

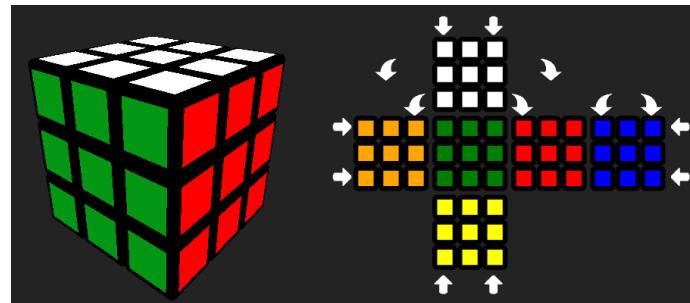


ISEL – INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
ADEETC – ÁREA DEPARTAMENTAL DE ENGENHARIA DE
ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES

LEIM

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA
UNIDADE CURRICULAR DE PROJETO

PFC Rubik's Cube



Carlos Rosa 46348

João Cunha 45412

Miguel Matos 45423

Orientador

Professor Doutor António Teófilo

Setembro, 2021

Resumo

O nosso trabalho baseia-se em criar um website onde vários utilizadores podem aprender a fazer o cubo mágico e disputar entre si em competições de tempo.

As principais motivações para a realização deste trabalho foram os vários websites para a interação com o cubo mágico e os vídeos de implementação do mesmo com o JavaScript. A partir dos websites nós adquirimos uma ideia básica do que o nosso website deve ou não possuir e qual deveria ser a interação com o cubo mágico. Por outro lado, os vídeos de implementação do cubo mágico com o JavaScript ajudaram-nos a ter uma percepção do funcionamento do cubo mágico e concederam-nos novas funcionalidades que poderíamos implementar.

As funcionalidades chave que destacam o nosso projeto de outros já existentes é a possibilidade de o utilizador reproduzir o seu cubo mágico físico no cubo mágico virtual através de reconhecimento por câmara e a competição de tempo para a resolução do cubo mágico. Estas implementações permitem um aprendizado mais prático para o utilizador e um motivo para continuar a fazer uso do website após esse aprendizado estar completo.

Concluindo, o nosso projeto foi inspirado por outros websites existentes e destacasse dos mesmos através das novas funcionalidades que melhoraram e aprimoraram o aprendizado do utilizador.

Agradecimentos

Gostaríamos de agradecer ao nosso Orientador António Teófilo pelo tempo, orientação e a disponibilidade que sempre apresentou ao longo deste projeto, apesar de estar a coordenar vários projetos sempre arranjou tempo para nos ajudar em qualquer problema que fosse aparecendo neste projeto.

Queremos também agradecer ao ISEL pela quantidade e diversidade de projetos que disponibilizou, permitindo assim trabalharmos num projeto que gostamos e que nos identificamos.

Índice

Resumo	i
Agradecimentos	iii
Índice	v
Lista de Tabelas	ix
Lista de Figuras	xi
1 Introdução	1
2 Trabalho relacionado e tecnologias utilizadas	3
2.1 Trabalhos relacionados	3
2.2 OpenCV.js	4
2.3 Firebase	4
2.4 Three.js	4
2.5 Blender	5
2.6 Conclusão	5
3 Modelo Proposto	7
3.1 Requisitos	7
3.2 Fundamentos	8
3.3 Casos de utilização	10
3.4 Wireframes	11
3.5 Arquitetura	11
3.6 Modelos	12
3.7 Conclusão	13

4 Implementação do Modelo	15
4.1 Planeamento	15
4.2 Firebase	16
4.2.1 Auth	16
4.2.2 Real time database	17
4.2.3 Storage	18
4.2.4 Registar	18
4.2.5 Login	18
4.2.6 Tipos de Utilizadores	18
4.3 Cubo magico	19
4.3.1 Modelo 2D	19
4.3.2 Modelo 3D	20
4.3.3 Movimentos	22
4.3.4 Fórmulas	23
4.3.5 Resolução automática	25
4.3.6 Utilização da resolução automática da identificar cubos com estados impossíveis	27
4.3.7 Desafio de tempo	27
4.4 Webcam	28
4.4.1 Diagrama de sequência	28
4.4.2 Acesso à câmara	29
4.4.3 Análise e transformações de cada frame do vídeo	29
4.4.4 Deteção do cubo	31
4.4.5 Deteção das cores	35
4.4.6 Apoio ao utilizador	37
4.5 Conclusão	38
5 Validação e Testes	39
5.1 Resolução do cubo automaticamente	39
5.2 Filtro de Anti-Aliasing	41
5.3 Ordenação dos contornos do cubo	42
5.4 Espaço de cores RGB e YUV	43
5.5 Inserir o cubo utilizando a camera	44
5.6 Inserir o cubo utilizando a ferramenta manual	44
6 Conclusões e Trabalho Futuro	45

<i>CONTENTS</i>	vii
A Wireframes	47
Bibliografia	51

Lista de Tabelas

5.1	Distâncias entre quadrados adjacentes horizontalmente	42
5.2	Distâncias entre quadrados adjacentes verticalmente	42

Lista de Figuras

3.1	Diagrama de casos de utilização	10
3.2	Arquitetura geral.	11
3.3	Diagrama de classe: index.html	12
3.4	Diagrama de classe: Manual2D.html	13
3.5	Diagrama de classe: camera.html	13
4.1	Cronograma	15
4.2	Autenticação	16
4.3	Base de dados em tempo real	17
4.4	Storage	18
4.5	Modelo do Cubo 2d	19
4.6	Evolução do modelo do Cubo 3d	20
4.7	Animacao do modelo do cubo 3d	21
4.8	Alguns dos movimentos implementados	22
4.9	Primeira fórmula para a resolução.	23
4.10	Segunda e terceira fórmula para a resolução.	23
4.11	Quarta fórmula para a resolução.	24
4.12	Quinta fórmula para a resolução.	24
4.13	Sexta fórmula para a resolução.	24
4.14	Sétima fórmula para a resolução.	24
4.15	Resolução da primeira, segunda camada e terceira camada do cubo mágico respetivamente.	25
4.16	Diagrama de sequência: Resolução do cubo.	26
4.17	Firebase: “realtime database”	27
4.18	Diagrama de sequência: Detecção do cubo com a câmara.	28
4.19	Aumento do brilho da imagem.	30
4.20	Resultado das operações morfológicas.	31

4.21	Distâncias entre os cubos para diferentes tamanhos	33
4.22	Contornos estimados	34
4.23	Apresentação das cores ao utilizador.	35
4.24	Espaço de cores YUV	36
4.25	Deteção das cores	37
4.26	Animação que indica a rotação a ser efetuada.	38
5.1	Resolução através de um botão	39
5.2	Resolução através dos passos recomendados	40
5.3	Resolução através das camadas recomendadas	40
5.4	Anti-Aliasing teste 1: modelo 3d antes e depois do filtro	41
5.5	Anti-Aliasing teste 2: modelo 3d antes e depois do filtro	41
5.6	Teste com o espaço de cores RGB	43
5.7	Teste com o espaço de cores YUV	43
5.8	Inserir o cubo dinamicamente	44
5.9	Inserir o cubo manualmente	44
A.1	Login	47
A.2	Registrar	48
A.3	Index	48
A.4	Cubo 2d inserido manualmente	49
A.5	Desafio	49
A.6	Selecionar cores	50
A.7	Selecionar faces	50

Capítulo 1

Introdução

Este trabalho baseia-se em criar um website onde é possível aprender a resolver o cubo mágico e aplicar este aprendizado para competir com outros utilizadores em concursos de tempo onde cada um tenta resolver o cubo mágico no menor tempo possível.

A principal motivação deste projeto é a de conferir uma melhor interação com o utilizador através de ferramentas e funcionalidades que facilitem o seu aprendizado e explorem o interesse do mesmo em continuar a utilizar o website. Para atingir este objetivo foi conferida a possibilidade de o utilizador implementar o seu cubo mágico físico no cubo mágico do website através do reconhecimento pela câmara ou pela configuração manual do utilizador. Também foi implementado um desafio de tempo onde cada utilizador tenta resolver o cubo mágico no menor tempo possível para competir com outros utilizadores. Estas funções distinguem este projeto de outros websites que também implementam o cubo mágico destacando-se dos seus competidores.

O desenvolvimento deste trabalho foi constituído em três partes que foram divididas pelos membros do grupo. A primeira parte encapsulava as regras e a lógica por de trás dos movimentos e da resolução do cubo mágico, que foram testadas inicialmente em um cubo 2D antes de ser implementada no projeto final. A segunda parte constituía o desenvolvimento do cubo 3D, configuração manual do cubo e o desafio de tempo, estas funcionalidades foram validadas consoante se o cubo apresentava o formato e as cores corretas, se a configuração manual implementava o que o utilizador pretendia e se o desafio de tempo guardava o tempo correto respetivamente. Por último, a terceira parte baseava-se em configurar o reconhecimento do cubo pela

câmara do utilizador, através da determinação das cores de cada face com o objetivo de reconstruir o cubo.

A partir deste documento será explicado os processos de planeamento e desenvolvimento que ocorreram durante a realização deste trabalho, a demonstração será explicada da seguinte forma.

- **Capítulo 2 - Trabalho relacionado e tecnologias utilizadas:**

Será apresentado os trabalhos que inspiraram a nossa página web e as tecnologias utilizadas para implementar as nossas funcionalidades.

- **Capítulo 3 - Modelo Proposto:**

Neste capítulo é explicado o modelo sobre qual o nosso trabalho foi criado e as funcionalidades mínimas que a nossa página web precisa possuir.

- **Capítulo 4 - Implementação do Modelo:**

Aqui será explicado toda a implementação e lógica por de trás de todas as funcionalidades do nosso projeto.

- **Capítulo 5 - Validação e Testes:**

Neste capítulo vai ser demonstrado todos os testes efetuados durante a realização deste trabalho.

Através deste relatório, vamos expor todos os conhecimentos adquiridos ao longo do nosso projeto.

Capítulo 2

Trabalho relacionado e tecnologias utilizadas

Neste capítulo será demonstrado os vários trabalhos que serviram de inspiração para este projeto e as tecnologias utilizadas para implementar o mesmo.

2.1 Trabalhos relacionados

Durante a realização deste projeto, foram pesquisados na Internet outros trabalhos semelhantes ao nosso para servirem de exemplo sobre o que fazer e páginas que explicam a lógica sobre os vários tipos de movimentos e possíveis resoluções do cubo mágico.

O aspetto do nosso website foi inspirado nos vários exemplos que existem na Internet, de forma a retirar as melhores qualidades de cada um deles e juntar no nosso projeto. A existência simultânea do cubo 3D e o cubo 2D foi inspirado por,e.g., [Beke,] onde é possível trocar a perspetiva do cubo mágico de 3D para 2D. Por outro lado, o menu de opções que aparece de baixo do cubo mágico foi feito com base no, e.g., [Smith, 2014] que dispõem várias opções para interagir com o cubo localizadas abaixo do mesmo.

Algumas das funcionalidades do nosso trabalho, da mesma forma que o aspetto do website, foram inspiradas nas funcionalidades de sítios na Internet que implementavam ou explicavam as mesmas. A resolução automática e movimentos do cubo mágico foram implementados com base no, e.g., [Xu, 2015] e no, e.g., [Ruwix,], estes websites implementam uma formula para resolver

o cubo mágico e a nomenclatura de cada movimento do mesmo respetivamente. O desafio de tempo foi inspirado no, e.g., [Riera,] que disponibiliza a opção do utilizador competir consigo mesmo em vez de outros utilizadores como foi implementado neste trabalho.

2.2 OpenCV.js

O OpenCV.js é um ficheiro de JavaScript que possui um subconjunto de funções do OpenCV para as plataformas web, e.g., [Opencv.js, 2017]. Este ficheiro faz uso do Emscripten, um copilador LLVM para Javascript, para copilar as suas funções em asm.js ou WebAssembly, que permitem a otimização e eficiência no tempo de compilação das funções do OpenCV.js. Com o OpenCV.js é possível conceder às aplicações web uma variedade de ferramentas multimédia para a análise de imagens disponíveis no OpenCV.

2.3 Firebase

O Firebase é uma plataforma de desenvolvimento móvel e web, com foco em ser um back-end completo e de fácil usabilidade. São disponibilizados diversos serviços diferentes que ajuda no desenvolvimento e gestão de aplicações. O Firebase distingue se de muitos competidores por ser grátis, fácil de utilizar e ser baseado na tecnologia de NoSQL.

Foram utilizados os seguintes serviços do Firebase:

- Autenticação;
- Base de dados;
- Sistema de ficheiros.

2.4 Three.js

O Three.js é uma biblioteca JavaScript que permite criar e demonstrar gráficos 3D em um navegador web sem o esforço de os construir na maneira tradicional e sem a necessidade da utilização de qualquer plugin. Esta biblioteca permite criar cenas onde os objetos 3D são renderizados e colocados em uma

página web, os objetos podem ser criados ou importados de outros ficheiros de modelos 3D, por exemplo neste trabalho foi importado um objeto construído no Blender para criar o cubo mágico. A partir desta biblioteca é possível construir objetos 3D como o cubo mágico de maneira simples, eficaz e sem muitas complicações.

2.5 Blender

O Blender é um programa grátsis de código aberto usado para criar conteúdos 3D como modelação, animação, simulação, renderização e captura de movimentos. A sua grande extensão de ferramentas de modulação de objetos 3D permite uma maior facilidade na construção do cubo mágico do que se o mesmo fosse feito no Javascript. A partir da biblioteca Three.js (cf., secção 2.4) é possível importar o modelo criado no Blender para a página web e manipulá-lo através do Javascript para que possa ser responsivo às ações do utilizador.

2.6 Conclusão

Este capítulo descreveu os trabalhos que inspiraram este projeto e as tecnologias utilizadas para realizar o mesmo. Os trabalhos relacionados são na maior parte páginas web que também implementam o cubo mágico e as tecnologias adicionam mais funcionalidades que são pontos-chave para a maioria das ferramentas do nosso trabalho.

Capítulo 3

Modelo Proposto

Neste capítulo vai ser demonstrado o modelo sobre o qual este trabalho foi construído, os casos de utilização, os fundamentos teóricos e tecnológicos e os principais contributos para o desenvolvimento deste projeto.

3.1 Requisitos

Os requisitos essenciais para este trabalho incluem a interação com o cubo mágico, a configuração manual e através da câmara do cubo mágico, resolução automática e o desafio de tempo.

A interação com o cubo mágico deve ser simples e fácil de aprender a utilizar. Esta função é a mais importante do projeto, pois permite ao utilizador interagir com o cubo mágico sem a necessidade de possuir um ele mesmo. A maioria dos utilizadores deste site farão uso dele para aprender a resolver o cubo mágico, por consequente esta função melhorará a experiência dos utilizadores uma vez que a mesma permite que eles experimentem as suas ideias e vejam os resultados das mesmas quando interagem com o cubo mágico.

Para o utilizador, as ferramentas de inserir o cubo mágico manualmente e através da câmara, devem ser fáceis de aprender a usar e adaptáveis para qualquer ambiente que o utilizador se encontrar. Estas funções permitem ao utilizador adaptar o cubo mágico para qualquer configuração que o mesmo desejar, desde que, a configuração exista. Caso um utilizador procurasse reconstruir o cubo mágico para uma configuração específica de forma que pudesse receber conselhos ou determinar a resolução do cubo, ele fará uso destas funções que melhoram a interação entre o utilizador e o website e

fortalecem a aprendizagem do mesmo.

A resolução automática deve funcionar para todas as configurações possíveis do cubo e estar no completo controlo do utilizador para que o mesmo possa acompanhar a resolução. Esta função ajudará o utilizador a perceber a lógica por de trás da resolução do cubo mágico de uma forma simples e eficaz. Um utilizador que esteja interessado em utilizar o nosso website procurará aprender a resolver o cubo mágico, caso o mesmo não possua conhecimento para o fazer, e fará uso da resolução automática para aprender passo por passo a solução para resolver o cubo.

O desafio de tempo deve guardar um vídeo do utilizador a resolver o cubo e guardá-lo no servidor (base de dados) para que possa ser posteriormente analisado por um administrador. Um utilizador experiente na resolução do cubo mágico não estará interessado em aprender a resolver o cubo mágico, por consequente, este módulo foi criado para entreter este tipo de utilizadores. O nosso website fornece um sistema de classificação onde é possível verificar os utilizadores que conseguiram completar o cubo mágico no menor tempo possível, qualquer utilizador experiente que esteja interessado em entrar para os melhores do website e competir com outros utilizadores vai procurar utilizar esta função para demonstrar as suas habilidades.

Concluindo, estes são os requisitos mínimos que o nosso projeto deve implementar para estar considerado completo.

3.2 Fundamentos

Para implementar os requisitos mencionados na secção 3.1 foi necessário estudar algumas tecnologias que fossem capaz de executar o que era pretendido.

Na interação com o cubo mágico e a resolução automática foi necessário estudar várias fórmulas de resolução para o cubo mágico de forma que fosse possível implementá-las para qualquer que fosse a configuração do cubo. Por outro lado, existiu a necessidade de criar uma estrutura onde todas as informações do cubo, face, posição e cor, pudessesem guardadas e apresentadas em um ambiente 2D e 3D.

A configuração manual e deteção por câmara do cubo mágico devem acecer a estrutura de dados referida anteriormente de forma a implementar a configuração do utilizador ou o cubo detetado pela câmara. A configuração

manual vai alterar o cubo quadrado por quadrado até o mesmo ficar com a composição desejada pelo utilizador. Por outro lado, a deteção por câmara deve reconhecer o cubo do utilizador através de várias transformações de imagens que tornam a deteção adaptável para qualquer ambiente que o utilizador se encontrar.

O desafio de tempo guardará o vídeo da resolução do cubo e o tempo que o utilizador demorou para completar a tarefa numa base de dados para ser posteriormente analisado pelos administradores do website. Os utilizadores apenas entraram para a classificação dos melhores do website se obterem a aprovação dos administradores, por consequente é necessário guardar o vídeo e o tempo de resolução em algum lugar para ser posteriormente avaliado, que para este projeto foi escolhido um servidor.

A partir destas aplicações teóricas e tecnológicas foi possível implementar cada um dos requisitos necessários para este trabalho.

3.3 Casos de utilização

O sistema pode ser dividido em vários tipos de utilizadores.

- **Utilizadores inexperientes:**

Não sabem resolver o cubo magico e querem resolvê-lo e eventualmente aprender a resolver.

- **Utilizadores medianos:**

Sabem resolver o cubo, mas podem precisar de ajuda. Estes utilizadores também podem entrar no desafio de tempo, onde o utilizador tem de solucionar o cubo magico o mais rapidamente possível.

- **Utilizadores avançados:**

Sabem resolver o cubo bastante rapidamente, e apenas estão interessados na ferramenta do desafio de tempo.

- **Administrador:**

Gerem os desafios de tempo, validando ou apagando as submissões dos outros utilizadores.

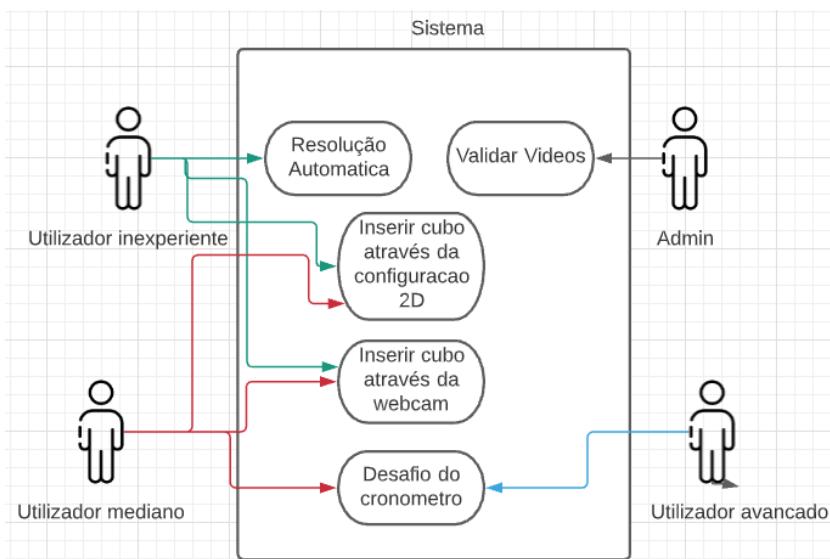


Figura 3.1: Diagrama de casos de utilização

3.4 Wireframes

Com o objetivo de construir uma ideia geral do website que o grupo quer projetar, foram utilizados wireframes das diversas páginas que iriam existir no website.

Os wireframes tem a vantagem de serem relativamente rápidos de fazer e ajudar os projetistas a ter feedback externo. Para o efeito, o grupo decidiu optar por uma base bastante simples e amiga do utilizador, permitindo começar a melhorar o design da aplicação sem muito esforço, assim como servir de base para obter uma melhor compreensão visual do que se pretende implementar. Em apêndice podem ser encontrados todos os wireframes.

3.5 Arquitetura

A arquitetura do nosso projeto separa as componentes de JavaScript, de HTML, de imagens e de modelos para o cubo mágico.

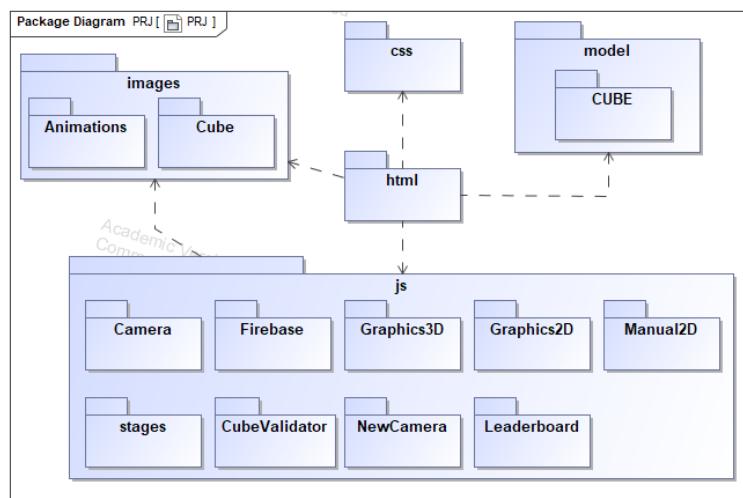


Figura 3.2: Arquitetura geral.

As componentes de Javascript estão separadas entre si consoante as funcionalidades que as mesmas proporcionam. Cada uma destas pastas possuem um ficheiro Javascript que fornece às outras componentes métodos para interagirem com as funcionalidades que as mesmas proporcionam, desta maneira cada componente é reutilizável e não é dependente das outras componentes.

As componentes HTML representam cada página existente no nosso trabalho. Estes ficheiros fazem uso das componentes de Javascript para responderem às interações do utilizador com a página web.

As componentes de imagens e de modelos para o cubo mágico contêm respetivamente as imagens e animações para melhorar a experiência do utilizador e o ficheiro do modelo 3D do cubo mágico feito no Blender. Estas componentes são utilizadas para construir algumas funcionalidades da nossa página web e melhorar o aspetto da mesma.

3.6 Modelos

Os modelos servem para definir como as várias componentes do nosso trabalho relacionam-se entre si.

Todas as funcionalidades do nosso projeto fazem uso do CSS e do Firebase, devido ao facto de que cada página da nossa aplicação web estar ligado ao servidor do Firebase e fazer uso do CSS para melhorar o aspetto de cada página.

Como se pode verificar nas figuras 3.3, 3.4 e 3.5, todas as páginas web possuem um ficheiro JavaScript que faz uso das outras entidades para executar a funcionalidade da página que está a ser utilizada. A partir desta análise, é possível identificar a estrutura básica de todas as funcionalidades deste trabalho e as classes mais importantes que estão presentes em todos os diagramas (matrixCube, stage1, stage2 e stage3).

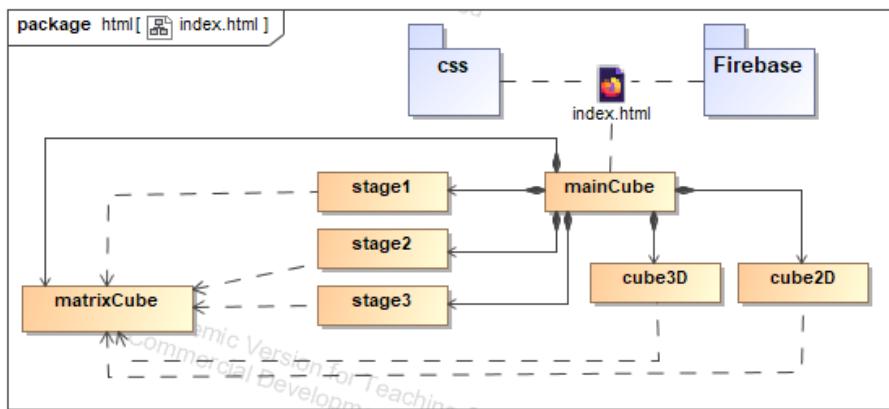


Figura 3.3: Diagrama de classe: index.html

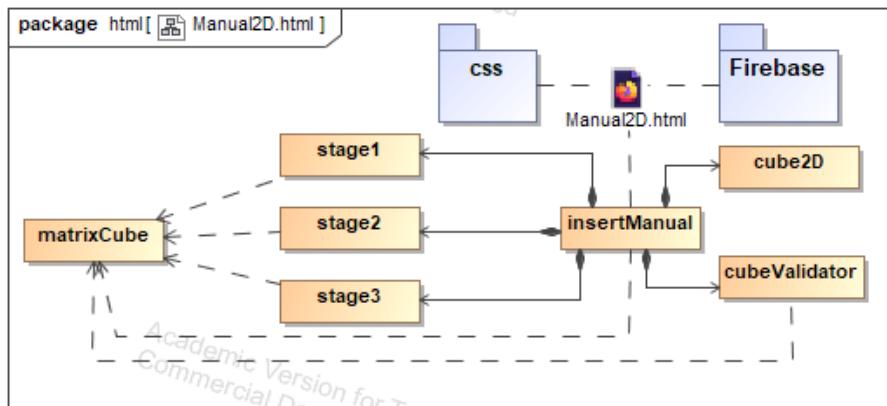


Figura 3.4: Diagrama de classe: Manual2D.html

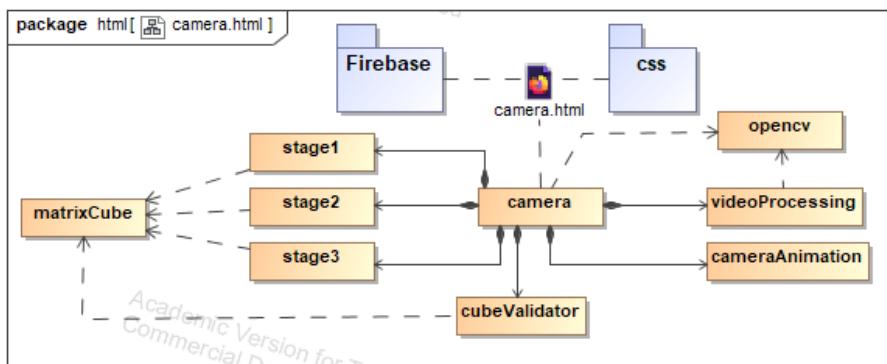


Figura 3.5: Diagrama de classe: camera.html

3.7 Conclusão

No presente capítulo, foram determinadas as funcionalidades que precisam ser implementadas, as estruturas das páginas, a arquitetura geral do projeto e os modelos das ferramentas disponíveis ao utilizador.

Esta análise permitiu-nos ter uma ideia da estrutura do projeto como um todo e o que o mesmo promete fornecer ao utilizador.

Capítulo 4

Implementação do Modelo

4.1 Planeamento

Foi utilizador um cronograma temporal para ajudar na previsão das tarefas a serem realizadas, desta forma, o grupo conseguiu facilmente construir um plano suscetível a adiantamentos ou atrasos.

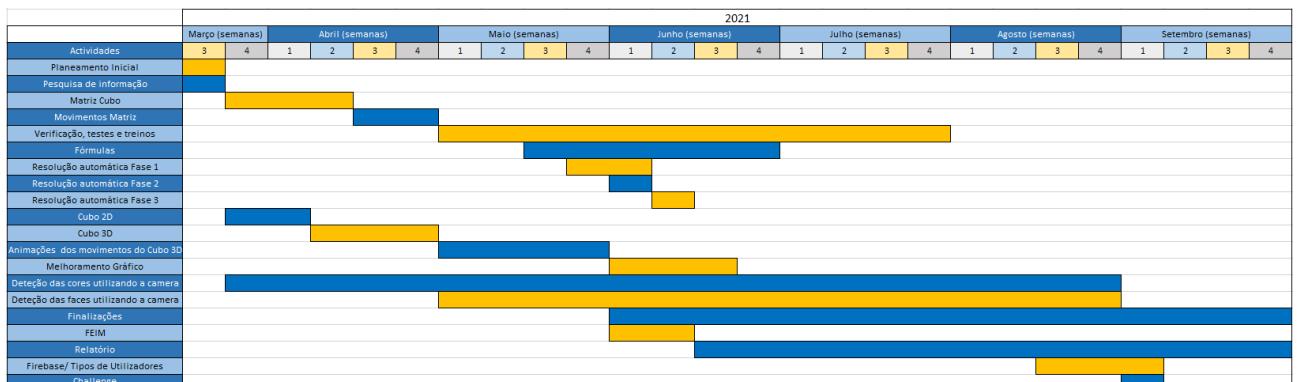


Figura 4.1: Cronograma

O cronograma ajudou bastante o grupo a concluir as tarefas ao longo da execução do projeto, isto porque, algumas tarefas mais trabalhosas tiveram de ser finalizadas por mais do que um elemento do grupo para cumprir o plano delimitado, com isto, todas as tarefas foram realizadas dentro dos objetivos definidos pelo grupo.

Embora algumas datas tiveram de ser ajustadas durante o projeto, no geral, o grupo considerou o cronograma uma mais-valia, definindo objetivos claros, é perfeito para um grupo de três alunos.

4.2 Firebase

4.2.1 Auth

Os diversos utilizadores que usam o website são distinguidos através de uma ferramenta do Firebase, esta ferramenta permite realizar a autentificação dos utilizadores de forma simples.

A abstração que o Firebase oferece permite uma interação simples e compacta entre o mesmo e o site, tendo a segurança infalível dos serviços da google.

Identificador	Provedores	Data de criação	↓	Último login	UID do usuário
guest@guest.pt	✉	14 de set. d...		14 de set. d...	ifgYENAvYZqmg9uTvCAIHxAZBR2
admin@admin.pt	✉	14 de set. d...		14 de set. d...	maFHp1f4U5aVTupQHDCcl74oxt43
deime.daniel.silva.rosa@g...	✉	26 de ago. d...		26 de ago. d...	nyRbYFP17LNI5c9kZX3twYABeps1
joao@isel.pt	✉	15 de ago. d...		5 de set. de ...	PQTXtAGRzRdC5iN6yI9kzdOPjT2
miguel_jessi@hotmail.com	✉	14 de ago. d...		14 de ago. d...	vUYbCiFbhV1cuPYhOlBaaLT8zx2

Figura 4.2: Autenticação

Quando um utilizador realiza o registo o seu email e password ficam ligadas a sua conta, sendo depois criado um identificador único para cada utilizador o ”UID”, este identificador é utilizado para distinguir os vários utilizadores.

4.2.2 Real time database

Real time database é uma ferramenta utilizada para guardar toda a informação relevante como os dados utilizadores ou as submissões de desafios.



Figura 4.3: Base de dados em tempo real

Do utilizador é guardado o seu “UID”, a permissão (admin ou user), o username, também são guardadas as cores caso o utilizador tenha efetuado a leitura do cubo rubik através da camara. As cores guardadas possuem 3 valores correspondentes ao seu “R”, “G”, “B”.

As submissões de desafios quando são submetidas são armazenadas na tabela “notValidated”, visto que um admin tem de verificar se o vídeo é válido, os dados armazenados são: “UID”, “time”, “username”, “videoRef”. Após ser validado os dados são transferidos para a tabela “Validated”.

4.2.3 Storage

Storage é mais uma das ferramentas disponibilizadas pelo Firebase, esta permite guardar ficheiros numa cloud assim como fotos ou vídeos.

<input type="checkbox"/>	 PQTXtAGRzRdC5IN6yl9kzdQPjT21629032030945	306.7 KB	video/webm	15 de ago. de 2021
<input type="checkbox"/>	 IAm0EM09InRi0qAwu5ynCpdYL5d21628938236389	21.19 KB	video/webm	14 de ago. de 2021
<input type="checkbox"/>	 IAm0EM09InRi0qAwu5ynCpdYL5d21628938331769	17.67 KB	video/webm	14 de ago. de 2021
<input type="checkbox"/>	 IAm0EM09InRi0qAwu5ynCpdYL5d21628940937491	88.4 KB	video/webm	14 de ago. de 2021
<input type="checkbox"/>	 IAm0EM09InRi0qAwu5ynCpdYL5d21628950878415	97.26 KB	video/webm	14 de ago. de 2021
<input type="checkbox"/>	 IAm0EM09InRi0qAwu5ynCpdYL5d21628950919709	94.3 KB	video/webm	14 de ago. de 2021

Figura 4.4: Storage

Neste projeto o Storage é utilizado para armazenar os vídeos que os utilizadores fazem quando realizam os desafios, de forma a identificar simplesmente os vídeos o seu nome foi composto do “UID” do utilizador que realizou o desafio juntamente com a data atual convertida em milissegundos.

Para ser possível aceder aos vídeos é guardado o nome no vídeo como uma referencia na tabela “NotValidated” ou “Validated”.

4.2.4 Registrar

Quando um utilizador faz o seu registo, é lhe pedido o username, o email e a password. No Firebase, através da ferramenta de autenticação, é criado um “UID” único para cada utilizador. Este “UID” é então utilizado como chave primaria para identificar os utilizadores.

4.2.5 Login

A ferramenta de autenticação do Firebase permite o acesso a informação da “realtime database” em caso do login ser efetuado com sucesso.

4.2.6 Tipos de Utilizadores

Os utilizadores podem ser simplificados em dois grupos: “administradores” e “convidados”. Os administradores têm acesso a mais ferramentas, uma delas validar os desafios submetidos por outros utilizadores.

4.3 Cubo magico

4.3.1 Modelo 2D

A representação 2D foi implementada com o intuito que o utilizador consiga visualizar o cubo na sua totalidade, assim como permitir ao utilizador a interação com a representação 3d do cubo através das setas que existem na mesma, cada seta corresponde a um movimento.

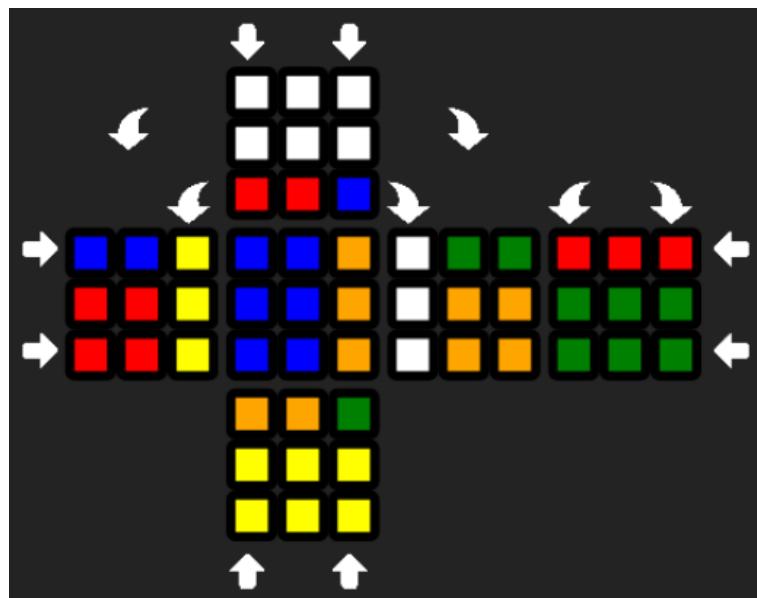


Figura 4.5: Modelo do Cubo 2d

Como a aplicação se trata de um site em HTML foi utilizada a ferramenta “canvas”, o canvas é um elemento do HTML que é utilizado para desenhar gráficos, os gráficos tem de ser desenhados através do código implementado em JavaScript.

Utilizando JavaScript foi desenhado os cubos, faces e as setas que são visíveis para o utilizador, assim como criado eventListener para saber onde e quando um utilizador esta a clicar no canvas.

4.3.2 Modelo 3D

Uma das partes cruciais era a visualização de um cubo rubik, desta forma o utilizador obtém uma ajuda mais visual quando estiver a tentar resolver o cubo.

Para ser possível a visualização do cubo primariamente foi necessário criar um modelo do cubo, utilizando a ferramenta Blender foi criado um dos 27 cubos que juntos constituem o cubo principal, criado esse cubo foi apenas necessário criar as 6 faces que o mesmo pode ter.

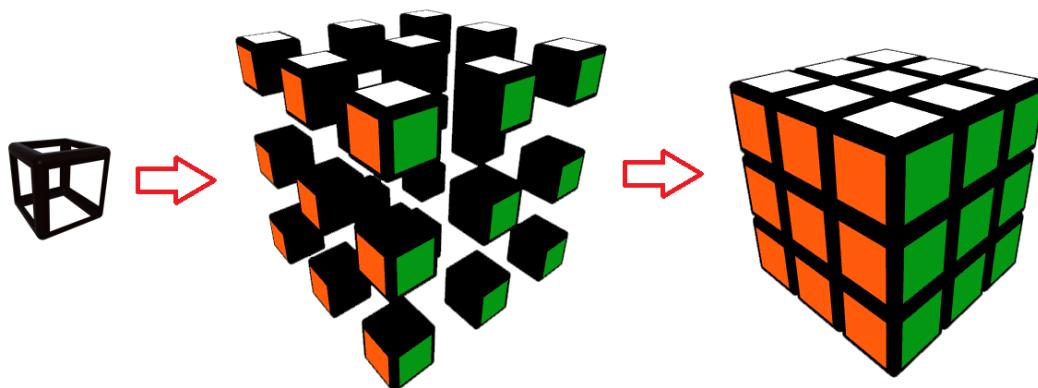


Figura 4.6: Evolução do modelo do Cubo 3d

Para apresentar o modelo criado em Blender no browser utilizou-se uma biblioteca de JavaScript o “Three.js”, esta biblioteca permite importar objetos, como o criado anteriormente no Blender, para o browser. Para se conseguir visualizar os objetos importados é necessário possuir:

- **Cena:**

Dentro da cena é onde todos os objetos importados podem ser renderizados, uma cena permite ter objetos, luzes e câmaras.

- **Render:**

O render permite definir as opções de renderização dos objetos, neste projeto foi utilizado a opção de anti-aliasing com o intuito de suavizar o cubo.

- **Camera:**

A câmara é a forma do browser “copiar” o olho humano e mostrar essa imagem no ecrã.

- **Luz:**

Todos os objetos na cena precisam de uma luz para puderem ser visíveis, o grupo escolheu uma luz ambiente porque não é muito forte mas não se perde qualidade na visualização do cubo.

Animações

Inicialmente a representação 3d do cubo rubik era estática, ou seja, as cores eram substituídas instantaneamente pela cor seguinte.

Esta forma apesar de eficaz era pouco visual e tornava a compreensão do que estava a acontecer com o cubo complicada para o utilizador, então foram adicionadas animações de forma a dar um pouco mais de vida ao cubo.

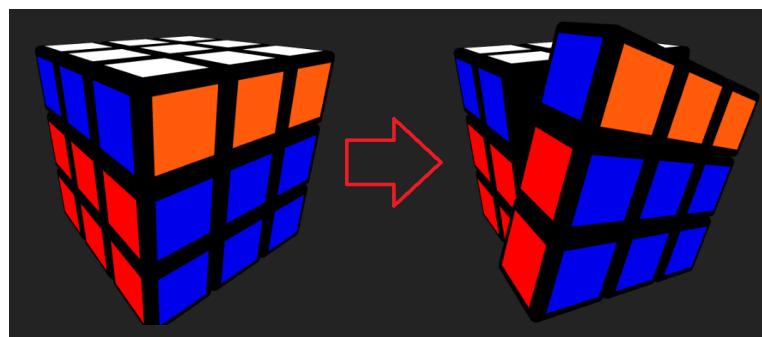


Figura 4.7: Animacao do modelo do cubo 3d

As animações foram realizadas utilizando o Three.js uma biblioteca do JavaScript, todas as rotações foram realizadas da seguinte forma, consoante a rotação desejada descobria-se as peças que precisavam de ser rodadas, isto era feito com hard code porque uma rotação afetava sempre as mesmas peças. Após isso rodava-se as peças em “X”, “Y” ou “Z” consoante a rotação que era, esta rotação era realizada utilizando o método **rotateOnWorldAxis**.

4.3.3 Movimentos

A interação mais simples entre o utilizador e o cubo mágico é o movimento, que representam todas as formas possíveis que o utilizador pode rodar o cubo. Primeiramente foram definidos 12 movimentos diferentes na realização do projeto.

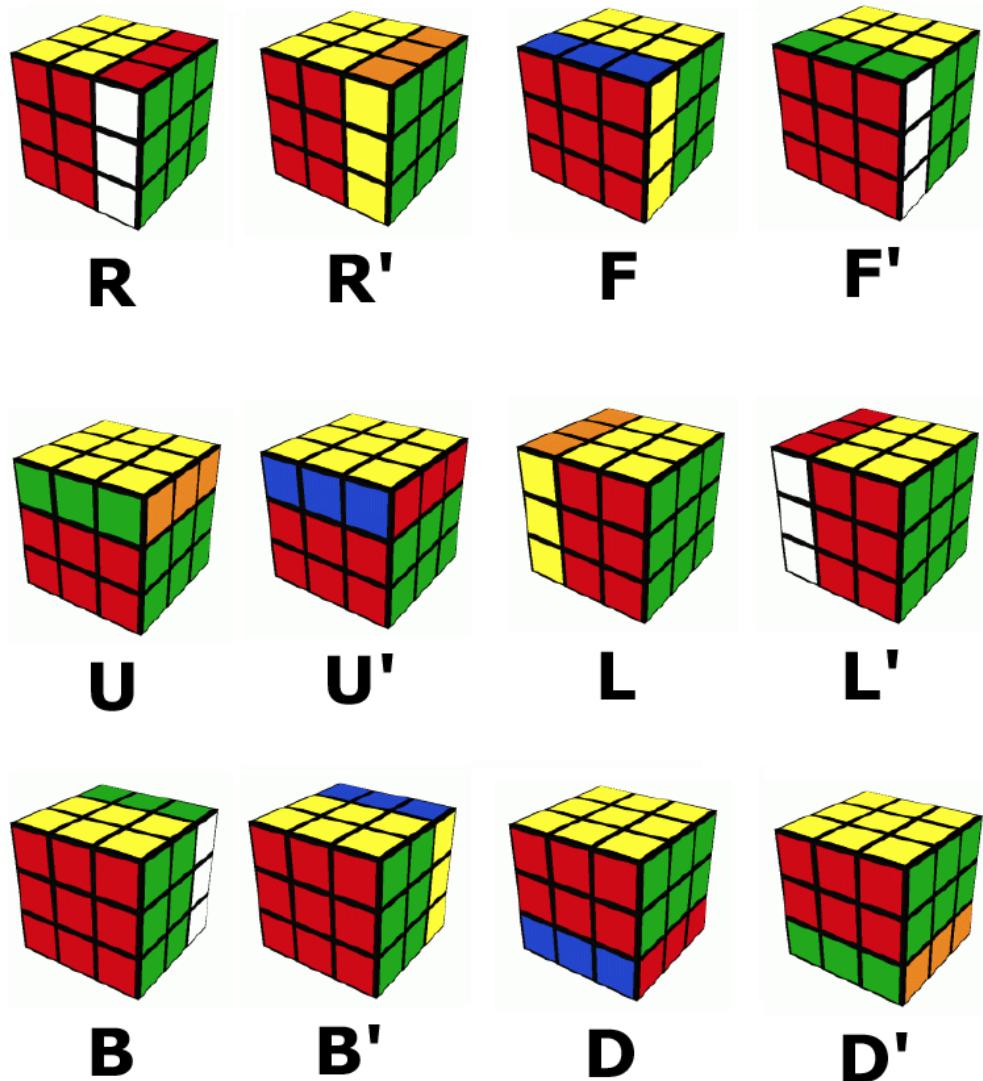


Figura 4.8: Alguns dos movimentos implementados

4.3.4 Fórmulas

Um conjunto de movimentos é chamado de fórmula.

Foram utilizadas 7 formulas na resolução automática.

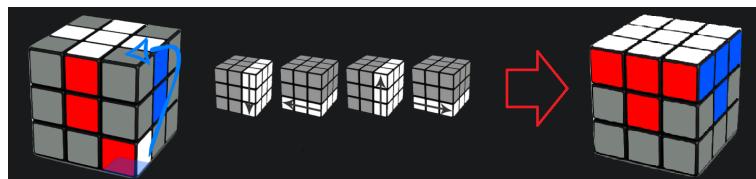


Figura 4.9: Primeira fórmula para a resolução.

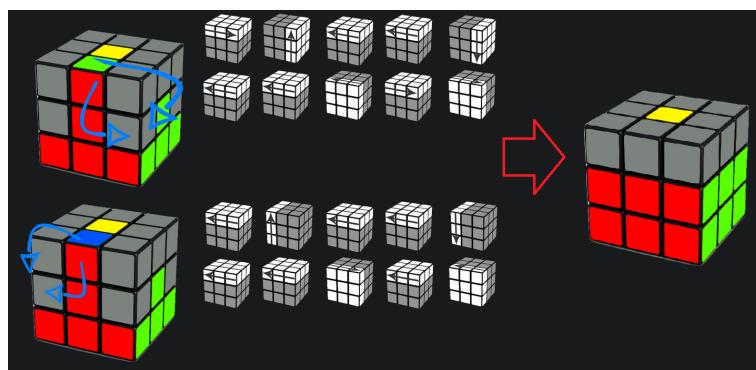


Figura 4.10: Segunda e terceira fórmula para a resolução.

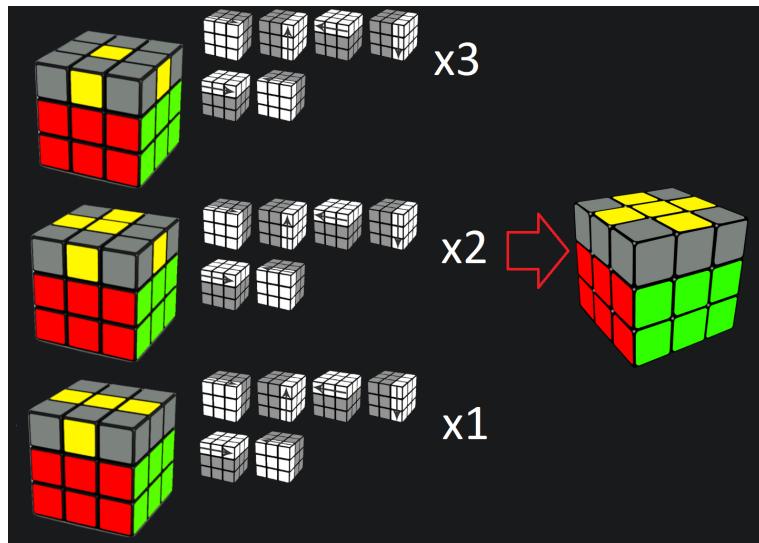


Figura 4.11: Quarta fórmula para a resolução.

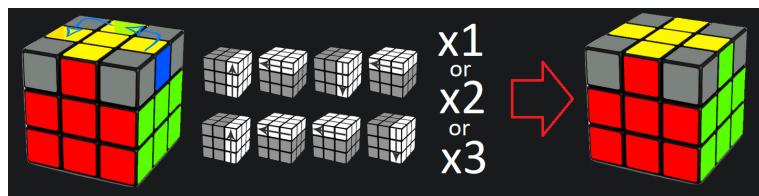


Figura 4.12: Quinta fórmula para a resolução.

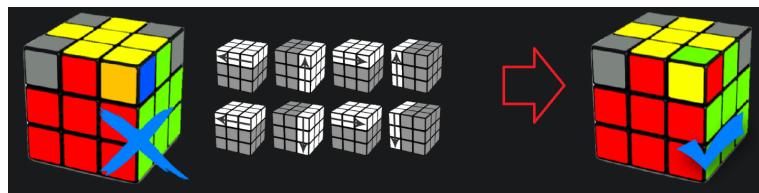


Figura 4.13: Sexta fórmula para a resolução.

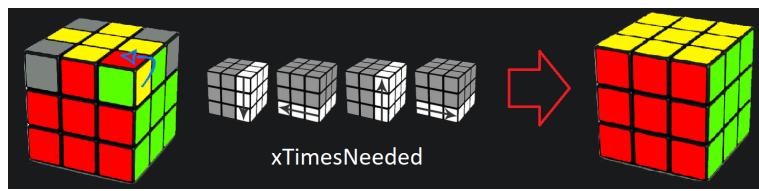


Figura 4.14: Sétima fórmula para a resolução.

4.3.5 Resolução automática

As várias peças do cubo mágico são armazenadas num array com várias dimensões, sabendo que o cubo mágico tem 6 faces e cada face tem 9 quadrados, este array guarda a informação de 54 quadrados. Foram feitas várias funções que simulam os movimentos do cubo mágico, estas funções trocam as várias posições dos quadrados do array de acordo com a alteração nas faces.

Como o cubo mágico tem 6 faces, foram feitas 12 funções para rodar as faces individualmente para trás e para a frente. Também foram desenvolvidas mais 6 funções para rodar as faces do cubo dando a possibilidade ao utilizador de selecionar a face principal e posicionar o cubo de todas as formas possíveis.

Lógica

Para resolver o cubo mágico, este passa por um processo composto por 7 passos, sendo que cada passo recorre a um procedimento lógico, sendo que este pode utilizar várias fórmulas e estas podem ser aplicadas mais do que uma vez. Este procedimento foi feito de forma não bloqueante e sequencial.

De forma geral, quando a resolução automática quer resolver uma dada parte, o programa verifica se um conjunto de peças estão de acordo com as respetivas fórmulas, e em caso positivo aplica a fórmula, em caso negativo reage de uma certa maneira pré-definida, um movimento específico, para entrar num estado que seja possível aplicar a respetiva fórmula.

A primeira fórmula serve para resolver o último passo da primeira camada, a segunda camada é completada com duas fórmulas e para a camada final são necessárias quatro fórmulas.

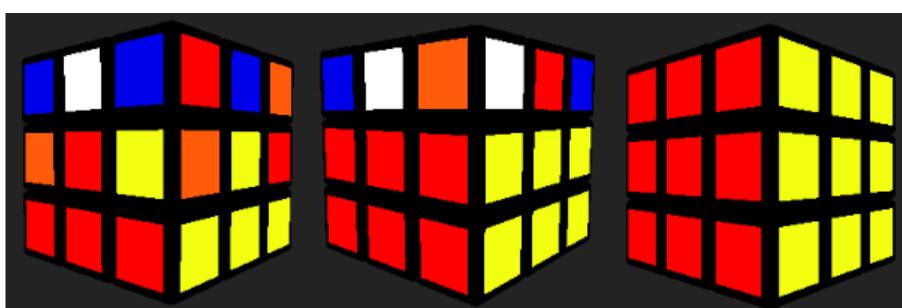


Figura 4.15: Resolução da primeira, segunda camada e terceira camada do cubo mágico respetivamente.

Diagrama de sequêncial

O estado do cubo (posição das peças) esta guardado num objeto matrixCube da classe mainCube. Sendo assim, para prever uma resolução, é necessário enviar uma copia do estado do cubo como argumento para uma das classes “stages” para este estado ser analisado a parte e ser calculado a sua resolução. Como foram implementadas três classes diferentes para resolver as três camadas individualmente, é preciso enviar como argumento o estado.

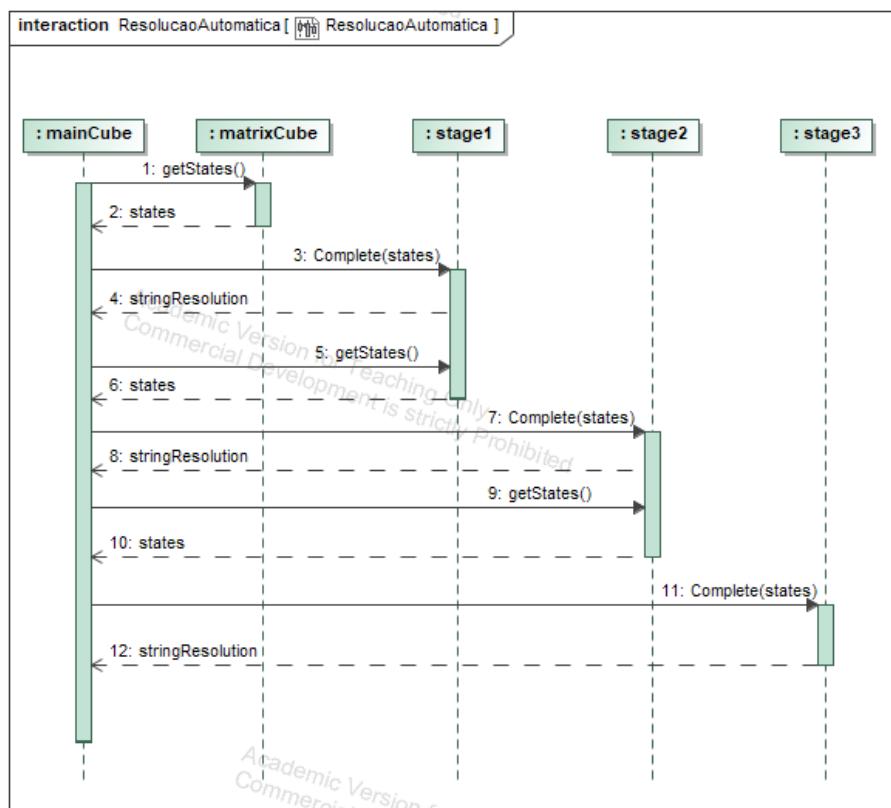


Figura 4.16: Diagrama de sequência: Resolução do cubo.

Se um cubo baralhado for resolvido, primeiro é enviada uma cópia do seu estado para um método da primeira camada responsável pela sua resolução. Esse método retorna a resolução apenas da primeira camada, e para obter a resolução completa é também preciso resolver as outras camadas. Para esse efeito é preciso ir buscar o estado final do cubo resolvido depois da primeira camada para continuar a sua resolução sequencial...

4.3.6 Utilização da resolução automática da identificar cubos com estados impossíveis

A resolução automática também é utilizada para validar um cubo magico inserido pelo utilizador, sendo este inserido pelo modo manual ou pela câmara. O sistema apenas deixa o cubo ser inserido depois de encontrar solução com a resolução automática, pois se o cubo não tiver solução, esse cubo não existe (as peças não são validadas).

4.3.7 Desafio de tempo

Quando um utilizador submete um vídeo da sua resolução do cubo mágico, foi utilizada a ferramenta do Firebase “Storage” que nos permitiu armazenar de forma gratuita todos os vídeos. A chave identificadora dos vídeos é guardada na “realtime database”.

As classificações dos desafios podem ser armazenadas na ”realtime database” de 2 tipos diferentes: “validadas” ou “não validadas”. Sempre que um desafio de tempo é submetido por um utilizador, este entra na tabela das “não validadas”, e após isso, um administrador tem a possibilidade de verificar o vídeo tendo as opções de eliminar ou passa-lo para a categoria de “validadas”.

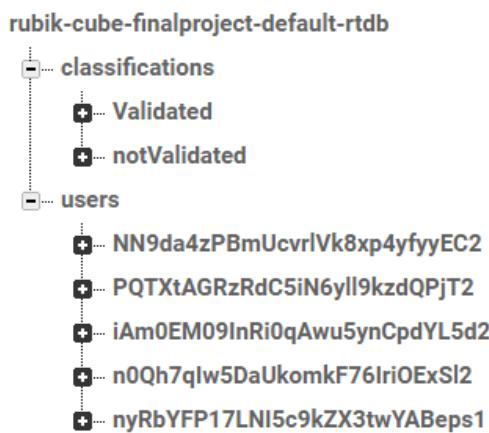


Figura 4.17: Firebase: “realtime database”

Quando o desafio é “validado”, ele é disponibilizado em uma grelha mundial que todos os utilizadores têm acesso.

4.4 Webcam

4.4.1 Diagrama de sequência

Este diagrama de sequência demonstra o processo de análise de frames, estimativa da face do cubo mágico e detecção das cores do mesmo.

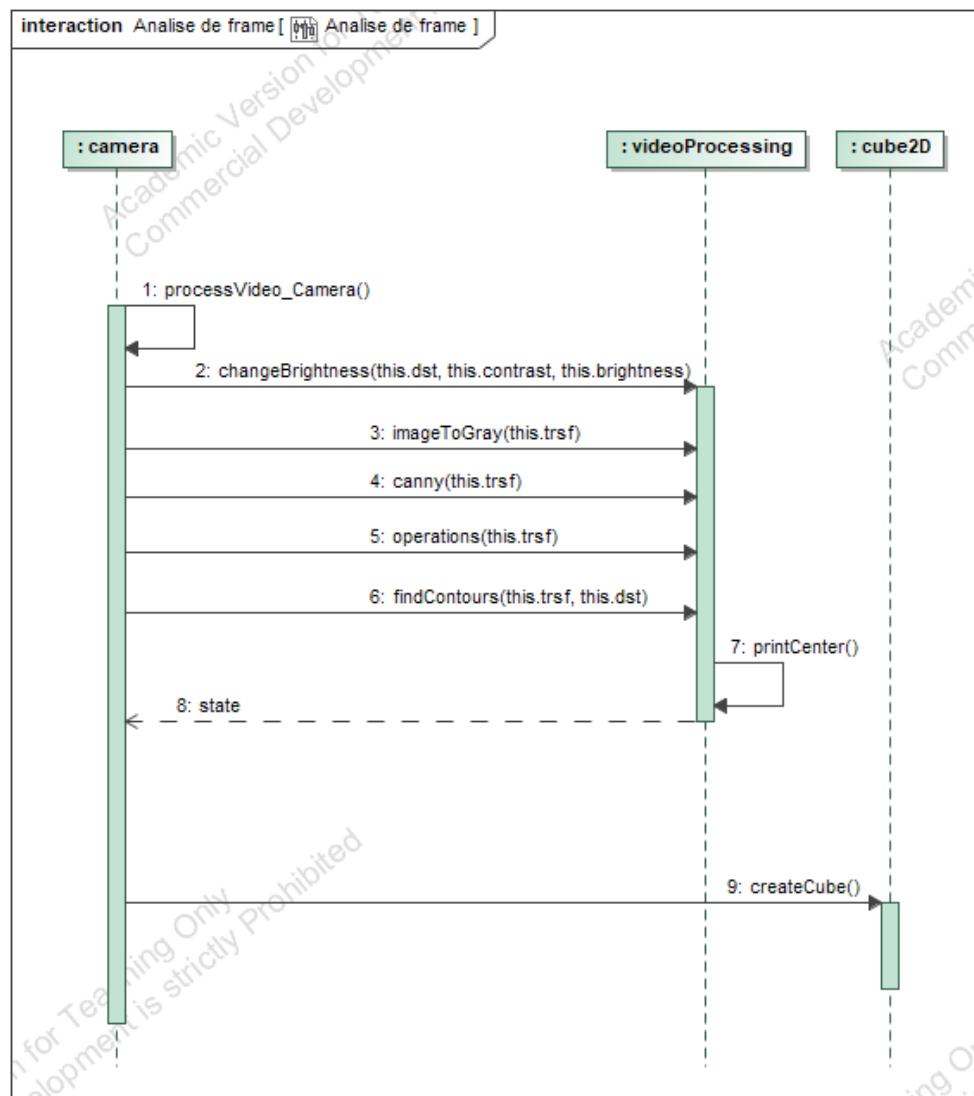


Figura 4.18: Diagrama de sequência: Detecção do cubo com a câmera.

Os ficheiros responsáveis por esta função da página web são a camera.js, videoProcessing.js e cube2D.js. A camera.js influência os elementos HTML da página web e fornece as funções para o processamento e análise de cada frame. O videoProcessing.js é a classe que possui os métodos de transformação de frames e estimativa de cores, a camera.js opera ao chamar várias funções desta classe para analisar cada frame e mostrá-la ao utilizador. A classe cube2D serve o único propósito de demonstrar ao utilizador as faces que o mesmo já registou através da câmara.

A explicação sobre cada passo do processo de análise será apresentada nas próximas secções.

4.4.2 Acesso à câmara

Para obter acesso à câmara, é necessário que o utilizador forneça a sua permissão para o programa obter os dados recebidos por ela. Com estes dados, é utilizado a função cv.VideoCapture para transformar as frames recebidas pela câmara em um objeto Mat que é o objeto base usado em todas as funções do OpenCV.js para a transformações de imagem, este objeto guarda os valores dos pixéis da frame em vários formatos de dados para serem posteriormente analisados por diferentes funções.

4.4.3 Análise e transformações de cada frame do vídeo

Após os valores das frames recebidas pela câmara do computador serem obtidas, é necessário analisar e aplicar algumas transformações nas mesmas para que possam ser utilizadas nas páginas da aplicação web e para a deteção do cubo e das cores.

A primeira transformação é o aumento do brilho e do contraste entre as cores da face do cubo. Para facilitar a deteção de cada cor na face de um cubo mágico foi implementado a função “convertScaleAbs” que permite manipular o brilho e o contraste das cores na frame através dos valores β e α . O valor β é somado ou subtraído, dependendo do valor do mesmo, a cada pixel da frame de forma a aumentar ou subtrair os seus valores, caso o valor do pixel fique fora da zona de valores entre 0 e 255 ele será saturado e o seu valor passará a ser 255. O valor α é utilizado para definir o nível de contraste entre as várias cores da frame, um valor inferior a 1 fará como as

cores possuam menos contraste, o que não é desejável, e um valor superior a 1 aumentará o contraste das cores. Esta função foi utilizada para aumentar consideravelmente o contraste entre as cores do cubo para facilitar a deteção dos contornos das mesmas para ambientes com uma iluminação variada.

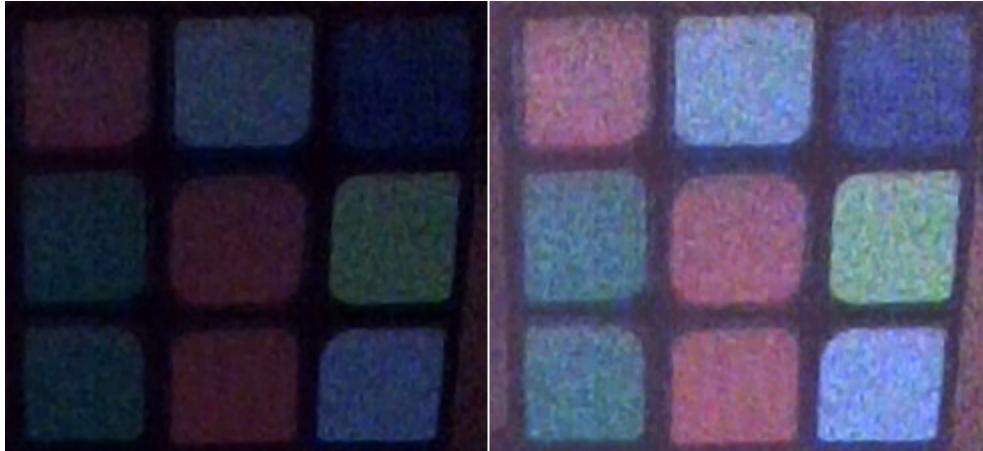


Figura 4.19: Aumento do brilho da imagem.

Após o aumento do contraste estar efetuado, a frame é transformada para níveis de cinzento para que se possa aplicar o algoritmo de canny para a deteção de contornos. O algoritmo de canny suaviza a frame que analisa e a passa por um filtro de forma a determinar a intensidade dos gradientes dos pixéis da mesma, desta maneira este algoritmo localiza as bordas da face do cubo mágico eficientemente.

Por fim, antes que os contornos sejam analisados são efetuadas transformações morfológicas com o objetivo de eliminar ou delimitar contornos desnecessários. A estrutura elementar destas transformações será uma cruz, pois é pretendido determinar os contornos dos quadrados de cada face do cubo mágico e a cruz permitirá realçar esses contornos e eliminar aqueles que não cumprem este critério. A primeira transformação é uma operação de fecho para fechar os contornos que o algoritmo da canny não fechou, de seguida é efetuada uma dilatação para aumentar os contornos que sobreviveram à primeira transformação, depois é executado mais uma vez a operação de fecho para fechar os novos contornos dilatados e por último é realizada uma erosão para eliminar eventuais ruídos que tenham sobrevivido às transformações anteriores. Com estas transformações é possível preparar a frame

para ser analisada e realçar os contornos mais importantes para essa análise.

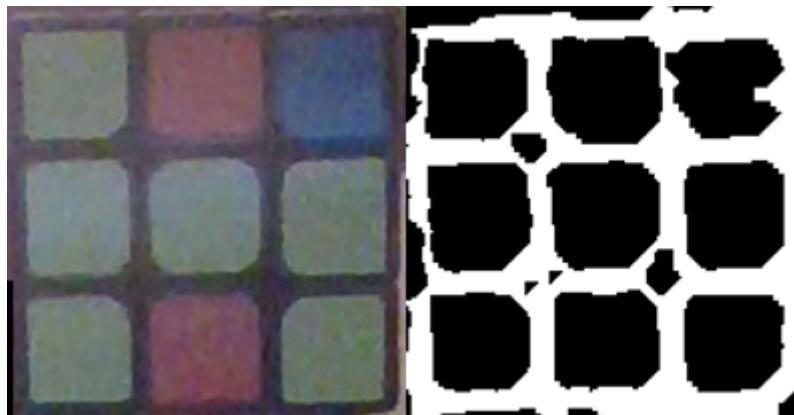


Figura 4.20: Resultado das operações morfológicas.

4.4.4 Deteção do cubo

A deteção do cubo será efetuada depois que as frames tenham sido transformadas e preparadas para a análise. Esta deteção basear-se-á em extrair as características dos contornos detetados em cada frame e usá-los para determinar se fazem parte do cubo mágico, a sua posição na face e a sua respetiva cor.

O primeiro passo a ser executado é a eliminação de contornos que não representam a face do cubo mágico. O algoritmo de deteção de contornos de canny irá localizar todos os contornos presentes na frame e não apenas os contornos presentes no cubo mágico, consequentemente, é necessário separar os contornos que representam as faces do cubo mágico dos outros contornos. Este objetivo é atingido ao efetuar a verificação de se o contorno possui características quadradas e se possui um contorno pai. A verificação de se o contorno possui características quadradas é efetuada ao extrair a área, A , e o perímetro, P , do contorno e aplicar a fórmula $P^2/16$ e comparar o seu resultado com a área. Esta fórmula advém da característica do cubo de que todos os seus lados possuem o mesmo comprimento, ou seja, a fórmula da área e do perímetro podem ser traduzidas da seguinte forma:

$$A = l * l = l^2 \quad (4.1)$$

$$P = l + l + l + l = 4l \quad (4.2)$$

Como se pode verificar, ambas as fórmulas podem ser simplificadas ao ponto de apenas precisarem de apenas uma variável l , lado. Portanto podemos fazer uso da fórmula anteriormente mencionada para igualar o valor do perímetro ao valor da área:

$$\frac{P^2}{16} = \frac{(4l)^2}{16} = \frac{16l^2}{16} = l^2 \quad (4.3)$$

A partir do resultado desta equação é apenas preciso compará-lo com o valor da área e verificar se ele se encontra dentro de uma margem de erro de ± 200 . Esta margem de erro foi criada devido ao fato de que os contornos raramente formam um quadrado perfeito. A necessidade de verificar se o contorno possui um contorno pai advém do fato de que a verificação de se o contorno possui características quadradas não é perfeita e pode capturar outros objetos quadrados que o utilizador possa possuir no ambiente, portanto esta verificação reforça a deteção das faces do cubo mágico em relação a outros objetos que possam existir na imagem. Um contorno pai é um contorno que possui dentro de si um ou mais contornos. Todos os cubos mágicos são envolvidos por bordas pretas que separam cada quadrado presentes nas faces do cubo, portanto todos os contornos que representam os quadrados na face do cubo mágico devem possuir um contorno pai.

Após os contornos que podem representar a face do cubo mágico estarem delimitados, será necessário extraír as coordenadas das posições desses contornos e a área que os mesmos ocupam. Estas informações serão necessárias para ordenar os contornos consoante as suas posições na face do cubo mágico para as diferentes distâncias a qual o cubo mágico possa se encontrar da câmara do utilizador e permitirão a estimativa das posições dos quadrados na face do cubo mágico que não foram detetados pelo algoritmo de deteção de contornos de canny. Por outro lado, nesta fase também é determinado a cor que os contornos já detetados ocupam na face do cubo mágico através do algoritmo de deteção de cores que será explicado posteriormente.

Quando os contornos que representam o cubo são detetados, na maior parte das vezes eles se encontraram desorganizados, que por consequente cria a necessidade de organizá-los para que a cor detetada pelo contorno esteja na localização correta. Para organizar os contornos detetados é inicialmente estimado os maiores valores das coordenadas x e y, as coordenadas

que representariam o canto inferior direito do cubo mágico.

De seguida, são comparadas as coordenadas dos contornos detetados com os maiores valores das coordenadas x e y para determinar a posição de cada contorno. Caso a diferença entre as coordenadas, x ou y seja, inferior a 10, o contorno estará localizado na última coluna ou última linha do cubo respetivamente. Por outro lado, se a diferença for superior a 10 é verificado a área do contorno e dependendo desse valor é efetuado uma comparação para observar se o contorno pertence à linha do meio ou à coluna do meio. Este algoritmo foi configurado desta forma para que o mesmo fosse adaptável às várias distâncias possíveis que o utilizador poderia colocar o cubo da câmara. Se o valor da diferença for superior ao valor determinada pela área, 66 se a área for inferior a 2000, 77 se for superior a 2000 e inferior a 3000 e 86 se for superior a 3000 e inferior a 4000, o contorno será colocado na primeira linha ou na primeira coluna. Por fim, os valores mencionados foram estimados depois de terem sido efetuados vários testes com o valor da área dos contornos e a distância entre as coordenadas.



Figura 4.21: Distâncias entre os cubos para diferentes tamanhos.

Por último, foi adicionado uma função à deteção do cubo para que ela fosse capaz de estimar a posição de quadrados que não foram detetados na face do cubo mágico. Por vezes o algoritmo da deteção do cubo pode falhar e não ser capaz de detetar a face completa do cubo mágico, portanto esta função foi desenvolvida para auxiliar o algoritmo a resolver este problema. Esta estimação apenas é executada se o algoritmo for capaz de obter no mínimo 7 contornos que podem representar o cubo mágico, pois ela é dependente da função para organizar os contornos na face do cubo mágico. A função de organização necessita que pelo menos uma linha completa e uma

coluna completa sejam detetadas para que a mesma saiba a localização dos valores extremos na face do cubo mágico, por consequente é preciso que sejam detetados no mínimo 7 contornos numa face uma vez que isso garante que pelo menos uma linha e uma coluna completa são detetadas. A função de estimação faz uso desta característica da função de organização para determinar as posições que precisam ser determinadas e que valores deve usar para a determinação. A estimação começa por procurar o quadrado na face que precisa ser estimado, ao encontrar-lo ela aplica um de dois métodos consoante as condições da situação para encontrar a localização do quadrado. O primeiro método faz uso dos contornos que se encontram na mesma linha do quadrado a ser estimado, neste caso a posição y seria a média dos outros dois contornos e o cálculo da posição x vai depender do lugar que está a ser estimado. Por outro lado, existe a possibilidade de um dos contornos localizados na mesma linha que o quadrado a ser estimado também precisar de ser estimado, neste caso é feito uso dos contornos pertencentes à mesma coluna que o quadrado a ser estimado. Quando são usados os contornos pertencentes à mesma coluna em vez dos contornos pertencentes à mesma linha, a posição x é a média dos dois contornos e o cálculo da posição y vai depender do lugar que está a ser estimado. Por fim, quando todos os quadrados em falta forem estimados é aplicado o algoritmo de deteção de cor para determinar a cor das localizações estimadas.

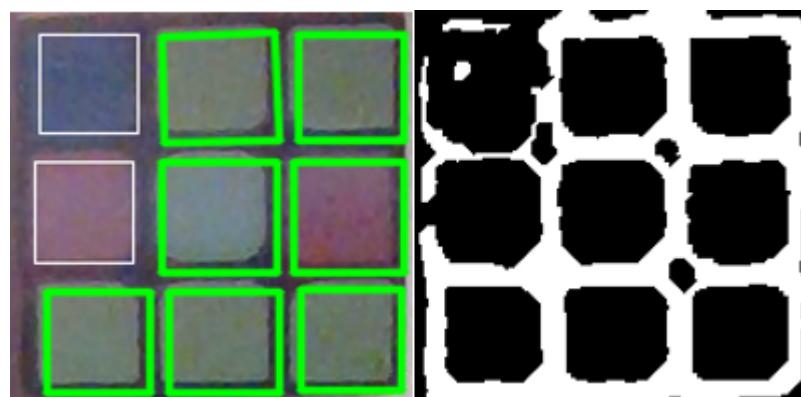


Figura 4.22: Contornos estimados

4.4.5 Deteção das cores

O algoritmo de deteção das cores é executado durante a primeira determinação dos contornos que fazem parte do cubo e durante a estimativa dos contornos em falta. Este algoritmo permitirá que seja possível determinar a cor de cada quadrado na face do cubo mágico a partir dos contornos que representam os mesmos.

Antes que o algoritmo seja executado, logo após o utilizador começar a deteção do cubo, é pedido que seja mostrado cada uma das cores do cubo mágico na câmara do computador do utilizador. Este passo é de extrema importância para a deteção das cores, pois permite que o algoritmo possa se adaptar a qualquer ambiente em que se encontre. Durante esta fase é disponibilizado ao utilizador algumas ferramentas para alterar a luminosidade das imagens recebidas para que o algoritmo possa funcionar em ambientes mais escuros onde a distinção entre as cores não é tão clara. Quando uma cor é mostrada à câmara do utilizador, o algoritmo faz uma média dos pixéis que representam essa cor em dois espaços de cores, RGB e YUV. O espaço de cores RGB é utilizado para demonstrar ao utilizador a cor que está a ser detetada, de forma que o mesmo possa ajustá-la através das ferramentas para alterar a luminosidade até ficar do seu gosto. Por outro lado, o espaço de cores YUV é usado no algoritmo de deteção de cores.

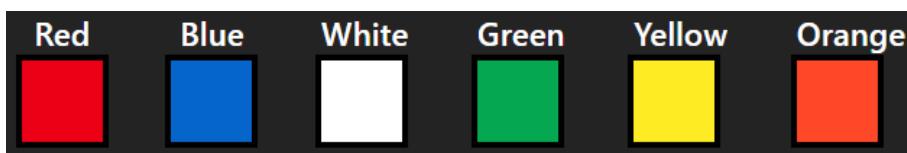


Figura 4.23: Apresentação das cores ao utilizador.

O espaço RGB não é utilizado no algoritmo devido ao facto de que o olho humano é mais sensível à luminância do que à crominância das cores, que por consequente dificulta a deteção delas quando o utilizador está a alterar a luminosidade da imagem para tornar as cores mais percepíveis na sua visão, pois pode diminuir o contraste das mesmas nos canais RGB. Porém, o espaço de cores YUV tem em consideração a percepção humana, ou seja, favorece mais a luminância do que a crominância e esta característica torna este espaço de cores uma mais – valia para a deteção das mesmas, pois quando as cores se tornam mais distintas no olhar do utilizador, elas também se tornam mais

distintas neste espaço de cores. Após o processo obter os resultados das médias dos valores dos pixéis nos dois espaços de cores, ele guarda esses valores para serem usados posteriormente durante a deteção do cubo.



Figura 4.24: Espaço de cores YUV

O primeiro passo que o algoritmo executa é a determinação do retângulo que delimita toda a área encobrida pelos contornos detetados e estimados com o objetivo de aplicar a função “roi” do Opencv que devolve a secção da imagem onde o retângulo delimitador se encontra. A partir desta secção de imagem é possível calcular a média de todos os pixéis presentes nela e obter o valor médio da cor do contorno a ser correntemente analisado. Este valor médio é calculado no espaço de cor YUV.

O segundo passo que o algoritmo executa é o cálculo da distância entre os valores obtidos na tarefa executada pelo utilizador e os valores obtidos na deteção do cubo. A fórmula da determinação das cores do cubo mágico funciona como um cálculo da distância euclidiana entre os resultados observados. O cálculo da distância euclidiana é efetuado com o objetivo de estimar de forma direta o quanto longe a cor detetada se encontra das cores guardadas, basicamente o raciocínio por de trás deste processo é de que os valores que representam as mesmas cores vão possuir uma distância menor do que os valores que representam cores diferentes. Este cálculo é efetuado usando os canais do espaço de cores YUV, luma, blue projection e red projection, como se fossem coordenadas de um ponto num espaço 3D, desta maneira é possível estimar a distância entre pontos detetados e os pontos guardados. Por fim, após o processo ter sido efetuado para todas as cores guardadas, a cor que obter uma menor distância entre todas as outras é a escolhida pelo algoritmo

para representar o contorno detetado ou estimado.

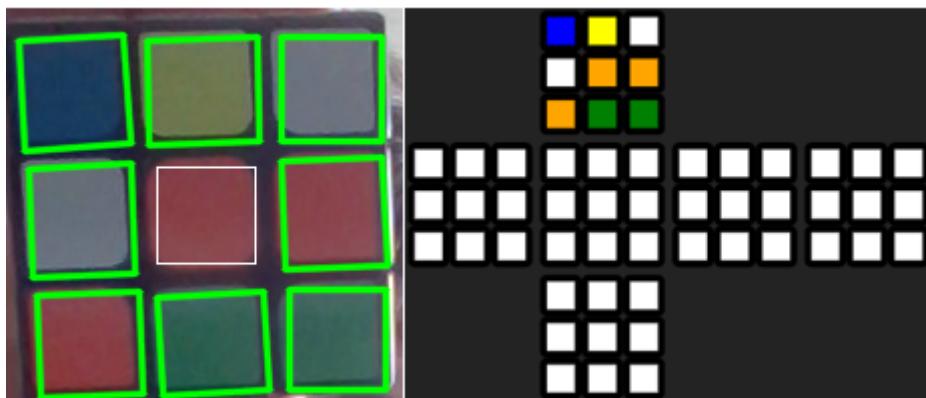


Figura 4.25: Deteção das cores

4.4.6 Apoio ao utilizador

O pedido para mostrar as cores do cubo mágico e o processo para determinar as faces do mesmo pode ser bastante complicado para o utilizador perceber na sua primeira vez. Portanto, foram adicionadas algumas ferramentas para facilitar a interação entre o utilizador e o algoritmo de deteção por câmara do cubo mágico.

Todos os utilizadores do website vão possuir os seus próprios ambientes de iluminação e cubo mágicos que podem possuir cores diferentes do padrão, por consequente seria apreciável guardar as cores detetadas no ambiente em que o utilizador se encontra para evitar que o mesmo tenha que fornecer as cores do seu cubo todas as vezes que ele queira utilizar a deteção por câmara. Quando o utilizador termina de fornecer as cores do cubo mágico, elas são guardadas na base de dados utilizando o Firebase e associadas ao utilizador que efetuou a deteção por câmara. Desta forma, sempre que o utilizador fazer uso desta função do website ele não precisará fornecer as cores do seu cubo mágico novamente, a menos que as condições do seu ambiente mudem.

Por outro lado, enquanto o utilizador está a mostrar as faces a serem detetadas pelo algoritmo pode haver uma confusão sobre que faces o mesmo deve mostrar e a orientação das mesmas. Para resolver este problema, foi adicionado uma animação no canto inferior direito da câmara do utilizador que indica como o mesmo deve rodar o seu cubo mágico para obter a face

correta para ser detetada. Esta animação é adaptável para o caso em que o utilizador esteja a guardar uma nova face ou a eliminar uma face já detetada, mostrando as rotações que devem ser efetuadas para atingir a próxima face ou a face antiga.

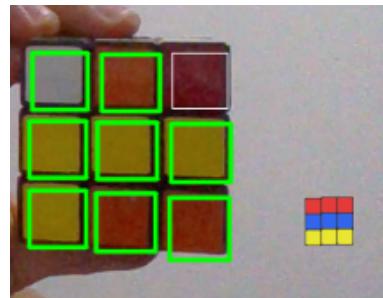


Figura 4.26: Animação que indica a rotação a ser efetuada.

4.5 Conclusão

Durante a realização desta fase do projeto, determinou-se que as tecnologias escolhidas e os métodos de implementação são capazes de reproduzir as funcionalidades pretendidas.

A utilização do Firebase permitiu uma maneira simples e eficaz guardar a informação de cada utilizador e, por consequente, aumentar as capacidades das outras funcionalidades através das informações na base de dados.

O uso da biblioteca Three.js e a aplicação Blender provou facilitar a construção do cubo mágico, devido ao facto de não haver a necessidade de construir o cubo manualmente. A partir do modelo criado no Blender, é possível criar um cubo suficientemente bom sem a necessidade de modelá-lo pelo Javascript.

O OpenCv.js foi uma mais-valia para este trabalho por fornecer funções de transformação e análise de imagens que ajudaram a deteção do cubo mágico pela câmara do utilizador.

Por fim, a maneira como os dados do cubo mágico são guardados é simples, mas eficiente para os vários movimentos possíveis e fórmulas complexas que o cubo pode possuir.

Capítulo 5

Validação e Testes

5.1 Resolução do cubo automaticamente

Fizemos o teste da resolução automática, onde começamos com o cubo mágico baralhado e o sistema encontra os movimentos para o estado resolvido do cubo mágico. O algoritmo encontra sempre a solução do cubo baralhado e o sistema aplica-a.

Testamos as três maneiras diferentes do utilizador resolver o cubo magico:

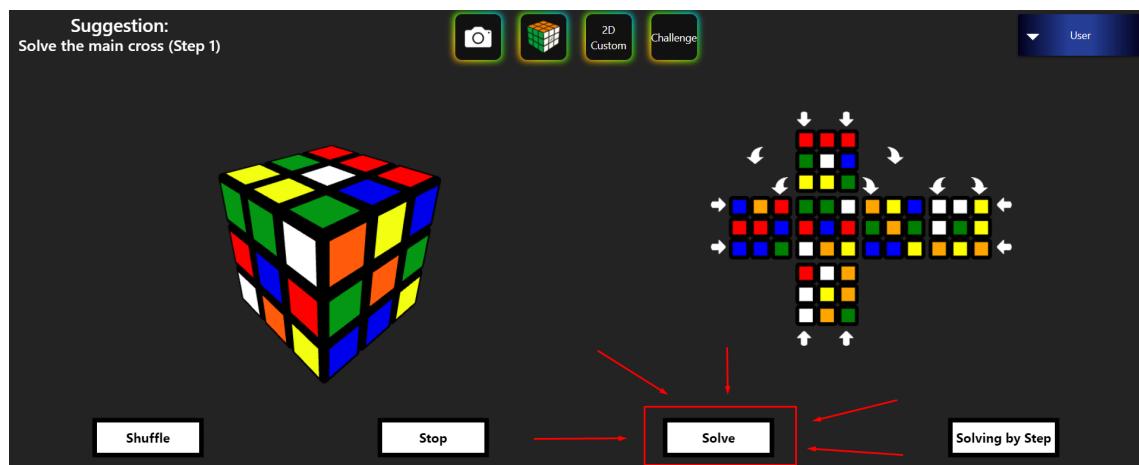


Figura 5.1: Resolução através de um botão

O utilizador consegue resolver o cubo todo apenas clicando em um botão, conseguindo visualizar a sua resolução passo a passo.

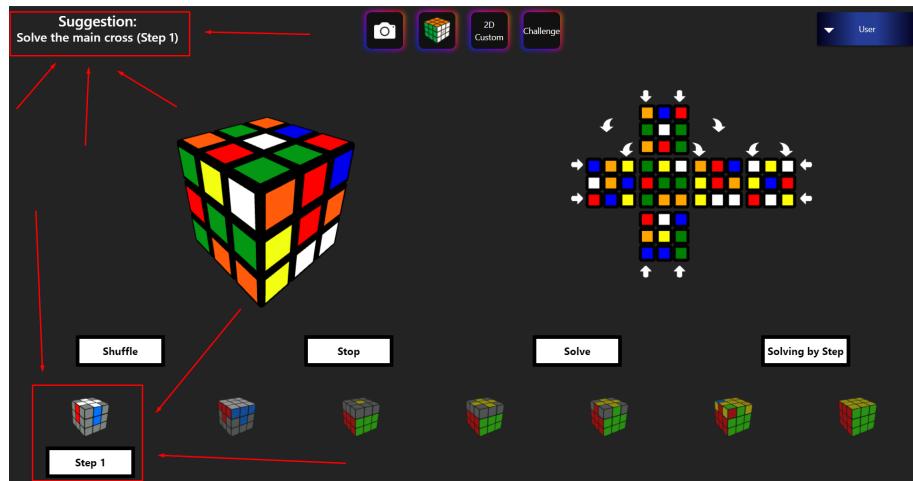


Figura 5.2: Resolução através dos passos recomendados

O utilizador consegue resolver o cubo passo a passo (sete passos) recebendo dicas e imagens de fórmulas de ajuda, ou seguir a resolução automática passo a passo.

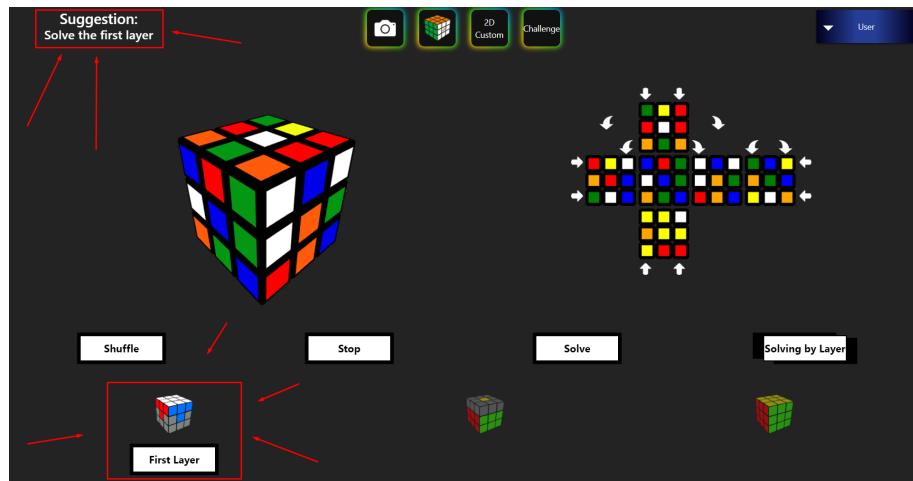


Figura 5.3: Resolução através das camadas recomendadas

O utilizador consegue resolver o cubo por camadas (três camadas) recebendo dicas e imagens de fórmulas de ajuda, ou seguir a resolução automática camada a camada.

5.2 Filtro de Anti-Aliasing

O filtro Anti-Aliasing atua como uma “borracha”, pois quando passa nas bordas de uma figura age como um blur, esta ferramenta é utilizada para tornar os contornos mais realistas.

Foi aplicado um filtro ao render do cubo 3d para suavizar a imagem, sendo aplicado o anti-aliasing a cada frame durante a execução.

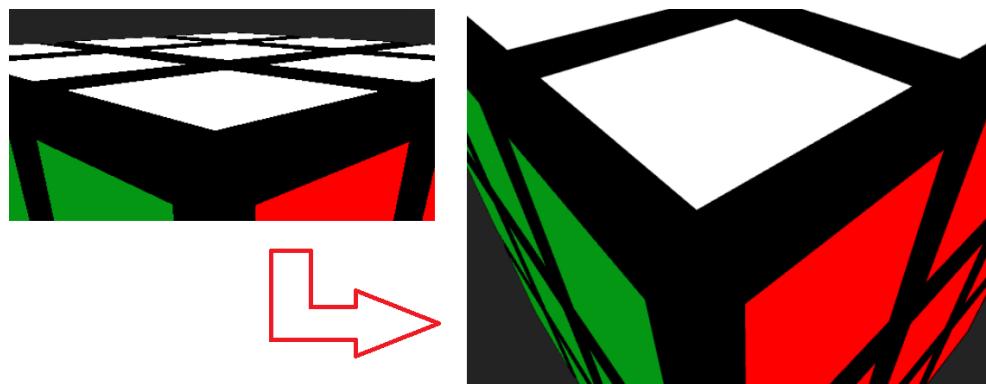


Figura 5.4: Anti-Aliasing teste 1: modelo 3d antes e depois do filtro

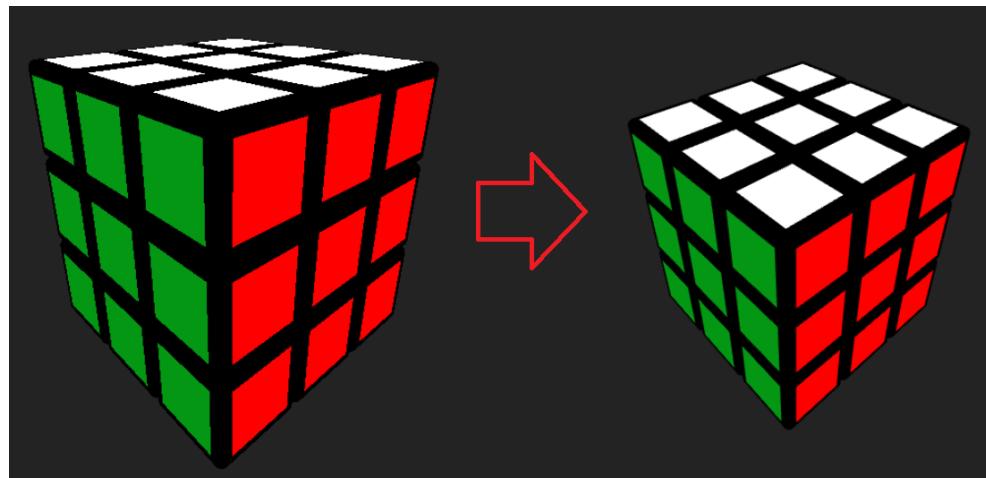


Figura 5.5: Anti-Aliasing teste 2: modelo 3d antes e depois do filtro

O filtro não utiliza muitos recursos da máquina, sendo uma ferramenta recorrente nos projetos que pesquisamos de “Three.js”.

5.3 Ordenação dos contornos do cubo

Durante a explicação sobre o funcionamento da deteção do cubo, 4.4.4, foi referido a importância da ordenação dos contornos que representam a face do cubo mágico e o método utilizado para realizar esta tarefa, a estimativa das coordenadas do canto inferior direito da face e a distância entre os quadrados da mesma.

A distância entre os quadrados de uma face é uma informação importante a se ter durante esta ordenação, pois permite estimar a posição dos outros contornos com base no quadrado localizado no canto inferior direito da face que foi determinado a partir dos maiores valores x e y dos contornos detetados.

A fim de adquirir esta informação, foram efetuados alguns testes para descobrir a distância máxima a qual quadrados adjacentes horizontalmente e verticalmente se encontram para várias áreas de forma que a deteção do cubo fosse adaptável à sua distância da câmara do utilizador.

Área	< 2000	< 3000	< 4000
max x	55	66	79
max y	1	2	5

Tabela 5.1: Distâncias entre quadrados adjacentes horizontalmente

Área	< 2000	< 3000	< 4000
max x	2,5	6	7
max y	56	68	80,5

Tabela 5.2: Distâncias entre quadrados adjacentes verticalmente

Como se pode verificar, tanto a distância vertical, y, para quadrados na mesma linha como a distância horizontal, x, para quadrados na mesma coluna são sempre inferiores a 10 para todas as áreas e para uma área inferior a 2000 todas as distâncias são inferiores a 66, para uma área inferior a 3000 todas as distâncias são inferiores a 77 e para uma área inferior a 4000 elas são inferiores a 86, reforçando a explicação sobre os valores utilizados na secção 4.4.4.

5.4 Espaço de cores RGB e YUV

Na secção 4.4.5 foi explicado o processo pelo qual o algoritmo de deteção de cores executa as suas funções através do uso do espaço de cores YUV em vez do espaço de cores RGB devido às características do YUV que o torna mais favorável que o RGB. Nesta secção será apresentado os testes que demonstram a vantagens da utilização do YUV.

O espaço de cores RGB favorece a crominância, a mudança no valor das cores, porém como o olho humano não é muito percepçável à variância das cores esta característica não possui muito impacto e pode atrapalhar a deteção das mesmas quando existem cores parecidas na mesma face.

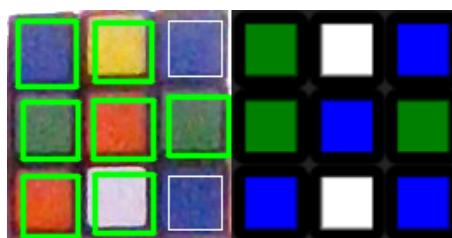


Figura 5.6: Teste com o espaço de cores RGB

Como se pode observar, as cores mais escuras como o laranja, azul e verde são confundidas como sendo a mesma cor, da mesma forma as cores mais claras como o amarelo e o branco também são detetadas como sendo da mesma cor. O espaço de cores YUV favorece a luminância tal como o olho humano e por consequente consegue distinguir melhor as cores que se apresentam mais distintas nos olhos do utilizador.

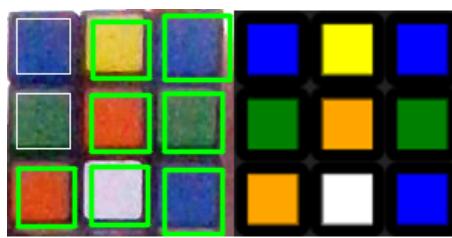


Figura 5.7: Teste com o espaço de cores YUV

Tal como é apresentado, neste espaço as cores são corretamente distinguidas e não existe nenhuma confusão por parte do algoritmo de deteção de cores.

5.5 Inserir o cubo utilizando a camera

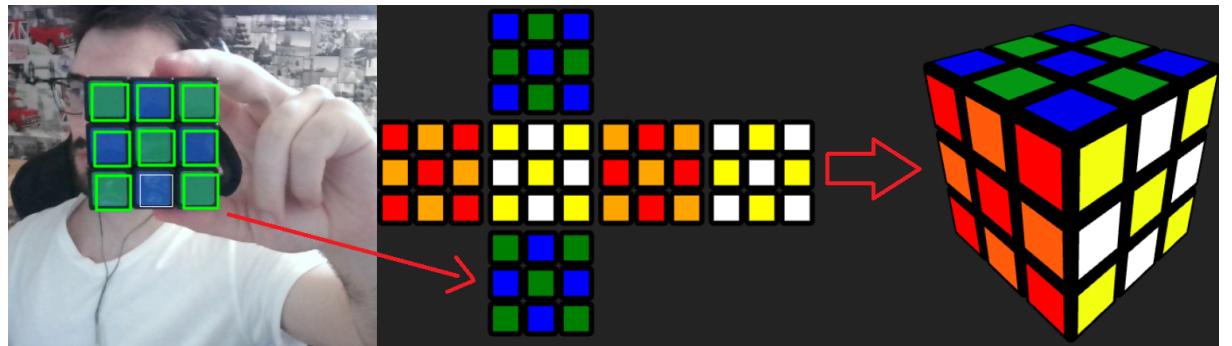


Figura 5.8: Inserir o cubo dinamicamente

Neste teste o grupo demonstra como o utilizador consegue inserir o seu estado do cubo através da câmara, onde uma função valida que todas as peças do cubo existem, o cubo pode ser inserido com sucesso, ficando todas as ferramentas de resolução automática disponíveis ao utilizador.

5.6 Inserir o cubo utilizando a ferramenta manual

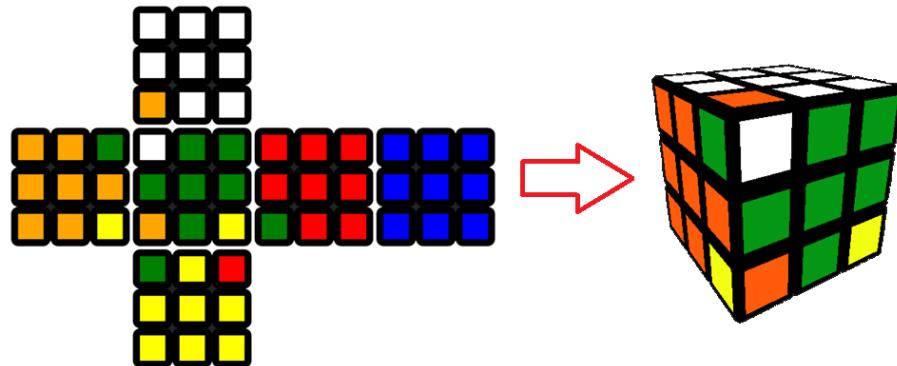


Figura 5.9: Inserir o cubo manualmente

Para complementar o trabalho, o grupo decidiu implementar uma ferramenta manual para os utilizadores que não possuem uma câmara.

Capítulo 6

Conclusões e Trabalho Futuro

Neste projeto, nós planeamos implementar o cubo mágico como era originalmente pedido através do Javascript e construir a ferramenta para reconhecer o cubo através da câmara. Porém, devido ao facto do nosso grupo ser composto por três pessoas, mais ferramentas foram implementadas no nosso trabalho ao longo do tempo, como a configuração manual do cubo, o desafio de tempo e a implementação de uma base de dados com diferentes tipos de utilizador. No final todas as ambições iniciais e objetivos que surgiram ao longo da realização do projeto foram realizados com sucesso.

Durante a realização deste trabalho, melhoramos o nosso conhecimento sobre as tecnologias aprendidas e a fazer uso de novas tecnologias. Através da implementação do cubo 3D, a resolução automática e a configuração manual exploramos mais as capacidades do Javascript e a lógica por detrás dos movimentos, construção e resolução do cubo mágico.

O reconhecimento do cubo mágico através da câmara foi uma das partes mais complexas de implementar devido à mesma fazer uso de uma das novas tecnologias utilizadas neste projeto, o OpenCV.js. A biblioteca OpenCV já era conhecida e fora utilizada em trabalhos anteriores, porém a sua versão do Javascript aplicava uma forma diferente de ser usada do que foi aprendido para outras linguagens. Por consequente, a realização desta ferramenta ensinou-nos como trabalhar com esta versão do OpenCV para o Javascript, assim como nos mostrou como trabalhar com análises de imagem e espaços de cor de forma a detetar o cubo mágico.

O desafio de tempo apresentou alguns desafios devido à necessidade de guardar o tempo de resolução do cubo de cada utilizador e mostrá-lo em

uma classificação mundial. Nós utilizamos o Firebase, uma tecnologia que havíamos aprendido anteriormente, para implementar esta ferramenta, porém, da mesma forma que o OpenCV.js, trabalhámos com uma versão adaptada para o Javascript. Consequentemente, nós aprendemos a guardar e recolher informações de uma base de dados através do Javascript e como organizar conteúdos guardados no FireBase.

Por outro lado, foi notado o potencial para a procura em espaço de estados onde definindo os estados e os operadores de transição seria possível descrever o problema. Assim, seria possível utilizar cubos com a dimensão variável como $2 \times 2 \times 2$ ou $4 \times 4 \times 4$, sendo apenas necessário alterar o problema da procura de espaço de estados. Isto teria bastantes benefícios como reduzir drasticamente a quantidade de movimentos para a resolução do mesmo, aumentar o número de puzzles do site, mas por outro lado, a procura em espaço de estados aumenta a necessidade do poder computacional, e no geral, num possível atraso entre o tempo que o utilizador pede a resolução, e a resposta do algoritmo.

Em suma, este projeto exigiu a pesquisa por novas formas de representação e novas tecnologias para o Javascript que permitissem a implementação dos objetivos pretendidos durante a realização deste trabalho. Esta exigência permitiu que pensássemos por fora dos conhecimentos que havíamos adquirido ao longo dos anos, pesquisássemos e aprendêssemos a utilizar outros métodos para atingir a nossa visão para o projeto.

Apêndice A

Wireframes

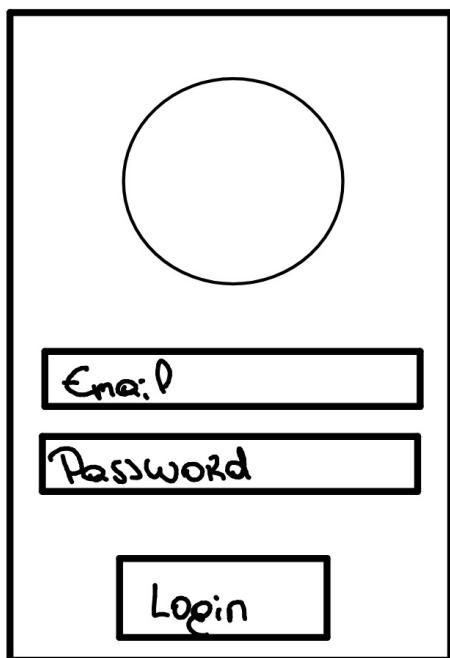


Figura A.1: Login

A hand-drawn wireframe of a registration form. It features a large circular placeholder at the top, followed by four rectangular input fields labeled "USERNAME", "Email", "Password", and "Repeat password". Below these fields is a single button labeled "Sign UP".

Figura A.2: Registrar

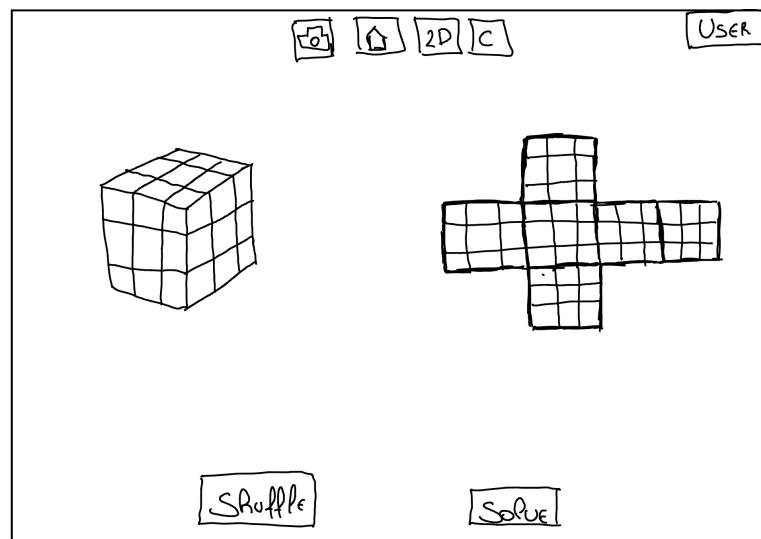


Figura A.3: Index

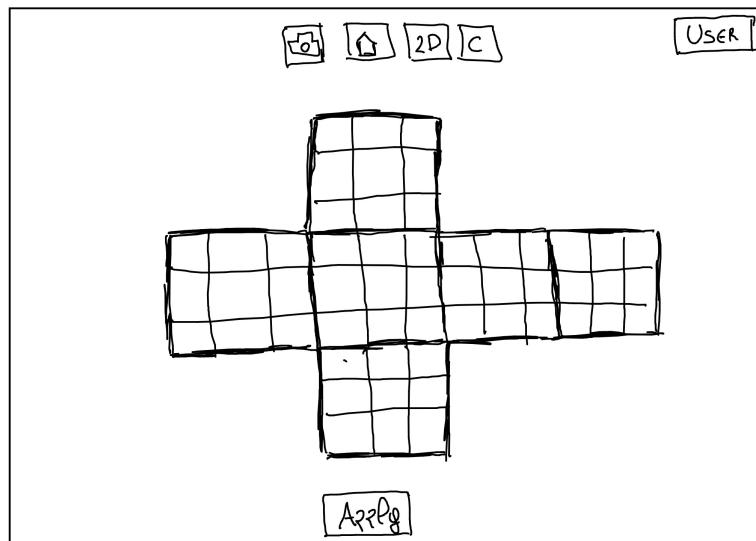


Figura A.4: Cubo 2d inserido manualmente

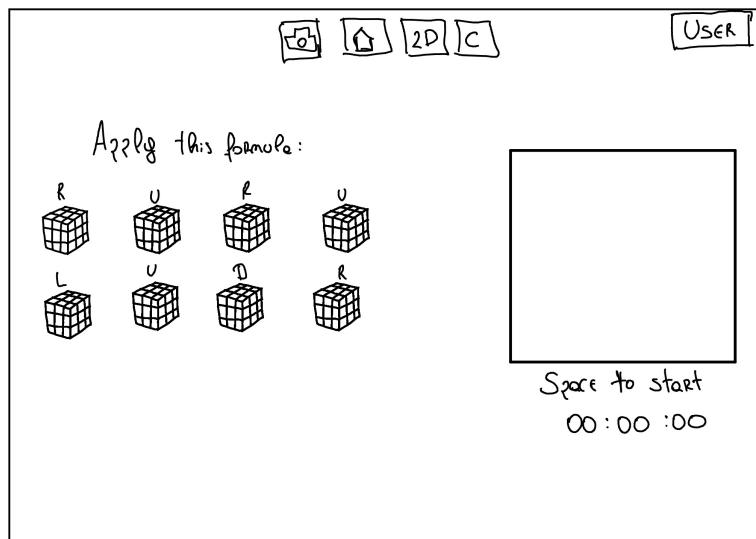


Figura A.5: Desafio

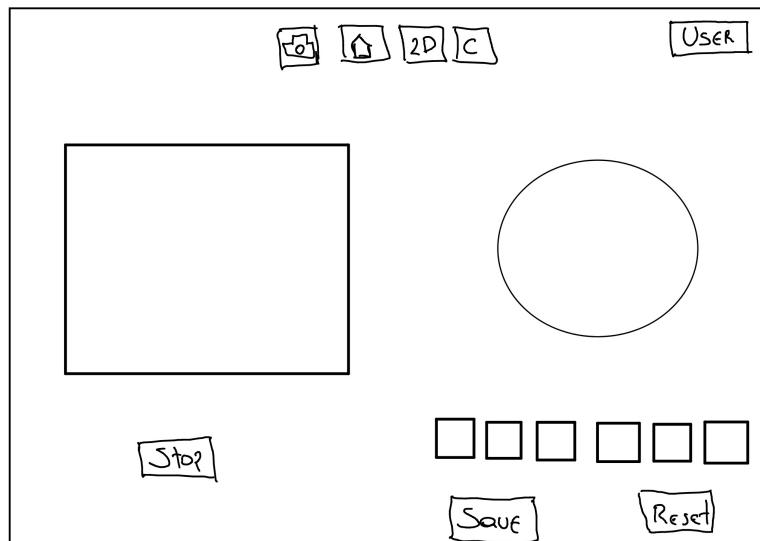


Figura A.6: Selecionar cores

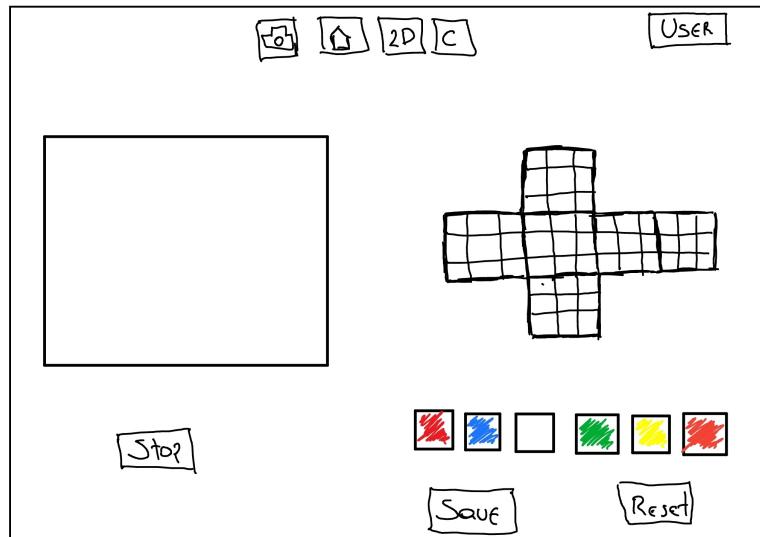


Figura A.7: Selecionar faces

Bibliografia

[Beke,] Beke, M. Online rubik's cube solver. <https://rubiks-cube-solver.com>.

[Opencv.js, 2017] Opencv.js (2017). Open source computer vision library. https://docs.opencv.org/3.4/d5/d10/tutorial_js_root.html.

[Riera,] Riera, A. Rubik's cube. <http://www.adrianriera.com/other/rubik/>.

[Ruwix,] Ruwix. Advanced rubik's cube notation. <https://ruwix.com/the-rubiks-cube/notation/advanced/>.

[Smith, 2014] Smith, S. (2014). Rubik's cube explorer. <https://iamthecu.be/>.

[Xu, 2015] Xu, D. (2015). Rubik's cube solver. <https://codepen.io/dfx113/pen/YwGgX0>.