DDETC – Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

MEIM - Mestrado Engenharia informática e multimédia

Visão Artificial e Realidade Mista

Trabalho prático 2

Turma:

MEIM-2N

Trabalho realizado por:

Miguel Távora Nº45102

Docente:

Pedro Jorge

Data: 12/06/2022

Índice

Índice ilustrações

1. Introdução

O segundo trabalho prático da unidade curricular de Visão Artificial e Realidade Mista tem como objetivo implementar um programa que realiza a deteção de marcas para inclusão de objetos virtuais tridimensionais. Os objetos virtuais têm de estar alinhados com a realidade nomeadamente com as marcas.

A realidade mista é a fusão do mundo real e virtual para produzir novos ambientes e visualizações, onde objetos físicos e digitais coexistem e interagem em tempo real. Por isso a realidade mista não ocorre exclusivamente no mundo físico ou virtual, ocorre nos dois simultaneamente.

Para a implementação da deteção das marcas foi utilizada a biblioteca ArUco incluído na biblioteca *open-source* para *computer vision* OpenCV. Esta biblioteca possuí a funcionalidade de estimação de pose que possui diversas funcionalidades em diversas áreas. Este processo geralmente é um passo difícil e por isso é comum utilizar marcadores sintéticos para o facilitar. Os marcadores mais utilizados são geralmente quadrados binários por possuir vantagens na estimação de pose da camara, possibilidade de aplicação técnicas de deteção e correção de erros.

2. Desenvolvimento

1. Calibração da câmara

No desenvolvimento do trabalho, primeiramente foi implementado a calibração da câmara. A calibração da câmara tem o objetivo de estimar os parâmetros de uma lente e sensor de imagem. Esta informação pode ser utilizada para corrigir a distorção da lente, medir o tamanho de um objeto e determinar a localização da câmara na cena. No caso do trabalho prático a calibração da câmara será utilizada para corrigir a distorção da lente. Uma distorção na lente corresponde a um desvio da projeção retilínea, onde linhas direitas numa cena deveriam corresponder a linhas direitas na imagem.

Para isso foi utilizado o código fornecido na documentação do OpenCV de calibração da câmara para obter os parâmetros intrínsecos e extrínsecos da calibração. Para isso foi necessário o telemóvel com uma foto de um tabuleiro de xadrez e tirar diversas fotos com várias inclinações e em múltiplas posições de forma a obter uma boa calibração. Para obter as fotos foi criado um *script* para quando se clica numa tecla ele tira uma foto. No total foram utilizadas treze fotos diferentes. Um dos resultados obtidos pela calibração da câmara foi o que se segue:

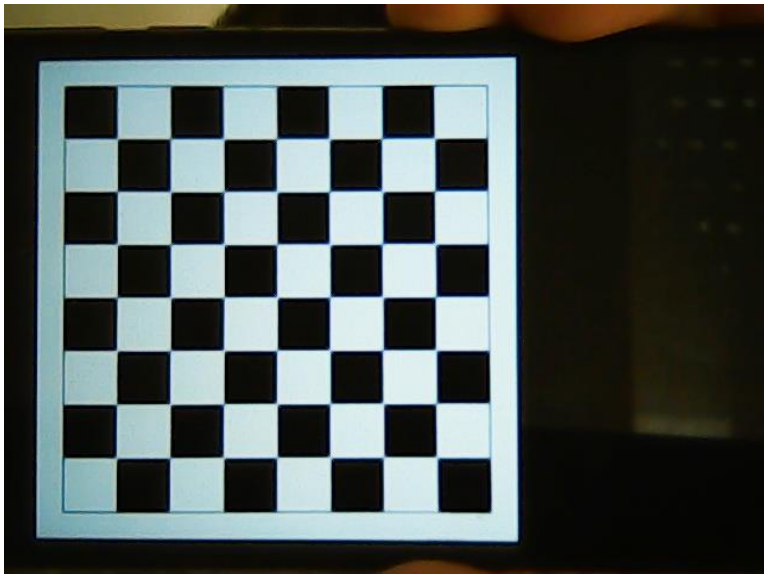

Figura 1 - imagem original

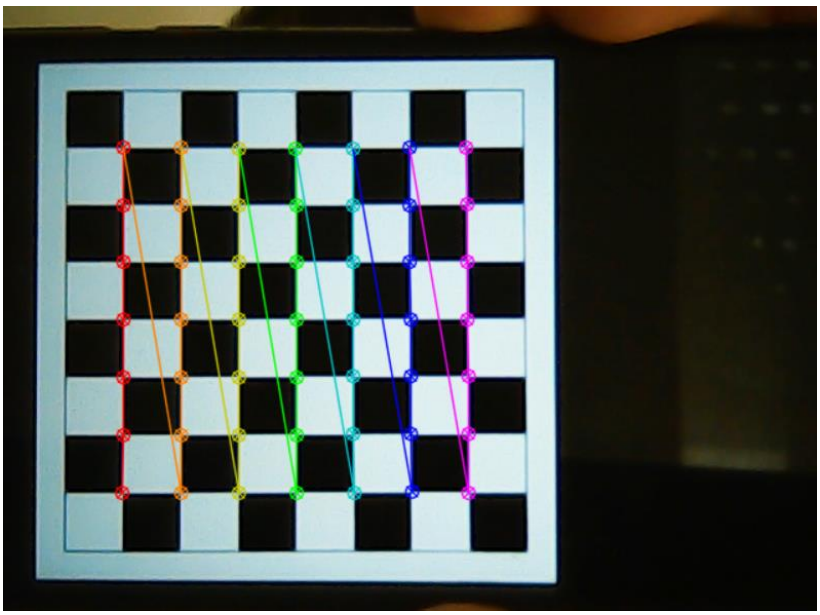

Figura 2 - imagem após calculados os cantos do xadrez

A partir dos valores obtidos na calibração é possível utilizar o método *undistort* do OpenCV para obter a imagem sem distorção.

2. Deteção de marcadores

Após feita a calibração da câmara é necessário detetar os marcadores. Um marcador é quadrado composto por uma larga borda preta e uma matriz binária interna que determina o seu identificador (id).

Figura 3 - exemplo de marcador

A borda preta facilita a deteção rápida e a matriz binária serve para identificação e aplicação de técnicas de deteção e correção de erros. Para as marcas serem detetadas é necessário primeiro serem criadas e serem postas no ambiente(cena). O módulo aruco possui uma função para geração destas marcas designada *drawMarker*, contudo não foi utilizada função pois foi utilizado imagens com identificadores pré-definidos.

Primeiramente para ser possível detetar um marcador é necessário possuir algum dispositivo ou papel com um identificador e apostá-lo para a câmara. No caso do trabalho foi sempre utilizado o telemóvel. Para realizar a deteção foi utilizada a função *detectMarkers* do módulo aruco do OpenCV. Esta função recebe como argumentos a imagem para detetar os marcadores, um dicionário que indica o tipo de marcadores que serão procurados, que no caso do trabalho foi 6x6_250 sendo 6x6 bits e a distância mínima de hamming entre dois códigos é 11 obtendo no máximo 250 códigos distintos. A função retorna então os cantos do quadrado detetado e também os seus ids. Contudo esta função não realiza estimação de pose.

Após detetados os marcadores o módulo aruco também fornece uma função para desenhar os ids e os retângulos em volta dos marcadores detetados. Os resultados foram os que se segue:

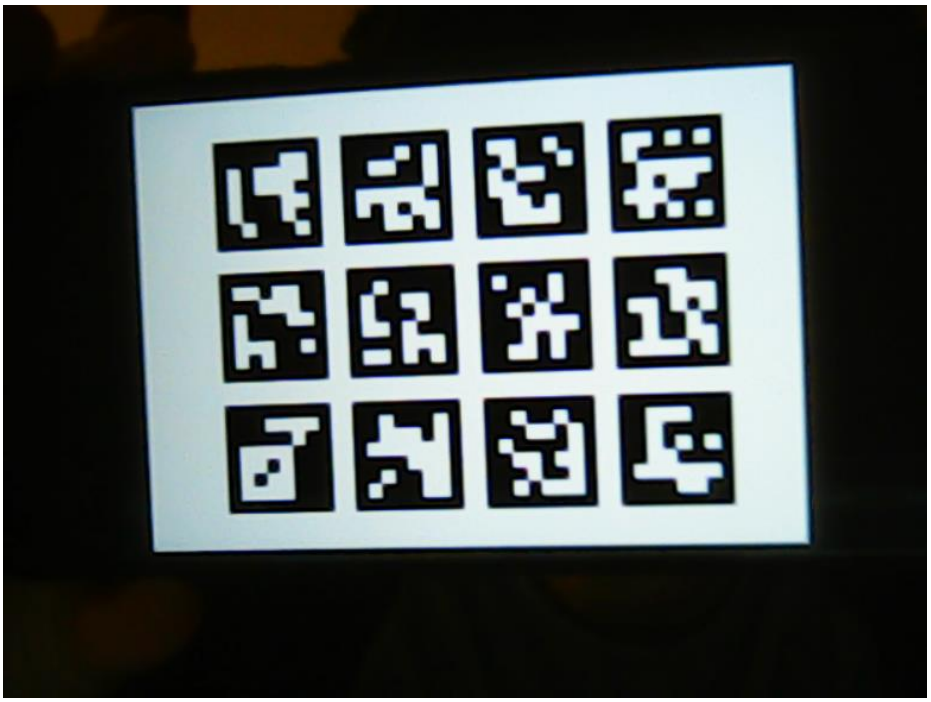

Figura 4 - imagem original

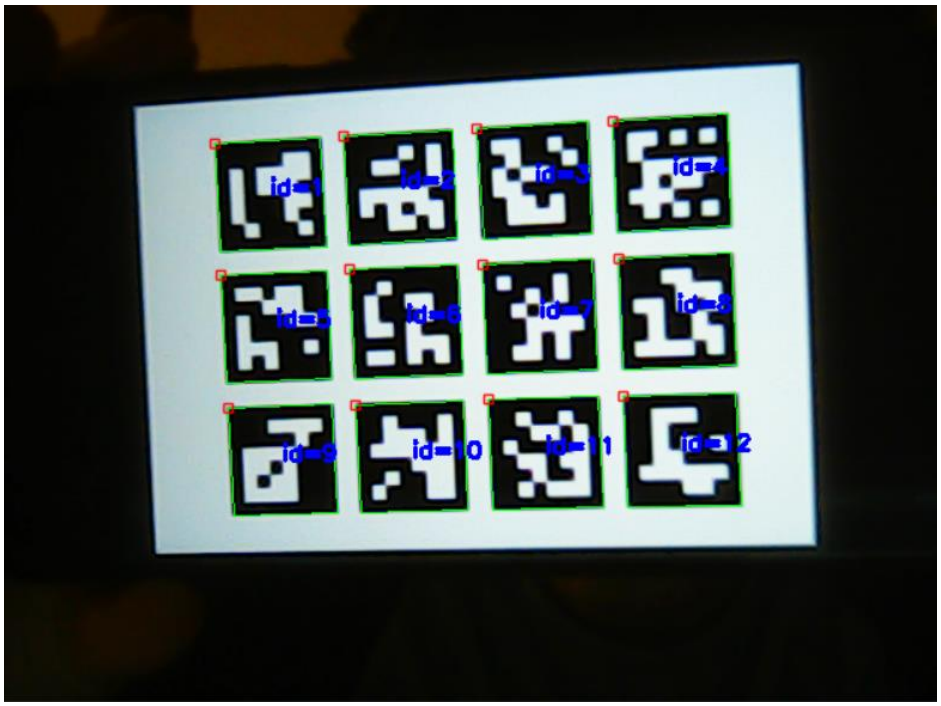

Figura 5 - imagem com deteção dos marcadores

3. Estimação de pose

A estimação de pose 3D é um processo de predição da transformação de um objeto de uma pose de referência feita pelo utilizador. Esta técnica permite estimar rotações e translações em três dimensões somente a partir de uma foto bidimensional, isto se for conhecido um modelo aproximado do objeto 3D e os pontos correspondentes numa imagem 2D.

Em termos de implementação como foi referido anteriormente a função *detectMarkers* não calcula a estimação de pose. Contudo existe uma outra função que realiza a estimação de pose designada *estimatePoseSingleMarkers*. Esta função recebe os marcadores detetados e os coeficientes de distorção obtido na calibração da câmara e retorna a estimação da pose relativamente com a câmara. Então por cada marcador é retornado um vetor com a rotação e outro com a translação. O sistema de coordenadas do marcador é no centro com o eixo do Z perpendicular ao plano de x, y.

Além desta função foi também utilizada uma função designada *drawAxis* para pintar os eixos na imagem e confirmar se o cálculo da pose estava a ser calculada corretamente. Os resultados foram os que se segue:

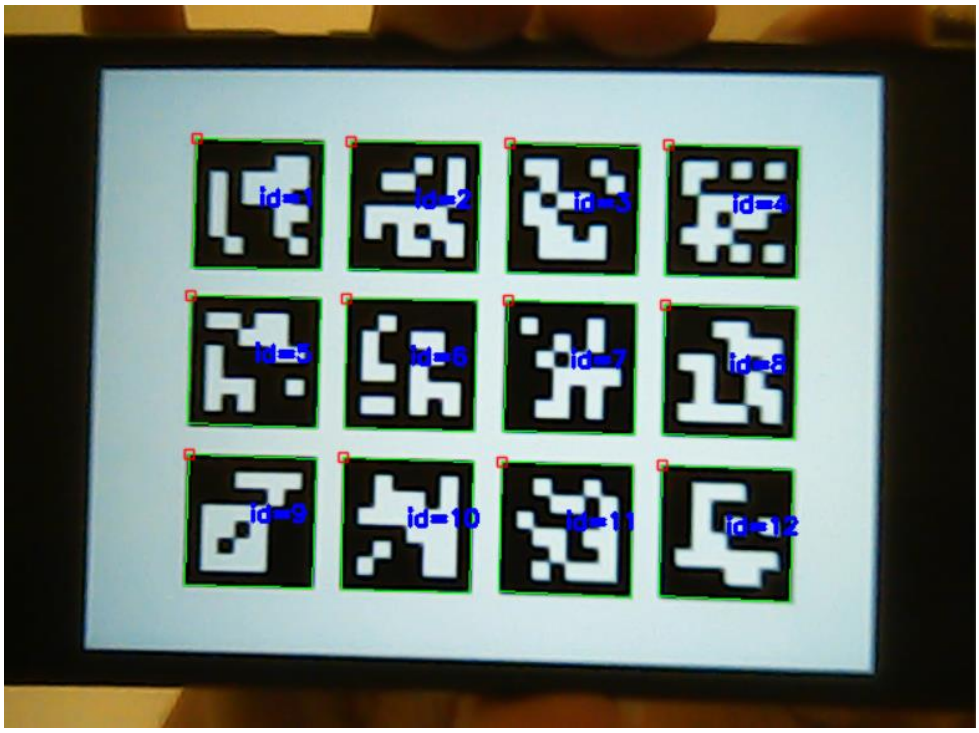

Figura 6 - deteção dos marcadores

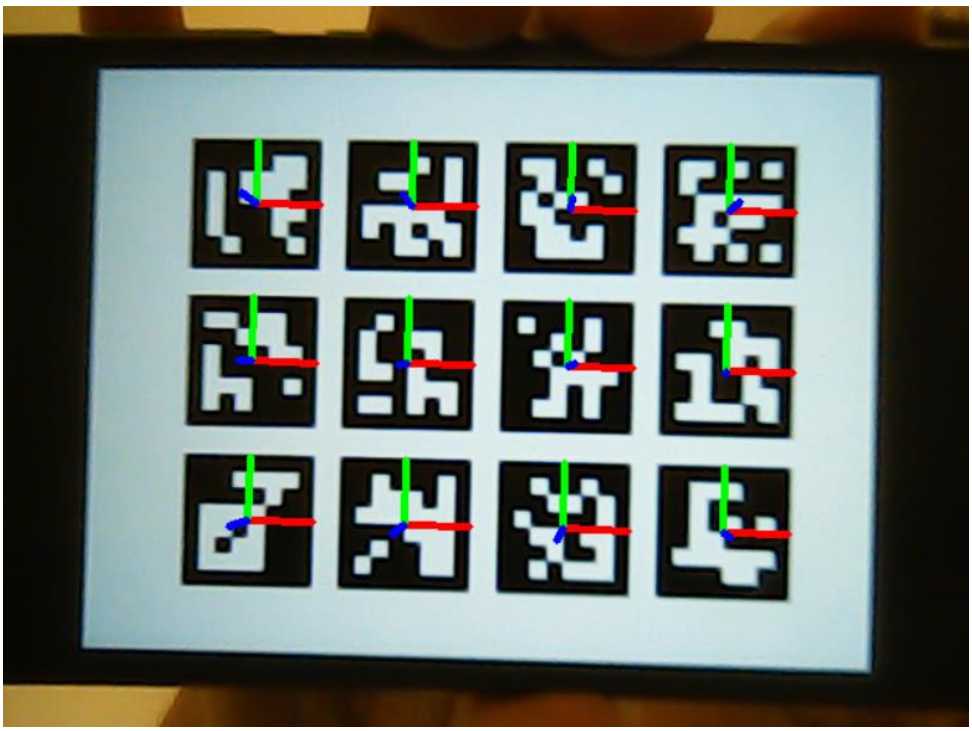

Figura 7 - estimação de pose dos marcadores

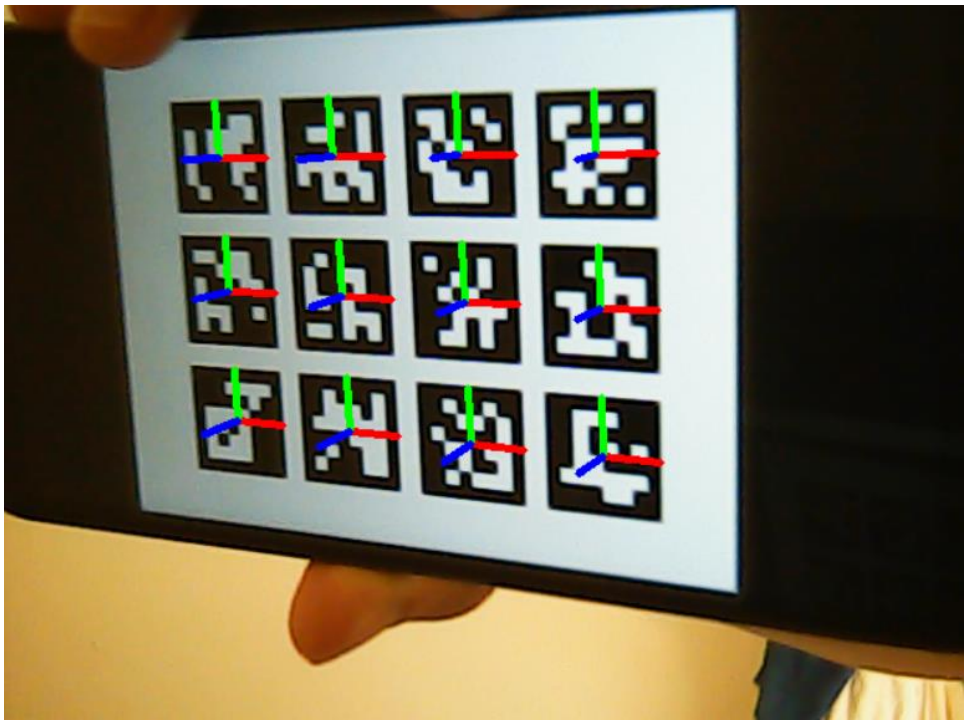

Figura 8 - estimação de pose dos marcadores rodado

De notar que o eixo vermelho corresponde ao eixo do X, o verde ao eixo do Y e o azul ao eixo do Z.

4. Aplicação de objeto 2D

Antes de construir o objeto foi testado primeiro por somente uma imagem no lugar do marcador. Para conseguir isto foi utilizado uma técnica de homografia que a partir de uma matriz de 3x3 mapeia os pontos de uma imagem para os pontos noutra imagem. Uma característica destas técnicas é que quando a câmara se move, mas o centro ótico é mantido fixo, os pontos em correspondência nas imagens estão relacionados através de transformações projetivas planas.

De seguida é utilizada a função *warpPerspective* que aplica uma transformação de perspetiva a uma imagem, que neste caso é aplicar a matriz obtida na homografia na imagem que se pretende adicionar. Por fim utiliza-se a função *fillConvexPoly* para se criar um buraco preto na imagem original onde será posta a imagem e junta-se as duas imagens.

Adicionalmente foi criado um método que por cada identificador é atribuído uma imagem diferente, mas constante durante toda a execução do programa. Os resultados foram os que se segue:

Figura 9 - imagem com deteção de marcas

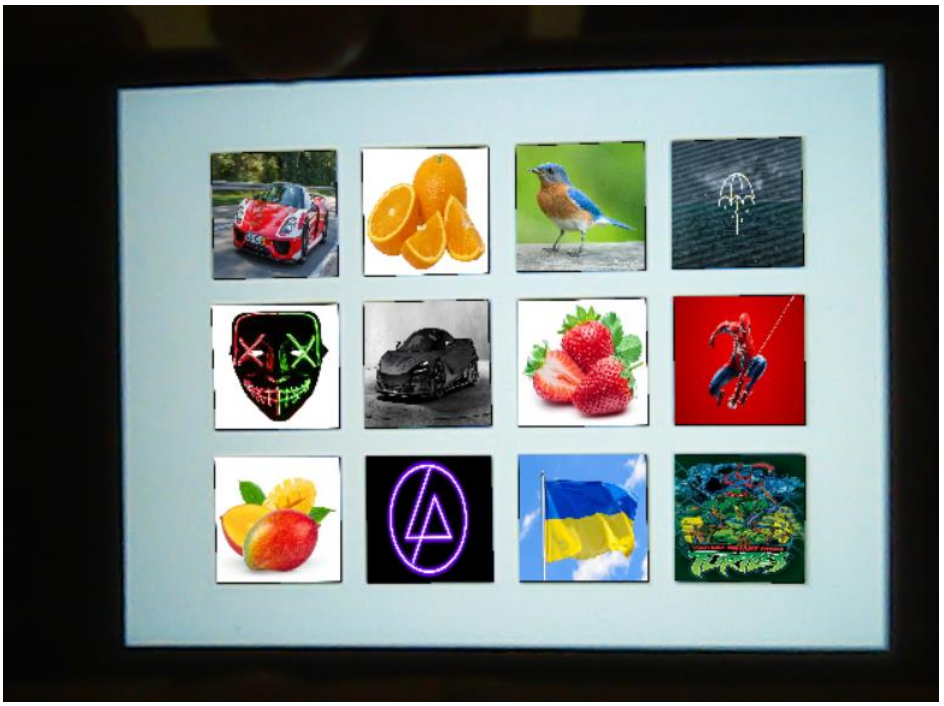

Figura 10 - imagem após aplicação das imagens nos marcadores

Adicionalmente na deteção dos marcadores foi criado um método para pôr sempre as imagens com a rotação correta. Isto deve-se ao facto de algumas imagens não possuírem o seu ponto de referência como o canto superior esquerdo. Por isso são feitos cálculos sobre as coordenadas obtendo sempre os cantos corretamente e conseguindo sempre por a imagem com a rotação correta. Os resultados foram os que se segue:

Figura 11 - imagem sem aplicação do método

Figura 12 - imagem após aplicação do método

5. Aplicação de objeto 3D

Para conseguir criar um objeto tridimensional é necessário conseguir realizar a projeção de coordenadas tridimensionais para bidimensionais para ser possível obter no plano da imagem. Os pontos bidimensionais obtidos serão posteriormente utilizados para construir o objeto. A função que permite realizar esta operação é *projectPoints* da biblioteca do OpenCV. A função calcula as projeções 2D de pontos 3D para o plano da imagem, dados os parâmetros intrínsecos e extrínsecos da câmara (vetor rotação, vetor translação, matriz intrínseca A, coeficientes de distorção).

A partir dos pontos obtidos pela função anterior é possível representar um cubo. Inicialmente foi feito um cubo somente com a função line do OpenCV e os resultados foram os que se segue:

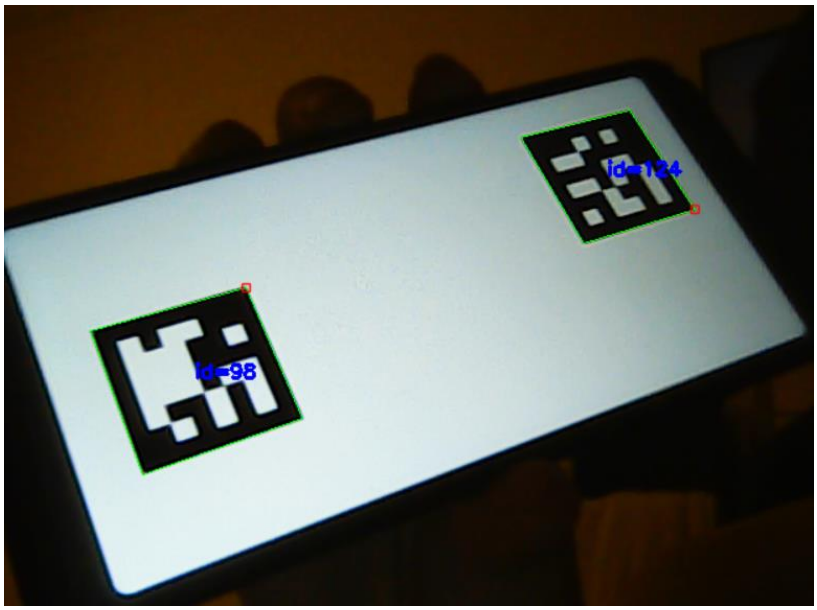

Figura 13 - imagem com marcador

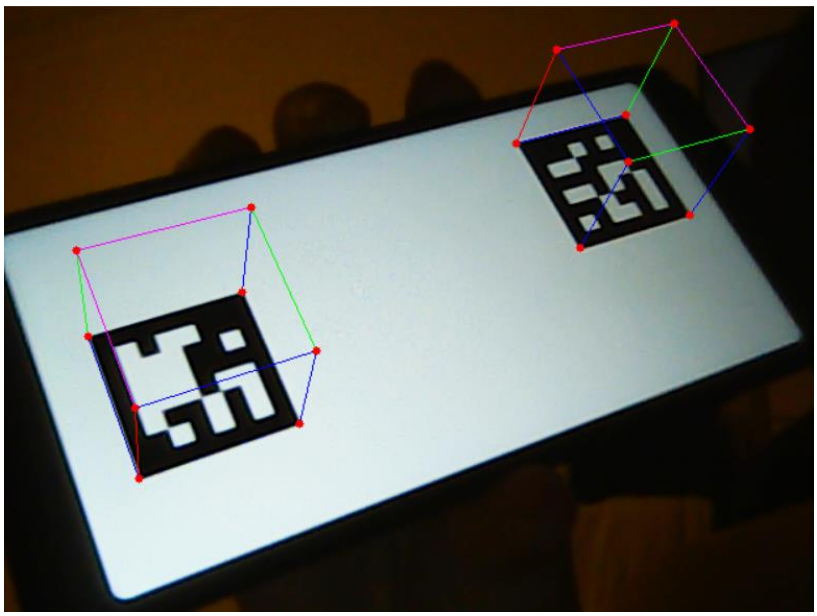

Figura 14 - desenho dos vértices e arestas do cubo a partir dos pontos projetados

A partir dos pontos obtidos para realizar o cubo exibido anteriormente é possível aplicar imagens nas faces do cubo. Para pintar as imagens nas faces foi utilizada a mesma metodologia utilizada para pôr a imagem no marcador (ponto 4.). Adicionalmente foi necessário fazer alguns cálculos sobre os pixéis para saber quais as faces que devem ser renderizadas primeiro e as que devem ser em último. Isto pelo facto de produzir um efeito estranho na altura da renderização. Contudo existem alguns ângulos que não foi possível resolver este problema, mas foi melhorado muito comparativamente a não realizar nenhum cálculo. Os resultados obtidos foram os que se segue:

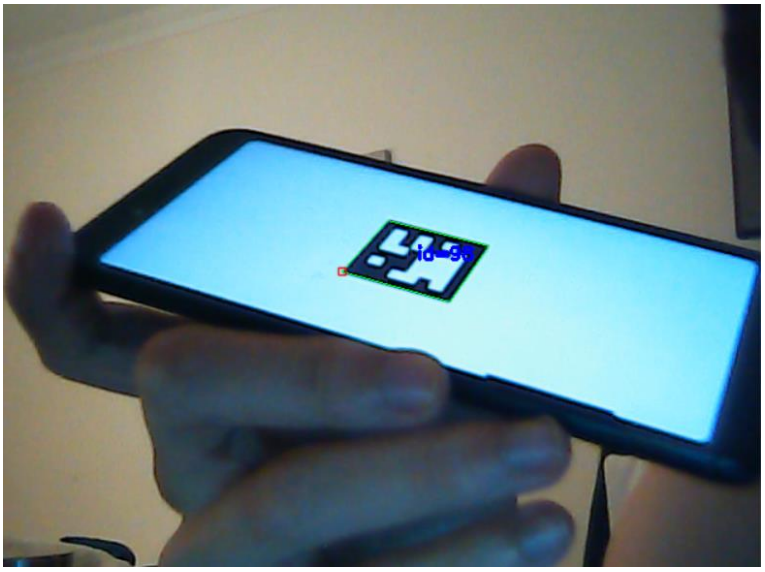

Figura 15 - imagem com marcador

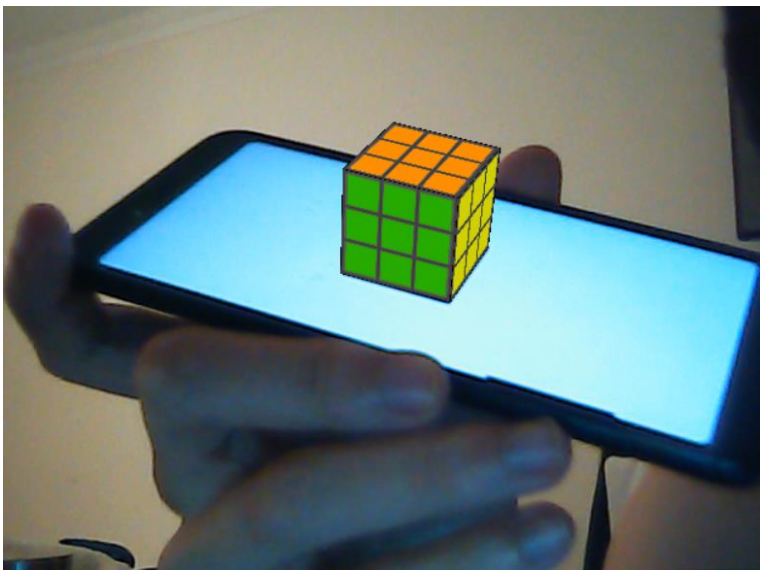

Figura 16 - inserção do objeto 3d

Foram utilizadas mais do que um conjunto de imagens para as faces do cubo, no total foram utilizados três conjuntos de imagens para renderizar as faces dos cubos. Os resultados para vários cubos foram os que se segue:

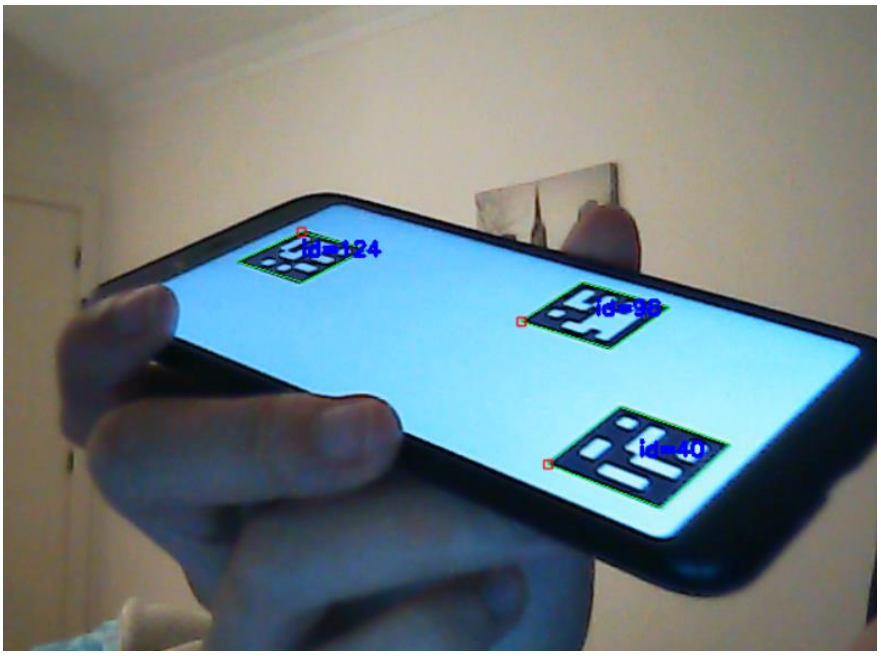

Figura 17 - imagem com marcadores

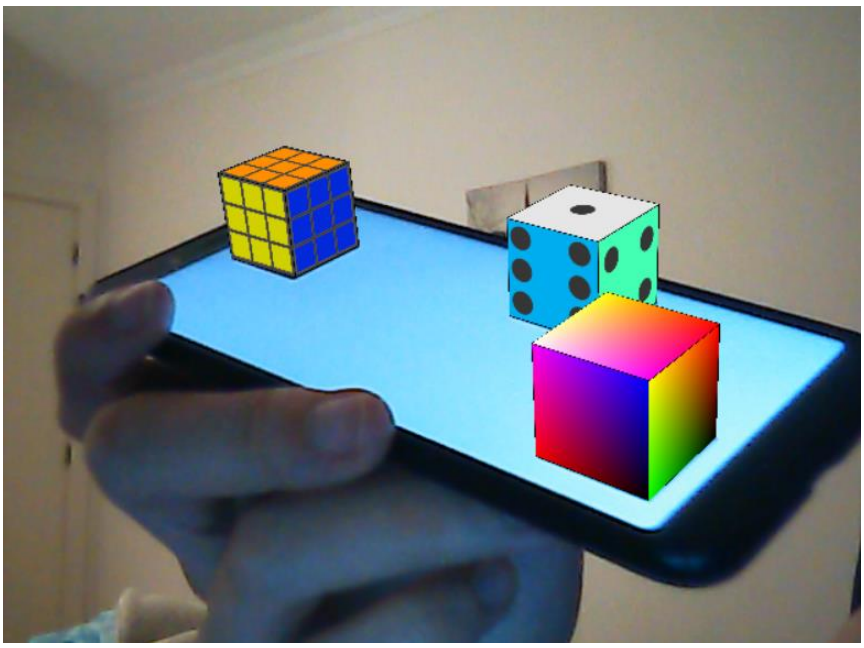

Figura 18 - imagem com os diferentes objetos para diferentes identificadores

6. Z-Buffer

O Z-Buffer tem como objetivo quando dois pontos estão sobreposto ele exibe o que se encontra mais próximo da câmara. Para isso primeiramente é preciso saber as coordenadas do ponto relativamente à câmara. Para isso primeiramente foi utilizado a matriz de posição calculada na calibração da câmara. Contudo é preciso uma matriz de rotação e o que é obtido na calibração da câmara é um vetor de rotação. Para converter esse vetor para matriz é utilizado a função Rodrigues do OpenCV.

De seguida é necessário realizar a seguinte conta: $Ponto_{reltivo\ câmara} = [R \mid T] . \begin{bmatrix} x \\ y \\ z \end{bmatrix}$. Na fórmula anterior o R representa a matriz de rotação e T a matriz da posição e os valores x, y e z são as coordenadas do ponto no mundo. Desta forma é possível obter o ponto relativamente à câmara, com coordenadas x, y e z. Por fim é necessário verificar se a coordenada está dentro da imagem e caso o seu valor z (distância à câmara) seja menor que outro ponto ele desenha o

ponto sobreposto. Os resultados foram os que se segue:

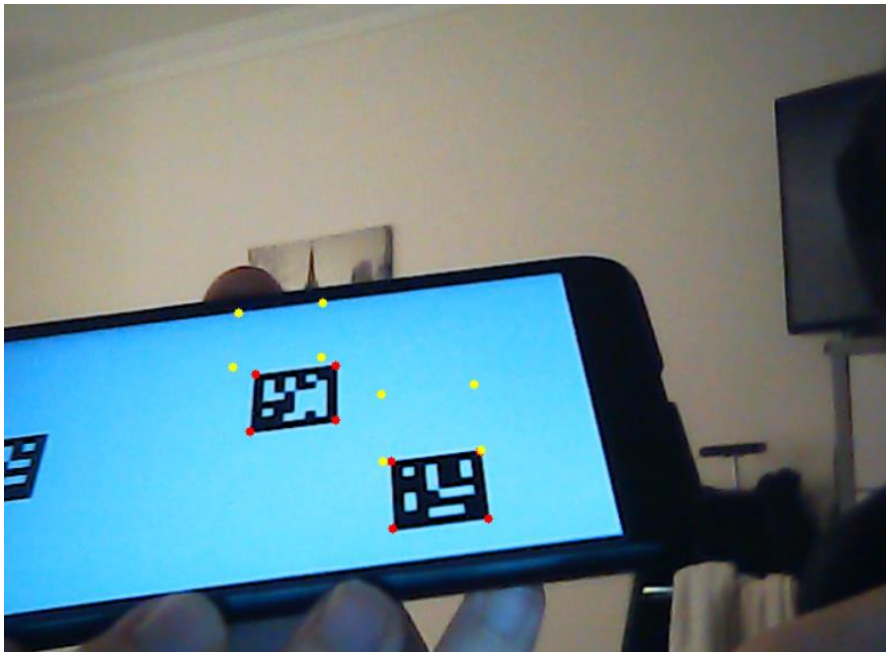

Figura 19 - resultado de sobreposições dos pontos

7. Resultados experimentais

Em termos dos resultados obtidos, o trabalho ficou todo ele a funcionar.

Exceto em algumas rotações do marcador em que as faces visíveis são as faces da parte de trás do cubo, fazendo um afeito visualmente estranho. Mas é pouco frequente este acontecimento.

Um outro problema é o facto de quando a face é muito estreita a imagem é distorcida fazendo um efeito visual estranho por toda a imagem. Para resolver este problema foi tentado não incluir a imagem na face quando a face era mais estreita que 2 pixéis em altura ou largura. Contudo este problema persistia e acabou por não se conseguir resolver pois é complicado replicar este problema.

Os resultados obtidos sequencialmente foram os que se segue:

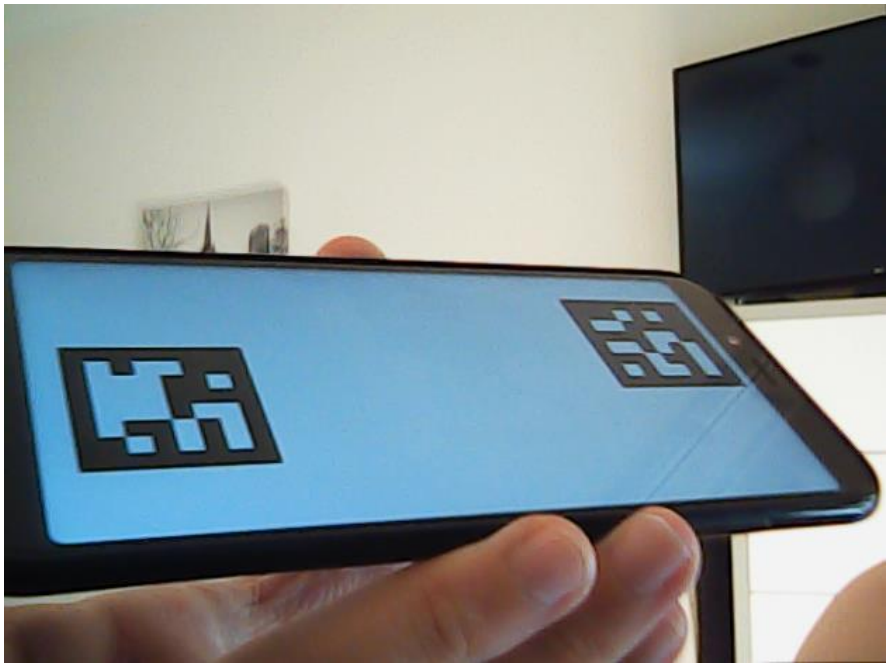

Figura 20 - imagem com calibração da câmara

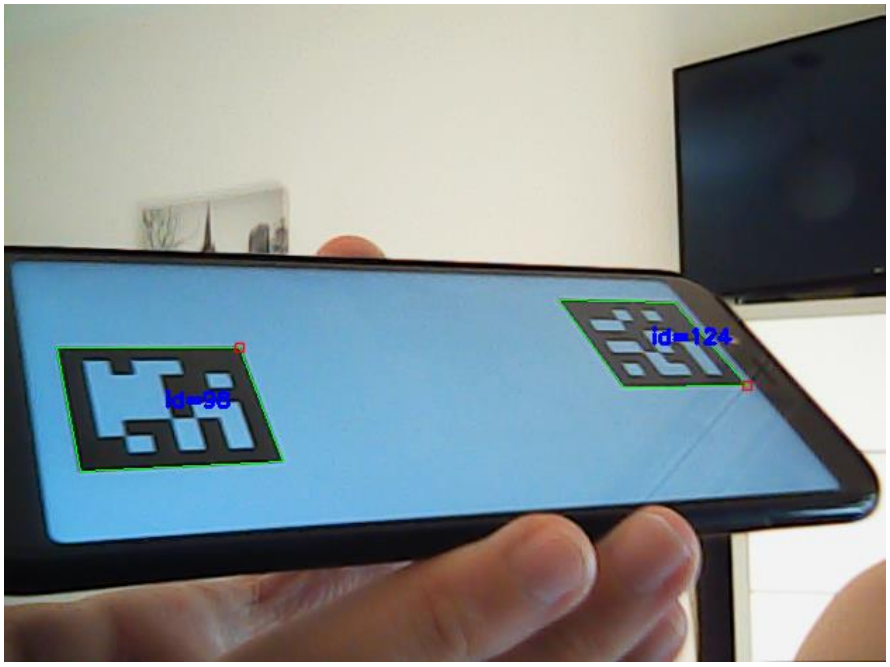

Figura 21 - imagem com a deteção dos marcadores

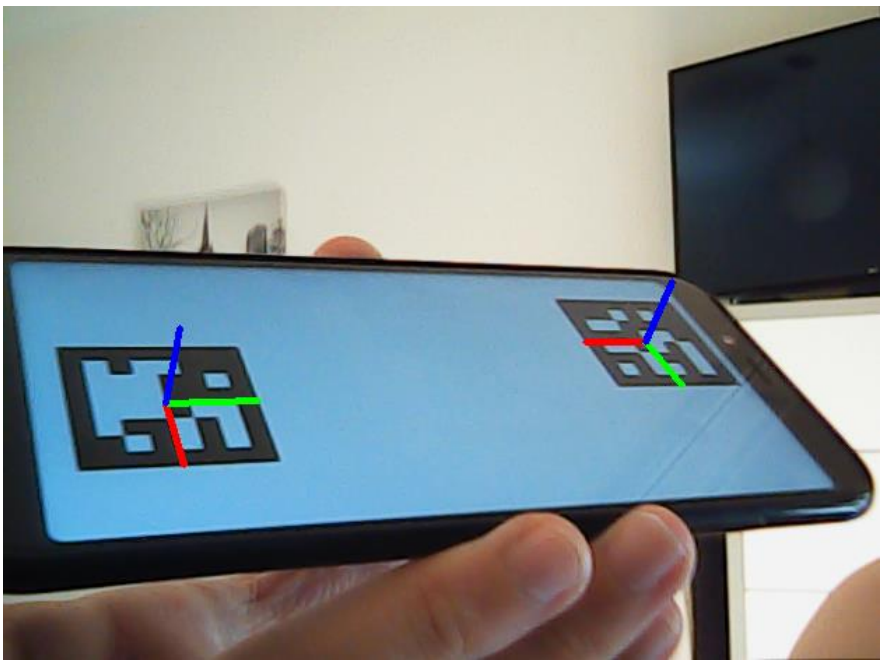

Figura 22 - imagem da estimação de pose do marcador

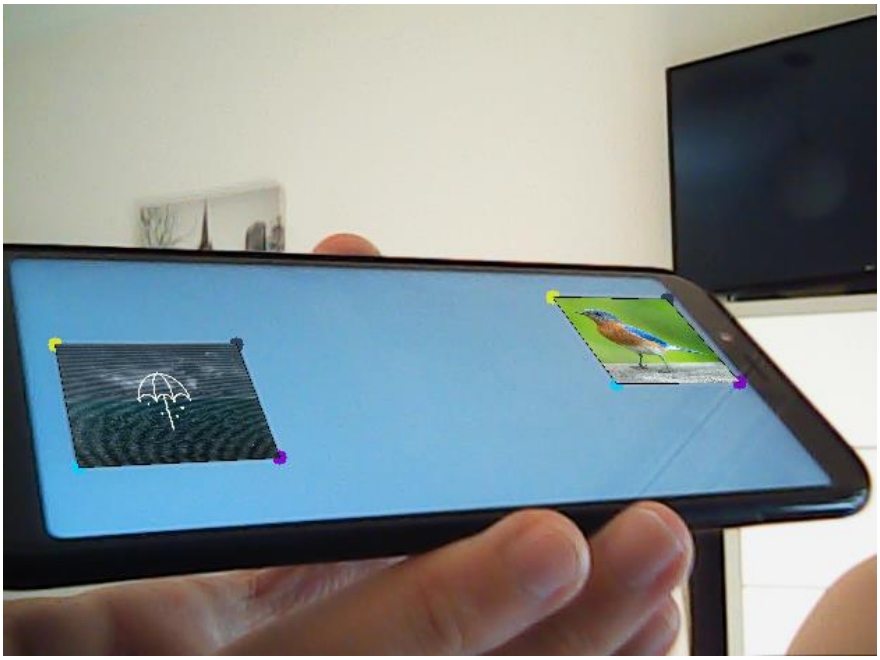

Figura 23 - aplicação de uma imagem no marcador através de homografia

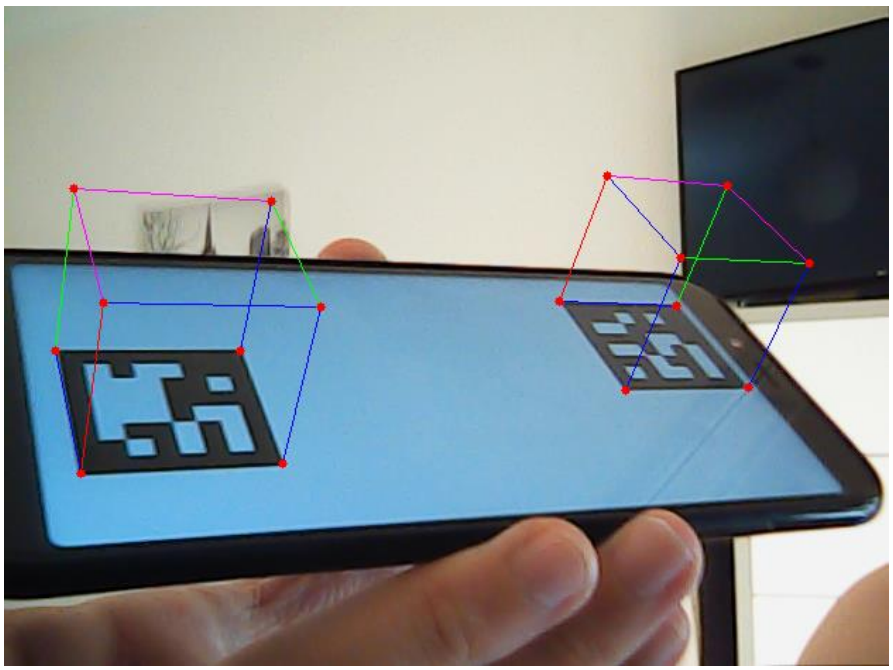

Figura 24 - construção do cubo com linhas e pontos

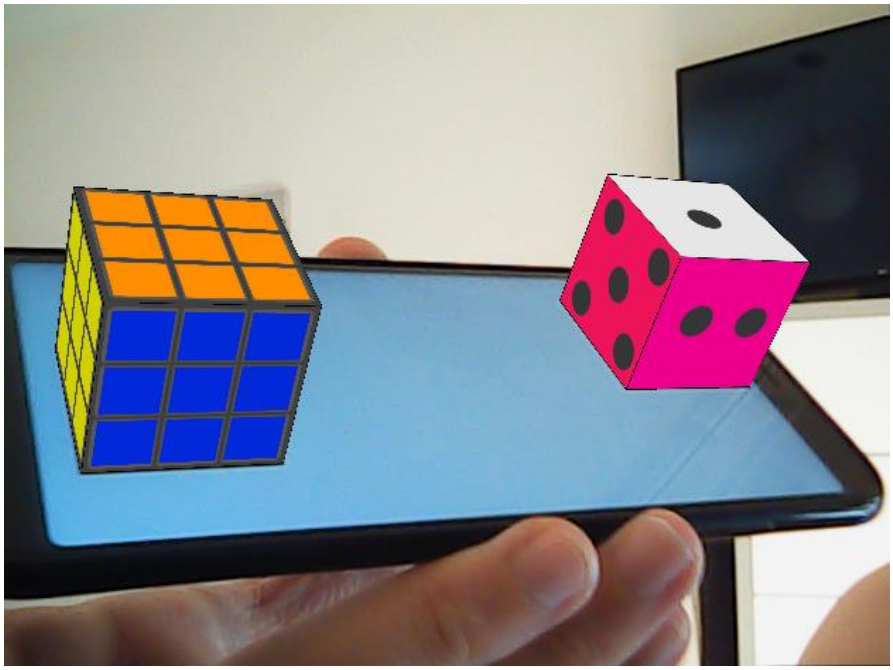

Figura 25 - construção dos cubos com imagens

3.Conclusões

Com a realização deste trabalho prático foi possível desenvolver conhecimentos na área da realidade mista e como adicionar objetos tridimensionais a uma cena.

Os conhecimentos adquiridos foram nomeadamente:

- Realização da calibração da câmara para obter coeficientes que permitem remover a distorção das lentes da câmara.
- Utilização de marcadores e as suas vantagens para conseguir realizar a deteção do local onde será aplicado a realidade aumentada.
- Necessidade do cálculo da estimação de pose para conseguir obter a posição e rotação do objeto para conseguir adicionar o objeto virtual de acordo com o mundo real.
- Como aplicar uma imagem no local do marcador detetado, através da homografia das imagens.
- Utilização da projeção de pontos para conseguir construir objetos 3D em imagens 2D.
- Utilização dos pontos projetados para conseguir construir o objeto na imagem.
- Aplicação de imagens a objetos 3D numa imagem.
- Utilidade da existência de identificadores nos marcadores.

4. Bibliografia

- https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
- https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html
- https://docs.opencv.org/4.x/d7/d53/tutorial_py_pose.html