Mestrado Engenharia Informática e Multimédia

Visão Artífical e Realidade Mista

Trabalho 2

Data: July 21, 2023

Docente:
Eng. Pedro Jorge

Realizado por:
Mihail Ababii - 46435

Índice de Conteúdos

Lista de Figuras

1 Introdução

Este projeto tem como objetivo desenvolver uma aplicação de visão computacional que utiliza realidade aumentada por marcadores. Com o auxílio da biblioteca OpenCV do Python, teremos acesso a funções para o processamento de imagens. Essa aplicação permitirá alinhar objetos virtuais aos marcadores presentes no mundo real, criando uma experiência imersiva e interativa.

Através da deteção e identificação dos marcadores, será possível sobrepor os objetos virtuais em tempo real, proporcionando uma experiência envolvente de realidade aumentada. Com este projeto, esperamos explorar todo o potencial da visão computacional e da realidade aumentada baseada em marcadores.

A Realidade Aumentada baseada em marcadores é uma tecnologia que utiliza marcadores físicos como referência para sobrepor objetos virtuais no mundo real. Os marcadores são imagens ou padrões específicos que são detetados pela câmara de um dispositivo e usados como pontos de ancoragem para posicionar os objetos virtuais de forma precisa. Essa abordagem permite criar experiências imersivas de interação entre o mundo real e o virtual.

Para simplificar a criação e deteção dos marcadores, utilizamos a ferramenta ArUco disponível no OpenCV. O ArUco oferece recursos específicos para gerar marcadores personalizados e para detectá-los em tempo real. Essa integração facilita o desenvolvimento da aplicação de realidade aumentada baseada em marcadores.

Neste trabalho iremos desenvolver iremos desenvolver aplicações para os seguintes problemas:

- **Calibração da câmera**: Crie um aplicativo para calibrar a câmera do computador offline e salvar os parâmetros intrínsecos para uso futuro.
- **Detecção e estimativa de pose da câmera com ArUco**: Utilize a biblioteca ArUco para detectar os marcadores ArUco e estimar a pose da câmera para cada marcador. Isso permitirá o alinhamento preciso de objetos virtuais com os marcadores. Procure exemplos em Python para implementar essa funcionalidade.
- **Registro de objetos virtuais**: Adicione objetos virtuais ao aplicativo, registrando-os com o sistema de coordenadas do mundo associado aos marcadores detectados. Certifique-se de que cada objeto virtual esteja corretamente associado a um ID de marcador específico para um registro preciso e adequado dos objetos virtuais.

2 VARM - Visão artificial e Realidade aumentada

2.1 Calibração da Câmara

A calibração da câmara é um processo fundamental em visão computacional e processamento de imagens, que tem como objetivo determinar os parâmetros intrínsecos e extrínsecos de uma câmara. É possível determinar a relação entre um ponto 3D no mundo real e sua projeção 2D na imagem usando todos os parâmetros e coeficientes da câmara. Para tal, é necessário obter os seguintes parâmetros:

- **Parâmetros intrínsecos**: Parâmetros intrínsecos correspondem á distancia focal e ao centro ótico, isto é parâmetros referentes à câmara. A distância focal é (fx, fy) e, o centro ótico (cx, cy) presentes na matriz da câmara (parâmetros intrínsecos) com o seguinte aspeto.

$$\text{Matriz da Câmara} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **Parâmetros extrínsecos**: Parâmetros extrínsecos correspondem aos vetores de rotação e de translação que traduzem as coordenadas de um ponto 3D para um sistema de coordenadas, ou seja, parâmetros que traduzem do sistema de coordenadas do mundo para o sistema de coordenadas da câmara.

- **Coeficientes de distorção**: Algumas câmaras apresentam uma distorção significativa nas imagens. Sendo que, existem dois principais tipos de distorção: a distorção radial e a distorção tangencial.

  A distorção radial faz com que linhas retas pareçam curvas. A quantidade de distorção radial pode ser representada da seguinte forma:

$$xdistorted = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$
$$ydistorted = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

  A distorção tangencial ocorre porque a lente de captura de imagem não está perfeitamente alinhada paralelamente ao plano de imagem. Assim, algumas áreas na imagem podem parecer mais próximas do que o esperado. A quantidade de distorção tangencial pode ser representada da seguinte forma:

$$xdistorted = x + [2p1xy + p2(r2 + 2x2)]$$
$$ydistorted = y + [p1(r2 + 2y2) + 2p2xy]$$

  Em soma, temos que calcular 5 parâmetros, conhecidos como coeficientes de distorção:

$$coeficientes\_de\_distorcao = (k_1,\ k_2,\ p_1,\ p_2,\ k_3)$$

2.2 Deteção e estimativa de pose da câmara

Os marcadores Aruco são quadrados com uma borda preta e um padrão binário no centro. Eles são usados em visão computacional e realidade aumentada para rastreamento baseado em marcadores. Os marcadores Aruco são facilmente detetáveis e robustos em várias condições de iluminação, sendo que cada marcador tem um ID único para identificação.

Fig. 1: Marker E xample

A biblioteca Aruco do OpenCV fornece funções para gerar e detetar esses marcadores, sendo estes, usados como pontos de referência para sobrepor objetos virtuais ou estimar a posição da câmara em aplicações de realidade aumentada. Os marcadores Aruco são versáteis e têm aplicações em jogos, robótica e automação industrial, como por exemplo no jogo dos *Invisimals*.

Todos os markers estão presentes em dicionário, sendo estes definidos pelo seu tamanho NxN, tendo sido utilizado para este trabalho N=6. Estes markers são constituídos por NxN quadrados criando uma padrão binário (preto ou branco), criando um padrão único, sendo esta rodeado por uma camada de quadrados pretos, como podemos observar na figura 1.

Para detetar o padrão é realizado um thresholding da frame para binarizar a mesma, sendo posteriormente realizada uma deteção deteção de contornos, para identificar potenciais markers. Após obter os potenciais markers estes são analisados para verificar a sua presença nos dicionários utilizados. Caso pretensa ao dicionário utilizado a sua posição e orientação relativa à câmara são calculdas.

2.3 Registo de objetos virtuais

O registo de objetos virtuais refere-se ao processo de alinhar e posicionar corretamente os objetos virtuais em relação aos marcadores ou ao sistema de coordenadas do mundo real. Ao detetar os marcadores ou pontos de referência no ambiente real, é possível estimar a pose da câmara e determinar a posição e orientação dos marcadores no espaço tridimensional. O registo adequado dos objetos virtuais garante que eles sejam exibidos com precisão e estejam alinhados corretamente, proporcionando uma experiência envolvente e realista para o usuário.

2.3.1 Projeção homográfica

Devido ao facto das equações de projeção de perspetiva serem não lineares é importante realizar uma projeção Homográfica dos pontos das imagens a projetar, por forma a serem lineares quando se exprimem em coordenadas homogéneas, aassim, sendo os sistemas de coordenadas envolvidos são: câmara, imagem e mundo.

- Referencial da câmara: $\lambda \begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix}$
- Referencial da imagem: $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & -k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda x^c \\ \lambda y^c \\ 1 \end{bmatrix}$
- Referencial da mundo: $\lambda \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix}$

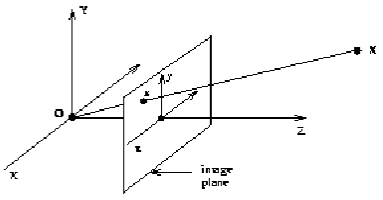


Fig. 2: Referencial da câmara

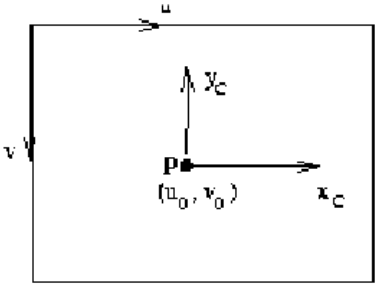


Fig. 3: Referencial da imagem

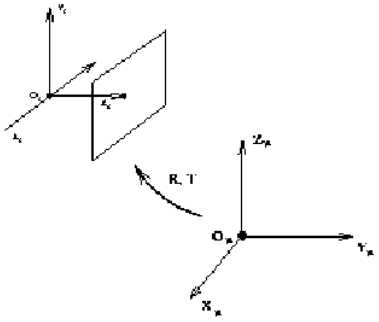
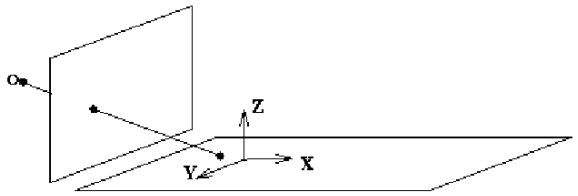


Fig. 4: Referencial do mundo

Concatenando e simplificando os diferentes modelos obtemos: $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix}$, onde X^w, Y^w, Z^w são as coordenadas do ponto a ser projetado e P = C(R|t), que é a multiplicação da matriz de parâmetros intrínsecos (câmara) com a matriz de parâmetros extrínsecos (imagem).

2.3.2 Projeção das imagens nos planos

Ao projetar o objeto virtual é importante escolher o referencial do mundo de tal forma que os pontos no plano tenham coordenadas Z nulas, permitindo assim obter uma matriz de homografia (3x3).



$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Fig. 5: Projeção homógrafa com Z nullo

3 Desenvolvimento

3.1 Calibração da câmara

No desenvolvimento deste projeto, seguimos o tutorial fornecido pelo OpenCV[2], que utiliza uma imagem de um tabuleiro de xadrez para obter os dados mencionados anteriormente. Sendo, necessários os seguintes dados de entrada: um conjunto de pontos 3D do mundo real e as coordenadas correspondentes desses pontos em 2D na imagem, sendo estes correspondentes às interseções onde dois quadrados pretos se encontram no tabuleiro de xadrez, e podendo ser obtidas atraves da função "findChessboardCorners" e da função "cornerSubPix", para refinar as coordenadas 2D desses pontos.

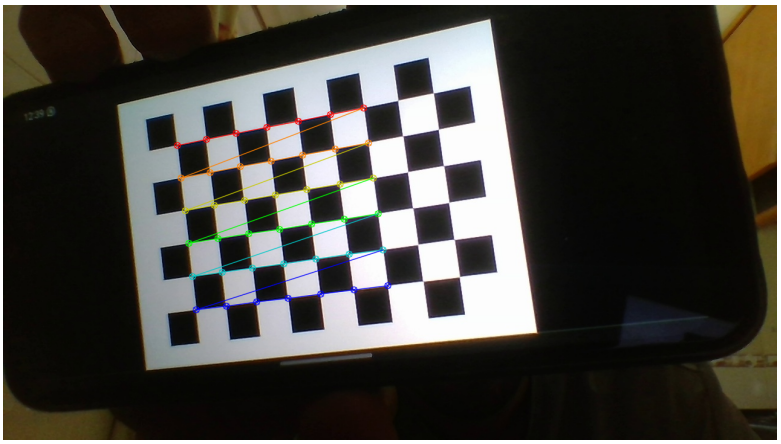


Fig. 6: Calibração da Câmara

Após obter os pontos é possível calcular a matriz da câmara, os coeficientes distorção, os vetores de rotação e os vetores de translação através da função cv.calibrateCamera(). Estes parâmetros são guardados, por forma a serem posteriormente utilizados para remover a distorção da mesma câmara. Sendo estes utilizados para remover a distorção nos pontos seguintes to projecto.

3.2 Deteção e estimativa de pose da câmara

Utilizando a biblioteca aruco do opencv é possível detetar markers criando um detetor passando-lhe o dicionário com os markers que se pretende utilizar e os parâmetros (aruco.DetectorParameters) que definem o comportamento do detetor. Assim, com a assistência deste detetor é possível obter os cantos dos markers e o respetivos ids, caso estes sejaam emcontrados.

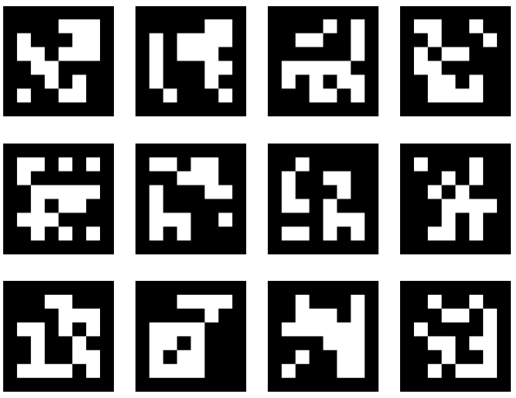


Fig. 7: First 12 Markers generated

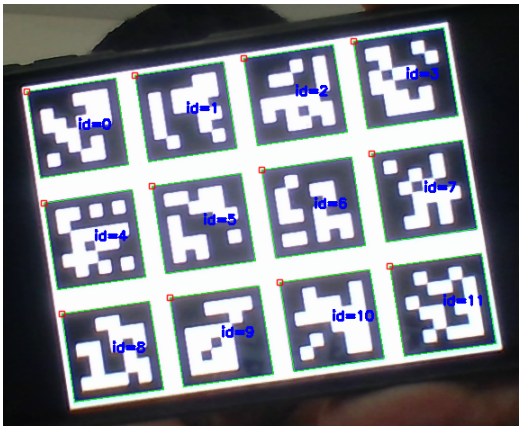


Fig. 8: First 12 Markers identified

3.3 Registo de objetos virtuais

Ao obter os cantos e os ids de cada markers encontrado é possível adicionar objetos virtuais, no entanto para tal é necessário projetar os mesmo, para incorporarem os cantos encontrados.

Esta tarefa divide-se em 3 passos:

- Obter-se a matriz homográfica, utilizando a função *cv2.findHomography*.
- Utilizaar-se a função *warpPerspective*, para distorcer o objeto de acordo com matriz homográfica, garantindo que a imagem sobrepõe-se perfeitamente em cima do marker.
- Adiciona-se a imagem resultante da aplicação da distorção no topo da imagem original, utilizando-se a função *fillConvexPoly* para preencher a zona correspondente ao marker com 0, momento após o qual adiciona-se o objeto á frame. É necessário utilizar a função *fillConvexPoly*, por forma a manter as cores originais do objeto ao adicionar a o mesmo.

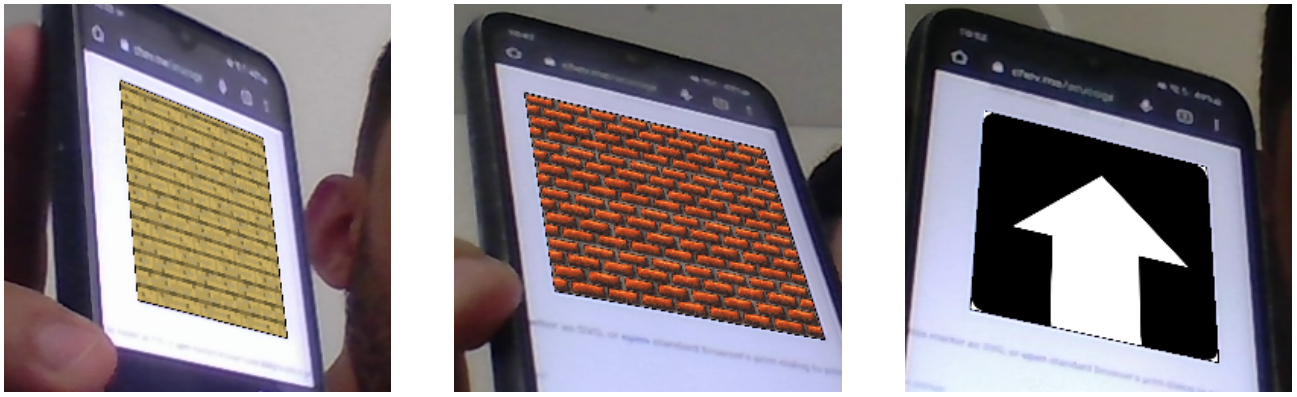


Fig. 9: Resultados 2D

Ao obter resultados positivos para uma projeção 2D, tentou-se criar uma objecto 3D, neste caso o cubo, criada a partir de diversas projeções 2D, tendo-se utilizado a lógica seguinte:

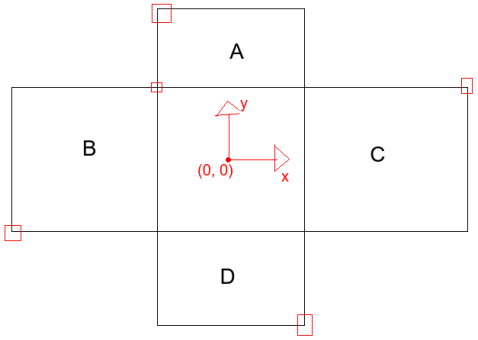


Fig. 10: 3D Lógica

Por forma desenhar cada face decidiu-se começar no canto superior esquerdo de cada fce e seguir o sentido dos relógios para deferir a a ordem dos cantos. Assim obtemos as seguintes faces:

- A: (-x, y, 1) → (x, y, 1) → (x, y, 0) → (-x, y, 0)
- B: (-x, -y, 1) → (-x, y, 1) → (-x, y, 0) → (-x, -y, 1)
- C: (x, y, 1) → (x, -y, 1) → (x, -y, 0) → (x, y, 0)
- D: (x, -y, 1) → (-x, -y, 1) → (-x, -y, 0) → (x, -y, 0)

Dest forma o objeto adicionado está sempre virado para cima

Assim sendo, ao termos a ordem dos pontos podemos mudar o eixo no qual o objeto virtual é representado.

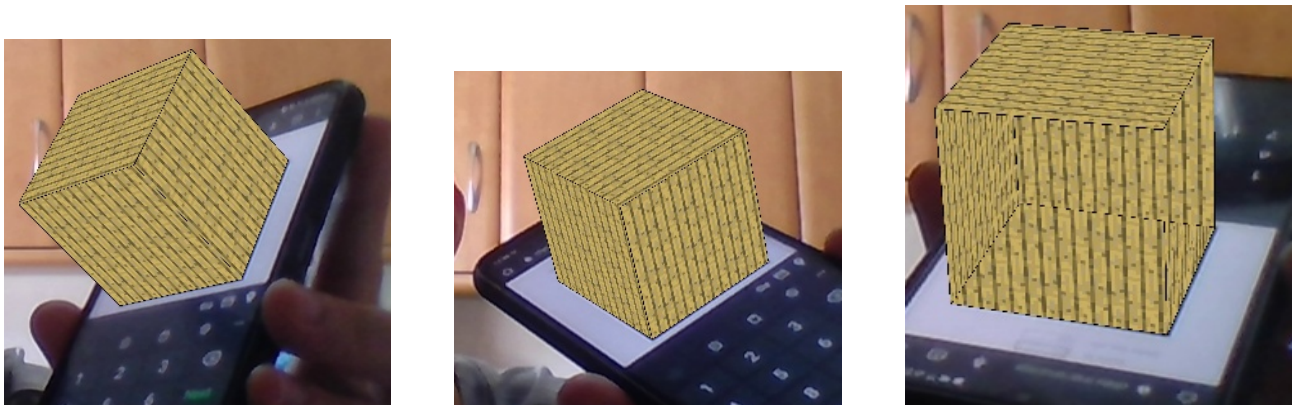


Fig. 11: Resultados 3D

Com isto surgiu um novo problema. Ao adicionar objetos estes aparecem na ordem na qual foram adicionado, criando o efeito presente na ultima imagem da figura 11. Por forma a resolver este problema, são calculados os vetores de translação das novas faces, que nos permite calcular a ordem pela qual se deve adicionar os objetos. A terceira componente componente do vetor de translação, componente z, representa a distancia do objeto á câmara, assim é possível ordenar os objetos por forma a adicionar primeiro os objetos mais longe da câmara, sendo apresentado por cima o objeto o objeto com menor z.

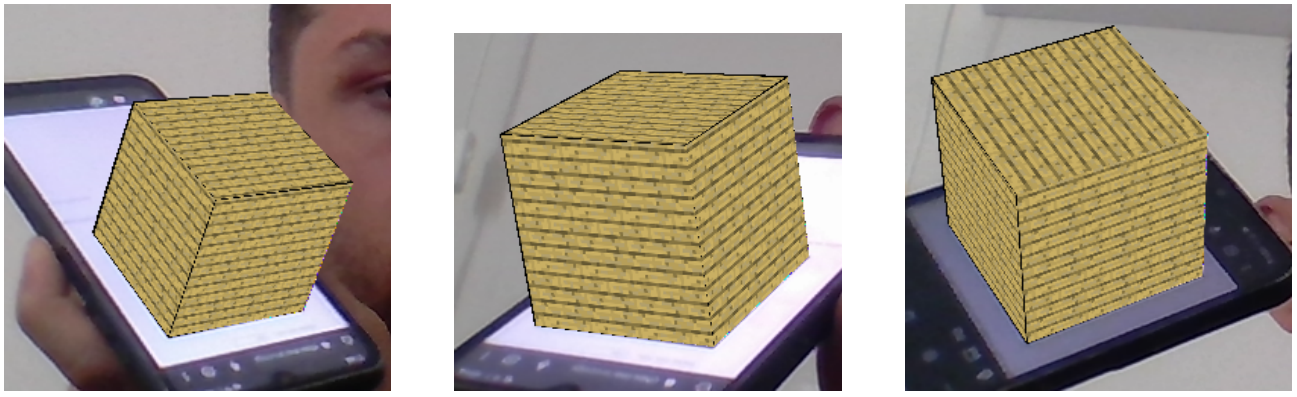


Fig. 12: Resultados 3D ordenados

4 Conclusão

Ao realizar o projeto, foi possível compreender o potencial do campo da realidade aumentada e, ao mesmo tempo, despertar uma maior curiosidade em relação a essa área. Realizar este projeto ajudou não só, a compreender melhor a área, mas também ter uma melhor compreender de como a visão funciona.

O maior obstáculo presente neste trabalho consistiu na resolução do problema referente ao orientação do objeto por forma projetar as faces mais próximas em ultimo, tendo este problema sido resolvido pós melhor compreender os parâmetros extrínsecos.

Mesmo com as dificuldades em questão, foi possível desenvolver o projeto com sucesso, tendo demonstrado bons resultados, não só na deteção de markers como na adição de objetos virtuais. Tendo em consideração os objetivos pedidos no enunciado, considero os mesmo como completos.

References