HSV para RGB encontram-se compreendidos entre 0 e 1 para cada uma das componentes RGB.

3.3 Conceito de *Kernel*

Uma imagem digital, no âmbito da dissertação, é abordada como sendo uma matriz com informação referente a cada *pixel.* Posto isto, é comum o uso de operações sobre a matriz da imagem de forma a serem obtidos os resultados pretendidos. Muitas das operações utilizadas recorre a uma matriz máscara auxiliar denominada de *kernel*.

Uma matriz *kernel* é uma matriz que possui os valores que indicam o quanto os *pixels* vizinhos, e o próprio *pixel*, irão influenciar o valor do novo *pixel* recalculado. De um ponto de vista matemático é realizada uma média ponderada com os valores especificados.

A Figura 21 demonstra um exemplo simples da aplicação de um filtro de aumento de contraste de uma imagem.

Figura 21 – Aplicação de um filtro (uso de *kernel*) para aumento de contraste[21]

O processo de filtragem pode ser expresso sobre a seguinte fórmula para cada *pixel* da imagem,

$$g(x,y) = 5 \times f(x,y) - [f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1)] \quad (6)$$

Ou então recorrendo ao conceito de *kernel* e expresso por,

$$g(x,y) = f(x,y) \times M, onde\ M = \begin{matrix} {}^{x}/_{y} \\ -1 \\ 0 \\ +1 \end{matrix} \begin{matrix} \begin{matrix} -1 & 0 & +1 \end{matrix} \\ \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \end{matrix} \quad (7)$$

Em que,
$g(x,y)$ – *Pixel* na posição x, y da imagem resultante da operação;
$f(x,y)$ – *Pixel* na posição x, y da imagem original;
M – Matriz máscara a ser utilizada (*kernel)*.

A operação utilizando a matriz *kernel* coloca o centro deste no *pixel* a ser recalculado e, de seguida, efetua o somatório das multiplicações dos *pixels* sobrepostos pelo *kernel*. A utilização da notação com base no uso de *kernel* facilita quando as matrizes máscara a serem utilizadas assumem tamanhos maiores.

[21]http://static.tumblr.com/79149353ca0c887f7766aed2ff9ef5b5/9hiqgmd/4M1n2y7rl/tumblr_static_eva002.png

3.4 Operações de *Thresholding*

A aplicação de um *threshold* é o método mais simples de segmentação existente no processamento de imagem. Um exemplo de um *threshold* consiste em separar as regiões de um objeto desejado para posterior análise. Esta separação é baseada em variações de intensidade entre os *pixels* do objeto e os pixéis do fundo em que este se encontra inserido.

De forma a separar os *pixels* de interesse dos restantes (que serão eventualmente rejeitados) é realizada uma comparação entre a intensidade de cada *pixel* e um respetivo *threshold* (determinado de acordo com o problema em questão). Uma vez separados os *pixels* propriamente, estes podem tomar um determinado valor de forma a serem identificados. Por exemplo, para os valores acima do *threshold* tomam o valor 0 e para os valores abaixo do *threshold* o valor 255.

As Figura 23 a Figura 27 demonstram o resultado da aplicação, em conjugação com a equação a que se referem, dos diferentes tipos de *threshold* comummente utilizados quando aplicados à Figura 22 com um valor de *threshold* de 128.

Figura 22 – Imagem a ser aplicado o *threshold*

Figura 23 – *Threshold* binário

Threshold binário

$$g(x,y) = \begin{cases} maxVal & f(x,y) > thresh \\ 0 & f(x,y) \leq thresh \end{cases} \quad (8)$$

Figura 24 – *Threshold* binário, invertido

Threshold binário, invertido

$$g(x,y) = \begin{cases} 0 & f(x,y) > thresh \\ maxVal & f(x,y) \leq thresh \end{cases} \quad (9)$$

Figura 25 – *Threshold* truncado

Threshold truncado

$$g(x,y) = \begin{cases} thresh & f(x,y) > thresh \\ f(x,y) & f(x,y) \leq thresh \end{cases} \quad (10)$$

Figura 26 – *Threshold* para zero

Threshold para zero

$$g(x,y) = \begin{cases} f(x,y) & f(x,y) > thresh \\ 0 & f(x,y) \leq thresh \end{cases} \quad (11)$$

Threshold para zero, invertido

$$g(x,y) = \begin{cases} 0 & \mathrm{f}(x,y) > thresh \\ f(x,y) & f(x,y) \leq thresh \end{cases} \quad (12)$$

Figura 27 - *Threshold* para zero, invertido

Em que,
$g(x,y)$ – *Pixel* na posição x, y da imagem resultante da operação;
$f(x,y)$ – *Pixel* na posição x, y da imagem original;
$maxVal$ – Valor de intensidade máximo possível para um *pixel* (neste caso 255);
$thresh$ – Limiar estipulado para a aplicação do *threshold* (neste caso 128).

3.5 Operações morfológicas básicas

Imagens binárias podem conter um número elevado de imperfeições, normalmente produzidas pelo uso de uma operação de *thresholding* simples, as quais se encontram normalmente distorcidas pelo ruído na imagem e pela textura da mesma. As operações morfológicas têm assim o intuito de remover essas imperfeições tendo em conta a forma e estrutura da imagem a ser manipulada.

As operações morfológicas básicas apresentadas em seguida são as operações que se encontram na dissertação, ou então operações que são necessárias para explicação de uma posterior operação morfológica. As operações dilatação, erosão, abertura e fecho encontram-se aplicadas à imagem modelo presente na Figura 28.

Figura 28 – Imagem modelo[22]

[22] http://docs.opencv.org/_images/Morphology_1_Tutorial_Theory_Original_Image.png

3.5.1 Dilatação

Esta operação consiste na convolução da imagem da Figura 28 através do uso de um *kernel*, o qual pode tomar qualquer tamanho ou forma, normalmente um quadrado ou um círculo. O *kernel* tem definido um ponto-âncora, sendo este normalmente localizado no seu centro. À medida que o *kernel* é passado pela imagem e calculado o *pixel* de valor máximo sobreposto pelo *kernel* e seguidamente substituído pelo *pixel* sobreposto pelo ponto-âncora na imagem. Como pode ser deduzido esta operação de maximização causa com que as áreas da imagem com mais brilho "cresçam" sendo o seu resultado apresentado na Figura 29.

Figura 29 – Resultado da operação de dilatação[23]

3.5.2 Erosão

A operação de erosão é bastante similar à operação de dilatação, sendo que esta é utilizada para calcular o mínimo local dentro da área do *kernel.* Sendo assim, à medida que o *kernel* é passado pela imagem da Figura 28 é calculado o *pixel* de valor mínimo sobreposto pelo *kernel* e substituído na posição do ponto-âncora na imagem. Analogamente ao exemplo da dilatação, a operação de erosão foi aplicada à imagem da Figura 28 sendo o seu resultado apresentado na Figura 30.

Figura 30 – Resultado da operação de erosão[24]

[23] http://docs.opencv.org/_images/Morphology_1_Tutorial_Theory_Dilation.png
[24] http://docs.opencv.org/_images/Morphology_1_Tutorial_Theory_Erosion.png

3.5.3 Abertura

Uma operação de abertura é composta pela aplicação de uma operação de erosão seguida de uma operação de dilatação

$$g = abertura(f, kernel) = dilatação(erosão(f, kernel), kernel) \quad (13)$$

Em que,
g – Imagem resultante da aplicação da operação de abertura;
f – Imagem original.

Esta operação é bastante útil para remover pequenos objetos, assumindo que os objetos são mais brilhantes (maior intensidade) que o fundo em que se encontram inseridos. O resultado da aplicação da operação de abertura na imagem da Figura 28 encontra-se representado na Figura 31.

Figura 31 – Resultado da operação de abertura[25]

3.5.4 Fecho

Uma operação de fecho é composta por a aplicação de uma operação de dilatação seguida de uma operação de erosão.

$$g = fecho(f, kernel) = erosão(dilatação(f, kernel), kernel) \quad (14)$$

Em que,
g – Imagem resultante da aplicação da operação de abertura;
f – Imagem original.

[25] http://docs.opencv.org/_images/Morphology_2_Tutorial_Theory_Opening.png

Esta operação é bastante útil para remover pequenos "buracos" regiões de baixa intensidade numa imagem. O resultado da aplicação da operação de fecho na imagem da Figura 28 encontra-se representado na Figura 32.

Figura 32 – Resultado da operação de fecho[26]

3.6 Redução polinomial de regiões

A redução polinomial de regiões pretende simplificar a representação de áreas que delimitam uma certa região obtida por um determinado algoritmo, como por exemplo o algoritmo de descoberta de contornos [Susuki and Abe, 1985]. Esta simplificação é efetuada através da utilização do algoritmo de Douglas-Pecker [Heckbert and Garland, 1997].

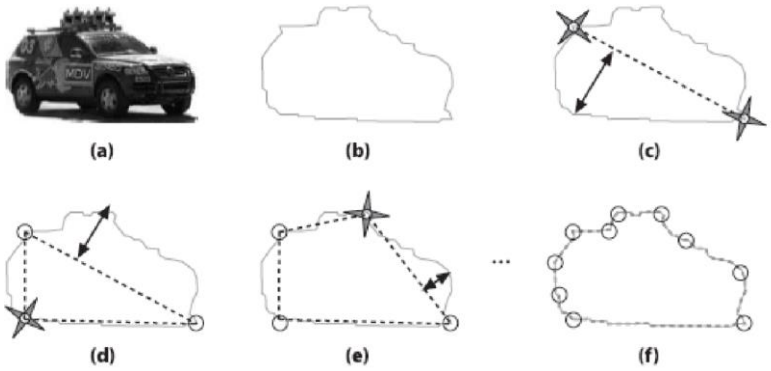


Figura 33 – Representação visual do algoritmo Douglas-Pecker[27]

O algoritmo de Douglas-Pecker procura pelos dois pontos mais distantes [Figura 33 - (c)] no contorno que define a região que representa o carro [Figura 33 - (b)]. De seguida é procurado o ponto mais distante pertencente ao contorno original em relação ao segmento de reta gerado pelos dois pontos anteriores [Figura 33 - (d)]. Depois e iterativamente é procurado outro ponto pertencente ao contorno original que se encontre o mais distante do polígono gerado [Figura 33 - (e)].

Este processo repete-se até ser satisfeita a condição de paragem do algoritmo que neste caso é a inexistência de pontos do contorno original a uma distância mínima do novo polígono gerado [Figura 33 - (f)], sendo assim, o novo contorno é definido pelo polígono resultante.

[26] http://docs.opencv.org/_images/Morphology_2_Tutorial_Theory_Closing.png
[27] http://shop.oreilly.com/product/0636920022497.do

3.7 Algoritmo k-means clustering

O algoritmo de *k-means clustering* [Ng, 2014] é um método de quantização vetorial, originalmente usado para processamento de sinais. O algoritmo tem como objetivo particionar um número N de observações num número K de partições, sendo que cada observação pertence à partição que tiver a média mais aproximada do seu valor. A seguinte explicação do algoritmo *kmeans clustering* pretende explicar como se processa o algoritmo de forma intuitiva, sendo os conceitos matemáticos maioritariamente omitidos.

Considerando que é necessário agrupar a informação presente na Figura 34 em dois grupos (*clusters*).

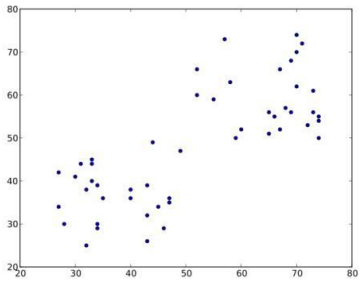


Figura 34 – Conjunto de informação modelo[28]

O primeiro passo consiste em escolher dois pontos aleatórios, C_1 e C_2 [Figura 35], sendo que estes costumam ser quaisquer dois pontos do conjunto de informação existente.

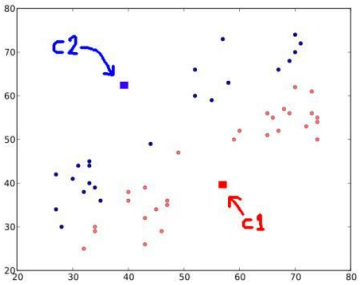


Figura 35 – Escolha dos pontos C_1 e C_2[29]

O segundo passo consiste em calcular a distância de cada um dos pontos do conjunto em relação aos pontos C_1 e C_2. Se o ponto se encontrar mais próximo de C_1, então o resultado é rotulado de '0'. Caso o ponto se encontrar mais próximo de C_2, então este é rotulado de '1'. Os rótulos são incrementados consoante os pontos C existentes.

[28] http://docs.opencv.org/trunk/_images/testdata.jpg
[29] http://docs.opencv.org/trunk/_images/initial_labelling.jpg

O terceiro passo consiste no cálculo das médias para cada conjunto de pontos rotulado, neste caso '0' e '1'. Os resultados destas novas médias serão considerados os novos pontos, C_1 e C_2 respetivamente. A Figura 36 apresenta a posição dos novos pontos, C_1 e C_2.

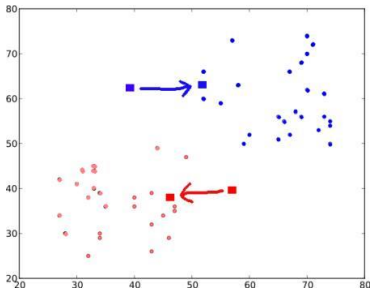


Figura 36 – Novos pontos calculados C_1 e C_2[30]

Desta forma, o segundo e terceiro passos são repetidos até os pontos C_1 e C_2 convirjam para pontos fixos. Este processo também pode ser parado com a introdução de um critério de paragem, como por exemplo o número máximo de iterações, ou então o nível de precisão estipulado se encontre alcançado. Estes pontos fixos são tais que a soma entre a distâncias dos pontos dos conjuntos e os pontos C correspondentes é mínima, ou seja, para o caso em particular, a soma das distâncias entre C_1 e os pontos vermelhos e C_2 e os pontos azuis é mínimo.

O resultado final do algoritmo encontra-se visualmente representado na Figura 37, onde o conjunto de informação se encontra separado em 2 conjuntos (vermelho e azul).

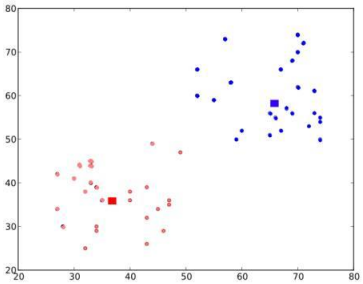


Figura 37 – Resultado final do algoritmo *k-mean clustering*[31]

3.8 Algoritmo background subtraction

Um algoritmo de *background subtraction* é um dos maiores processos de pré-processamento em determinadas aplicações de computação visual. Por exemplo no caso da contagem de visitantes onde uma câmera estática recolhe o número de visitantes que entra e sai de uma determinada sala, ou então uma câmera de controlo rodoviário que recolhe informação referente aos veículos que circulam na estrada à qual ela se encontra associada.

Generalizando o conceito de *background subtraction*, como o nome sugere, baseia-se na subtração da imagem atual capturada pelo dispositivo e da imagem de fundo, denominada de

[30] http://docs.opencv.org/trunk/_images/update_centroid.jpg
[31] http://docs.opencv.org/trunk/_images/final_clusters.jpg

modelo de fundo. A Figura 38 ilustra o resultado da hipotética aplicação de um algoritmo de *background subtraction*.

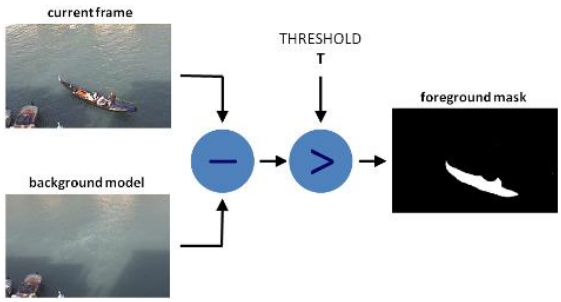


Figura 38 – Algoritmo de *background subtraction* genérico[32]

Como pode ser verificado o resultado obtido representa a região do barco, o qual não existe na imagem do modelo de fundo. Por norma os algoritmos de *background subtraction* são algoritmos dinâmicos, ou seja, é possível atualizar o modelo de fundo consoante o tempo, de forma a ter em conta objetos inseridos na cena, que permaneçam durante um determinado espaço de tempo definido, de modo a serem considerados parte integrante do fundo.

3.9 Jogo Banca Francesa

O jogo *Banca Francesa* [Jogo Responsável, 2014] é um jogo de fortuna ou azar, bancado, jogado maioritariamente em casinos portugueses, onde o seu objetivo é prever o resultado do lançamento de três dados lançados pelo funcionário do casino denominado de *croupier,* que em gíria é denominado de "cavalinho" no jogo *Banca Francesa*.

Nos casinos físicos este jogo é jogado com 3 dados, em bancas simples ou duplas, lançados por um pagador (neste caso o *croupier*). A Figura 39 apresenta uma típica mesa do jogo em Portugal.

Figura 39 – Mesa típica do jogo *Banca Francesa*[33]

O jogador pode jogar numa das 3 hipóteses: ASES, PEQUENOS e GRANDES.

[32] http://docs.opencv.org/trunk/_images/Background_Subtraction_Tutorial_Scheme.png
[33] http://www.casino-lisboa.pt/wp-content/uploads/2014/07/395.jpg

Os dados são lançados as vezes necessárias até que o total das suas pintas (nas faces superiores) perfaça 5, 6 ou 7 encaixando na categoria do PEQUENO, 14, 15 ou 16 na categoria do GRANDE. Caso o jogador acerte na aposta numa destas hipóteses este recebe a importância igual à jogada na hipótese.

No caso de saírem ASES (soma das pintas nas faces superiores dos três dados perfazer 3, ou seja, tudo a uns), os jogadores que tiverem jogado nesta hipótese recebem 61 vezes o valor da sua aposta.

Enquanto se vão lançando os dados, e até que haja um resultado que decida (PEQUENO, GRANDE ou ASES), o jogador pode apostar, aumentar, diminuir ou mesmo, mudar a aposta já efetuada.

As hipóteses e os correspondentes prémios encontram-se apresentados na tabela.

Tabela 1 – Tipos de aposta no jogo *Banca Francesa*

Aposta	**Dados**	**Pagamentos da casa (*odds*)**
ASES	Soma dos valores é 3	61 vezes o valor da aposta
PEQUENO	Soma dos valores é 5, 6 ou 7	Valor igual ao da aposta
GRANDE	Soma dos valores é 14, 15 ou 16	Valor igual ao da aposta

4 Proposta de algoritmos de controlo e deteção de jogadas no jogo *Banca Francesa*

A solução proposta na dissertação tem como intuito apresentar as componentes de um sistema capaz de possibilitar a deteção de dados numa mesa de jogo de Banca Francesa, assim como verificar se foi realizado algum movimento ilegal durante o decorrer do jogo com o auxílio de métodos de computação visual. O sistema proposto encontra-se representado na Figura 40.

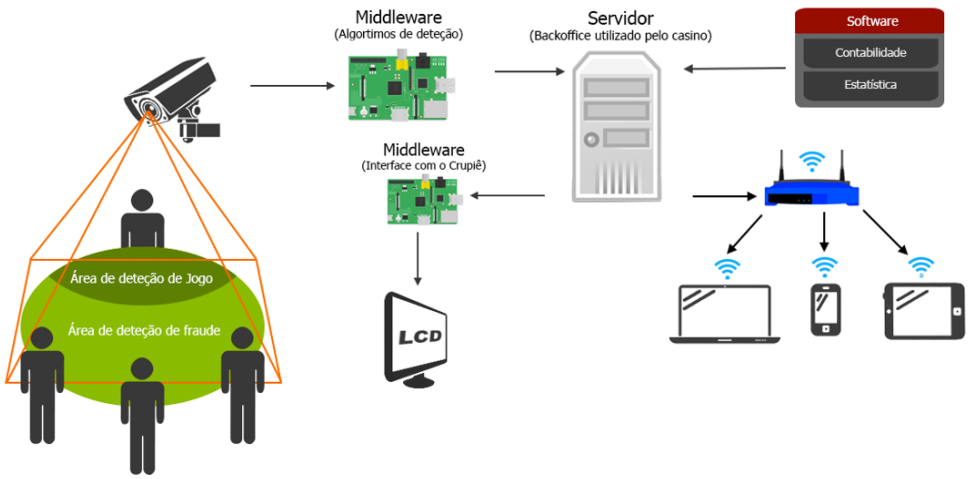


Figura 40 – Sistema proposto ao *Passaporte do Empreendedorismo*

O sistema idealizado encontra-se dividido em 4 componentes principais, sendo elas:

***Middleware* (algoritmos de deteção)** – Um computador de tamanho reduzido e de baixo custo de produção onde todo o seu *hardware* se encontra embutido numa única placa, ligado

a uma câmera digital. Esta componente é responsável pela execução do conjunto de algoritmos propostos na dissertação.

***Middleware* (Interface com o crupiê)** – Dispositivo similar ao utilizado para utilização dos algoritmos de deteção, no entanto este é responsável por realizar a ligação entre o responsável por uma determinada mesa de jogo de *Banca Francesa* e a aplicação central do servidor do casino.

***Servidor* (Serviço de controlo)** – No servidor central encontrar-se-á um programa responsável pela gestão de informação e redireccionamento para respetivos canais. Este programa é responsável também por averiguar o estado das mesas, o qual pode ser verificado através de dispositivos móveis.

***Software* (Contabilidade e Estatística)** – Programa que faz uso da informação recebida pelas diferentes mesas de jogo de forma a gerar dados significativos sobre as operações ocorridas, quer sejam estas balanço monetário da mesa jogo, número de jogadas efetuadas, quantidade de infrações detetadas, entre outros.

O *Middleware* (algoritmos de deteção) o objeto principal da proposta da dissertação, encontra-se delineado com as responsabilidades presentes na Figura 41.

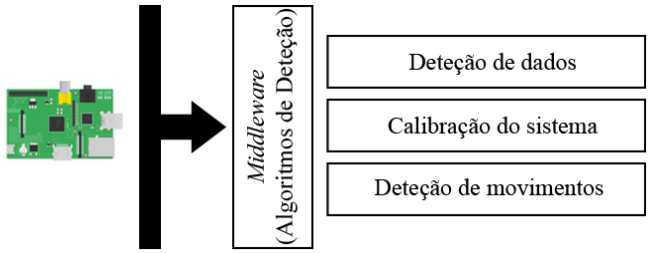


Figura 41 – Algoritmos/Responsabilidades do *Middleware* (algoritmos de deteção)

Assim sendo o seguinte capítulo encontra-se divido nas seguintes propostas teóricas:

Dice Sample Generator (DSG) – Solução proposta responsável pela criação de casos de teste (dados em cima de uma mesa de jogo) em grande escala de forma a comprovar o funcionamento dos algoritmos de deteção de dados.

***Dice Sample Analyzer* (DSA)** – Solução proposta responsável pela análise da "área de deteção de jogo" [Figura 40] onde são detetados os valores dos dados presentes em jogo.

***Dice Calibration* (DC)** – Solução proposta responsável pela calibração do sistema de forma a ser possível a integração de novos tipos de dados com métricas diferentes, assim como a calibração inicial do dispositivo aquando da instalação do sistema.

***Motion Detection* (MD)** – Solução proposta responsável pela análise dos movimentos que ocorrem na mesa de jogo, desde da paragem dos dados, posição relativa das fichas e às ações dos jogadores durante o decorrer do jogo.

4.1 Dice Sample Generator (DSG)

De forma a ser possível criar casos de teste suficientes para uma primeira triagem do algoritmo de reconhecimento de dados proposto foi desenvolvida a ferramenta intitulada de *Dice Sample Generator* (DSG). Esta ferramenta tem como função gerar um determinado número, $N_s = [1, 5000]$, de amostras (imagens) de uma mesa de jogo onde se encontra presente um determinado número de dados definidos pelos limites, $N_{d1} = [1, 20] \wedge N_{d2} = [N_{d1}, 20]$.

O sistema DSG foi desenvolvido em Java, uma vez que esta linguagem oferece manipulação básica de imagens e *parsing* de ficheiros XML sem qualquer tipo de dependências externas, e também devido à sua portabilidade entre sistemas operativos.

O sistema DSG baseia-se na leitura de um ficheiro XML que contem toda a informação referente aos recursos a serem utilizados para gerar as amostras, uma vez carregados os recursos necessários para memória o DSG procede à criação de amostras com base nos parâmetros definidos anteriormente (N_s, N_{d1}, N_{d2}).

4.1.1 Modelo XML: exemplo de entrada

O ficheiro XML de entrada para o DSG contém a informação referente à base de jogo a ser utilizada assim como a informação do dado a ser reproduzido nas amostras [Código 1].

```
<?xml version="1.0"?>
<DSGinfo version="1.0">
    <mat>
            <image>base.jpg</image>
            <area type="rectangle">
                    <point x="37" y="41"/>
                    <point x="1639" y="1167"/>
            </area>
    </mat>
    <die>
            <metrics>
                    <metric id="image_edge" unit="px">110</metric>
                    <metric id="real_edge" unit="cm">1.90</metric>
                    <metric id="dot" unit="cm">0.50</metric>
                    <metric id="A" unit="cm">0.55</metric>
                    <metric id="B" unit="cm">0.80</metric>
                    <metric id="C" unit="cm">1.10</metric>
                    <metric id="D" unit="cm">1.60</metric>
            </metrics>
            <faces>
                    <face value="1">1.jpg</face>
                    <face value="2">2.jpg</face>
                    <face value="3">3.jpg</face>
                    <face value="4">4.jpg</face>
                    <face value="5">5.jpg</face>
                    <face value="6">6.jpg</face>
            </faces>
    </die>
</DSGinfo>
```

Código 1 – Exemplo de XML do DSG

A base da mesa de jogo é definida pelo elemento `</mat>` o qual especifica a imagem a ser utilizada como recurso `<image>`, assim como a área de jogo onde os dados podem aparecer na amostra `<area>`. Neste caso a área definida como área de jogo é uma secção retangular da imagem base.

O dado é definido pelo elemento `<die>`, este elemento é constituído pelas métricas do dado `<metrics>,` sendo que estes valores indicam as proporções reais do dado assim como o seu tamanho em *pixels* a ser utilizado nas amostras a serem geradas. A informação referente ao dado conta também com a especificação da localização das faces `<faces>` a serem utilizadas como recurso para gerar as amostras.

As imagens utilizadas no DSG foram adquiridas utilizando a câmera *Logitech C270 HD*. A imagem dos dados vermelhos translúcidos encontra-se representada na Figura 42.

Figura 42 – Faces de um dado utilizadas como recursos do DSG

A Figura 43 representa a base utilizada para simular uma mesa secção da mesa de jogo, sendo esta constituída por uma folha de cartolina verde presa a um retângulo de cortiça.

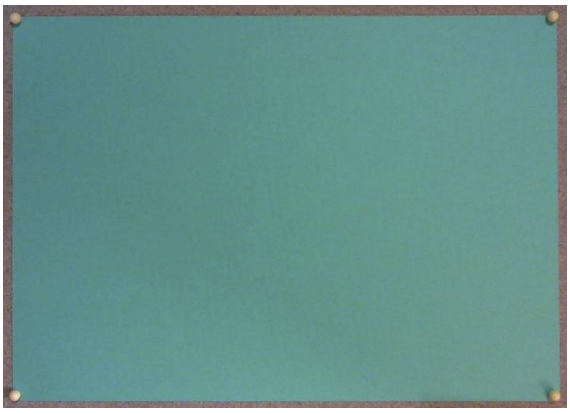


Figura 43 – Base de jogo utilizada como recurso do DSG

4.1.2 Método de produção de amostras

O método de produção de amostras consiste na colocação dos dados na área de jogo de maneira a que nenhum destes se intercete. A Figura 44 ilustra o processo através de um fluxograma para a criação de uma amostra do DSG.

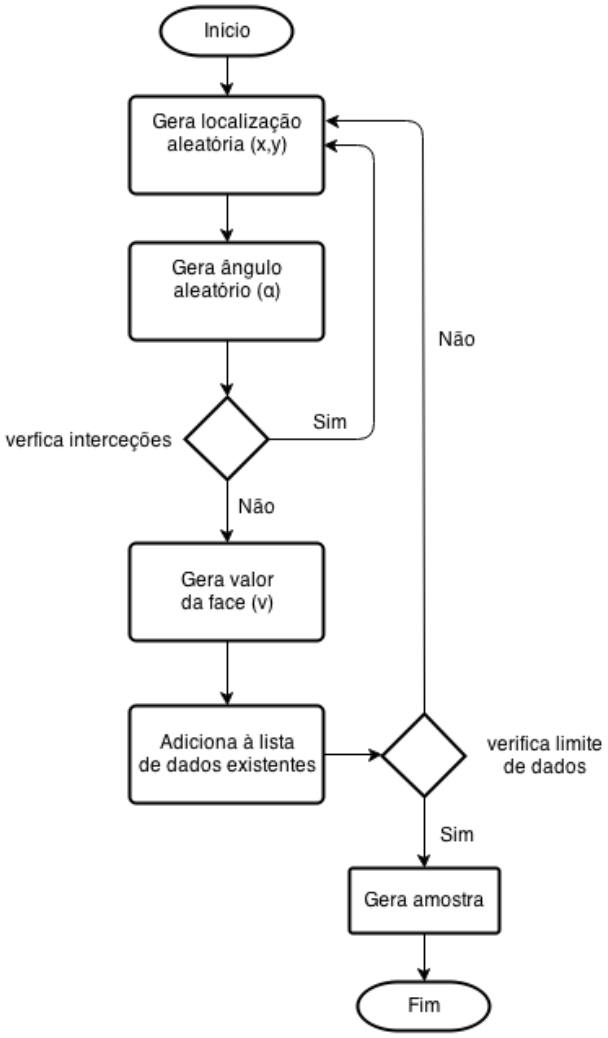


Figura 44 – Fluxograma de produção de uma amostra do DSG

Desta forma para cada amostra é utilizada uma lista, L_d, dos dados presentes na mesa de jogo, que inicialmente se encontra vazia. Em seguida é gerada aleatoriamente uma posição (x, y) compreendida dentro dos limites da área de jogo especificada no ficheiro XML e tendo em conta o tamanho do dado (de forma a que o dado não fique parcialmente fora da área de jogo), assim como, um ângulo sobre o qual o dado se encontra rodado, α.

O dado candidato gerado é avaliado em relação aos dados já presentes na área de jogo, ou seja os dados da lista L_d, de forma a verificar se existem interceções com os mesmos [Figura 45].

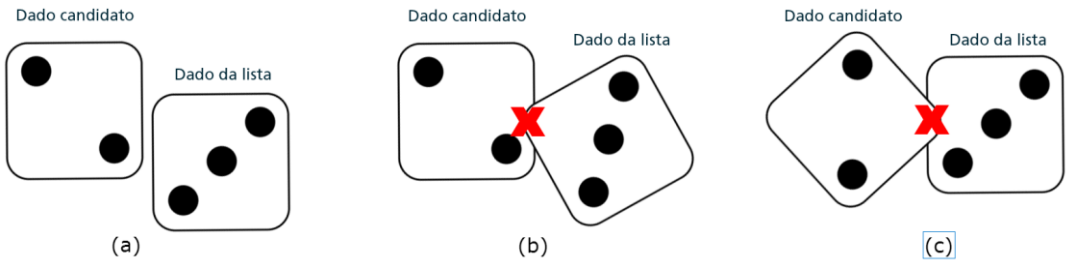


Figura 45 – Exemplos de casos possíveis durante a colocação de dados

As interceções são verificadas com base nos quatro pontos que definem os vértices do novo dado em relação ao quadrado que define o dado já existente [Figura 45 - (b)], e vice-versa [Figura 45 - (c)].

Os quatro pontos que definem os vértices são calculados com base no centro do dado (x, y) e o ângulo de rotação, α, da seguinte forma,

$$Vx_j = \left(x + \cos\left(\alpha - \frac{\pi}{4}\right) + \frac{\pi}{2} \times j\right) \times r$$
$$Vy_j = \left(y + \sin\left(\alpha - \frac{\pi}{4}\right) + \frac{\pi}{2} \times j\right) \times r \quad (15)$$

Em que,
Vx_j - Coordenada x do vértice com índice j;
Vy_j - Coordenada y do vértice com índice j;
x - Coordenada x do centro do dado candidato;
y – Coordenada y do centro do dado candidato;
α – Ângulo de rotação do dado candidato;
j – Índice $j = [0,4]$;
r – Metade da diagonal do quadrado que forma o dado;
π – Constante PI.

Caso uma das interceções seja verificada o dado candidato é descartado, e caso nenhuma interceção seja verificada este é adicionado à lista L_d. Este processo repete-se até que o limite de dados para a amostra seja atingido.

Uma vez gerados todos os dados para uma dada amostra esta é exportada em formato JPG [Figura 46] para uma localização predefinida com o nome referente à sua posição no espectro de amostras gerado.

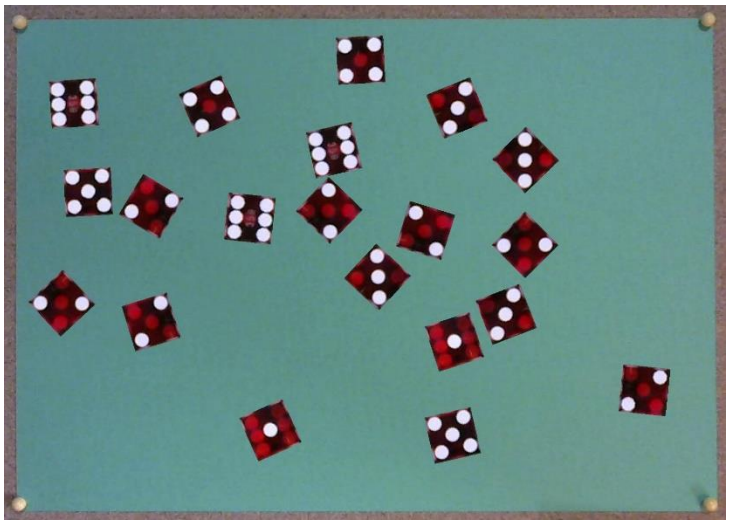


Figura 46 – Amostra aleatória gerada pelo DSG (*image95.jpg*)

Em paralelo é gerado um ficheiro de texto que indica para cada amostra qual a quantidade de dados presentes na área de jogo para cada uma das faces, este ficheiro é usado posteriormente para verificar se os resultados do algoritmo de deteção de dados correspondem à realidade sem ser necessária uma verificação manual para cada uma das amostras.

4.2 Dice Sample Analyzer (DSA)

A ferramenta *Dice Sample Analyzer* (DSA) é a componente principal da proposta apresentada nesta dissertação. Este sistema tem como intuito ser capaz de reconhecer o valor de um certo número indeterminado de dados, N, presentes na cena.

O reconhecimento dos dados é feito através da análise das pintas das faces presentes na cena, tentando compreender as distâncias das pintas de forma a gerar possíveis padrões que representem as configurações dos diferentes tipos de faces de dados possíveis.

O sistema DSA foi elaborado em C++ devido a esta ser a linguagem sobre a qual a biblioteca OpenCV foi desenvolvida, assim como esta é também considerada a sua interface primária. Apesar da existência de *wrappers* para outras linguagens como Java, Python, C# entre outras, o desenvolvimento em C++ foi o optado uma vez que a documentação apresentada, assim como exemplos práticos encontram-se maioritariamente escritos nessa linguagem, agilizando assim desta forma o desenvolvimento da aplicação.

A solução apresentada nesta ferramenta tem como principais características a sua rapidez de execução, sendo capaz de aferir os valores dos dados presentes na cena em tempo real, assim como a versatilidade com que esta ópera, sendo possível realizar a calibração da mesma de forma a conseguir incorporar diferentes tipos de dados de modo a estes serem analisados.

4.2.1 *Workflow* do processo

A solução proposta para análise de dados (DSA) encontra-se dividida em duas subcomponentes: a primeira subcomponente engloba todo o processo de pré-processamento de imagem onde todo o sistema é responsável por fazer realçar as características de interesse das imagens que este recebe como *input*. Esta subcomponente é maioritariamente baseada na implementação encadeada de métodos disponibilizados pela biblioteca OpenCV de forma a obter-se o resultado final. A segunda subcomponente recaí sobre o uso de elementos de segmentação de imagem de forma a extrair a informação pretendida da imagem pré-processada. Neste caso a localização de cada uma das pintas presentes de maneira a aferir o valor dos dados, com base na análise dos padrões possíveis de gerar. A Figura 47 apresenta o *workflow* do processo utilizado para o DSA, expondo sucintamente os sub-blocos de cada uma dos subcomponentes que formam o DSA.

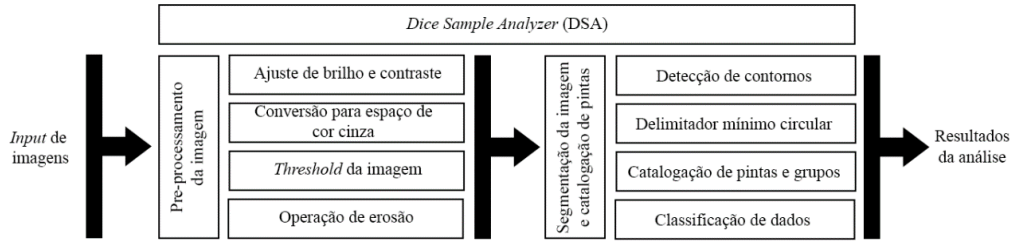


Figura 47 – Diagrama do *workflow* do processo

O sistema DSA é um sistema que foi evoluindo durante o decorrer da elaboração da dissertação, sendo que este primariamente focou-se na análise de dados das amostras estáticas criadas pelo DSG. Uma vez validado o reconhecimento de amostras geradas pelo DSG este foi posteriormente adaptado para análise em tempo real, com base no *input* de uma câmera (*Logitech C270 Webcam HD*) de forma a ser analisado o seu comportamento em casos reais e resolver as problemáticas associadas, como por exemplo as reflexões causadas pelos diferentes tipos de iluminação.

4.2.2 Pré-processamento da imagem

De forma a adquirir as características chave, neste caso as pintas, dos dados presentes na cena, é necessário proceder ao tratamento da imagem de *input* para que essas mesmas características se tornem mais evidentes a nível computacional. De modo a obter esse resultado foi aplicado à imagem original um conjunto de algoritmos no qual a sua resultante apenas contém as pintas dos dados, eliminando assim toda a sua envolvente, a qual neste caso foi considerada como ruído na imagem.

O processo de pré-processamento de imagem é constituído pelos seguintes algoritmos:

- Ajuste de brilho e contraste da imagem;
- Conversão para o espaço de cores cinza;
- *Threshold* da imagem;
- Operação de erosão da imagem.

4.2.2.1 Ajuste de brilho e contraste

O principal objetivo deste passo consiste na alteração das componentes de brilho e contraste da imagem de maneira a homogeneizar as cores que simbolizam as pintas dos dados presentes na cena, para que, desta forma se torne possível uma segmentação das pintas mais eficaz.

O método utilizado para ajustar o brilho e contraste da imagem baseia-se na explicação dada por Richard Szeliki [Szeliki, 2010] onde,

$$g(i,j) = \alpha \times f(i,j) + \beta \qquad (16)$$

Em que,
$g(i,j)$ – *Pixel* (i,j) da imagem resultante do ajuste de brilho e contraste;
$f(i,j)$ – *Pixel* (i,j) da imagem original;
α – Parâmetro *gain* responsável pelo controlo do contraste sendo que $\alpha > 0$;
β – Parâmetro *bias* responsável pelo controlo do brilho.

Os parâmetros α e β foram iterados de maneira a que as áreas correspondentes aos *pixels* das pintas dos dados tomassem um valor próximo de um *pixel* de cor branca ($R = 255, B = 255, G = 255$). Os valores foram iterados de 1 em 1 até se encontrados os valores para ambas as variáveis que reproduzissem o resultado pretendido. Sendo os valores ideias apresentados na tabela [Tabela 2],

Tabela 2 – Valores utilizados para *gain* (α) e *bias* (β)

Variável	Valor
α	3
β	-200

O ajuste de brilho e contraste uma vez aplicado a uma amostra produz o resultado apresentado na Figura 48.

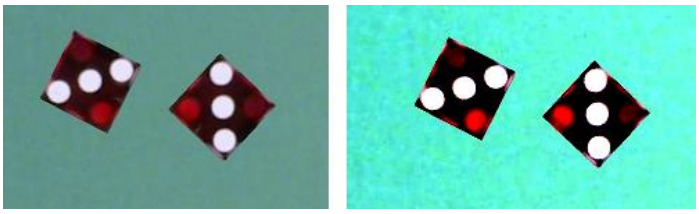


Figura 48 – Imagem original $f(x)$ e imagem resultante $g(x)$ (16)

4.2.2.2 Conversão para espaço de cor cinza

Uma vez aplicado o ajuste de brilho e contraste na imagem, o seu espaço de cor é alterado para o espaço de cor adequado, de forma a ser possível a utilização de algoritmos de *thresholding*. Uma vez que os algoritmos de *thresholding* trabalham com intensidade de *pixels*

em vez da comum representação RGB, o espaço de cores foi convertido para o espaço de cores *grayscale* (escala de cinza).

Esta representação faz com que a imagem não se encontre mais caraterizada pelas 3 componentes de cor mas sim pela intensidade de cada *pixel*, passando assim de 3 componentes para apenas uma, existindo assim uma perda de informação entre as transformações, ou seja, não é possível reverter a transformação de forma a obter a imagem original.

A imagem original encontra-se convertida com base na seguinte fórmula [ITU-BT.601-7, 2011],

$$I(i,j) = 0.299 \times R(i,j) + 0.587 \times G(i,j) + 0.114 \times B(i,j) \quad (17)$$

Em que,
$I(i,j)$ – *Pixel* (i,j) da imagem resultante da conversão de espaço de cor RGB → *grayscale*;
$R(i,j)$ – Componente vermelha do *pixel* (i,j) da imagem original;
$G(i,j)$ – Componente verde do *pixel* (i,j) da imagem origina;l
$B(i,j)$ – Componente azul do *pixel* (i,j) da imagem original.

Uma vez aplicado o método de conversão de espaços de cor os resultados obtidos são similares ao apresentado na Figura 49.

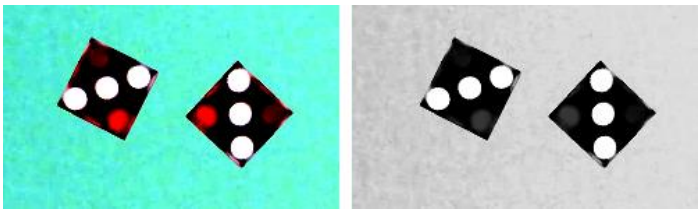


Figura 49 - Imagem original $RGB(x)$ e imagem resultante $I(x)$ (17)

4.2.2.3 *Threshold* da imagem

A imagem atual encontra-se agora numa representação apenas definida pela sua intensidade, desta forma os *pixels* referentes às pintas encontram-se com um valor de intensidade próximo de 255, sendo 255 o valor de um *pixel* branco.

De forma a segmentar os *pixels* referentes às pintas é utilizado um método de *thresholding* binário com base na seguinte equação [Sezgin and Sankur, 2003],

$$\mathrm{g}(i,j) = \begin{cases} 255 & , \quad f(i,j) > t \\ 0 & , \quad f(i,j) \leq t \end{cases} \quad (18)$$

Em que,
$g(i,j)$ – *Pixel* (i,j) da imagem resultante do algoritmo de *thresholding;*
$f(i,j)$ – *Pixel* (i,j) da imagem original;
t – Valor do threshold sendo que $0 \leq t \leq 255$.

Uma vez que os *pixels* referentes às pintas dos dados se encontram com um valor de intensidade elevado, em relação ao resto da cena, é possível desta forma utilizar um valor bastante elevado para o limiar t.

O valor de t foi iterado a partir de 255 (valor máximo) com decréscimo de 1 em 1 até ser encontrado o valor ideal apresentado na tabela [Tabela 3]

Tabela 3 – Valor utilizado para o limiar do *threshold*

Variável	Valor
t	240

A aplicação do algoritmo de *thresholding* uma vez utilizado numa amostra produz o resultado apresentado na Figura 50.

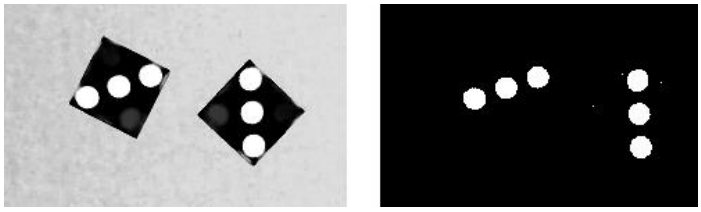


Figura 50 – Imagem original $f(x)$ e imagem resultante $g(x)$ (18)

4.2.2.4 Operação de erosão

Apesar dos pontos se encontrarem segmentados da cena, é necessário proceder a uma operação de erosão [Jankowski, 2006], esta operação é necessária devido à resolução na qual as amostras utilizadas se encontram, uma vez que os dados de valor de face 6 podem encontrar-se unidos devido à distância entre as suas pintas ser bastante reduzida (0.5mm em tamanho real). Tal união faria com que as pintas que se encontrassem conectadas fossem descartadas no processo de catalogação.

De forma a aplicar a operação de erosão é necessário definir o *kernel* utilizado na operação. Este *kernel* é caracterizado pelo seu tamanho e forma, sendo a forma utilizada a de um retângulo (dentro das 3 possíveis, retângulo, cruz, elipse) e o tamanho definido pela seguinte fórmula,

$$k_s = k \times 2 + 1 \qquad (19)$$

Em que,
k_s – Tamanho do *kernel;*
k – Índice de tamanho do *kernel* sendo que $k > 0$.

O tamanho do *kernel* é obrigatoriamente superior a 3 e sempre ímpar de forma a poder ser definido o ponto de ancoragem do mesmo como sendo o centro do *kernel*¸ sendo que isto seria impossível caso o *kernel* tivesse um tamanho de valor par.

De forma a encontrar o índice de tamanho, k, adequado do *kernel* para a operação de erosão, o valor k foi iterado até que as pintas dos dados de valor de face 6 se encontrassem efetivamente separadas.

Tabela 4 – Valor utilizado para o índice de tamanho (k) do *kernel*

Variável	Valor
k	2

O valor de k foi rapidamente encontrado uma vez que a resolução de imagens utilizadas é bastante reduzida. A aplicação da operação de erosão uma vez utilizada numa amostra produz o resultado apresentado na Figura 51.

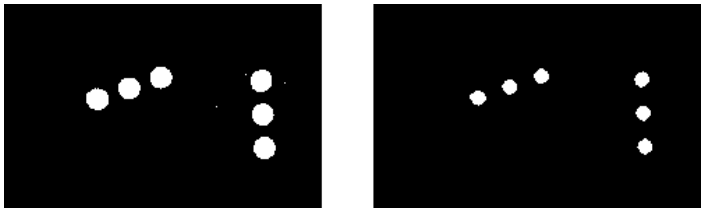


Figura 51 – Operação de erosão, imagem original (esquerda) e imagem resultado (direita)

Apesar do tamanho original das pintas se encontrar adulterado a informação perdida não tem relevo para a proposta apresentada, uma vez que apenas é necessária a informação do centro das pintas na cena.

4.2.3 Segmentação da imagem

O processo anteriormente referido no subcapítulo [4.2.2] apresentou o método de pré-processamento de imagem utilizado de forma a eliminar o ruído considerado desnecessário para o processo de análise dos dados. O segundo passo a tomar foca-se na segmentação e catalogação das pintas numa estrutura de dados, para que, em seguida, esta seja analisada com base nas suas características chave de forma a aferir o valor dos dados com base na análise dos padrões formados por essas mesmas características chave.

Desta forma o processo de segmentação da imagem é constituído pelas seguintes operações:

- Deteção de contornos e delimitador mínimo circular;
- Catalogação de pontos e definição de grupos;
- Classificação de dados com base em padrões de faces.

4.2.3.1 Deteção de contornos e delimitador mínimo circular

No estado atual a amostra utilizada apesar de se encontrar mais limpa devido à remoção de ruído continua a não ter nenhum significado que a diferencie da original para um computador, ou seja, independentemente do conteúdo de ambas as imagens o computador neste momento apenas as interpreta como uma matriz de *pixels* e nada mais, não sendo possível para o mesmo fazer qualquer tipo de juízo sobre ambas as imagens. Desta forma é necessário

proceder à segmentação da informação presente na amostra de forma a delimitar as regiões que correspondem às pintas dos dados, para tal, foi utilizado um algoritmo de deteção de contornos [Susuki and Abe, 1985] que permite identificar as regiões da imagem constituídas por conjuntos de *pixels* brancos (sendo que neste momento a amostra é uma imagem binária).

O resultado do algoritmo é constituído por uma estrutura de dados em que cada elemento corresponde a uma região presente na imagem, neste caso uma pinta, sendo que cada um desses elementos por sua vez é caracterizado por um conjunto de pontos que correspondem aos limites da região na qual se encontram inseridos [Figura 52].

Figura 52 – Resultado visual da utilização do algoritmo de deteção de contornos

As seguintes estruturas de dados são em seguida analisadas de forma a extrair o menor círculo delimitador existente para cada uma delas com base no algoritmo *Minimum Enclosing Circles* [Barquet et. al., 2005]. Este algoritmo retorna para cada uma das correspondentes áreas o centro [Figura 53] e o raio do círculo que as delimita.

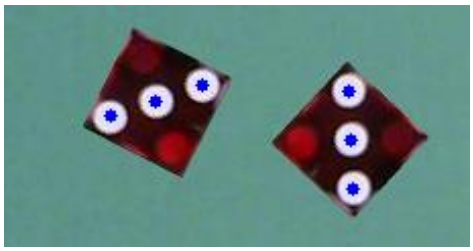


Figura 53 – Resultado dos centros dado pelo algoritmo *Minimum Enclosing Circles*

No caso do uso das amostras criadas pelo DSG para *input* do DSA, a integridade das pintas como áreas de maior intensidade encontra-se assegurada, sendo impossível segmentar áreas consideradas falsos positivos.

Em contrapartida o uso de *input* baseado na câmera encontra-se propicio a apresentar áreas que não representem uma pinta de um dado na realidade, desta forma o raio do círculo delimitador é tido em conta de maneira a assegurar que o raio da área a que se refere encontra-se dentro dos limites de tamanho para uma pinta de um dado. Uma vez eliminados os falsos positivos que não estejam compreendidos no espectro de tamanho aceitável para uma pinta do dado, uma segunda verificação é efetuada de forma a assegurar a circularidade da área, recorrendo para esse efeito ao uso do algoritmo *Hough Circle Transform* [Yuen et. al., 1990].

4.2.3.2 Catalogação de pontos e definição de grupos

O subcapítulo 4.2.3.1 explicou como extrair a informação relativa às pintas da imagem no entanto essa representação de informação apenas contém a disposição das pintas dos dados num plano bidimensional. De forma a aferir os valores dos dados presentes na cena é necessário relacionar os pontos com base nas suas distâncias relativas.

O processo de catalogação dos centros das pintas (pontos) é assim feito com base em distâncias significativas. Estas distâncias significativas são aferidas com base no modelo generalizado que possibilita criar qualquer tipo de face de um dado [Figura 54].

Figura 54 – Modelo generalizado de criação de faces de um dado

Com base no modelo generalizado é possível definir as restrições geométricas para cada uma das distâncias significativas a serem utilizadas [Figura 55].

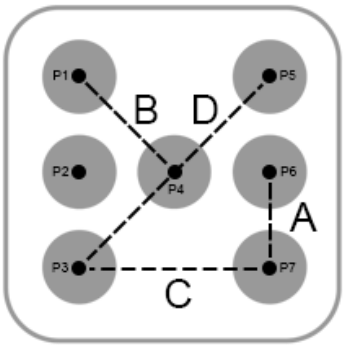


Figura 55 – Representação visual das distâncias significativas

Os pontos do modelo generalizado foram numerados com base na sua aparição quando vistas de cima para baixo e da esquerda para a direita, com base nessa indexação resulta o conjunto de pontos de P_1 a P_7.

As distâncias A, B, C e D constituem todas as distâncias significativas possíveis que possibilitam a criação de qualquer face de um dado com base no modelo generalizado, sendo estas definidas pelos seguintes vetores:

$$\begin{aligned} \|\vec{A}\| &= \|\overrightarrow{P_7P_6}\| \\ \|\vec{B}\| &= \|\overrightarrow{P_4P_1}\| \\ \|\vec{C}\| &= \|\overrightarrow{P_7P_3}\| \\ \|\vec{D}\| &= \|\overrightarrow{P_5P_3}\| \end{aligned} \qquad (20)$$

Uma vez definidas as distâncias significativas, os pontos são catalogados num grafo em que os seus vértices são constituídos pelas coordenadas dos pontos no plano bidimensional e os seus ramos representam as distâncias entre pontos do plano. A Figura 56 ilustra o caso modelo utilizado para explicação do método de catalogação de pontos com base nas distâncias.

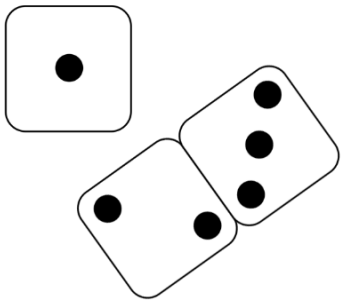


Figura 56 – Exemplo teórico para algoritmo de catalogação

Para cada ponto, N_a, na Figura 57 a distância euclidiana é calculada em relação aos restantes pontos, N_b, presentes na figura.

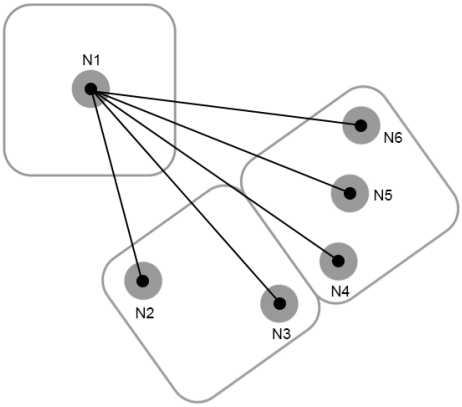


Figura 57 – Distância entre N_1 e todos os outros pontos

As distâncias calculadas são em seguida filtradas com base nas distâncias significativas. A Figura 58 ilustra o caso em que o ponto N_2 e o ponto N_3 partilham uma relação de distância significativa $\|\vec{D}\|$, sendo assim esta distância encontrar-se-á presente na estrutura de dados.

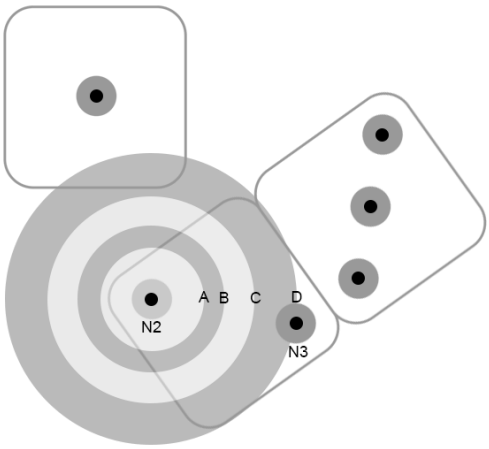


Figura 58 – Análise entre os pontos N_2 e N_3

Este processo é reproduzido para todos pontos de forma a aferir todas as distâncias candidatas. Quando é elaborada a filtragem de distâncias é necessário ter em conta a introdução de uma margem de erro uma vez é praticamente impossível aferir valores exatos aquando da medição das distâncias, esta margem de erro deve ser assim baseada na resolução do *input* de imagens utilizado para testes.

A seguinte Figura 59 ilustra as distâncias que são retidas no grafo para o exemplo teórico.

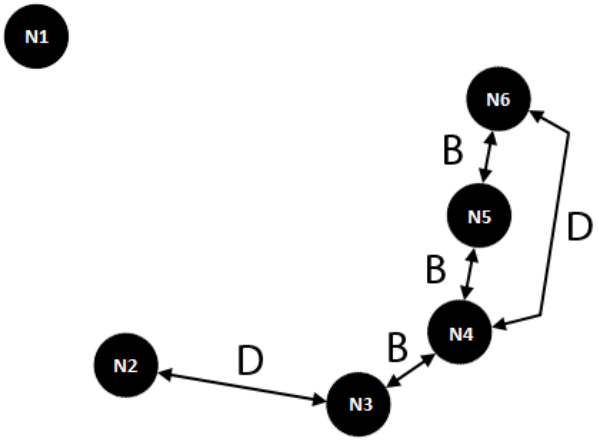


Figura 59 – Grafo resultado com as distâncias filtradas

Desta forma temos que as ligações de pontos em relação aos dados em que se encontram inseridos produz os resultados da tabela [Tabela 5].

Tabela 5 – Resultado da relação dado/ponto/distância.

		DADO 1	**DADO 2**		**DADO 3**		
		N_1	N_2	N_3	N_4	N_5	N_6
DADO 1	N_1	-	-	-	-	-	-
DADO 2	N_2	-	-	$\|\vec{D}\|$	-	-	-
	N_3	-	$\|\vec{D}\|$	-	$\|\vec{B}\|$	-	-
DADO 3	N_4	-	-	$\|\vec{B}\|$	-	$\|\vec{B}\|$	$\|\vec{D}\|$
	N_5	-	-	-	$\|\vec{B}\|$	-	$\|\vec{B}\|$
	N_6	-	-	-	$\|\vec{D}\|$	$\|\vec{B}\|$	-

Com base na informação apresentada na tabela [Tabela 5] é possível verificar que os pontos N_3 e N_4, apesar de pertencerem a dados diferentes, encontram-se relacionados por uma distância significativa $\|\vec{B}\|$.

A Figura 60 ilustra a situação que ocorre durante o processo de filtragem sobre o exemplo teórico no ponto N_3.

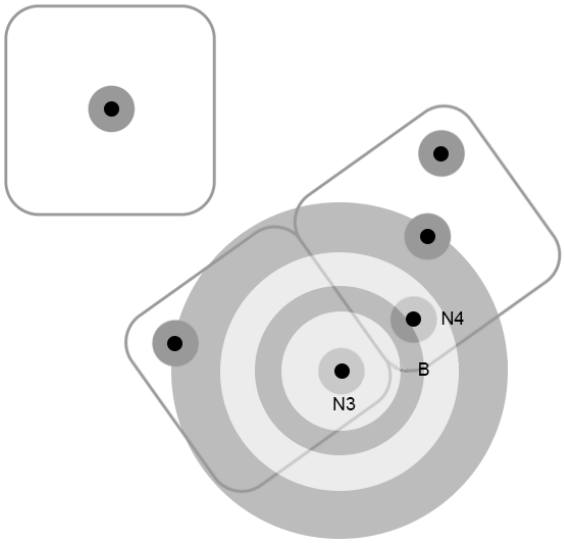


Figura 60 – Visualização da distância aferida entre N_3 e N_4

Devido à existência de pontos ligados por distâncias significativas que não fazem parte do mesmo dado é impossível aferir o valor dos dados no estado corrente, uma vez que um grupo de pontos ligados entre si não significa estritamente a presença de uma só face de um dado e assim tornando impossível a aferição dos dados baseada apenas na contagem dos pontos ligados.

De forma a ser possível catalogar corretamente os pontos é proposta uma abordagem com base na análise de conjuntos de pontos. Estes conjuntos são estipulados com base nas ligações formadas por distâncias significativas aferidas anteriormente, sendo que no exemplo teórico utilizado é possível verificar a existência de 2 grupos [Figura 61].

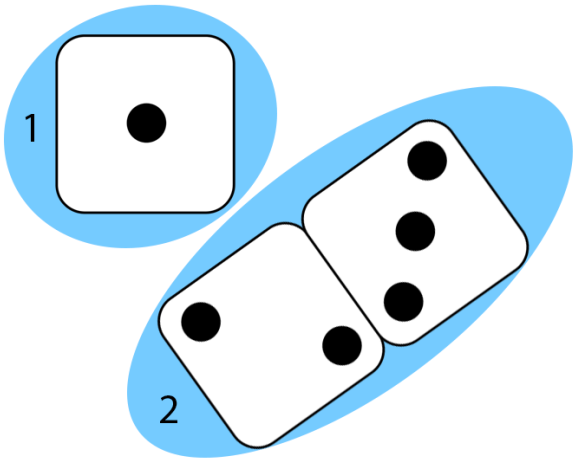


Figura 61 – Grupos de pontos formados com base no exemplo teórico

Os pontos encontram-se organizados nos grupos com base na sua aparição no plano bidimensional da esquerda para a direita e de baixo para cima. Este passo é vantajoso uma vez que permite com que o pontos diretamente ligados apareçam seguidos na estrutura de dados agilizando o processo de deteção.

A probabilidade de um grupo de pontos ser formando por mais de dois dados é bastante escassa, no entanto o algoritmo proposto encontra-se projetado para trabalhar com um número indefinido de dados a pesquisar.

Apesar de um grupo de pontos poder conter mais que um dado é possível agilizar o processo de reconhecimento atendendo aos casos especiais, isto é, a existência de grupos de pontos constituídos por 1 ou 2 pontos apenas. Estes casos são considerados especiais pois no caso da existência de um ponto isolado é óbvio que se trata de um dado de valor de face 1, no caso de um grupo de pontos com 2 pontos apenas é possível afirmar que este é um dado de valor de face 2 [Figura 62].

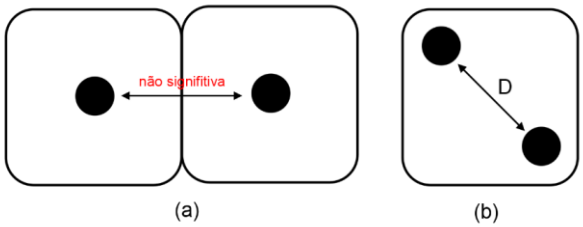


Figura 62 – Visualização do caso especial de grupos de 2 pontos

A afirmação é possível de ser feita pois um grupo de 2 pontos apenas permitiria duas soluções possíveis, dois dados de valor de face 1 [Figura 62 – (a)] ou um dado de valor de face 2 [Figura 62 – (b)], no entanto a primeira solução é inválida uma vez que apenas dois dados de valor de

face 1 nunca iriam pertencer ao mesmo grupo pois é os seus pontos nunca se iriam encontrar a uma distância significativa igual ou inferior a $\|\vec{D}\|$.

Com base nesta informação é possível então concluir que no exemplo teórico [Figura 61] o primeiro grupo de pontos é efetivamente um dado de valor de face 1, sendo como tal desnecessário fazer uma análise detalhada ao mesmo, análise esta que ocorrerá no entanto com o grupo de pontos 2.

4.2.3.3 Classificação de dados com base em padrões de faces

Uma vez que um grupo de pontos pode conter mais que um dado no seu conjunto é necessário efetuar uma separação dos mesmos de forma a aferir o valor dos dados, desta maneira, de seguida, é proposto um método baseado na análise de padrões de pontos formados por estes grupos que correspondem aos padrões existentes nas diferentes faces de um dado.

O método proposto consiste na análise de cada ponto de um determinado grupo, de forma a verificar se este se encontra num inserido num determinado padrão em colaboração com os outros pontos presentes no grupo. Caso seja descoberto um padrão os pontos que o formam são dados como utilizados, sendo o processo repetido para os restantes pontos do grupo até que todos os pontos se encontrem utilizados. O processo de descobrimento do padrão para cada uma das faces encontra-se ilustrado na Figura 63.

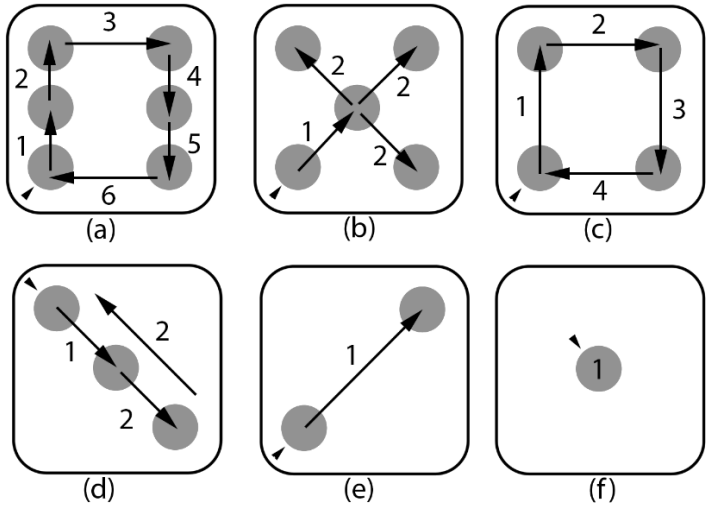


Figura 63 – Processo de descobrimento de padrão para cada face de um dado

O processo utilizado recorre a um método recursivo o qual procura para cada ponto um vizinho com a distância significativa correspondente ao padrão utilizado. Caso a distância significativa seja encontrada o algoritmo passa para o próximo ponto e repete o processo com base na distância a procurar, caso sejam encontrados pontos suficientes para formar um padrão este é dado como descoberto e todos os pontos são marcados como utilizados, impedindo-os assim de serem utilizados para formar um novo padrão. Uma vez descoberto um padrão é sinalizado. O processo utilizado para cada uma das faces, segundo o apresentado na Figura 63 é o seguinte:

(a) Face 1

Nível profundidade 1

Ponto isolado, automaticamente catalogado como dado de valor de face 1;

(b) Face 2

Nível profundidade 1

Procura de um vizinho com distância significativa $\|\vec{D}\|$;

(c) Face 3

Nível profundidade 1

Procura de um vizinho com distância significativa $\|\vec{B}\|$;

Nível profundidade 2

Procura de um vizinho com distância significativa $\|\vec{B}\|$;

Verifica a existência de uma distância $\|\vec{D}\|$ entre o ponto vizinho candidato e o ponto de origem do padrão;

(d) Face 4

Nível profundidade 1

Procura de um vizinho com distância significativa $\|\vec{C}\|$;

Nível profundidade 2

Procura de um vizinho com distância significativa $\|\vec{C}\|$;

Nível profundidade 3

Procura de um vizinho com distância significativa $\|\vec{C}\|$;

Nível profundidade 4

Verifica a existência de uma distância $\|\vec{C}\|$ entre o ponto vizinho candidato e o ponto de origem do padrão;

(e) Face 5

Nível profundidade 1

Procura de um vizinho com distância significativa $\|\vec{B}\|$;

Nível profundidade 2

Procura de três vizinhos com distâncias significativas $\|\vec{B}\|$;

(f) Face 6

Nível profundidade 1

Procura de um vizinho com distância significativa $\|\vec{A}\|$;

Nível profundidade 2

Procura de um vizinho com distância significativa $\|\vec{A}\|$;

Nível profundidade 3

Procura de um vizinho com distância significativa $\|\vec{C}\|$;

Nível profundidade 4

Procura de um vizinho com distância significativa $\|\vec{A}\|$;

Nível profundidade 5

Procura de um vizinho com distância significativa $\|\vec{A}\|$;

Nível profundidade 6

Verifica a existência de uma distância $\|\vec{C}\|$ entre o ponto vizinho candidato e o ponto de origem do padrão.

O algoritmo proposto começa por procurar para todos os pontos não utilizados o padrão correspondente à face de valor mais alto (face de valor 6), caso falhe em encontrar a correspondência para o padrão este é mudado para uma face de valor inferior, sendo a última hipótese um ponto de valor de face 1. Este processo permite que seja impossível serem encontrados falsos positivos, como por exemplo a existência de um dado de valor 4 dentro de um dado de valor 5 como é exemplificado na Figura 64.

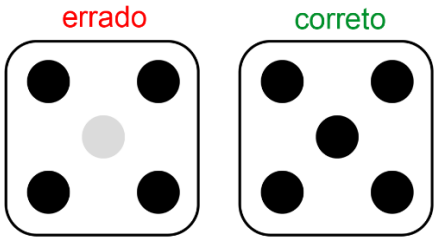


Figura 64 – Possível falso positivo teórico (*errado*) e a solução correta (*correto*)

O algoritmo proposto para descobrimento de padrões nos pontos utiliza sempre como ponto de origem um dos pontos próximos das extremidades da face do dado, caso o primeiro ponto a ser analisado fosse um ponto que não pertencesse a uma extremidade o algoritmo irá produzir falsos positivos. No entanto devido ao agrupamento ponderado dos pontos nos grupos mencionado anteriormente, da esquerda para a direita e de baixo para cima, é teoricamente impossível que o ponto de origem de um padrão a ser analisado seja um ponto que não se encontre próximo de uma extremidade da face do dado.

A Figura 65 ilustra um exemplo teórico onde se encontram sinalizados os pontos de origem para cada um dos padrões assim como os pontos capazes de gerar falsos positivos.

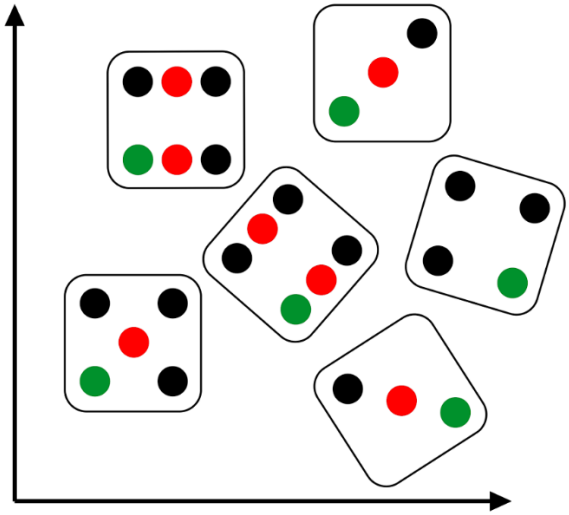


Figura 65 – Pontos de origem do padrão (*verdes*) e falsos positivos (*vermelhos*)

Esta situação poderia ocorrer em dados com face de valor 3, 5 e 6 pois são as únicas faces nas quais existem pontos que não podem ser considerados pontos de origem para um padrão uma vez que estes não são os pontos mais próximos a uma das extremidades do dado.

4.2.3.4 Demonstração do algoritmo com base no exemplo teórico

Uma vez estipulados todos os passos da proposta para a deteção dos dados é possível verificar o seu comportamento com base no modelo teórico apresentado anteriormente. A Figura 66 apresenta esse mesmo modelo já com as distâncias significativas evidenciadas, distâncias estas que foram recolhidas com base na filtragem das distâncias existentes entre todos os pontos com base nas distâncias permitidas pelo modelo genérico para criação de face de dados [Figura 55].

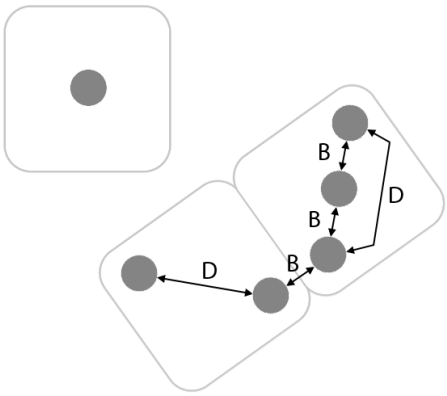


Figura 66 – Exemplo teórico, com distâncias significativas, para algoritmo de catalogação

De seguida é efetuada uma separação em grupos de pontos, sendo que estes grupos são formados por conjuntos de pontos ligados entre si. A Figura 67 demonstra os conjuntos de pontos gerados com base no modelo teórico apresentado, sendo possível verificar a criação de dois conjuntos sendo o primeiro conjunto criado com base no ponto isolado e o segundo conjunto criado pelos restantes pontos que se encontram ligados entre si.

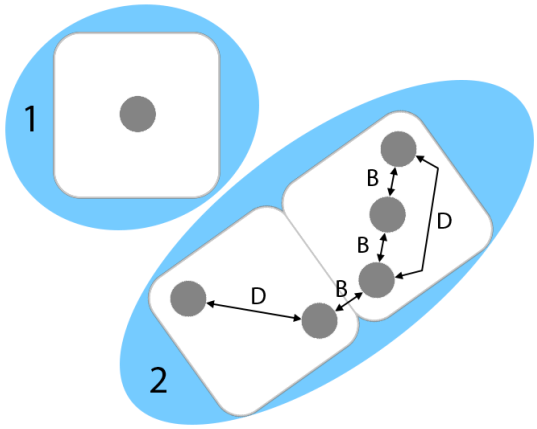


Figura 67 - Exemplo teórico, separação de pontos por grupos

Uma vez feita a separação dos pontos por grupos é possível aferir a informação referente a uma parte dos dados que constituem o exemplo teórico. Esta informação é possível de ser aferida com base na condição que estipula que um grupo constituído por 1 ponto é automaticamente um dado de valor de face 1 e um grupo constituído por 2 pontos é automaticamente um dado de valor de face 2. Deste modo o grupo de pontos 1 apresentado na Figura 67 é considerado um dado de valor de face 1.

Devido ao grupo de pontos 1 já se encontrar resolvido é prosseguida a análise do grupo de pontos 2, de forma a ser visualizado o comportamento do algoritmo a Figura 68 evidência a ordem dos pontos no grupo, esta ordem é baseada na aparição dos pontos na imagem da esquerda para a direita e de baixo para cima.

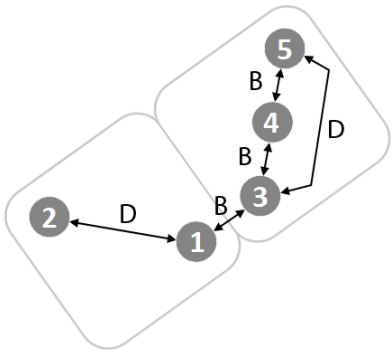


Figura 68 – Exemplo teórico, ordem dos pontos no grupo

De maneira a aferir os valores dos dados presentes no grupo de pontos é procedida à descoberta de padrões com base nas faces dos dados.

De forma a ser possível uma melhor visualização do processo de descobrimento é utilizada a seguinte nomenclatura exposta na Figura 69.

$N_x: P_a \rightarrow \left\|\overrightarrow{dist}\right\| \rightarrow P_b$

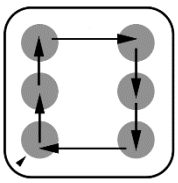
$N_x: P_a \rightarrow \left\|\overrightarrow{dist}\right\| \rightarrow P_b$

Figura 69 – Dado e nomenclatura genéricos

Em que,
N_x– Nível de profundidade do algoritmo;
P_a– Ponto a ser analisado;
P_b– Próximo ponto do algoritmo (caso seja encontrado);
$\left\|\overrightarrow{dist}\right\|$– Distância necessária para o próximo ponto.

Deste modo o processo de descoberta de padrões para o modelo teórico ocorre da seguinte forma, com base nos pontos numerados na Figura 68:

Para face de valor 6:



$N_1: 1 \rightarrow \left\|\vec{A}\right\| \rightarrow ?$

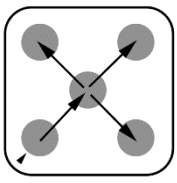
$N_1: 2 \rightarrow \left\|\vec{A}\right\| \rightarrow ?$

$N_1: 3 \rightarrow \left\|\vec{A}\right\| \rightarrow ?$

$N_1: 4 \rightarrow \left\|\vec{A}\right\| \rightarrow ?$

$N_1: 5 \rightarrow \left\|\vec{A}\right\| \rightarrow ?$

Para face de valor 5:



$N_1: 1 \rightarrow \left\|\vec{B}\right\| \rightarrow 3$
$N_2: 3 \rightarrow \left\|\vec{B}\right\| \rightarrow 4$
$N_2: 3 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$
$N_2: 3 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$

$N_1: 2 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$

$N_1: 3 \rightarrow \left\|\vec{B}\right\| \rightarrow 1$
$N_2: 1 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$
$N_2: 1 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$
$N_2: 1 \rightarrow \left\|\vec{B}\right\| \rightarrow ?$

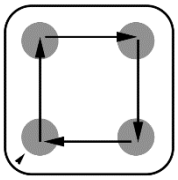
$N_1: 3 \rightarrow \left\|\vec{B}\right\| \rightarrow 4$

$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow 5$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow ?$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow ?$

$N_1: 4 \rightarrow \|\vec{B}\| \rightarrow 3$
$N_2: 3 \rightarrow \|\vec{B}\| \rightarrow 1$
$N_2: 3 \rightarrow \|\vec{B}\| \rightarrow ?$
$N_2: 3 \rightarrow \|\vec{B}\| \rightarrow ?$

$N_1: 5 \rightarrow \|\vec{B}\| \rightarrow 4$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow 3$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow ?$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow ?$

Para face de valor 4:



$N_1: 1 \rightarrow \|\vec{C}\| \rightarrow ?$

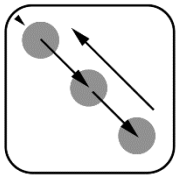
$N_1: 2 \rightarrow \|\vec{C}\| \rightarrow ?$

$N_1: 3 \rightarrow \|\vec{C}\| \rightarrow ?$

$N_1: 4 \rightarrow \|\vec{C}\| \rightarrow ?$

$N_1: 5 \rightarrow \|\vec{C}\| \rightarrow ?$

Para face de valor 3:



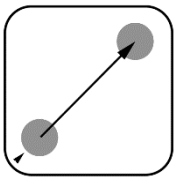
$N_1: 1 \rightarrow \|\vec{B}\| \rightarrow 3$
$N_2: 3 \rightarrow \|\vec{B}\| \rightarrow 4$
$N_3: 4 \rightarrow \|\vec{D}\| \rightarrow ?$

$N_1: 2 \rightarrow \|\vec{B}\| \rightarrow ?$

$N_1: 3 \rightarrow \|\vec{B}\| \rightarrow 4$
$N_2: 4 \rightarrow \|\vec{B}\| \rightarrow 5$
$N_3: 5 \rightarrow \|\vec{D}\| \rightarrow$ **3**

Foi encontrada correspondência para os pontos 3,4 e 5 sendo que estes marcados como usados para uma face de valor 3;

Para face de valor 2:

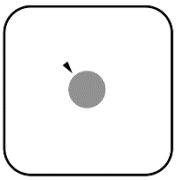


$N_1: 1 \rightarrow \left\|\vec{B}\right\| \rightarrow \mathbf{2}$

Pontos 3,4,5 já se encontram usados;

Foi encontrada uma correspondência para os pontos 1 e 2 sendo estes marcados como usados para uma face de valor 2;

Para face de valor 1:



$N_?: ? \rightarrow \left\|\vec{?}\right\| \rightarrow ?$

Todos os pontos do grupo já se encontram em uso;

4.3 Dice Calibration (DC)

A proposta apresentada para o DSA faz uso recorrente do termo distâncias significativas, distâncias estas utilizadas para a descoberta de padrões em conjuntos de pontos. Os valores destas distâncias variam consoante o modelo de dado utilizado os quais têm de ser estipulados antes da utilização do DSA.

Nas primeiras iterações da criação da proposta do DSA os valores utilizados foram inseridos manualmente através da sua medição com uma régua, no entanto verificou-se a necessidade da criação de uma solução para a calibração automática dos dados de modo a facilitar a introdução de novos modelos de dados. Devido a esta necessidade surgiu a proposta para um módulo de calibração automático denominado de *Dice Calibration* (DC).

4.3.1 Definição de razão entre tamanhos (*spatial multiplier*)

De maneira a aferir os valores das distâncias significativas é necessário aferir a razão entre as imagens capturadas pelo dispositivo de entrada, neste caso a câmera, e os tamanhos reais dos objetos. Numa primeira instância o cálculo desta relação foi efetuado com base na medição do valor real do lado de uma face do dado e do seu valor correspondente em *pixels* na imagem, obtendo assim a seguinte fórmula,

$$S_p = \frac{E_{r\_dado}}{E_{p\_dado}} \quad (21)$$

Em que,
S_p– Razão entre tamanhos (spatial multiplier);
E_{r_dado}– Tamanho real do lado da face do dado em *cm* (*edge real size*);
E_{p_dado}– Tamanho em *pixels* do lado da face do dado (*edge pixel size*).

No entanto de forma a homogeneizar o processo, deixando com que este se tornasse dependente do tamanho do dado em questão foi utilizado o mesmo princípio do algoritmo de calibração de uma câmera. Desta forma é possível calcular a razão entre tamanhos com base num modelo conhecido sem a necessidade da introdução manual de informação. Sendo assim o algoritmo proposto faz uso de uma imagem axadrezada [Figura 70] na qual são conhecidas as medidas dos quadrados que constituem a imagem.

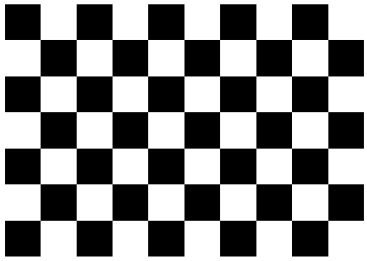


Figura 70 – Imagem axadrezada 10x7

O algoritmo de calibração da câmera deteta automaticamente os cantos interiores da imagem axadrezada uma vez que esta se encontre presente na cena, estes cantos são tipicamente devolvidos pelo algoritmo de forma a ser calculada a matriz de distorção da imagem. No entanto estes valores serão na realidade utilizados para criar a relação entre tamanhos, uma vez que a imagem utilizada já se encontra livre de distorção. A seguinte Figura 71 apresenta o resultado da utilização parcial do algoritmo de calibração de modo a aferir a posição dos cantos interiores da imagem axadrezada.

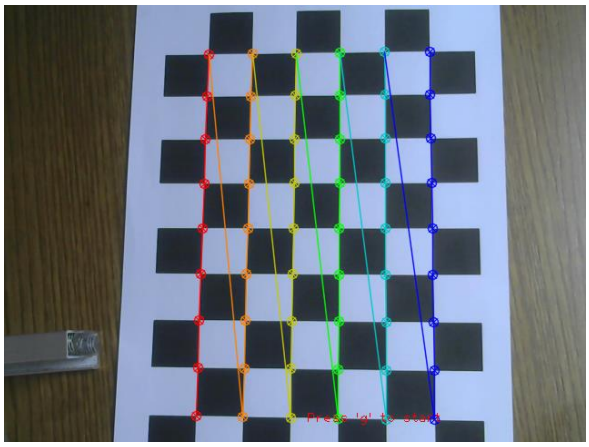


Figura 71 – Representação gráfica dos cantos interiores de uma imagem axadrezada

A informação referente a cada um dos pontos que representam os cantos é alocada num vetor ordenado, deste modo é possível calcular a razão entre os tamanhos com base na distância de dois pontos consecutivos, obtendo-se assim a seguinte fórmula,

$$S_p = \frac{E_{r\_quad}}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \quad (22)$$

Em que,
S_p– Razão entre tamanhos (*spatial multiplier*);
E_{r_quad}– Tamanho real do lado dos quadrados da imagem axadrezada em *cm*;
x_1– Coordenada x do primeiro ponto do vetor de cantos interiores da imagem;
x_2– Coordenada x do segundo ponto do vetor de cantos interiores da imagem;
y_1– Coordenada y do primeiro ponto do vetor de cantos interiores da imagem;
y_2– Coordenada y do segundo ponto do vetor de cantos interiores da imagem.

Uma vez calculada a razão entre tamanhos (S_p) é possível converter distâncias em *pixel* da imagem para distâncias em centímetros, usando a seguinte fórmula,

$$V_{cm} = V_{pixel} \times S_p \quad (23)$$

Em que,
V_{cm}– Valor da distância medida em centímetros;
V_{pixel}– Valor da distância medida em *pixels*;
S_p– Razão entre tamanhos (*spatial multiplier*).

Esta fórmula será posteriormente utilizada para todos os cálculos que envolvam distâncias para que todos os processos se encontrem a trabalhar na mesma escala.

4.3.2 Razão entre distâncias significativas e seleção de face utilizada

O método de cálculo automático de distâncias significativas baseia-se na segmentação de um dado de valor de face 4 em várias posições na imagem capturada, seguido da medição das suas distâncias significativas $\|\vec{C}\|$ que por sua vez servem para o cálculo do valor mediano de $\|\vec{C}\|$ que representará o valor escolhido para essa distância significativa.

O processo proposto faz uso apenas de uma face de um dado (neste caso face de valor 4) para o cálculo das distâncias significativas devido à possibilidade de expressar todas as distâncias significativas como múltiplos da distância significativa $\|\vec{A}\|$ como apresentado na Figura 72.

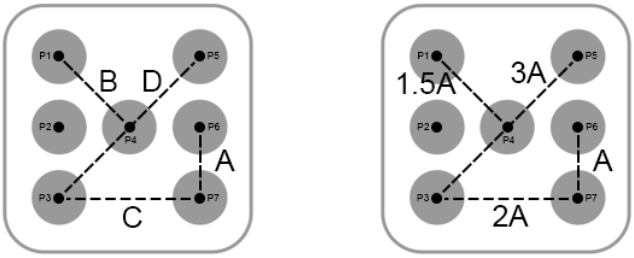


Figura 72 – Representação visual das distâncias significativas em múltiplos de $\|\vec{A}\|$

A escolha da face de valor 4 para cálculo das distâncias significativas deve-se ao facto de que esta face possuiu as melhores condições para ser efetuada a análise das distâncias em contrapartida com as restantes faces. A Tabela 6 apresenta as distâncias significativas presentes em cada face de um dado.

Tabela 6 – Distâncias significativas presentes nas faces dos dados

Face	Distâncias
1	0
2	$\|\vec{D}\|$
3	$2\|\vec{B}\|, \|\vec{D}\|$
4	$4\|\vec{C}\|, 2\|\vec{D}\|$
5	$4\|\vec{B}\|, 4\|\vec{C}\|$
6	$4\|\vec{A}\|, 5\|\vec{C}\|, 2\|\vec{D}\|, 4\|\vec{?}\|$

A face de valor 4 foi escolhida uma vez que apresenta um grande número de distâncias significativas idênticas, $4\|\vec{C}\|$, assim como possui um baixo número de outras distâncias significativas. Apesar da face de valor 6 possuir mais uma distância significativa $\|\vec{C}\|$ em relação a uma face de valor 4, esta contém no entanto um número bastante superior de outras distâncias as quais têm de ser eliminadas durante o processo de seleção das distâncias de tamanho $\|\vec{C}\|$. As distâncias $\|\vec{?}\|$ apresentadas na face de valor 6 representam distâncias existentes entre pontos que não são consideradas distâncias significativas.

4.3.3 Cálculo automático de distâncias significativas

O processo proposto para o cálculo automático de distâncias significativas utiliza um processo de pré-processamento de imagem idêntico ao de deteção de dados [4.2.2] e parte do processo de segmentação de imagem [4.2.3.1] de forma a ser possível aferir a localização relativa dos pontos no espaço.

A Figura 73 ilustra o resultado da aplicação dos algoritmos anteriormente mencionados.

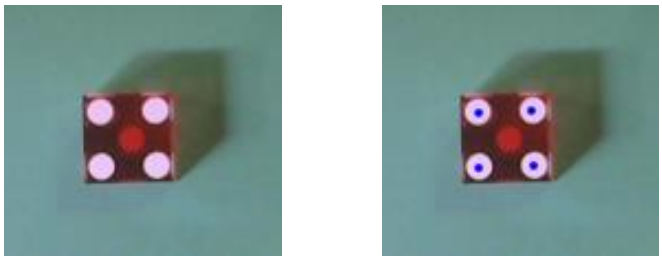


Figura 73 – Representação dos pontos do dado com base no algoritmo de deteção de dados

Devido à possível existência de distorção da imagem por parte da câmera utilizada é definido na proposta que as medições devem ser efetuadas em diferentes posições na área de jogo de forma a aferir um valor médio ideal para as distâncias significativas. Desta forma são efetuadas medições para 5 áreas distintas de jogo, sendo elas os cantos da área de jogo, assim como o centro da mesa. A Figura 74 demonstra a área de jogo a ser considerada, delimitada pelo retângulo azul, assim como uma subsecção onde é efetuada uma das análises, delimitada pelo retângulo verde.

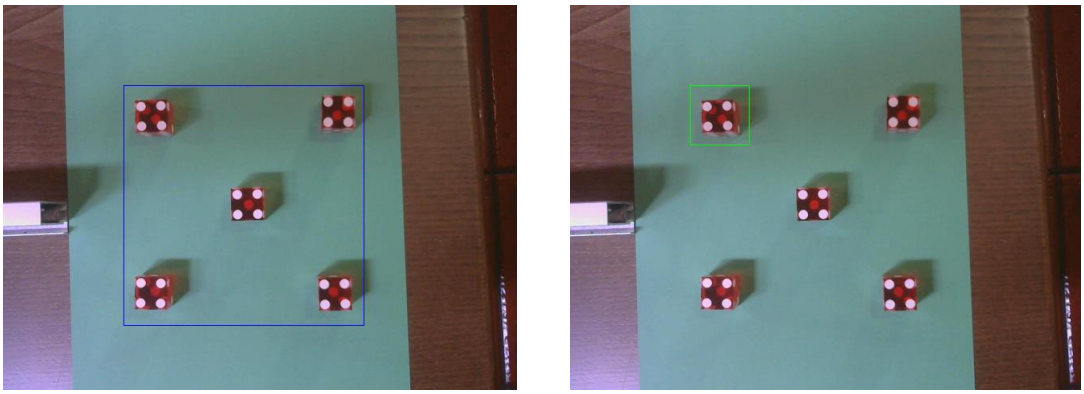


Figura 74 – Área de jogo e subseção de análise do algoritmo de calibração

A localização das subseções, assim como o número de análises por subsecção, pode ser alterada sendo no entanto aconselhado o uso das 5 subsecções apresentadas na proposta pois estas cobrem as localizações onde é possível existir um maior índice de distorção na imagem.

Para cada subsecção é capturado um número, N, de medições estipulado pelo utilizador, para cada medição são guardadas as 4 distâncias com valor mais baixo, sendo que cada medição é constituída por 6 distâncias ($4\|\vec{C}\|$ e $2\|\vec{D}\|$) e que $\|\vec{C}\| < \|\vec{D}\|$, tornando assim o processo de seleção de distâncias bastante simples.

Uma vez efetuadas todas as medições para cada uma das subseções os valores de $\|\vec{C}\|$ apurados são utilizados para o cálculo do valor ideal, $\|\vec{C}\|_{ideal}$, a ser utilizado segundo a seguinte fórmula,

$$\|\vec{C}\|_{ideal} = \frac{1}{4(S \times N)} \sum_{i=1}^{4(S \times N)} x_i \qquad (24)$$

Em que,
$\|\vec{C}\|_{ideal}$– Valor de $\|\vec{C}\|$ a ser utilizado para o dado modelo;
S– Número de secções utilizadas;
N– Número de medições por secção;
x_i– Distância índice i da estrutura de dados de medições;
4– Constante representativa da existência de 4 distâncias por medição.

Dado que o valor de $\|\vec{C}\|$ a ser utilizado já se encontra aferido é em seguida efetuado o cálculo dos restantes valores para as distâncias significativas com base relação entre distâncias enunciada no subcapítulo anterior [4.3.2], sendo obtidas as seguintes expressões,

$$\begin{aligned} \|\vec{C}\| &= \|\vec{C}\|_{ideal} \\ \|\vec{A}\| &= \frac{\|\vec{C}\|}{2} \\ \|\vec{B}\| &= 1.5 \times \|\vec{A}\| \\ \|\vec{D}\| &= 3 \times \|\vec{A}\| \end{aligned} \qquad (25)$$

Estes valores serão os valores posteriormente utilizados no DSA aquando da utilização do dado ao qual se referem.

4.3.4 Cálculo automático do tamanho do dado

O processo de cálculo do tamanho de um dado segue uma abordagem relativamente diferente no que toca ao pré-processamento de imagem a efetuar. Nas propostas até agora apresentadas o processo de pré-processamento utilizado transforma a imagem numa representação binária de forma a facilitar a sua manipulação, isto deve-se ao facto das pintas dos dados serem passiveis de ser alteradas de forma a ser possível tal representação sem perda de informação importante, sendo que o mesmo não se verifica para o restante corpo do dado.

No entanto o corpo do dado não é apenas composto por um grupo de *pixels* com uma intensidade uniforme definida, em contraste com as pintas, mas com *pixels* de diferentes intensidades e níveis de saturação como pode ser evidenciando na Figura 75.

Figura 75 – Complexidade de cor do corpo de um dado (vários "tipos" de vermelho)

Desta forma o método proposto para o pré-processamento do corpo do dado, passa pela utilização de um método de seleção de cor que consiga capturar toda a informação referente ao corpo do dado.

Numa primeira instância foi ponderada a utilização de um filtro de cor de forma a capturar os tons que constituem o corpo do dado (no caso da Figura 75 os tons de vermelhos), no entanto foi verificado que este método produziria resultados pouco exatos devido à complexidade de cor existente caso sejam utilizados dados translúcidos. Em contrapartida a utilização do fundo em que o dado se encontra inserido, sendo este opaco e uniforme, passa a ser um candidato mais apropriado ao uso do filtro de cor.

No entanto a utilização do filtro de cor sem qualquer tipo de tratamento da imagem é passível de reproduzir resultados também estes pouco exatos, pois existe a possibilidade das sombras criadas pelo dado assim como diferentes tipos de reflexos gerados pelas fontes de iluminação distorcerem os resultados esperados. No caso da Figura 75 é possível verificar que existem traços avermelhados sobre a mesa na zona da sombra gerada pelo dado, isto deve-se ao facto do dado utilizado ser translúcido produzindo assim este efeito.

Deste modo é proposta a utilização do algoritmo *k-means clustering* [Ng, 2014] como método utilizado para o pré-processamento de imagem, este algoritmo pretende agrupar os N *pixels* da imagem em K grupos com base no valor médio mais próximo de cada *pixel*, o resultado obtido é uma nova imagem com as cores simplificadas com base na quantidade de grupos escolhida. A Figura 76 demonstra o resultado do algoritmo *K-means* aplicado à imagem da figura anterior [Figura 75].

Figura 76 – Resultado do algoritmo *K-means* para partição em 6 grupos

A quantidade de grupos selecionada para o algoritmo *k-means clustering* deverá apontar sempre para um valor mínimo, sendo este valor selecionado mal se verifique uma clara distinção entre o plano de fundo e o corpo do dado. Uma vez aplicado o algoritmo é possível verificar que as cores que constituem o plano de fundo são bastante reduzidas (apenas duas cores) e todas elas dentro da mesma tonalidade (verde). Assim sendo é agora indicada a utilização do algoritmo de filtragem de cor para o fundo da imagem.

De forma a ser possível realizar a filtragem de cor é necessário converter a imagem do plano de cores *RGB* para o plano de cores *HSV*, este passo é necessário pois é impossível através de uma representação *RGB* selecionar uma gama de cores através da definição de dois limites para cada uma das componentes. Uma vez convertida a imagem é possível definir a seguinte equação do filtro de cor com base nos limites estipulados,

$$g(H,S,V) = H_0 \leq f(H) \leq H_1 \wedge S_0 \leq f(S) \leq S_1 \wedge V_0 \leq f(V) \leq V_1 \qquad (26)$$

Em que,
$g(H,S,V)$– Representação *HSV* da imagem resultante do filtro de cor;
$f(H,S,V)$– Representação *HSV* da imagem resultante do algoritmo *K-means*;
H_0 e H_1– Limite inferior e superior, respetivamente, do nível de tonalidade do filtro;
S_0 e S_1– Limite inferior e superior, respetivamente, do nível de saturação do filtro;
H_0 e H_1– Limite inferior e superior, respetivamente, do nível de luminosidade do filtro.

A equação (26) apresentada pretende filtrar as cores da imagem que se insiram na secção do cilindro *HSV* definida pelos limites impostos, a Figura 77 demonstra esse corte de secção realizado no cilindro com base em valores limite para H, S e V genéricos.

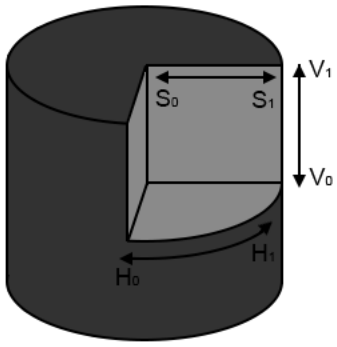


Figura 77 – Representação do corte de secção num cilindro com limites *HSV* genéricos

Desta forma para a imagem ilustrada na Figura 76 do resultado da aplicação do algoritmo *k-means clustering* foram usados os seguintes valores presentes na Tabela 7 para a utilização do algoritmo de filtragem de cor.

Tabela 7 – Valores utilizados para os limites de HSV

Limite	Valor
H_0	45
H_0	90
S_1	0
S_1	255
V_1	0
V_1	255

Sendo que os resultados produzidos pela aplicação do algoritmo encontram-se ilustrados na Figura 78, sendo a imagem (a) referente à máscara produzida pela aplicação do algoritmo (*pixels* brancos representam o fundo da imagem), imagem (b) referente à aplicação da máscara sobre a imagem original, e por fim imagem (c) referente à aplicação do inverso da máscara à imagem original.

Figura 78 – Resultados da aplicação do algoritmo de filtragem de cor

De forma a ser calculado o tamanho do dado será utilizada o inverso da máscara da imagem (a) da Figura 78, com base nessa imagem é aplicado o mesmo processo de deteção de contornos e delimitador mínimo circular enunciado no subcapítulo 4.2.3.1, este processo permite que seja encontrado o círculo que delimita o quadrado que representa o corpo do dado.

O resultado deste processo encontra-se ilustrado na Figura 79.

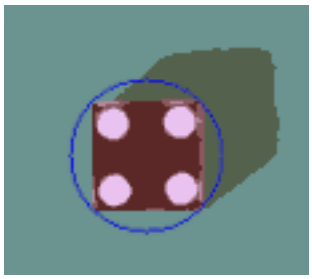


Figura 79 - Resultado visual da utilização do algoritmo de delimitador mínimo circular

Uma vez estipulado o círculo que delimita o corpo do dado é possível matematicamente encontrar o valor do seu lado pois o diâmetro do círculo é também na realidade a diagonal do quadrado que delimita o corpo do dado, como pode ser visualizado na Figura 80.

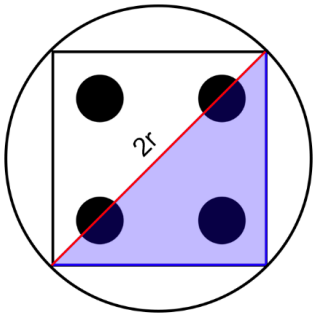


Figura 80 – Relação entre diâmetro (2r) do círculo e a diagonal do corpo do dado

Desta forma uma vez que a informação referente ao raio do círculo já se encontra aferida através do uso do algoritmo delimitador mínimo circular é possível encontrar o valor do lado do dado através da seguinte fórmula,

$$E = \sqrt{\frac{(2r)^2}{2}} \qquad (27)$$

Em que,
E– Valor do lado do dado em *pixels*;
r– Valor do raio, em *pixels,* aferido através do algoritmo delimitador mínimo circular.

Uma vez encontrado o valor do lado do círculo em conjugação com os valores das distâncias significativas é possível adicionar o novo modelo de dado ao sistema, para que este consiga efetuar a sua deteção.

4.4 Motion Detection (MD)

O módulo Motion Detection (DC) tem como intuito avaliar dois tipos distintos de áreas, consoante a zona a analisar no jogo *Banca Francesa*. A Figura 81 ilustra concetualmente a divisão entre as duas áreas de jogo.

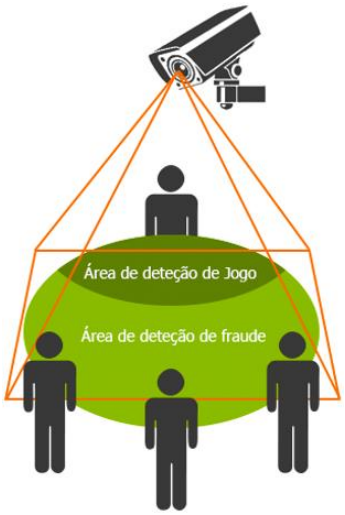


Figura 81 – Áreas de deteção existentes no jogo *Banca Francesa*

A primeira área a ser avaliada consiste na área de deteção dos dados, denominada "área de deteção de jogo", nesta área o algoritmo é responsável por verificar quando os dados lançados cessam o seu movimento.

A segunda área a ser avaliada é a área onde são efetuadas apostas, denominada "área de deteção de fraude", o algoritmo nesta área é responsável por detetar qualquer tipo de infração enquanto existir movimento na área de deteção de jogo.

4.4.1 Separação das áreas de jogo

O modelo conceptual apresentado faz uso de apenas uma câmera para recolha das imagens das duas zonas. Esta solução é proposta de forma a ser possível relacionar os tempos de cada imagem sem dificuldade. A utilização de mais dispositivos de recolha de imagem obrigaria à sincronização destes de forma a garantir a integridade temporal de cada análise efetuada.

Sendo assim é proposta a separação das zonas recorrendo ao uso de uma máscara binária de forma a criar duas imagens distintas contendo cada uma das zonas de deteção.

A Figura 82 apresenta uma representação genérica das duas áreas de jogo capturadas pela câmera.

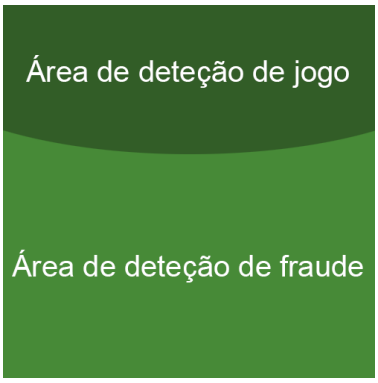


Figura 82 – Representação genérica das áreas constituintes do jogo

Sendo que a imagem binária utilizada como máscara é definida através da separação por uma linha a qual representa o limite entre as duas áreas. Esta máscara é utilizada para criar uma imagem com uma das áreas isoladas e em seguida é invertida de forma a recolher a outra área correspondente. A Figura 83 ilustra ambas as máscaras (normal e invertida) assim como os resultados obtidos quando utilizada sobre a imagem modelo genérica.

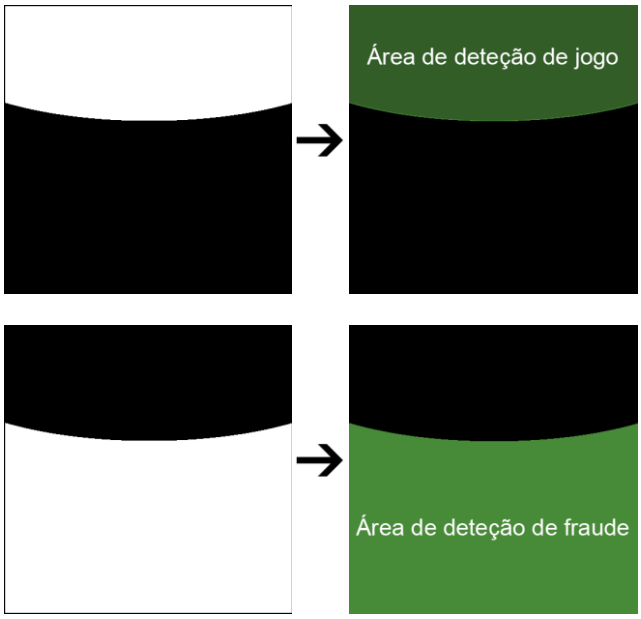


Figura 83 – Aplicação das mascaras para separação das zonas de jogo

Uma vez separadas as zonas de jogo em duas imagens distintas é prosseguida a sua análise individual de forma produzir resultados conforme os movimentos existentes em cada uma.

4.4.2 Área de deteção de jogo

A análise da área de deteção de jogo consiste em verificar a existência de dados na mesa de jogo e, caso estes existam, verificar se os mesmos cessaram o seu movimento.

A proposta para a resolução destas situações passa pela utilização de um algoritmo de *background subtraction*, MOG [KadewTraKuPong and Bowden, 2001], de forma a avaliar a situação atual da mesa de jogo em conjugação com a informação fornecida pelo DSA.

Os algoritmos de *background subtraction* normalmente possuem um tempo de aprendizagem bastante elevado de forma a dar tempo para que um determinado objeto seja considerado como parte constituinte da cena de fundo. No entanto o algoritmo proposto nesta dissertação diminui o tempo de aprendizagem do algoritmo de *background subtration* até que este aparente ser instantâneo ao olho humano. De forma a ser obtido o efeito é reduzido o tempo de aprendizagem do algoritmo para ser realizada uma atualização do fundo com a entrada de cada novo *frame* capturado. A Figura 84 apresenta 3 instâncias em espaços de tempo diferentes e a sua respetiva máscara para o algoritmo de *background subtraction*. Como pode ser visualizado a máscara apenas apresenta conteúdo na secção que se encontra em movimento na imagem.

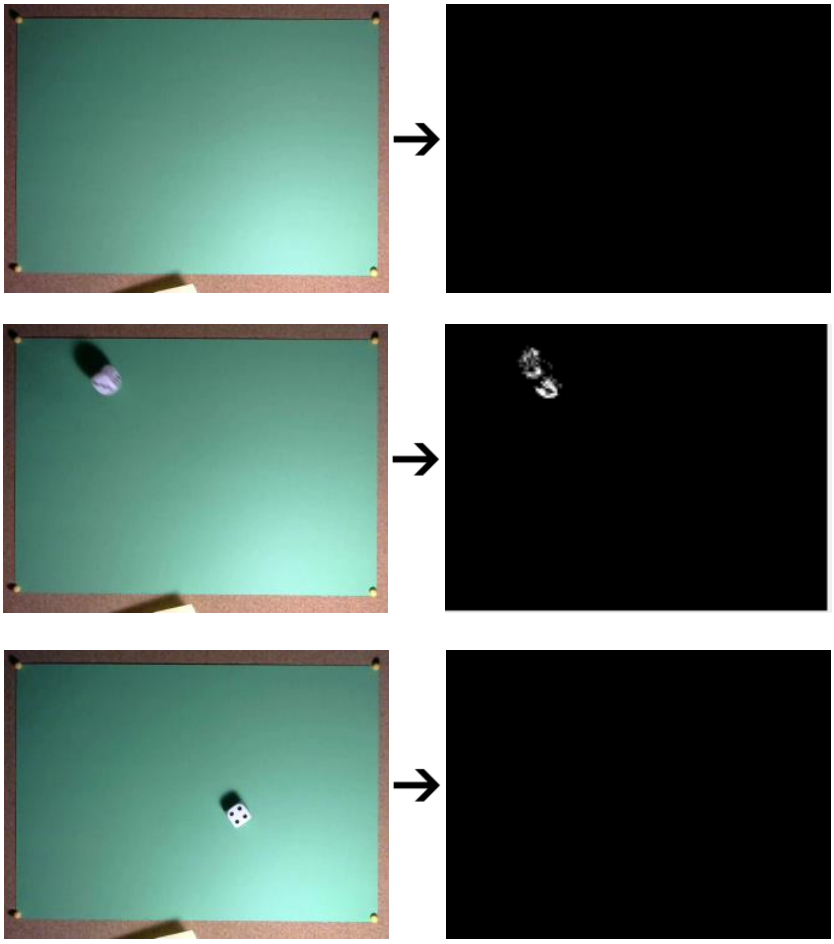


Figura 84 – Três instâncias da máscara do algoritmo de *background subtraction* MOG

Sendo assim, de forma a aferir se existe movimento num determinado *frame* é verificada a existência de uma determinada percentagem de *pixels* brancos na máscara do algoritmo de *background subtraction*.

Caso valor de *pixels* brancos seja superior ao limite definido então é assumido que existe movimento no *frame* analisado conforme o sugerido na seguinte fórmula,

$$M = \begin{cases} L_{max} > C & \rightarrow não\ existe \\ L_{max} \leq C & \rightarrow existe \end{cases} \quad (28)$$

Em que,
M– Resultado aferido para cada *frame* da situação de movimento no momento;
L_{max}– Valor de limite máximo da percentagem de *pixels* brancos para ser aferido movimento;
C– Percentagem de *pixels* brancos calculados.

Em seguida, caso não seja aferido movimento, é procurado pela existência de 3 dados na mesa através da utilização do DSA. Caso estes sejam encontrados o programa assume que não é possível serem realizadas apostas ou alterações de apostas a partir daquele determinado momento, sinalizando assim uma *flag* que será posteriormente utilizada em conjugação com o resultado do algoritmo de análise da área de deteção de fraude.

4.4.3 Área de deteção de fraude

A análise realizada para a área de deteção de fraude também faz uso de um algoritmo de *background subtraction* MOG2 [Zivkovic, 2004]. Neste caso, em vez do modelo de fundo ser atualizado instantaneamente, em relação com o que acontecia com o algoritmo utilizado na área de deteção de jogo, o algoritmo faz exatamente o oposto, retendo assim o modelo inicial durante todo o processo, até que seja sinalizado pelo programa que se deve realizar uma nova atualização do modelo de fundo. Desta forma é possível aferir qualquer tipo de alteração da cena em relação ao modelo de fundo. A Figura 85 ilustra uma situação onde uma das fichas brancas é transladada para uma posição diferente da inicial e o consequente resultado capturado pelo algoritmo de *background subtraction*.

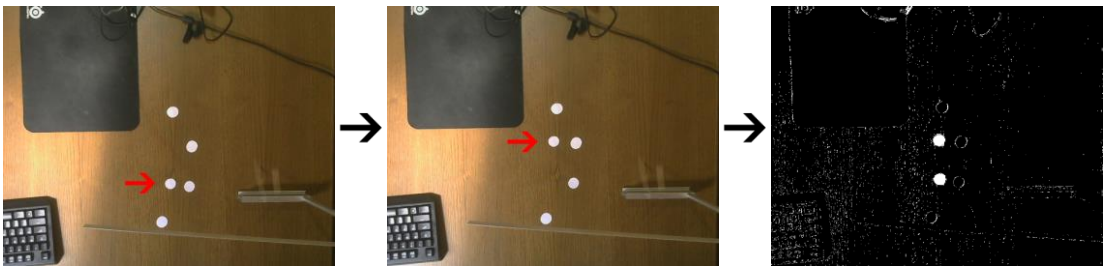


Figura 85 – Resultado do algoritmo *background subtraction* MOG2

Com base na Figura 85 é possível verificar a existência de bastante ruído na imagem resultante da aplicação do algoritmo de *background subtraction*, isto acontece devido à sensibilidade da implementação MOG2 em relação à implementação MOG usada no subcapítulo 4.4.2. No entanto a maior parte deste ruído é uma característica do algoritmo MOG2, que tenta capturar possíveis sombras na imagem, representadas por *pixels* de intensidade média, sendo que os *pixels* de intensidade máxima (255), representam as alterações efetivas na cena.

Sendo assim é aplicado um filtro de forma a serem removidas as sombras da máscara segundo a seguinte fórmula,

$$g(x,y) = \begin{cases} f(x,y) = 255 \rightarrow 255 \\ f(x,y) \neq 255 \rightarrow 0 \end{cases} \quad (29)$$

Em que,
$g(x,y)$ – Imagem resultante da operação de filtragem de sombras;
$f(x,y)$ – Máscara resultante da aplicação do algoritmo MOG2.

Após aplicada a operação de filtragem de sombras é aplicada uma operação morfológica de abertura [Haralick and Shapiro, 1991] de forma a efetuar a remoção do restante ruído.

De forma a aplicar a operação de abertura é necessário definir o *kernel* utilizado na operação. Este *kernel* é definido de forma similar à apresentada na equação (19) presente no subcapítulo 4.2.2.4 para a operação de erosão. De forma a encontrar o índice de tamanho k, adequado do *kernel* para a operação de abertura, o valor k foi iterado até que o ruído da imagem despareça [Tabela 8].

Tabela 8 – Valor utilizado para o índice de tamanho (k) do *kernel*

Variável	Valor
k	2

O valor de k foi rapidamente encontrado uma vez que a resolução de imagens utilizadas é bastante reduzida. A aplicação da operação de abertura uma vez utilizada produz o resultado apresentado na Figura 86.

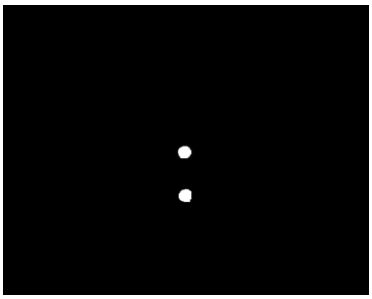


Figura 86 – Resultado da operação de abertura

A imagem da Figura 86 contém uma máscara para a localização da posição anterior da ficha branca e a localização da posição atual, neste caso as fichas já se encontravam na mesa quando o algoritmo foi despoletado, sendo este uma das 3 situações possíveis de serem aferidas:

Adição – uma ficha nova é inserida na mesa, não existindo localização anterior representada na máscara;

Remoção – uma ficha existente é removida da mesa, existindo apenas localização anterior representada na máscara;

Alteração – Uma ficha altera a sua posição na mesa, existindo um par localização anterior e localização atual para a ficha representada na máscara (exemplo que ocorre na Figura 86).

As localizações anteriores são caracterizadas pelas regiões da máscara sinalizadas onde, na imagem atual, apenas se encontra o fundo da mesa representado. A Figura 87 apresenta o recorte das localizações anterior e atual efetuado a partir da máscara resultante do algoritmo aplicado.

Figura 87 – Recorte das localizações anterior e atual respetivamente na imagem final

As adições e remoções de fichas são simplesmente verificadas recorrendo à análise de pares das localizações anterior e atual entre a imagem inicial e a imagem final, caso exista incoerência nas quantidades é sinalizada a existência de um ato de fraude.

A verificação das alterações é efetuada segunda a análise de possíveis pares de localização anterior e atual entre a imagem inicial e a imagem final, sendo para tal utilizada a implementação do algoritmo deteção de características SURF [Bay *et al.*, 2006]. Este algoritmo tem como objetivo verificar se uma determinada ficha na localização anterior consegue encontrar a sua correspondência na localização atual.

Caso a correspondência não seja encontrada é sinalizada a ocorrência de um ato fraudulento. Caso a correspondência seja encontrada, mas a localização das fichas seja diferente da zona de aposta inicial, por exemplo a ficha foi alterada de localização entre aposta GRANDE para aposta PEQUENA no jogo *Banca Francesa*, então é sinalizado também a ocorrência de um ato fraudulento.

Por fim a decisão final é obtida cruzando os resultados obtidos pelo algoritmo referente à área de deteção de jogo e pelo algoritmo referente à área de deteção de fraude, segundo o fluxograma apresentado na Figura 88.

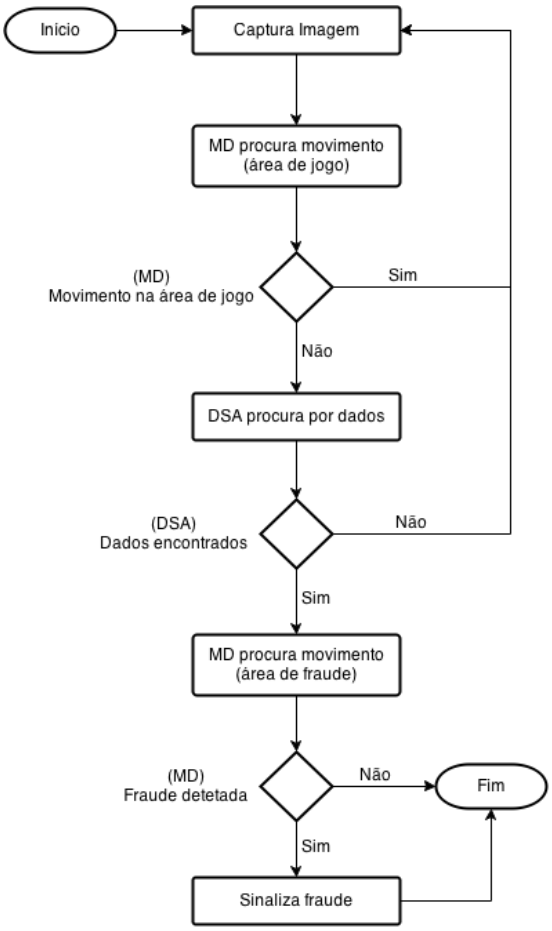


Figura 88 – Fluxograma de deteção de fraude

5 Testes e análise de resultados

Este capítulo referência os testes realizados de forma a validar a proposta apresentada no capítulo anterior, uma análise crítica dos resultados obtidos, assim como as configurações utilizadas para esse propósito.

Este capítulo engloba também uma menção aos testes realizados fora do âmbito da proposta apresentada no capítulo anterior. Estes testes serviram para averiguar o comportamento do algoritmo sobre determinadas condições, assim como a sua viabilidade em diferentes dispositivos.

5.1 Configuração do ambiente e especificação do material utilizado

O material utilizado para o desenvolvimento dos testes e soluções, no qual se baseia esta dissertação, foram alterando ao longo do tempo da realização da proposta. Desta forma neste subcapítulo encontra-se apresentado de modo sucinto as configurações utilizadas para cada um dos módulos de maneira a que seja possível replicar o ambiente utilizado para os testes realizados.

A câmera utilizada tanto para recolha de imagens modelo como para dispositivo de entrada para os módulos que constituem a proposta foi uma *webcam Logitech HD Webcam C270*.

O computador utilizado para o desenvolvimento das soluções foi um Toshiba Satellite L500-13W, CPU Intel Pentium Dual-Core CPU T4200 2.0GHz, 4GB RAM DDR2 (800Mhz) e GPU ATI Mobility Radeon HD 4570 512MB.

O sistema operativo selecionado para a o desenvolvimento dos testes foi o *Ubuntu 14.04.1 LTS*.

O IDE utilizado para o desenvolvimento dos programas de teste foi o *Eclipse Standard/SDK Version: Kepler Service Realease 2,* com os *plugins Eclipse CDT* e *Eclipse EGit* de relevo.

O módulo *Dice Sample Generator* (DSG) foi elaborado utilizando a linguagem *Java,* sem adição de qualquer tipo de biblioteca, através da implementação *open source* da *Java Platform Standard Edition*, o *OpenJDK 7*. Esta linguagem foi a optada pois operações básicas de edição de imagem e operações de manipulação de ficheiros DOM encontram-se disponíveis sem qualquer necessidade de ligação a bibliotecas externas assim como o grau de dificuldade bastante reduzido na sua implementação.

Os módulos *Dice Sample Analyzer* (DSA), *Dice Calibration* (DC) e *Motion Detection* (MD) foram desenvolvidos utilizando a linguagem *C++*, compilada através do compilador *gcc 4.8.2*, em conjugação, numa primeira instância, com a biblioteca *OpenCV 2.4.8* para arquiteturas *x64* baseadas nos sistemas operativos *Linux*. Apesar da portabilidade bastante versátil disponibilizada pela tecnologia *Java*, a linguagem *C++* foi a escolhida para o desenvolvimento dos módulos mencionados, pois é a linguagem na qual a biblioteca *OpenCV* se encontra primariamente desenvolvida e consequentemente a linguagem onde a maior parte da informação se encontra apresentada. Sendo também importante mencionar que os *bindings* para a linguagem *Java* encontrava-se incompletos até à data, a título de exemplo temos a falta de *interface* para uma implementação do algoritmo *Mixtures of Gaussians* disponível no *OpenCV* e utilizada num dos módulos.

As imagens modelo utilizadas no módulo DSG foram capturadas com a câmera anteriormente mencionada, que se encontrava a aproximadamente 42cm do centro da "mesa de jogo" numa vista de topo (90 graus em relação à mesa). A "mesa de jogo" consistia numa folha de cartolina A4 (ISO 216) verde colocada sobre um quadro de cortiça fixada em cada canto [Figura 89].

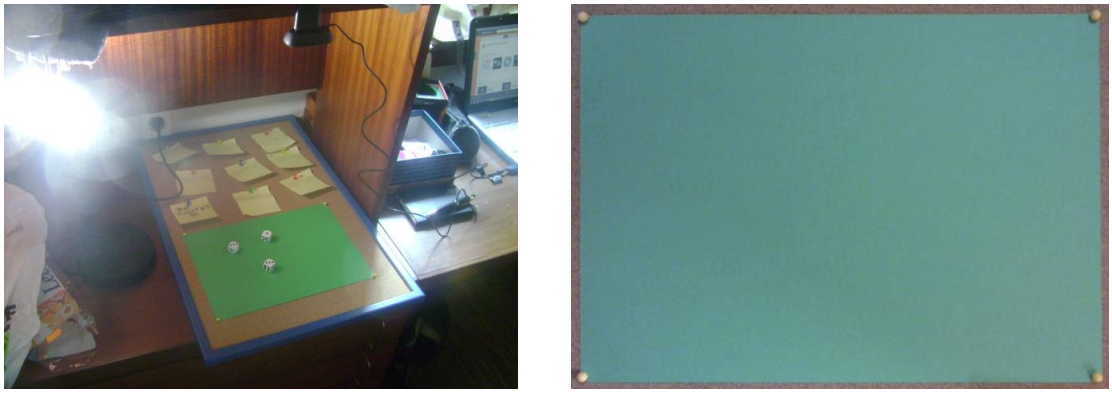


Figura 89 – Vista em 3ª pessoa e vista a partir da câmera da "mesa de jogo"

O dado utilizado para criar as imagens modelo do módulo DSG consistia num dado com cantos polidos, vermelho translúcido, com 1.9cm de lado e com pintas brancas de 0.5cm de diâmetro [Figura 90].

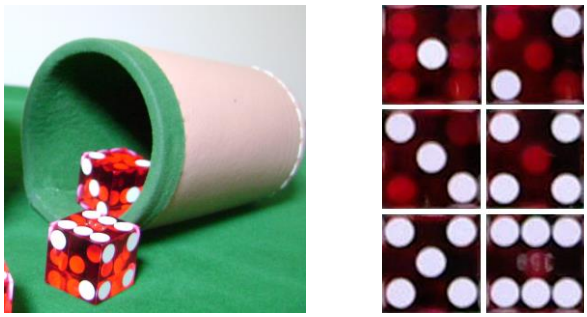


Figura 90 – Vista em perspetiva (esquerda) e resultado da recolha de imagens (direita)

Todas as imagens modelo utilizadas no DSG foram guardadas com uma resolução de câmera de 1692x1194 *pixels*.

De forma a ser possível trabalhar no projeto em diferentes localizações foi posteriormente criado um suporte metálico em conjugação com uma área de jogo móvel. O suporte utilizado replicou as mesmas métricas do suporte original, ou seja, a câmera continuou a encontrar-se a aproximadamente 42 cm da mesa numa vista de topo. A área de jogo manteve o mesmo tamanho de uma folha A4, no entanto tornou-se possível alterar o aspeto da mesma para diferentes cores. Este ponto foi incluído pois foram realizados testes para verificar o comportamento do algoritmo mediante a utilização de um fundo de cor similar ao do corpo do dado utilizado. A Figura 91 mostra a configuração utilizada em diferentes instâncias.

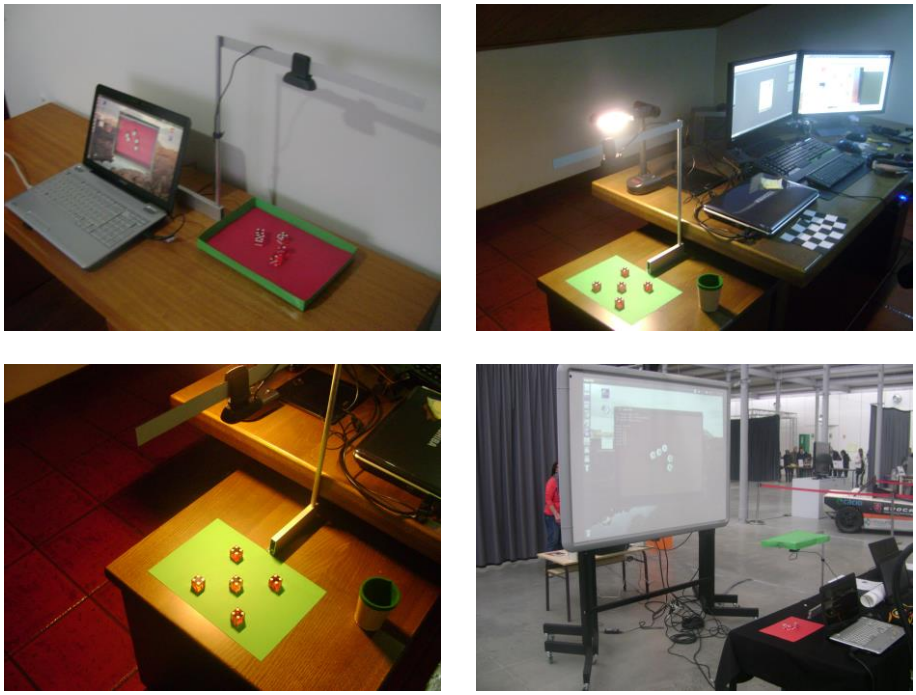


Figura 91 – Múltiplas instâncias da configuração utilizada para a realização de testes.

Numa fase inicial da dissertação foi também dado uso a um tipo diferente de dados [Figura 92], brancos opacos, com 1.5cm de lado cantos redondos com pintas pretas de 0.3cm de diâmetro. No entanto estes foram substituídos pelo conjunto de dados anteriormente mencionados pois são o modelo vulgarmente usado em ambientes de casino.

Figura 92 – Conjunto de dados brancos utilizados inicialmente

O material mencionado neste subcapítulo refere-se exclusivamente às configurações utilizadas para replicar os resultados obtidos com base na proposta elaborada. No entanto em certos casos foi utilizado outro tipo de material para a elaboração de diferentes testes, os quais se encontram referidos nos subcapítulos seguintes.

5.2 Dice Sample Generator (DSG) – Testes e Resultados

O módulo DSG tem como principal objetivo agilizar o processo de criação de amostras para utilização no módulo *Dice Sample Analzyer* (DSA), de forma a ser possível gerar um maior número de amostras num curto espaço de tempo, para assim verificar se existe algum caso específico onde o algoritmo proposto tem um comportamento inesperado.

Os testes efetuados ao DSG pretendem aferir se existe ou não uma vantagem clara em relação ao método convencional (manual) de recolha de amostras, deste modo o DSG foi executado com os seguintes parâmetros:

```
> time java –jar DSG.jar –min 5 –max 5 –n 10
```

Em que,
`min` - Limite mínimo de dados presentes numa amostra;
`max` - Limite máximo de dados presentes numa amostra;
`n` - Número de amostras a serem geradas.

Sendo que as amostras geradas são similares à apresentada na Figura 93.

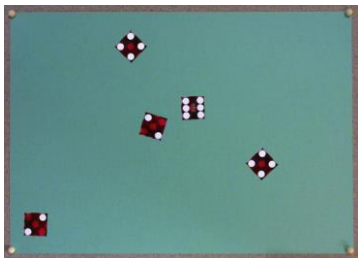


Figura 93 – Amostra gerada pelo DSG (5 dados)

Os resultados do tempo são baseados no *output* do comando `time`, sendo o resultado final de cada uma das execuções dada pelo somatório dos resultados de `user` e `sys`. A aplicação foi executada um total de 11 vezes sendo que para cada uma das instâncias foram criadas mais 10 amostras do que na instância anterior, começando com o processo para a criação de 0 amostras.

A Figura 94 apresenta o resultado obtido para cada uma das iterações.

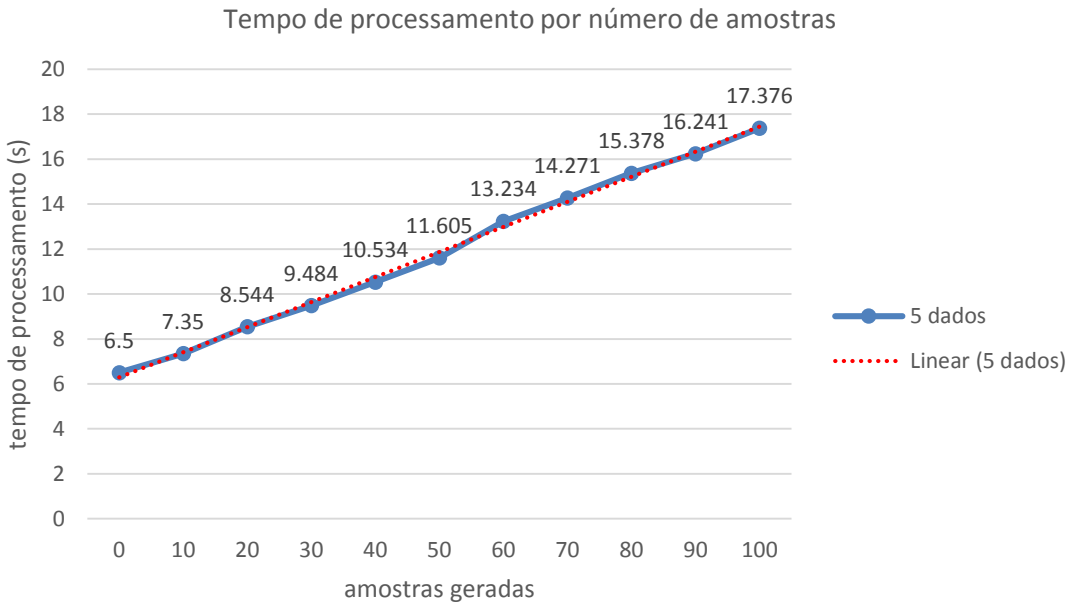


Figura 94 – Tempo de processamento por número de amostras

Com base nos resultados obtidos é possível constatar que a aplicação demora cerca de 6/7 segundos a inicializar, a carregar os recursos para a memória (as imagens necessárias), assim como a realizar o *parsing* do documento de XML que especifica o modelo de dado a utilizar aquando da criação de 0 amostras. Apesar do processo poder ser otimizado para não realizar estas ações quando são pedidas 0 amostras estas foram mantidas de forma a verificar o tempo que a aplicação realmente utiliza para criar efetivamente as amostras.

Com base no gráfico da Figura 94 é possível verificar que o tempo de processamento tende a progredir linearmente consoante o número de amostras gerado. Desta forma pode aferir-se que o tempo médio necessário para geração de 10 amostras de 5 dados é de aproximadamente 1.088 segundos, perfazendo assim uma média de 0,109 segundos para a criação de cada amostra.

Em contrapartida a recolha manual de apenas 10 amostras, com 5 dados cada, apontou um valor de 76,865 segundos. Assumindo que o comportamento da função de recolha de amostras manuais é semelhante à função da Figura 94, ou seja esta também é baseada numa progressão linear, a média para captura de cada uma das amostras é de aproximadamente 7.687 segundos.

O *dataset* criado para testes preliminares do DSA consistiu num conjunto de 2000 amostras geradas pelo DSG sendo que cada amostra continha um número aleatório de dados entre 1 e 20. O tempo total para a geração das 2000 amostras foi de 199.919 segundos.

Sendo assim é aferido que a utilização do DSG prova-se vantajosa para a criação de casos de teste preliminares pois é indiscutivelmente mais rápido que a recolha manual, e, também a existência da possibilidade de criar amostras com um número elevado de dados o qual se tornou impraticável na recolha manual, uma vez que não se encontravam disponíveis mais de 5 dados físicos, diminuindo assim a quantidade de casos possíveis numa determinada amostra.

5.3 Dice Sample Analyzer (DSA) – Testes e Resultados

O módulo DSA foi a componente com mais impacto no decorrer da dissertação, uma vez que todas as outras componentes desenvolvidas encontram-se relacionadas com o DSA. O módulo DSA é responsável pela captura e reconhecimento de um conjunto de dados que lhe sejam apresentado.

5.3.1 Abordagem preliminar efetuada e testes elaborados

A abordagem exposta neste subcapítulo refere-se a uma das primeiras abordagens apresentadas para a resolução do problema de deteção, apesar de esta abordagem não ter apresentado resultados satisfatórios serviu de ponto de partida para a solução apresentada na proposta do subcapítulo 4.2, uma vez que partes do processo de segmentação, assim como os problemas que impossibilitaram o seu sucesso foram tidos em conta nas iterações seguintes. Tendo isto este subcapítulo expõe muito resumidamente o processo utilizando em conjugação com os resultados obtidos no decorrer do mesmo.

Esta abordagem teve como objetivo tentar segmentar um dado com base na análise da proporção de *pixels* referentes às pintas do dado em relação à totalidade de *pixels* da face do dado. A abordagem utilizada pré-processou a imagem não com base nas suas pintas mas sim com base na face dos dados.

A Figura 95 demostra o resultado das várias etapas do pré-processamento utilizado por esta abordagem.

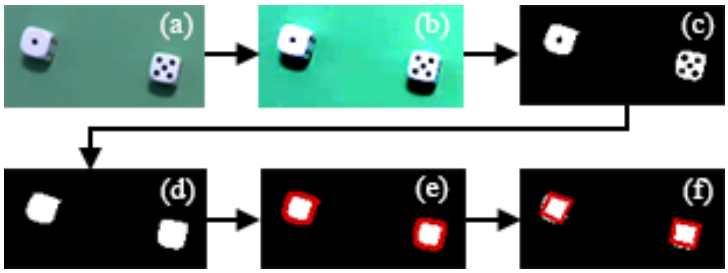


Figura 95 – Pré-processamento da análise de proporção de *pixels*

Sendo que cada uma imagens presentes na figura representam,

a) Imagem original capturada;
b) Ajuste de brilho e constraste;
c) Threshold de forma a evidênciar pixels com intensidade muito alta (perto de branco);
d) Operação de fecho para remover as pintas;
e) Deteção de contornos para delimitar a região das faces;
f) Redução polinomial para delimitar a região por apenas 4 pontos (quadrado).

Após aplicados os passos anteriormente mencionados torna-se possível recortar a imagem com base nos limites definidos pelos 4 pontos de cada região, assim sendo, para cada região foi criada uma nova imagem para cada recorte efetuado. A aplicação do recorte da imagem conteve também, numa primeira instância, uma mudança de perspetiva para que o dado não se encontre rodado [Figura 96].

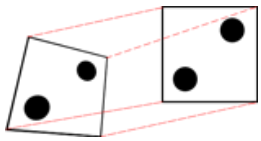


Figura 96 – Exemplo teórico de mudança de perspetiva baseada em 4 pontos

Sendo os resultados dos recortes e mudanças de perspetiva apresentados na Figura 97.

Figura 97 – Resultado do recorte e mudança de perspetiva

Uma vez efetuados os recortes e mudanças de perspetiva a cada uma das regiões é efetuada uma contagem de *pixels* de valor 0 (preto) e é calculada a sua percentagem em relação à totalidade de *pixels* da imagem recortada.

Os valores são em seguida comparados com os valores de percentagem padrão estipulados para cada face de um dado, presentes na Tabela 9.

Tabela 9 – Percentagem de *pixels* 0 (preto) padrão para cada face de um dado

Face	Percentagem (≈)
1	5.44%
2	13.50%
3	19.44%
4	25.33%
5	30.50%
6	35.33%

A abordagem proposta foi em seguida testada utilizando uma bateria de testes de 100 imagens similares à apresentada na Figura 98.

Figura 98 – Imagem exemplo da bateria de testes utilizada

Sendo que numa primeira instância foram detetadas corretamente 58 amostras das 100 utilizadas perfazendo assim uma precisão de 58%. Uma amostra é considerada como detetada corretamente quando todos os 3 dados são corretamente identificados.

Após uma análise dos resultados obtidos foi verificada a existência de dois problemas recorrentes. O primeiro problema evidenciado refere-se à aplicação da mudança de perspetiva após o recorte das regiões referentes à face do dado.

Ao serem verificados os casos específicos em que o problema ocorre foi possível aferir que aplicação da mudança de perspetiva tem um impacto significativo na proporção das pintas dos dados como pode ser visualizado na Figura 99.

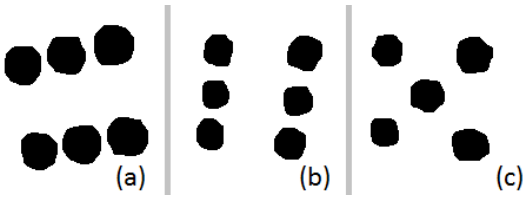


Figura 99 – Exemplo de imagens analisadas com resultados díspares

Neste exemplo os valores aferidos em relação aos valores reais não se encontram em conformidade como pode ser verificado na Tabela 10.

Tabela 10 – Resultado no qual é demonstrada disparidade de valores

Imagem (Figura 99)	Valor Real	Valor Aferido
(a)	6	6
(b)	6	5
(c)	5	5

A explicação encontrada para este problema deve-se ao facto da câmera utilizada possuir uma distorção significativa devido à lente utilizada quando os dados se encontram perto dos cantos da amostra, fazendo assim com que a alteração de perspetiva não efetue apenas uma rotação da imagem do plano bidimensional causando uma alteração das proporções das pintas.

O segundo problema evidenciado ocorre quando dois ou mais dados encontra-se fisicamente em contacto, neste caso o processo de segmentação de faces falha no reconhecimento dos dados conectados uma vez que a aplicação não consegue distinguir o conjunto dos dois dados como sendo uma objeto composto por duas faces. A Figura 100 ilustra um dos casos em que a aplicação falha em reconhecer os dados conectados.

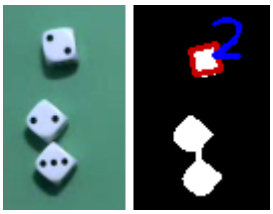


Figura 100 – Resultado em que os dados conectados não são identificados

De forma a solucionar o problema da alteração de proporções foi modificado o processo de aferição utilizado.

Nesta segunda instância em vez de serem utilizadas as proporções dos *pixels* como referência, apenas foi efetuado uma contagem das regiões de *pixels* 0 (pretos) [Figura 101].

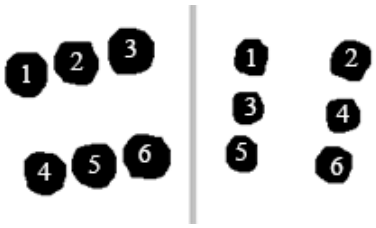


Figura 101 – Resultado da contagem de regiões de *pixels* 0 (pretos)

Apesar do aumento de complexidade do algoritmo não foi verificada uma alteração de velocidade no tempo de resposta percetível ao olho humano.

Utilizando a mesma bateria de testes de 100 amostras da iteração anterior foi possível detetar corretamente 83 amostras perfazendo assim uma precisão de 83%. Efetuando uma análise mais detalhada dos resultados foi verificado que a precisão melhorou devido à aferição correta dos dados onde o seu valor tinha sido previamente mal aferido devido à utilização do método de proporções de *pixels* 0 (pretos). No entanto o método utilizando continuou a falhar como esperado nos casos em que os dados se encontravam em contacto sendo assim optado pela procura de uma nova abordagem para solucionar os casos em questão.

5.3.2 Testes de implementação do DSA (amostras do DSG)

A proposta apresentada no subcapítulo 4.2 foi testada em duas formas diferentes, na primeira forma foram utilizadas as amostras criadas a partir do DSG e na segunda foi utilizado o *input* real de uma câmera.

Na primeira instância de forma a validar o algoritmo proposto, foram criadas 1000 amostras a partir do DSG, sendo que cada amostra continha um número de dados entre 1 em 20 escolhidos ao acaso aquando da criação da amostra.

As imagens geradas pelo DSG possuem uma resolução de 1672x1194 *pixels*, mas o DSA ajusta o tamanho da imagem para metade, 836x597 *pixels*. Esta operação é efetuada de forma a assemelhar o tamanho das imagens analisadas ao tamanho de imagens que são recolhidas por uma *webcam* convencional.

A Tabela 11 apresenta os tamanhos utilizados pelo DSA para os valores das distâncias significativas do dado utilizado.

Tabela 11 – Distâncias significativas utilizadas para o DSA

Distância	Valor (cm)
$\|\vec{A}\|$	0.55
$\|\vec{B}\|$	0.80
$\|\vec{C}\|$	1.10
$\|\vec{D}\|$	1.60

Neste caso a razão entre tamanhos (*spatial multiplier*) para converter unidades no DSA foi calculado com base no tamanho em *pixels* de um dado da amostra e da medida real do modelo de dado utilizado, sendo o valor da razão calculada a partir da seguinte fórmula,

$$S_p = \frac{E_{r\_dado}}{(\frac{E_{p_{dado}}}{2})} \quad (30)$$

Em que,
S_p– Razão entre tamanhos (spatial multiplier);
E_{r_dado}– Tamanho real do lado da face do dado em *cm;*
E_{p_dado}– Tamanho em *pixels* do lado da face do dado*;*
2– Constante que representa a redução de tamanho para metade da amostra.

Sendo que para as seguintes variáveis foram utilizados os valores presentes na Tabela 12.

Tabela 12 – Valores utilizados para o cálculo da razão entre tamanhos do DSA

Distância	Valor
E_{r_dado}	$1.9\ cm$
E_{p_dado}	$110\ pixels$

O valor estipulado como margem de erro a utilizar foi de 0.11 cm.

Das 1000 amostras utilizadas, 997 foram corretamente identificadas e 3 incorretamente identificadas, o que perfez uma taxa de precisão de 99.7%.

Após efetuada a análise das 3 amostras que não foram corretamente detetadas foi verificado que em todos os casos o algoritmo reconhece indevidamente um caso especial onde dois ou mais dados de valor de face 1 ou 2 se encontram com uma orientação e distância específicas.

A Figura 102 ilustra uma das situações onde existe esta ambiguidade no resultado.

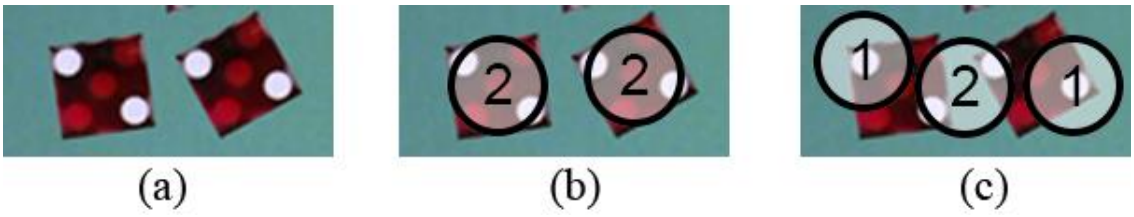


Figura 102 – Exceção ocorrida no algoritmo do DSA proposto para amostras do DSG

A exceção apresentada ocorre quando o algoritmo faz a análise do caso da imagem (a), nesta situação especial o resultado obtido deveria ser o apresentado na imagem (b) (dois dados de valor de face 2) no entanto o resultado que o algoritmo apresenta é o resultado da imagem (c) (três dados, um de valor de face 2 e dois de valor de face 1).

Esta situação ocorre devido a 3 fatores, o primeiro deve-se à forma de como os pontos são encontrados na cena, o qual se encontra ilustrado na Figura 103 (a). O segundo fator deve-se ao falso positivo gerado pelo filtro de distâncias significativas, apesar dos dois pontos não pertencerem à mesma face de um dado continuam a encontrar-se a uma distância significativa $\|\vec{D}\|$, como se encontra ilustrado na Figura 103 (b). O terceiro fator deve-se às possibilidades restantes que o DSA tem para gerar padrões uma vez que os dados presentes têm padrões extremamente simples a possibilidade de gerar um falso positivo é maior, neste caso é necessário apenas a existência de 2 pontos a uma distância $\|\vec{D}\|$.

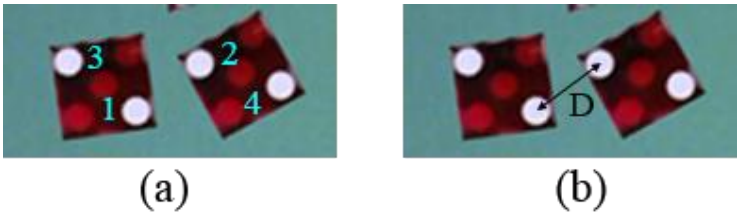


Figura 103 – Fatores responsáveis pela ambiguidade no resultado do DSA

A conjugação destes 3 fatores leva a que o DSA infira que o ponto 1 e o ponto 2 apresentados na imagem (a) da Figura 103 formam um dado de valor de face 2, bloqueando estes pontos para utilização em novos padrões, uma vez que ainda restam pontos a serem utilizados estes são automaticamente considerados pontos de dados de valor de face 1 uma vez que se torna impossível gerar um novo padrão com eles.

Uma das soluções propostas para a resolução do problema passa por refazer a procura de padrões para um dado de valor de face 2 a partir de outro ponto, no entanto esta solução é apenas possível caso seja conhecido o número de dados presentes na mesa. No caso do jogo Banca Francesa, como este apenas utiliza sempre 3 dados é possível aplicar esta solução.

No caso de não ser possível saber o número de dados presentes na mesa é necessário proceder a uma análise dos pontos candidatos a dados de valor de face 1.

A análise proposta passa por verificar a área envolvente do ponto na imagem como é ilustrado na Figura 104.

Figura 104 – Análise da envolvente dos candidatos a faces de valor 1

Caso esta área não seja uniforme é aferido que este não é de facto um candidato válido para um ponto de um dado de valor de face 1, sendo assim efetuado um *rollback* na pesquisa de padrões de forma a encontrar o padrão correto para o ponto correspondente.

5.3.3 Testes de implementação do DSA (casos reais)

Uma vez analisados os resultados do algoritmo proposto para o DSA, com base nas amostras geradas pelo DSG e verificado que estes foram bem-sucedidos, este foi adaptado para funcionar com casos reais. A configuração utilizada produziu resultados bastante similares às amostras produzidas pelo DSG sendo que todas as variáveis utilizadas para o cálculo de distâncias e margem de erro foram mantidas em conformidade com o apresentado no subcapítulo 5.3.2.

Para os testes em casos reais foram utilizados 300 amostras cada uma contendo 5 dados vermelhos translúcidos com as características referidas no subcapítulo 0. Destas 300 amostras 293 foram corretamente identificadas e 7 foram incorretamente identificadas, o que perfez uma taxa de precisão de 97.7%.

Uma vez testado o algoritmo com casos reais foi verificado um maior número de amostras que produziram um resultado incorreto. A presença de falsos positivos foi expectável uma vez que o algoritmo implementado ainda não contempla a solução proposta para os pontos de candidatos de valor de face 1 que geram falsos positivos como ilustrado na Figura 105.

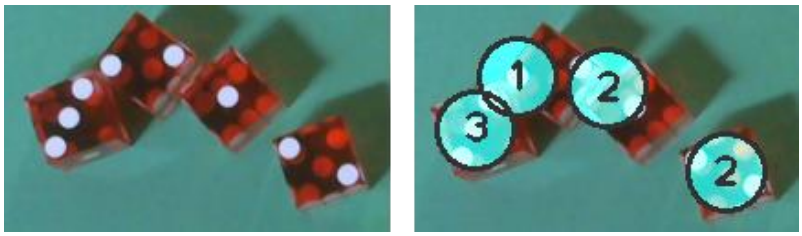


Figura 105 – Exceção ocorrida no algoritmo do DSA proposto para amostras reais

No entanto o que levou à presença de um maior número de falsos positivos deve-se ao facto das condições de iluminação terem um papel importante na forma de como a imagem é capturada.

Neste conjunto de testes foi verificado que perante certas condições o algoritmo segmenta incorretamente as imagens pois existem zonas de alta intensidade na imagem que não correspondem às pintas dos dados, como pode ser verificado na Figura 106.

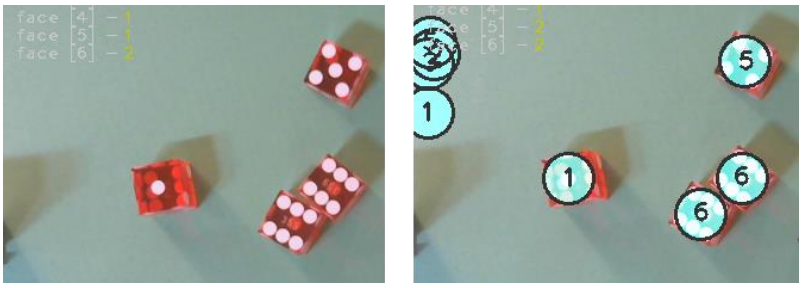


Figura 106 – Falsos positivos gerados devido às condições de iluminação

O aparecimento destes resultados deve-se ao facto de que a superfície onde os dados estão inseridos consegue refletir parcialmente a luz e também devido ao facto de que a iluminação ambiente onde foram recolhidas as amostras foi sendo alterado consoante o tempo.

A primeira utilização da proposta do DSA aplicada a casos reais foi elaborada durante o decorrer da *Mostra Tecnológica – Engenharia na Fábrica* na fábrica de Santo Thyrso. Durante este evento foi verificado o problema da inconsistência do algoritmo perante tipos de luminosidade inesperados. Esta situação foi constatada uma vez que a mesa onde se encontrava o equipamento estava diretamente abaixo de uma lâmpada de luz fluorescente, o que causava um reflexo de luz bastante elevado na superfície dos dados fazendo com que certos pontos dos dados aparecessem conectados por uma linha branca nas imagens devolvidas pela câmera. De forma a solucionar o problema na *Mostra Tecnológica* foi colocado sobre a câmera uma cobertura que fazia com que a maior parte da luz proveniente da lâmpada não passasse para a mesa [Figura 107].

Figura 107 – Solução apresentada na mostra realizada na Fábrica de Santo Thyrso

Este problema foi posteriormente analisado e não foi encontrada solução imediatamente após a sua descoberta. No entanto após a participação no programa intensivo de Erasmus ComVics 2014 após o contato e discussão com docentes e alunos nas áreas de computação visual foi aferida uma solução com base na utilização de um lente polarizada aplicada na câmera [Figura 108].

Figura 108 – Lente polarizada aplica na câmera utilizada na proposta do DSA

Com base na solução encontrada é então possível atenuar os efeitos de reflexão nas imagens capturadas pela câmera utilizada, sendo que a Figura 109 ilustra uma cena sem o uso (a) e com o uso (b) da lente polarizada em questão.

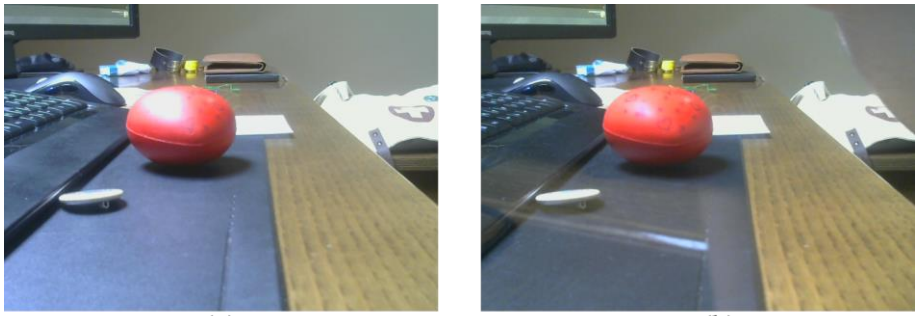


Figura 109 – Resultado da aplicação do filtro polarizado

Com base na Figura 109 é possível verificar que o objeto vermelho na segunda instância (b) encontra-se substancialmente mais nítido que na instância (a), sendo possível distinguir as marcas no mesmo, o que era impossível na imagem original sem filtro.

5.4 Dice Calibration (DC) – Testes e Resultados

Os testes realizados ao módulo Dice Calibration (DC) basearam-se numa prova de conceito simples utilizando os dados vermelhos translúcidos com as características referidas no subcapítulo 5.1. Sendo assim o primeiro passo efetuado foi o calculo da razão entre tamanhos com base na folha axadrezada apresentada na proposta do módulo DC do subcapítulo 4.3.

A Figura 110 apresenta a amostra utilizada para o cálculo das razões de tamanhos, sendo que esta é constituída por quadrados com 2.5cm de lado.

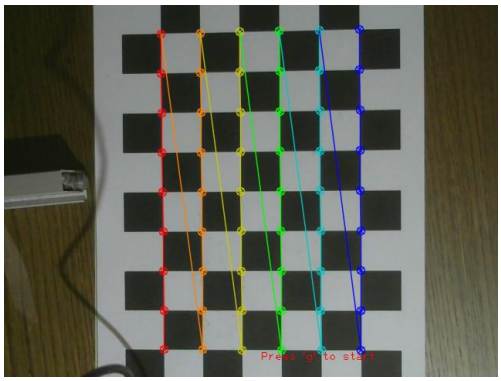


Figura 110 – Amostra usada para teste de cálculo da razão de tamanhos

O resultado obtido da aplicação do algoritmo foram os dois pontos apresentados na Tabela 13.

Tabela 13 – Pontos resultantes da aplicação do algoritmo de calibração

Ponto	Coordenada X	Coordenada Y
P_1	206.628	444.927
P_2	206.247	394.369

Uma vez conhecidos a localização dos pontos e o tamanho real do lado dos quadrados utilizados, o algoritmo aplica a fórmula (22) apresentada no subcapítulo 4.3.1 sendo o resultado obtido de 0.049 *cm/pixel*.

De seguida foi efetuado o cálculo do tamanho do dado modelo utilizado, para tal foi utilizada câmera de forma a recolher o conjuntos de amostras necessárias para a execução do algoritmo, sendo que, a Figura 111 apresenta um *frame* pertencente ao conjunto das amostras utilizadas.

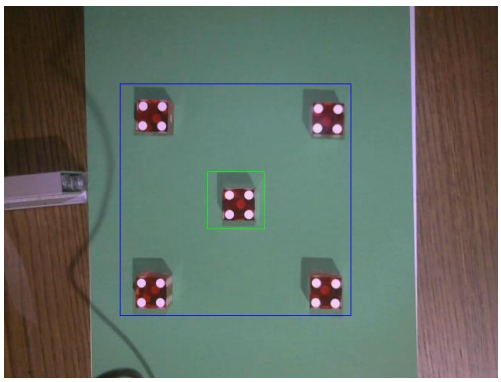


Figura 111 – *Frame* de uma amostra utilizada para o algoritmo de calibração

O processo de calibração capturou 10 *frames* da câmera os quais foram em seguida processados com base algoritmo apresentado no subcapítulo 4.3.4. A Tabela 14 apresenta o resultado aferido para cada uma das amostras recolhidas.

Tabela 14 – Amostras utilizadas para o teste do cálculo do tamanho do dado

Amostra (1-5)					
Tamanho (px)	40.56	41.34	41.52	41.44	40.87
Amostra (6-10)					
Tamanho (px)	41.12	41.06	41.36	40.61	40.54

Com base nos resultados obtidos o valor médio encontrado para o lado do dado em *pixels* foi de 41.042. Este valor é de seguida convertido para centímetros usando a fórmula (23) apresentada no subcapítulo 4.3.4, sendo o seu resultado 2.01 cm. Sendo assim é possível constatar que o valor calculado pelo algoritmo é bastante próximo do valor real de 1.9 cm calculado inicialmente com uma régua.

Uma vez calculado a razão entre tamanhos foi procedido ao cálculo das distâncias significativas. Neste processo foram recolhidos 20 *frames* de amostra para cada uma das 5 posições. A Figura 112 exemplifica as 5 posições onde foram recolhidas as amostras.

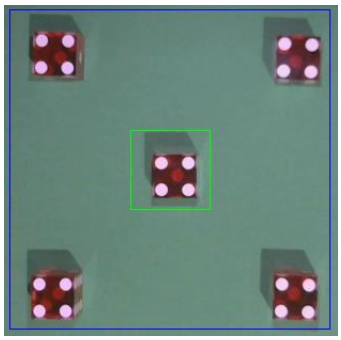


Figura 112 – Configuração para cálculo das distâncias significativas

Com base nas amostras recolhidas foi calculado o valor ideal da distância significativa $\|\vec{C}\|$ com base na média das distâncias recolhidas pelas amostras, como se encontra proposto pelo algoritmo presente no subcapítulo 4.3.3, uma vez conhecido o valor de $\|\vec{C}\|$ foram por corelação, calculados todos os restantes valores das distâncias significativas.

Os valores calculados através do algoritmo encontram-se presentes na Tabela 15.

Tabela 15 – Distâncias significativas aferidas

Distância	Valor (px)	Valor calculado (cm)	Valor real (cm)
$\|\|\vec{A}\|\|$	12.58	0.61	0.60
$\|\|\vec{B}\|\|$	18.87	0.92	0.90
$\|\|\vec{C}\|\|$	25.16	1.23	1.20
$\|\|\vec{D}\|\|$	37.74	1.85	1.80

Desta forma com base na informação da tabela é possível verificar que o algoritmo proposto consegue apurar com bastante precisão o valor das distâncias significativas para um determinado dado.

5.5 Motion Detection (MD) – Testes e Resultados

Devido à dificuldade em replicar um ambiente de casino para o jogo *Banca Francesa*, o módulo Motion Detection (MD) foi testado de forma bastante reduzida. Como foi apresentado no subcapítulo 4.4 o módulo MD encontra-se separado em duas vertentes "área de deteção de jogo" e "área de deteção de fraude". Na proposta estas são apresentadas como capturadas por apenas um dipositivo, no entanto devido à capacidade e configuração do material disponível, as duas áreas foram testadas em separado, sendo que, para efeitos de prova de conceito não deve ser visto como um problema, uma vez que as imagens na proposta encontram-se separadas através de uma máscara logo no primeiro passo, e são seguidamente tratadas em separado, sendo que, a decisão final é apenas aferida com o cruzamento de ambos os resultados finais.

5.5.1 Área de deteção de jogo

De forma a testar a deteção de movimento na área de jogo foram recolhidas 100 amostras de vídeo similares às apresentadas na Figura 113.

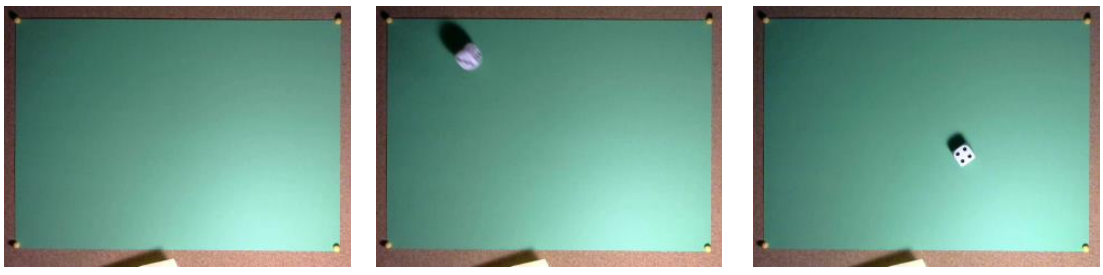


Figura 113 – Três *frames* distintos amostra de uma amostra de vídeo recolhida

O limite máximo estipulado para a percentagem de *pixels* brancos presentes na cena foi de 0%, isto levou a que o mínimo ruído existente indicasse movimento na cena. Apesar da implementação do algoritmo MOG [KadewTraKuPong and Bowden, 2001] utilizado

apresentasse uma sensibilidade razoavelmente baixa, foi necessário a aplicação de uma operação de abertura igual à apresentada no subcapítulo 4.4.3 de forma a remover o ruído indesejado. A Figura 114 apresenta um *frame* do algoritmo MOG e consequente resultado da aplicação da operação de abertura.

Figura 114 – Máscara do algoritmo MOG e resultado da operação de abertura

Todas as amostras testadas apresentaram o comportamento esperado, sendo detetado movimento no devido momento. A Figura 115 apresenta duas instâncias temporais (com movimento e sem movimento) do protótipo desenvolvido em ação para uma amostra e os seus respetivos resultados.

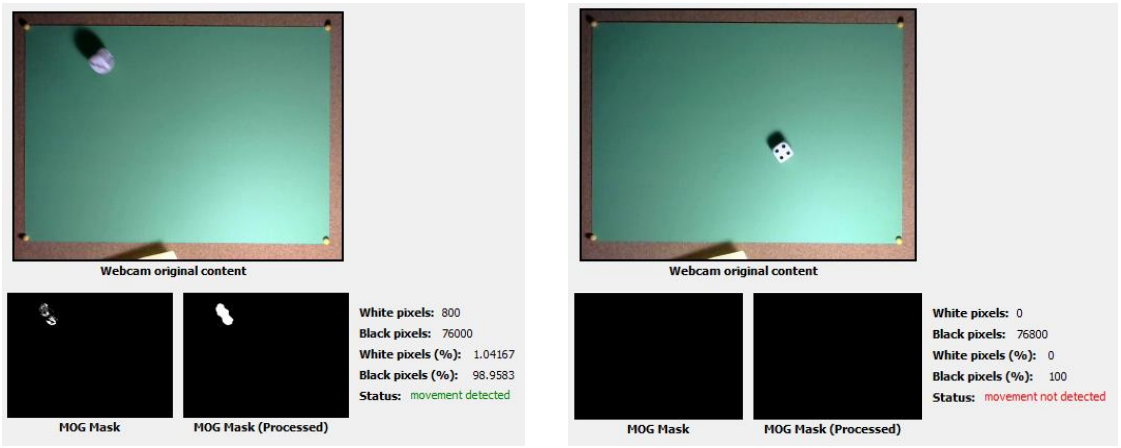


Figura 115 – Resultados do teste de deteção de movimento para uma amostra de vídeo.

Como pode ser visualizado numa das instâncias do protótipo presente na Figura 115 a aplicação apresenta o estado atual aferido com base na informação da etiqueta *Status*, sendo calculado a partir da informação das etiquetas *white pixels (%)* e o limite máximo estipulado anteriormente, 0%, como pode ser visto na primeira instância, a percentagem de *pixels* brancos é superior ao limite estipulado assumindo assim a existência de movimento.

5.5.2 Área de deteção de fraude

Os testes da área de jogo não sofrem um impacto significativo devido às dificuldades em reproduzir um ambiente de jogo de casino de *Banca Francesa*, em contrapartida os testes para a área de deteção de fraude necessitam de um ambiente o mais perto que possível do ambiente original em que se baseia a solução.

Desta forma ao contrário dos testes efetuados até agora para as propostas mencionadas, as quais apresentam um conjunto considerável de casos de teste efetuados, a parte referente à área de deteção de fraude apenas incidiu na análise de comportamento dos algoritmos para uma determinada configuração e uma análise crítica em relação aos resultados obtidos e a maneira de que estes se conseguem enquadrar, contado com possíveis problemáticas, no caso da sua utilização num ambiente original de um jogo de *Banca Francesa*.

A proposta apresentada para o algoritmo de deteção da área de fraude faz uso do método *background subtraction* MOG2 [Zivkovic, 2004], que difere do método MOG [KadewTraKuPong and Bowden, 2001] anteriormente mencionado devido à sua capacidade de deteção de sombras e também à baixa tolerância para deteção de diferenças entre a imagem atual e o modelo que utiliza, conseguindo assim produzir uma máscara mais precisa da zona que sofreu a alteração.

Devido a esta baixa tolerância tiveram de ser efetuados ajustes na forma como são capturadas as imagens, uma vez que existem discrepâncias entre *frames* recolhidos. Apesar destas discrepâncias não terem efeito sobre a implementação do algoritmo MOG (o qual é bastante mais tolerante), o mesmo não se verificou com o algoritmo MOG2, fazendo com que a máscara do algoritmo MOG2 assinalasse a imagem toda como alterada.

Estas discrepâncias entre *frames* foram verificadas como sendo geradas pelo ajuste automático que o dispositivo efetua para a correção de cor, neste caso quando um novo objeto é introduzido na cena a câmera efetua um balanço automático das cores os quais entram em conflito com o algoritmo de baixa tolerância MOG2. A Figura 116 demostra duas imagens recolhidas com o ajuste automático ligado sendo que na imagem da direita foi introduzido um objeto na cena (folha branca).

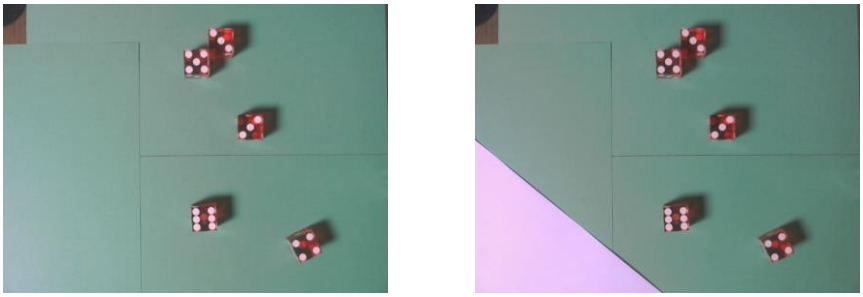


Figura 116 – Imagem com ajuste automático ligado

Como pode ser verificado nas imagens da Figura 116 existe uma diferença nas cores, apesar de subtil ao olho humano, de ambas as imagens, este efeito é provocado devido à existência do ajuste automático efetuado pela câmera.

A biblioteca *OpenCV* oferece uma *interface* para manipulação de dispositivos de recolha de imagem através da API *Video4Linux* (V4L), no entanto a interface utilizada apenas faz uso da API na versão 1, sendo que as opções de manipulação dos parâmetros *white balance* e *exposure* não se encontra disponível.

De forma a contornar este problema foi criado um *script*, apresentado no excerto de Código 2, que faz uso da versão 2 da API *Video4Linux* (V4L2).

```
#!/bin/sh
if [ "$1" = "--enable" ] || [ "$1" = "-e" ]
then
  echo "applying custom settings configuration for webcam..."
  v4l2-ctl -d /dev/video1 -c white_balance_temperature_auto=0
  v4l2-ctl -d /dev/video1 -c exposure_auto=1
  v4l2-ctl -d /dev/video1 -c white_balance_temperature=10000
  v4l2-ctl -d /dev/video1 -c exposure_absolute=500
  v4l2-ctl -d /dev/video1 -c gain=64
  v4l2-ctl -d /dev/video1 -c power_line_frequency=0
  v4l2-ctl -d /dev/video1 -c sharpness=255
  v4l2-ctl -d /dev/video1 -c backlight_compensation=0
  v4l2-ctl -d /dev/video1 -c exposure_auto_priority=0
  echo "...custom settings configuration applied"
fi

if [ "$1" = "--disable" ] || [ "$1" = "-d" ]
then
  echo "applying default settings configuration for webcam..."
  v4l2-ctl -d /dev/video1 -c white_balance_temperature_auto=0
  v4l2-ctl -d /dev/video1 -c exposure_auto=1
  v4l2-ctl -d /dev/video1 -c white_balance_temperature=4000
  v4l2-ctl -d /dev/video1 -c exposure_absolute=166
  v4l2-ctl -d /dev/video1 -c gain=64
  v4l2-ctl -d /dev/video1 -c power_line_frequency=2
  v4l2-ctl -d /dev/video1 -c sharpness=24
  v4l2-ctl -d /dev/video1 -c backlight_compensation=0
  v4l2-ctl -d /dev/video1 -c exposure_auto_priority=0
  v4l2-ctl -d /dev/video1 -c white_balance_temperature_auto=1
  v4l2-ctl -d /dev/video1 -c exposure_auto=3
  echo "...default settings configuration applied"
fi
```

Código 2 – *Script* de configuração da câmera através da API V4L2

Este *script* permite ligar e desligar o ajuste de cor automático realizado pelo dispositivo, tendo isto torna-se possível capturar imagens que não entrem em conflito com o algoritmo de *background subtraction*.

O resultado de uma amostra capturada com o ajuste automático desligado encontra-se evidenciado na Figura 117.

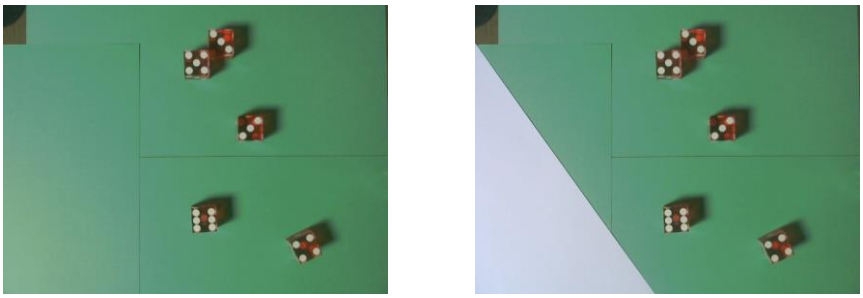


Figura 117 – Imagem com ajuste automático desligado

Analisando a Figura 117 é possível verificar que as cores entre ambas as imagens mantem-se uniforme mesmo depois de inserido um objeto na cena (folha branca).

Uma vez corrigido o problema de discrepância entre *frames* procedeu-se à aplicação do algoritmo MOG2 num *stream* de vídeo, no qual a máscara resultante do algoritmo para um *frame* encontra-se representada na Figura 118.

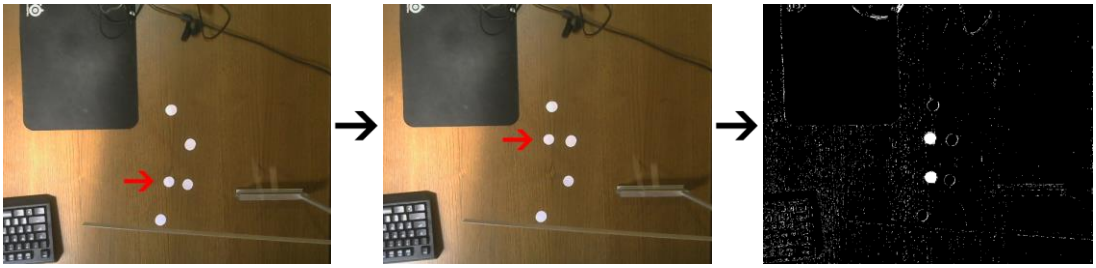


Figura 118 – Máscara resultante da aplicação do algoritmo MOG2

Após aplicadas as operações de filtragem de sombras e de abertura descrita no subcapítulo 4.4.3 obtém-se o resultado presente na Figura 119, uma máscara sem qualquer tipo de ruído contendo apenas as posições, anterior e atual, da ficha.

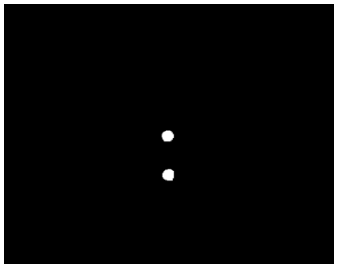


Figura 119 – Máscara MOG2 após filtragem de ruído

Depois de recortada a área referente às fichas, com base nos seus contornos, é proposta a utilização do algoritmo de descrição de características SURF, sendo que este foi testado fora do espectro da amostra da Figura 118 uma vez que as fichas utilizadas eram apenas bocados de papel branco recortado o que faz com que não possuíssem características significativas.

De forma a testar a usabilidade do algoritmo SURF foram utilizadas imagens mais complexas como substituto das “fichas” de papel branco. A Figura 120 ilustra a imagem modelo, que neste caso representaria uma ficha à procura de correspondência na imagem final para aferição da alteração de localização.

Figura 120 – Imagem modelo a ser procurada pelo algoritmo SURF

Sendo o resultado obtido da aplicação do algoritmo SURF para duas cenas diferentes apresentado na Figura 121.

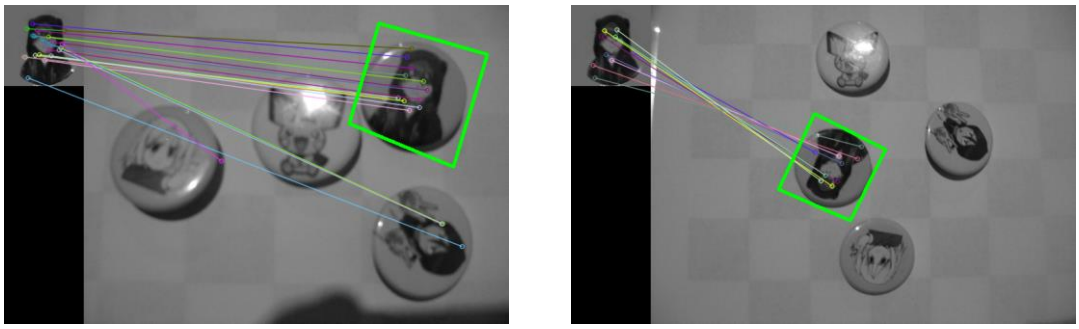


Figura 121 – Resultado da aplicação do algoritmo SURF

Sendo que a Figura 120 utilizada como modelo contém um número bastante alargado de características foi necessário o teste do algoritmo com uma imagem de complexidade mais reduzida, neste caso foi utilizado o logótipo da *Moche – Random Generation* para esse efeito, sendo o resultado de captura positivo como é evidenciado na Figura 122.

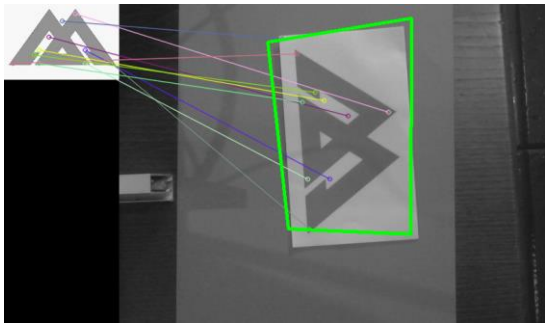


Figura 122 – Aplicação do algoritmo SURF a uma imagem de baixa complexidade

O resultado do algoritmo SURF é utilizado para tentar criar pares entre os vários recortes efetuados através da máscara do algoritmo MOG2. Caso não seja encontrado um par para uma determinada ficha ou então um par em que as suas fichas se encontrem em posições da mesa referentes a zonas de aposta diferentes é sinalizada a ocorrência de uma fraude.

5.6 Integração com dispositivo *Raspberry Pi*

Uma vez que o sistema proposto no capítulo 4 encontra-se idealizado para correr em dispositivos de capacidade reduzida em relação a um computador pessoal, foi efetuada a transposição de alguns algoritmos para efeitos de teste para um computador *Raspberry Pi Model B* com a câmera de infravermelhos *Pi NoIR*. A Figura 123 apresenta o material utilizado antes de ser montado e depois deste se encontrar montado.

Figura 123 – Computador *Raspberri Pi*, desmontado e montado

Uma vez montado o dispositivo procedeu-se à instalação do sistema operativo *Raspbian* e atualizado o *software* e *firmware* para o mesmo. De seguida foram instaladas ou ativadas um conjunto de aplicações de forma a ser possível trabalhar com o dispositivo sendo elas:

XDRP Server – Utilizado para a possibilitar o trabalho por ambiente remoto através do sistema operativo *Windows*;

Camera Module – Módulo ativado para possibilitar o uso da câmera por parte do *Raspberry Pi*;

SSH Server – Módulo ativado para possibilitar o acesso através de SSH por meio do terminal de Linux noutro computador;

OpenCV 2.4.8 (e dependências) – Compilação e instalação da biblioteca *OpenCV* para *Raspberry Pi* e respetivas dependências;

RaspiCam 0.1.1 – Biblioteca que possibilita a interface entre a câmera do *Raspberry Pi* e a biblioteca *OpenCV*.

A compilação do *OpenCV* teve de ser efetuada no próprio dispositivo uma vez que esta biblioteca encontra-se diretamente relacionada com o tipo de processador utilizado, sendo que o processador do *Raspberry Pi* (Arquitetura *ARM*) difere dos processadores comuns dos computadores pessoais (normalmente arquiteturas x86 e x64, *AMD* ou *Intel*). A compilação através do dispositivo demorou cerca de 11 horas a ser concluída.

Depois de serem efetuadas todas as instalações de *software* necessárias, os algoritmos definidos foram ajustados, incorporando a biblioteca *RaspiCam,* de forma a ser possível a captura de imagem através da utilização da câmera do *Raspberry Pi*.

Os primeiros testes efetuados consistiram na análise de performance do dispositivo para recolha e apresentação de imagem através da biblioteca *OpenCV* para diferentes resoluções e espaços de cor. A Tabela 16 apresenta os resultados obtidos dos testes mencionados, sendo que os dados recolhidos são analisados em duas formas uma sem apresentação do conteúdo ao utilizador e outra com apresentação do conteúdo ao utilizador (colunas com rótulo *imshow*).

Tabela 16 – Teste de performance do *Raspberry Pi* (captura de imagem simples)

Espaço de cores	**Número de *frames***	**Resolução**	**Tempo decorrido (*imshow*) (segundos)**	**FPS (*imshow*)**	**Tempo decorrido (segundos)**	**FPS**
RGB (3 Canais)	100	1280x960	47	2.128	13	7.692
RGB (3 Canais)	100	640x480	11	9.091	4	25
RGB (3 Canais)	100	320x240	7	14.286	3	33.333
Cinza (1 Canal)	100	1280x960	27	2.704	4	25
Cinza (1 Canal)	100	640x480	8	12.500	4	25
Cinza (1 Canal)	100	320x240	7	14.286	3	33.333

Com base nos resultados apresentados na Tabela 16 é possível verificar que o algoritmo consegue em 5 das 6 configurações capturar imagens com uma velocidade razoável, no entanto existe uma quebra de performance acentuada caso seja também apresentada a imagem ao utilizador (colunas *imshow*).

Uma vez efetuada a análise de recolha de imagem simples procedeu-se à recolha de amostras com base na aplicação do algoritmo MOG, sendo os resultados apresentados na Tabela 17.

Tabela 17 – Teste de performance do *Raspberry Pi* (captura de imagem e utilização do algoritmo MOG)

Espaço de cores	**Número de *frames***	**Resolução**	**Tempo decorrido (*imshow*) (segundos)**	**FPS (*imshow*)**	**Tempo decorrido (segundos)**	**FPS**
Cinza (1 Canal)	100	1280x960	201	0.498	179	0.559
Cinza (1 Canal)	100	640x480	49	2.0408	46	2.174
Cinza (1 Canal)	100	320x240	17	5.882	15	6.667

A utilização do algoritmo MOG foi apenas testada para a recolha de imagens no espaço de cores cinza, uma vez que este não trabalha com o espaço de cores RGB, no entanto, mesmo sendo utilizado o espaço de cores mais simples de todos, o algoritmo apresenta uma performance bastante baixa, sendo o melhor caso aproximadamente 7 *frames* por segundo para uma resolução bastante baixa. Sendo assim encontra-se provado que a utilização do *Raspberry Pi* para efetuar o reconhecimento de movimento numa cena através do método proposto não é de todo adequado.

Por fim foi testada a performance do algoritmo adaptado correspondente ao Dice Sample Analyzer (DSA) com base na mesma resolução de imagem e espaço de cores utilizado para a versão de computador pessoal mencionado no subcapítulo 5.1. Os resultados obtidos encontram-se expostos na Tabela 18.

Tabela 18 – Teste de performance do *Raspberry Pi* (captura de imagem e utilização do DSA)

Espaço de cores	**Número de *frames***	**Resolução**	**Tempo decorrido (*imshow*) (segundos)**	**FPS (*imshow*)**	**Tempo decorrido (segundos)**	**FPS**
Cinza (1 Canal)	100	640x480	45	2.222	40	2.5

Apesar do número de *frames* por segundo ser bastante reduzido é necessário ter em conta que o algoritmo utilizado efetua um número de rotinas extraordinárias, como a composição de imagem para apresentar os resultados visualmente, o que cria um atraso na apresentação do resultado final. Uma vez que o algoritmo procura por dados na mesa para cada *frame* pode-se concluir que o resultado de 2 *frames* por segundo é bastante favorável, pois num caso real seria apenas necessário calcular o primeiro *frame* após ter sido sinalizado o fim de movimento na área de jogo, perfazendo assim um tempo de processamento de aproximadamente 0.5 segundos.

6 Conclusões e trabalho futuro

O trabalho apresentado nesta dissertação teve em vista explorar soluções de forma a colmatar a falta de sistemas de controlo de gestão automática para o jogo de casino *Banca Francesa*.

As soluções propostas fizeram uso de componentes da área de computação visual de forma a capturar, tratar e analisar imagens de modo a aferir resultados consoante o âmbito em que estas se encontravam inseridas. Sendo estas denominadas *Dice Sample Generator, Dice Sample Analyzer*, *Dice Calibration* e *Motion Detection*.

Os testes efetuados à solução *Dice Sample Generator* revelaram que esta ferramenta, apesar de não ser enquadrada no sistema final proposto, possibilitou a capacidade da realização de um número bastante superior de testes. Testes estes que foram cruciais para a transição suave que existiu entre a utilização de amostras baseadas em casos gerados por computador e a utilização de amostras baseadas em casos reais.

Em relação aos testes efetuados à solução *Dice Sample Analyzer* estes excederam as espectativas iniciais, com 99.7% de precisão para casos gerados computacionalmente e 97.7% de precisão em casos reais, mesmo em cenários onde se encontravam presentes um número bastante elevado de dados (no máximo 20, para casos gerados por computador). O tempo de resposta da solução também se mostrou adequado ao pretendido, sendo obtida uma resposta sem que existisse quebra de *frames* por segundo no *input* da câmera.

A solução *Dice Calibration* mostrou ser possível serem aferidas as métricas de um dado para utilização do sistema através da utilização de uma câmera e sem recurso a qualquer tipo de medições manuais, sendo que, com base nos testes realizados, este apenas obteve uma discrepância, no máximo, de 0.5mm entre os resultados obtidos e as medições manuais efetuadas.

A solução apresentada para a deteção de fraude no jogo, *Motion Detection,* não possuiu uma análise com a extensão das soluções anteriores, uma vez que não foi possível replicar o ambiente necessário para a realização de testes, no entanto foi possível evidenciar uma proposta com capacidade de ser aplicada num contexto mais apropriado.

Como resultante do trabalho efetuado foi também efetuada uma candidatura à iniciativa *Passaporte do Empreendedorismo,* com base no modelo do sistema apresentado, a qual foi aceite nas duas instâncias de avaliação.

Com base no algoritmo do módulo *Dice Sample Analzyer* para deteção de dados foi elaborado um *paper* o qual foi submetido para a conferência *IPCV'14- The 2014 Internacional Conference on Image Processing, Computer Vision, and Pattern Recognition* em Las Vegas o qual foi aceite para apresentação e publicação, que no entanto teve de ser cancelado devido à falta de apoio para participação na conferência.

6.1 Trabalho futuro

A partir do ponto de situação apresentado na dissertação, um dos seguintes passos consiste no refinamento do algoritmo exposto na proposta do *Dice Sample Analyzer* de forma corrigir a exceção detetada apresentada no capítulo de testes, de forma a ser possível uma taxa de precisão de reconhecimento de 100%.

Em paralelo é sugerido serem procuradas alternativas ao computador *Raspberry Pi*, de forma a ser realizado um *benchmarking* dos algoritmos de maneira a ser encontrado o dispositivo ideal para utilizar no sistema de controlo e gestão.

Em relação ao módulo *Dice Calibration* é sugerida a possibilidade de incorporar novos tipos de dados com métricas diferentes dos dados ocidentais, que no entanto podem ser utilizados com os algoritmos adequados de deteção já existentes, expostos no estado de arte.

De modo a ser possível desenvolver um módulo de deteção de fraude robusto, referente à solução *Motion Detection*, é sugerida a entrada em contacto com casino que possuam o jogo *Banca Francesa* no seu reportório de forma a ser coletada informação referente ao processo de jogo.

Referências

[AVSS, 2014]	http://www.avss2014.org/ [último acesso Setembro de 2014]
[Barquet et. al., 2005]	BARQUET, Gill, ELBER, Gershon, MYUNG-SOO, Kim, Computing Minimum Enclosing Circles of a Set of Planar Curves, 2005
[Bay *et al.*, 2006]	BAY, TUYTELAARS, T., VAN GOOL, L., "SURF: Speeded Up Robust Features", 2006
[Bento et. al., 1995]	BENTO A. B. Correia, JERÓNIMO A. Silva, FERNANDO D. Carvalho, RUI Guilherme, FERNANDO C. Rodrigues, FERREIRA A. M. Silva, Automated detection and classification of dice, 1995
[BHS Böhm, 2014]	BHS Böhm, High End Casino Equipment http://www.p-eye.de/, 2014
[Boss and Alan, 2010]	BOSS, Derk, Zajic, ALAN, Casino Security and Gaming Surveillance, Novembro 2010
[Boss and Zajic, 2010]	BOSS, D., ZAJIC, A. Casino Security and Gaming Surveillance, 2010
[Chung et. al., 2009]	CHUNG, Chin-Ho, CHEN, Wen-Yuan, LIN, Bor-Liang, Image Identification Scheme for Dice Game, 2009
[Cooper and Dawson-Howe, 2004],	COOPER, W.m DAWSON-HOWE, K. Automation of Casino Game Surveillance Using Computer Vision, 2004
[Cammegh, 2014a]	Cammegh Brochure, p. 21, 2014
[Cammegh, 2014b]	Cammegh Brochure, p. 26-27, 2014
[GamersByEmail.com, 2013]	GamersByEmail.com, Dice-O-Matic mark II http://gamesbyemail.com/News/DiceOMatic, Dezembro 2013
[Goding, 2012]	GODING, Jim, Casino Surveilance Operations Manual, 2012
[Hamilton, 2014]	HAMILTON, Laura www.lauradhamilton.com/random-lessons-online-poker-exploit, Fevereiro 2014
[Haralick and Shapiro, 1991]	HARALICK, R., SHAPIRO, L., Computer Vision and Robot Vision, pp 174-185, 1992
[Heckbert and Paul, 1997]	HECKBERT, Paul, GARLAND, Michael, Survey of Polygonal Surface Simplification Algorithms, 1997
[Hollinger and Ward, 2004]	HOLLINGER, Geoff, WARD, Nick, Introducing Computers to Blackjack: Implementation of a Card Recognition System using Computer Vision Techniques, Março 2004
[Hsu *et al.*, 2012]	HSU, Gee-Sern, PENG, Hsiao-Chia, YEH, Shang-Min, Color and Illumination Invariant Dice Recognition, Outubro 2012
[Huang, 2007]	HUANG, Kuo-Yi, An Auto-Recognizing System for Dice Games Using a Modified Unsupervised Grey Clustering Algorithm, Novembro 2007
[ITU-BT.601-7, 2011]	Recommendation ITU-BT.601-7, Março 2011
[Jankowski, 2006]	JANKOWSKI, Mariusz; Erosion, Dilation and related operators, June 2006
[Jogo Responsável,2014]	Banca Francesa – Jogos de Casino, Jogo Responsável, http://jogoresponsavel.pt, 2014
[KadewTraKuPong and Bowden, 2001]	KADEWTRAKUPONG, P., BOWDEN, R., An improved adaptive background mixture model for real-time tracking with shadow detection, 2001

[Learned-Miller, 2011] LEARNED-MILLER, Erik G., Introduction to Computer Vision, Janeiro 2011

[Lei do Jogo, 2014] Inspeção de jogos – Legislação (Lei do Jogo) http://www.turismodeportugal.pt/Portugu%C3%AAs/AreasAtividade/InspecaoJogos/legislacao/Pages/Legislacao.aspx [último acesso Setembro de 2014]

[Lowe, 2013] LOWE, David, The computer vision industry, Novembro 2013

[Matas *et al.* 2002] MATAS, J., CHUM, O., URBAN, M. PAJDLA, T., Robust wide baseline stereo from maximally stable extremal regions", 2002

[Mikolajczyk and Schmid, 2002] MIKOLAJCZYK , Krystian, SCHMID, Cordelia, An affine invariant point detector, 2002

[Ng, 2014] NG, Andrew, Machine Learning Course – Video Lecture XIII. Clustering (Week 8) – K-Means Algorithm, 2014

[Otsu, 1979] OTSU, Nobuyuki, A threshold selection method from gray-level histograms, 1979

[Pimentel, 2010] PIMENTEL, João, Machine Vision in Casino Game Monitoring, Outubro 2010

[SET-Production, 2014] SET-Production Brochure, p. 1-3, 2014

[Sezgin and Sankur, 2003] SEZGIN, Mehmet, SANKUR, Bülent, Survey over image thresholding techniques and quantitative performance evaluation, Maio 2003

[Susuki and Abe, 1985] SUSUKI, Satoshi, ABE, Keiichei, Topological Structural Analysis of Digitalized Binary Images by Border Following, 1985

[Szeliki, 2010] SZELIKI, Richard, Computer Vision: Algorithms and Applications, Microsoft Research 2010

[WildLizzy, 2013] Wildlizzy, http://www.wildlizzy.de/index.php/en/, 2013

[Wu *et al*., 2012] WU, Hao-Yu , RUBINSTEIN, Michael, SHIH, Eugene, GUTTAG, John, DURAND, Frédo, FREEMAN, William T., Eulerian Video Magnification for Revealing Subtle Changes in the World, 2012

[Youtube, Paulof ,2013] Youtube PAULOF, Nick https://www.youtube.com/watch?v=WbzoYJLg60s, Maio 2013

[Yuen et. al., 1990] YUEN, H. K., PRINCEN, J., ILLINGWORTH, J., KITTLER, J., Comparative Study of Hough Transform for Circle Finding, 1990

[Zemanta, 2007] ZEMANTA, Computer Vision Startups, Tangam Gaming Systems, Outubro 2007 http://www.shawnlankton.com/2007/10/computer-vision-startups/

[Zhang amd Hsu, 2012] ZHANG, Xun-Jia, HSU, Gee-Sern, Stereo Vision Based Dice Recognition Based Dice Recognition System and Stereo Based Recognition Method for Uncontrolled Environments, Maio 2012

[Zheng and Green, 2007] ZHENG, Chunhui, GREEN, Richard, Playing Card Recognition Using Rotational Invariant Template Matching, Dezembro 2007

[Zivkovic, 2004] ZIVKOVIC, K., Improved adaptive Gausian mixture model for background subtraction, 2004

[Zutis and Hoey, 2009] ZUTIS, Krists, HOEY, Jesse, Who's Couting? Real-Time Blackjacking Monitoring for Card Counting Detection, 2009

Anexos

Anexo A – Tabela de empresas baseadas em computação visual

Ramo	Empresas
Assistência à condução automóvel	Iteris, MobilEye
Rastreamento de Olhos e Cabeça	Mirametrix, Smart Eye, SMI
Filmes e Vídeo (análise desportiva)	Hawkeye, Libero Vision, QuesTec, Red Bee Media, Amisco, Sportvision
Filme e Vídeo	2d3, Image Metrics, Imagineer Systems, MirriAd, Mova, Orad, Ooyala, VideoSurf
Jogos e Reconhecimento gestual	GestureTek, Microsoft Kinect, Reactrix, Sony EyeToy
Automatização e Inspeção Industrial: Indústria Automóvel	CogniTens, Perceptron, KLA-Tencor, Orbotech
Automatização e Inspeção Industrial: Alimentação e Indústria Agrícola	Montrose Technologies, Ellips
Automatização e Inspeção Industrial: Impressão e Indústria têxtil	Advanced Vision Technology, Elbit Vision Systems Ltd., Mnemonics, Xiris Automation
Automatização e Inspeção Industrial: Outros	Adept, Avalon Vision Solutions, Basler,Hermany Opto Electronics, JLI vision, LMI Technologies,MVTec, NeuroCheck GmbH, PPT Vision, SIGHTech, Seegrid, Virtek Vision Internacional, Wintriss Engineering
Médicas e Biomédicas	Claron Technology, Mirada Medical, Noesis
Reconhecimento de objectos para dispositivos móveis	Kooaba , Mobile Acuity, SnapTell
Fotografia Panorâmica	Cloudburst Research, HumanEyes, Kolor
Rastreamento de Pessoas	BrickStream, Reveal, VideoMining
Sistemas de monitorização de segurança	Poseidom/MG Internacional
Segurança: Biométricas	Aurora, AuthenTec, Digital Persona, L-1 Identity Solutions, Viion Systems
Segurança: monitorização e vigilância	Aimetis, Briefcam, Cernium, Cognimatics, EVITECH, Genetec Honeywell, Imagemetry, IntelliVision, ObjectVideo, Vitamin D
Modelação tridimensional	Creative Dimension Software, Eos Systems, Creaform
Tráfego e gestão rodoviária	Image Sensing Systems , Yotta
Aplicações Web	Face.com, Incogna, LUT Technologies,

Ramo	Empresas
	Photometria
Sistemas de visão com propósito geral	Cognex, Evolution Robotics, Matrox Imaging, National Instruments, Neptec, Newton Research Labs , Point Grey Research, Sarnoff, Seeing Machines, Soliton, SpikeNet, Supercomputing Systems, TYZX, ViSSee, VISIONx, Vitronic

Anexo B – Dice Recognition Based On Dot Segmentation And Clustering Analysis

Dice Recognition Based On Dot Segmentation And Clustering Analysis

Diogo Pinho
DEI
ISEP
Porto, Portugal
1090558@isep.ipp.pt

Pedro Oliveira
DEI
ISEP
Porto, Portugal
ped@isep.ipp.pt

Nuno Bettencourt
DEI
ISEP
Porto, Portugal
nmb@isep.ipp.pt

***Abstract*—The following article presents a solution for the automatic recognition and classification of dice on a typical casino gaming environment using a single camera. The solution makes use of computer vision technology to retrieve the key characteristics of the scene in this case the dots of dice. Retrieved dots are then cataloged and grouped into clusters based on their distance correlation. These clusters are analyzed in order to group the dots that form each individual die. The presented solution is theoretically capable of correctly classifying most sets of western-type dice independently of their size or color. For testing purposes two software prototypes were developed, a sample generator and a sample analyzer based on a typical western-style casino die.**

Keywords—object segementation, dice recognition, dice classification, real-time image processing, clustering analysis.

I. INTRODUCTION

The market of computer vision in recent years as seen a great improvement and growth variety, this is due to the fact that solutions that once were considered impractical from a hardware viewpoint are now capable of being implemented. These solutions can now be implemented in a way that are non-intrusive and can be kept low-profile so they do not disturb the people affected by it. The combination of these factors makes computer vision based solutions a great candidate for reliable monitoring system around in diverse areas. One of the areas that is always looking to improve its monitoring capabilities solutions, but also needs to keep them as discreet as possible is the casino security area. Current state of the art casino material and companies already provide solutions for roulette based games and some card based games, such as the *EyeBall*™ from *Cammegh*™. While it stands true that solutions for some games are already developed, the lack of material based on computer vision for dice based games is still on research phase. This solution, if created, could improve the way casinos store the records of the gaming rounds as well as prevent some cases of fraud that are inherent from dice based games, such as, the player making a bet when the dice already stopped moving, which is a rather unreliable task to be accomplished by one croupier only and that happens in the case of a game called *Banca Francesa.*.

Research on dice recognition through the usage of computer vision technology was already presented on an academic level by Hsu et al [1]. Their solution made use of multiple cameras in order to identify various types of dice under different lightning conditions. Although this solution can be considered the state of the art in dice recognition the usage of multiple cameras can be seen as an inconvenience under certain environments where discretion is a must.

Chung et al [2] makes use only of a single camera and also base their research on dot distances similar to this article, but the usage of Asian-type dice creates conditions that are favorable to the recognition such as the one faced die having a larger dot compared to the dots of the other faces, and also the dots on a Asian-type dice face are more compact than the Western-type, making the recognition more straightforward.

One known academic approach by Correia et al [3] in 1995 reached a prototype level and made use of a monochrome CCD camera in which the dice dots were extracted through online image analysis and afterwards a sort of template matching took place to classify the dice in play but for unknown reasons the system proposed was discontinued and never reached full production.

The rest of this article is organized as follows. Section II gives a summary of the structure of the solution presented as well as its components. Section III explains the setup of the environment as well as the material specification. Section IV explains the image processing steps of the solution. Section V and VI explains the theoretical concept of dice recognition used in the solution. Section VII gives the result data from the implemented solution. And finally Section VIII presents the conclusion of this work as well as the next route to take given the results acquired.

II. PROCESS WORKFLOW

The solution proposed in this article makes use of two software prototypes. The software called *Dice Sample Generator* (DSG), is responsible for generating samples for testing the solution proposed. This software is based on template images captured from a real life setting and simulates the final positions of a specified number of dice on a table. This piece of software is used to create large data sets that were impractical to be handmade created. The second piece of software is a *Dice Sample Analyzer* (DSA), which is responsible for analyzing the samples created by the DSG.

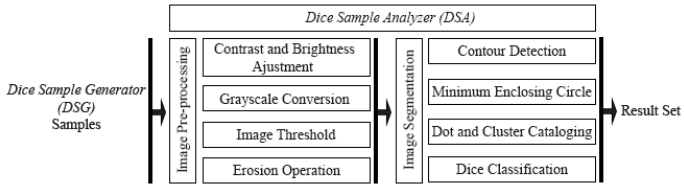


Fig. 1 – Process workflow

Fig. *1* shows the steps of the DSA that will be the main focus of this article. Two steps divide the DSA, the *Image Processing*, which is responsible to setup the image in the proper conditions so that the *Image Segmentation* step can take place and perform the detection and classification of the dice.

III. Environment Setup and Material Specification

The DSG was developed in pure Java because of the ease of use on simple image manipulation. The DSA was developed in C++ using the OpenCV 2.4.7 libraries for Operative Systems based on Linux x64 architectures.

The template material for the DSG was created using a Logitech HD Webcam C270. The camera was set approximately 42cm from the center of the "gaming table" on a top view (90 degrees). The "gaming table" consists on a green uniform paper with A4 (ISO 216) size attached to a corkboard by four pushpins on each corner. The dice used were translucent red with a 1.9cm razor edge and white dots with a 0.5cm diameter. All the templates used with the DSG were recorded with a 1692x1194 pixel resolution on the webcam.

Fig. 2 – Setup for template capturing, with white dice.

Fig. *2* shows the camera and "gaming table" setup used to record the dice templates for the DSG, to note that the dice presented on the image were not the dice used for template purposes.

IV. Image Processing and Dot Segmentation

In order to retrieve the key characteristics (the dots) of the dice displayed on a sample image it is necessary to process the image so the key characteristics become clearer. To do so, the application of common image pre-processing algorithms is necessary.

A. Image Contrast and Brightness Adjustment

To simulate a standard casino environment, red translucent dice with white dots and a green background was used. Therefore, it is possible to segment the dice dots based on their intensity value.

$$g(i,j) = \alpha \times f(i,j) + \beta \qquad (1)$$

In order to get all pixels that define the areas of the dots with approximately the same intensity value a contrast and brightness adjustment is applied (1) [4].

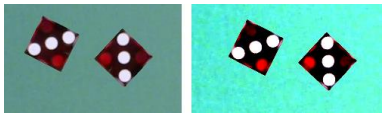


Fig. 3 - Original image $f(i,j)$ and result image $g(i,j)$

The values for α (*gain*) and β (*bias*) were iterated until it was perceptible that the pixel areas corresponding to the dots were rather uniform in intensity and all close to the value of a white pixel. The ideal value found for β was -200 and for α was 3 as show in Fig. 3.

B. Grayscale Conversion and Image Threshold

Once the contrast and brightness adjustment is applied the image is converted to a gray scale color space representation. This is done because threshold operations work with intensity values rather than the common *Red-Green-Blue* (RGB) representations. To do so, a conversion algorithm was used [5][6].

After the image is converted the pixel areas that represent the dots are now with values close to the maximum intensity allowed (white pixel, 255).

$$g(i,j) = \begin{cases} 255 & , \quad f(i,j) > t \\ 0 & , \quad f(i,j) \leq t \end{cases} \qquad (2)$$

The threshold operation [5][7] (2) is now applied in order to segment the dice dots from the rest of the image. Due to the fact that the dots were composed by pixels with high intensity values, the value for the threshold (t) can be set to a really high value.

Fig. 4 – Threshold operation result $g(i,j)$

The value 240 for t was selected by decreasing its value (starting at 255) until all the dots areas were represented on the binary image as shown in Fig. 4.

C. Morphological Operation (Erosion)

Even though the dots are segmented from the scene, given the resolution of the sample images used, some dots appeared to be connected, in the example the dots from a six-valued face of a die. In order to avoid false interpretation of the detected areas, two or more dot areas being treated as one single area, an erosion algorithm [5] [8] was applied.

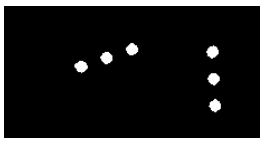


Fig. 5 – Erosion operation result.

This algorithm is used so that darker areas of the image can expand and the lighter areas of an image contract. This is done so the dots that are connected become separated elements as it is shown in Fig. 5. To apply the erosion operation a value k needs to be defined in order to set the size of the kernel to be used. The final given value is defined by increasing the value between the minimum one until the ideal value was found which the dots stopped being connected to each other. The value reached for k was 2.

Even though this process affects the radius of the dots, the lost information is irrelevant, because only the center point of the dots is relevant for the algorithm to be functional.

D. Contour Recognition and Minimum Enclosing Circles

After the previous steps, only white pixel areas compose the sample image corresponding to the dots. As a result, \a contour detection algorithm [5][9] was applied.

Each contour found by the algorithm represents the delimiting area of a dot. In this case no false positives are generated because of the guaranteed integrity of the data that is given by the DSG. On a real case scenario an additional step needs to be applied in order to guarantee that the detected contour was in fact a circular area, for example the *Hough Circle Transform* algorithm [5][10].

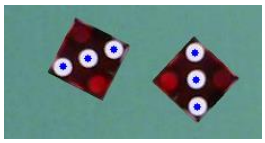


Fig. 6 – Dice dots centers represented on the original sample image

The center points are then defined by finding the minimum enclosing circle [5] [11] for each of the contours. Since the dots on the dice are also circular objects the center of the minimum enclosing circle for the contour analyzed is also the corresponding center for the dot. Fig. *6* shows a blue circle on top of each dot center.

V. Dot Cataloging and Cluster Definition

The previous chapter explained the process used to retrieve the information of the center points of each dot on the sample image. This information only represents a set of points on a two-dimensional plane with no other meaning. In order to classify the dice on the scene, distance relations between these points have to be defined based on the existing metrics inherited by the model die used.

A. Template Die Metric Relations

In a western-type die all possible face combinations are generated following a certain template model.

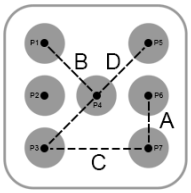


Fig. 7 – Die template model and distance relations.

Fig. *7* represents the scheme in which all possible die faces are generated as well as the geometric restrictions inherent to a die. The die centers were enumerated by their appearance when searching top-to-bottom and left-to-right. This gives the set of points ranging from P_1 to P_7.

$$\begin{aligned} \|\vec{A}\| &= \|\overrightarrow{P_7P_6}\| \\ \|\vec{B}\| &= \|\overrightarrow{P_4P_1}\| \\ \|\vec{C}\| &= \|\overrightarrow{P_7P_3}\| \\ \|\vec{D}\| &= \|\overrightarrow{P_5P_3}\| \end{aligned} \tag{3}$$

The vectors magnitude (3) represent all possible relevant distances between any set of two points that belong to the same face of a given die.

B. Dot Cataloging

Once the distance rules are established for the distances and the center points of the dots are already known, a structure graph, is created in order to catalog the information in a meaningful way, being the dot centers the vertices and the distances the edges of the graph.

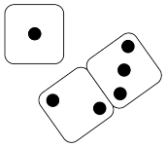


Fig. 8 – Theoretical example case.

Fig. *8* will represent an example case for the dot cataloging process explanation.

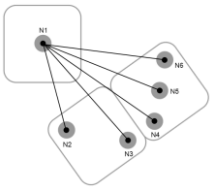


Fig. 9 – Distances between $N1$ and all other points.

For each dot center point (N_a) on the figure the Euclidean distance between every other dot center point (N_b) on the image is calculated.

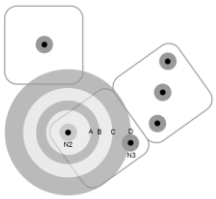


Fig. 10 – Analysis between points $N2$ and $N3$

These distances are filtered based on the previous enunciated relevant possible distance relations. In Fig. *10* it can be seen that the point N_2 and the point N_3 share a relation distance of $\|\vec{D}\|$, so this relation will be represented in the graph.

When filtering the distances in a real case scenario a deviation value needs to be taken into account based on the image resolution, this is because correct exact values for distances are almost impossible to achieve, so a small but yet existent error margin needs to be specified.

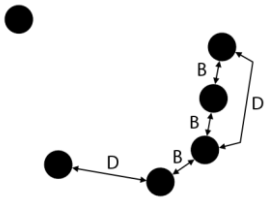


Fig. 11 – Final graph with filtered distances.

Fig. *11* shows the final generated graph for the theoretical example to denote that as it can be seen the dot's center points of N_3 and N_4 are connected with a $\|\vec{B}\|$ distance even though they are not part of the same die, this is a possibility that happens when sometimes dice are so close to each other that the distances between points of different dice are comprehended in the relevant distances spectrum. This occurrence prevents the categorization of dice by only counting the number of vertices that belong to a group of connected dots, a cluster.

C. Cluster Definition

A cluster is seen as a group of points that are connected with each other. Even though a cluster by itself does not represent a die face, as presented in the previous sub-chapter, the possibilities are that a cluster of points is still a good candidate for a singular die.

Even if a cluster has more than one die, chances are that normally no more than two or three dice compose the cluster. Although the algorithm is designed to work with the graph in a way that the whole graph can be treated as one cluster the separation by clusters is viable because it can resolve some dice faces beforehand.

Clusters that are defined by only one point are one value faced die and clusters defined by two points are two faced dice. This separation also helps in the comprehension of the algorithm and also to keep track of possible unexpected behavior by analyzing small clusters with normally two to three dice instead of one big group of dice. The dots are also organized within the cluster by their appearance on the two-dimensional plane from bottom-to-top and left-to-right, this is crucial because it will help to keep the integrity of the algorithm on the classification of the clusters.

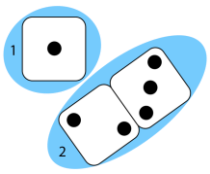


Fig. 12 – Clusters found on the theoretical example.

Fig. *12* shows the clusters defined on the example case. The image shows that two clusters are present. Even though three dice compose the scene it is already possible to classify one die because one of the clusters is composed by only one point.

VI. Cluster Analysis for Dice Classification

The dot information of the dice is now grouped into clusters based on the meaningful distances between dots, so it is necessary to evaluate the generated clusters in order to retrieve the dice values contained of each cluster. Since a cluster can contain more than one die, as it was shown on the previous chapter, a method for retrieving each individual die from a cluster is proposed.

A. Dice Classification Algorithm

The proposed method to find and classify a die in a cluster iterates through every point of a cluster. For each point that is not used in an already classified die a search algorithm is applied for each possible pattern of a die face. If the search algorithm successfully finds points in the cluster that corresponds to a given face pattern the correspondent dots are marked as used. The algorithm stops when every single dot is marked as used and all the dots are classified into their respective dice. Each cluster follows the same method until all the clusters are resolved.

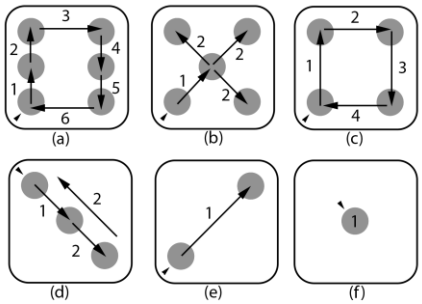


Fig. 13 – Die search pattern algorithm steps for each face

The search algorithm always searches for the highest face value first; this has to be done because for example a four faced die could be marked as two faced dice erroneously. Fig. *13* represents the depth step on the algorithm in which each points are searched, this always works because the starting points are always the extremity points of a die thanks to the early organization of the dots on the cluster.

VII. Exprimental Results and Evaluation

In order to create proof of the concept, one thousand samples were created with the DSG with each sample containing a number of dice that could range from one to twenty chosen randomly when the image is created.

A. Sample set measure conversions

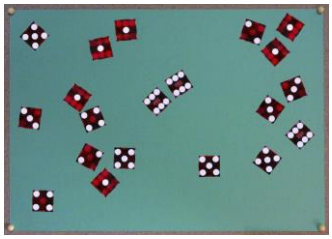


Fig. 14 – Example of a sample from the set used.

Fig. *14* illustrates a sample used in the experiment. The original images generated by the DSG had a resolution of 1672x1194 pixels, but the DSA shrinks the image to half of its width and height, 836x597 pixels accordingly, this image resize is done because webcams normally don't record at such higher resolution and real time analysis is also lighter if the image size is reduced.

TABLE I. MEASURED MEANINGFUL DISTANCES OF A DICE

Distance Type	Distance Value (cm)
$\|\vec{A}\|$	0.55
$\|\vec{B}\|$	0.80
$\|\vec{C}\|$	1.10
$\|\vec{D}\|$	1.60

Table 1 presents the corresponding values of each meaningful distance measures in the real dice used for the templates.

$$spatial\ multiplier = \frac{edge_{real}}{\left(\frac{edge_{pixel}}{scale\ factor}\right)} \quad (4)$$

The conversion between pixels and real life measures is done through the usage of a $spatial\ multiplier$(4). The $spatial\ multiplier$ takes into account the scaling factor of the shrinking ($scale\ factor$), in this case 2, the real edge value of the face of the die ($edge_{real}$), 1.9cm, and the edge pixel value of the templates used ($edge_{pixel}$), 110 pixels. A value of 0.11cm for deviation is also introduced when the distances between retrieved points is computed.

B. Sample set retrieved results

From the 1000 samples used, 997 where correctly identified, 3 were incorrectly identified. This gives a 99.7% accuracy.

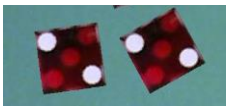


Fig. 15 – Exception case.

The three results that did not deliver an expected result contained cases that after analyzing where deemed ambiguous with the current algorithm implementation. Fig. *15* shows one of these cases where a two dice should be recognized but instead 3 dice were recognized, a two faced die and two one faced dice, this erroneous results can be prevented in two possible ways, first if the number of dice used is known then there is an impossibility for the generated result set and a new search as to be made, the other solution is to check the surround area of a one faced die, if the area is not uniform then a false positive was presented and a new search needs to be done.

VIII. CONCLUSIONS AND FUTURE WORK

The precision values achieved by the algorithm exceed the team expectations even in scenarios with more than twenty dice exposed. While this is still ongoing work, fine tuning the algorithm is one of our next steps.

The final results are favorable for the next stages of work, which would consist in adapting the current solution to real case scenarios via the input of a webcam. The distance rules suggested in this solution must also be setup dynamically through the analysis of a die face. In the future, the solution will be tested with different resolution in order to specify an algorithm to determine the ideal value of error deviation that is taken into account when calculating the relevant distances between dice. The future solutions that derive from this line of work should take in consideration if it is possible to adapt the camera feed for different angles and the results that derive from that adjustment. The solution works based on the color of the dots so future solutions must take into account how the lightning changes affect the algorithm, and how it should be adapted in order for it to work.

REFERENCES

[1] Gee-Sern, Hsu; Hsiao-Chia, Peng; Shang-Min, Yeh, "*Color and Illumination Invariant Dice Recognition*", Systems, Man, and Cybernetics (SMC), October 2012

[2] Chin-Ho, Chung; Wen-Yuan, Chen; Bor-Liang, Lin, "*Image Identification Scheme for Dice Game*", Brunel University, 2009

[3] Correia B.A.B.; Silva, J.A.; Carvalho F.D.; Guilherme, R.; Rodrigues, F.C.; Ferreira A.M.S., *Automated detection and classification of dice*, SPIE 1995

[4] Szeliki, Richard, Computer Vision: Algorithms and Applications, Microsoft Research, 2010.

[5] Bradski, Gary; Kaehler, Adrian; *Learning OpenCV*, O'Reilley Media, Inc.,September 2008.

[6] Recommendation ITU-R BT.601-7, March 2011

[7] Sezgin, Mehmet, Sankur, Bülent, Survey over image thresholding techniques and quantitative performance evaluation, May 2003

[8] Jankowski, Mariusz; Erosion, Dilation and related operators, June 2006

[9] Susuki, Satoshi; Abe, Keiichi, Topological Structural Analysis of Digitalized Binary Images by Border Following, 1985

[10] Yuen, H. K. and Princen, J. and Illingworth, J. and Kittler, J., *Comparative study of Hough transform methods for circle finding*, 1990.

[11] Barquet, Gill, Elber, Gershon, Kim, Myung-Soo Kim, Computing the Minimum Enclosing Circle of a Set of Planar Curves.

Anexo C – Candidatura ao *Passaporte do Empreendedorismo*

(Anexo retirado a pedido do estudante)