UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Desenvolvimento de Jogos em Realidade Aumentada: Problemas e Soluções

José Sousa Ribeiro

Relatório de projeto para obtenção do Grau de Mestre em
Design e Desenvolvimento de Jogos Digitais
(2º ciclo de estudos)

Orientador: Prof. Doutor Frutuoso Silva

Covilhã, Outubro de 2016

Agradecimentos

Este relatório de projeto é o resultado de mais um final de ciclo de estudos na minha vida e é o fruto de um trabalho que não seria possível sem a ajuda direta ou indireta de várias pessoas que contribuíram para o mesmo, por isso dou os meus sinceros agradecimentos a todas elas:
Agradeço ao meu orientador Prof. Doutor Frutuoso Silva tanto pela oportunidade de me direcionar para a obtenção de grau de Mestre em Design e Desenvolvimento de Jogos Digitais como pela disponibilidade em incentivar, motivar e oferecer todo o apoio e ajuda para a conclusão do projeto.
Agradeço aos meus Pais por me terem educado da melhor maneira e pela ajuda e apoio que sempre ofereceram para alcançar as minhas metas. Também um especial obrigado ao meu irmão pela ótima companhia e humor necessário para descontrair quando preciso.
Um final agradecimento a todos os colegas e amigos pela constante presença e apoio em todas as etapas do meu percurso académico, o que o tornou o mais acolhedor e agradável possível.

Obrigado a todos,
José Sousa Ribeiro

Resumo

Este relatório apresenta o trabalho realizado para obtenção do Grau de Mestre em Design e Desenvolvimento de Jogos Digitais (2º ciclo de estudos). A área de estudo em questão é a Realidade Aumentada, uma área que se tem vindo a desenvolver bastante nos últimos anos. Um crescimento que provém principalmente do baixo custo de *hardware* necessário para o desenvolvimento de aplicações. Numa era em que a presença de dispositivos móveis é tão grande, há uma aposta crescente na criação de produtos em Realidade Aumentada para este tipo de dispositivos. O presente trabalho tem como objetivo efetuar um levantamento dos principais problemas no desenvolvimento de videojogos em Realidade Aumentada e apresentar possíveis soluções. No desenvolvimento de videojogos são vários os fatores que levam ao sucesso do produto, entre eles, um bom *design* da mecânica e uma boa jogabilidade levam a uma maior imersão e motivação do jogador. Assim, são feitas análises de vários jogos em Realidade Aumentada, identificados problemas de mecânica e interação, e apresentadas soluções através de uma aplicação prática: desenvolvimento de um jogo em Realidade Aumentada que teste as soluções propostas.

Palavras-chave

Realidade Aumentada, videojogos, programação, problemas, soluções, *gameplay*, modos de interação, unity, opencv.

Abstract

This report presents the work done to obtain the Master's Degree in Design and Development of Videogames (2nd cycle of studies). The study area in question is the Augmented Reality, an area that has developed greatly in recent years. A growth that comes mainly from the low cost of hardware required for the development of applications. In an era in which the presence of mobile devices is so large, there is a growing focus on creating products in Augmented Reality for this type of devices. This paper aims to make a survey of the main problems in the development of video games in augmented reality and present possible solutions. In the development of video games there are several factors leading to the success of the product, among them, a good design of mechanics and good gameplay lead to greater immersion and motivation of the player. Thus, there will be made analyzes of various videogames of augmented reality, identified problems of game mechanics and interaction, and presented solutions through a practical application: development of a videogame in Augmented Reality to test the proposed solutions.

Keywords

Augmented Reality, videogames, programming, problems, solutions, *gameplay*, interaction modes, unity, opencv.

Conteúdo

1 Introdução **1**
1.1 Objectivos 1
1.2 Organização do Documento 1

2 Estado da Arte **3**
2.1 Introdução 3
2.2 Realidade Aumentada 3
2.2.1 Definição 3
2.2.2 Utilização 4
2.3 Videojogos em Realidade Aumentada 6
2.3.1 Warp Runner 6
2.3.2 Ingress 7
2.3.3 Drakerz-Confrontation 7
2.3.4 PulzAR 8

3 Identificação do Problema e Definição de Soluções **9**
3.1 Introdução 9
3.2 Identificação do problema 9
3.3 Definição de soluções 10

4 Tecnologias e Ferramentas Utilizadas **11**
4.1 Introdução 11
4.2 Unity 11
4.3 OpenCV 12

5 Implementação **13**
5.1 Introdução 13
5.2 Integração da biblioteca (OpenCV) 13
5.3 Acesso à câmara e obtenção de *frames* 13
5.4 Processamento dos frames 14
5.4.1 *GrayScale* 15
5.4.2 *Blur* 15
5.4.3 *Canny* 16
5.4.4 *Dilate* 17
5.4.5 *Find Contours* 18
5.4.6 Deteção de figuras geométricas 19
5.5 Estabelecimento de marcadores 22
5.6 Videojogo de teste - ARSlidingPuzzle 23

6 Conclusão e Trabalho Futuro **25**

Bibliografia **27**

Lista de Figuras

2.1 Visualização de produtos de catálogo em Realidade Aumentada. 4
2.2 Furacão aumentado nos estúdios do *Weather Channel*. 4
2.3 Percurso de GPS sobreposto no trajeto usando Realidade Aumentada. 5
2.4 Informação sobre exposição aumentada utilizando a tecnologia de RA no dispositivo móvel. 5
2.5 Videojogo em Realidade Aumentada: *Magic Leap*. 6
2.6 Videojogo em Realidade Aumentada: *Warp Runner*. 6
2.7 Videojogo em Realidade Aumentada: *Ingress*. 7
2.8 Videojogo em Realidade Aumentada: *Drakerz*. 7
2.9 Videojogo em Realidade Aumentada: *PulzAR*. 8

5.1 Transformação de Perspetiva. 22
5.2 Marcador de RA ilustrado pelo contorno verde. 23
5.3 Imagem de quebra-cabeças dividida e baralhada. 23

Lista de Excertos de Código

5.1 *WebCamTexture to Mat* 14
5.2 *GrayScale* 15
5.3 *Blur* 16
5.4 *Canny* 16
5.5 *Dilate* 17
5.6 *Find Contours* 18
5.7 *ApproxPolyDP* 19
5.8 *Deteção de Figuras Geométricas* 20

Capítulo 1

Introdução

1.1 Objectivos

O projeto tem como principal objetivo a identificação de problemas e procura de soluções no desenvolvimento de jogos em Realidade Aumentada. Para o cumprimento do objetivo são propostas as seguintes tarefas:

1. Pesquisa de videojogos em Realidade Aumentada.
2. Análise de *gameplay* e modos de interação de jogos em Realidade Aumentada. Identificação de problemas.
3. Proposta de soluções para alguns dos problemas identificados.
4. Implementação prática das soluções: Desenvolvimento de um videojogo em Realidade Aumentada, utilizando um motor de jogo.

1.2 Organização do Documento

A estrutura do presente documento está dividida em vários capítulos que são:

1. **Introdução** - apresenta o projeto desenvolvido, a área onde se enquadra, os objetivos principais e a organização do documento.
2. **Estado da Arte** - aborda de forma sucinta o tema do projeto, e apresenta os trabalhos desenvolvidos na área.
3. **Identificação do Problema e Definição de Soluções** - faz o levantamento de problemas em jogos em Realidade Aumentada a partir de uma análise de *gameplay* e modos de interação e uma proposta soluções de desenvolvimento.
4. **Tecnologias e Ferramentas Utilizadas** - descreve as ferramentas que permitiram o desenvolvimento do projeto bem como as tecnologias utilizadas.
5. **Implementação** - apresenta o videojogo de teste desenvolvido com base nas soluções propostas, e descreve todas as fases de implementação.
6. **Conclusões e Trabalho Futuro** - apresenta as principais conclusões retiradas do projeto e respetivas análises, bem como apresenta o trabalho futuro.

Capítulo 2

Estado da Arte

2.1 Introdução

Neste capítulo é abordado o tema de Realidade Aumentada em videojogos e são apresentados alguns jogos desenvolvidos na área.

2.2 Realidade Aumentada

2.2.1 Definição

A Realidade Aumentada é uma tecnologia que integra elementos virtuais no nosso mundo real. Elementos estes como imagens e modelos 3D, sons ou até mesmo vídeos. A criação de aplicações de Realidade Aumentada (RA) requer a utilização de software adequado que permite ao programador associar informação digital a marcadores de RA no mundo real. Através de dispositivos com câmara, o programa faz um reconhecimento de imagens e padrões do mundo real e o utilizador vê os elementos virtuais a serem estrategicamente adicionados, posicionados e orientados no mundo real, de maneira a que se crie um ambiente misto (real e virtual). É de realçar a diferença entre Realidade Aumentada e a também conhecida Realidade Virtual. Esta segunda não adiciona elementos virtuais num mundo real mas sim provoca uma sensação de imersão, isto é, o individuo tem a sensação real de estar inserido num mundo virtual e ser capaz de manipular elementos virtuais como se fossem reais.

2.2.2 Utilização

Atualmente esta tecnologia tem uma variedade enorme de utilizações pois está presente em inúmeras áreas:

1. Catálogo IKEA [12]

   O IKEA disponibiliza uma *App* para *Android* e *iOS* onde os utilizadores podem visualizar produtos do catálogo como sofás e mesas posicionados em qualquer sitio das suas casas antes de procederem com a compra.

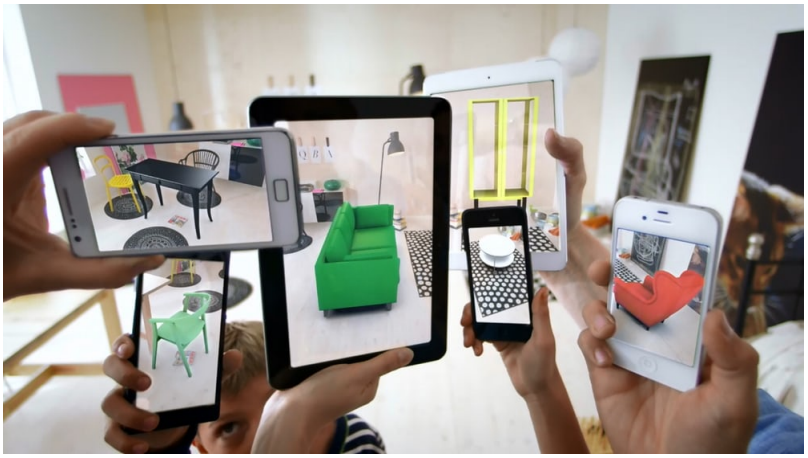

Figura 2.1: Visualização de produtos de catálogo em Realidade Aumentada.

2. Realidade Aumentada em canal de meteorologia [6]

   O *Weather Channel* aplica a tecnologia de Realidade Aumentada no próprio estúdio na época de furacões nos Estados Unidos de maneira a conseguirem mostrar o perigo temporal.

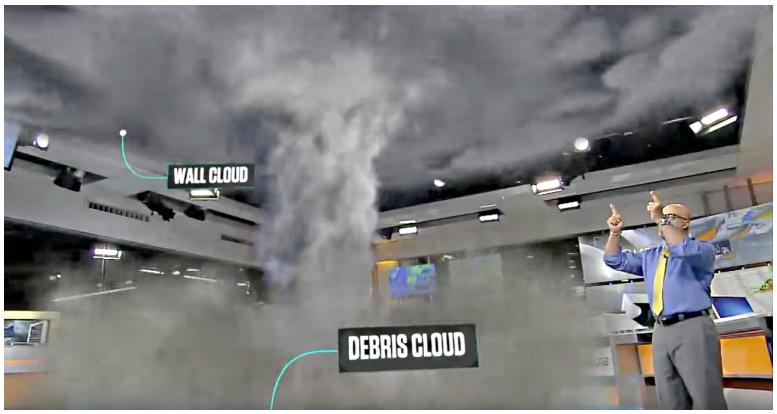

Figura 2.2: Furacão aumentado nos estúdios do *Weather Channel*.

3. Wikitude Drive [14]

   Uma *App* de GPS para *Android* onde o trajeto é aumentado com a sobreposição do percurso em vez do típico mapa abstrato.

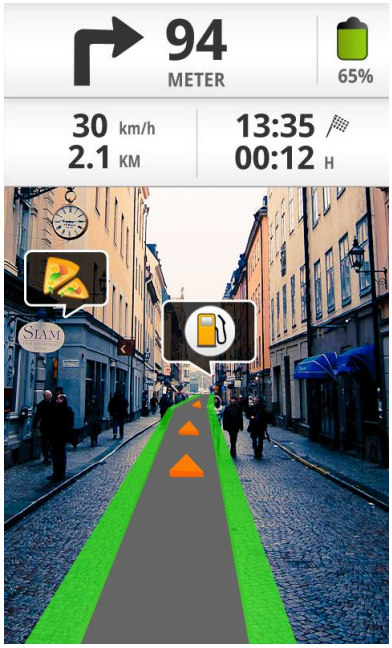

Figura 2.3: Percurso de GPS sobreposto no trajeto usando Realidade Aumentada.

4. Realidade Aumentada em Museus [11]

   A tecnologia utilizada em museus a partir de *Apps* para dispositivos móveis, onde as exposições podem ”ganhar vida” através de animações ou com partilha de informações históricas.

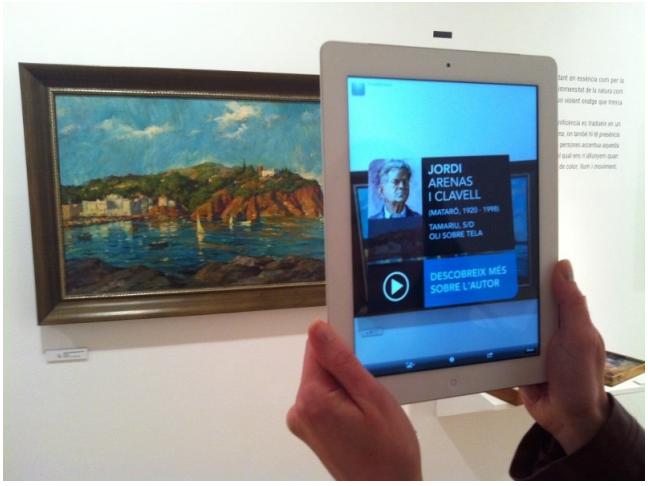

Figura 2.4: Informação sobre exposição aumentada utilizando a tecnologia de RA no dispositivo móvel.

5. Realidade Aumentada na educação[9]

   Aplicações para dispositivos móveis a utilizar nas salas de aula onde os estudantes podem fazer *scan* a páginas dos livros para aparecer vídeos a explicar os problemas.

6. Realidade Aumentada em videojogos[8]

   Videojogos com elementos digitais a serem misturados no ambiente real do utilizador.

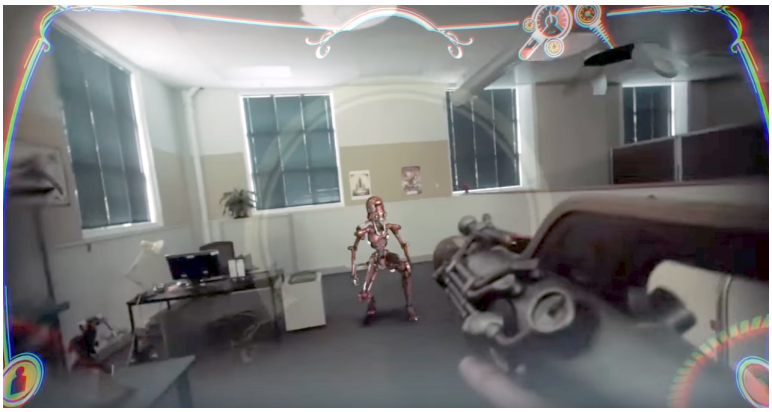

Figura 2.5: Videojogo em Realidade Aumentada: *Magic Leap*.

2.3 Videojogos em Realidade Aumentada

A utilização desta tecnologia no desenvolvimento de videojogos permite que os jogadores experimentem jogos digitais num ambiente real, mostrando uma interação completamente diferente da tradicional. São vários os caminhos que os criadores tomam quando decidem aplicar esta tecnologia nas suas criações, levando à existência de um leque enorme de jogos em Realidade Aumentada, muitos deles bastante únicos, inovadores e criativos:

2.3.1 Warp Runner

Warp Runner [16] é um jogo inovador de puzzle em Realidade Aumentada. A personagem é presa no mundo real e o jogador tem que o ajudar a escapar, deformando o ambiente real. Através de toques no ecrã do dispositivo móvel o jogador pode modificar o terreno criando elevações ou declives e mover a personagem para completar os puzzles. O interessante é que apesar de nada acontecer no ambiente real, essas modificações do terreno originam virtualmente deformações desse ambiente.

Figura 2.6: Videojogo em Realidade Aumentada: *Warp Runner*.

2.3.2 Ingress

Ingress [7] é um videojogo de Realidade Aumentada baseado em localização geográfica, produzido pela *Niantic Labs*. O mundo é dividido em duas fações que controlam portais e o jogador tem que escolher um lado e ajudar a fação a obter a vitória. A inovação neste jogo é que o jogador tem que fisicamente se deslocar pelo mundo real, pois estes portais só aparecem em locais públicos e monumentos conhecidos, tendo em conta a sua localização geográfica.

Figura 2.7: Videojogo em Realidade Aumentada: *Ingress*.

2.3.3 Drakerz-Confrontation

Durante décadas, pessoas por todo o mundo jogavam jogos de cartas colecionáveis como *Magic: The Gathering*, *Yu-Gi-Oh!* e *Pokémon*. A *Peoleo Entertainment* mostra um jogo que usa Realidade Aumentada para dar vida a cartas colecionáveis originando modelos 3D de monstros designados *Drakos*. *Drakerz-Confrontation* [4] usa uma *webcam* de um computador e os monstros 3D saltam das cartas para entrarem em batalhas épicas. Neste caso as cartas são os marcadores de RA e cada uma tem o seu próprio monstro e animação adequada para a batalha.

Figura 2.8: Videojogo em Realidade Aumentada: *Drakerz*.

2.3.4 PulzAR

PulzAR [5] é um jogo de puzzle também inovador. Disponível na *Playstation Store* para a *PS Vita*, este jogo usa a câmara da consola para criar puzzles dinâmicos. Com o objetivo de apontar um laser por vários obstáculos de maneira a ser direcionado para o espaço e destruir um asteróide, o jogador tem que usar cartas que originam esses obstáculos para redirecionar o laser. Também aqui as cartas são os marcadores de RA.

Figura 2.9: Videojogo em Realidade Aumentada: *PulzAR*.

Capítulo 3

Identificação do Problema e Definição de Soluções

3.1 Introdução

Neste capítulo faz-se um levantamento de jogos em Realidade Aumentada onde se apresenta uma análise de *gameplay* e modos de interação para avaliar os problemas e soluções no desenvolvimento.

3.2 Identificação do problema

Todos os videojogos em Realidade Aumentada, e tendo como referência os apresentados no capitulo de Estado da Arte 2, têm as suas características próprias e inovações que os diferem uns dos outros. Analisando *Warp Runner* 2.3.1, *Drakerz-Confrontation* 2.3.3 e *PulzAR* 2.3.4 podemos verificar que todos eles têm um estilo de *gameplay* e de interação com o jogador diferentes: Em *Warp Runner* o jogador controla uma personagem por vários níveis de puzzles, tudo digital aumentado na realidade, a partir de toques no ecrã do dispositivo móvel; em *Drakerz* e *PulzAR* o jogador controla e manipula cartas para no primeiro derrotar os monstros e no segundo superar os puzzles. Porém estes videojogos têm um aspeto fundamental que é comum: os marcadores, que no caso de *Drakerz* e *PulzAR* são as próprias cartas de jogo.
Apesar de ser uma tecnologia que quando aplicada na área dos videojogos os consegue inovar e diferenciar seja no *gameplay* ou no modo de interação com o jogador, pondo de parte o tradicional teclado de computador ou controladores de consolas, a utilização de marcadores específicos para cada videojogo tem uma particularidade que pode constituir um problema ou obstáculo. Um utilizador pode ter no seu dispositivo móvel uma quantidade grande de videojogos para poder jogar em qualquer altura, porém se grande parte desses videojogos forem em Realidade Aumentada a pessoa teria que andar com os marcadores físicos de cada um dos jogos consigo.
O principal foco deste projeto será a análise das implicações deste obstáculo para se definir uma solução, apresentada na próxima secção 3.3, e desenvolver um videojogo para testar a solução proposta.

3.3 Definição de soluções

Confrontando o problema referido na secção anterior 3.2 diretamente, a ideia seria o desenvolvimento de um videojogo em Realidade Aumentada que não precise de marcadores físicos. Físicos pois na verdade qualquer aplicação e videojogo desenvolvido nesta tecnologia precisa de marcadores para se poder adicionar, posicionar e orientar estrategicamente os elementos virtuais a serem aumentados, mas não precisam de ser necessariamente marcadores físicos e impressos pois a informação dos marcadores que permite aumentar os elementos virtuais obtida pela câmara do dispositivo onde a aplicação esteja a correr pode também ser obtida (com um grau de complexidade maior) analisando o cenário e fazendo o reconhecimento de objetos.
É nesta solução que o projeto se baseia. Para uma aplicação prática, o videojogo de teste será então desenvolvido no motor de jogo Unity recorrendo à biblioteca do OpenCV para criar o algoritmo de reconhecimento do cenário e estabelecer marcadores de Realidade Aumentada em tempo real utilizando figuras geométricas conhecidas como quadrados, retângulos, triângulos, obtidas com a deteção do objetos do cenário com as formas respetivas. Todo este desenvolvimento é documentado no capítulo Implementação 5.

Capítulo 4

Tecnologias e Ferramentas Utilizadas

4.1 Introdução

Este capítulo refere as tecnologias e ferramentas utilizadas no desenvolvimento do projeto. Destaca-se o motor de jogo Unity e a biblioteca de funções de programação OpenCV.

4.2 Unity

Também conhecido como Unity 3D, é um dos mais populares motores de jogo da atualidade, criado pela *Unity Technologies*. Tem atualmente três versões disponíveis [3]: *Unity Personal*, *Unity Plus* e *Unity Pro*. A primeira versão é grátis e apesar de ter pequenas limitações continua a ser legível para usos comerciais, sendo esta a versão utilizada para o desenvolvimento do projeto.
Caracterizado por ter um poderoso motor de renderização integrado com um conjunto bastante completo de ferramentas intuitivas para criação não só de jogos mas de qualquer conteúdo interativo, seja 2D ou 3D. A escolha deste motor para a criação do videojogo de teste deste projeto deve-se maioritariamente pela intuitiva manipulação de modelos, criação e utilização de *scripts*, facilidade na programação de eventos e sobretudo pela simplicidade do porte dos projetos para dispositivos móveis. A importância destes fatores deve-se ao facto de agilizarem o processo de desenvolvimento do *gameplay* do protótipo para haver um maior foco na criação dos algoritmos que originam as soluções para os problemas identificados no desenvolvimento em Realidade Aumentada. O Unity serviu também para integrar a tecnologia do OpenCV, referenciada na próxima secção 4.3.

4.3 OpenCV

OpenCV (Open Source Computer Vision Library) é uma biblioteca de funções de programação para o desenvolvimento de aplicações na área de Visão Computacional. Sendo de código aberto é adequada tanto para o uso académico como comercial. Possui mais de 2500 algoritmos optimizados [10] que podem ser utilizados em várias situações como:

- deteção e reconhecimento de faces.
- identificação de objetos.
- classificação de ações de humanos em vídeos.
- *tracking* de movimentos de objetos.
- extração de modelos 3D de objetos.
- processamento de imagens.
- extração de imagens semelhantes numa base de dados.
- remoção de olhos vermelhos de fotografias tiradas com flash.
- seguir o movimento dos olhos.

Para o propósito deste projeto a utilização do OpenCV teve um maior foco nos algoritmos de reconhecimento de cenários com processamento de imagem para a deteção de objetos, com o objetivo de se estabelecerem marcadores que serão sobrepostos com Realidade Aumentada, tudo em tempo real e que será explicado no próximo capítulo: Implementação 5.

Capítulo 5

Implementação

5.1 Introdução

Este capítulo apresenta o videojogo em Realidade Aumentada a desenvolver e todas as fases ordenadas por ordem de ocorrência presentes na sua implementação. Para o desenvolvimento dos algoritmos a linguagem de programação escolhida é o `C#`, sendo uma das linguagens suportadas pelo motor de jogo Unity.

5.2 Integração da biblioteca (OpenCV)

O Unity possui uma funcionalidade de importação e exportação de *packages* que é nada mais que uma maneira prática de partilhar e re-utilizar projetos e coleções de *assets* do motor de jogo. Possui uma *Asset Store* onde qualquer artista ou programador pode partilhar grátis ou vender as suas criações que podem ser utilizadas em qualquer projeto, respeitando as respetivas licenças de uso.
Como o objetivo do projeto não se foca em como fazer a integração da biblioteca OpenCV no Unity mas sim saber utiliza-la para a concretização das soluções propostas, foi utilizado um Unity *package* próprio para a integração da biblioteca no motor de jogo: *OpenCVForUnity* da *Enox Software* [15]. Funciona para ambas as versões grátis e paga do Unity, tem suporte multi-plataforma tanto para *Windows*, *Mac*, *Linux*, *Android*, *iOS* e *Windows Phone*, e sendo um clone do OpenCV Java tem suporte para utilização da mesma API.

5.3 Acesso à câmara e obtenção de *frames*

Quando são retiradas imagens digitais do mundo real a partir de câmaras, o que o ser humano vê são apenas imagens, porém quando transferidas para dispositivos digitais o que é guardado é matrizes que contêm os valores de intensidade para cada pixel.
`Mat` é basicamente uma classe que é dividida em duas partes. A primeira: o cabeçalho da matriz (que contém informação como o tamanho da matriz, o método usado para armazenamento dos dados, o endereço) e a segunda: o ponteiro para a matriz que contém os valores de cada pixel da imagem. O OpenCV sendo uma biblioteca de processamento de imagens contém uma grande coleção de funções para tal, e por essa razão passar imagens para essas funções a partir de parâmetros é uma prática comum. Ora se os algoritmos de processamento de imagem tendem a ser computacionalmente pesados, a última coisa a se querer fazer é diminuir a velocidade dos programas fazendo cópias desnecessárias de imagens gigantes.
O OpenCV resolve esta questão pois cada objeto `Mat` tem o seu próprio cabeçalho mas a matriz pode ser partilhada entre duas instâncias do objeto por terem os ponteiros da matriz a apontar para o mesmo endereço. Além disso os operadores de cópia apenas copiam os cabeçalhos e os ponteiros para as matrizes, não para a data propriamente dita. Desta maneira as imagens

dos *frames* retirados diretamente da câmara (aqui designadas `WebCamTextures`) são convertidas primeiramente para `Mat` antes de serem processadas. O excerto de código 5.1 apresenta esta primeira parte descrita.

```
1  WebCamTexture webCamTexture;
2  Mat rgbaMat;
3  //...
4  public Mat GetMat ()
5  {
6    //...
7    Utils.webCamTextureToMat (webCamTexture, rgbaMat);
8    //...
9    return rgbaMat;
10 }
```

Listing 5.1: *WebCamTexture to Mat*

A classe *OpenCVForUnity.Utils* possui vários métodos utilitários de conversões, e para este caso utilizamos o `Utils.WebCamTextureToMat` que converte imagens do Unity (`WebCamTexture`) para o tipo `Mat` utilizado pelo OpenCV.

5.4 Processamento dos frames

Com o excerto de código 5.1 a correr num ciclo, os frames da câmara são obtidos e devem então ser processados em tempo real para se poder proceder ao reconhecimento do cenário que levará à deteção de possíveis objetos que servirão de marcadores para se poder aplicar a tecnologia de Realidade Aumentada.
Para se fazer o reconhecimento do cenário é preciso definir o que se quer reconhecer, isto é, que objetos ou tipos de objetos vão ser detetados no processamento dos frames. Os videojogos e aplicações em Realidade Aumentada tipicamente utilizam marcadores com características especificas e predefinidas, porém indo de encontro à solução pretendida de o utilizador poder encontrar marcadores em tempo-real, o algoritmo de deteção irá focar-se na procura de simples formas geométricas conhecidas que usualmente se encontram em grande escala em qualquer cenário.
Como o OpenCV é uma biblioteca com uma grande variedade de algoritmos otimizados na área de Visão Computacional são várias as formas de se processar uma imagem para procurar formas geométricas, retirando informações como diferenças de cor na imagem, deteção de pontos, alteração na continuidade de segmentos de recta. Para o desenvolvimento do algoritmo de deteção de formas geométricas foi utilizado como referência o blog [13] criado por Adrian Rosebrock, doutorado em Ciência da Computação, onde mostra o seu trabalho em *Image Search Engines* com vários tutoriais práticos utilizando a linguagem de programação *Python* e OpenCV.
Foi então criado um script de Unity próprio para representar o algoritmo de deteção de formas geométricas, designado `ShapeDetector`. Após a obtenção do frame a imagem sofre primeiramente um pré-processamento antes da parte da deteção, que é dividido em 5 fases:

- GrayScale.
- Blur.
- Canny.
- Dilate.
- Find Contours.

No script `ShapeDetector` foi criado um método `Mat Process(Mat source)` para fazer o processamento dos frames que recebe como parâmetro a imagem convertida para tipo `Mat` Para o pré-processamento das imagens são utilizadas funções do módulo `Imgproc`, um módulo do OpenCV específico para o processamento e manipulação de imagens. Estas fases de pré-processamento são demonstradas e explicadas nas próximas secções.

5.4.1 *GrayScale*

Para se poder detetar as formas geométricas a imagem tem que ser transformada de modo a ficar binária, de maneira a se poder fazer a distinção entre os objetos a serem detetados (formas geométricas) com o background que contém tudo o resto não relevante. É aqui que entra esta primeira fase do pré-processamento onde se transforma a imagem a cores para uma imagem em tons de cinza. O excerto de código 5.2 demonstra esta primeira fase.

```
1 Mat Process(Mat srcMat)
2 {
3   //...
4     //GrayScale
5     Imgproc.cvtColor(srcMat, srcMat, Imgproc.COLOR_BGR2GRAY);
6 }
```

Listing 5.2: *GrayScale*

O método `cvtColor` converte a imagem de um espaço de cores para outro. Tem como parâmetros a `Mat` de entrada, a `Mat` de saída, e um código `Integer` para especificar o tipo de conversão. Como se quer modificar a imagem diretamente, o parâmetro `Mat` de entrada e saída é o mesmo.

5.4.2 *Blur*

A seguinte fase no pré-processamento é o `Blur`. Tem como objetivo suavizar a imagem para reduzir o ruído, sendo um método típico no pré-processamento de imagens para melhorar o resultado final. O excerto de código 5.3 demonstra esta segunda fase.

```
1  Mat Process(Mat srcMat)
2  {
3    //...
4
5      //GrayScale
6      Imgproc.cvtColor(srcMat, srcMat, Imgproc.COLOR_BGR2GRAY);
7
8      //Blur
9      Imgproc.GaussianBlur(srcMat, srcMat, new Size(7, 7), 0);
10 }
```

Listing 5.3: *Blur*

O método `GaussianBlur` desfoca a imagem usando um *Gaussian filter*. Tem como parâmetros a `Mat` de entrada, de saída e o *Gaussian kernel size*. O tamanho do *kernel* não deve ser muito alto para não eliminar características mais pequenas da imagem, nem muito baixo pois não eliminaria o possível ruído existente.

5.4.3 *Canny*

O próximo passo do pré-processamento passa por identificar arestas na imagem utilizando o algoritmo desenvolvido por John F.Canny [1]. O resultado final após a aplicação deste algoritmo irá permitir posteriormente encontrar contornos na imagem para detetar e classificar figuras geométricas. O excerto de código 5.4 demonstra a programação desta fase.

```
1  Mat Process(Mat srcMat)
2  {
3    //...
4
5      //GrayScale
6      Imgproc.cvtColor(srcMat, srcMat, Imgproc.COLOR_BGR2GRAY);
7
8      //Blur
9      Imgproc.GaussianBlur(srcMat, srcMat, new Size(7, 7), 0);
10
11     //Canny
12     double otsu_thresh_val = Imgproc.threshold(srcMat, cannyThresholdMat, 0,
           255, Imgproc.THRESH_OTSU);
13     double high_thresh_val = otsu_thresh_val;
14     double lower_thresh_val = otsu_thresh_val * 0.5f;
15     Imgproc.Canny(srcMat, srcMat, lower_thresh_val, high_thresh_val);
16 }
```

Listing 5.4: *Canny*

A linha 15 mostra a função (`Canny`) do módulo `Imgproc` a ser usada para a deteção das arestas. Tem novamente como parâmetros as `Mat` de entrada e saída e dois valores de *threshold* para limitarem quais arestas a considerar. Para calcular estes valores automaticamente dependendo da imagem de entrada, foi utilizado o método de Otsu, com o mesmo nome do criador Nobuyuki

Otsu [2]. Este método envolve a iteração de todos os possíveis valores de limite e calcular uma medida de afastamento dos níveis de cada pixel para cada lado do limite, isto é, para saber que pixel pertence ao primeiro plano e ao background. O objetivo é encontrar o valor de limite onde a soma desta medida é mínima.

5.4.4 *Dilate*

Esta fase serve para intensificar os resultados da fase anterior, isto é, após a deteção das arestas na imagem (que sendo binária vão ser identificadas com cor branca, em contraste ao background de cor preta) é aplicado um algoritmo de dilatação demonstrado no excerto de código 5.5.

```
1  Mat Process(Mat srcMat)
2  {
3    //...
4
5      //GrayScale
6      Imgproc.cvtColor(srcMat, srcMat, Imgproc.COLOR_BGR2GRAY);
7
8      //Blur
9      Imgproc.GaussianBlur(srcMat, srcMat, new Size(7, 7), 0);
10
11     //Canny
12     double otsu_thresh_val = Imgproc.threshold(srcMat, cannyThresholdMat, 0,
           255, Imgproc.THRESH_OTSU);
13     double high_thresh_val = otsu_thresh_val;
14     double lower_thresh_val = otsu_thresh_val * 0.5f;
15     Imgproc.Canny(srcMat, srcMat, lower_thresh_val, high_thresh_val);
16
17     //Dilate
18     Imgproc.dilate(srcMat, srcMat, dilateKernel);
19 }
```

Listing 5.5: *Dilate*

A função `Dilate` do módulo `Imgproc` recebe como parâmetros a `Mat` de entrada, de saída e um valor para configurar a dilatação. Este pré-processamento faz com que as regiões brilhantes (arestas detetadas na fase anterior) "cresçam", daí o nome dilatação, para se obter melhores resultados na fase final do pré-processamento: *Find Countours*.

5.4.5 *Find Contours*

Nesta ultima fase faz-se a deteção de contornos, usando as arestas identificadas em 5.4.3 e dilatadas em 5.4.4. A demonstração da função para esta fase pode ser encontrada no excerto de código 5.6.

```
Mat Process(Mat srcMat)
{
  //...

    //GrayScale
    Imgproc.cvtColor(srcMat, srcMat, Imgproc.COLOR_BGR2GRAY);

    //Blur
    Imgproc.GaussianBlur(srcMat, srcMat, new Size(7, 7), 0);

    //Canny
    double otsu_thresh_val = Imgproc.threshold(srcMat, cannyThresholdMat, 0,
        255, Imgproc.THRESH_OTSU);
    double high_thresh_val = otsu_thresh_val;
    double lower_thresh_val = otsu_thresh_val * 0.5f;
    Imgproc.Canny(srcMat, srcMat, lower_thresh_val, high_thresh_val);

    //Dilate
    Imgproc.dilate(srcMat, srcMat, dilateKernel);

    //Find Contours
    List<MatOfPoint> contours = new List<MatOfPoint>();
    Mat hierarchy = new Mat();
    Imgproc.findContours(srcMat, contours, hierarchy, Imgproc.RETR_EXTERNAL,
        Imgproc.CHAIN_APPROX_SIMPLE);
}
```

Listing 5.6: *Find Contours*

A função `findContours` do módulo `Imgproc` retorna os contornos de uma imagem binária, que são uma ferramenta útil na análise de formas e deteção e reconhecimento de objetos. Os contornos são guardados numa lista de `MatOfPoint`, uma sub-classe de `Mat` no OpenCV. O parâmetro `Imgproc.RETR_EXTERNAL` serve para indicar que queremos apenas os contornos exteriores extremos e o parâmetro `Imgproc.CHAIN_APPROX_SIMPLE` serve para indicar o método de aproximação dos contornos, neste caso comprime os segmentos horizontais, verticais e diagonais e deixa apenas os seus pontos finais, onde por exemplo um contorno rectangular irá ser codificado apenas com 4 pontos.
É então com estes contornos que procedemos à deteção das figuras geométricas.

5.4.6 Deteção de figuras geométricas

Neste momento com os contornos das imagens obtidos procedemos à análise dos mesmos para se decidir se existem figuras geométricas em cada frame. Os contornos não são nada mais do que curvas poligonais, por isso antes de se fazer a análise teve que se usar uma função para aproximar estas curvas a outras com menos vértices. Esta função está presente no módulo `Imgproc` designada `approxPolyDP`. O excerto de código 5.7 demonstra a aplicação desta função.

```
1  Mat Process(Mat srcMat)
2  {
3    //...
4
5      //Find Contours
6      List<MatOfPoint> contours = new List<MatOfPoint>();
7      Mat hierarchy = new Mat();
8      Imgproc.findContours(srcMat, contours, hierarchy, Imgproc.RETR_EXTERNAL,
           Imgproc.CHAIN_APPROX_SIMPLE);
9
10     for (int i = 0; i < contours.Count; i++)
11     {
12         //Calc the approx
13         MatOfPoint2f approx = new MatOfPoint2f();
14         Imgproc.approxPolyDP(new MatOfPoint2f(contours[i].toArray()), approx,
               Imgproc.arcLength(new MatOfPoint2f(contours[i].toArray()), true) *
               0.02f, true);
15
16         //Get the area, isConvex, height
17         double area = Imgproc.contourArea(approx);
18         bool isConvex = Imgproc.isContourConvex(new MatOfPoint(approx.toArray()
               ));
19         double height = approx.size().height;
20
21         //Skip non-convex
22         if (isConvex)
23         {
24             contourData.Add(new ContourData(approx, approx.toList(), i, area,
                   height));
25             if (area > maxArea)
26                 maxArea = area;
27         }
28     }
29 }
```

Listing 5.7: *ApproxPolyDP*

Percorrendo todos os contornos num ciclo (linha 10), é calculada a aproximação do contorno na linha 14. A função recebe 4 parâmetros: um `MatOfPoint2f` que contém a curva inicial, um `MatOfPoint2f` onde se vai guardar a curva aproximada, um `double` epsilon que especifica a precisão da aproximação, isto é, a distância máxima entre a curva original e a aproximada, e

um `bool` que determina se a aproximação é fechada, ou seja, se o primeiro e último vértice são ligados.
Além da curva aproximada, também é calculada a sua área (linha 17), se é convexa (linha 18) e o número de vértices (linha 19), para se poder filtrar curvas que não se encaixem na categoria de figuras geométricas conhecidas. Na linha 22 fazemos a filtragem e pomos de parte figuras não convexas e determinamos o contorno com maior área para podemos posteriormente eliminar contornos com áreas reduzidas. Todos os contornos não eliminados nesta parte são guardados numa lista de `ContourData`, uma classe criada para este efeito.
Finalmente para se fazer a deteção das figuras geométricas é feita a iteração dessa lista, no excerto de código 5.8.

```
1  Mat Process(Mat srcMat)
2  {
3    //...
4
5    foreach (ContourData data in contourData)
6    {
7      //Skip small
8      if (maxArea < 5000)
9        break;
10     if (data.area < areaRatio * maxArea)
11       continue;
12
13     // Triangles
14     if (data.height == 3)
15     {
16       setLabel(imgMatFinal, "TRI", new MatOfPoint(data.approx.toArray()));
17       Imgproc.drawContours(outputMat, contours, data.index, new Scalar(255, 0,
             0, 255), 3);
18     }
19
20     //Rectangles, Pentagons, Hexagons
21     else if (data.height >= 4 && data.height <= 6)
22     {
23       // Number of vertices of polygonal curve
24       int v = (int)data.height;
25       if(v == 4)
26       {
27         OpenCVForUnity.Rect r = Imgproc.boundingRect(new MatOfPoint(data.approx
               .toArray()));
28         float aspectRatio = r.width / (float)r.height;
29         if (aspectRatio >= 0.95 && aspectRatio <= 1.05)
30         {
31           setLabel(matFinal, "SQUARE", new MatOfPoint(data.approx.toArray()));
32           Imgproc.drawContours(outputMat, contours, data.index, new Scalar(0,
                 255, 0, 255), 3);
33         }
```

```
34          else
35          {
36            setLabel(matFinal, "RECT", new MatOfPoint(data.approx.toArray()));
37            Imgproc.drawContours(outputMat, contours, data.index, new Scalar(0,
                  0, 255, 255), 3);
38          }
39        }
40        else if(v == 5)
41        {
42          setLabel(matFinal, "PENTA", new MatOfPoint(data.approx.toArray()));
43          Imgproc.drawContours(outputMat, contours, data.index, new Scalar(255,
                255, 255, 255), 3);
44        }
45        else if(v == 6)
46        {
47          setLabel(matFinal, "HEXA", new MatOfPoint(data.approx.toArray()))
48          Imgproc.drawContours(outputMat, contours, data.index, new Scalar(0, 0,
                0, 255), 3);
49        }
50      }
51      // circles
52      else
53      {
54        //setLabel(matFinal, "CIRC", new MatOfPoint(approx.toArray()));
55        Imgproc.drawContours(outputMat, contours, data.index, new Scalar(255,
              255, 0, 255), 3);
56      }
57    }
58  }
```

Listing 5.8: *Deteção de Figuras Geométricas*

Na linha 7 é feita a exclusão de figuras com áreas reduzidas pois não servirão de bons marcadores para Realidade Aumentada. O parâmetro `areaRatio` define a área mínima em relação ao contorno da imagem com área máxima, isto é, se o valor tiver como exemplo 0.7 então só serão aceites figuras com áreas superiores a pelo menos setenta por cento da área máxima.
Na linha 14 começa a deteção de figuras geométricas. Caso o número de vértices do contorno seja 3 então podemos classificar a figura como sendo um triângulo. Na linha 21 podemos classificar as figuras como sendo retângulos, pentágonos ou hexágonos, dependendo se o número de vértices seja quatro, cinco ou seis.
Para fazer a distinção entre quadrados e retângulos é calculado o rácio entre a altura e largura. Caso o valor deste seja aproximadamente um (1) então será classificado como quadrado, caso contrário como retângulo.
Por exclusão de partes, e devido a termos eliminado figuras não convexas, na linha 53 podemos classificar o contorno como sendo um circulo.
Em todos os casos são desenhados os contornos da figura com cores diferentes dependendo da classificação e feita uma legenda, que será sobreposta no frame original, em tempo real.

5.5 Estabelecimento de marcadores

Com as figuras geométricas detetadas a partir do reconhecimento de objetos no cenário falta proceder com a fase de utilização das mesmas para estabelecer marcadores de Realidade Aumentada em tempo real. Para isso temos que primeiramente entender como funciona uma cena de jogo normal no Unity. O desenvolvimento de videojogos em Unity funciona com a utilização de *GameObjects*, uma classe base que representam qualquer conteúdo do jogo como personagens, objetos, efeitos, cenários. Um dos *GameObjects* indispensáveis numa cena é a câmara principal que quando controlada pelo computador ou jogador deve mostrar as cenas de jogo. No caso de videojogos em realidade aumentada o que é mostrado ao jogador é o cenário real capturado pela câmara física do dispositivo móvel ou pela *webcam* de um computador juntamente com os *GameObjects* virtuais aumentados.
Para posicionar corretamente os objetos virtuais a serem aumentados a câmara deve capturar o marcador e perceber em que orientação este se encontra em relação à mesma. Num exemplo prático: caso a câmara capture o marcador numa vista perpendicular de cima para baixo e o objeto virtual a ser aumentado seja uma personagem que se encontre de pé, a câmara poderá só apanhar a cabeça da personagem. Quando a câmara (controlada pelo utilizador) se inclinar para um dos lados, o programa terá que perceber que a câmara já não se encontra perpendicular ao marcador e deverá ter em conta a nova perspetiva para orientar a personagem e mostrar corretamente o tronco e pés desta. Para este efeito é utilizada uma função do módulo `Imgproc` do OpenCV designada `warpPerspective` que aplica uma transformação de perspetiva. A figura 5.1 ilustra um exemplo desta transformação.

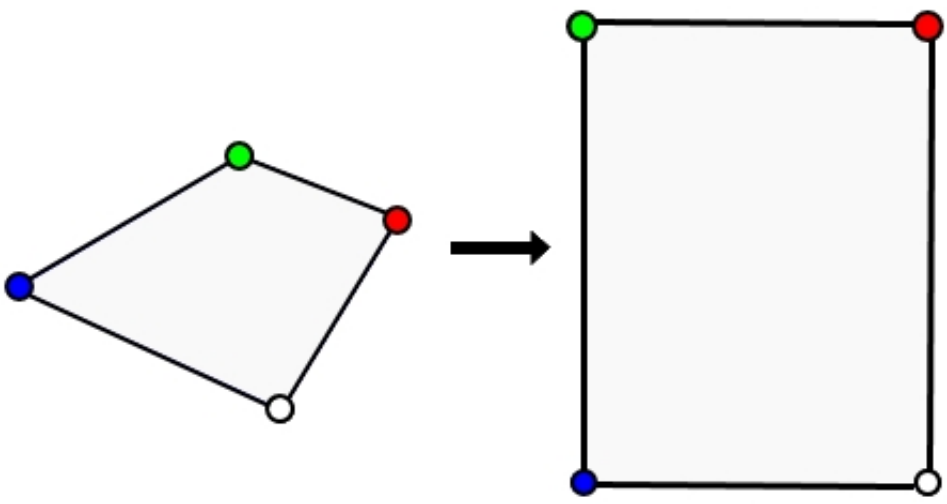

Figura 5.1: Transformação de Perspetiva.

Fazendo uma analogia onde a figura representa um marcador detetado pelo algoritmo em tempo real, quando a câmara de jogo está orientada de maneira a capturar o marcador com a perspetiva ilustrada na figura geométrica da esquerda, o algoritmo calcula qual a transformação necessária para que a perspetiva fosse a da figura geométrica da direita. Utiliza-se assim essa transformação para posicionar e orientar os elementos virtuais corretamente.

5.6 Videojogo de teste - ARSlidingPuzzle

Para uma aplicação prática de todos os algoritmos demonstrados neste capitulo e procurando finalizar a solução proposta em 3.3 foi idealizado um videojogo para dispositivos móveis, designado *ARSlidingPuzzle*, uma adaptação dos tradicionais jogos de quebra-cabeças onde uma imagem é dividida em várias secções e estas são baralhadas com o objetivo do utilizador voltar à imagem inicial. Usando todo o algoritmo apresentado neste capítulo, desde o pré-processamento dos frames até ao estabelecimento dos marcadores em tempo real, o jogador utiliza a câmara do dispositivo móvel para fazer o reconhecimento do cenário e escolher um objeto que apresente uma forma geométrica de maneira ao videojogo reconhecer o objeto, capturar uma fotografia, dividi-la em secções, baralha-la e começar o quebra-cabeças.
A partir do reconhecimento do cenário, o videojogo ilustra os possíveis marcadores desenhando um contorno verde caso detete uma figura geométrica (figura 5) e o jogador clica no objeto para o escolher como imagem a utilizar no quebra-cabeças (figura 5).

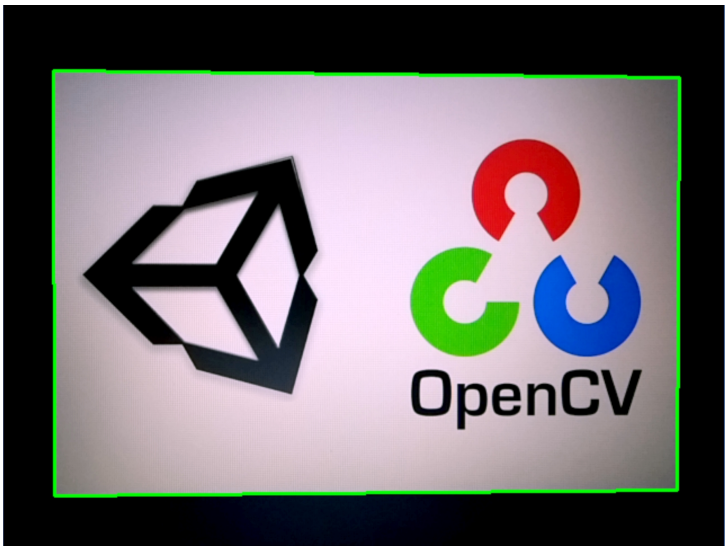

Figura 5.2: Marcador de RA ilustrado pelo contorno verde.

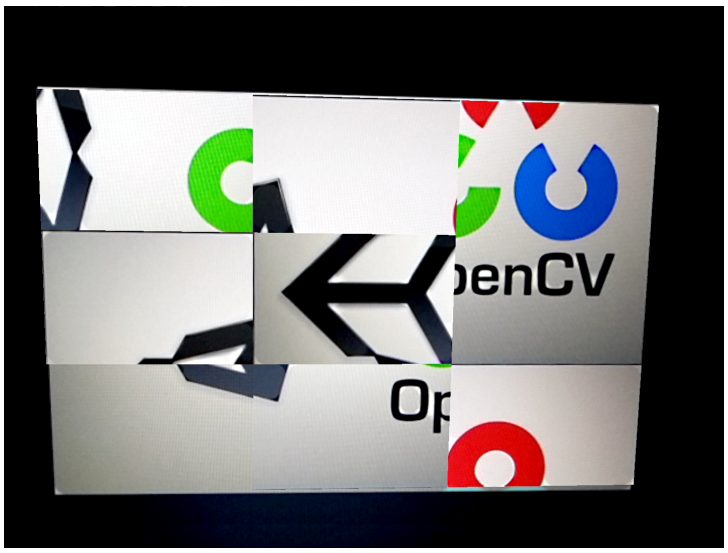

Figura 5.3: Imagem de quebra-cabeças dividida e baralhada.

Capítulo 6

Conclusão e Trabalho Futuro

Pela leitura do presente relatório pode-se concluir que a aplicação prática da solução proposta para a resolução do problema/obstáculo apresentado foi concluída com êxito. Toda a análise, pesquisa e implementação dos algoritmos demonstrados contribuíram para o desenvolvimento do videojogo de teste que serviu satisfatoriamente para a demonstração de marcadores de Realidade Aumentada estabelecidos em tempo real como solução para o problema identificado. Também se pode concluir que a biblioteca de código aberto do OpenCV possui algoritmos otimizados que podem ser postos em prática no desenvolvimento de aplicações e videojogos em Realidade Aumentada, nomeadamente no reconhecimento de cenários para criação de marcadores em tempo real. É de referir também que o motor de jogo Unity conseguiu integrar facilmente esta biblioteca e produzir resultados bastante satisfatórios.
Obviamente os resultados obtidos não limitam o potencial da utilização destas ferramentas na criação de videojogos com Realidade Aumentada. São várias as melhorias que ainda podem ser feitas, tanto do lado mais técnico e teórico de algoritmia no reconhecimento de cenários como do lado mais prático do desenvolvimento de *gameplay*, modos de interação e mecânicas inovadoras, como por exemplo um reconhecimento de objetos e cenários mais complexos e desenvolvimento de mecânicas especificas para hardware como óculos de Realidade Aumentada e Virtual.

Bibliografia

[1] Canny Edge Detection. http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_canny/py_canny.html. [Online; acedido a 10 de Setembro de 2016]. 16

[2] Otsu Thresholding. http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html. 17

[3] Clive Downie. EVOLUTION OF OUR PRODUCTS AND PRICING. https://blogs.unity3d.com/2016/06/16/evolution-of-our-products-and-pricing/, 2016. [Online; acedido a 4 de Setembro de 2016]. 11

[4] Peoleo Entertainment. Drakerz - Augmented Reality Video Game. http://www.drakerz.com/. [Online; acedido a 25 de Março de 2016]. 7

[5] Sony Computer Entertainment. PulzAR - Augmented Reality Video Game. http://www.ign.com/games/pulsar/vita-123205. [Online; acedido a 25 de Março de 2016]. 8

[6] Andrew Freedman. Weather Channel turns to augmented reality for 2016 hurricane season coverage. http://mashable.com/2016/06/28/weather-channel-hurricane-season-surge/#2LAtHSecRiqf, 2016. [Online; acedido a 3 de Agosto de 2016]. 4

[7] Niantic Inc. Ingress - Augmented Reality Video Game. https://www.ingress.com/. [Online; acedido a 25 de Março de 2016]. 7

[8] Patrick Klepek. If This Represents The Future Of Augmented Reality Video Games, I'm In. http://kotaku.com/if-this-represents-the-future-of-augmented-reality-vide-1692619674, 2015. [Online; acedido a 17 de Maio de 2016]. 6

[9] Todd Nesloney. Augmented Reality Brings New Dimensions to Learning. http://www.edutopia.org/blog/augmented-reality-new-dimensions-learning-drew-minock, 2013. [Online; acedido a 20 de Fevereiro de 2016]. 5

[10] OpenCV. About. http://opencv.org/about.html. [Online; acedido a 4 de Setembro de 2016]. 12

[11] Alex Radsky. Where History Comes Alive: Augmented Reality in Museums. https://medium.com/synapse/where-history-comes-alive-augmented-reality-in-museums-64a81825b799#.15j7wov9j, 2015. [Online; acedido a 20 de Fevereiro de 2016]. 5

[12] Paul Ridden. IKEA catalog uses augmented reality to give a virtual preview of furniture in a room. http://newatlas.com/ikea-augmented-reality-catalog-app/28703/, 2013. [Online; acedido a 15 de Fevereiro de 2016]. 4

[13] Adrian Rosebrock. pyimagesearch - About. http://www.pyimagesearch.com/about/. [Online; acedido a 5 de Setembro de 2016]. 14

[14] Gloria Sin. Bring augmented reality to your road trip with Wikitude Drive app for Android. http://www.zdnet.com/article/bring-augmented-reality-to-your-road-trip-with-wikitude-drive-app-for-android/, 2011. [Online; acedido a 15 de Fevereiro de 2016]. 5

[15] Enox Software. OpenCV for Unity. https://enoxsoftware.com/opencvforunity/. [Online; acedido a 5 de Setembro de 2016]. 13

[16] Visionaries777. Warp Runner - Augmented Reality Video Game. http://www.visionaries777.com/warprunner/. [Online; acedido a 25 de Março de 2016]. 6