ArUco Marker Pose Estimation and Detection in Real-Time

Họ và tên: Vũ Đức Minh

MSSV: 20226393 Mã lớp: 162498

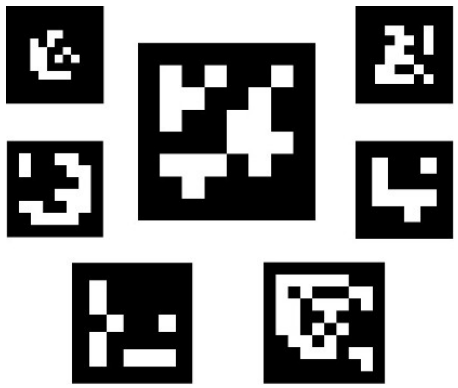
I. Giới thiệu chung:

Pose Estimate đóng vai trò rất quan trọng trong nhiều ứng dụng thị giác máy tính: điều hướng robot, thực tế tăng cường và nhiều ứng dụng khác. Quá trình này dựa trên việc tìm kiếm sự tương ứng giữa các điểm trong môi trường thực và hình chiếu 2D của chúng. Đây thường là một bước khó khăn, do đó người ta thường sử dụng các điểm đánh dấu tổng hợp hoặc điểm đánh dấu tham chiếu để đơn giản hóa quá trình này.

Một trong những phương pháp phổ biến nhất là sử dụng các điểm đánh dấu tham chiếu hình vuông nhị phân. Lợi ích chính của các điểm đánh dấu này là một điểm đánh dấu duy nhất cung cấp đủ các điểm tương ứng (bốn góc của nó) để xác định tư thế của camera. Ngoài ra, mã hóa nhị phân bên trong làm cho chúng đặc biệt mạnh mẽ, cho phép khả năng áp dụng các kỹ thuật phát hiện và sửa lỗi.

II. Marker and Dictionaries

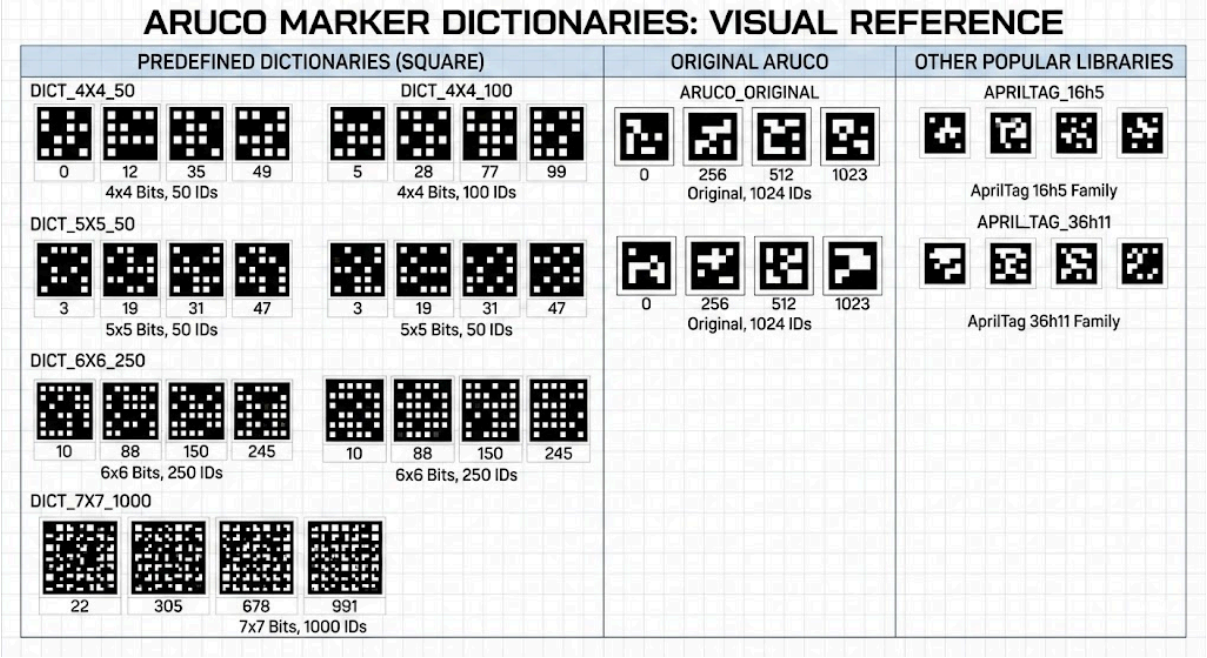
Dấu hiệu ArUco là một dấu hiệu hình vuông tổng hợp được cấu tạo bởi một đường viền đen rộng và một ma trận nhị phân bên trong xác định mã định danh (id) của nó. Đường viền đen giúp phát hiện nhanh chóng dấu hiệu trong ảnh và mã hóa nhị phân cho phép nhận dạng cũng như áp dụng các kỹ thuật phát hiện và sửa lỗi. Kích thước của dấu hiệu xác định kích thước của ma trận bên trong. Ví dụ, một dấu hiệu có kích thước 4x4 được cấu tạo bởi 16 bit.



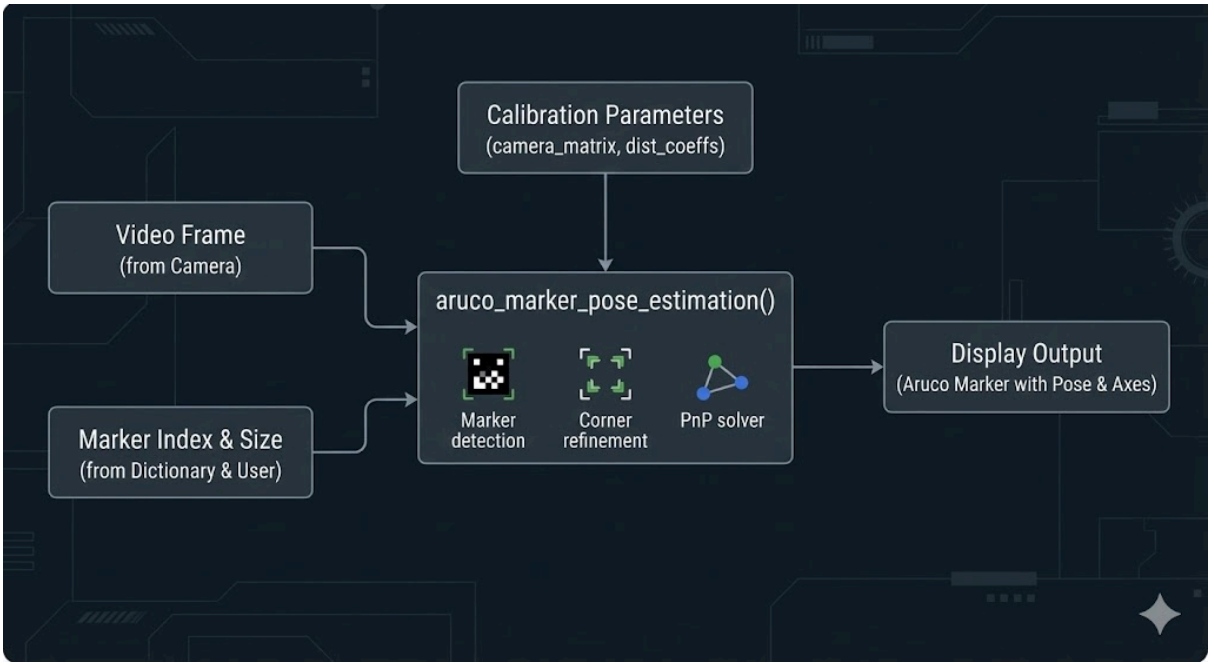
Từ điển các ký hiệu là tập hợp các ký hiệu được xem xét trong một ứng dụng cụ thể. Nó đơn giản là danh sách các mã nhị phân của mỗi ký hiệu trong từ điển đó.

Các thuộc tính chính của từ điển là kích thước từ điển và kích thước dấu hiệu.

- Kích thước từ điển là số lượng các từ chỉ thị cấu thành nên từ điển đó.
- Kích thước điểm đánh dấu là kích thước của các điểm đánh dấu đó (số bit/mô-đun).



III. Pose Estimate Program



Sơ đồ sử dụng aruco marker cho pose estimation

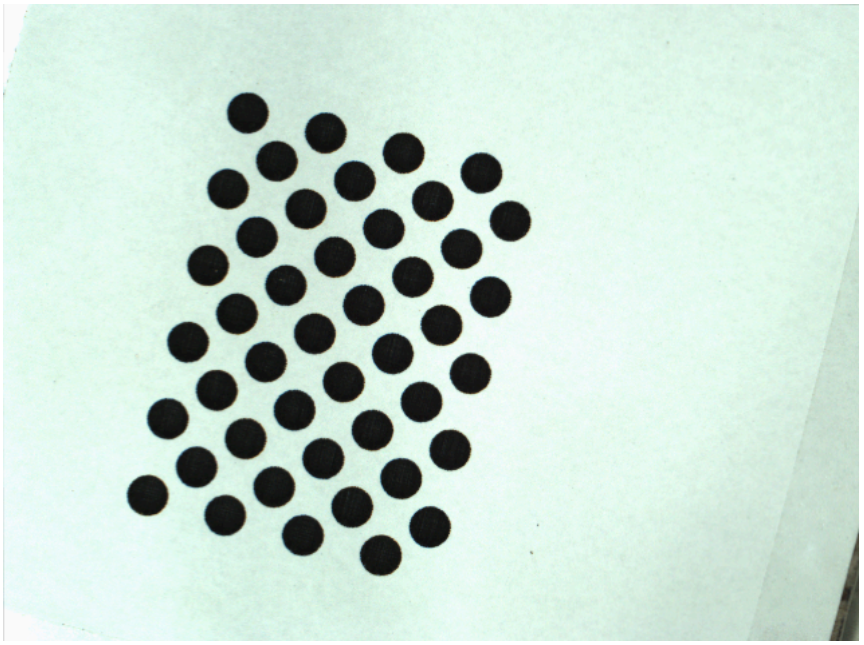
Quy trình sử dụng pose estimate:

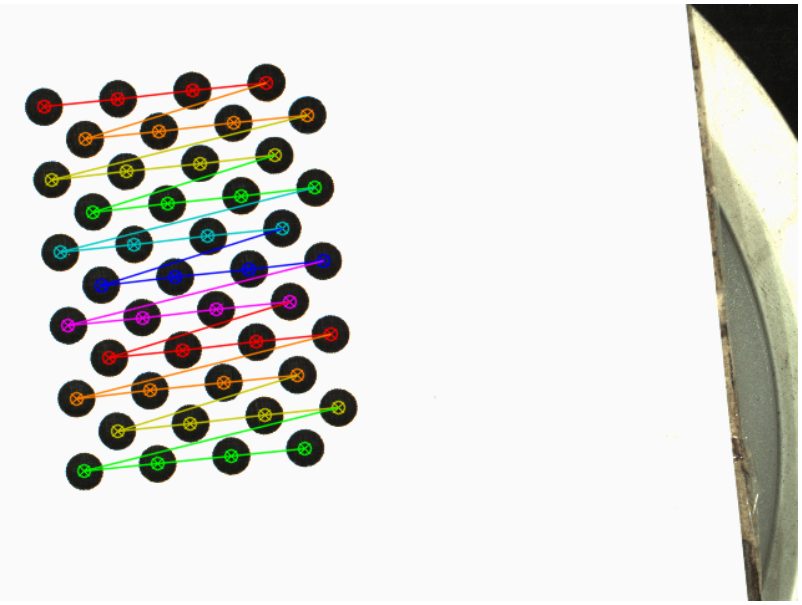
1. **Dữ liệu hình ảnh:** Luồng video 2D thời gian thực từ camera.
2. **Thông tin Marker:** Kích thước vật lý thực tế (để làm chuẩn tỷ lệ không gian) và từ điển định danh.
3. **Thông số hiệu chuẩn Camera:** Ma trận camera và hệ số méo để đảm bảo tính toán hình học chính xác.

Khối xử lý trung tâm thực hiện chuỗi thuật toán: phát hiện marker và giải mã ID trên ảnh 2D, tinh chỉnh tọa độ góc ở cấp độ dưới pixel (sub-pixel), và cuối cùng giải bài toán **PnP (Perspective-n-Point)**. Kết quả đầu ra là vector tịnh tiến và vector xoay , biểu diễn tư thế 6 bậc tự do (6-DoF) của marker trong không gian thực tế.

Program

1. Calibration Camera: Từ bài báo cáo trước tinh chỉnh bằng cách sử dụng các hình ảnh chấm tròn lệch





Sau khi tinh chỉnh lưu file calibration_camera.npz lưu lại `Camera matrix` và `Distortion coefficients.`

2. Lấy ảnh từ thư viện của acuro marker https://chev.me/arucogen/. Chụp lại ảnh rồi đo kích thước cạnh của acuro. (MAKER_SIZE)
3. Viết chương trình:

Load thư viện và cấu hình tham số:

```
import cv2
import numpy as np
import sys


ARUCO_DICT_TYPE = cv2.aruco.DICT_5X5_100


MARKER_SIZE = 0.041
```

Tải thông số hiệu chuẩn camera:

```
def main():
    try:
        data = np.load("camera_calib.npz")
        camera_matrix = data["camera_matrix"]
        dist_coeffs = data["dist_coeffs"]
        print("Đã load thông số camera thành công.")
    except:
        print("LỖI: Không tìm thấy file")
        return
```

Khởi tạo camera và trình phát hiện:

```
cap = cv2.VideoCapture(0)
aruco_dict =
cv2.aruco.getPredefinedDictionary(ARUCO_DICT_TYPE)
```

```
parameters = cv2.aruco.DetectorParameters()
```

Vòng lặp xử lý và phát hiện:

```
while True:
    ret, frame = cap.read()
    if not ret: break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    corners, ids, rejected =
cv2.aruco.detectMarkers(gray, aruco_dict,
parameters=parameters)
```

Chuyển ảnh sang xám để phát hiện các cạnh tương phản cao của marker

Pose Estimate:

```
if ids is not None:
    cv2.aruco.drawDetectedMarkers(frame, corners,
ids)


    rvecs, tvecs, _ =
cv2.aruco.estimatePoseSingleMarkers(
        corners, MARKER_SIZE, camera_matrix,
dist_coeffs
    )
```

`estimatePoseSingleMarkers`: Hàm thực hiện giải bài toán **PnP (Perspective-n-Point)**. Nó kết hợp tọa độ 2D trên ảnh

Kết quả trả về:

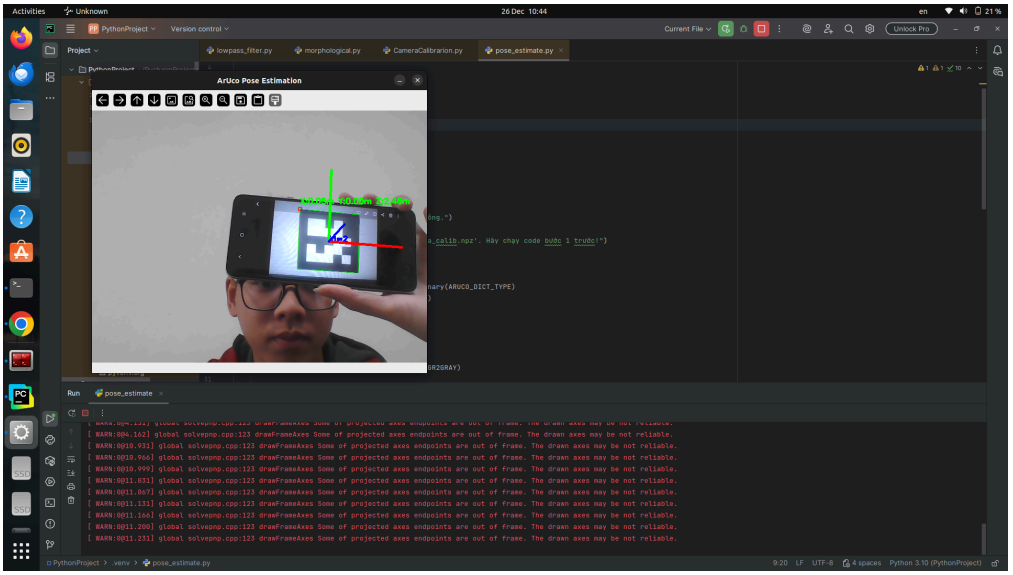
- **`rvecs` (Rotation Vectors):** Vector xoay, biểu diễn hướng quay của marker so với camera (theo định dạng Rodrigues).
- **`tvecs` (Translation Vectors):** Vector tịnh tiến (x, y, z), biểu diễn vị trí chính xác của tâm marker trong không gian 3D (đơn vị mét).

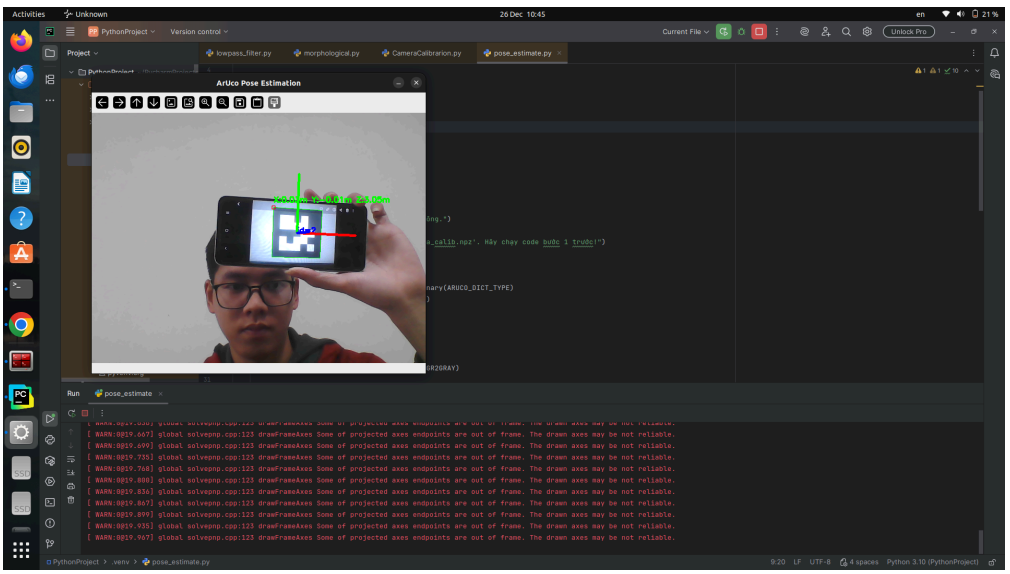
Vòng lặp thể hiện kết quả:

```
for i in range(len(ids)):

    cv2.drawFrameAxes(frame, camera_matrix,
dist_coeffs, rvecs[i], tvecs[i], 0.05)


    x, y, z = tvecs[i][0]
    text = f"X:{x:.2f}m Y:{y:.2f}m Z:{z:.2f}m"
```

```
    cv2.putText(frame, text,
                (int(corners[i][0][0][0]),
int(corners[i][0][0][1]) - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
255, 0), 2)
```

Kết quả:





IV. Tài liệu tham khảo và code

1. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
2. https://gemini.google.com/share/e07de67ff87a
3. https://www.youtube.com/watch?v=bS00Vs09Upw&t=24s
4. ComputerVision