Pré-requisitos

Antes de iniciar a calibração da câmera, verifique se você possui os seguintes pré-requisitos:

- Câmera conectada ou imagens capturadas previamente
- Python 3.x instalado
- Bibliotecas OpenCV e NumPy, instaladas
- Tabuleiro de xadrez (pode ser uma foto)

Como funciona o algoritmo

1. Crie uma variável que contenha o tamanho do tabuleiro de xadrez (linhas, colunas).
2. Crie uma variável que contenha o tamanho das imagens utilizadas, por exemplo (640, 360)
3. Crie uma variável para definir o critério de avaliação, ou seja, a máxima tolerância do erro do algoritmo. No algoritmo exemplo foi utilizado a biblioteca do OpenCV da seguinte forma: criteria = cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001, onde o primeiro termo é o erro máximo permitido e o segundo a quantidade de iterações que o algoritmo vai efetuar até chegar no erro, no caso 30, esse limite impede de o algoritmo entrar em looping se ele não alcançar o erro máximo permitido.
4. Crie um array de objetos para representar as coordenadas em 3D do tabuleiro de xadrez e depois um array para representar as coordenadas em 2D do tabuleiro.
5. Crie arrays para armazenar os pontos dos objetos e das imagens

6. Crie um array para armazenar todas as imagens que serão utilizadas no algoritmo de calibração, esse array deve conter o caminho onde está a imagem.

7. Faça um loop para iterar sobre cada imagem do array anterior, onde você deve ter uma variável que vai ler a imagem, ou seja, pegar o caminho do array anterior e botar a imagem nessa variável, depois disso transforme a escala dessa imagem em cinza

8. O OpenCV tem uma função chamada: findChessBoardCorners, onde você vai passar a imagem na escala cinza, o tamanho do tabuleiro (linhas, colunas), ela retorna dois valores, o primeiro um booleano indicando se foi possível encontrar os cantos do tabuleiro de xadrez e o segundo quais são esses cantos.

9. O OpenCV tem uma função chamada: cornerSubPix, ela é utilizada para refinar a localização dos cantos do tabuleiro, com isso em mente, criamos uma nova variável para armazenar os cantos refinados e ela recebe o retorno dessa função do OpenCV, essa função espera receber como parâmetros: a imagem em escala cinza, os cantos obtidos pela função do passo 8 e o critério de parada obtidos no passo 3.

10. Armazenamos no array de objetos criado no passo 5 os pontos com as coordenadas 3D do tabuleiro que pegamos no passo 4

11. Armazenamos no array de pontos de imagens criado no passo 5 a variável com os cantos do tabuleiro refinados.

12. Utilizando a função drawChessBoardCorners, pedimos para o OpenCV desenhar os cantos do tabuleiro, enviando a imagem, o tamanho do tabuleiro e os cantos refinados, depois mostramos essa imagem na tela e esperamos 0,5 segundos para ir para a próxima imagem do loop, lembrando que esses 0,5 segundos não são fixos, podendo ser alterados

13. Após sair do loop, utilizamos a função calibrateCamera que recebe como parâmetros o array de pontos de objetos que foi criado no passo 5 e populado no passo 10, o array de pontos da imagem que foi criado no passo 5 e populado no passo 11 e o tamanho do frame, variável criada no passo 2. Recebemos um booleano dizendo se a câmera foi calibrada ou não, a matriz da câmera, os coeficientes de distorção e os vetores de rotação e translação.

14. No algoritmo de exemplo foi criado uma pasta para armazenar os resultados, mas isso é opcional, você pode indicar uma pasta padrão para onde você quer que os resultados fiquem.

15. No algoritmo aplicamos a calibração em todas as imagens que usamos, mas isso é opcional, você pode fazer somente para as imagens que quiser.

16. Pegamos a imagem em que será aplicada a calibração e usando a função (variável que contém a imagem).shape[:2] recebemos os valores da altura e da largura dela

17. Em seguida, usamos a função getOptimalNewCameraMatrix para obter a nova matriz da câmera utilizando a calibração, passamos como parâmetros a matriz da câmera, os coeficientes de distorção, o tamanho da imagem (largura, altura), o valor da escala que por padrão é 1 para não alterar ela, e o ROI que é a região de interesse da imagem, no nosso caso é toda a imagem, essa função retorna a nova matriz da câmera e o ROI

18. Criamos uma variável que vai receber a imagem calibrada, usamos a função undistort do OpenCV e passamos como parâmetro a imagem sem calibração, a matriz da câmera antiga, os coeficientes de distorção e a nova matriz que calculamos no passo 17.

19. Criamos 4 variaveis (x,y, largura, altura) que recebe os valores guardados no ROI que recebemos no passo 17 e depois usamos esses valores para recortar a imagem calibrada obtidas no passe 18: novaImg = imagemCalibrada[y:y+altura, x:x+largura]

20. Por fim, salvamos a imagem calibrada em alguma pasta a sua escolha

Passos para a Calibração

Siga os passos abaixo para realizar a calibração da câmera:

1. Conecte a sua câmera (se não for utilizar a webcam) ou certifique-se de ter fotos capturadas de diferentes ângulos. Recomenda-se ter entre 5 e 10 imagens para uma calibração adequada.

2. Certifique-se de ter as imagens do tabuleiro de xadrez em uma pasta imagens no mesmo diretório do arquivo. Caso não tenha as imagens, você pode executar o método capture_photo() no arquivo calibration.py para capturar as fotos utilizando a webcam. Certifique-se de ter permissão para acessar a câmera.

3. Abra o arquivo calibration.py e siga os comentários no código para fazer as alterações necessárias de acordo com o seu caso de uso. Verifique, por exemplo, o tamanho do tabuleiro de xadrez em chessBoardSize, o tamanho do quadro da imagem em frameSize, e o diretório das imagens em images.

4. Execute o arquivo calibration.py. O código irá realizar a calibração da câmera utilizando as imagens disponíveis. Os resultados, como a matriz da câmera e os coeficientes de distorção, serão exibidos no terminal. Além disso, as imagens corrigidas serão salvas na pasta imagens/resultados.