INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Tradução em Tempo Real para Aplicações Gráficas

NUNO RAMALHO FARIA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de Tradução em Tempo Real para Aplicações Gráficas

Nuno Ramalho Faria

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

Orientador: Doutor João Paulo Pereira

Aos meus pais e amigos por todo o apoio que sempre me deram, e ao Doutor João Pereira que sempre me orientou prontamente durante todo o desenvolvimento desta dissertação.

Resumo

Num mundo perfeito, qualquer pessoa, de qualquer parte do mundo, seria capaz de falar uma língua em comum com qualquer outro indivíduo, de qualquer outro ponto do globo. A globalização do ensino de inglês tornou-a nesta mesma língua comum, no entanto, existem (e continuarão a existir) casos em que a língua utilizada não é compreensível para um público global.

Para resolver este problema foi idealizada, e desenvolvida, uma potencial solução que tenta aproximar um pouco o trabalho de diferentes culturas ao permitir que aplicações cuja interface se encontre numa língua incompreensível, possam ser traduzidas de forma a que qualquer utilizador seja capaz de, no mínimo, compreender as funcionalidades da interface em questão.

Esta proposta sugere que uma aplicação capaz de, em tempo real, observar a interface de uma outra aplicação, reconhecendo todo o texto apresentado através de um sistema de Reconhecimento Ótico de Carateres – *Optical Character Recognition* (ou *OCR*), traduzindo-o de seguida através de *machine translation,* e apresentando os resultados diretamente sobre o texto original, seria uma grande mais-valia para uma vasta comunidade de potenciais utilizadores, que de outra forma não seriam capazes de tirar partido da aplicação estrangeira.

Para testar o quão exequível é a solução idealizada, foi desenvolvido um protótipo de uma ferramenta seguindo a ideologia proposta. A esta ferramenta foi-lhe dado o nome de *Direct Overlay Translator*, ou *DOT*.

Com isto pretende-se que, uma grande quantidade de aplicações que até ao momento não eram utilizáveis por uma grande quantidade de potenciais utilizadores, possam ser traduzidas o suficiente, mesmo que de forma imperfeita, para que se tornem acessíveis.

Conceitos Chave:

Comunicação; Barreiras Linguísticas; Interface com o Utilizador ; *Machine Translation; Optical Character Recognition;* Deteção e Segmentação de Texto; *Overlays* sobre Aplicações Gráficas.

Abstract

In a perfect world, any person, from anywhere in the world, would be able to speak and understand a single common language, eliminating all language barriers and vastly improving intercultural communication. The wide spread of the English language came as a response to this problem, however there are still many cases where media and software are incomprehensible to a large amount of its potential audience simply due to language barriers.

In order to solve this issue, a potential solution was idealized, and developed, so that the user interface of any given software, developed by and for different cultures, could be translated in such a way that would still be of use to any foreign individual.

The proposed solution to this problem envisions an application that, by observing the interface of any other target application; interpreting all of its text with Optical Character Recognition (OCR) techniques; and translating it through a machine translation software, would be capable of displaying in real time a translated version of said interface directly over its original counterpart. This would be of great benefit to a potentially large group of people since it would allow them to interact with foreign applications which would otherwise be unusable to them.

In order to test the feasibility of the proposed idea, a prototype tool was developed using the proposed solution's ideology. The final prototype was named Direct Overlay Translator, or DOT for short.

The intent behind this project is to provide the public with a tool that is able to convert an otherwise completely unusable application, due to a lack of comprehension of its language, into a usable version of this app by translating, even if in an imperfect way, as much as possible of its user interface.

Key Concepts:

Communication; Language Barriers; User Interface; Machine Translation; Optical Character Recognition; Text Detection; Graphical Overlays.

Índice

Resumo v

Abstract vii

1 Introdução 17

1.1 Contexto 17

1.2 Interpretação do Problema 18

1.3 Objetivo 19

1.4 Resultados Esperados 19

1.5 Análise de Valor 20

1.6 Abordagem Preconizada 21

1.7 Estrutura do Documento 22

2 Contexto e Estado da Arte 23

2.1 Contexto 23

2.2 Restrições 24

2.3 Estado da Arte em Abordagens Existentes 25

2.3.1 Word Lens (Anon., 2017) 25

2.3.2 WayGo (WayGo, s.d.) 26

2.3.3 Visual Novel Reader (Jichi, 2016) 27

2.3.4 Translator Aggregator & Interactive Text Hooker 28

2.4 Estado da Arte em Tecnologia Relevante 29

2.4.1 Deteção e Segmentação de Texto em Imagens 29

2.4.2 Optical Character Recognition 34

2.4.3 Machine Translation 36

2.4.4 *Overlay* sobre Aplicações Gráficas 38

2.5 Análise de Valor 39

2.5.1 New Concept Development Model 40

2.5.2 Value, Value for the Customer e Perceived Value 41

2.5.3 Proposta de Valor 42

2.5.4 Modelo Canvas 43

2.5.5 Modelo de Verna Allee 46

2.5.6 Análise de Valor Multicritério 47

2.6 Conclusão 50

3 Avaliar Soluções e Abordagens Existentes 51

3.1 Avaliação das Abordagens Existentes 51

3.1.1 Word Lens 51

3.1.2 WayGo 52
3.1.3 Visual Novel Reader (VNR) 52
3.1.4 Translator Aggregator & Interactive Text Hooker 53
3.2 Avaliação das Tecnologias Relevantes 53
3.2.1 Deteção e Segmentação de Texto em Imagens 53
3.2.2 Optical Character Recognition 55
3.2.3 Machine Translation 56
3.2.4 *Overlay* sobre Aplicações Gráficas 57
3.3 Conclusão 58
4 Design da solução 59
4.1 *Design* da Aplicação 59
4.2 Conclusão 63
5 Desenvolvimento da Solução 65
5.1 Deteção e Segmentação de Texto 65
5.2 Optical Character Recognition 72
5.3 Machine Translation 74
5.4 Overlay sobre Aplicações Gráficas 75
5.5 PlugIn API 77
5.6 Configuração de Registos 78
5.7 Interface Gráfica 79
5.8 Integração de Suporte a *Text Hooking* 82
5.9 Conclusão 83
6 Avaliação da solução 85
6.1 Experiências e Avaliação da Solução 85
6.1.1 Grandezas a Utilizar 85
6.1.2 Hipóteses a Testar 86
6.1.3 Metodologia de Avaliação 88
6.1.4 Primeira Fase – Testes Funcionais e de Desempenho 89
6.1.5 Segunda Fase – Testes Não Funcionais 96
6.1.6 Respostas às Hipóteses Testadas 98
6.2 Conclusão 101
7 Conclusão e Perspetivas de Trabalho Futuro 103
8 Referências 105
9 Anexos 109

Lista de Figuras

Figura 1 - Esquema do processo manual necessário para a interpretação de uma aplicação 18
Figura 2 - À esquerda: Menu Original em Russo; À direita: Exemplo de solução com um overlay de tradução 19
Figura 3 - Processo simplificado do funcionamento da solução proposta 21
Figura 4 - WordLens 26
Figura 5 - WayGo 26
Figura 6 - VNR 28
Figura 7 - Translator Aggregator em conjunto com o Interactive Text Hooker 29
Figura 8 - Funcionamento geral de um algoritmo de Sliding Window 30
Figura 9 - A – Original (Anon., 2017); B - Filtragem por Arestas; C – Filtragem por Alta frequencia (High Pass Filter); Filtragem baseada na espessura de traços; 31
Figura 10 - Cálculo da Espessura de Traços (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010) 32
Figura 11 - A - Original; B - Filtragem de Gabor; C - Mascara Binária; D - Resultado Final (Kumar, 2014) . 33
Figura 12 - Exemplos de Overlay; Canto Superior Esquerdo: RivaTuner (Youtube, s.d.); Canto Superior Direito: Fraps (Anon., 2017); Canto Inferior Esquerdo: Origin Client (Yin-Poole, 2012); Canto Inferior Direito: Steam Client (Youtube, s.d.) 38
Figura 13 - Modelo Canvas de uma possível implementação comercial da solução 44
Figura 14 - Mapa de Value Network Analysis (Anklam, s.d.) 46
Figura 14 - Diagrama de Classes (Parte 1)) 60
Figura 15 - Diagrama de Classes (Parte 2) 61
Figura 16 - Diagrama de Sequência Simplificado do Pipeline de Tradução 62
Figura 17 - Calibração inicial DOT 67
Figura 18 - Filtragem por Contraste DOT 67
Figura 19 - Dilatação Horizontal DOT 68
Figura 20 - Filtragem por Cor DOT 70
Figura 21 - Filtragem por Blobs DOT (Cores) 71
Figura 22 - Filtragem por Blob (Binária) 71
Figura 23 - Diagrama de sequência do processo de OCR 73
Figura 24 - Diagrama de sequência do processo de tradução 75
Figura 25 - À esquerda, LensWindow em modo Automático; À direita LensWindow customizada manualmente 76
Figura 26 - Diagrama de sequência do processo de display de linhas de texto 76
Figura 27 - À esquerda: Janela de Loading; À direita LensWindow 79
Figura 28 - Menu de Calibração 80
Figura 29 - Menu de Calibração (performance meter) 81
Figura 30 - Menu Principal 81
Figura 31 - Resultados dos testes de Utilização (Imagem 1) 96
Figura 32 - Resultados dos testes de Utilização (Imagem 2) 97

Figura 33 - Resultados dos testes de Utilização (Imagem 3) 97
Figura 34 - Resultados dos testes de Utilização (Video 1) 97
Figura 35 - Questionário para grupo de testes de usabilidade (Parte 1) 112
Figura 36 - Questionário para grupo de testes de usabilidade (Parte 2) 112
Figura 37 - Questionário para grupo de testes de usabilidade (Parte 3) 112
Figura 38 - Img01 utilizada durante os testes de usabilidade 112
Figura 40 - Img03 utilizada durante os testes de usabilidade 112
Figura 39 - Img02 utilizada durante os testes de usabilidade 112

Lista de Tabelas

Tabela 1 Matriz de Comparação de Critérios 47
Tabela 2 - Características associadas a cada solução de tradução 48
Tabela 3 - Tabela AHP para o cálculo da percentagem por critério de avaliação 49
Tabela 4 - Tabela AHP para o cálculo da percentagem resultado de cada solução 49
Tabela 5 - Comparação das caraterísticas das soluções de OCR estudadas 55
Tabela 6 - Comparação das caraterísticas das soluções de tradução estudadas 56
Tabela 7 - Testes: Filme 'The Hobbit: The Desolation of Smaug' 90
Tabela 8 - Testes: Panda AV 90
Tabela 9 - Testes: Rise of the Tomb Raider 91
Tabela 10 - Testes Malwarebytes 91
Tabela 11 - Testes: Skype 92
Tabela 12 - Testes extra de Overlay 93
Tabela 13 - Teste de Suporte a Text Hooking 95
Tabela 14 - Exemplo do método de teste das capacidades da aplicação DOT (Parte 1) 109
Tabela 15 - Exemplo do método de teste das capacidades da aplicação DOT (Parte 2) 111

Acrónimos e Símbolos

Lista de Acrónimos

AHP	*Analytic Hierarchy Process*
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
BSD	*Berkeley Software Distribution*
DLL	*Dynamic-link library*
DOT	*Direct Overlay Translator*
FPS	*Frames per second*
GDI	*Graphics Device Interface*
GPL	*General Public License*
ITH	*Interactive Text Hooker*
MODI OCR	*Microsoft Office Document Imaging Optical Character Recognition*
NCD	*New Concept Development*
OCR	*Optical Character Recognition*
REST	*Representational State Transfer*
SWT	*Stroke Width Transform*
TA	*Translator Aggregator*
UI	*User Interface*
VNR	*Visual Novel Reader*
WinAPI	*Windows Application Programming Interface*
WPF	*Windows Presentation Foundation*

1 Introdução

Neste capítulo será feita de forma resumida a contextualização do problema que será discutido ao longo desta dissertação: a idealização de uma solução capaz de traduzir em *overlay* o conteúdo escrito de uma outra aplicação à medida que este é apresentado, assim como o desenvolvimento do seu respetivo protótipo funcional. A esta aplicação foi-lhe dado o nome de *Direct Overlay Translator* ou *DOT*.

Assim, este capítulo irá abordar o contexto geral deste problema, a sua definição, os objetivos a atingir e os resultados esperados. Por fim será feita uma breve descrição da análise de valor associada à solução prevista; assim como a sua abordagem preconizada, que descreverá muito resumidamente os passos necessários à tradução de uma aplicação gráfica.

1.1 Contexto

A solução proposta nesta dissertação tem por objetivo aproximar um pouco mais diferentes culturas ao automatizar o processo de tradução da componente principal de interação com o utilizador em software aplicativo: a interface com o utilizador. Para atingir este objetivo a abordagem proposta engloba conceitos tais como *machine translation, optical character recognition* (*OCR*)*,* deteção de texto e técnicas de *overlay* sobre aplicações gráficas. Todos estes conceitos são definidos em maior detalhe no capítulo 2.4.

Os únicos intervenientes que dizem respeito à solução preconizada são o utilizador da aplicação, a aplicação estrangeira e a aplicação de tradução descrita ao longo desta dissertação.

1.2 Interpretação do Problema

Aplicações somente disponíveis em línguas estrangeiras (não dominadas pelo utilizador) tornam-se difíceis de utilizar devido a uma simples barreira linguística. Em certos casos, tais como em menus, barras de ferramentas ou mensagens informativas, uma tradução simples (facilmente realizada por computador) é suficiente para permitir que a aplicação se torne compreensível e, desta forma, útil. Das aplicações deste tipo encontradas, nenhuma permite uma tradução direta (que não específica a uma única língua), apresentada na própria aplicação de forma a substituir o texto original. Isto é um problema visto que para certos casos, as únicas formas de compreender a aplicação envolvem aprender a língua em questão, o que não é viável na grande maioria dos casos, ou um processo manual (Figura 1) que obriga o utilizador a obter uma captura de imagem da aplicação, utilizar uma ferramenta de edição de imagem para recortar as linhas de texto em questão, passar todas estas novas secções de imagem por uma ferramenta de *OCR;* e por fim passar o resultado obtido por uma ferramenta de tradução, tal como o *Google Translator* (Google, s.d.), de forma a obter resultados compreensíveis.

Figura 1 - Esquema do processo manual necessário para a interpretação de uma aplicação

Este processo pode demorar vários minutos por cada frase a traduzir, o que o torna num processo muito pouco eficiente, especialmente quando este pode ser realizado de forma automatizada. Alternativamente, o utilizador poderia ainda prescindir das etapas de segmentação e de OCR, copiando manualmente o texto para uma ferramenta de tradução; mas este processo seria, na maioria dos casos, ainda mais ineficiente do que o descrito anteriormente, especialmente caso o utilizador desconheça como escrever corretamente os carateres que compõem o texto a traduzir.

1.3 Objetivo

A solução que será desenvolvida terá por objetivo a automatização do processo de tradução manual referido no subcapítulo 1.2, tornando-o num processo eficiente, rápido, simples e com o mínimo de esforço possível para o utilizador.

De forma resumida, pretende-se encontrar e interpretar todo o texto apresentado numa aplicação alvo, e seguidamente sobrepor ao mesmo, na própria janela da aplicação, a sua tradução automática. Esta tradução deverá ser feita preferencialmente de forma rápida para que a interação com a aplicação gráfica se mantenha coerente e satisfatória. De forma a abranger o maior número de aplicações possíveis, esta aplicação deverá estar preparada para aceitar aplicações tanto em janela como em *fullscreen*. Deve também ser possível aceitar uma variedade de diferentes línguas, especificamente Inglês, Português, Espanhol, Francês, Alemão, Japonês, Chinês Simplificado, Coreano e Russo, visto todas elas fazerem parte das 15 mais faladas mundialmente (Alphatrad, s.d.).

1.4 Resultados Esperados

Espera-se, durante o período reservado para esta dissertação, desenvolver uma aplicação para Windows que seja capaz de cumprir com todos os objetivos definidos no subcapítulo 1.3.

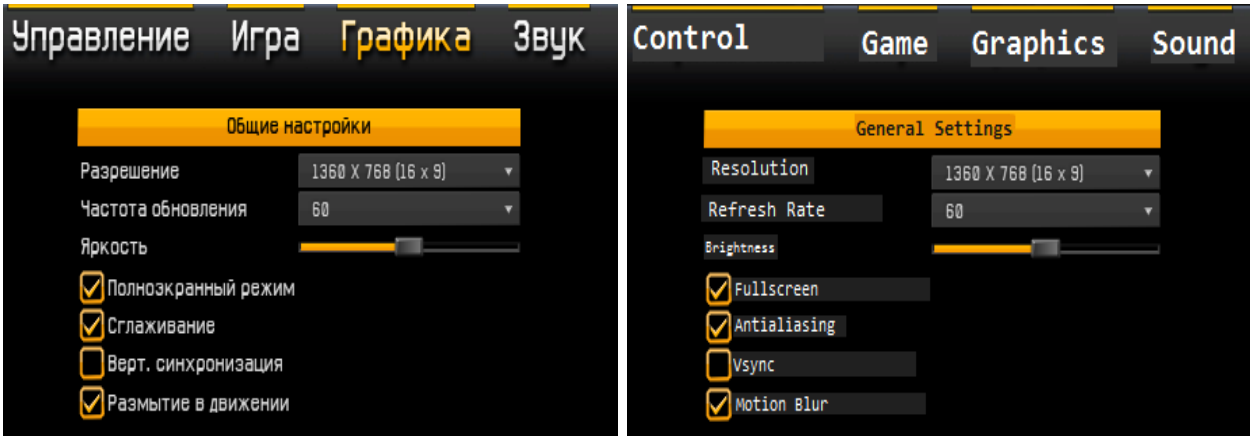


Figura 2 - À esquerda: Menu Original em Russo; À direita: Exemplo de solução com um overlay de tradução

Desta forma, são esperados da solução resultados que se assemelhem aos apresentados no exemplo da Figura 2, onde todo o texto identificado pela solução é traduzido e apresentado sobreposto ao texto original. Esta solução deverá ser capaz de manter tempos de reação razoáveis para que o utilizador consiga tirar partido da aplicação de forma muito semelhante à que utilizaria, caso esta estivesse originalmente na sua língua materna.

1.5 Análise de Valor

A análise de valor da solução proposta teve como base os potenciais benefícios que esta pode trazer aos seus utilizadores. Estes benefícios podem ser resumidos aos seguintes três:

- A tradução é realizada de forma muito rápida para que o utilizador se consiga focar nos aspetos da aplicação que realmente importam;
- Os segmentos de texto da aplicação são identificados de forma automática, reduzindo o esforço por parte do utilizador;
- A tradução é apresentada diretamente sobre a aplicação a traduzir, o que torna a interação com a aplicação mais intuitiva.

Outro aspeto importante ligado à análise de valor desta aplicação é o seu público-alvo. Prevê-se que os principais utilizadores desta aplicação se encontrem dentro das quatro seguintes categorias:

- Utilizadores frequentes de aplicações não traduzidas e sem previsão de tradução futura;
- Fãs de videojogos apenas destinados a mercados não internacionais (comum na China, Japão e Rússia);
- Indivíduos no processo de aprendizagem de uma nova língua;
- Utilizadores da *Web* com interesse em páginas estrangeiras.

1.6 Abordagem Preconizada

A solução proposta nesta dissertação, de forma a cumprir os objetivos apresentados na secção 1.3, terá por base o desenvolvimento de um sistema automatizado capaz de capturar a imagem apresentada pela aplicação a traduzir; encontrar e segmentar todas as linhas de texto existentes na imagem; permitindo assim que todas as linhas encontradas possam ser, logo de seguida, interpretadas por uma *framework* de *OCR,* capaz de interpretar a língua da aplicação em questão. O resultado da componente de *OCR* será então traduzido por uma das várias soluções de *machine translation* disponíveis atualmente; e por fim, o resultado traduzido deverá ser sobreposto ao texto original, através de técnicas de *overlay* de texto e imagem sobre a aplicação alvo. Este processo é realizado ciclicamente (com exceção das componentes de *OCR* e tradução, que apenas devem ser executadas quando necessário) para que a solução possa responder rapidamente a quaisquer mudanças na aplicação, tais como alterações do posicionamento de texto já traduzido ou mesmo o aparecimento de novo texto ainda por traduzir. Este sistema, de forma a manter um equilíbrio entre resultados de qualidade consistente; e uma boa autonomia (minimizando a necessidade de intervenção do utilizador) deverá requerer apenas uma calibração inicial rápida e simples, para que se possa adaptar de forma ótima à interface de qualquer aplicação.

O processo descrito no parágrafo anterior pode ser resumido ao apresentado no diagrama da Figura 3.

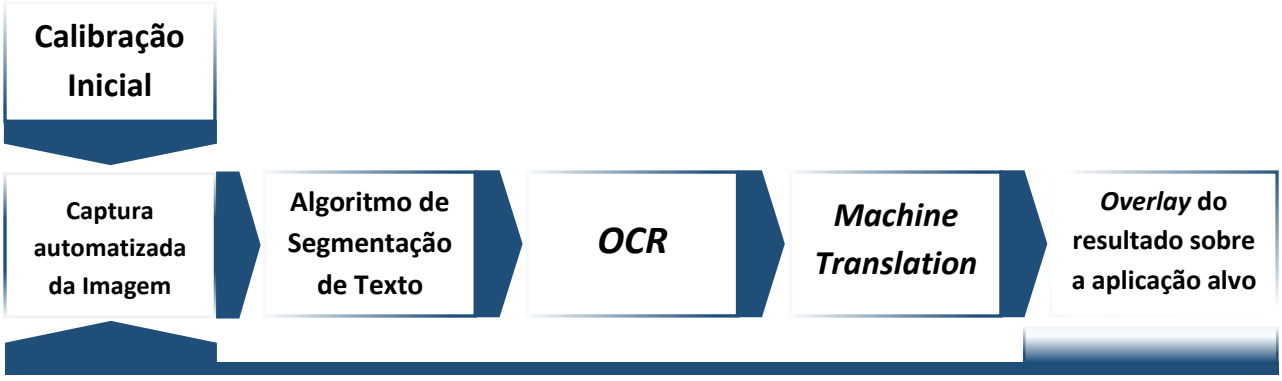


Figura 3 - Processo simplificado do funcionamento da solução proposta

1.7 Estrutura do Documento

Este documento é constituído por sete capítulos. O primeiro, a Introdução, descreve resumidamente o problema, o contexto onde este se insere, os objetivos e resultados esperados, assim como a abordagem proposta e uma introdução à análise de valor da solução.

No capítulo 2 (Contexto e Estado da Arte), é aprofundado o contexto deste projeto e as restrições que lhe estão associadas, é realizado um estado da arte relativo a abordagens já existentes e à tecnologia necessária para o desenvolvimento da solução. Para terminar o capítulo, é ainda realizada uma análise de valor da solução em maior detalhe.

No capítulo 3 (Avaliar Soluções e Abordagens Existentes), é feita uma avaliação tanto das abordagens já existentes como das tecnologias relevantes, referidas anteriormente no capítulo 2. É neste capítulo onde são abordados os prós e contras de cada uma das componentes a utilizar durante o desenvolvimento da solução.

O capítulo 4 (Design da solução) refere-se ao *design* da solução proposta e apresenta um modelo detalhado da estrutura atual do protótipo desenvolvido.

O capítulo 5 (Desenvolvimento da Solução) será dedicado à descrição dos procedimentos de desenvolvimento da aplicação, explicando o porquê do desenvolvimento de cada aspeto chave.

O capítulo 6 (Avaliação da solução) irá apresentar os processos principais utilizados para testar a aplicação desenvolvida, assim como os seus resultados e respetivas conclusões.

Por fim, no capítulo 7 (Conclusão e Perspetivas de Trabalho Futuro) são apresentadas todas as conclusões finais desta dissertação, assim como as suas respetivas perspetivas de trabalho futuro.

2 Contexto e Estado da Arte

Neste capítulo será inicialmente apresentado o contexto do problema em maior detalhe, incluindo uma apresentação das restrições atuais associadas a este projeto. De seguida será realizada uma análise do estado da arte, focada inicialmente em aplicações existentes que partilham semelhanças com a solução proposta e, seguidamente, será analisado o estado da arte relativo às tecnologias individuais que, em conjunto, deverão constituir a solução final. Para terminar o capítulo será ainda apresentada, em maior detalhe, a análise de valor da solução.

2.1 Contexto

Desde os primeiros contactos entre diferentes culturas que a primeira (e na maioria das vezes a maior) dificuldade de comunicação é causada pelo desconhecimento de ambas as partes de uma língua em comum. Com o surgimento deste problema, surge também uma nova profissão, o intérprete ou tradutor. A solução proposta nesta dissertação tem por objetivo aproximar um pouco mais diferentes culturas ao automatizar o processo de tradução da componente principal de interação com o utilizador em software aplicativo: a interface com o utilizador. Para atingir este objetivo a abordagem proposta engloba conceitos tais como *machine translation, optical character recognition,* deteção de texto e técnicas de *overlay* sobre aplicações gráficas. Todos estes conceitos são definidos em maior detalhe no capítulo Estado da Arte em Tecnologia Relevante, mais especificamente em cada um dos seus subtópicos que lhes foram reservados.

A abordagem proposta deve permitir que um utilizador, sem conhecimentos de uma determinada língua, consiga interagir diretamente com uma aplicação, inteiramente apresentada nessa mesma língua, necessitando apenas de configurar uma segunda aplicação (a solução proposta) para que esta interprete e apresente resultados compreensíveis para o utilizador sobre a própria aplicação estrangeira. Assim, podem ser identificados 3 atores que intervêm diretamente nesta situação, o **utilizador do sistema**, que interage com a **aplicação**

estrangeira, que é capaz de interpretar após configurar e calibrar a **aplicação de tradução** ***Direct Overlay Translator*** *(DOT)*.

2.2 Restrições

O nível de qualidade e eficácia da solução final está limitado a uma série de restrições ligadas maioritariamente à capacidade atual dos sistemas de *machine translation* e *optical character recognition*.

Relativamente a ***machine translation,*** apesar desta tecnologia ter evoluído muito nos últimos anos, os resultados de uma tradução realizada através desta técnica tendem a ser muito inferiores aos realizados por um intérprete, por vezes resultando em frases com significados fora do contexto, ilógicas ou mesmo completamente incompreensíveis. Estes resultados são especialmente sensíveis a ambiguidades, sendo afetados negativamente ao traduzir calão e línguas muito sensíveis ao contexto.

De momento, nenhum sistema de ***optical character recognition*** é capaz de obter resultados com uma precisão de 100% de forma consistente. Dos fatores que propiciam o aparecimento de erros nos resultados destes sistemas destacam-se:

- **O tamanho e definição da letra** - é recomendado um mínimo de 20 píxeis de altura;
- **O tipo de *font* usada** - *fonts* muito estilizadas são difíceis de reconhecer;
- **O número e complexidade dos carateres da língua alvo** - quanto maior o número de carateres, maior a probabilidade de obter falso-positivos. Quanto mais complexo o caráter maior terá que ser a qualidade de imagem para que este seja bem lido;
- **O nível de contraste e *aliasing* da imagem analisada** – quanto menor o nível de contraste mais difícil é o seu reconhecimento.
- **O nível de complexidade do *background*** – *backgrounds* constituídos por imagens complexas tendem a causar graves problemas de interpretação a estes sistemas.

Até que sejam realizados futuros avanços relativos a estas duas áreas, a solução proposta não será capaz de apresentar traduções de qualidade em aplicações dependentes da apresentação de grandes quantidades de texto. No entanto, mesmo com as restrições apresentadas, esta solução continua a ter utilidade prática quando a quantidade de texto a traduzir é relativamente pequena (tal como em menus e instaladores) e quando a qualidade de tradução é irrelevante desde que a mensagem traduzida seja compreensível.

2.3 Estado da Arte em Abordagens Existentes

Neste subcapítulo são abordadas quatro soluções que partilham semelhanças com o projeto proposto: *Word Lens; WayGo; VNR* e uma solução que engloba duas aplicações distintas, o Translator Aggregator e o Interactive Text Hooker.

2.3.1 Word Lens (Anon., 2017)

Word Lens é uma aplicação de tradução para Android e iOS que usa realidade aumentada para substituir frases numa determinada língua pela sua tradução, numa outra língua desejada.

Esta aplicação foi desenvolvida pela *Quest Visual* que, após o seu lançamento, foi adquirida pela *Google*.

A tradução é realizada em tempo real sem necessitar de acesso à internet e utiliza um sistema de *OCR* para identificar palavras nas imagens capturadas pela câmara. Apesar desta identificação não ser perfeita, visto ter problemas com algumas *fonts* e texto manuscrito, “Perfeição nunca foi o objetivo” (Olson, 2010) nas palavras de Otávio *Good,* um dos *developers*.

Esta aplicação é, de momento, capaz de traduzir cerca de 6 línguas diferentes de e para inglês, sendo estas português, espanhol, italiano, francês, alemão e russo.

Observando a Figura 4 é possível ter uma ideia simples do funcionamento desta aplicação. Ao apontar a câmara do dispositivo para um segmento de texto, o *Word Lens* automaticamente identifica, traduz e substitui o texto encontrado pela sua versão traduzida.

Figura 4 - WordLens

2.3.2 WayGo (WayGo, s.d.)

WayGo é uma aplicação para *Android* e *iOS* que funciona um pouco como uma versão simplificada do *Word Lens,* pois o resultado não aparece sobreposto à imagem. No entanto o *WayGo* destaca-se do *Word Lens* pela sua especialização em traduzir línguas orientais para inglês, nomeadamente japonês, coreano e chinês.

Figura 5 - WayGo

A partir das imagens apresentadas na Figura 5 é possível ter uma ideia simples do funcionamento do WayGo. Ao apontar a câmara de forma a que a linha de texto a traduzir se encontre na caixa

delineada a laranja, esta aplicação interpreta e apresenta os resultados da tradução diretamente abaixo, como é possível ver na Figura 5.

2.3.3 Visual Novel Reader (Jichi, 2016)

VNR é uma aplicação *freeware* para *Windows* especializada na captura e tradução de texto de *visual novels* japonesas. *Visual novels* são um género de videojogos caracterizados por se focarem apenas na sua narrativa e manterem um tipo de jogabilidade muito minimalista (em alguns casos até inexistente). São exemplos deste género títulos tais como "*Steins;Gate*" e as séries "*Ace Attorney*" e "*Zero Escape*".

Desta forma, apesar de suportar traduções para várias línguas, apenas identifica e traduz japonês.

Para além de utilizar *machine translation*, o *VNR* é ainda capaz de receber traduções manuais associadas a uma *visual novel* específica, o que, na grande maioria das vezes, melhora de forma muito significativa a qualidade de tradução.

Esta aplicação aborda a componente de captura de texto de forma diferente da apresentada pelo *Word Lens* e o *WayGo*. Visto ser desenvolvida com o objetivo muito específico de traduzir *visual novels*, que por norma se caracterizam pelo envio de grandes quantidades de texto sob a forma de *strings* para a sua interface, esta aplicação tira partido desta especificidade capturando o texto a traduzir através de *text hooks*.

De forma simplificada, um *text hook* é capaz de obter o texto de uma aplicação ao monitorizar certas funções associadas ao tratamento e *rendering* de texto, tais como *GetTextExtentPoint32* (Microsoft, s.d.) e *GetGlyphOutline* (Microsoft, s.d.), assim como funções ligadas ao *Console Output*. A vantagem da utilização deste tipo de sistema comparativamente a um sistema baseado em *OCR* é o nível superior de precisão dos resultados obtidos, assim como a maior velocidade da sua obtenção. No entanto a utilização das funções referidas anteriormente varia grandemente

de aplicação para aplicação o que faz com que esta técnica apenas seja viável em casos muito específicos.

Figura 6 - VNR

Na Figura 6 é possível ver uma *visual novel* com o *overlay* do *VNR* destacado dentro do retângulo vermelho de cima, a realizar uma tradução do texto destacado no retângulo a vermelho diretamente abaixo.

2.3.4 Translator Aggregator & Interactive Text Hooker

Para além do *VNR* existem também outras soluções menos práticas, novamente específicas para a tradução de *visual novels* japoneses. Esta solução é constituída por duas aplicações diferentes. O Translator Aggregator (*TA*) e o Interactive Text Hooker (*ITH*).

A única função do *TA* é, a partir de uma frase, apresentar múltiplas traduções ao interagir com vários *websites* de tradução simultaneamente. Desta forma, com acesso a várias traduções da mesma frase, o leitor é capaz de melhor interpretar a frase original, visto que a grande maioria das soluções disponíveis de *machine translation* ainda têm muitos problemas em interpretar frases complexas (particularmente problemático em japonês e chinês).

O *ITH* é a ferramenta que extrai da aplicação o texto a ser traduzido, tal como o *VNR,* tirando partido de *text hooking*, e o escreve para o *clipboard*, onde o *TA* obtém as frases a traduzir.

Figura 7 - Translator Aggregator em conjunto com o Interactive Text Hooker

Na Figura 7 é possível ver à esquerda a janela de tradução do *TA* a apresentar a tradução dos resultados obtidos pelo *ITH*. Do lado direito encontra-se a janela da *visual novel* cujo texto está a ser obtido pelo *ITH*.

2.4 Estado da Arte em Tecnologia Relevante

Neste subcapítulo será feito um estudo do estado da arte atual de todas as componentes relevantes para o projeto em questão. Estas componentes englobam a deteção e segmentação de texto numa imagem; *OCR*, para que o texto encontrado possa ser interpretado; *machine translation*, para que o texto interpretado seja traduzido; e por fim *overlay* em aplicações gráficas, para que o resultado da tradução possa ser apresentado diretamente sobre a aplicação alvo.

2.4.1 Deteção e Segmentação de Texto em Imagens

Existem várias formas de identificar a posição de texto em imagens ou *frames* de vídeo em ambientes complexos, onde um *OCR* sozinho não é capaz de resultados positivos. Esta área de

estudo tem sido alvo de atenção de muitas dissertações, e em geral estes métodos podem ser divididos, tendo em conta a forma de abordagem ao problema, em duas categorias: *texture-based* e *region-based* (Sanmartín, 2015).

Algoritmos ***Texture-based*** tentam identificar texto numa imagem utilizando métodos de *sliding window*, com uma janela de tamanho estático, sobre uma pirâmide de escalas da imagem a analisar (Sanmartín, 2015), como é possível ver na Figura 8.

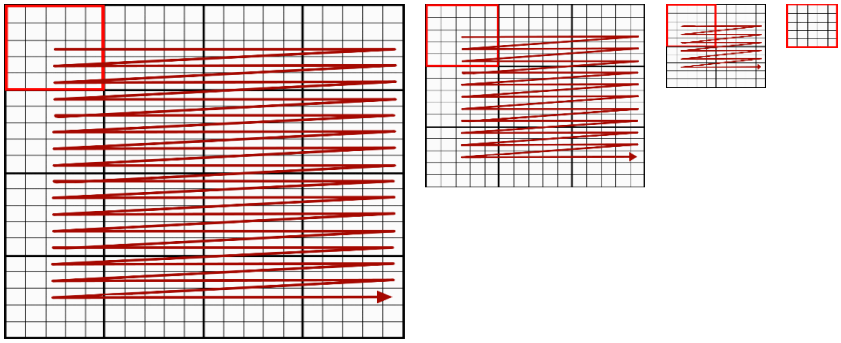


Figura 8 - Funcionamento geral de um algoritmo de Sliding Window

A janela deslizante identifica zonas com texto assim que certas propriedades de textura da área correspondam a propriedades textuais preestabelecidas.

Dentro desta abordagem destacam-se os seguintes trabalhos de (Li, et al., 1998) , (Yangxing, et al., 2006) e (Lienhart, & Wernicke, 2002).

Métodos de deteção de texto que utilizam esta abordagem, por norma, sofrem de falta de precisão e baixo desempenho, devido à complexidade computacional associada à análise de várias escalas da mesma imagem (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010).

Algoritmos ***Region-based*** tentam segmentar texto de uma imagem através de certas propriedades características de elementos textuais. Destas características destacam-se a alta densidade de cantos e arestas, zonas de grande flutuação de contraste (zonas de alta frequência), consistência da espessura e cor do traço dos carateres, entre outras (Sanmartín, 2015). Na Figura 9 é possível observar implementações de filtragens utilizando as características textuais referidas.

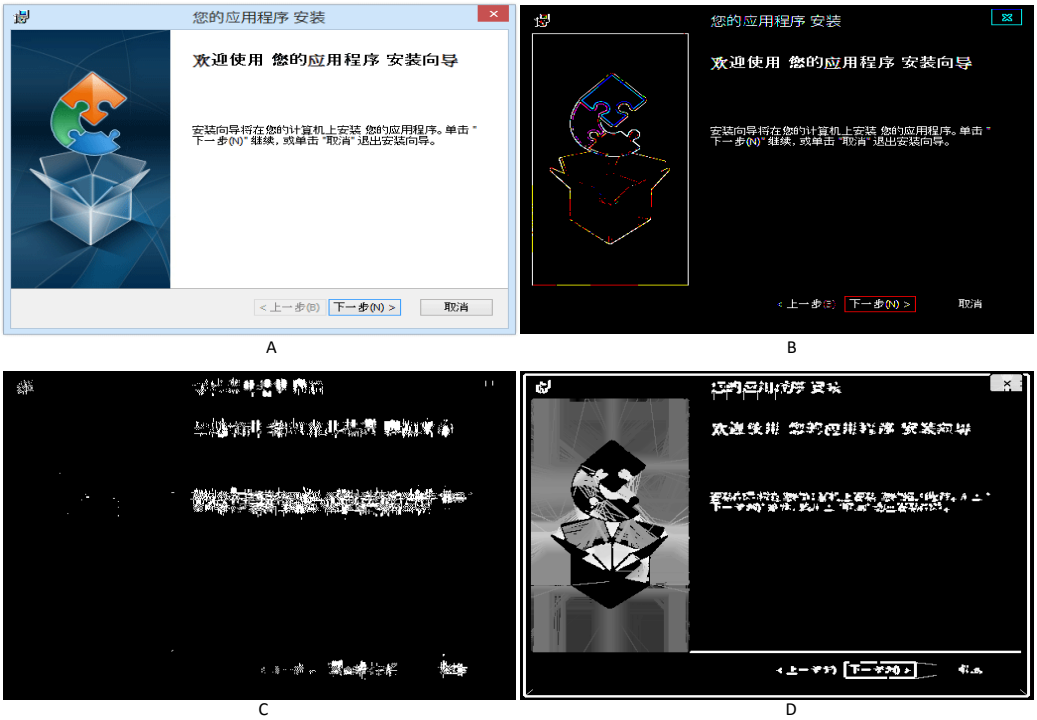


Figura 9 - A – Original (Anon., 2017); B - Filtragem por Arestas; C – Filtragem por Alta frequencia (High Pass Filter); Filtragem baseada na espessura de traços;

A primeira imagem (A) mostra uma captura da interface de um instalador. As restantes imagens (B, C e D) apresentam o resultado da passagem de um filtro baseados em arestas; um filtro baseado em diferenças de contraste; e um filtro com base na espessura de traços, respetivamente. Os resultados obtidos em B, C e D tiveram todos eles como origem A, e todos eles podem ser analisados heuristicamente de forma a obter localizações onde a probabilidade de existir texto é elevada.

Dentro da categoria de algoritmos *Region-Based* destacam-se os seguintes trabalhos: "*Detecting Text in Natural Scenes with Stroke Width Transform"* (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010), "*An Efficient Approach for Text Extraction in Images and Video Frames Using Gabor Filter*" (Kumar, 2014) e "*Efficient Text Extraction Algorithm Using Color Clustering for Language Translation in Mobile Phones*" (Canedo-Rodríguez, et al., 2012) dos quais será feita uma breve e muito simplificada descrição.

Stroke Width Transform (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010)

Esta técnica de identificação de texto em cenas naturais baseia-se principalmente num aspeto comum à grande maioria dos carateres usados em escrita digital, a espessura constante das linhas que compõem um caráter. Este algoritmo, após realizar uma operação de *Edge Detection* (*Canny*) sobre a imagem a analisar, calcula as distâncias entre as arestas detetadas e, a partir dos resultados obtidos, é capaz de identificar áreas interligadas onde a distância é aproximadamente constante (Figura 10). Estas áreas são pintadas a preto, enquanto que todas as restantes zonas se mantêm brancas. Desta forma é possível, após alguma filtragem adicional de resultados, identificar localizações onde é altamente provável existir texto.

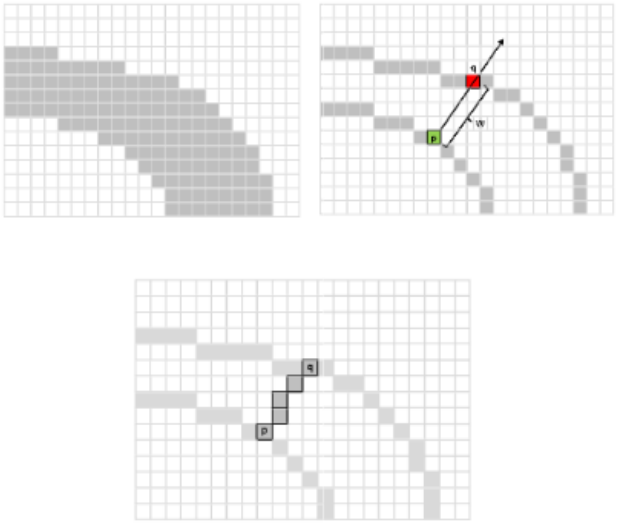
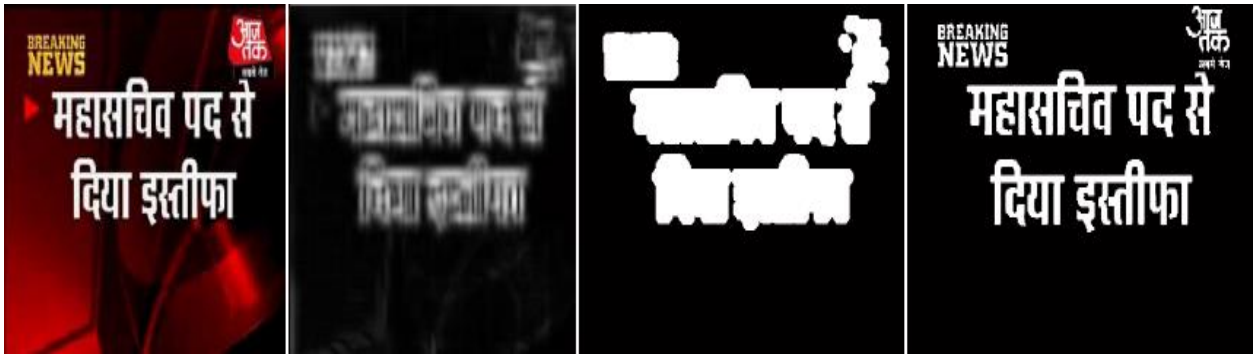


Figura 10 - Cálculo da Espessura de Traços (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010)

Efficient Text Extraction using Gabor Filter (Kumar, 2014)

Esta técnica de extração de texto de uma imagem explora a alta densidade de arestas que caracterizam áreas ricas em elementos textuais. Inicialmente a imagem original é transformada numa *Edge Intensity Image* utilizando filtros de *Gabor*. De seguida, é aplicado um *threshold* (*Otsu*) de forma a tornar a *Edge Intensity Image* numa imagem puramente binária (apenas preto e

branco). A imagem binária é tratada com uma série de operações morfológicas, tais como *Binary Open* e *Dilation*, de forma a remover ruído e aglomerar os resultados obtidos. Por fim, os resultados são encapsulados em *bounding boxes* e filtrados tendo em conta parâmetros como o seu rácio de altura/largura e área. É possível visualizar os resultados dos processos referidos anteriormente na Figura 11.



A B C D

Figura 11 - A - Original; B - Filtragem de Gabor; C - Mascara Binária; D - Resultado Final (Kumar, 2014)

Efficient Text Extraction Using Color Clustering (Canedo-Rodríguez, et al., 2012)

Este processo de extração tira partido das caraterísticas cromáticas da maioria das componentes textuais numa imagem: a sua cor constante.

O método proposto começa por aplicar uma quantização de cor à imagem, de forma a aglomerar e reduzir o número de cores a processar. De seguida, calcula as principais componentes de cor da imagem e decide quais destas pertencem ao fundo da imagem e quais fazem parte de elementos textuais. Desta forma é possível isolar de forma computacionalmente eficiente a área onde o texto se encontra.

2.4.2 Optical Character Recognition

Optical Character Recognition, ou *OCR*, é uma tecnologia que permite que, a partir de uma imagem, se consiga reconhecer os carateres nela presentes. Desta forma é possível obter o texto presente na imagem numa forma editável e reconhecível por qualquer editor de texto. Esta componente tem um papel importante pois, após a deteção do texto, este terá de ser convertido para uma *string* de carateres para que o seu conteúdo possa ser trabalhado.

Atualmente existem várias soluções de *OCR* disponíveis das quais, para o propósito deste projeto, foram analisadas as seis mais relevantes.

Tesseract (zdenop, et al., 2017)

Originalmente desenvolvido pela *HP* (*Hewlett Packard*) entre 1985 e 1994 e mantido pela *Google* desde 2006, esta ferramenta é atualmente *open source* (sob licença Apache 2.0) e mantém-se uma das melhores soluções gratuitas de *OCR,* tanto em relação ao número de línguas que é capaz de reconhecer como relativamente à precisão dos seus resultados. Foi desenvolvido em C e C++ e suporta tanto *Windows* como *Linux* e *OSX* (de forma não oficial) (Anon., 2016).

Esta *framework* mantém-se relevante, tendo a sua última *stable release* (3.05.01) até ao momento de escrita deste documento sido publicada a 1 de Junho de 2017. Recentemente (25/03/2017) foi também disponibilizada a versão alfa 4.00.00 que utiliza agora redes neuronais para melhorar a precisão dos resultados obtidos.

Estão atualmente disponíveis cerca de 98 ficheiros de treino para diferentes línguas, entre as quais, para este projeto, se destacam Inglês, Português, Espanhol, Francês, Alemão, Italiano, Japonês, Chinês Simplificado, Coreano e Russo.

A9T9's Free OCR (A9T9, 2015)

Esta framework foi desenvolvida pela *A9T9 Autonomous Technology* para sistemas operativos *Windows*. É *open source* (licença *GPL V2*), foi implementada em C# e está disponível no formato de *NuGet package,* o que simplifica a sua possível integração neste projeto. Esta *framework* é

atualmente capaz de identificar cerca de 21 línguas incluindo (tal como o *Tesseract*) Inglês, Português, Espanhol, Francês, Alemão, Italiano, Japonês, Chinês Simplificado, Coreano e Russo.

NHocr (Goto, Hideaki; Tohoku University, 2014)

Este *OCR* é um projeto *open source* (*Apache License 2.0*) desenvolvido especificamente para detetar carateres japoneses (no futuro possivelmente chineses também). Esta especialização melhora a precisão da deteção de texto em japonês à custa de uma muito fraca deteção de carateres ASCII.

Esta framework foi desenvolvida em C++ e suporta múltiplas plataformas, tais como *Windows*, *Linux*, Solaris SPARC/x86 assim como outras plataformas do tipo *Unix*.

Puma.NET (Temarez, 2017)

Puma.Net é um *wrapper* em C# para a *framework* de *OCR CuneiFrom*. O *CuneiFrom* foi desenvolvido pela *Cognitive Technologies* (uma companhia russa de desenvolvimento de software) e, apesar de ter sido originalmente um produto comercial, desde 2008 que se tornou num projeto *open source* (BSD license). Esta solução suporta cerca de 23 línguas, incluindo Inglês, Português, Espanhol, Francês, Alemão, Italiano e Russo. É ainda capaz de reconhecer textos que contenham russo e inglês em conjunto. Esta é uma opção apenas viável para as línguas referidas anteriormente, visto que esta solução é incapaz de reconhecer carateres asiáticos (chineses, japoneses, coreanos, indianos etc) e educar este sistema nesse sentido seria uma tarefa complexa, visto que a estrutura dos ficheiros de treino nunca foi divulgada ao público.

GOCR (Schulenburg, 2013)

O *GOCR* é um programa de *OCR open source* (*GNU Public License*) desenvolvido em C++. O seu criador, Joerg Schulenburg, publicou a primeira versão estável deste projeto em Setembro de 2010 e, a partir desse momento, Joerg tem liderado uma pequena equipa de *developers*, tendo a última versão estável até ao momento (0.50) sido publicada em 2013. Esta solução é pouco prática tendo em conta as necessidades deste projeto, pois para além de apenas ser capaz de

reconhecer carateres latinos e ter problemas com fontes do tipo *serif*, a versão *library* deste projeto foi abandonada (24 de Dezembro de 2006) antes de ter sido terminada.

ABBYY FineReader (ABBYY, 2017)

O FineReader (licença comercial) foi desenvolvido pela ABBYY, uma empresa sediada em Moscovo e fundada por David Yang em 1989. Desenvolvido para converter *PDF* e fotografias digitais para formatos pesquisáveis e editáveis, esta é possivelmente a ferramenta de *OCR* de maior renome e afirma ser capaz de um "reconhecimento de texto incomparável" com a sua competição.

No entanto, testes que comparam a precisão do *FineReader* com o *Tesseract* demonstram que a diferença entre os dois é pouco significativa (Heliński, et al., s.d.).

Recentemente (24/01/2017) a *ABBYY* disponibilizou para venda o seu sistema de *OCR* mais recente: o *FineReader 14* (Wirth, 2017).

2.4.3 Machine Translation

Machine Translation é a área de computação que estuda o uso de *software* para traduzir texto (ou fala) de uma língua para outra.

A forma mais simples de *machine translation* é conhecida por *word-by-word* ou tradução literal. Esta forma de tradução, tal como o nome indica, limita-se a substituir cada palavra pela sua equivalente na língua alvo. Por oposição, *sense-for-sense translation*, tenta ter em conta o sentido da frase inteira, e é a norma usada pela maioria das soluções de *machine translation* atuais (Anon., 2017).

Assim, foram analisadas as quatro soluções mais relevantes da *Google*, *Microsoft, Yandex* e *Systran*.

Google Translator API (Google, 2016)

A solução de tradução da Google é provavelmente a mais conhecida das quatro *API* analisadas neste documento. Esta *API* (comercial) é capaz de traduzir mais de 90 línguas diferentes e até detetar o idioma de um texto desconhecido. Funciona exclusivamente *online*, através de pedidos *REST* e requer apenas uma *API key*, um indicador da linguagem alvo e a *string* do texto a traduzir.

Microsoft Translator API (Microsoft, 2016)

A *Microsoft translator API* funciona de forma muito semelhante à versão da sua concorrente da *Google*. No entanto, esta não requer pagamentos desde que o número de carateres enviados seja inferior a 2 milhões por mês. É capaz de traduzir cerca de 60 diferentes línguas e, assim como o *Google Translator,* consegue detetar a língua de um texto não identificado. A única potencial dificuldade é a obrigatoriedade de criação de uma conta gratuita *Microsoft Azure* para cada utilizador. Para além das vantagens já referidas, a qualidade de tradução desta solução tem vindo a melhorar ao longo dos anos, e neste momento é capaz de resultados que rivalizam em qualidade com os resultados da *Google*.

Yandex Translate (Yandex, 2016)

A *API* de tradução da *Yandex* é capaz de traduzir 89 diferentes línguas e, tal como as últimas duas, funciona por *REST*. Esta é uma *API* grátis, desde que o limite de 10 milhões carateres mensais não seja ultrapassado, e requer apenas uma *API key* que pode ser facilmente obtida no *site* da *Yandex*.

BabelFish (Systran, 2016)

Outra solução, utilizada por motores de busca como o *Yahoo* e o *AltaVista,* é o *BabelFish.* Este tradutor foi desenvolvido pela Systran e foi um dos pioneiros na área de *machine translation*.

O BabelFish é capaz de traduzir 75 línguas diferentes mas, ao contrario das soluções da *Microsoft* e da *Google*, este tradutor não é capaz de identificar a língua de um segmento de texto.

A Systran até ao momento não disponibilizou uma *API* para a utilização do BabelFish, pelo que apenas funciona através do *site* disponibilizado.

2.4.4 *Overlay* sobre Aplicações Gráficas

Graphical application overlay é uma técnica que permite sobrepor imagens ou texto na tela de qualquer aplicação externa. Exemplos de aplicações que tiram partido desta técnica são a *Steam Client*, Origin Client, *Fraps* e *RivaTuner* como é possível observar na Figura 12.

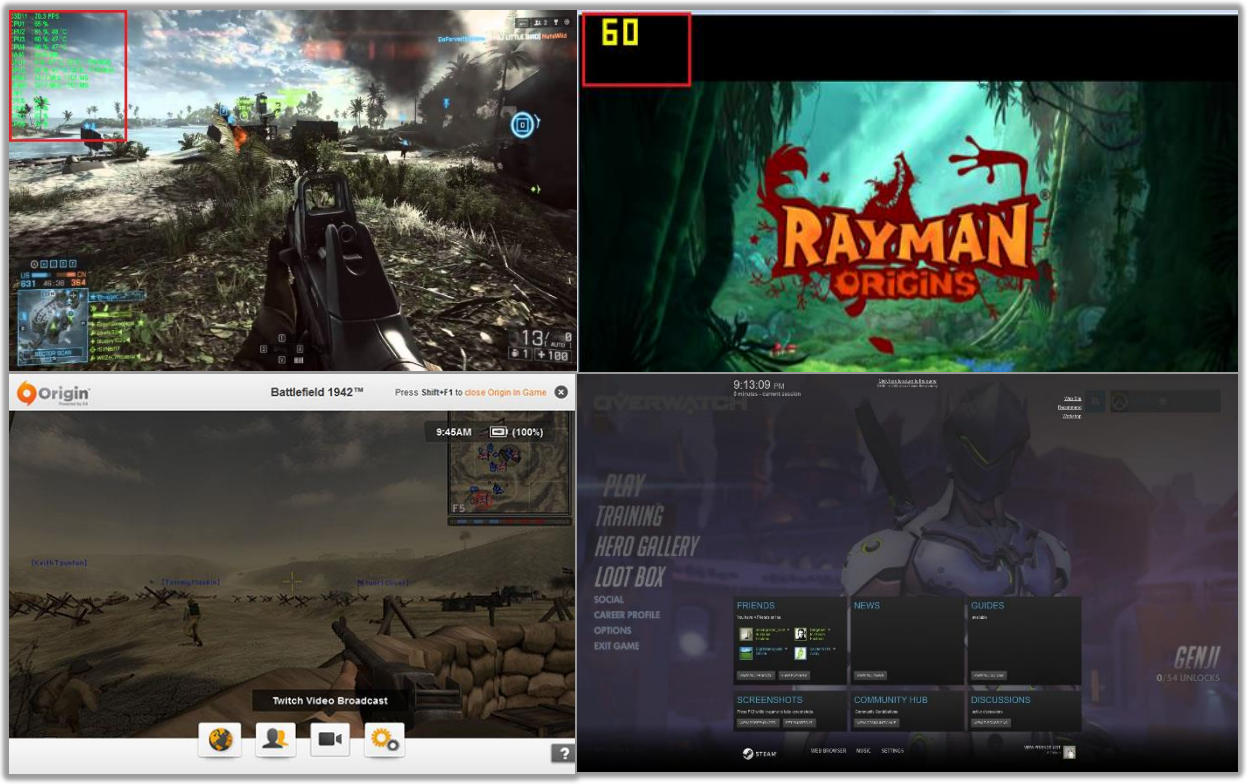


Figura 12 - Exemplos de Overlay; Canto Superior Esquerdo: RivaTuner (Youtube, s.d.); Canto Superior Direito: Fraps (Anon., 2017); Canto Inferior Esquerdo: Origin Client (Yin-Poole, 2012); Canto Inferior Direito: Steam Client (Youtube, s.d.)

Todas as aplicações mencionadas anteriormente são *closed source,* o que torna muito difícil saber exatamente que técnicas foram utilizadas para o seu desenvolvimento.

Assim, e antes de mais, é preciso distinguir métodos que apenas funcionam em aplicações *Windowed Mode* (*Borderless Window; Fake Fullscreen*) de métodos capazes de funcionar com *Exclusive Fullscreen Mode*.

Quando o objetivo é criar um *overlay* sobre uma aplicação em ***Windowed Mode***, a solução é simples e requer apenas que uma janela *borderless*, *topmost* (que se mantém sempre à frente de qualquer outra janela) e transparente se sobreponha à janela alvo. É nesta nova janela que

todo o desenho do *overlay* é realizado, utilizando *WinAPI/GDI* ou quaisquer outras ferramentas como *Qt* ou *WPF*. Exemplos de aplicações que utilizam esta técnica são a *Windows Magnifier* (Microsoft, 2016), *Image Overlay Utility* (imageoverlayutility@gmail.com, 2013–2016) e *Custom Desktop Logo* (Wong, 2008).

Quando é necessário realizar um *overlay* sobre uma aplicação em ***Exclusive Fullscreen****,* a solução anterior deixa de ser viável, visto que este modo, tal como o nome indica, dá exclusividade de desenho a uma única aplicação baseada em *DirectX* ou *OpenGL*. Desta forma, para criar um *overlay* sobre este tipo de aplicações existem uma série de soluções, todas elas envolvendo formas de *DLL injection*, o que permite que código extra seja executado dentro do espaço de endereçamento do processo alvo. Desta forma é possível, ao capturar certas funções chave do processo de *rendering* da aplicação, adicionar o código de desenho do *overlay* e continuar a execução da função original. Destas soluções destacam-se *Windows Hooks, Proxy DLL e Reflective Injection,* todas elas descritas em detalhe por (Berdajs & Bosnic, 2010). É possível obter uma descrição técnica mais detalhada de como realizar *Windows Hooks* no capitulo 22 ("*DLL Injection and API Hooking*") do livro (Richter & Nasarre, 2007).

Para uma aplicação *DirectX*, por norma é necessário verificar quando o *DLL d3d9.dll* ou *d3d11.dll* (*DirectX* 9 ou 11) é carregado pela aplicação, obter a posição de memória do método *EndScene* através de uma análise da *vtable*, e substituir este método por um novo com o código de desenho do *overlay* antes de voltar a chamar o *EndScene* original. Para uma aplicação em *OpenGL,* o método que é necessário substituir é o *SwapBuffers*, onde novamente será escrito o código de desenho do *overlay* antes do método original ser novamente chamado. Alguns exemplos de projetos *open source* que abordam implementações semelhantes às referidas são a *DirectDraw Overlay Library* (Haggag, 2007); *Taksi* (Jouline & Robinson, 2006); *Overlay Tools* (Metty, 2007) e *Overlay Using Raphook DLL* (Metty, 2008).

2.5 Análise de Valor

Esta secção foi construída de forma a que cada um dos seus subcapítulos responda, pela ordem respetiva, às questões impostas pelo módulo de análise de valor.

2.5.1 New Concept Development Model

O modelo *New Concept Development* (*NCD*) foi desenvolvido para padronizar toda a linguagem e terminologia associada aos processos de otimização da *front end* de inovação (Koen, 2004). Este modelo foi aplicado a este projeto, abordando todas as suas diferentes fases do processo de desenvolvimento de um conceito.

- **Opportunity Identification**

  Em várias situações verificou-se que certas aplicações, que de outra forma seriam úteis, se encontravam indisponíveis para certos utilizadores apenas devido a uma barreira linguística.

- **Opportunity analysis**

  Após uma análise de mercado foi possível chegar à conclusão que não existia concorrência direta neste nicho específico de mercado, ou seja, nenhuma outra aplicação respondia de forma direta às necessidades identificadas.

- **Idea Generation and Enrichment**

  Do problema identificado surgiu uma solução, o desenvolvimento de uma aplicação capaz de funcionar em conjunto com a aplicação a traduzir de forma a apresentar uma tradução simples sobre todo o seu texto original.

- **Idea Selection**

  Visto que apenas foi possível chegar a uma ideia com o potencial para solucionar o problema em questão, foi esta a ideia que foi adotada.

- **Concept definition**

  A partir da ideia selecionada foram definidas as tecnologias que irão sustentar o desenvolvimento da solução, assim como determinadas todas as especificações que deverão garantir a satisfação do problema inicial identificado.

2.5.2 Value, Value for the Customer e Perceived Value

Para melhor compreender estes três conceitos será de seguida apresentada uma pequena definição para cada um deles.

- **Value**

  Valor pode ser definido pela necessidade, desejo ou interesse de cada indivíduo pelo produto em questão, e pode variar de acordo com ideais, diferentes culturas, personalidades e comportamentos dos mesmos.

- **Perceived Value**

  Este conceito define-se pelo valor que cada consumidor vê ou reconhece que o produto tem para si mesmo. Desta forma, é compreensível que o valor que o produtor vê no seu produto ou serviço seja visto de forma diferente pelo consumidor final. Por exemplo Lindgreen e Wynstra (Lindgreen & Wynstra, 2005) referem que, por norma, o produtor tem menos sensibilidade ao preço do produto/serviço que o consumidor. É também referido que o consumidor tende a ser mais influenciado pela qualidade do produto.

- **Value for the Customer**

  Value for the Customer, ou valor para o consumidor, é a perceção de vantagem, por parte do cliente, que surge da oferta apresentada por uma determinada organização, sendo resultado de uma redução de esforço ou sacrifício; existência de benefícios; ou uma resultante dos dois (Woodall, 2003).

Relativamente à solução em questão, os benefícios diretos que esta apresenta para o cliente são a capacidade de resolver o problema já referido. O utilizador, com esta solução, é capaz de utilizar de forma prática e natural qualquer aplicação que, devido a uma barreira linguística, de outra forma lhe era incompreensível. Os sacrifícios para o cliente são a necessidade da instalação da aplicação na sua máquina, o que implica algum nível de confiança na aplicação e no seu distribuidor; a aprendizagem do funcionamento desta nova aplicação; e o investimento na compra do produto, caso o utilizador tenha necessidade de utilizar a versão *premium* da aplicação.

2.5.3 Proposta de Valor

Tendo necessidade de utilizar uma aplicação cujo texto apresentado se encontra numa língua estrangeira (cujo utilizador não tem fluência), qual a maneira mais prática de solucionar este problema? A solução a desenvolver propõe solucionar este mesmo problema ao sobrepor à aplicação estrangeira uma tradução automática diretamente sobre o texto original. De forma resumida, a aplicação em questão possibilita a compreensão da interface de qualquer programa que se apresente numa língua incompreensível para o utilizador.

Os benefícios que esta solução traz ao utilizador por oposição a uma solução de tradução manual, com recurso a ferramentas de edição de imagem (como o *Microsoft Paint*), identificação de texto (*ABBYY FineReader*) e *machine translation* (tal como o *Google Translator*), são os seguintes:

- A tradução é realizada de forma muito rápida, na maioria dos casos dentro de frações de segundo, o que permite que o utilizador evite perdas de tempo e produtividade, focando-se assim nos aspetos da aplicação que realmente importam.
- Os segmentos de texto da aplicação são identificados de forma automática, após a realização de uma calibração simples baseada em ajustes de *sliders* com *feedback* visual. Isto permite reduzir muito significativamente o esforço por parte do utilizador.
- A tradução é apresentada diretamente sobre a aplicação a traduzir, o que torna a interação mais intuitiva e elimina a necessidade de alternar o foco do utilizador entre a aplicação alvo e qualquer outra ferramenta dedicada de tradução.

Outro beneficio associado a esta aplicação é sua capacidade de aceitar alterações e melhoramentos desenvolvidos por terceiros, a partir da simples instalação de *plugins*. Esta funcionalidade permite que caso a aplicação não se ajuste ao caso específico de um utilizador, esta tenha potencial para ser customizada de forma a cumprir estes novos requisitos.

Por fim, é importante ainda referir que, salvo poucas exceções especializadas para um tipo específico de aplicações, esta solução é única dentro da sua área, visto que é capaz de atuar genericamente sobre uma vasta variedade de aplicações.

2.5.4 Modelo Canvas

Na Figura 13 é possível ver o modelo *canvas* que foi definido para uma potencial implementação comercial da solução proposta. Neste modelo foram preenchidas as categorias características de um modelo *canvas* da seguinte forma:

Parceiros Chave

- A *Microsoft*, devido à necessidade da realização de acordos de utilização da sua ferramenta de tradução;
- A *Yandex*, novamente devido à sua ferramenta de tradução;
- A *Google*, caso a expansão da solução beneficie da sua ferramenta de tradução;
- A *ABBYY*, caso a expansão da solução beneficie da sua ferramenta de *OCR,* o *FineReader.*

Atividade Chave

- Baseia-se apenas no desenvolvimento e manutenção (sob a forma de atualizações) da aplicação em questão.

Recursos Chave

- A Equipa de desenvolvimento, constituída inicialmente por um único indivíduo mas com capacidade de expansão dependendo da rentabilidade da solução e carga de trabalho associada à sua expansão e manutenção.

Proposição de Valor

- Permite que qualquer utilizador, independentemente das línguas que compreende, seja capaz de interagir corretamente com a interface de qualquer programa que se apresente numa língua desconhecida;
- Apesar de ser possível realizar uma tradução manual da aplicação desconhecida, esta solução é capaz de realizar a tradução de forma automatizada, numa fração do tempo que levaria a realiza-la manualmente; e sem a necessidade de tirar o foco da aplicação alvo;

Figura 13 - Modelo Canvas de uma possível implementação comercial da solução

- Caso a solução apresentada não cubra todas as necessidades de um utilizador em específico, esta oferece a capacidade de customização e expansão por terceiros, através da simples inclusão de um ou vários *plugins*.

Relação com o Cliente

- O modelo de relação com o cliente é baseado no modelo de *self-service*, sendo que todas as ferramentas e tutoriais que o consumidor possa precisar assim como respostas a dúvidas comuns se encontram na página *web* da aplicação;

Canais de Distribuição

- Assim como com a relação com o cliente, o canal de distribuição preferencial é a própria página *web* da aplicação.

Segmentos Alvo

- Utilizadores frequentes de aplicações específicas das quais a interface se encontre numa língua incompreensível para o utilizador e sem previsão de tradução futura;
- Fãs de alguns nichos dentro do mercado de videojogos, especificamente jogos de origem chinesa, japonesa ou russa, que nunca chegam a ser traduzidos;
- Indivíduos no processo de aprendizagem de uma nova língua podem beneficiar de uma aplicação capaz de realizar traduções em tempo real para, por exemplo, ajudar a completar falhas no seu vocabulário;
- Algumas páginas *web* utilizam frequentemente imagens com texto, por oposição a texto editável. Esta prática impede ferramentas como o *Google Translator* de traduzir a página por completo. A solução proposta não tem essa limitação, e desta forma tem potencial para atrair utilizadores da *Web* com interesse em páginas estrangeiras;

Estrutura de Custos

- Custos associados ao desenvolvimento da solução e *software* relacionado, incluindo os custos associados à equipa de desenvolvimento;
- Custos associados à expansão da aplicação, com a incorporação das *frameworks* comerciais *Google Translato*r e *ABBYY FineReader*.

Fontes de Receita

- Vendas da subscrição mensal da versão Premium, que oferece mais opções e maior capacidade mensal de tradução ao utilizador;
- Doações associadas à versão gratuita da solução.

2.5.5 Modelo de Verna Allee

Segundo a metodologia de análise de Verna Allee, é possível realizar uma análise de valor através de um sistema de redes de valor (*Value Networks*), tendo em conta fatores como os padrões de troca, o impacto de transações de valor, todas as trocas (tangíveis ou intangíveis), assim como todas as dinâmicas de criação e aproveitamento de valor. O desenvolvimento de um mapa de redes de valor (Figura 14), que inclui todas as trocas e interações relativas ao negócio, é essencial tendo em conta que facilitará processos de análise de valor tanto a níveis operacionais, táticos, como até a níveis estratégicos e macroeconómicos (Allee, s.d.).

Ambos o modelo de Verna Allee e a cadeia de Porter permitem, ao analisar em detalhe as dependências de uma organização, maximizar o valor do/s produto/s ou serviço/s que a organização oferece ao otimizar as interações com os diferentes grupos que interagem com a organização (tanto de forma interna como externa) e o equilíbrio entre os diferentes *workflows*.

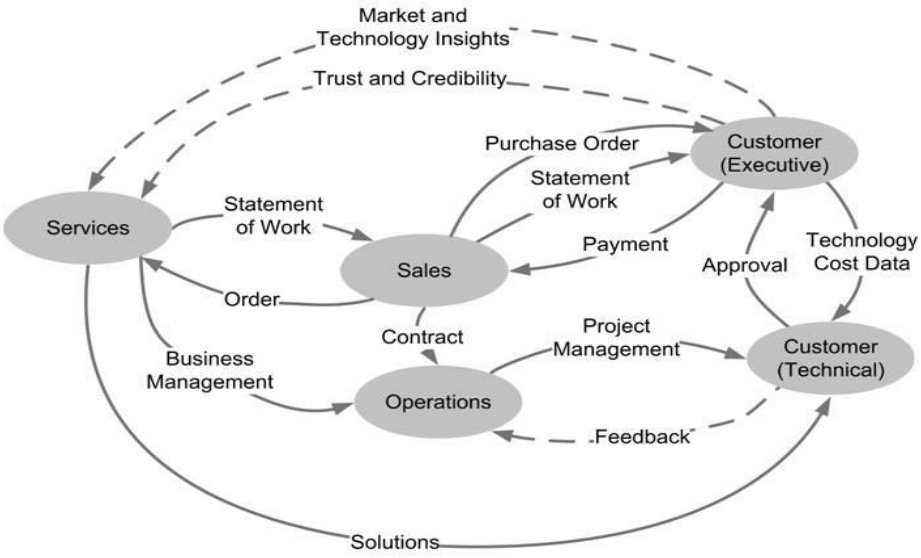


Figura 14 - Mapa de Value Network Analysis (Anklam, s.d.)

2.5.6 Análise de Valor Multicritério

Neste subtópico foi realizada uma análise multicritério, utilizando o método AHP, de forma a selecionar corretamente a melhor solução de tradução a integrar neste projeto. Para isto, foram inicialmente definidos os critérios a usar para a avaliação das três soluções destacadas, o *Google Translator*, o *Microsoft Translator* e o *Yandex Translate*. Os critérios definidos foram os seguintes:

- Custo de utilização;
- Facilidade de uso;
- Número de carateres disponíveis;
- Qualidade de tradução;
- Tempo de resposta;
- Tipo de *API*
- Número de línguas disponíveis;

De seguida estes critérios foram comparados entre eles resultando na matriz apresentada na **Erro! A origem da referência não foi encontrada.**.

Após definidas as percentagens para cada critério, foram definidas as percentagens para cada uma das opções possíveis dentro de cada um dos critérios (apresentadas na Tabela 2). Os resultados associados a esta fase do processo encontram-se na Tabela 3, onde é visível o resultado individual para cada uma das opções. Note-se ainda que, tendo em conta que as três soluções a ser avaliadas partilham tempos de resposta muito semelhantes (ao ponto desta

Tabela 1 Matriz de Comparação de Critérios

Matrix		Custo	Facilidade de Uso	Nº de Caracteres	Q. de Tradução	T. de Resposta	Tipo de API	Nº de Línguas	0	0	0	normalized principal Eigenvector
		1	2	3	4	5	6	7	8	9	10	
Custo	1	-	5	7	2	8	2	5	-	-	-	34.94%
Facilidade de Uso	2	1/5	-	2	1/5	6	3	1	-	-	-	10.37%
Nº de Caracteres	3	1/7	1/2	-	1/7	6	2	3	-	-	-	9.06%
Q. de Tradução	4	1/2	5	7	-	6	8	2	-	-	-	30.21%
T. de Resposta	5	1/8	1/6	1/6	1/6	-	1/2	1/2	-	-	-	2.73%
Tipo de API	6	1/2	1/3	1/2	1/8	2	-	1	-	-	-	5.95%
Nº de Línguas	7	1/5	1	1/3	1/2	2	1	-	-	-	-	6.74%
0	8	-	-	-	-	-	-	-	-	-	-	0.00%
0	9	-	-	-	-	-	-	-	-	-	-	0.00%
0	10	-	-	-	-	-	-	-	-	-	-	0.00%

diferença não ser significativa no contexto deste processo), e o mesmo tipo de *API,* o resultado individual para estas componentes foi definido como 0% de forma a não afetar o resultado final.

Por fim foram somados os resultados de cada uma das parcelas das suas respetivas soluções. Após observar os resultados apresentados na Tabela 4, foi possível concluir que a solução que melhor se enquadra no contexto deste projeto é a apresentada pela *Microsoft*, cuja classificação final obtida é de 59.1%. A solução da *Yandex*, aparece logo de seguida com uma classificação de 54.3%, o que a torna ainda viável como segunda opção. Por fim, a solução da *Google* recebe uma classificação de 41.3% principalmente devido à penalização que sofre devido ao peso associado ao critério de custo. Caso o custo de utilização do serviço da *Google* não fosse problemático, esta seria, no entanto, a solução preferencial.

Tabela 2 - Características associadas a cada solução de tradução

Solução	Custo	Requisitos de Utilização	Nº de Carateres Disponíveis	Qualidade de Tradução	Tempo de Resposta	Tipo de API	Nº de Línguas Disponíveis
Google Translator	$20/mês	*API Key*	1,000,000/mês	+++	-	*REST*	*90+*
Microsoft Translator	Sem custos	*Conta Microsoft Azure*	2,000,000/mês	+++	-	*REST*	*60*
Yandex Translate	Sem custos	*API Key*	10,000,000/mês	++	-	*REST*	*89*

Tabela 3 - Tabela AHP para o cálculo da percentagem por critério de avaliação

Custo	Sem Custo		$20/mês por utilizador		Resultado por Fator	Resultado Individual
Sem Custo	-		9		90%	**0.3494*0.9 = 31%**
$20/mês por utilizador	1/9		-		10%	**0.3494*0.1 = 3.5%**
Facilidade de Uso	API key		Conta Microsoft Azure		-	
API key	-		9		90%	**0.1037*0.9 = 9.3%**
Conta Microsoft Azure	1/9		-		10%	**0.1037*0.1 = 1%**
Número de Carateres	1M char/mês	2M char/mês		10M char/mês	-	
1M char/mês	-	1/2		1/9	8%	**0.0906*0.08 = 0.7%**
2M char/mês	2	-		1/5	16%	**0.0906*0.16 = 1.4%**
10M char/mês	9	5		-	76%	**0.0906*0.76 = 6.9%**
Qualidade de Tradução	+++		++		-	
+++	-		5		83%	**0.3021*0.83 = 25%**
++	1/5		-		17%	**0.3021*0.17 = 5.1%**
Tempo de Resposta	-				-	
-	N/A				0%	**0.0273*0 = 0%**
Tipo de *API*	*REST*					
REST	N/A				0%	**0.0595*0 = 0%**
Número de Línguas Disponíveis	90+	89	75	60	-	
90+	-	2	2	3	42%	**0.0674*0.42 = 2.8%**
89	1/2	-	3	3	29%	**0.0674*0.29 = 2%**
75	1/2	1/2	-	2	18%	**0.0674*0.18 = 1.2%**
60	1/3	1/3	1/2	-	11%	**0.0674*0.11 = 0.7%**

Tabela 4 - Tabela AHP para o cálculo da percentagem resultado de cada solução

Soluções	Custo	Facilidade de uso	Número de Carateres	Qualidade de Tradução	Tempo de Resposta	Tipo de *API*	Número de Línguas Disponíveis	**TOTAL**
Google	3.5%	9.3%	0.7%	25%	0.0%	0.0%	2.8%	**41.3%**
Microsoft	31%	1.0%	1.4%	25%	0.0%	0.0%	0.7%	**59.1%**
Yandex	31%	9.3%	6.9%	5.1%	0.0%	0.0%	2.0%	**54.3%**

2.6 Conclusão

Este capítulo teve como foco principal o estudo do estado da arte visto que este engloba o estudo das ferramentas que irão inspirar o desenvolvimento da solução e, em alguns casos, fazer diretamente parte da mesma. Dentro deste tópico foram abordados métodos de segmentação de texto que terão utilidade durante o processo de identificação do posicionamento de todo o texto na aplicação alvo. Estes métodos são também úteis visto que facilitam e melhoram os resultados do processo de *OCR*. Do mesmo modo, foram analisadas as soluções atuais de *OCR* e de *machine translation* mais relevantes, sendo que todas estas serão avaliadas nos capítulos 3.2.2 e 3.2.3 respetivamente. Para além disto, foi feito um estudo das formas mais apropriadas de desenvolvimento de *overlays* sobre aplicações gráficas onde, novamente, serão apresentadas mais conclusões no capítulo 3.2.4. Para finalizar o capítulo, foi desenvolvido um estudo em maior detalhe sobre a relevância e valor associados à solução proposta.

3 Avaliar Soluções e Abordagens Existentes

Este capítulo será dividido em duas partes. A primeira será reservada a uma análise comparativa, entre a abordagem proposta e as abordagens existentes em aplicações semelhantes, anteriormente referidas no subcapítulo *Estado da Arte em Abordagens Existentes*. A segunda parte será destinada à avaliação das tecnologias mais relevantes, novamente, mencionadas no subcapítulo "Estado da Arte em Tecnologia Relevante". É nesta parte onde cada uma das componentes analisadas serão filtradas de acordo com as características que mais beneficiam o projeto.

3.1 Avaliação das Abordagens Existentes

Neste subtópico são discutidos os pontos fortes e fracos de cada uma das soluções apresentadas anteriormente no subtópico *Estado da Arte em Abordagens Existentes*. São também identificadas as características que distinguem cada uma das soluções e como estas se diferenciam da solução proposta.

3.1.1 Word Lens

Esta aplicação, apesar de identificar e substituir o texto original pela sua versão traduzida, tem por base um sistema de realidade aumentada, tirando partido da câmara do dispositivo móvel onde se encontra instalada. Para além disto, de momento encontra-se limitada a 6 línguas, português; espanhol; italiano; francês; alemão e russo, e todas estas apenas traduzem de e para inglês.

Excluindo a possibilidade de apontar o dispositivo móvel para o monitor onde a aplicação a traduzir se encontra, esta aplicação não é viável para traduzir interfaces de outras aplicações.

Além disso trata-se de uma aplicação *closed source*, o que significa que não é possível ter acesso simples à forma específica como esta solução foi implementada.

3.1.2 WayGo

O *WayGo*, estando na mesma categoria que o *Word Lens*, foi desenvolvido com um propósito diferente do problema proposto. O seu objetivo nunca foi traduzir aplicações, mas sim texto em imagens naturais do mundo real. Infelizmente, tal como o *Word Lens*, esta aplicação é comercial, ou seja, não é possível visualizar de forma simples a forma concreta como foi implementada.

No caso do *WayGo,* é também importante referir que é apenas capaz de identificar e traduzir japonês, chinês e coreano para inglês.

3.1.3 Visual Novel Reader (VNR)

Esta é provavelmente a aplicação que mais se assemelha à proposta de solução em questão, visto que utiliza *machine translation* e apresenta o resultado sobreposto à janela da aplicação alvo.

No entanto, para além de não apresentar os resultados sobrepostos diretamente sobre a área do texto traduzido e ser especializado na tradução de *visual novels* japonesas, esta aplicação utiliza *text hooks* para obter o texto a traduzir, o que não lhe permite obter de forma automática o conteúdo de botões ou *labels* cujo texto se encontre num formato de imagem (o utilizador pode manualmente selecionar o texto a reconhecer em *MODI OCR*). Esta restrição não lhe permite realizar traduções automatizadas nas interfaces com o utilizador. Apesar destas limitações, *text hooking* tem a vantagem de ser muito mais preciso nos seus resultados que os métodos baseados em *OCR*. Desta forma, pode ter interesse implementar um sistema semelhante mas com um funcionamento cooperativo entre o sistema de *OCR* existente e o de

text hooking, aumentando assim a precisão dos resultados sem as desvantagens anteriormente referidas.

3.1.4 Translator Aggregator & Interactive Text Hooker

Esta solução distancia-se ainda mais do problema que o *VNR*, pois não chega sequer a apresentar os resultados na janela da aplicação alvo, mas sim numa janela própria. Isto obriga a que a aplicação principal corra sempre em janela, de forma a não obstruir a visibilidade para a janela de tradução, o que está longe de constituir uma solução ideal.

Para além do problema já referido, esta solução não é adequada para utilizadores menos experientes, devido ao processo de interação entre aplicações; e a interface do *ITH* ser muito pouco usável, pois requer um conhecimento relativamente avançado de alguns termos técnicos.

3.2 Avaliação das Tecnologias Relevantes

Tendo em conta os critérios que serão definidos nos subtópicos seguintes, é possível comparar as soluções analisadas em *Estado da Arte em Tecnologia Relevante* de forma ter uma ideia mais concreta das tecnologias que poderão vir a ser usadas, assim como as limitações que estas impõem ao projeto a desenvolver.

3.2.1 Deteção e Segmentação de Texto em Imagens

Para que as soluções apresentadas em *Deteção e Segmentação de Texto em Imagens* possam ser avaliadas comparativamente entre elas, foram definidos dois fatores que devem ser tidos em conta ao caracterizar a melhor solução. A velocidade de execução (em resoluções próximas de

1280x720 píxeis) e o nível de precisão dos resultados (capacidade de identificar a maior quantidade de texto possível com a menor quantidade de falso-positivos). A velocidade de execução é importante no contexto deste projeto visto que afeta diretamente a perceção do utilizador relativamente à fluidez desta aplicação. A precisão de resultados tem importância também pela mesma razão: a velocidade de resposta. Quanto maior o número de falso-positivos apresentados pelo algoritmo de segmentação, maior a quantidade de segmentos de imagem que serão processados pelo *OCR* desnecessariamente, o que desperdiça bastante tempo de execução. Por outro lado, a precisão destes algoritmos é também de grande importância visto que, caso exista texto que escape ao processo de segmentação, este terá de ser manualmente assinalado pelo utilizador, processo que deve ser evitado.

Tendo em conta estes fatores, é possível excluir inicialmente todos os algoritmos com uma abordagem ***texture-based,*** visto que, de acordo com (Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010), esta abordagem sofre de falta de precisão e baixo desempenho, devido à necessidade destes algoritmos analisarem a imagem em múltiplas diferentes escalas para que possam ser compatíveis com *fonts* de diversas dimensões.

Consequentemente, algoritmos ***region-based*** deverão ser o foco principal deste estudo. Dentro destes foi dada especial atenção aos algoritmos mencionados em 2.4.1, pois todos eles abordam o problema focando-se em diferentes características visuais do texto, por oposição ao seu *background*. Todas estas soluções tentam segmentar o texto numa imagem de forma completamente automatizada. No entanto, no contexto deste projeto, e de forma a obter melhores resultados, é permitida a realização de calibrações simples (ajustes rápidos de valores em *sliders* com *feedback* visual) de forma ao algoritmo se poder adaptar o melhor possível a cada aplicação alvo. É assim possível simplificar o algoritmo a utilizar de maneira a que este seja menos computacionalmente complexo e mesmo assim tirando partido de uma combinação de algumas das técnicas apresentadas nas soluções estudadas.

3.2.2 Optical Character Recognition

O sistema ideal de *OCR* para o projeto proposto é um capaz de um alto grau de **precisão**; sem **custos** de utilização; com uma capacidade **rápida de resposta** e capacidade de reconhecer o **maior número possível de línguas**, com especial ênfase nas línguas destacadas para este projeto (Inglês, Português, Espanhol, Francês, Alemão, Japonês, Chinês Simplificado, Coreano e Russo).

Tabela 5 - Comparação das caraterísticas das soluções de OCR estudadas

Designação	Custo	Precisão relativa	Tempo de Resposta	Inclui Línguas Destacadas
Tesseract 3.04	Sem custos (*Apache License 2.0*)	+++	-	Sim
A9T9's Free OCR	Sem custos (*GPL V2 License*)	++	-	Sim
NHocr	Sem custos (*Apache License 2.0*)	+++	-	Não (Apenas Japonês)
Puma.NET (*CuneiFrom*)	Sem custos (BSD *License*)	++	-	Não (Não inclui carateres asiáticos)
GOCR	Sem custos (*GNU Public License*)	+	-	Não (Apenas carateres latinos)
ABBYY FineReader	Licença Comercial	+++	-	Sim

Tendo em conta os valores apresentados na Tabela 5, é possível excluir de imediato certas soluções, nomeadamente o *ABBYY FineReader* pelo seu custo de utilização, e o *GOCR* por apenas ser capaz de identificar carateres latinos e pelo seu grau de precisão inferior às restantes soluções. O *Puma.NET* falha ao não ser capaz de reconhecer carateres asiáticos que, por norma, são muito mais complicados de reconhecer, em parte devido à sua complexidade e diversidade (são

regularmente utilizados cerca de 7000 carateres chineses (Garza, 2017)). Sobram o *Tesseract,* o *A9T9's Free OCR* e o *NHocr*. Destes três, a solução que se destaca é a apresentada pelo *Tesseract*, devido ao seu alto nível de precisão mesmo comparativamente à solução comercial da *ABBYY* (Heliński, et al., s.d.) e a sua vasta coleção de ficheiros de treino para diferentes línguas. No entanto, ambos, *A9T9's Free OCR* e *NHocr* poderiam ter uso como "segunda opinião", de forma a melhorar os resultados obtidos pelo módulo de *OCR*, em particular o *NHocr,* visto ser especializado numa língua em que os resultados obtidos pelas restantes soluções tendem a ser inferiores ao expectável.

3.2.3 Machine Translation

O módulo de *machine translation* é um constituinte essencial para o funcionamento correto deste projeto e a qualidade de tradução da solução selecionada influência de forma muito significativa a qualidade final do mesmo.

Os principais requisitos para este projeto são a capacidade de tradução de grandes quantidades de dados (no mínimo 1M carateres por mês); qualidade de tradução; e tempo de resposta. Para além disto, é necessário que os custos e requisitos de utilização sejam minimizados.

Tabela 6 - Comparação das caraterísticas das soluções de tradução estudadas

Solução	Custo	Requisitos de Utilização	Nº de Carateres Disponíveis	Qualidade de Tradução	Tempo de Resposta	Tipo de API	Nº de Línguas Disponíveis
Google Translator	$20/mês	*API Key*	1,000,000/mês	+++	-	*REST*	*90+*
Microsoft Translator	Sem custos	*Conta Microsoft Azure*	2,000,000/mês	+++	-	*REST*	*60*
Yandex Translate	Sem custos	*API Key*	10,000,000/mês	++	-	*REST*	*89*
BabelFish	Sem custos	N/A	N/A	+++	N/A	Não Disponível	*75*

Com base na informação disponibilizada na Tabela 6 é possível fazer uma comparação rápida dos pontos fortes e fracos de cada uma das soluções de *machine traslation* analisadas.

A solução da *Systran*, o *BabelFish*, pode ser descartada logo desde o início por não disponibilizar uma *API* para o desenvolvimento de projetos como este. A solução da *Google*, apesar de ser provavelmente a mais conhecida, para uma versão comercial deste projeto requer pagamentos mensais, no entanto é viável durante a fase de prototipagem. Restam as soluções da *Microsoft* e da *Yandex.* Ambas se apresentam como opções sólidas para incorporar neste projeto. A da *Microsoft* no entanto requer a criação de uma conta *Microsoft Azure*, o que é um processo trabalhoso para o utilizador final, devido aos requerimentos associados à criação destas contas. A solução da *Yandex*, por outro lado, destaca-se pela facilidade de obtenção das *API Keys* necessárias para a sua integração e o grande número de carateres que disponibiliza mensalmente, o que torna viável a sua integração neste projeto.

3.2.4 *Overlay* sobre Aplicações Gráficas

Para comparar as vantagens e desvantagens de cada metodologia de desenvolvimento de *overlays* é importante analisar fatores como a facilidade de implementação e manutenção, a capacidade de funcionar em *fullscreen mode*, a velocidade de resposta e a segurança do utilizador.

Em termos de facilidade de implementação e manutenção, a solução preferencial será o *overlay* em *WinAPI/GDI*, visto que é tecnicamente mais simples de implementar e não requer uma implementação específica para cada versão de *DirectX* ou *OpenGL*, ao contrário de uma solução baseada em *DLL injections*. Por outro lado, esta implementação tem a desvantagem de não ser compatível com aplicações *fullscreen* e ter um tempo de resposta inferior à sua contraparte. Em termos de segurança, a solução baseada em *WinAPI/GDI* é preferível, visto que não requer permissões de administração do sistema para funcionar. Uma implementação baseada em *DLL injections*, para além de ter potencial para causar falso-positivos no antivírus do utilizador, caso seja utilizada em aplicações *online*, como vídeo jogos, pode ser vista como uma tentativa ilegal

de modificação da aplicação e resultar, em casos extremos, em restrições de acesso do utilizador às suas contas.

Desta forma, a utilização de *DLL injection* será reduzida (ou até inexistente) e o utilizador deve ser devidamente informado dos riscos da sua utilização, caso esta seja implementada.

Resumindo, sempre que possível deverá ser utilizado o método baseado em *WinAPI/GDI*, visto que face ao tempo disponível para o desenvolvimento da aplicação final; e aos problemas de segurança inerentes às soluções baseadas em *DLL injection*, este último método é pouco adequado às condições atuais deste projeto.

3.3 Conclusão

Os pontos importantes a retirar deste capítulo são todos eles baseados na escolha das soluções a utilizar para o desenvolvimento da solução proposta. Destas destacam-se na área de *OCR* o *tesseract*; em *machine translation* o *Yandex Translate*; e na área de *overlays* gráficos foi decidido minimizar o uso de *DDL injections* por questões de segurança. Na área de segmentação de texto chegou-se à conclusão que, tirando partido das técnicas utilizadas pelos algoritmos estudados, será possível desenvolver um método com melhores resultados utilizando um sistema simples de calibração, que o utilizador deve ajustar para cada aplicação. Por fim, ao estudar o funcionamento de ferramentas como o *VNR* e *ITH,* foi decidido que a utilização de um sistema de *text hooking*, em casos onde este sistema é aplicável, seria a melhor solução para obter melhores resultados na maioria dos casos.

4 Design da solução

Este capítulo terá como foco principal os detalhes de *design* planeados e desenvolvidos para a aplicação *DOT*. Para que estes detalhes possam ser melhor compreendidos foram desenhados dois diagramas, que irão servir de ponto de suporte principal para a explicação da estrutura desenvolvida.

4.1 *Design* da Aplicação

Foi desenvolvido um diagrama de classes que permite visualizar rapidamente toda a estrutura da aplicação, no seu presente estado (Figura 15, Figura 16).

A modularidade neste projeto é assegurada pela forma como o *design* desta solução separa os quatro módulos principais (Deteção de Texto, *OCR*, Tradução e *Overlay*) através da utilização de um padrão *strategy*. Desta forma, ao encapsular a implementação destes módulos, é possível alterá-los de forma muito simples. A utilização deste padrão permite também que todos estes módulos possam ser desenvolvidos de forma externa ao projeto, através de *DLLs*, substituindo a implementação *standard* por uma nova, integrada como um *plugin*. Ao aceitar *plugins*, esta solução ganha a capacidade de aceitar potenciais atualizações ou melhoramentos desenvolvidos por terceiros, sendo as únicas restrições impostas pelas interfaces associadas a cada módulo (*I_TextDetection, I_OCR, I_Translation, I_Overlay* e *I_ModuleSettings*).
Relativamente à interface com o utilizador, toda ela está ligada a um controlador que, por sua vez, faz a comunicação com restante parte da aplicação. Assim, este controlador permite desacoplar e facilitar a interação (*facade*) da interface com a restante implementação. Desta forma, toda uma nova *UI* pode ser adaptada ao sistema com uma simples mudança de controlador, sem que para isso seja necessário modificar o *back-end* da solução.

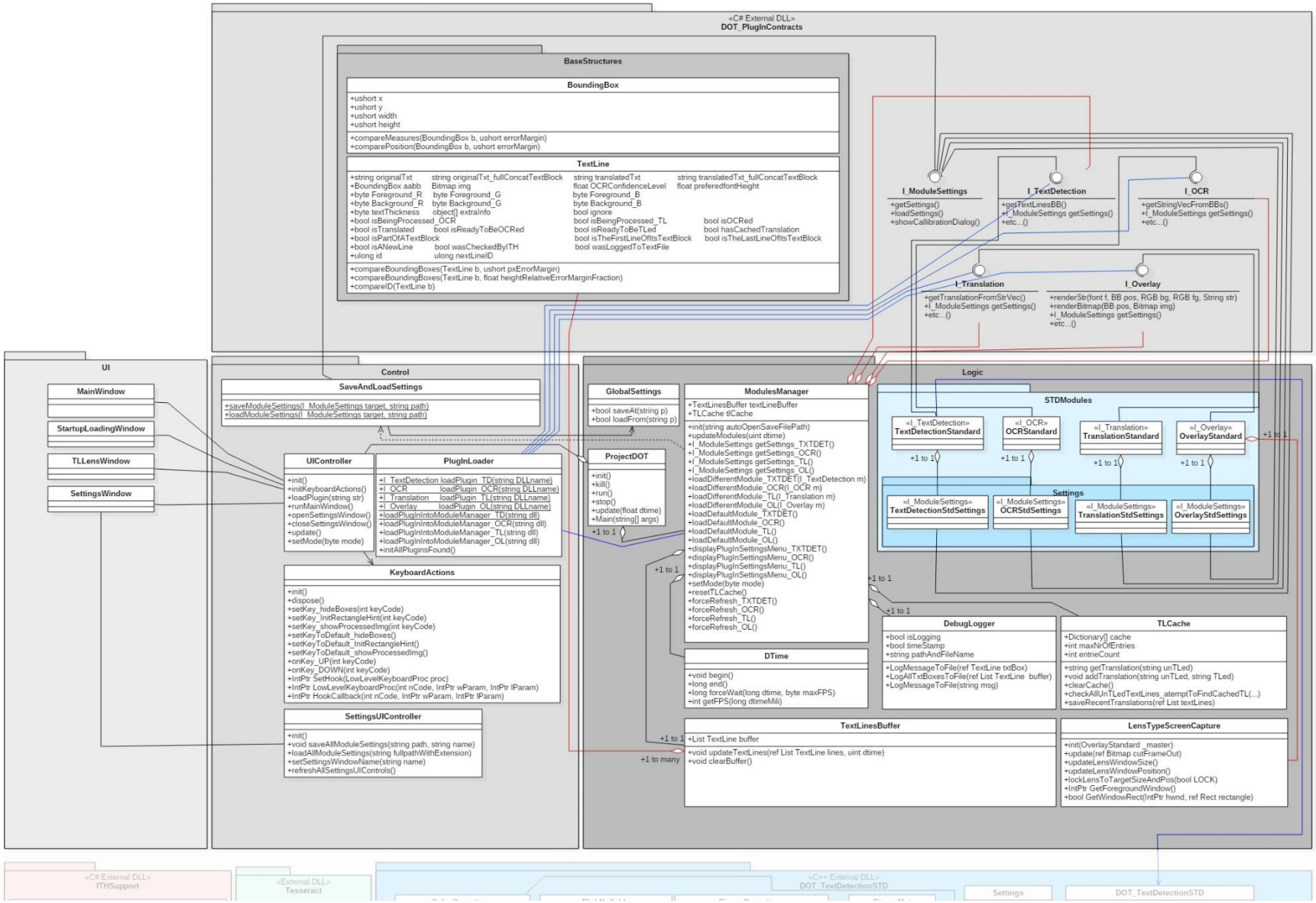


Figura 15 - Diagrama de Classes (Parte 1))

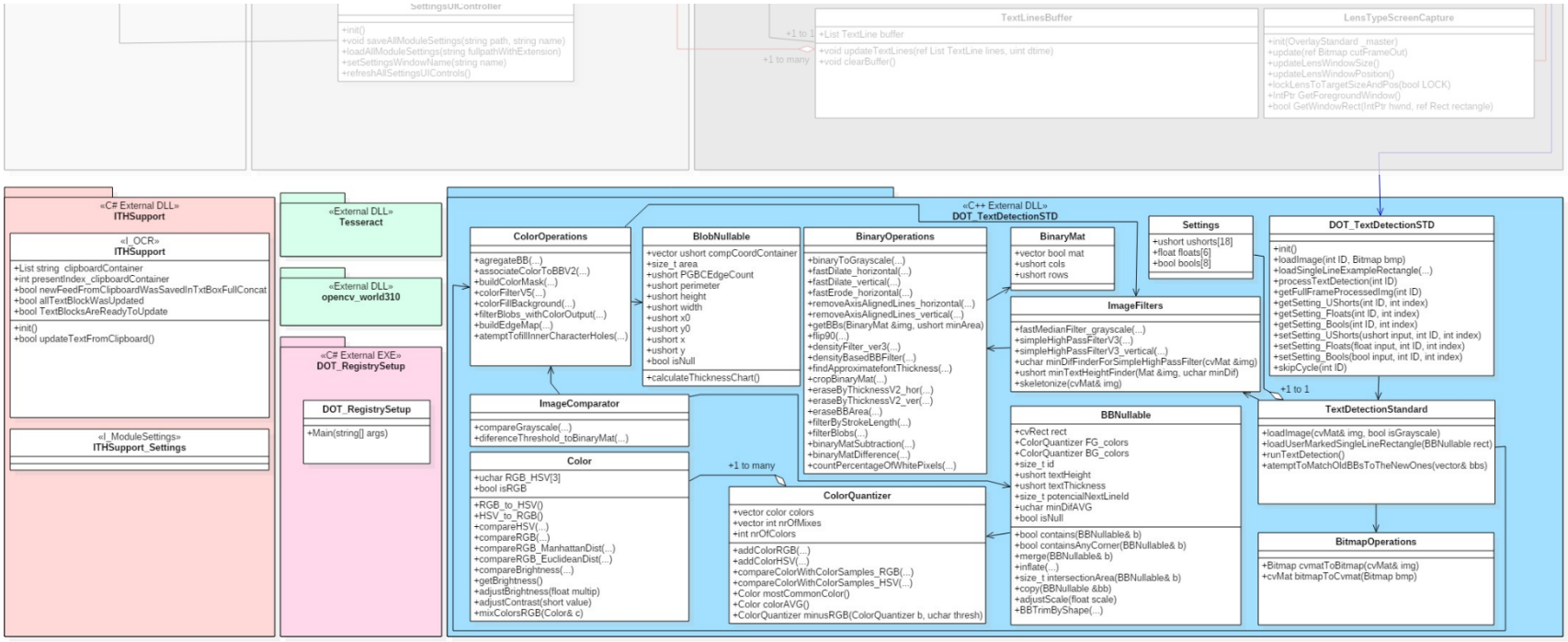


Figura 16 - Diagrama de Classes (Parte 2)

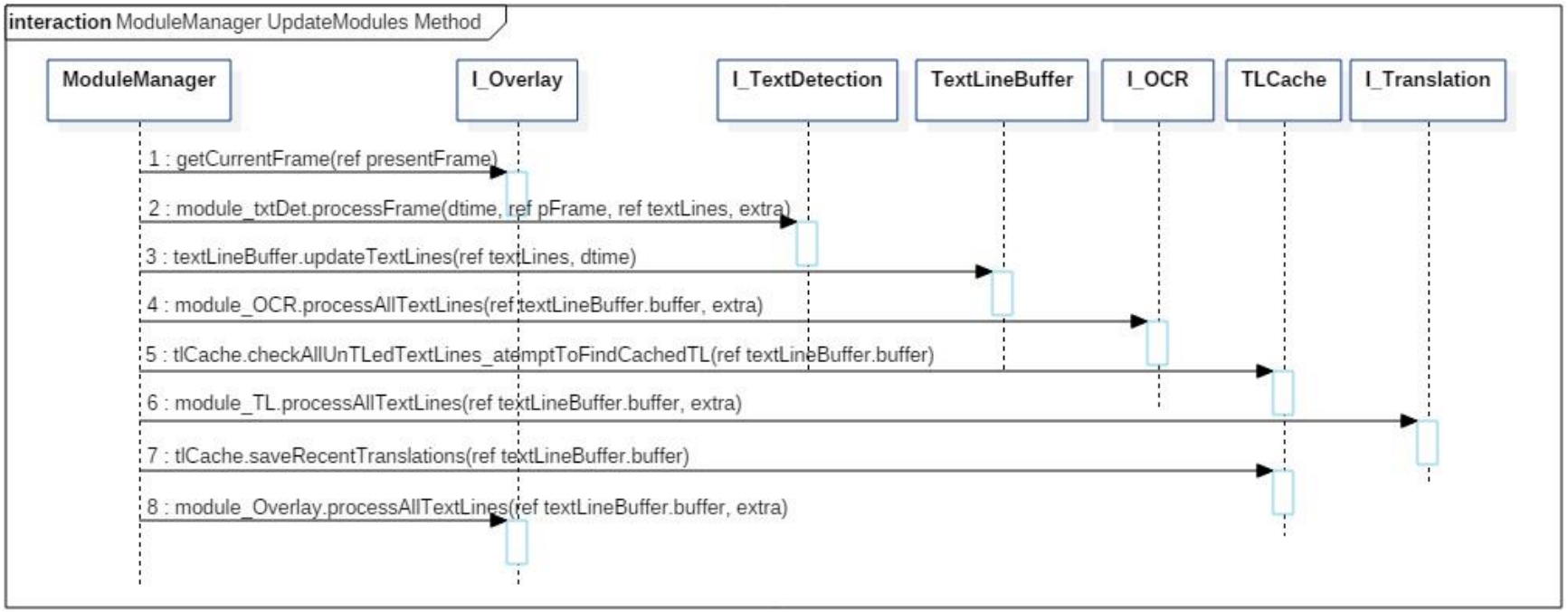


Figura 17 - Diagrama de Sequência Simplificado do Pipeline de Tradução

Por fim, todos os módulos principais são orquestrados por um *ModuleManager* que estrutura o *pipeline* de tradução da forma apresentada no diagrama de sequência simplificado da Figura 17. Inicialmente, a imagem obtida da *canvas* da aplicação é enviada para o módulo de deteção e segmentação de texto (*I_TextDetection*) que, depois de processar a imagem, retorna as caixas (ou *TextLines*) das posições prováveis do texto na imagem. As imagens contidas nestas caixas são de seguida enviadas para o módulo de *OCR* (*I_OCR*) que irá tentar identificar os carateres nelas contidos. Os resultados do *OCR* são então associados à sua respetiva *TextLine*. O conteúdo de cada *TextLine* é então traduzido (individualmente ou em grupo, no caso de blocos de texto), verificando primeiro se a tradução já foi realizada anteriormente (verificando a *TLCache*) e, caso isto não se verifique, esta interage com o módulo de tradução (*I_Translation*) de forma a pedir uma tradução para o seu conteúdo. Assim que a tradução termine e seja adicionada à *cache* de tradução, o resultado é enviado para o módulo de *overlay* (*I_Overlay*), que deverá apresentar a tradução sobre a janela da aplicação alvo.

4.2 Conclusão

Neste capítulo foi possível observar que uma grande parte do *design* deste projeto foi construído de forma a permitir que os seus pilares fundamentais (Segmentação de Texto, *OCR*, Tradução e *Overlay* gráfico) possam ser facilmente atualizados e até mesmo completamente substituídos, tanto pelo próprio *developer* da solução como por terceiros, sob a forma de *plugin*s.

Através dos dois diagramas apresentados foi também possível ter uma ideia da estrutura de funcionamento atual da aplicação.

5 Desenvolvimento da Solução

Este capítulo irá explicitar de forma aprofundada todas as principais técnicas utilizadas e desenvolvidas durante o processo de desenvolvimento da aplicação *DOT*. O capítulo será dividido em seis partes, uma para cada um dos quatro módulos principais, e as restantes duas para uma referência rápida ao funcionamento e tecnologias por detrás da *UI* e sistema de *plugins*. O maior ênfase será dado ao funcionamento do módulo específico de deteção de texto, visto este ser o módulo mais complexo e desenvolvido, por inteiro, especificamente para esta aplicação.

5.1 Deteção e Segmentação de Texto

A grande maioria dos sistemas deteção e segmentação estudados (secção 2.4.1) foram desenvolvidos com o intuito de encontrar texto num ambiente natural, onde linhas de texto podem aparecer nos mais diversos ângulos, dimensões e gradientes de cores. Este módulo foi construído de raiz de forma a responder às necessidades específicas desta aplicação, ou seja, encontrar e segmentar (com resultado em imagem binária) texto presente numa aplicação alvo o mais rápido possível, de forma a reduzir ao máximo o tempo de resposta. Desta forma, como o texto presente numa qualquer aplicação tende a aparecer perfeitamente horizontal, com uma variedade relativamente reduzida de tamanhos e com um preenchimento de cores constante (visto não existir problemas de luminosidade tais como os presentes num cenário natural) foi possível aproximar este problema de uma forma diferente dos métodos estudados, mantendo no entanto muitas das suas características base. Uma outra característica que é importante ter em conta em algumas aplicações (tais como legendas em *video players*) é a complexidade do *background*, que sendo de rara ocorrência num cenário natural, é relativamente comum em muitas aplicações, obrigando ao desenvolvimento de um algoritmo capaz de diferenciar e filtrar componentes textuais, de qualquer elemento pertencente ao *background*.

O método desenvolvido pode ser dividido em 13 partes:

- Calibração inicial;
- Filtragem por Contraste;
- Remoção de Componentes Lineares Puramente Horizontais ou Verticais
- Filtragem por Densidade de Contraste;
- Dilatação Horizontal;
- Determinação de Bounding Boxes;
- Determinação da Cor Estimada das Componentes Textuais;
- Agregação e Filtragem Heurística das Bounding Boxes Definidas;
- Filtragem Binária Baseada na Cor Estimada do Texto;
- (Opcional) Filtragem pelas Características Individuais de Cada Blob;
- Comparação com as *Bounding Boxes* da *frame* anterior;
- Identificação de Blocos de Texto;
- Construção das *DOT_PlugInContracts*::TextLine.

5.1.1.1 Calibração inicial

Esta calibração pode ser definida manualmente, ao interagir com os *sliders* do menu de calibração, ou de forma semiautomática ao estabelecer uma área retangular à volta de uma única linha de texto (Figura 18). Esta área é analisada de forma a obter uma média controlada das áreas de alto contraste encontradas através de múltiplas passagens horizontais sobre a secção definida da versão em *grayscale* da imagem original. É também aqui que a altura prevista do texto a procurar é calculada.

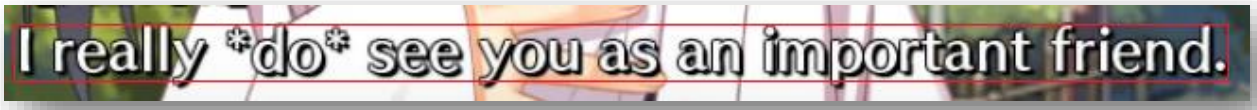


Figura 18 - Calibração inicial DOT

5.1.1.2 Filtragem por Contraste

A filtragem por contraste é baseada em dois valores: o nível mínimo de contraste e a espessura média do texto a segmentar. Ambos estes valores são definidos durante a calibração inicial descrita em 5.1.1.1. Esta filtragem é realizada através de um *scan* horizontal seguido de um *scan* vertical da imagem original em *grayscale*. Estes *scans* resultam numa imagem binária (apenas preto ou branco) que por norma se assemelha à apresentada na Figura 19.

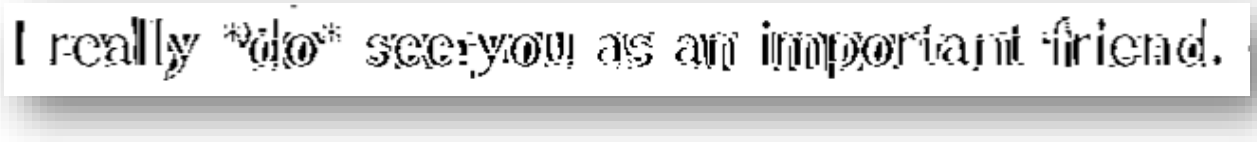


Figura 19 - Filtragem por Contraste DOT

O exemplo apresentado foi realizado de forma automática, no entanto é possível manualmente indicar ao sistema se o texto a procurar é claro num fundo escuro, ou vice-versa, no respetivo menu de calibração. Esta indicação manual facilita a identificação de áreas textuais e tende, em alguns casos, a melhorar significativamente os resultados deste filtro.

5.1.1.3 Remoção de Componentes Lineares Puramente Horizontais ou Verticais

Devido à grande prevalência de regiões de alto contraste em zonas de separação de elementos, tais como extremidades de botões, caixas de texto e áreas de menu, este processo é necessário para eliminar os erros resultantes destes elementos do resultado do filtro de contraste (5.1.1.2). Visto que estes elementos tendem a formar linhas perfeitamente horizontais ou verticais, a sua remoção é realizada muito rapidamente por um scan horizontal e um vertical à respetiva imagem

binária, eliminando qualquer componente que se prolongue para lá do limite máximo de altura e largura estabelecido para um qualquer caráter.

5.1.1.4 Filtragem por Densidade de Contraste

Esta filtragem tem por base a imagem binária resultante de 5.1.1.3 e tem por objetivo realizar uma remoção inicial de qualquer área de alto contraste, encontrada pelo filtro 5.1.1.2, cuja densidade de píxeis de alto contraste seja inferior ao que seria esperado de uma área que contenha elementos textuais. Isto é conseguido através de uma implementação simplificada de um algoritmo de *sliding window* cuja escala da janela se mantém constante (apenas uma passagem) e é definida pelo valor de altura do texto definido em 5.1.1.1. Esta janela vai calculando a densidade de píxeis de alto contraste dentro da área analisada removendo regiões cuja densidade seja superior ou inferior a uma extensão de valores esperados, limpando a imagem binária de muito do barulho criado pelo *background*, mantendo no entanto as linhas de texto intactas.

5.1.1.5 Dilatação Horizontal

É de seguida realizada uma dilatação horizontal cuja distância é determinada pela altura do texto (5.1.1.1). Esta dilatação permite facilitar o trabalho seguinte de determinação e agregação de *bounding boxes* de forma a obter a área que engloba uma linha inteira de texto.

Este processo, tendo em conta que é realizado linha a linha, horizontalmente e numa única iteração é várias vezes mais rápido que o processo comum de dilatação binária, e apresenta melhores resultados para o caso específico deste projeto.

Este é o aspeto (Figura 20) da imagem quando terminado o processo de dilatação horizontal.

Figura 20 - Dilatação Horizontal DOT

5.1.1.6 Determinação de Bounding Boxes

É com a imagem já dilatada que são determinadas as *Bounding Boxes* através de um algoritmo que analisa de forma muito rápida (relativamente ao tempo de processamento do processo equivalente em *OpenCV*) os limites espaciais de cada *blob* (região composta por todos os píxeis diretamente ligados a outro píxel do mesmo tipo) negro. A partir deste momento, todas as restantes operações apenas irão utilizar a área definida por cada *bounding box,* reduzindo muito a área a processar o que permite que este módulo mantenha um bom desempenho mesmo recorrendo a métodos mais pesados computacionalmente nos processos de filtragem seguintes.

5.1.1.7 Determinação da Cor Estimada das Componentes Textuais

A partir dos píxeis encontrados pelo filtro de contraste é possível analisar a cor dos seus equivalentes na imagem original, e desta forma, realizar uma quantização das cores encontradas, mantendo no fim apenas a mais prevalente. A cor selecionada é então associada à *bounding box* a que se refere para que possa ser utilizada no processo de filtragem por cor.

Nos casos em que a cor de texto é sempre constante, tal como em legendas de filmes, é possível ainda indicar a cor do texto a procurar diretamente no respetivo menu de calibração. A indicação manual da cor do texto pode em certos casos melhorar a qualidade final da imagem processada, ao corrigir quaisquer potenciais desvios na previsão anterior da cor do texto.

5.1.1.8 Agregação e Filtragem Heurística das Bounding Boxes Definidas

Tendo uma lista com todas as *Bounding Boxes* encontradas, é então analisada cada uma delas individualmente de forma a tentar agrupar e fundir com uma outra caso esta respeite certas condições, tais como limites de distância, posicionamento relativo e cor semelhante.

Após terminado este processo, todas a *bounding boxes* que não respeitem os critérios de dimensão definidos (área mínima/máxima, altura mínima/máxima, largura mínima/máxima, relação de aspeto mínima/máxima) são ignoradas e não serão mais processadas.

Este processo visa a reduzir o número de *bounding boxes* a processar, formando ao mesmo tempo *bounding boxes* que englobem linhas de texto completas. Caso este processo esteja a

agrupar caixas a mais ou a menos, tanto verticalmente como horizontalmente, é possível ajustar manualmente os valores dos *thresholds* utilizados nos respetivos *sliders* do menu de calibração.

5.1.1.9 Filtragem Binária Baseada na Cor Estimada do Texto

Utilizando por base a imagem a cores original, a área de cada *bounding box* é filtrada de forma a destacar todos os píxeis com uma cor semelhante à associada à sua *bounding box*. Este processo é realizado em duas fases. A primeira destaca todos os píxeis cuja diferença de cor (*Manhattan Distance* das componentes RGB) seja inferior a um *threshold X (Non Connected Component Color Thresh)* e com pelo menos um píxel de alto contraste na sua vizinhança. A segunda fase utiliza estes resultados para destacar todos os píxeis diretamente ligados a um píxel já destacado cuja diferença de cor seja inferior a um *threshold Y (Connected Component Color Thresh)*, tendencialmente de valor mais alto que o *threshold X*. São realizadas uma série de iterações até a imagem se manter inalterada ou até que um valor máximo seja atingido.

O resultado deste processo é visível na Figura 21. A imagem utilizada para esta filtragem foi escolhida propositadamente pelo seu *background* complexo, com zonas que partilham a mesma cor que o próprio texto, para demonstrar que em certos casos uma filtragem mais aprofundada é necessária. É exatamente esta a filtragem que será descrita de seguida em 5.1.1.10.

Figura 21 - Filtragem por Cor DOT

5.1.1.10 Filtragem pelas Características Individuais de Cada Blob

Esta filtragem é apenas necessária em casos onde o *background* da imagem é complexo e confunde o sistema de filtragem definido em 5.1.1.9.

Ao analisar certas características individuais de cada *blob*, tais como o seu rácio de arestas de alto contraste e dimensões, é possível distinguir de forma automática muitos dos *blobs* que

correspondem a barulho na imagem e removê-los completamente da imagem binária resultante. Este filtro foi ainda desenvolvido de forma a que o utilizador seja capaz de ajustar a intensidade do filtro automático, assim como definir os valores máximos e mínimos, de dimensão, área e perímetro, permitidos a cada *blob*.

A Figura 22 permite visualizar o resultado de uma filtragem automática realizada por este sistema. A cores é possível ver todos os *blobs* que foram aprovados pelo filtro, e a cinzento-escuro todos os que falharam o teste. O resultado binário final (Figura 23) corresponde à imagem que será mais tarde processada pelo sistema de *OCR*, que com o *background* removido o é agora capaz de a interpretar.

Figura 22 - Filtragem por Blobs DOT (Cores)

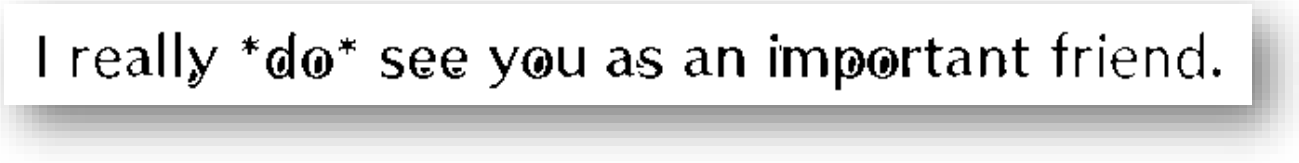


Figura 23 - Filtragem por Blob (Binária)

5.1.1.11 Comparação com as Bounding Boxes da frame anterior

Caso já tenha sido processada uma imagem antes da presente, as suas *bounding boxes* serão comparadas às presentes em termos de posicionamento, cor e grau de aproximação ao novo resultado processado. Caso sejam encontrados casos positivos, as novas *bounding boxes* irão partilhar o mesmo número de identificação que as suas antecessoras, de forma a informar o resto do sistema que estas linhas de texto se tratam apenas de uma atualização da anterior.

Quando terminado este processo, as *bounding boxes* mais recentes substituem as antigas para que o processo se possa repetir na *frame* seguinte.

5.1.1.12 Identificação de Blocos de Texto

De seguida, são analisadas todas as novas bounding boxes, as suas dimensões e posicionamentos relativos umas às outras, para que possam ser identificados blocos de texto.

Blocos de texto consistem simplesmente em grupos de linhas de texto cujo seu posicionamento relativo indica pertencerem ao mesmo parágrafo ou fazerem parte do mesmo contexto.

Visto que este sistema deteta apenas linhas individuais, esta comparação é necessária para que blocos de texto possam ser futuramente agrupados e traduzidos por inteiro, o que resulta em traduções de melhor qualidade pois possibilita que todo o contexto do bloco seja tido em conta durante o processo de tradução.

5.1.1.13 Construção das DOT_PlugInContracts::TextLine

Para terminar este processo falta apenas transformar as *bounding boxes* processadas, e toda a informação que lhes está associada, num objeto C# (*DOT_PlugInContracts::TextLine*) que possa ser processado pelo resto do sistema.

Este processo envolve apenas passar toda a informação relevante ao resto do sistema para uma nova *TextLine*. Desta informação destaca-se um *Bitmap* com a imagem processada, espessura média dos carateres, dimensões e posicionamento da *bounding box,* cor de *foreground* e *background*, ID e ID da linha seguinte (caso faça parte de um bloco), entre outras.

5.2 Optical Character Recognition

Para o módulo de *OCR* foi utilizada a versão mais recente (*4.0.0-alpha* 26/03/2017) da biblioteca *open source Tesseract* (Weld, 2017).

Tendo em conta que o módulo de deteção e segmentação de texto filtra a imagem original, de forma a obter uma imagem binária (fundo branco e texto preto) e recorta e associa a cada caixa

de texto apenas a área que lhe é devida, o módulo de OCR apenas envia a imagem processada de cada caixa de texto para o motor do *Tesseract*, que responde com a *string* de texto que pensa ter identificado, e respetivo valor de confiança que lhe atribui. De forma a obter os melhores resultados possíveis, foi definido um *threshold* de confiança que apenas permite que resultados com níveis de confiança acima de um determinado valor sejam processados. Valores abaixo deste *threshold* deverão voltar a ser processados pelo OCR assim que uma nova imagem processada entre no sistema, até que o valor de confiança encontrado seja alto o suficiente para ser considerado. Desta forma, certas características da imagem processada que possam causar problemas durante o processamento pelo OCR podem, em alguns casos, desaparecer por completo (em situações onde o background é complexo e se move relativamente ao texto) resultando numa interpretação mais correta da imagem. Este valor de *threshold* está originalmente definido como 0.7 (onde o mínimo é 0 e o máximo 1), visto que tende a apresentar um bom rácio entre quantidade de texto interpretado e qualidade de resultados, no entanto pode ser alterado manualmente pelo utilizador durante a utilização do *DOT*.

De forma a eliminar erros de ocorrência constante pelo *Tesseract* foi também definida uma lista de *strings* de substituição que permite ao utilizador substituir qualquer resultado que considere errado pela sua correção manual.

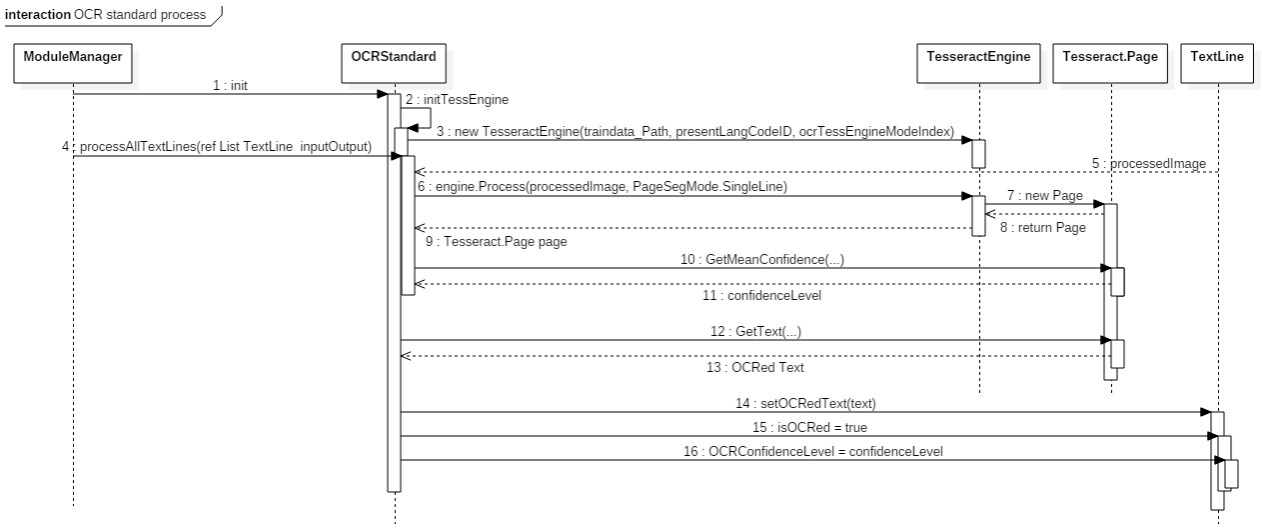


Figura 24 - Diagrama de sequência do processo de OCR

O diagrama de sequência apresentado na Figura 24 apresenta de forma simplificada o processo básico realizado pelo módulo de *OCR.*

5.3 Machine Translation

Para traduzir todo o texto encontrado pelo módulo de OCR, foi decidido tirar partido de duas soluções de tradução, o *Google Translator*, e o *Yandex Translate*. A decisão de não usar a solução da *Microsoft* deveu-se exclusivamente ao grau de dificuldade associado à criação (obrigatória) de contas *Microsoft Azure* para cada utilizador do *DOT*. A solução da *Google* tornou-se viável (apesar de não comercializável) com a descoberta do pedido REST associado a uma aplicação de tradução já existente, o que permitiu que este serviço pudesse ser utilizado.

O módulo de tradução processa todas as caixas de texto encontrando todas as que contêm texto já interpretado pelo OCR e ainda por traduzir. As caixas que correspondem a estes critérios são tratadas de forma diferente caso pertençam a um bloco de texto ou não (definido pelo módulo de deteção e segmentação). O texto associado a caixas pertencentes a um bloco é agrupado de forma a que a tradução seja feita por inteiro (por oposição a linha a linha), potenciando assim uma tradução de melhor qualidade, ao apresentar o contexto completo em que a frase se insere ao respetivo tradutor.

As linhas não pertencentes a um bloco de texto são tratadas e traduzidas individualmente.

A tradução é feita por *REST* aos respetivos serviços de tradução e assim que obtida a resposta, é associada à respetiva caixa de texto, pronta para ser apresentada pelo módulo de *overlay*.

Por fim, foram dadas ao utilizador as opções de manter este módulo desligado (não realizar tradução), apenas realizar tradução quando indicado pelo utilizador (modo manual) ou tradução automática assim que uma nova *string* de texto for encontrada (modo *ASAP*).

É também possível forçar a aplicação a traduzir cada linha individualmente (caso esta pertença a um bloco de texto), para casos em que as linhas sejam agrupadas mas não seja relevante traduzir em bloco, tal como acontece por vezes em menus de aplicações.

Todo este processo está apresentado sob a forma de um diagrama de sequência na Figura 25.

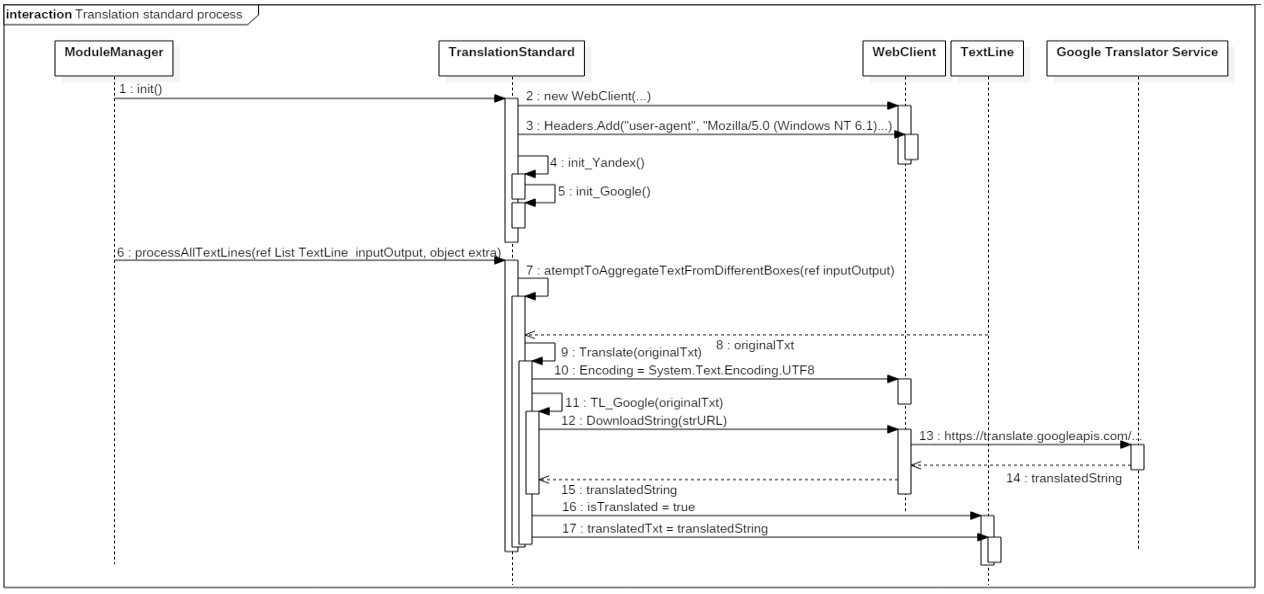


Figura 25 - Diagrama de sequência do processo de tradução

5.4 Overlay sobre Aplicações Gráficas

Durante a fase de desenvolvimento deste módulo foi decidido que só seria desenvolvido um *overlay* para aplicações não *exclusive fullscreen*. Esta decisão deveu-se aos fatores de segurança referidos em 3.2.4, a constrangimentos de tempo inesperados durante o desenvolvimento e ao facto da vasta maioria das aplicações capazes de correr em *exclusive fullscreen* serem também capazes de correr em *standard* ou *borderless window*, o que torna esta característica num requisito com um rácio muito baixo de valor/custo de desenvolvimento.

Desta forma, a solução desenvolvida utiliza uma janela em *WPF*, completamente transparente e "*click-throught*" que se pode ou não ajustar à janela da aplicação alvo. Esta janela é referida no contexto da aplicação como *TLLensWindow* ou apenas *Lens Window*, visto o seu comportamento ser semelhante ao de uma lente colocada sobre um qualquer objeto.

É nesta janela transparente onde as caixas de texto *DOT* são desenhadas. Estas caixas, assim como a barra superior e inferior da *Lens Window*, são customizáveis em termos de cor de letra e *background* e mesmo da família da fonte a ser utilizada. Estas opções, quando em modo automático, utilizam a informação de cor obtida pelo módulo de deteção de texto para manter a cor da letra e de *background* o mais próxima possível do original.

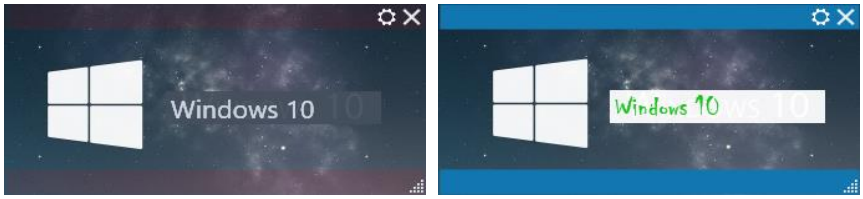


Figura 26 - À esquerda, LensWindow em modo Automático; À direita LensWindow customizada manualmente

A segunda função do módulo de *overlay* é a recolha sistemática de *screenshots* da área onde a *Lens Window* se sobrepõe à janela da aplicação alvo. Esta funcionalidade foi desenvolvida tirando partido de funções da Windows *API* tais como *GetForegroundWindow* (*user32.dll*) para obter o *handle* da aplicação alvo, *GetWindowRect* (*user32.dll*) para obter o tamanho e posição no ecrã da aplicação alvo comparativamente à *Lens Window*, e *CreateCompatibleDC*, *createCompatibleBitmap*, *BitBlt* (*gdi32.dll*) e *FromHbitmap* (*System.Drawing.Bitmap*) para obter um *bitmap* da imagem completa da janela da aplicação alvo.

Este método revelou-se eficaz para a grande maioria das aplicações, no entanto foram detetadas algumas exceções durante a fase de testes, pelo que deverá ser atualizado no futuro.

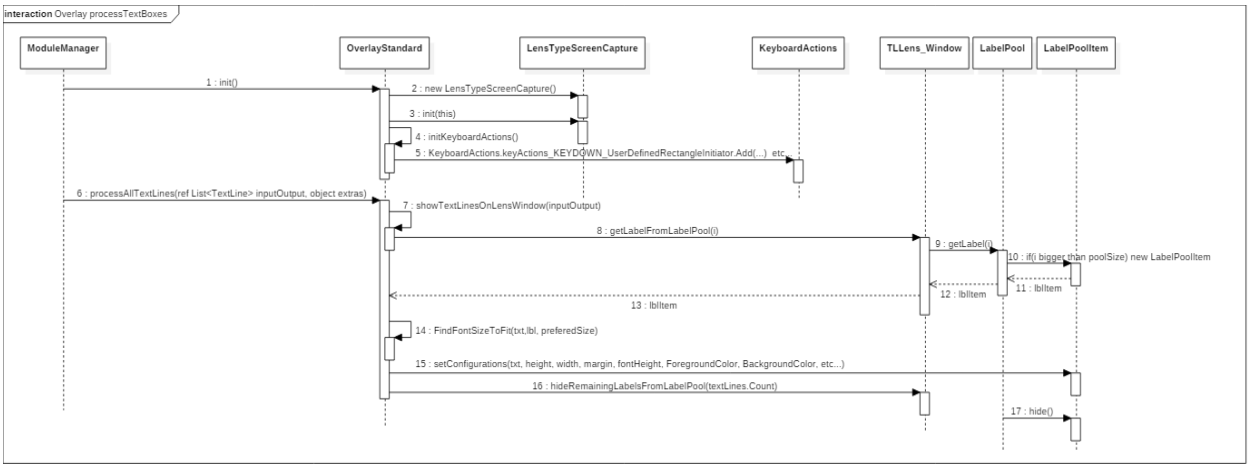


Figura 27 - Diagrama de sequência do processo de display de linhas de texto

O diagrama da Figura 27 descreve em maior detalhe o processo de apresentação das linhas de texto sobre a *LensWindow*.

5.5 PlugIn API

Para que seja possível a substituição rápida de qualquer módulo *DOT*, foi desenvolvida uma *API* especifica para este projeto (*DOT_PlugInContracts*) que permite a qualquer utilizador com conhecimentos de C# criar um ou mais *DLLs* que respondam às suas necessidades específicas. Esta API é constituída pela implementação das estruturas básicas, nomeadamente as *bounding boxes* e as *Text Lines*, assim como cinco interfaces, uma para cada módulo (I_*TextDetection; I_OCR; I_Translation; I_Overlay*) e a restante para a configuração da calibração de módulos (*I_ModuleSettings*). Estas interfaces obrigam à criação de todos os métodos obrigatórios para o funcionamento correto de cada módulo. O seguinte excerto de código faz parte da interface associada ao módulo de deteção e segmentação de texto, e serve de exemplo para os restantes três.

```
namespace DOT_PlugInContracts
{
    public interface I_TextDetection
    {
        void init();
        void loadSingleLineRectangleHint(int x, int y, int width, int height);
        object processFrame(uint dtime, ref Bitmap image, ref List<TextLine> output, object extraInputIfNeeded = null);
        void forceRefresh();
        I_ModuleSettings getSettings();
        void runSettingsWindow();
    }
}
```

Note-se também que no método *processFrame/processAllTextLines* (existente em todos os módulos) é sempre possível retornar um *object* e enviar um outro no parâmetro "*extraInputIfNeeded*". Todos os módulos *DOT standard* (não *plugIn*) retornam neste método o parâmetro *extraInputIfNeeded* sem qualquer alteração, reencaminhando-o de seguida para o módulo seguinte. Isto permite que uma série de dados extra possam ser passados entre os diferentes módulos, o que permite que as novas implementações possam comunicar entre si com facilidade sem recorrer a métodos de comunicação externos à aplicação.

Por fim, de forma a permitir total liberdade para a criação da *UI* de calibração do novo módulo, este deverá ser desenvolvido pelo criador do *plugIn* e integrado no mesmo de forma a que a sua janela seja aberta sempre que chamado o método *runSettingsWindow*, definido na interface de todos os módulos.

Para acrescentar qualquer *plugIn* à aplicação principal basta apenas colocar o respetivo *DLL* dentro da pasta associada ao módulo a ser alterado, dentro da pasta *PlugIns*, que se encontra no diretório do executável (Exemplo: "*…/DirectOverlayTranslator/PlugIns/1_OCR/ITHSupportPlugin_DOT.dll*"). Assim que a aplicação seja iniciada é possível selecionar o novo *plugIn* no menu *Settings* da janela principal.

5.6 Configuração de Registos

De forma a manter a aplicação completamente portátil, foi desenvolvido um pequeno executável externo cuja única função é realizar as associações necessárias nos registos da máquina a ser utilizada, para que os *save files* do *DOT* possam estar associados ao respetivo ícone da aplicação.

Visto que este tipo de operações nos registos obriga a obtenção de direitos de administração, esta aplicação foi desenvolvida separadamente da aplicação principal *DOT*, o que permite que esta seja a única que necessite de solicitar permissões administrativas ao sistema. Desta forma a aplicação principal apenas precisa de correr o programa de registos uma vez por máquina, sem necessitar dela própria ganhar permissões administrativas, o que ajuda a prevenir potenciais problemas de segurança derivados da utilização de *plugIns* externos (dado que os seus conteúdos podem ser desconhecidos).

5.7 Interface Gráfica

Toda a *UI* desta aplicação foi desenvolvida em *Windows Presentation Foundation* (*WPF*) visto que permite criar uma interface muito flexível e customizável, com aspeto moderno e com um bom desempenho para as necessidades deste projeto.

A interface do *DOT* foi toda ela customizada visualmente de forma a manter um aspeto simples e limpo, e foi dividida num total de quatro janelas:

- A Janela de *Loading* Inicial (Figura 28);
- A Janela Lente, ou LensWindow(Figura 28);
- A janela Principal (Figura 31);
- A janela de Calibração (Figura 29).

A janela de *loading* (Figura 28) tem como única utilidade dar algum *feedback* ao utilizador após este iniciar a aplicação, visto que certos módulos, principalmente o de *OCR,* tendem a demorar alguns segundos a carregar inicialmente, o que poderia levar o utilizador a pensar incorretamente que a aplicação não foi iniciada ou se encontra congelada.

A Janela Lente, ou *LensWindow* tem como funcionalidade definir a área a ser analisada pela aplicação assim como apresentar todas a caixas de texto (com o respetivo texto) diretamente sobre as linhas originais. Tem também como funcionalidade secundária apresentar ao utilizador a imagem processada pelo módulo de deteção de texto, de forma a apresentar o *feedback* direto,

Figura 28 - À esquerda: Janela de Loading; À direita LensWindow

em tempo real, das alterações feitas ao calibrar este módulo. Isto simplifica o processo de calibração visto que permite que o utilizador veja de imediato o que o módulo de *OCR* irá "*ver*".

Esta lente foi desenhada de forma a minimizar o impacto visual que tem sobre a aplicação alvo, permitindo que o utilizador escolha qualquer possibilidade de cor para as *dragbars* incluindo até que estas sejam completamente transparentes. A customização das caixas de texto apresentadas pela *LensWindow* foram já anteriormente descritas no subcapítulo 5.4 (*Overlay* sobre Aplicações Gráficas).

Todas estas alterações podem ser realizadas com ajuda da janela de calibração (Figura 29). Esta janela permite alterar todas as configurações necessárias de qualquer um dos quatro módulos *DOT*.

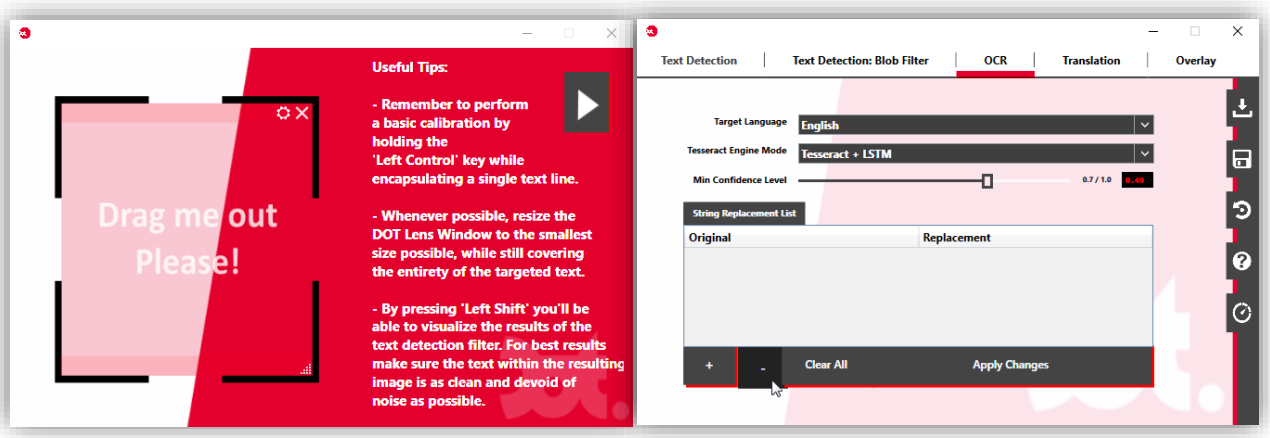


Figura 29 - Menu de Calibração

Este menu permite ainda que, a qualquer momento, as configurações de qualquer módulo, individual ou em conjunto, sejam gravadas para um *save file*, carregadas ou restabelecidas aos valores originais (*reset*). Por fim, é possível analisar o desempenho de cada módulo individual ao observar a média móvel dos tempos de processamento por ciclo de cada um (Figura 30).

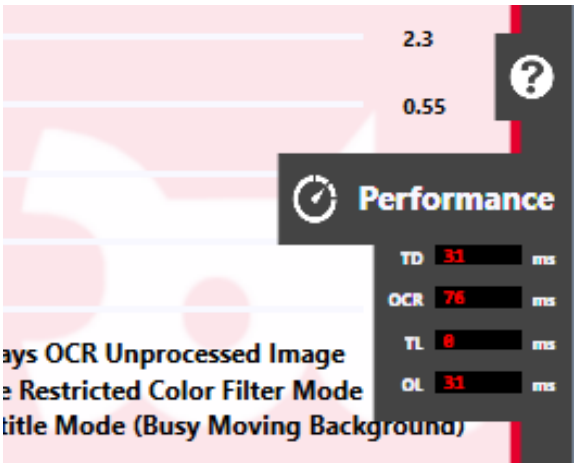


Figura 30 - Menu de Calibração (performance meter)

A janela principal da aplicação (Figura 31) é constituída por quatro menus:

- Menu principal;
- Configurações Gerais;
- Ajuda;
- Extras.

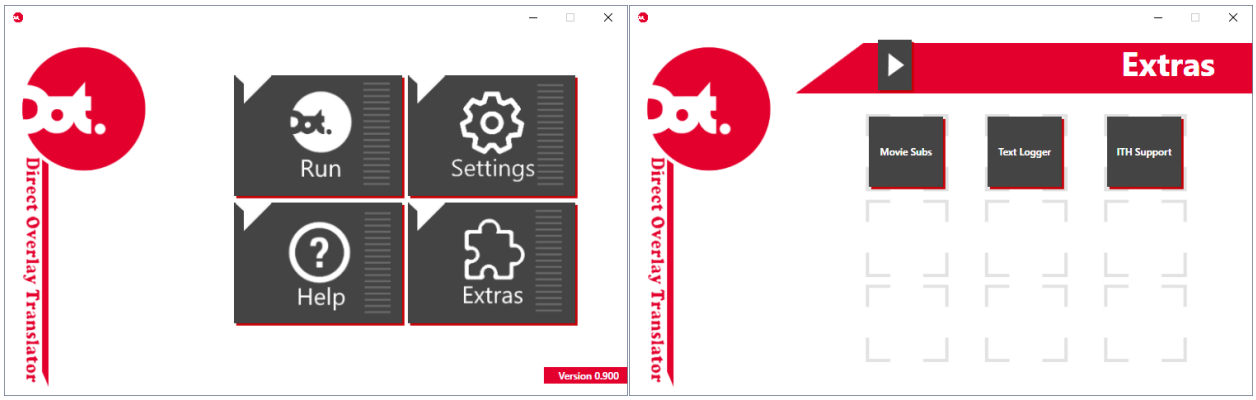


Figura 31 - Menu Principal

O menu principal, é a capa da aplicação e permite inicializar a *LensWindow* e aceder a qualquer um dos restantes menus.

O menu de configurações gerais (*Settings*) permite selecionar os módulos a utilizar (*DOT standard* ou *plugIn*) assim como alterar as teclas de controlo definidas de origem.

O menu de ajuda está de momento sem conteúdo devido a constrangimentos de tempo, mas a sua estrutura principal está completa.

Por fim, o menu de extras contém os controlos para um *text logger*, capaz de gravar para um ficheiro todo o texto processado, e o menu de inicialização do *Interactive Text Hooker*, útil para aplicações alvo que suportem a sua utilização. Existe ainda o menu de criação automática de legendas para vídeos, mas encontra-se desativado visto que esta funcionalidade ainda precisa de ser corretamente testada e incorporada no sistema principal.

5.8 Integração de Suporte a *Text Hooking*

De forma a permitir que a aplicação seja capaz de utilizar uma solução baseada em *text hooking* sempre que conveniente, foi desenvolvido um *plugIn* para este propósito que substitui o módulo de *OCR* predefinido. Este *plugIn* (*ITHSupportPlugin_DOT.dll*) limita-se a colocar nas *TextLines* encontradas, de forma controlada, os resultados encontrados a cada momento pela aplicação dedicada de *text hooking ITH* (estudada no subcapítulo 2.3.4). Isto permite que, após calibrada uma instância do *ITH* (inicializada através do menu de extras na aplicação *DOT*), seja possível obter um grau de precisão constante de 100% de carateres corretamente detetados. As *strings* de texto detetadas continuam o seu percurso pela aplicação de forma normal, sendo traduzidas e apresentadas sobre as suas respetivas linhas de texto da aplicação alvo.

Esta interação entre o *DOT* e o *ITH* é possível visto que o *ITH* coloca todos os resultados que encontra no *clipboard* da máquina em questão, o que permite que estes sejam facilmente obtidos e processados adequadamente.

5.9 Conclusão

Em conclusão, a grande maioria das funcionalidades previstas para a aplicação foram desenvolvidas com sucesso, salvo a funcionalidade de *overlay* sobre aplicações *exclusive fullscreen*; e a secção de ajuda e tutoriais de interação com a interface do *DOT*. Ambas estas funcionalidades foram postas em segundo plano devido a constrangimentos de tempo durante o período reservado para o desenvolvimento da aplicação. Esta é no entanto uma situação aceitável visto que aplicações *exclusive fullscreen* por norma permitem também correr em janela (*borderless* ou normal), e está previsto o desenvolvimento da secção de tutoriais num futuro próximo.

As capacidades de tudo o que foi aqui descrito serão postas à prova no capítulo 6.

6 Avaliação da solução

Neste capítulo serão abordados os planeamentos, realização e resultados da fase de testes da solução final, o que permitirá tirar conclusões sobre a qualidade final da aplicação e os pontos onde a realização de trabalho futuro beneficiará o desempenho e qualidade futura da aplicação.

6.1 Experiências e Avaliação da Solução

Este projeto foi testado de modo a verificar se todos os objetivos foram atingidos com sucesso; e avaliar a qualidade da solução final. Para isto, o processo de testes foi dividido em duas fases. A primeira engloba todos os testes funcionais e de desempenho, necessários para avaliar o funcionamento da solução contra os objetivos definidos inicialmente. Na segunda fase, os testes não funcionais, compreendem todos os testes relativos à facilidade de utilização e compreensão da aplicação.

6.1.1 Grandezas a Utilizar

As grandezas a utilizar durante a avaliação de cada módulo dentro da solução principal serão as seguintes:

- **Segmentação de Texto**

  O módulo de segmentação de texto será avaliado tendo em conta o número de segmentações realizadas corretamente, número de frases não encontradas e número de falso-positivos apresentados pelo algoritmo.
- **OCR**

  A qualidade do módulo de *OCR* será determinada pelo rácio de sucesso, obtido ao dividir o número de carateres interpretados corretamente pelo número total de carateres analisados.

- ***Machine Translation***

  Tendo em conta que a avaliação de uma tradução é dependente do contexto em que a frase é utilizada, esta será uma avaliação relativa que deverá avaliar dois fatores: a Inteligibilidade e a fidelidade da tradução. A Inteligibilidade é a medida que avalia o quão compreensível a frase traduzida é. A fidelidade, por outro lado, mede a quantidade de informação original que a tradução foi capaz de reter. Estes dois fatores serão avaliados numa escala de 0 a 10 quando comparados com uma tradução manual.

- ***Overlay***

  Os testes à componente de *overlay* devem garantir que a aplicação é capaz de funcionar com uma grande variedade de aplicações. Estes englobam também testes de tempo de resposta, apresentados em milissegundos, e testes de impacto sobre a aplicação alvo, medidos pela diferença em milissegundos, caso esta exista, entre a aplicação quando afetada ou não pelo *overlay*.

6.1.2 Hipóteses a Testar

As hipóteses a testar durante a avaliação de cada módulo dentro da solução principal serão as seguintes:

- **Segmentação de Texto**
  - A aplicação é capaz de consistentemente encontrar todo o texto apresentado num grupo predefinido de aplicações*?*
  - Caso não seja capaz de consistentemente encontrar todos os blocos de texto, a aplicação é capaz de manter uma média de identificações corretas superior a 80%?
  - A aplicação é capaz de consistentemente separar devidamente o texto em caixas (*bounding boxes*) de uma só linha?
  - A aplicação é capaz de consistentemente apresentar uma média de falso-positivos inferior a 20% num grupo predefinido de aplicações*?*

- **OCR**
    - A aplicação é capaz manter uma média superior a 80% de identificações corretas de carateres num grupo predefinido de aplicações*?*
- **Machine Translation**
    - Tendo em conta a escala de 0 a 10 definida anteriormente, a aplicação é capaz de manter uma média de inteligibilidade superior a 5 num grupo predefinido de aplicações*?*
    - Tendo em conta a escala de 0 a 10 definida anteriormente, a aplicação é capaz de manter uma média de fidelidade superior a 5 num grupo predefinido de aplicações*?*
- **Overlay**
    - A aplicação é capaz de realizar *overlays* sobre aplicações em *windowed mode,* num grupo predefinido de aplicações?
    - A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *DirectX9* em *Fullscreen mode,* num grupo predefinido de aplicações?
    - A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *DirectX11* em *Fullscreen mode,* num grupo predefinido de aplicações?
    - A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *OpenGL (4.0)* em *Fullscreen mode,* num grupo predefinido de aplicações?
    - O tempo médio de resposta do sistema de overlay mantem-se inferior a 200 milissegundos, num grupo predefinido de aplicações?
    - A diferença média de tempo por ciclo de *rendering* medida entre a aplicação quando afetada ou não pelo *overlay* mantém-se inferior a 30 milissegundos, num grupo predefinido de aplicações?
- **Testes não funcionais**
    - A apresentação da aplicação é atrativa para qualquer tipo de utilizador?
    - A aplicação foi capaz de apresentar uma explicação coerente e simples de todos os conceitos necessários para a sua operação?
    - O modo de utilização da aplicação é simples de aprender?

- A aplicação é simples de utilizar de forma a apresentar resultados favoráveis?
- Os resultados são apresentados de forma compreensível sem prejudicar a aplicação alvo de forma percetível?
- Esta solução apresenta resultados de forma mais rápida e mais prática comparativamente com a sua versão manual?

6.1.3 Metodologia de Avaliação

As hipóteses relativas à interface da aplicação foram postas à prova por um grupo de teste constituído por indivíduos com diferentes níveis de facilidade de interação e conhecimento tecnológico. Este grupo respondeu a um questionário simples que permitiu uma fácil obtenção de dados estatísticos sobre as características foco dos testes realizados.

Os resultados dos questionários associados às hipóteses relativas à interface da aplicação serão apresentados sob a forma de gráficos, que permitirão tirar conclusões relativas às hipóteses a que se referem.

As restantes hipóteses foram testadas pelo próprio aluno dentro de uma amostra de aplicações alvo, focando-se nos aspetos relacionados com a tradução das suas interfaces. Estas aplicações foram escolhidas especificamente por incluírem texto nas seguintes línguas: Inglês; Português; Espanhol; Francês; Alemão; Japonês; Chinês Simplificado; Coreano e Russo. Desta forma é possível rapidamente alternar entre as diferentes línguas disponíveis, o que permite isolar a componente de tradução durante a realização dos respetivos testes.

Relativamente às hipóteses que dizem respeito ao sistema de tradução, após a realização da avaliação nas amostras selecionadas, foram obtidos os valores mínimos, máximos e médios das avaliações realizadas de forma a que seja possível avaliar a competência da solução de tradução. O procedimento seguido para a realização destes testes requereu que as aplicações alvo que fizeram parte do teste tenham a capacidade de alterar a língua da sua *UI*. Assim, torna-se possível,

mesmo não tendo um bom conhecimento da língua a ser testada, comparar os resultados traduzidos pela solução contra a sua tradução oficial.

As restantes, após testadas pelo aluno (numa máquina que não a de desenvolvimento) serão apresentadas de forma binária, ou seja se confirmam ou não a hipótese, juntamente com as médias obtidas da avaliação da amostra de aplicações alvo, sempre que relevante.

6.1.4 Primeira Fase – Testes Funcionais e de Desempenho

Esta fase foi dedicada aos testes funcionais e ao desempenho da aplicação, onde foi avaliada a competência dos quatro principais módulos desta solução (segmentação de texto, *OCR*, tradução e *overlay*) assim como o funcionamento da aplicação como um todo.

Todos os seguintes módulos foram avaliados de acordo com o seu desempenho quando aplicados a uma amostra de aplicações, focando-se nos aspetos relacionados com a tradução das suas interfaces com o utilizador. Estas aplicações deverão incluir texto nas seguintes línguas: Inglês, Português, Espanhol, Francês, Alemão, Japonês, Chinês Simplificado, Coreano e Russo, e por esta razão foram selecionadas para teste as seguintes:

- *Malwarebytes Anti-Malware;*
- *Panda Protection Antivirus;*
- *Rise of The Tomb Raider* (Menus *e Settings*);
- *Skype;*
- *The Hobbit The Desolation of Smaug* (apresentado no *VLC video player*).

As tabelas Tabela 7, Tabela 8, Tabela 9, Tabela 10 e Tabela 11 mostram o resultado dos testes realizados a cada uma das aplicações. É ainda possível observar o processo de obtenção dos resultados em maior detalhe na Tabela 14 e Tabela 15 presente na secção de anexos.

Tabela 7 - Testes: Filme 'The Hobbit: The Desolation of Smaug'

Aplicação Alvo	**Máquina de Testes**					
The Hobbit: The Desolation of Smaug (00:09:12 - 00:15:40)	i5 4200U 1.6Ghz; GTX740M; 4Gb DDR3					
Língua						
De	**Russo**	**Português**	**Alemão**	**Japonês**	**Coreano**	**Chinês**
Para	**Português**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**
Overlay						
É capaz de capturar *screenshots*	**Sim**					
Afetou a aplicação alvo	**Não**					
Tempo médio de resposta	**31 ms**					
Deteção e Segmentação de Texto						
Linhas corretamente encontradas	**45**	**61**	**83**	**58**	**39**	**54**
Linhas incorretamente ignoradas	**0**	**0**	**0**	**1**	**0**	**0**
Falso-positivos	**0**	**0**	**0**	**0**	**0**	**0**
Tempo médio de resposta	**61 ms**	**60 ms**	**62 ms**	**65 ms**	**63 ms**	**66 ms**
OCR						
Carateres corretamente identificados	**2281**	**1435**	**1859**	**1833**	**1770**	**1379**
Carateres incorretamente identificados	**2**	**4**	**7**	**9**	**3**	**11**
Percentagem de precisão	**99.91%**	**99.72%**	**99.62%**	**99.51%**	**99.83%**	**99.21%**
Tradução						
Nível Médio de Inteligibilidade (0-10)	**7.78**	**8.23**	**7.96**	**6.67**	**5.97**	**5.83**
Nível Médio Fidelidade (0-10)	**8.76**	**8.83**	**8.07**	**7.21**	**6.13**	**6.83**
Mínimo de Inteligibilidade (0-10)	**2**	**4**	**2**	**1**	**1**	**1**
Mínimo Fidelidade (0-10)	**4**	**4**	**5**	**1**	**1**	**1**
Máximo de Inteligibilidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**
Máximo Fidelidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**

Tabela 8 - Testes: Panda AV

Aplicação Alvo	**Máquina de Testes**				
Panda AV	i5 4200U 1.6Ghz; GTX740M; 4Gb DDR3				
Língua					
De	**Inglês**	**Francês**	**Alemão**	**Russo**	**Chinês**
Para	**Português**	**Inglês**	**Inglês**	**Inglês**	**Inglês**
Overlay					
É capaz de capturar *screenshots*	**Sim**				
Afetou a aplicação alvo	**Não**				
Tempo médio de resposta	**23 ms**				
Deteção e Segmentação de Texto					
Linhas corretamente encontradas	**43**	**43**	**43**	**43**	**43**
Linhas incorretamente ignoradas	**0**	**0**	**0**	**0**	**0**
Falso-positivos	**7**	**4**	**3**	**3**	**0**
Tempo médio de resposta	**118**	**124**	**117**	**122**	**128**
OCR					
Carateres corretamente identificados	**784**	**744**	**922**	**816**	**518**
Carateres incorretamente identificados	**1**	**2**	**1**	**13**	**19**
Percentagem de precisão	**99.87%**	**99.73%**	**99.89%**	**98.43%**	**96.46%**
Tradução					
Nível Médio de Inteligibilidade (0-10)	**9.05**	**9.45**	**9.42**	**8.43**	**7.42**
Nível Médio Fidelidade (0-10)	**8.89**	**9.71**	**9.42**	**8.88**	**7.91**
Mínimo de Inteligibilidade (0-10)	**1**	**1**	**1**	**1**	**1**
Mínimo Fidelidade (0-10)	**1**	**1**	**1**	**1**	**1**
Máximo de Inteligibilidade (0-10)	**10**	**10**	**10**	**10**	**10**
Máximo Fidelidade (0-10)	**10**	**10**	**10**	**10**	**10**

Tabela 9 - Testes: Rise of the Tomb Raider

Aplicação Alvo	**Máquina de Testes**						
Rise of The Tomb Raider (Menus and Settings)	i7 2600 3.5Ghz; GTX980TI; 8Gb DDR3						
Língua							
De	**Inglês**	**Português**	**Alemão**	**Japonês**	**Coreano**	**Chinês**	**Russo**
Para	**Português**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**
Overlay							
É capaz de capturar *screenshots*	**Sim**						
Afetou a aplicação alvo	**Não**						
Tempo médio de resposta	**27 ms**						
Deteção e Segmentação de Texto							
Linhas corretamente encontradas	**42**	**40**	**42**	**41**	**39**	**40**	**42**
Linhas incorretamente ignoradas	**0**	**2**	**0**	**1**	**3**	**2**	**0**
Falso-positivos	**8**	**0**	**6**	**0**	**0**	**2**	**3**
Tempo médio de resposta	**330 ms**	**330 ms**	**330 ms**	**330 ms**	**330 ms**	**330 ms**	**330 ms**
OCR							
Carateres corretamente identificados	**2391**	**2579**	**2744**	**2293**	**1316**	**1544**	**2963**
Carateres incorretamente identificados	**4**	**8**	**13**	**5**	**21**	**102**	**9**
Percentagem de precisão	**99.83%**	**99.69%**	**99.53%**	**99.78%**	**98.43%**	**93.80%**	**99.70%**
Tradução							
Nível Médio de Inteligibilidade (0-10)	**7.467**	**8.543**	**9.301**	**8.769**	**8.913**	**5.175**	**7.667**
Nível Médio Fidelidade (0-10)	**7.012**	**8.496**	**9.34**	**8.769**	**8.957**	**5.273**	**7.675**
Mínimo de Inteligibilidade (0-10)	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Mínimo Fidelidade (0-10)	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Máximo de Inteligibilidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Máximo Fidelidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**	**10**

Tabela 10 - Testes Malwarebytes

Aplicação Alvo	**Máquina de Testes**					
Malwarebytes Anti-Malware	i5 4200U 1.6Ghz; GTX740M; 4Gb DDR3					
Língua						
De	**Inglês**	**Português**	**Alemão**	**Japonês**	**Coreano**	**Chinês**
Para	**Português**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**
Overlay						
É capaz de capturar *screenshots*	**Sim**					
Afetou a aplicação alvo	**Não**					
Tempo médio de resposta	**44 ms**					
Deteção e Segmentação de Texto						
Linhas corretamente encontradas	**85**	**88**	**85**	**84**	**83**	**75**
Linhas incorretamente ignoradas	**2**	**0**	**0**	**1**	**0**	**0**
Falso-positivos	**5**	**3**	**4**	**3**	**3**	**3**
Tempo médio de resposta	**183 ms**	**191 ms**	**186 ms**	**188 ms**	**191 ms**	**178 ms**
OCR						
Carateres corretamente identificados	**1323**	**2013**	**2166**	**1267**	**1148**	**1012**
Carateres incorretamente identificados	**2**	**2**	**6**	**7**	**57**	**13**
Percentagem de precisão	**99.85%**	**99.90%**	**99.72%**	**99.45%**	**95.27%**	**98.73%**
Tradução						
Nível Médio de Inteligibilidade (0-10)	**9.65**	**9.83**	**9.201**	**8.74**	**6.849**	**7.01**
Nível Médio Fidelidade (0-10)	**9.93**	**9.83**	**9.902**	**8.9**	**5.701**	**7.63**
Mínimo de Inteligibilidade (0-10)	**8**	**2**	**4**	**1**	**1**	**1**
Mínimo Fidelidade (0-10)	**8**	**2**	**5**	**1**	**1**	**1**
Máximo de Inteligibilidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**
Máximo Fidelidade (0-10)	**10**	**10**	**10**	**10**	**10**	**10**

Tabela 11 - Testes: Skype

Aplicação Alvo	**Máquina de Testes**							
Skype	i5 4200U 1.6Ghz; GTX740M; 4Gb DDR3							
Língua								
De	**Russo**	**Português**	**Alemão**	**Francês**	**Espanhol**	**Japonês**	**Coreano**	**Chinês**
Para	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**	**Inglês**
Overlay								
É capaz de capturar *screenshots*	**Sim**							
Afetou a aplicação alvo	**Não**							
Tempo médio de resposta	**15 ms**							
Deteção e Segmentação de Texto								
Linhas corretamente encontradas	**17**	**12**	**13**	**15**	**12**	**12**	**11**	**9**
Linhas incorretamente ignoradas	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
Falso-positivos	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Tempo médio de resposta	**40 ms**	**33 ms**	**35 ms**	**38 ms**	**31 ms**	**33 ms**	**31 ms**	**29 ms**
OCR								
Carateres corretamente identificados	**418**	**367**	**384**	**469**	**350**	**341**	**207**	**179**
Carateres incorretamente identificados	**0**	**0**	**0**	**0**	**0**	**5**	**30**	**19**
Percentagem de precisão	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**98.56%**	**87.34%**	**90.40%**
Tradução								
Nível Médio de Inteligibilidade (0-10)	**10**	**10**	**8**	**10**	**10**	**5**	**8**	**2**
Nível Médio Fidelidade (0-10)	**10**	**10**	**9**	**10**	**10**	**6**	**8**	**2**
Mínimo de Inteligibilidade (0-10)	**10**	**10**	**8**	**10**	**10**	**5**	**8**	**2**
Mínimo Fidelidade (0-10)	**10**	**10**	**9**	**10**	**10**	**6**	**8**	**2**
Máximo de Inteligibilidade (0-10)	**10**	**10**	**8**	**10**	**10**	**5**	**8**	**2**
Máximo Fidelidade (0-10)	**10**	**10**	**9**	**10**	**10**	**6**	**8**	**2**

Foi ainda testada a capacidade de captura de imagem (Tabela 12) pelo módulo de *overlay* nas seguintes aplicações:

- *Discord;*
- *Door Kickers;*
- *GoG Galaxy;*
- *Google Chrome;*
- *I am Setsuna;*
- *Kerbal Space Program;*
- *Little Nightmares;*
- *Mass Effect Andromeda;*
- *Mozilla Firefox;*
- *Nine Hours, Nine Persons , Nine Doors;*
- *Overwatch;*
- *Resident Evil: Revelations 2;*
- *Root Double: Before Crime, After Days;*
- *Shadow Tactics: Blades of the Shogun;*
- *Virtues Last Reward;*
- *World of Warcraft.*

Tabela 12 - Testes extra de Overlay

Aplicação Alvo	Máquina de Testes
Discord	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim (WIN7), Não (WIN10)**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**11 ms**

Aplicação Alvo	Máquina de Testes
Root Double: Before Crime, After Days	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**44 ms**

Aplicação Alvo	Máquina de Testes
GoG Galaxy	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim (WIN7), Não (WIN10)**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**15 ms**

Aplicação Alvo	Máquina de Testes
I am Setsuna	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**41 ms**

Aplicação Alvo	Máquina de Testes
Google Chrome	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim (WIN7), Não (WIN10)**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**17 ms**

Aplicação Alvo	Máquina de Testes
Kerbal Space Program	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**34 ms**

Aplicação Alvo	Máquina de Testes
Door Kickers	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Não**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**N/A**

Aplicação Alvo	Máquina de Testes
Nine Hours, Nine Persons , Nine Doors	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**37 ms**

Aplicação Alvo	Máquina de Testes
Shadow Tactics: Blades of the Shogun	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**41 ms**

Aplicação Alvo	Máquina de Testes
Virtues Last Reward	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**28 ms**

Aplicação Alvo	Máquina de Testes
World of Warcraft	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**34 ms**

Aplicação Alvo	Máquina de Testes
Resident Evil: Revelations 2	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**35 ms**

Aplicação Alvo	Máquina de Testes
Little Nightmares	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Não**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**N/A**

Aplicação Alvo	Máquina de Testes
Overwatch	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**33 ms**

Aplicação Alvo	Máquina de Testes
Mass Effect Andromeda	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Não**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**N/A**

Aplicação Alvo	Máquina de Testes
Mozilla Firefox	**i7 2600; GTX980TI; 8Gb DDR3**
Overlay	
É capaz de capturar *screenshots*	**Sim**
Afetou a aplicação alvo	**Não**
Tempo médio de resposta	**20 ms**

Após analisados os resultados destas tabelas foi possível tirar as seguintes conclusões:

- O módulo de deteção de texto, quando calibrado corretamente, apresenta resultados muito positivos. A ocorrência ocasional de falsos-positivos deve-se ao facto de o módulo desenvolvido propositadamente não conter qualquer identificação específica de carateres. Esta decisão foi tomada com o objetivo de manter o grau de complexidade relativamente baixo, e tornar o sistema capaz de, sem qualquer tipo de treino, identificar qualquer tipo de escrita, constituída por qualquer tipo de carateres.
- Tendo em conta que o módulo de *overlay* não interage diretamente com a aplicação alvo (o que aconteceria num *overlay* para aplicações em *exclusive fullscreen*), o *framerate* desta nunca é afetado pela aplicação de tradução, salvo a influência causada pela maior utilização dos recursos da máquina em uso. Por outro lado, o processo de obtenção de *screenshots* que foi usado não foi capaz de obter resultados em 3 das 22 aplicações quando testadas em Windows 7, e 6 das mesmas 22 em Window 10.

  O tempo de resposta varia de acordo com o tamanho da *Lens Window*, tal como seria de esperar, quanto maior o *screenshot* a obter, maior será o tempo de processamento. No entanto, os valores mantiveram-se relativamente baixos, sendo que a média mais alta rondou os 44 milissegundos.
- O módulo de *OCR* foi capaz de mostrar resultados muito positivos em línguas que utilizam carateres latins, gregos e cirílicos. Para certos silabários tal como o *hiragana* e *katakana* (parte dos carateres utilizados em japonês) os resultados foram também bastante positivos, no entanto quando a complexidade dos carateres aumenta, tal como nos carateres chineses (ou *kanji* japonês) ou no *hangul* coreano o nível de precisão do sistema desce rapidamente. Isto é especialmente problemático visto que a alteração de um único caráter chinês, altera por completo a sua palavra, o que pode levar a traduções com resultados bastante peculiares.
- Os resultados do módulo de tradução assemelham-se aos obtidos pelo módulo de *OCR* no facto de serem bastante positivos para línguas tais como Inglês, Português, Espanhol, Francês, Alemão e Russo, mas mostra dificuldades em frases mais complexas em Chinês,

Japonês e Coreano, na grande maioria das vezes devido à má interpretação do contexto da frase.

De qualquer forma, ignorando o resultado mais extremo, a média mais baixa de inteligibilidade e fidelidade (5.175; 5.273 em chinês no jogo *Rise of the Tomb Raider*) manteve-se superior a 5/10, o que indica que muito do texto traduzido manteve um nível pelo menos compreensível.

Por fim, foi ainda testada uma aplicação capaz de interagir corretamente com o *ITH*.

Nos resultados apresentados na Tabela 13 é visível um melhoramento de 99.45 para 100% na percentagem de carateres corretamente obtidos. Esta melhoria pode parecer pequena, no entanto foi suficiente para afetar de forma beneficial a qualidade da tradução obtida.

Apesar de não estar apresentado na tabela, outro beneficio do uso do *ITH* foi uma visível melhoria nos tempos de resposta do *DOT*, visto que o sistema de *OCR* é, na grande maioria das vezes, responsável pela maior porção do tempo de processamento desta aplicação.

Tabela 13 - Teste de Suporte a Text Hooking

Aplicação Alvo	**Máquina de Testes**	
きっと、澄みわたる朝色よりも (Kitto, Sumiwataru Asairo Yori mo)	**i5 4200U; GTX740M; 4Gb DDR3**	
Língua		
De	**Japonês**	**Japonês**
Para	**Inglês**	**Inglês**
Overlay		
É capaz de capturar *screenshots*	**Sim**	**Sim**
Afetou a aplicação alvo	**Não**	**Não**
Tempo médio de resposta	**15 ms**	**15 ms**
Deteção e Segmentação de Texto		
Linhas corretamente encontradas	**52**	**52**
Linhas incorretamente ignoradas	**0**	**0**
Falso-positivos	**0**	**0**
Tempo médio de resposta	**44 ms**	**41 ms**
OCR	**Sem ITH**	**Com ITH**
Carateres corretamente identificados	**1997**	**2008**
Carateres incorretamente identificados	**11**	**0**
Percentagem de precisão	**99.45%**	**100.00%**
Tradução		
Nível Médio de Inteligibilidade (0-10)	**7.212**	**7.5**
Nível Médio Fidelidade (0-10)	**7.364**	**8.156**
Mínimo de Inteligibilidade (0-10)	**2**	**3**
Mínimo Fidelidade (0-10)	**2**	**3**
Máximo de Inteligibilidade (0-10)	**10**	**10**
Máximo Fidelidade (0-10)	**10**	**10**

6.1.5 Segunda Fase – Testes Não Funcionais

Para esta segunda fase de testes, a aplicação foi disponibilizada a um grupo de teste. Este grupo utilizou a aplicação sobre duas imagens de menus de jogos (em japonês; Figura 39 e Figura 41), uma imagem com background complexo em inglês (Figura 40) e um pequeno trailer de uma série (legendas em russo). Note-se também que apenas foram dadas instruções básicas de utilização da aplicação, e que de momento não existe nenhuma forma de tutorial que permita um conhecimento mais aprofundado da aplicação.

Assim, a aplicação foi testada de um ponto de vista não funcional, focando-se nos aspetos de facilidade de utilização e usabilidade da aplicação. Cada indivíduo pertencente a este grupo de teste preencheu no fim um questionário (Figura 36Figura 37Figura 38), que permitiu a análise de duas questões: Quão fácil é a calibração; e quão compreensível foi o resultado final.

Os resultados dos testes realizados estão apresentados nos gráficos das Figura 32, Figura 33, Figura 34 e Figura 35.

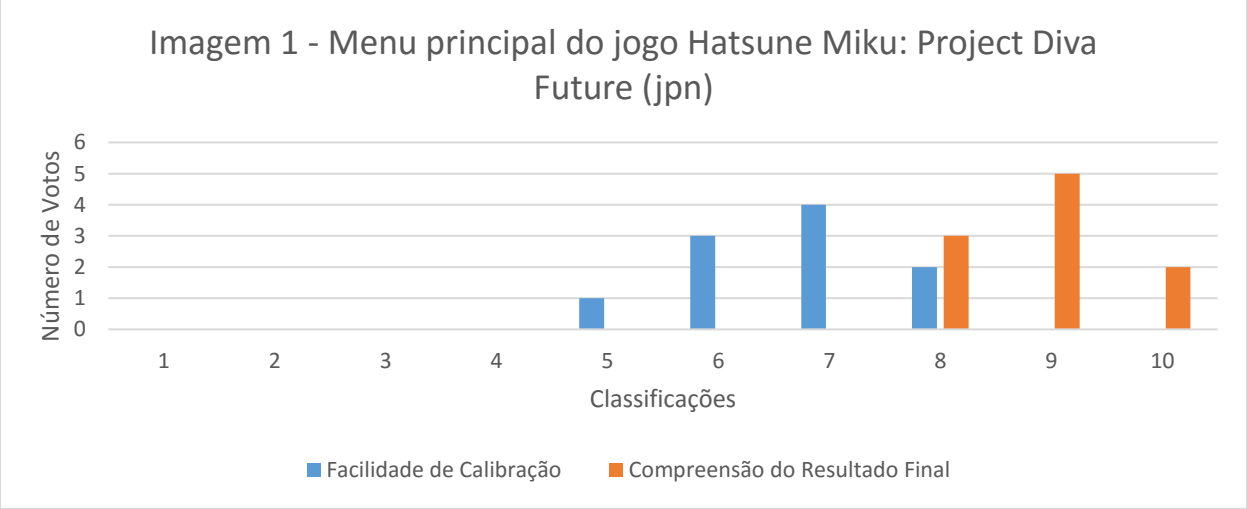


Figura 32 - Resultados dos testes de Utilização (Imagem 1)

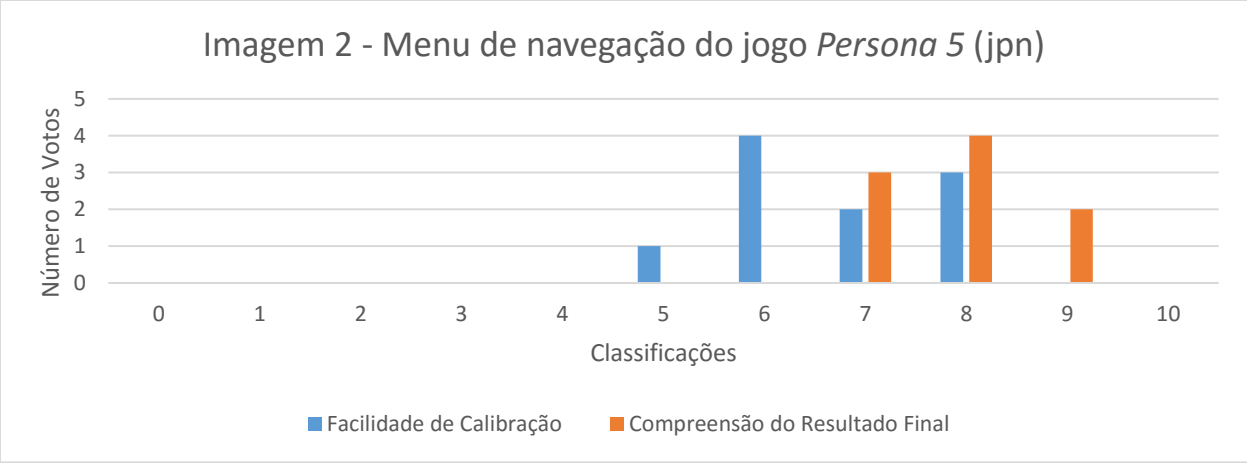


Figura 33 - Resultados dos testes de Utilização (Imagem 2)

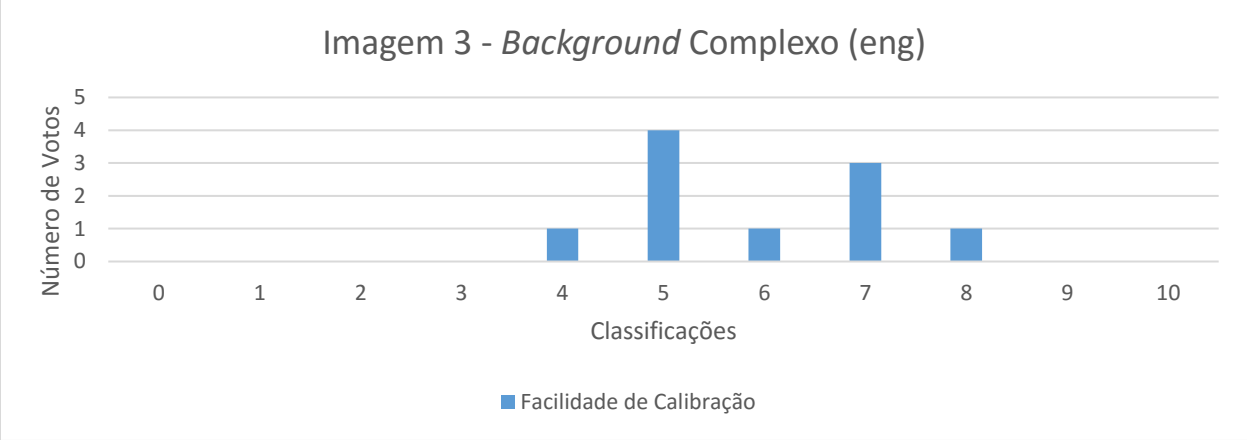


Figura 34 - Resultados dos testes de Utilização (Imagem 3)

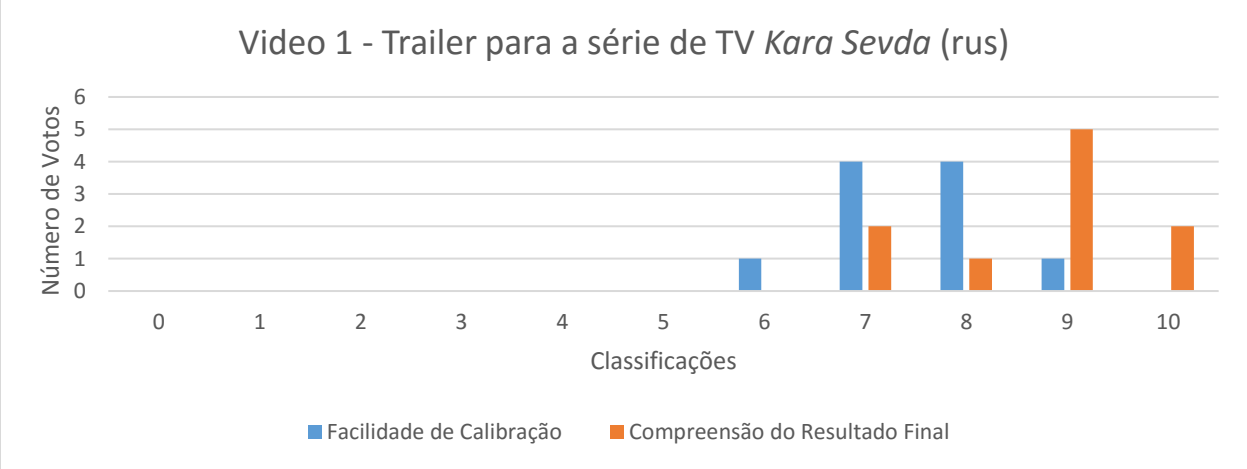


Figura 35 - Resultados dos testes de Utilização (Video 1)

Como é possível observar pelos resultados obtidos, um dos pontos fracos desta aplicação é o relativamente longo tempo de aprendizagem necessário para a utilizar de forma rápida e eficaz, devido ao grau de complexidade inerente à filtragem e segmentação de qualquer componente textual numa imagem. Para que qualquer texto possa ser corretamente interpretado pelo sistema de *OCR*, esta filtragem é obrigada a que o resultado final seja muito próximo de uma imagem perfeita do texto em questão, o que em alguns casos requer pequenas afinações manuais.

Assim, a construção de uma fase de tutoriais mais aprofundados iria beneficiar de forma muito significativa a compreensão do modo de utilização desta aplicação. Um outro ponto a melhorar seria também a consistência da qualidade da calibração automática, visto que certas variações no tamanho da área de calibração selecionada podem causar diferenças relevantes na qualidade da calibração do módulo de deteção e segmentação de texto.

Por fim, os resultados relativos à compreensão geral das respostas de tradução obtidas foram bastante positivos, o que mostra o potencial prático da aplicação assim que os seus aspetos de calibração sejam devidamente aperfeiçoados (ou dominados pelo utilizador no seu estado atual).

6.1.6 Respostas às Hipóteses Testadas

De seguida serão dadas respostas às hipóteses apresentadas em 6.1.2 de acordo com os resultados obtidos durante as fases de testes 6.1.4 e 6.1.5.

- **Segmentação de Texto**
    - **A aplicação é capaz de consistentemente encontrar todo o texto apresentado num grupo predefinido de aplicações*?***
        - ✓ Sim, desde que calibrada corretamente, salvo raras exceções.
    - **Caso não seja capaz de consistentemente encontrar todos os blocos de texto, a aplicação é capaz de manter uma média de identificações corretas superior a 80%?**
        - ✓ Sim, a percentagem de linhas ignoradas mais alta é inferior a 8%.

- **A aplicação é capaz de consistentemente separar devidamente o texto em caixas (*bounding boxes*) de uma só linha?**
  - ✓ Sim, em nenhum dos casos testados foram identificadas falhas deste tipo.
- **A aplicação é capaz de consistentemente apresentar uma média de falso-positivos inferior a 20% num grupo predefinido de aplicações*?***
  - ✓ Sim, o valor mais alto encontrado foi de 16%.

- ***OCR***
  - **A aplicação é capaz manter uma média superior a 80% de identificações corretas de carateres num grupo predefinido de aplicações*?***
    - ✓ Sim, o valor mais baixo encontrado foi de 87.34% e a média geral de 95.2%.

- ***Machine Translation***
  - **Tendo em conta a escala de 0 a 10 definida anteriormente, a aplicação é capaz de manter uma média de inteligibilidade superior a 5 num grupo predefinido de aplicações*?***
    - ✓ Sim, a média geral é de cerca de 8.01/10.
  - **Tendo em conta a escala de 0 a 10 definida anteriormente, a aplicação é capaz de manter uma média de fidelidade superior a 5 num grupo predefinido de aplicações*?***
    - ✓ Sim, a média geral é de cerca de 8.22/10.

- ***Overlay***
  - **A aplicação é capaz de realizar *overlays* sobre aplicações em *windowed mode,* num grupo predefinido de aplicações?**
    - ✓ Sim, salvo algumas exceções (13.6% dos programas testados foram incompatíveis com o sistema desenvolvido).
  - **A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *DirectX9* em *Fullscreen mode,* num grupo predefinido de aplicações?**
    - × Não.

- **A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *DirectX11* em *Fullscreen mode,* num grupo predefinido de aplicações?**
  - ✗ Não.
- **A aplicação é capaz de realizar *overlays* sobre aplicações baseadas em *OpenGL (4.0)* em *Fullscreen mode,* num grupo predefinido de aplicações?**
  - ✗ Não.
- **O tempo médio de resposta do sistema de *overlay* mantem-se inferior a 200 milissegundos, num grupo predefinido de aplicações?**
  - ✓ Sim, os tempos de resposta mantiveram-se sempre abaixo de 45ms.
- **A diferença média de tempo por ciclo de *rendering* medida entre a aplicação quando afetada ou não pelo *overlay* mantém-se inferior a 30 milissegundos, num grupo predefinido de aplicações?**
  - ✓ Sim, o método de *overlay* utilizado não afeta diretamente a aplicação alvo.

- **Testes não funcionais**
  - **A apresentação da aplicação é atrativa para qualquer tipo de utilizador?**
    - ✓ Em geral as respostas relativas ao aspeto da interface foram positivas.
  - **A aplicação foi capaz de apresentar uma explicação coerente e simples de todos os conceitos necessários para a sua operação?**
    - ✗ Não, deve ainda ser desenvolvido um conjunto de tutoriais compreensivos da aplicação.
  - **O modo de utilização da aplicação é simples de aprender?**
    - ◈ Muito dependente do utilizador, no entanto a maioria dos resultados foram favoráveis.
  - **A aplicação é simples de utilizar de forma a apresentar resultados favoráveis?**
    - ◈ Dependendo da aplicação alvo, pode ser muito simples ou bastante trabalhoso.
  - **Os resultados são apresentados de forma compreensível sem prejudicar a aplicação alvo de forma percetível?**
    - ✓ Sim, na maioria dos casos os resultados foram compreensíveis e não chegaram a afetar a utilização correta da aplicação.

- **Esta solução apresenta resultados de forma mais rápida e mais prática comparativamente com a sua versão manual?**
  - ✓ Sim, em certos casos a versão manual nem sequer funcionaria sem uma edição e filtragem manual correta da imagem antes da sua passagem para o sistema de *OCR.*

6.2 Conclusão

Neste capítulo foram apresentados os resultados dos vários testes realizados à aplicação, o que permitiu que fossem tiradas conclusões relativas ao grau de funcionamento da aplicação, assim como à sua componente de interação com o utilizador. Os pontos de destaque que foram encontrados foram os seguintes:

- Existe uma pequena percentagem de aplicações (13.6% das testadas) que não permite a obtenção de *screenshots* do modo que o módulo de *overlay* foi desenvolvido. Este método terá que ser revisto no futuro.
- Apesar de muitas vezes ser necessário ajustar manualmente as suas calibrações, o módulo de deteção de texto provou o seu funcionamento mesmo contra *backgrounds* complexos tais como os da tela de um filme.
- Dado o mesmo tamanho de letra, o sistema de *OCR* mostra muito mais dificuldades ao interpretar carateres complexos tais como os chineses e coreanos, em comparação com carateres latins, gregos ou cirílicos. Apesar destas dificuldades, a utilização do filtro desenvolvido para o módulo de deteção de texto em conjunto com as capacidades da versão mais recente do sistema de *OCR Tesseract*, a média de precisão mais baixa obtida foi de 95.2% com carateres coreanos. A média mais alta da amostra de testes realizados foi de 99.8%, com texto em português.
- Tanto o sistema de tradução da *Google* como o da *Yandex* ainda estão muito longe de realizar traduções que rivalizem com a de um tradutor humano. No entanto, salvo algumas exceções em línguas como o chinês, coreano e japonês, os resultados apresentados são compreensíveis o suficiente para, por exemplo, navegar por um menu de configurações, ou compreender as legendas de um filme.

7 Conclusão e Perspetivas de Trabalho Futuro

Esta dissertação teve como foco principal a idealização de uma ferramenta capaz de traduzir qualquer interação por texto de uma aplicação, de forma a facilitar a compreensão e utilização de aplicações cuja língua utilizada não seja falada pelo utilizador em questão. Para isto, a aplicação teorizada (e desenvolvida) deve ser capaz de identificar, através de técnicas de *OCR*, todo o texto encontrado e segmentado de uma dada aplicação alvo, traduzindo-o tirando partido de sistemas de *machine translation* atuais, e por fim apresentando-o diretamente sobre o texto original. Tudo isto deve ser realizado em tempo real, o que implica que os tempos de processamento se mantenham relativamente baixos.

Desta forma, foram realizados estudos do estado da arte que incluíram o estado atual das tecnologias de deteção e segmentação de texto, *OCR*, *machine translation* e overlay sobre aplicações gráficas. Os resultados destes estudos foram então avaliados, tendo algumas das ferramentas estudadas sido integradas no protótipo final.

Com o objetivo de manter os tempos de processamento baixos e melhorar os resultados das ferramentas de *OCR* em situações complexas, uma grande parte do desenvolvimento deste protótipo foi dedicada ao desenvolvimento de um novo algoritmo específico para uma eficiente deteção e segmentação de texto, pelo que o foco principal do capítulo de desenvolvimento foi reservado para este mesmo tópico. A este protótipo final foi dado o nome de *Direct Text Overlay*, ou *DOT.*

O *Direct Text Overlay* provou facilitar o processo de tradução inerente à compreensão de qualquer aplicação, imagem ou vídeo que apresente texto, de outra forma, incompreensível a um utilizador que não domine a língua em questão. Os resultados atuais deste sistema demonstram que, apesar da falta de qualidade inerente às técnicas atuais de *machine translation*, é possível compreender a vasta maioria do conteúdo processado, caso este se mantenha curto e direto, tal como o conteúdo encontrado em menus de aplicações, assistentes de instalação e até

certos vídeos legendados. Esta aplicação, tendo em conta a sua grande dependência aos sistemas de *OCR* e *machine translation* atuais, verá potencialmente grandes melhoramentos na qualidade de tradução apresentada, à medida que estas tecnologias evoluírem. Isto é especialmente verdade tendo em conta a sua estrutura modular, que permite que a qualquer momento, tirando partido da *API* fornecida pela aplicação, qualquer módulo, quer este seja a componente de deteção de texto, tradução, OCR ou *overlay*, possa ser substituído por uma versão superior, ou especializada para uma qualquer outra funcionalidade.

O *DOT* foi testado sobre 22 diferentes aplicações, mostrando um funcionamento correto em 19 destas (em Windows 7). As restantes 3 mostram, no entanto, que o sistema de *overlay* atual ainda não é perfeito e, desta forma, precisará de sofrer uma atualização no futuro. Para além desta atualização, será também necessário desenvolver a componente do sistema de *overlay* que permitirá que este funcione sobre aplicações em modo *exclusive fullscreen* (o sistema atual apenas permite a captura de imagem em aplicações em janela ou *borderless window*).

Num futuro próximo deverá também ser desenvolvido por completo o menu de ajuda e a apresentação de tutoriais de interação com a aplicação. Será também ainda desenvolvido o conteúdo da secção destinada a *developers* que tenham interesse em expandir a aplicação, explicando em detalhe como interagir corretamente com a *API* de *plugin* de forma a obter os melhores resultados possíveis.

Finalmente, caso seja possível, serão ainda estudadas as melhores maneiras de aumentar o nível de automatização da calibração do sistema de deteção e segmentação de texto desenvolvido, de forma a permitir que o sistema se auto-calibre corretamente num maior número de cenários. Isto permitirá reduzir o tempo e nível de complexidade inerentes à calibração da aplicação em casos de difícil obtenção de resultados.

8 Referências

A9T9, 2015. *Free OCR Software for Windows 8.1/10.* [Online]
Available at: https://ocr.space/blog/p/free-ocr-software.html
[Acedido em 15 09 2017].

ABBYY, 2017. *ABBYY FineReader 14.* [Online]
Available at: https://www.abbyy.com/en-eu/finereader/
[Acedido em 15 09 2017].

Allee, V., s.d. *Understanding Value Networks.* [Online]
Available at: http://www.gurteen.com/gurteen/gurteen.nsf/id/L001020/$File/understanding-value-networks.pdf
[Acedido em 23 02 2017].

Alphatrad, s.d. *As 50 línguas mais faladas no mundo.* [Online]
Available at: http://www.alphatrad.pt/50-linguas-mais-faladas-no-mundo
[Acedido em 17 02 2017].

Anklam, P., s.d. *Value Network Analysis.* [Arte].

Anon., 2016. *Tesseract-ocr.* [Online]
Available at: https://github.com/tesseract-ocr/tesseract/

Anon., 2017. *Fraps - Record your gameplay.* [Online]
Available at: http://ccm.net/faq/5406-fraps-record-your-gameplay

Anon., 2017. *Machine Translation.* [Online]
Available at: https://en.wikipedia.org/wiki/Machine_translation

Anon., 2017. *Word Lens.* [Online]
Available at: https://en.wikipedia.org/wiki/Word_Lens
[Acedido em 06 01 2017].

Berdajs, J. & Bosnic, Z., 2010. *Extending applications using an advanced approach to DLL injection and API hooking,* Faculty of Computer and Information Science, University of Ljubljana, Trˇzaˇska cesta 25, 1000 Ljubljana, Slovenia: Wiley InterScience.

Canedo-Rodríguez, A. et al., 2012. *Efficient Text Extraction Algorithm Using Color Clustering for Language Translation in Mobile Phone,* Spain; USA; South Korea: JSIP.

Epshtein, Boris; Ofek, Eyal; Wexler, Yonatan; Microsoft Corporation, 2010. *Detecting Text in Natural Scenes with.* s.l., IEEE.

Garza, J. d. l. L. S., 2017. *How many Chinese characters are there?.* [Online]
Available at: http://www.hutong-school.com/how-many-chinese-characters-are-there

Google, 2016. *Google Cloud Translation API.* [Online]
Available at: https://cloud.google.com/translate/

Google, s.d. *Translate.google.com.* [Online]
Available at: https://translate.google.com/
[Acedido em 23 02 2017].

Goto, Hideaki; Tohoku University, 2014. *NHocr.* [Online]
Available at: https://github.com/fireae/nhocr
[Acedido em 15 09 2017].

Haggag, M., 2007. *DirectDraw Overlay Library.* [Online]
Available at: http://directdrawoverlaylib.codeplex.com/
[Acedido em 30 01 2017].

Heliński, M., Kmieciak, M. & Parkoła, T., s.d. *Report on the comparison of Tesseract and ABBYY FineReader OCR engines,* Poznań Supercomputing and Networking Center, Poland: IMPACT.

imageoverlayutility@gmail.com, 2013–2016. *Image Overlay Utility.* [Online]
Available at: https://imageoverlayutility.bitbucket.io/?
[Acedido em 30 01 2017].

Jichi, 2016. *VNR.* [Online]
Available at: http://sakuradite.com/wiki/en/vnr
[Acedido em 04 12 2016].

Jouline, A. & Robinson, D., 2006. *Taksi.* [Online]
Available at: http://taksi.sourceforge.net/
[Acedido em 30 01 2017].

Koen, P. A., 2004. *web.stevens.edu.* [Online]
Available at: https://web.stevens.edu/cce/NEW/PDFs/FEIEffectiveMethodstoolWorkshop.pdf
[Acedido em 20 02 2017].

Kumar, A., 2014. *An Efficient Approach for Text Extraction in Images and Video Frames Using Gabor Filter,* s.l.: IJCEE.

Lienhart,, R. & Wernicke, A., 2002. *Localizing and Segmenting Text in Images and Videos,* s.l.: IEEE.

Li, H., Doermann, D. & Kia, O., 1998. *Automatic text detection and tracking in Digital Video,* s.l.: s.n.

Lindgreen, A. & Wynstra, F., 2005. *Value in business markets: What do we know? Where are we going?,* indhoven & Rotterdam, Holanda: Elsevier.

Metty, 2007. *Overlay Tools.* [Online]
Available at: https://www.codeproject.com/Articles/20764/Overlay-Tools
[Acedido em 30 01 2017].

Metty, 2008. *Overlay using raphook.dll.* [Online]
Available at: https://www.codeproject.com/Articles/29530/Overlay-using-raphook-dll
[Acedido em 30 01 2017].

Microsoft, 2016. *Microsoft Translator Text API.* [Online]
Available at: https://www.microsoft.com/en-us/translator/translatorapi.aspx

Microsoft, 2016. *Use Magnifier to see items on the screen.* [Online]
Available at: https://support.microsoft.com/en-us/help/11542/windows-use-magnifier
[Acedido em 30 01 2017].

Microsoft, s.d. *GetGlyphOutline function.* [Online]
Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/dd144891(v=vs.85).aspx
[Acedido em 16 02 2017].

Microsoft, s.d. *GetTextExtentPoint32 function.* [Online]
Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/dd144938(v=vs.85).aspx
[Acedido em 16 02 2017].

Olson, P., 2010. *Hot, New 'Word Lens' App Is Founder's First Project In Augmented Reality.* [Online]
Available at: http://www.forbes.com/sites/parmyolson/2010/12/22/hot-new-word-lens-app-is-founders-first-project-in-augmented-reality/#77a3f9a622dc
[Acedido em 16 02 2017].

Richter, J. & Nasarre, C., 2007. *Windows via C/C++ 5th Edition.* 5ª ed. Redmond, Washington, US: Microsoft Press.

Sanmartín, A. P., 2015. *Scene Text Segmentation Based on Thresholding,* Zagreb, Croatia: Fakultet elektrotehnike, strojarstva i brodogradnje.

Schulenburg, J., 2013. *GOCR open-source character recognition.* [Online]
Available at: http://jocr.sourceforge.net/
[Acedido em 15 09 2017].

Systran, 2016. *About Us.* [Online]
Available at: https://www.babelfish.com/about-us/

Temarez, 2017. *CuneiForm (software).* [Online]
Available at: https://en.wikipedia.org/wiki/CuneiForm_(software)
[Acedido em 15 09 2017].

WayGo, s.d. *WayGo.* [Online]
Available at: http://www.waygoapp.com/
[Acedido em 04 12 2016].

Weld, C., 2017. *GitHub.* [Online]
Available at: https://github.com/tdhintz/tesseract4win64/compare/4.0.0...master
[Acedido em 30 03 2017].

Wirth, K., 2017. *ABBYY Releases Version 14 of FineReader OCR and Document-Capture Software.* [Online]
Available at: http://wirthconsulting.org/2017/01/25/abbyy-releases-version-14-of-finereader-ocr-and-document-capture-software/
[Acedido em 19 09 2017].

Wong, E., 2008. *Custom Desktop Logo.* [Online]
Available at: http://customdesktoplogo.wikidot.com/
[Acedido em 30 01 2017].

Woodall, T., 2003. *Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis,* Nottingham, Inglaterra: Academy of Marketing Science Review.

Yandex, 2016. *Yandex Translate Developers.* [Online]
Available at: https://translate.yandex.com/developers

Yangxing, L., Goto, S. & Ikenaga, T., 2006. *A Contour-Based Robust Algorithm for Text Detection in Color Images,* Japan: IEICE TRANS.

Yin-Poole, W., 2012. *Origin gets in-built Twitch streaming as EA continues to improve games platform.* [Online]
Available at: http://www.eurogamer.net/articles/2012-11-08-origin-gets-in-built-twitch-streaming-as-ea-continues-to-improve-games-platform

Youtube, s.d. [Online]
Available at: https://i.ytimg.com/vi/75A4UUkl5as/maxresdefault.jpg

Youtube, s.d. *i.ytimg.com.* [Online]
Available at: https://i.ytimg.com/vi/pykMfsdPLNY/maxresdefault.jpg

z. et al., 2017. *Tesseract OCR.* [Online]
Available at: https://github.com/tesseract-ocr/tesseract
[Acedido em 15 09 2017].

9 Anexos

Tabela 14 - Exemplo do método de teste das capacidades da aplicação DOT (Parte 1)

Real time translation (00:09:12 - 00:15:40)	The Hobbit The Desolation of Smaug (00:09:12 - 00:15:40)			
Idioma Original	**Russo**			
DOT Translation (Google Translator)	**Russo para Português**	**Tradução Humana**	Inteligibilidade	Fidelidade
DOT OCR	Что это?		0	
DOT TL	O que é isso?	O que é aquilo?	9	7
DOT OCR	Это хозяин дома.			
DOT TL	Este dono da casa.	Aquilo é o nosso anfitrião.	5	8
DOT OCR	Его зовут Беорн.		0	
DOT TL	Seu nome é Beorn.	O seu nome é Beorn.	10	10
DOT OCR	Он оборотень.		0	
DOT TL	Ele é um lobisomem.	E é um lobisomem.	10	10
DOT OCR	Иногда он громадный чёрный медведь, а иногда громадный.		0	
DOT TL	Às vezes é um enorme urso preto, às vezes enorme.	Algumas vezes é um enorme urso negro	8	10
DOT OCR	МОГУЧИЙ человек.		0	
DOT TL	homem poderoso.	outras é um homem grande e forte.	8	5
DOT OCR	медведь непредсказуем. Но с человеком можно договориться.		0	
DOT TL	Tenha imprevisível. Mas com um homem pode ser negociado.	O urso é imprevisível, mas com o homem podemos chegar à razão.	6	8
DOT OCR	Однако, он не любит гномов.		0	
DOT TL	No entanto, ele não gosta de anões.	ele não gosta muito de anões.	10	10
DOT OCR	Он уходит.		0	
DOT TL	Ele deixa.	Está a ir-se embora.	8	8
DOT OCR	Отойди оттуда!		0	
DOT TL	Ficar longe de lá!	Afasta-te daí!	8	8
DOT OCR	Это ненормально, от и до!		0	
DOT TL	Não é normal, e por fora!	Não é natural. Nadinha.	7	8
DOT OCR	ОЧЕВИДНО, на нем чьи-то тёмные чары.		0	
DOT TL	Obviamente que feitiço escuro de alguém.	É óbvio que ele está sob o efeito de algum feitiço negro	5	7
DOT OCR	Не будь дураком. На нем нет чар, кроме его собственных.		0	
DOT TL	Não seja um tolo. Não é de encantamento, mas a sua própria.	Não sejas tolo. O único encantamento que possui é o seu próprio.	5	7
DOT OCR	НУ, а теперь, немного поспите, этой ночью вы будете в безопасности.		0	
DOT TL	Bem, agora, dormir um pouco aquela noite você estará seguro.	Muito bem, vejam se dormem. Aqui estarão a salvo esta noite.	5	7
DOT OCR	Я надеюсь.		0	
DOT TL	Espero.	Espero eu…	10	10
DOT OCR	Атакуй их сейчас.		0	

DOT TL	Atacá-los agora.	Ataque-os agora.	8	10
DOT OCR	Убей гномье отродье, пока они спят.		0	
DOT TL	Matar a prole anão enquanto eles dormem.	Mate esses anões sujos enquanto dormem.	8	9
DOT OCR	Нет.		0	
DOT TL	Não.	Não!	10	10
DOT OCR	Зверь стоит на страже.		0	
DOT TL	A besta fica de guarda.	A Criatura está de guarda.	8	10
DOT OCR	Мы убъём их в пути.		0	
DOT TL	Vamos matá-los de uma forma.	Vamos matá-los na estrada.	4	6
DOT OCR	Они собираются в Дол Гулдуре.		0	
DOT TL	Eles estão indo para Dol Guldur.	Eles estão a juntar-se em Dol Guldur.	10	9
DOT OCR	Хозяин зовёт тебя.		0	
DOT TL	O proprietário está chamando você.	O Senhor convocou-te!	10	8
DOT OCR	Наше число растёт.		0	
DOT TL	Nosso número está crescendo.	Estamos a crescer em número.	10	10
DOT OCR	Наша сила растёт.		0	
DOT TL	Nossa força cresce.	A crescer em força.	10	10
DOT OCR	Ты поведёшь мои армии.		0	
DOT TL	Você povedosh meu exército.	Irás liderar os meus exércitos.	3	5
DOT OCR	А что насчёт Дубощита?		0	
DOT TL	E sobre Oakenshield?	Então e o Escudo-de-Carvalho?	10	10
DOT OCR	Близится война.		0	
DOT TL	Aproximando-se a guerra.	A guerra aproxima-se	6	10
DOT OCR	Ты обещал мне его голову.		0	
DOT TL	Você me prometeu sua cabeça.	Prometeu-me a cabeça dele!	10	10
DOT OCR	Смерть придёт ко всем.		0	
DOT TL	A morte vem para todos.	A morte virá para todos!	9	10
DOT OCR	Мы прекращаем охоту?		0	
DOT TL	Nós paramos de caçar?	Cancelamos a caçada?	8	10
DOT OCR	Больг!		0	
DOT TL	Bolg!	Bolg!	10	10
DOT OCR	У меня заданййе для тебя.		1	
DOT TL	I zadanyye para você.	Tenho uma tarefa para ti.	2	4
DOT OCR	Ты ещё жаждешь гномьей крови?		0	
DOT TL	Você ainda anseiam Gnome sangue?	Ainda estás sedento por sangue de Anão?	6	9
DOT OCR	Значит ты тот, кого зовут Дубощитом.		0	
DOT TL	Então você é o único que é chamado de Escudo de Carvalho.	Então és tu o tal a que chamam de Escudo-de-Carvalho.	7	10
DOT OCR	Скажи, почему Азог Осквернитель охотится за тобой?		0	
DOT TL	Diga-me, por que o Azog the Defiler caza por você?	Diz-me, por que razão Azog, o Profanador te está a caçar?	8	10
DOT OCR	Ты знаешь Азога. Откуда?		0	
DOT TL	Você sabe Azog. Onde?	Conhece o Azog? Como?	5	7

DOT OCR	Мой народ был первым, кто обитал в горах.		0	
DOT TL	Meu povo foram os primeiros a viver nas montanhas.	O meu povo foi o primeiro a viver nas montanhas.	10	10
DOT OCR	Еще до того, как орки пришли с севера.		1	
DOT TL	Mesmo antes dos orcs vieram do norte.	Antes dos Orcs terem descido do norte.	7	10
DOT OCR	Осквернитель убил многих из моей семьи.		0	
DOT TL	Defiler matou muitos da minha família.	O Profanador matou quase toda a minha família.	10	9
DOT OCR	А некоторых пленил, но не для работы, вы же понимаете, а для развлечения.		0	
DOT TL	E alguns dos cativos, mas não para o trabalho, você sabe, e para o entretenimento.	Mas alguns ele escravizou. Não para trabalhar, entendem? Mas por desporto.	7	9
DOT OCR	Казалось, что держать оборотней в клетках и истязать их, забавляло его.		0	
DOT TL	Parecia que manter os lobisomens em gaiolas e torturá-los, divertiu-se.	Enjaular lobisomens e torturá-los parecia diverti-lo.	6	9
DOT OCR	Есть и другие вроде тебя?		0	
DOT TL	Há outros como você?	Há outros como você?	10	10
DOT OCR	Когда то было много.		0	
DOT TL	Quando era muito.	Outrora fomos muitos.	4	7
DOT OCR	А теперь?		0	
DOT TL	E agora?	E agora?	10	10
DOT OCR	Теперь всего лишь один.		0	
DOT TL	Agora apenas um.	Agora só há um.	10	10

Tabela 15 - Exemplo do método de teste das capacidades da aplicação DOT (Parte 2)

Original	**English**
Texto Original	Go now. While you have the light. (00:17:20)
DOT OCR	Go now. While you have the light.
DOT OCR Precisão	100%
Idioma Traduzido	**French**
Tradução Original	Partez tout de suite, tant qu'il fait jour.
DOT OCR	Partez tout de suite, tant qu'il fait jour.
DOT OCR Precisão	100%
DOT Translation	
Google Translator	Allez maintenant. Pendant que vous avez la lumière.
Yandex Translator	Aller maintenant. Alors que vous avez la lumière.
DOT TL de volta ao Original	
Google Translator	Go now, as long as it is day.
Yandex Translator	Leave immediately, as long as it is day.

Idioma Traduzido	German
Tradução Original	Geht jetzt. Solange es noch hell ist.
DOT OCR	Geht jetzt. Solange es noch hell ist.
DOT OCR Precisão	100%
DOT Translation	
Google Translator	Geh jetzt. Während du das Licht hast.
Yandex Translator	Gehen Sie jetzt. Während Sie das Licht haben.
DOT TL de volta ao Original	
Google Translator	Go now. As long as it is still light.
Yandex Translator	Now. While it is still light.

Idioma Traduzido	Portuguese
Tradução Original	o profanador te esta a caçar?
DOT OCR	o profanador te esta a caçar?
DOT OCR Precisão	100%
DOT Translation	
Google Translator	vá agora. Enquanto você tem a luz.
Yandex Translator	Vá agora. Enquanto tendes luz.
DOT TL de volta ao Original	
Google Translator	Go now while you still have light.
Yandex Translator	Go now while you still have light.

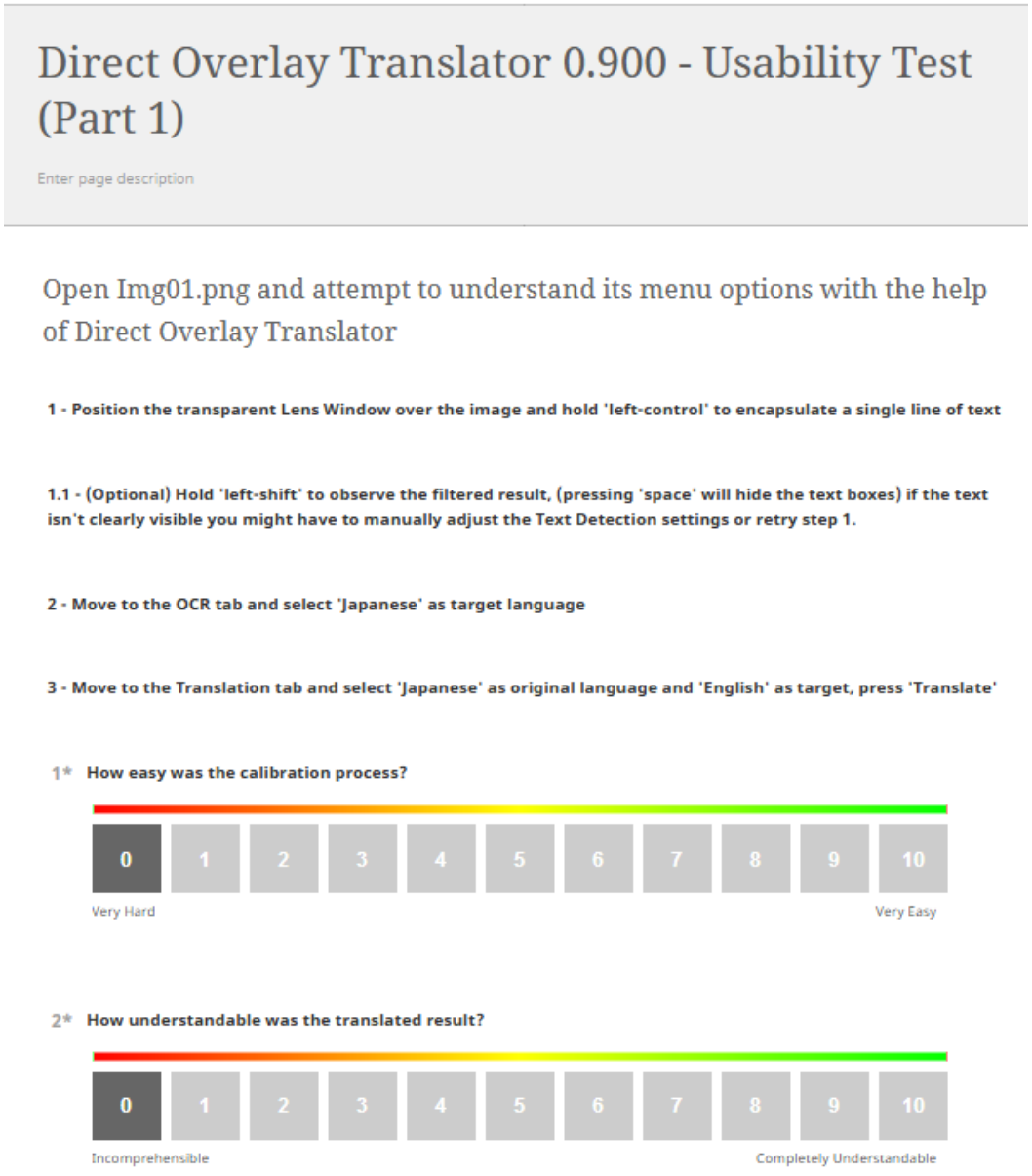
Idioma Traduzido	Spanish
Tradução Original	Váyanse ahora. Mientras aún hay luz.
DOT OCR	Váyanse ahora. Mientras aún hay luz.
DOT OCR Precisão	100%
DOT Translation	
Google Translator	Ve ahora. Mientras tienes la luz.
Yandex Translator	Vaya ahora. Mientras tenéis la luz.
DOT TL de volta ao Original	
Google Translator	Go away now. While there is still light.
Yandex Translator	Evacuate now. While there is still light.

Idioma Traduzido	Chinese
Tradução Original	快走吧趁現在天亮.
DOT OCR	快走吧趁現在天亮.
DOT OCR Precisão	100%
DOT Translation	
Google Translator	現在出發。當你有光。
Yandex Translator	现在去。虽然你有光。

DOT TL de volta ao Original	
Google Translator	Come and take advantage of the dawn now.
Yandex Translator	Let's go before now in the day light.

Idioma Traduzido	**Japanese**
Tradução Original	明るいうちに　早く行け
DOT OCR	明るいうちに　早く行け
DOT OCR Precisão	100%
DOT Translation	
Google Translator	今行く。あなたは光を持っている間。
Yandex Translator	行きます。 中の光になります。
DOT TL de volta ao Original	
Google Translator	Go early while you are bright
Yandex Translator	Bright, would soon go

Idioma Traduzido	**Russian**
Tradução Original	Идите сейчас, пока светло.
DOT OCR	Идите сейчас, пока светло.
DOT OCR Precisão	100%
DOT Translation	
Google Translator	Иди сейчас. Пока у тебя есть свет.
Yandex Translator	Идти сейчас. Пока есть свет.
DOT TL de volta ao Original	
Google Translator	Go now, while it's light.
Yandex Translator	Go now, while it's light.



Direct Overlay Translator 0.900 - Usability Test (Part 1)

Enter page description

Open Img01.png and attempt to understand its menu options with the help of Direct Overlay Translator

1 - Position the transparent Lens Window over the image and hold 'left-control' to encapsulate a single line of text

1.1 - (Optional) Hold 'left-shift' to observe the filtered result, (pressing 'space' will hide the text boxes) if the text isn't clearly visible you might have to manually adjust the Text Detection settings or retry step 1.

2 - Move to the OCR tab and select 'Japanese' as target language

3 - Move to the Translation tab and select 'Japanese' as original language and 'English' as target, press 'Translate'

1* **How easy was the calibration process?**

0 1 2 3 4 5 6 7 8 9 10

Very Hard — Very Easy

2* **How understandable was the translated result?**

0 1 2 3 4 5 6 7 8 9 10

Incomprehensible — Completely Understandable

Figura 36 - Questionário para grupo de testes de usabilidade (Parte 1)

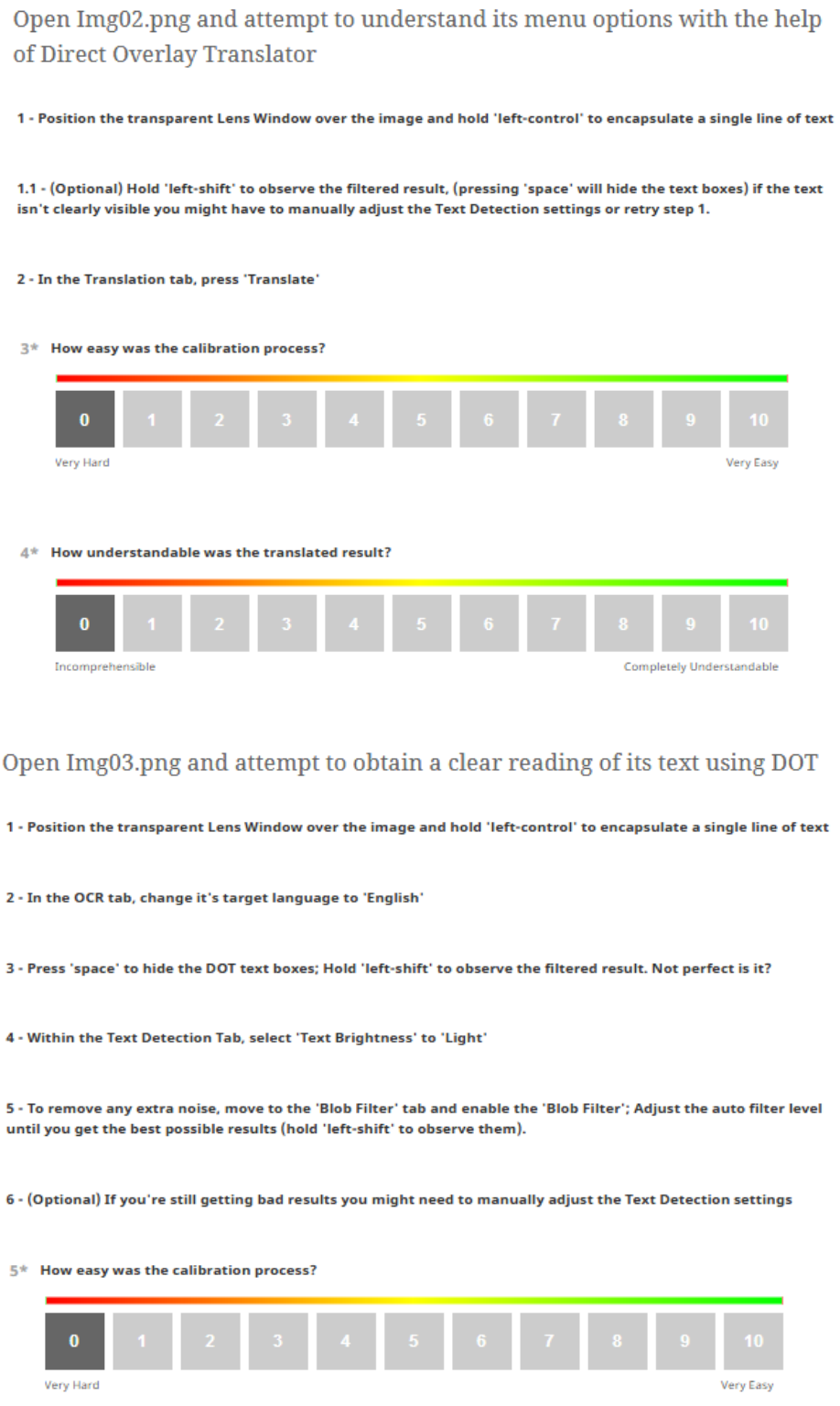
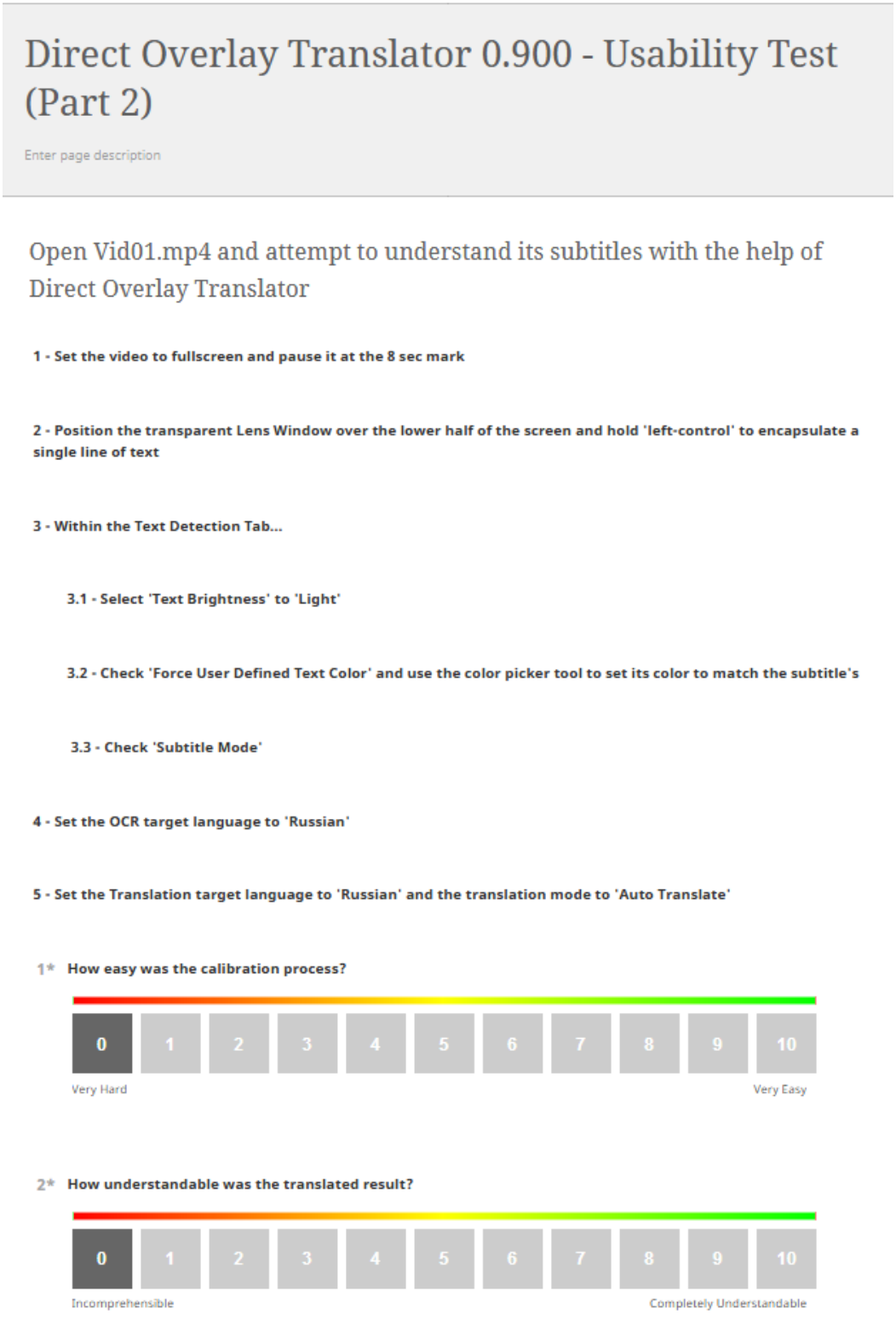


Figura 37 - Questionário para grupo de testes de usabilidade (Parte 2)



Direct Overlay Translator 0.900 - Usability Test (Part 2)

Enter page description

Open Vid01.mp4 and attempt to understand its subtitles with the help of Direct Overlay Translator

1 - Set the video to fullscreen and pause it at the 8 sec mark

2 - Position the transparent Lens Window over the lower half of the screen and hold 'left-control' to encapsulate a single line of text

3 - Within the Text Detection Tab...

3.1 - Select 'Text Brightness' to 'Light'

3.2 - Check 'Force User Defined Text Color' and use the color picker tool to set its color to match the subtitle's

3.3 - Check 'Subtitle Mode'

4 - Set the OCR target language to 'Russian'

5 - Set the Translation target language to 'Russian' and the translation mode to 'Auto Translate'

1* **How easy was the calibration process?**

0 1 2 3 4 5 6 7 8 9 10

Very Hard — Very Easy

2* **How understandable was the translated result?**

0 1 2 3 4 5 6 7 8 9 10

Incomprehensible — Completely Understandable

Figura 38 - Questionário para grupo de testes de usabilidade (Parte 3)

Figura 39 - Img01 utilizada durante os testes de usabilidade

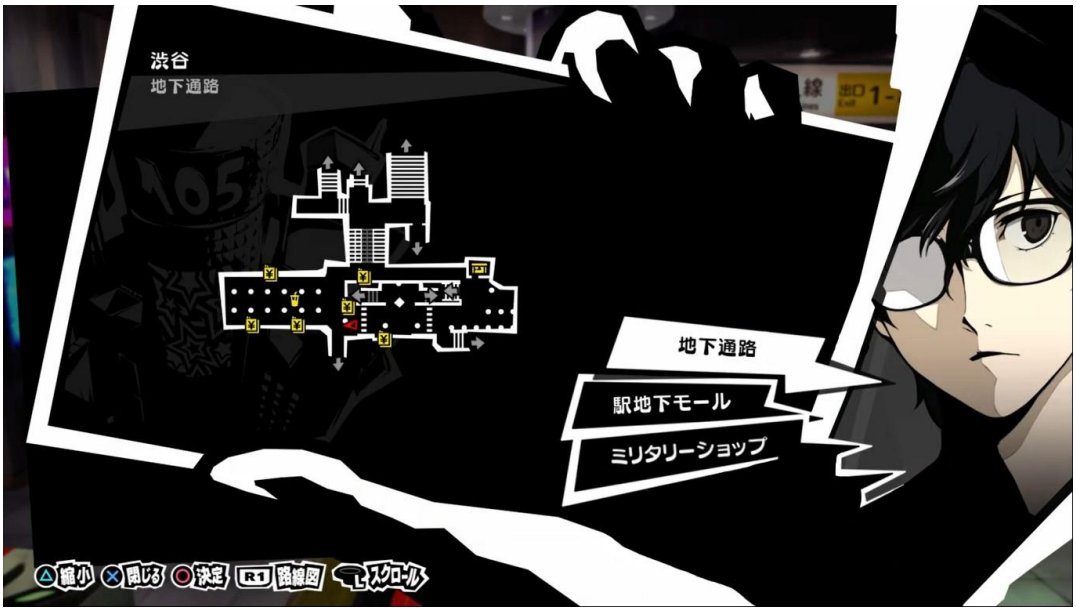


Figura 41 - Img02 utilizada durante os testes de usabilidade

Figura 40 - Img03 utilizada durante os testes de usabilidade