DDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

MEIM - Mestrado Engenharia informática e multimédia

Visão Artificial e Realidade Mista

Projeto 2

Turma:

MEIM-21N

Trabalho realizado por:

Duarte Domingues Nº45140

Docente:

Pedro Mendes Jorge

Data: 12/06/2022

Índice

Índice de figuras

1. Introdução

Este projeto baseia-se no desenvolvimento de uma aplicação baseada em realidade aumentada por marcadores, que permite a inclusão de elementos virtuais alinhados com marcadores fiduciais reais.

A primeira fase do projeto assenta-se na calibração da câmara utilizando funções do OpenCV. Neste projeto de forma a captar vídeo em tempo real usa-se a câmara do computador, este tipo de câmara tem uma lente do género "*pinhole*". Estes tipos de lentes introduzem distorção significativa às imagens, respetivamente distorção radial e tangencial. Distorção radial provoca que linhas retas apareçam curvas e distorção tangencial provoca que certas áreas da imagem apareçam mais próximas do que seria esperado. O processo de calibração da câmara foi utilizado para obter os parâmetros intrínsecos da câmara do computador e os coeficientes de distorção de forma a corrigir a distorção causada pela lente referida anteriormente.

A segunda fase do projeto baseia-se na deteção e estimação da pose da câmara com a biblioteca ArUco do OpenCV. Esta biblioteca é uma biblioteca baseada em marcadores ArUco para aplicações de realidade aumentada. Os marcadores Aruco são quadrados com uma estrutura de uma matriz binária interna dentro de uma borda preta. Estes marcadores são facilmente distinguíveis do resto do ambiente, o marcador é posicionado no objeto ou na cena que está a ser imaginada, tendo características que o permitem ser facilmente detetado e identificado. Nesta fase foi realizada a deteção dos marcadores e a estimação da pose da câmara para cada marcador, de forma a ser possível a inclusão de objetos virtuais alinhados com os marcadores.

A fase final do projeto baseia-se na introdução de objetos virtuais com as coordenadas do mundo real associadas com os marcadores identificados, cada marcador tem um identificador único. Os objetos virtuais projetados em tempo real foram cubos com diferentes cores, tendo cada cubo um identificador único e pirâmides.

2. Desenvolvimento

2.1. Calibração da câmara

Nesta etapa realizou-se a calibração da câmara de forma a corrigir distorção causada pela câmara do computador.
A distorção radial provoca que linhas retas apareçam curvas, quanto mais longe os pontos estiverem do centro da imagem maior é a distorção radial.
A distorção tangencial é provocada pela lente da câmara não estar alinhada paralelamente ao plano da imagem, provocando que certas partes da imagem pareçam mais próximas do que devido.

Foi então necessário obter os parâmetros intrínsecos e extrínsecos da câmara de forma a conseguir corrigir a distorção causada pela câmara em imagens com base nestes parâmetros.

De forma a conseguir entender a calibração da câmara foi necessário entender o sistema de coordenadas do mundo e da imagem.

2.2. Formação de imagens

De forma a projetar um ponto 3D para o plano de imagem são realizadas diferentes operações baseadas em três sistemas de coordenadas:

- Sistema de coordenadas do mundo.
- Sistema de coordenadas da câmara.
- Sistema de coordenadas da imagem.

2.2.1 Sistema de coordenadas do mundo

Dado um ponto 3D no mundo real, é necessário ter uma maneira de descobrir as coordenadas dos pixeis (u,v) deste ponto 3D na imagem tirada pela câmara.

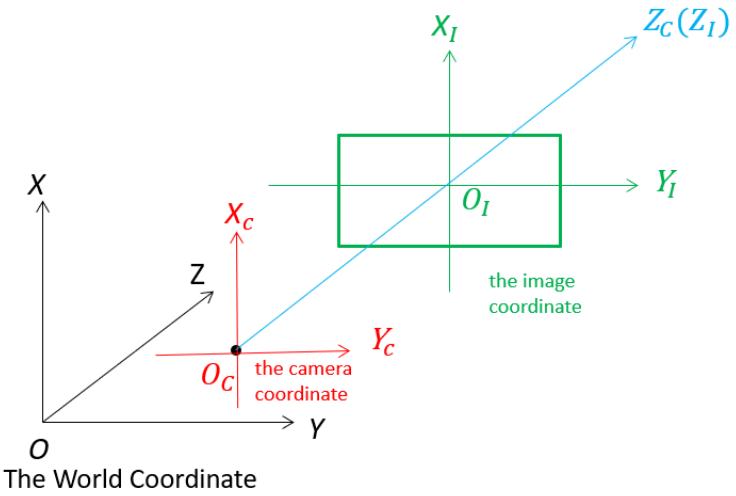

Figura 1 - Sistemas de coordenadas

Para definir a localização do ponto 3D no mundo real, temos que ter definido o sistema de coordenadas do mundo real, que requere:

- Origem – Ponto definido como o início (0,0,0) do mundo real.
- Eixos X, Z, Y – Os eixos são definidos a partir da origem, podendo representar assim uma coordenada 3D.

A partir deste sistema de coordenadas do mundo é possível encontrar as coordenadas de qualquer ponto 3D medindo a sua distância da origem nos eixos X, Y, Z.

2.2.2 Sistema de coordenadas da câmara

Como se pode observar na figura 1 a cima, o sistema de coordenadas do mundo e o sistema de coordenadas da câmara estão relacionados por uma rotação e translação.

Como a câmara pode estar em qualquer lugar no mundo real, as coordenadas da câmara estão transicionadas em relação às coordenadas do mundo. Como a câmara pode estar a apontar em diferentes direções, as coordenadas da câmara estão rodadas em relação ao mundo real.

A rotação e translação é definida por 6 parâmetros (3 rotação, 3 translação), que são os parâmetros extrínsecos da câmara.

Os pontos das coordenadas da câmara (Xc, Yc, Zc) são encontrados a partir dos pontos do mundo real (Xw, Yw, Zw) a partir da seguinte fórmula.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$$

2.2.3 Sistema de coordenadas da imagem

Apôs serem obtidas as coordenadas do ponto 3D no sistema de coordenadas da câmara, é possível projetar o ponto no plano da imagem, obtendo as coordenadas do ponto na imagem.

A partir de um centro ótico O, o plano de imagem é posto à distância f (distância focal) do centro ótico.

A imagem projetada (x,y) do ponto 3D (Xc,Yc,Zc) é dada pelas seguintes fórmulas:

$$x = f\frac{X_c}{Z_c}$$
$$y = f\frac{Y_c}{Z_c}$$

Em termos da equação em forma de matriz a fórmula é a seguinte:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

A seguinte matriz K, é a matriz intrínseca da câmara, contendo os parâmetros intrínsecos da câmara:

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A matriz K anterior só consta a distância focal. Entretanto, os pixéis no sensor de imagem podem não ser quadrados então tem-se distância focal (Fx) para o eixo x e (Fy) para o eixo y. O centro ótico (Cx, Cy) da câmara pode não coincidir com o centro de sistemas de coordenadas da imagem, o que tem que ser tomado em conta, transformando a matriz K na seguinte matriz:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Por fim, os parâmetros que vão ter que ser conhecidos para a calibração da câmara são:

- **Parâmetros internos da camara:** Distância focal, centro ótico e coeficiente de distorção radial das lentes.
- **Parâmetros externos da câmara:** Refere a orientação da camara (rotação e translação) em relação ao sistema de coordenadas do mundo.

2.3. Implementação da calibração da câmara

De forma a realizar a deteção e estimação da pose da câmara utilizou-se o OpenCV com a biblioteca ArUco. As imagens capturadas neste projeto foram obtidas a partir da câmara com lente "*pin-hole*" do computador.
A partir deste processo vai ser possível obter a informação necessária sobre a câmara para determinar uma relação adequada entre um ponto 3D no mundo real com a sua projeção 2D correspondente.

De forma a encontrar a projeção do ponto 3D para o plano de imagem, é necessário passar o ponto do sistema de coordenadas do mundo para o sistema de coordenadas da câmara utilizando a matriz de rotação e a matriz de translação, como mencionado anteriormente.

O sistema realizado para a calibração da câmara tem as seguintes entradas e saídas:

- **Entradas** – Coleção de imagens com pontos em que as coordenadas de imagem 2D e coordenadas do mundo 3D são conhecidas.
- **Saídas** – A matriz intrínseca 3x3 **K**, a matriz de rotação 3x3 **R**, vetor de translação 3x1 **t.**

2.3.1 Definir coordenadas do mundo real com o padrão xadrez

Para definir o sistema de coordenadas do mundo real, os parâmetros vão ser calculados a partir de um conjunto de pontos 3D e as suas coordenadas no plano de imagem. Para os pontos 3D utilizaram-se fotos que contêm um padrão xadrez em diferentes orientações.
O sistema de coordenadas do mundo baseia-se no padrão xadrez. Qualquer canto do xadrez pode ser a origem do sistema de coordenadas do mundo.

A imagem utilizada para o padrão xadrez foi a seguinte, um padrão xadrez (7, 7):

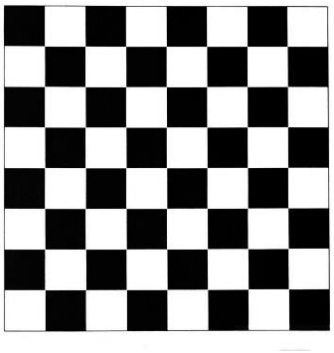

Figura 2 - Padrão xadrez (7,7)

Padrões xadrez têm a vantagem de serem facilmente distinguíveis na imagem. Os cantos do xadrez são muitos bons para serem localizados porque têm gradientes fortes em 2 direções. Os cantos estão também relacionados porque estão nas interseções das linhas do xadrez.

Para calibrar a câmara foram tiradas múltiplas imagens do xadrez de diferentes pontos de vista**.** Foram utilizadas 27 imagens diferentes com o xadrez em diferentes posições, de forma a conseguir obter bons resultados.

Na figura seguinte pode-se observar algumas imagens de referência do xadrez utilizadas para calibrar a câmara.

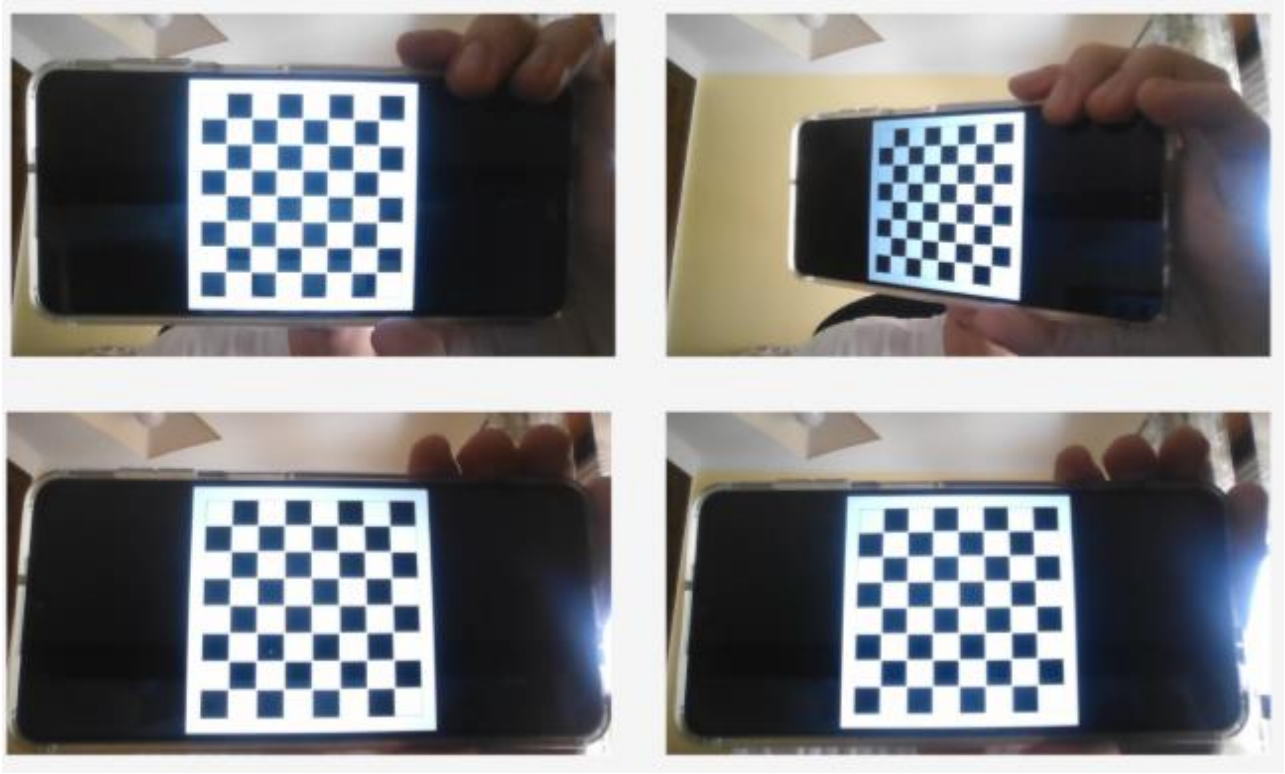

Figura 3 - Fotos com padrão xadrez para calibração da câmara

2.3.2 Encontrar cantos do xadrez

Precisamos agora das coordenadas 2D da localização dos pixéis dos cantos do xadrez nas diferentes imagens. De forma a encontrar os cantos do Xadrez foi utilizada a função findChessboardCorners do OpenCv, que procura por cantos de xadrez e retorna as coordenadas dos cantos. O *output* desta função é verdadeiro ou falso caso sejam encontrados cantos de xadrez na imagem.

2.3.3 Ajustar a precisão dos cantos do xadrez

De forma a conseguir obter ainda resultados mais precisos para a localização dos cantos, utilizou-se a função cornerSubPix do OpenCV. A partir da imagem original e da localização dos cantos a função procura pelos melhores cantos dentro da localização geral do canto original.
O processo de encontrar os cantos é iterativo então é necessário definir o critério de paragem (maxCount quando o canto se mexe por menos que um valor ou *epsilon* numa iteração especifica). Utilizou-se uma mistura dos dois critérios de paragem.

De seguida a serem redefinidos, são desenhados os cantos no xadrez, a partir da função do OpenCV drawChessboardCorners. Isto permitiu verificar que o processo de deteção dos cantos está a funcionar corretamente, como se pode observar na seguinte figura.

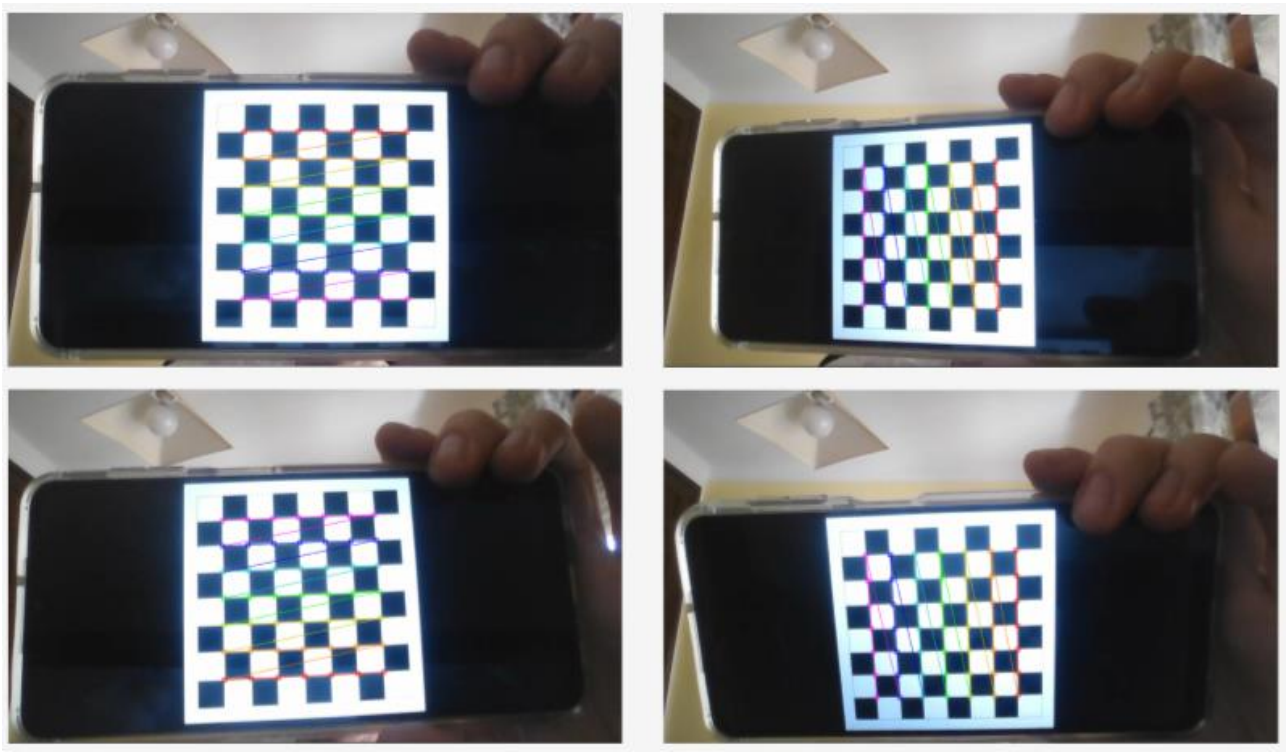

Figura 4 - Deteção dos cantos do xadrez

2.3.4 Processo geral de calibração da câmara

Para realizar a calibração da câmara foi criada a função calibrate_camera, esta função recebe como entrada:

- Um conjunto de imagens com padrão xadrez.
- A dimensão do padrão de xadrez (7, 7).
- O tipo de critério de paragem para encontrar cantos do xadrez.

A partir das operações mencionadas anteriormente, esta função permite retornar:

- Uma lista com as coordenadas 2D dos cantos do xadrez para cada imagem.
- Uma lista com as coordenadas 3D do mundo real dos cantos do xadrez para cada imagem.

2.3.5 Calibrar a câmara

Finalmente é realizada a calibração da câmara a partir dos vetores com os pontos 3D e 2D dos cantos do xadrez.
São passados os pontos 3D em coordenadas do mundo e as suas localizações 2D em todas as imagens no método calibrateCamera do OpenCV, que por sua vez retorna o seguinte:

- Matriz da câmara.
- Coeficientes de distorção.
- Vetores de rotação e translação.

Finalmente obteu-se o erro de re-projeção para ter uma estimação do quão bons os parâmetros da câmara encontrados são. O erro de re-projeção obtido foi igual a **0.035**, sendo um valor razoavél.

Agora é possível pegar em uma imagem e realizar a não distorção dela.

2.4. Deteção e estimação da pose da câmara

De forma a estimar a posição e orientação da câmara irão ser utilizados marcadores. O marcador é algo que é facilmente distinguido do resto do ambiente, o marcador é posicionado no objeto ou na cena que está a ser imaginada, tendo características que o permitem ser facilmente detetado e identificado.

Marcadores ArUco são quadrados com uma estrutura de uma matriz binária interna dentro de uma borda preta. A estrutura da matriz interna tem uma codificação única consoante a posição dos pretos e brancos permitindo ser um identificador único.
Os marcadores podem ser gerados em diferentes tamanhos, sendo o tamanho baseado no tamanho do objeto e da cena.

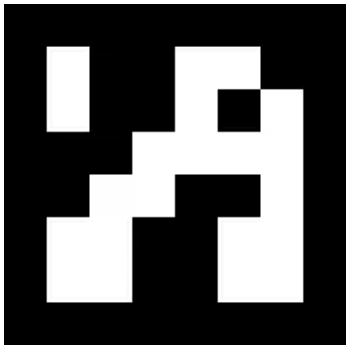

Figura 5 - Marcador ArUco

No projeto foram testadas duas alternativas para gerar os marcadores:

- Gerar marcador AruCo a partir de um Website, disponível em [4].
- Gerar marcador AruCo a partir do Python utilizando OpenCV.

2.4.1 Gerar marcador AruCo a partir de um Website, disponível em [4]

Poderia ter sido criado um marcador ArUco através de um método próprio para a criação destes marcadores, porém utilizou-se uma ferramenta externa, disponível em [4].
Neste Website criar um marcador ArUco é fácil e rápido tendo apenas que definir:

- Dicionário do marcador.
- Identificador do marcador.
- Tamanho desejado do marcador.

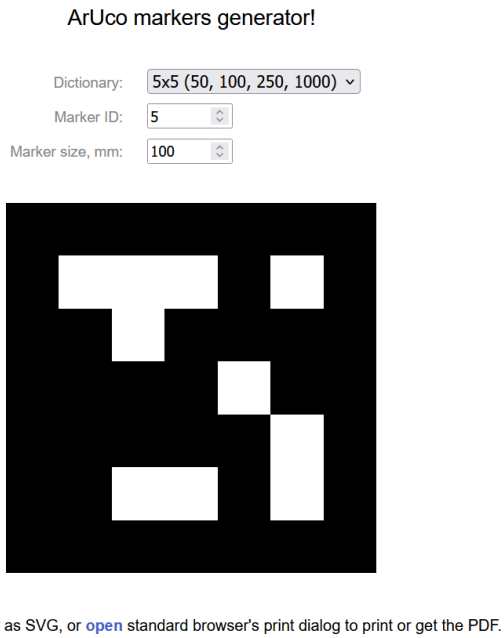

Figura 6 - Criação do marcador a partir do Website [6]

Após a geração do marcador este pode ser guardado em formato SVG ou PDF.

2.4.2 Gerar marcador AruCo a partir do Python utilizando OpenCV.

Também é possível gerar um marcador com o Python e o OpenCV. O módulo Aruco do OpenCV tem um conjunto de 20 dicionários predefinidos para criação dos marcadores, cada dicionário tem um conjunto de igual de bits e um número fixo de marcadores, por exemplo:

- DICT_4X4_50: é um dicionário predefinido de 4x4 bits e 50 marcadores.

Através da função Dictionary_get do módulo Aruco é carregado o dicionário predefinido.
De seguida para gerar o marcador utiliza-se a função drawMarker do OpenCV, esta função permite ajustar os seguintes parâmetros do marcador a ser gerado:

- Identificador do marcador.
- Tamanho da imagem gerada.
- Dicionário a ser utilizado.
- Espessura da borda.

Por fim grava-se o marcador em formato "*png*" a partir do método imwrite do OpenCV.

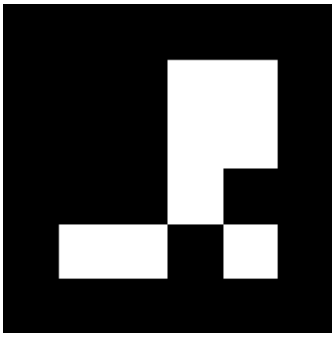

Figura 7 - Marcador resultante com dimensões 200x200

Os marcadores utilizados no trabalho prático foram criados a partir desta forma.

2.4.3 Detetar marcadores

Pretende-se agora detetar os marcadores na cena. Para isto foi criada uma função chamada detect_markers que consoante o tipo de dicionário dos marcadores permite encontrar os marcadores ArUco em uma imagem.

Inicialmente converte-se a imagem para *grayscale* de forma a facilitar os cálculos, de seguida são inicializados parâmetros de deteção de marcadores utilizando a função DetectorParameters_create do OpenCV .

De seguida, através da função detectMarkers do módulo Aruco do OpenCV obtêm-se as coordenadas dos cantos dos marcadores e os seus identificadores.

Finalmente, através da função drawDetectedMarkers do OpenCV Aruco é criada uma *bounding box* à volta dos marcadores encontrados.

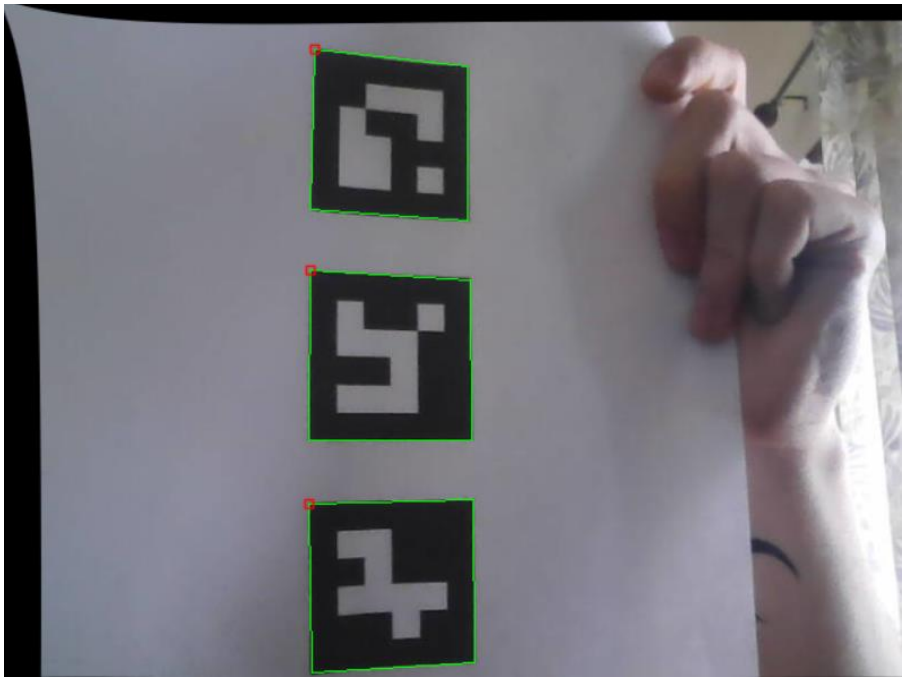

Figura 8 - Deteção dos marcadores, com uma bounding box verde à volta

Cada marcador encontrado é devidamente identificado, para poderem ser adicionados objetos virtuais consoante o identificador futuramente.

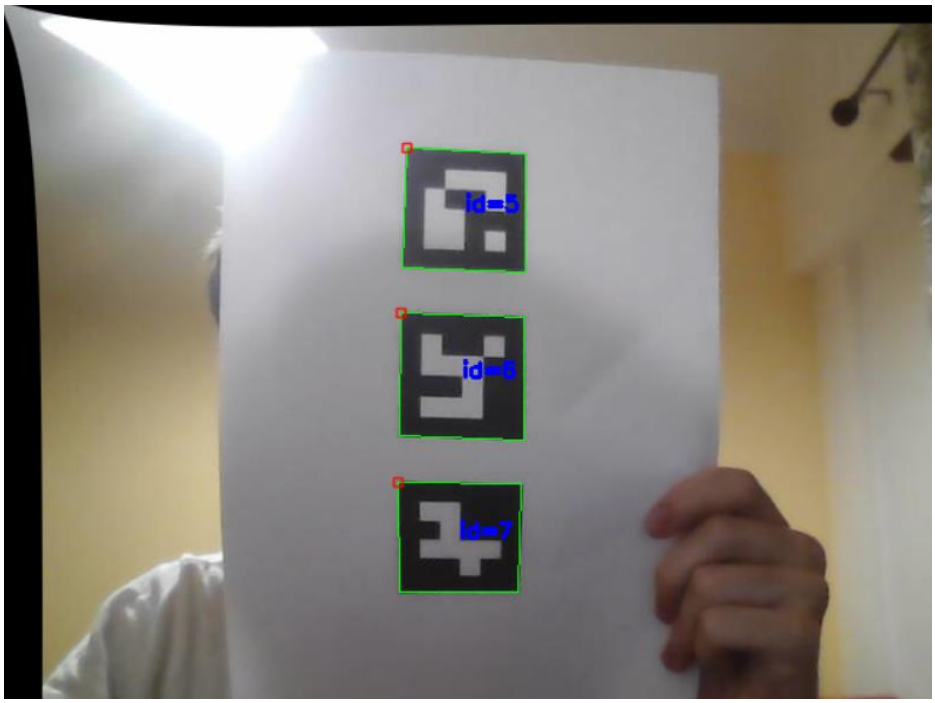

Figura 9 - Deteção dos marcadores, identificados

2.5. Realidade aumentada a partir dos marcadores

2.5.1 Realizar realidade aumentada de uma imagem a partir de um marcador

Neste momento, apôs a deteção dos marcadores, foi realizada a substituição dos marcadores ArUco detetados por outras imagens. Desta maneira é possível realizar o *overlay* de uma imagem individual em um sítio específico numa cena em tempo real.

Foi realizada uma função para realizar realidade aumentada nas imagens que se pretende que sejam substituídas pelos marcadores. Para realizar esta substituição é necessário inicialmente obter os quatros pontos dos cantos da imagem, a altura, comprimento e o *channel*. A partir desta informação é calculada a homografia, que retorna uma matriz. Esta matriz permite mapear os pontos da imagem aumentada para a imagem da cena em que se pretende substituir.

Para encontrar a homografia utiliza-se a função findHomography do OpenCV, que permite obter a função de homografia entre a origem (pontos dos cantos do marcador) e os pontos de destino (pontos dos cantos da imagem aumentada).

A matriz de homografia é agora usada para realizar um "*warp*" da nova imagem para caber nos pontos especificas onde se encontra o marcador.

Agora o *output* da imagem é apenas a imagem no marcador, mas com um fundo preto. Foi então necessário voltar a colocar o fundo.

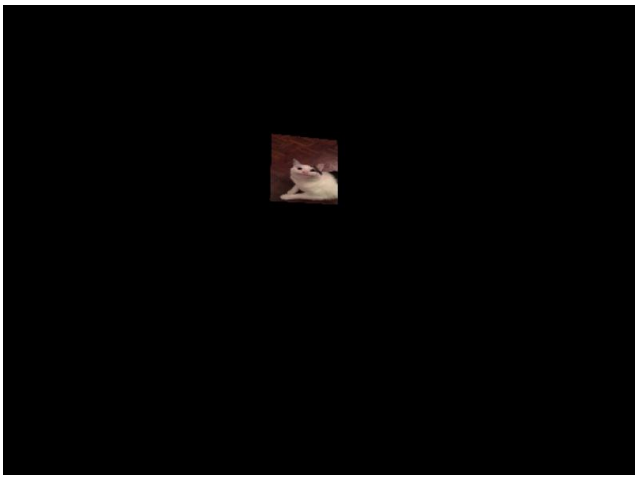

Figura 10 - Substituição apenas da imagem no marcador com fundo preto

Para isto todos os pixéis na área do marcador foram substituídos por pixéis pretos, utilizando a função fillConvexPoly do OpenCV, que permite preencher os pixéis de uma imagem dentro de um polígono convexo com uma determinada cor.

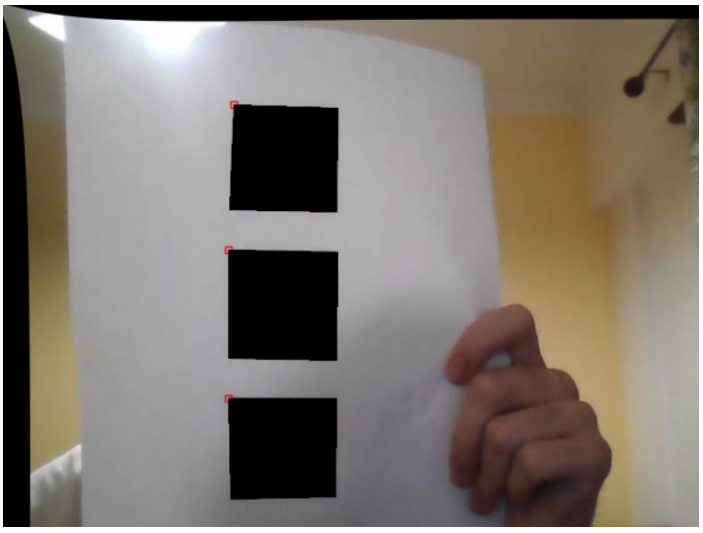

Figura 11 - Área dos marcadores preenchida a preto

Apôs ter os pixéis das áreas dos marcadores a preto é realizada uma junção desta imagem com a imagem aumentada, obtendo o *output* final.

Figura 12 - Output final para a substituição dos marcadores por imagens

2.5.2 Projeção de um cubo virtual a partir dos marcadores identificados

Nesta etapa foi realizada a projeção de objetos virtuais com o sistema de coordenadas do mundo associados com os marcadores identificados, na figura seguinte pode-se observar um exemplo disto.

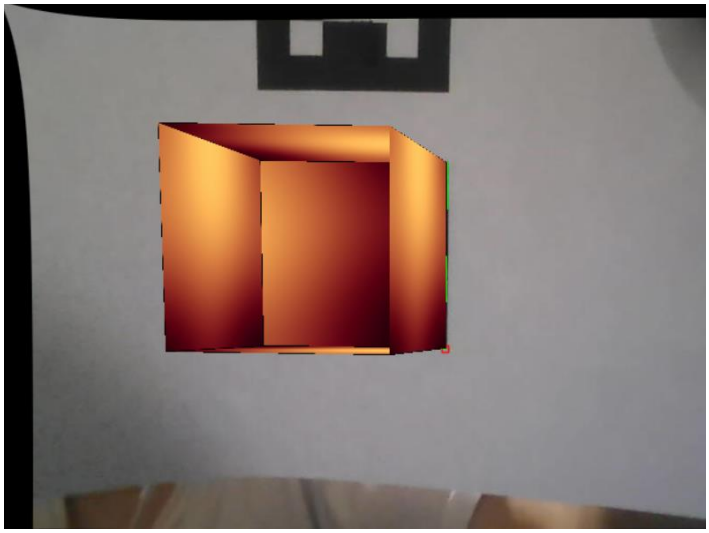

Figura 13 - Projeção de um objeto virtual (cubo) no marcador Aruco

A ideia base para projetar um objeto 3D para o plano de imagem é a seguinte, sabendo como um objeto se encontra situado no espaço, conhecendo parâmetros como a sua rotação e translação, podemos desenhar formas 2D para simular um efeito 3D. Sendo então possível projetar objetos como cubos no plano de imagem.

O problema que surge é o seguinte, como projetar as coordenadas 3D do mundo real para o plano de imagem 2D, para isto foram utilizadas duas funções do OpenCV, respetivamente:

- estimatePoseSingleMarkers – Permite obter a pose dos marcadores em relação à câmara.
- projectPoints – Permite projetar pontos 3D para o plano de imagem.

Inicialmente é necessário calcular a pose dos marcadores em relação à câmara, para ser possível projetar pontos 3D para o plano de imagem 2D. A função estimatePoseSingleMarkers do OpenCV recebe os marcadores detetados e retorna a estimação da pose dos marcadores em respeito à camara. Então para cada marcador é retornado um vetor de rotação e translação. A transformação retornada é a que transforma pontos do sistema de coordenadas de cada marcador para o sistema de coordenadas da câmara.

Tendo agora a estimação dos marcadores em relação à camara, de seguida são projetados os pontos 3D para o plano de imagem com função projectPoints do OpenCV. Esta função computa a projeção de pontos 3D para o plano de imagem dados os parâmetros intrínsecos e extrínsecos da camara, utilizando os vetores de rotação e translação obtidos a partir da função anterior e matriz da camara.

Por fim é necessário utilizar uma metodologia semelhante à utilizada anteriormente para adicionar uma imagem às faces dos objetos virtuais. É adicionada uma imagem nas coordenadas das faces, utilizando o mesmo funcionamento de obter a matriz de homografia e realizar o "*warp*" da nova imagem para caber nos pontos especificas onde se encontra o marcador.

O primeiro objeto virtual a ser projetado foi um cubo. Inicialmente foi necessário definir quatro pontos para cada uma das 6 faces do cubo. O funcionamento foi o seguinte:

- Definir as 6 faces do cubo, criando um numpy *array* com 4 pontos para cada uma das 6 faces.
- Obter a pose dos marcadores com a função estimatePoseSingleMarkers.
- Projetar pontos 3D para o plano de imagem com a função projectPoints.
- Substituir as faces do cubo por uma imagem aumentada.

2.5.3 Resultados experimentais

Os resultados obtidos foram os seguintes:

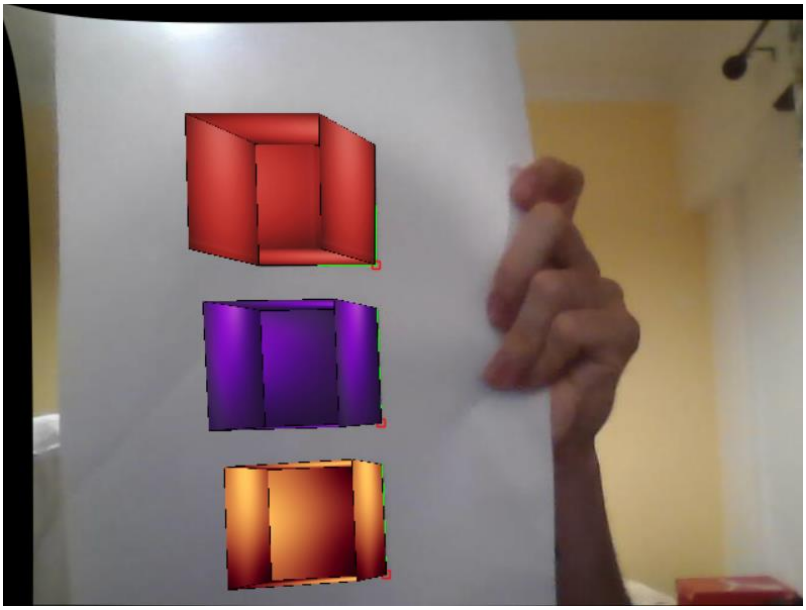

Figura 14 - Projeção de objetos virtuais nos marcadores consoante os identificadores

A partir das coordenadas para os pontos das faces, foram também criados cubos a partir de linhas utilizando o método line do OpenCV, que permite desenhar uma linha consoante dois pontos.

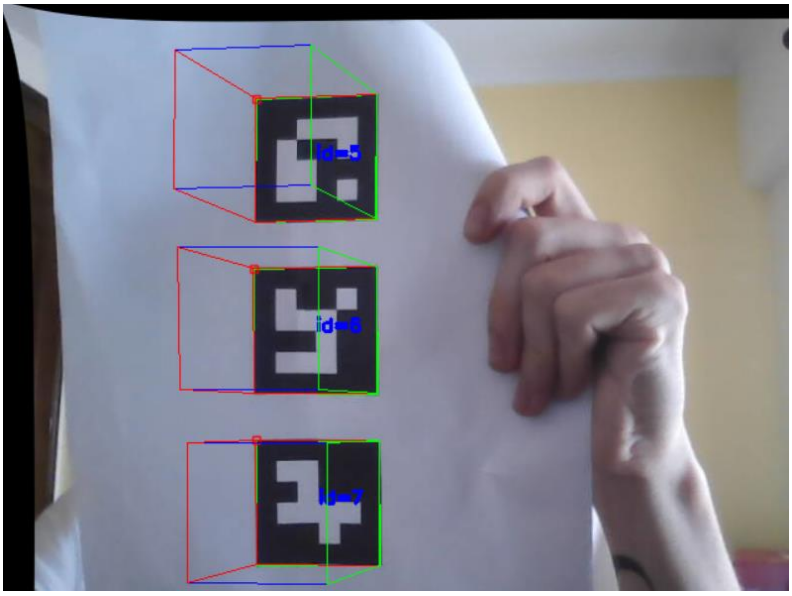

Figura 15 - Projeção de cubos virtuais a partir de linhas

De seguida foi também criada uma pirâmide a partir de linhas, utilizando o mesmo método.

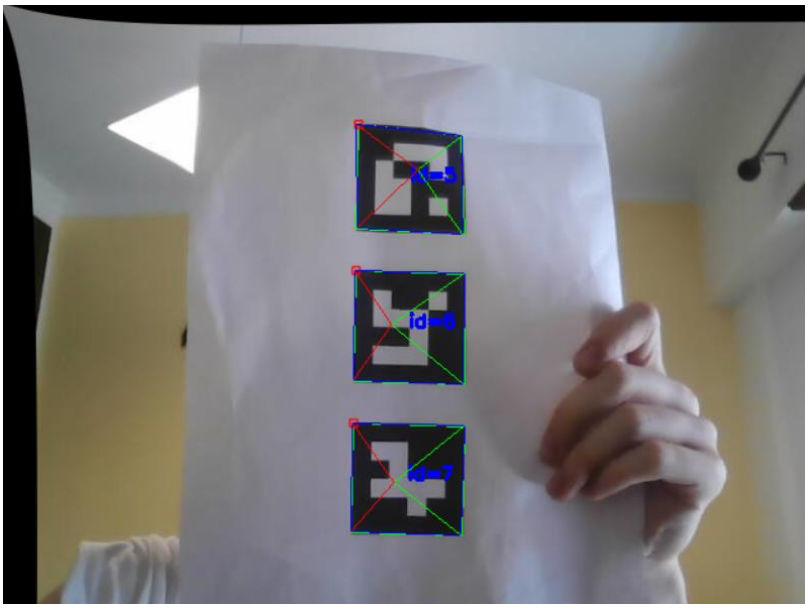

Figura 16 - Projeção de pirâmides virtuais a partir de linhas

Na figura seguinte pode-se observar os diferentes objetos virtuais criados, projetados consoante o identificador do marcador.

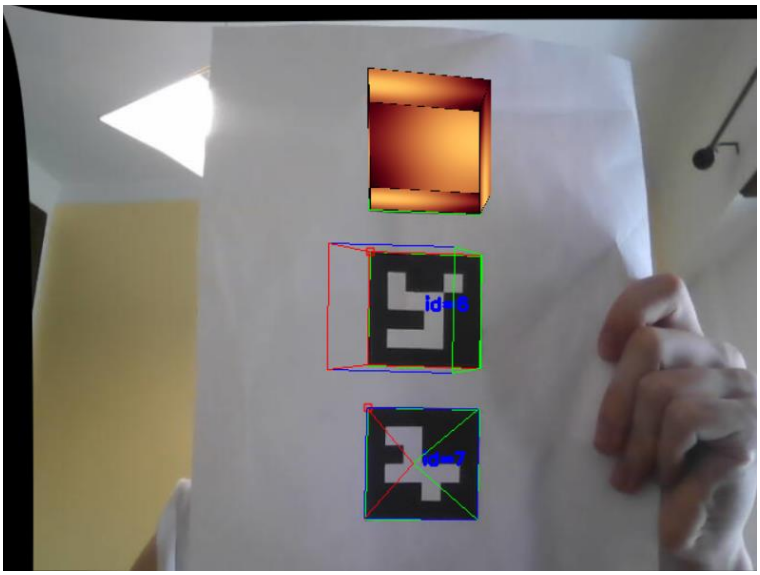

Figura 17 - Resultados finais obtidos

3. Conclusões

A partir da realiação deste trabalho prático foi possivél aplicar e estudar diferentes métodos e conceitos relacionados com realidade aumentada baseada em marcadores. No trabalho prático foi realizada com sucesso uma aplicação que permite a inclusão de objetos virtuais especificos consoante um identificador a partir de marcadores fiduciais, neste caso marcadores ArUco.

Uma das áreas abordadas neste projeto foi a importância da calibração da câmara. A calibração da câmara foi crucial para a elaboração do projeto, de forma a ser possível realizar a não distorção das imagens obtidas pela câmara do computador e a obtenção dos parâmetros intrisecos e exstrinsecos da câmara. Para realizar esta etapa utilizou-se o OpenCV devido à sua fidebilidade e experiência prévia com esta aplicação, porém poderiam ter sido utilizadas outras bibliotecas como por exemplo, a bilbioteca ArUco [5].

Na área da estimação e deteção da pose da câmara foi estudada a importância dos marcadores ArUco e como estes são utilizados em aplicações de *computer vision* e realidade aumentada. No processo de geração dos marcadores, foram abordadas duas alternativas, gerar marcador AruCo a partir de um Website, disponível em [4] ou a partir do Python utilizando OpenCV. A alternativa que acabou por ser usada foi a geração dos marcadores a partir do OpenCV, visto que é possível ter um maior controlo sobre a criação dos marcadores, mas ambas as alternativas eram válidas.

Finalmente na etapa final de realizar realidade aumentada de uma imagem a partir de um marcador conseguiu-se adicionar diferentes imagens aos marcadores corretamente, e a adição de diferentes objetos virtuais, neste caso cubos e pirâmides, consoante o respetivo identificador do marcador.

Um problema a mencionar no trabalho prático que não foi resolvido, é que na renderização dos cubos com imagens por vezes ocorrem problemas vizuais em que diferentes cores não desejadas começam a aparecer na imagem.

4. Bibliografia

[1] -Prof. Arnaldo Abrantes, Prof. Pedro Mendes Jorge, Visão Artificial e Realidade Mista, Geometria Projetiva

[2] - https://machinelearningknowledge.ai/augmented-reality-using-aruco-marker-detection-with-python-opencv/

[3] - https://docs.opencv.org/4.x/d7/d53/tutorial_py_pose.html

[4] - https://chev.me/arucogen/

[5] - https://www.uco.es/investiga/grupos/ava/node/26

[6] - https://learnopencv.com/geometry-of-image-formation/