Monocular Vision and Sensor Coupling for Indoor Localization

Houlin Chen, Lu Liang & Lei Wang
Computer Science Department
University of Wisconsin- Lacrosse
Lacrosse, Wisconsin, 54601
chen3653@uwlax.edu
lwang@uwlax.edu

Abstract

Although long used for positioning mobile devices, GPS has limitations in indoor environments due to blocked high-frequency signals and attenuation effects. This can lead to false readings. In this study, a low-cost, GPS-independent model of a neural network-assisted visual tracking system is proposed. The ArUco code is used to provide reference coordinates for the system and a neural network is used to optimize the location detection rate. To produce smooth tracking results, an inertial measurement unit (IMU) and vision-based position estimation are integrated. The proposed system significantly improves the accuracy, interference immunity, and real-time performance of the indoor tracking system. The instantiation of this system on a smartphone platform will likely enable a new cost-effective approach to indoor tracking. In the test phase, the proposed system obtained a tracking accuracy of 0.5 m without any help from expensive depth cameras or 3D LIDAR.

1. INTRODUCTION

To cope with the deviation of GPS readings for indoor positioning, a range of smart terminal-based indoor positioning technologies such as Wi-Fi, Bluetooth, Radio Frequency Identification (RFID) and Ultra-Wide Wave (UWB) technologies have emerged. wi-fi positioning systems (WPS) use the characteristics of nearby Wi-Fi hotspots and other wireless access points to discover the location of devices [1] [2].

Existing indoor localization and tracking systems either require prior knowledge of the environment, such as building floor plans, locations of Wi-Fi access points, Bluetooth beacons, and pre-established RF fingerprint databases, or expensive on-board equipment, such as 3D LiDAR, depth cameras, or omnidirectional cameras [3]. For example, Google Indoor Maps [4] can triangulate the approximate location of a user in an indoor mall (where the user is standing) using nearby WIFI points, user device Bluetooth, and the user device's built-in GPS. However, in practice, it is challenging to obtain comprehensive infrastructure information without infringing on the private rights of the user. Moreover, for existing computer vision techniques, the vision-only localization approach has extensive image processing, resulting in poor real-time performance [5] [6]. This approach cannot work in relatively poor lighting conditions. To address the problems of traditional visual localization algorithms such as poor anti-interference capability and limited real-time performance, we propose a deep learning-based visual localization method, leading to the design of an indoor localization system based on neural networks and sensor fusion. We improve the accuracy of visual localization by using neural network-based object detection and make the system efficient with real-time feedback results.

Vision-based indoor localization is used to extract information about the three-dimensional (3D) world from the two-dimensional (2D) images captured by a camera [7]. Discovering the correspondence between 3D points in the real-world environment and their 2D image projections is the most critical and complex step in this process. In our project, we use the ArUco code to help in the localization. The advantage of this marker is that a single marker provides enough correspondence information to calculate the camera pose. In addition, the internal binary encoding of the ArUco code allows the marker to maintain specific stability in terms of checks and corrections. We use neural networks to improve the accuracy of detecting markers under complex conditions such as fast motion and light changes.

In the process of detecting markers, we may encounter situations where the marker cannot be captured due to the limited view of the camera. The cell phone sensor has a relatively high acquisition frequency, and it can be used to fill this gap. On top of this, we also introduce Kalman filtering to correct the accumulation of sensor errors. The whole localization process starts by detecting the markers that appear in each frame of the video and the information obtained is the ID of the marker and the 2D coordinates of each of its corners. The 3D coordinates of the camera are obtained by solving the PnP (Perspective-n-Point) problem. At the same time, the sensors are working. The fusion of the accelerometer and gyroscope also allows for obtaining the current position. Therefore, this result will be used to fill the gaps where no markers are detected. Finally, based on the computed 3D coordinates of the device, the trajectory of the device can be plotted and compared with other systems.

2. METHODOLOGY

2.1. Overview

The goal of this project is to locate and track a mobile device and plot the 3D trajectory of the mobile device. The inputs to the system are video frames taken by the mobile device, measurements from inertial sensors, gyroscopes and magnetometers, and the outputs are its 3D motion trajectory. To accomplish the goal of this project, we divided the system into five modules. The whole process of this system is shown in Figure 1. The whole system is divided into six modules. The first module is the camera calibration. The results obtained from this module are the intrinsic and extrinsic parameters of the camera. This result will be used as input for the second module. The second module is the detection of the marker using YOLO. The result of the detection is the two-dimensional coordinates of the four corners of the marker and its unique ID. when the system does not detect the marker, it goes to the third module, which is the IMU-based localization. This module takes the values obtained from the IMU and fuses them to calculate the displacement of the device. The results obtained from this step are used as input to the Kalman filter in the fourth module to correct the results of the sensor fusion calculation. When the system clearly detects the marker, it proceeds to the fifth module, which uses the PnP algorithm to estimate the pose of the device. The main task of the last module is to present the results of the previous calculations and to plot the trajectory for comparison.

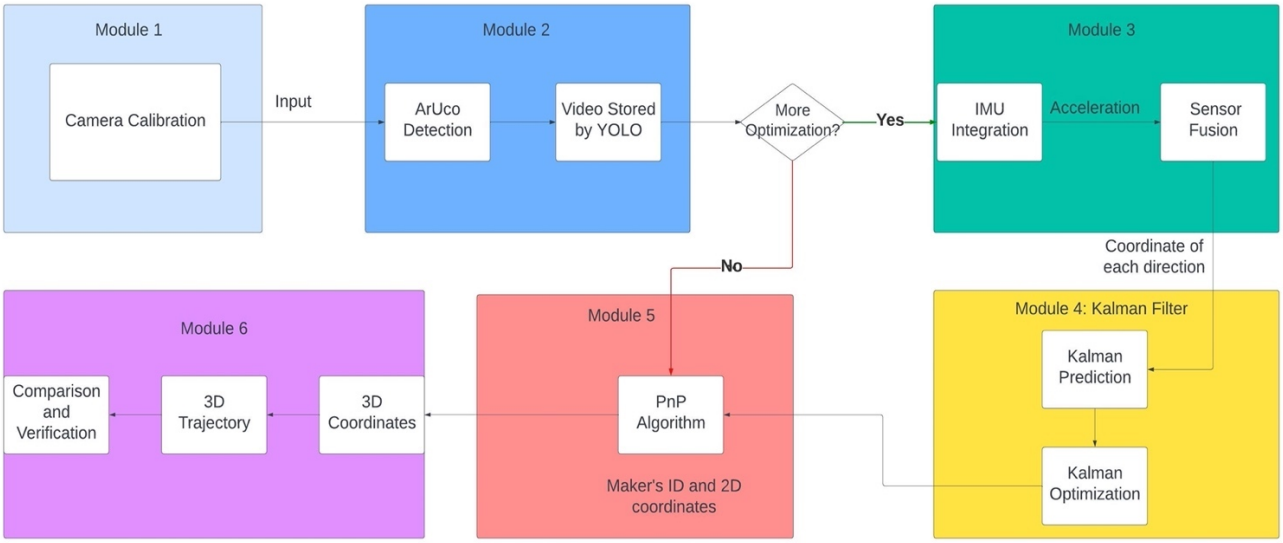


Fig. 1. The entire process of the system

2.2. Camera Calibration

The first module is the camera calibration. In machine vision applications, a geometric model of camera imaging is required to determine the 3D geometric position of a point on the surface of a spatial object and its corresponding point in the image. These geometric model parameters are the camera parameters. In most conditions, these parameters must be obtained through experiments and calculations, and this process of solving the parameters (intrinsic, extrinsic, and distortion parameters) is called camera calibration [8].

The first step in camera calibration requires converting the world coordinate system to the camera coordinate system. The world coordinate system (X_W, Y_W, Z_W), also known as the measurement coordinate system, is a three-dimensional right-angle coordinate system in which the spatial position of the camera and the object to be measured can be described. The position of the world coordinate system can be freely determined according to the actual situation. The camera coordinate system (X_C, Y_C, Z_C), also a three-dimensional right-angle coordinate system, the origin is located at the optical center of the lens, x and y axes are parallel to the two sides of the phase plane, the z-axis for the lens optical axis, and perpendicular to the image plane. The conversion process is shown in Equation (1).

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \quad (1)$$

where R is a 3×3 rotation matrix, t is a 3×1 translation vector,
$(X_C, Y_C, Z_C)^T$ and $(X_W, Y_W, Z_W)^T$ are the homogeneous coordinates of the camera coordinate system and world coordinate system, respectively.

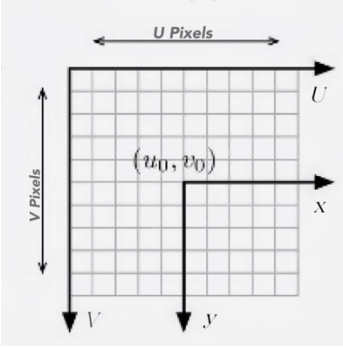


Fig. 2. Pixel coordinate and image coordinate

The next step is the conversion of pixel coordinates and image coordinates. As shown in Figure 2, the pixel coordinate system *uov* is a two-dimensional right-angle coordinate system that reflects pixel arrangement in the camera chip. The origin o is in the upper left corner of the image, and the *u* and *v* axes are parallel to the two sides of the image plane, respectively. The units of the axes in the pixel coordinate system are pixels. The pixel coordinate system is not conducive to coordinate transformation, so it is necessary to establish the image coordinate system *XOY*. The unit of its coordinate axis is usually millimeters (mm). The origin is the intersection of the camera's optical axis and the phase plane (called the principal point), which is the center of the image. X-axis and Y-axis are parallel to the u-axis and v-axis, respectively. Therefore, the two coordinate systems are translational, i.e., they can be obtained by translation. This conversion can be done by Equation (2).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/dX & 0 & u_0 \\ 0 & 1/dY & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (2)$$

where, *dX*, *dY* are the physical dimensions of the pixel in the X and Y-axis directions, respectively. u_0 and v_0 are the coordinates of the principal point.

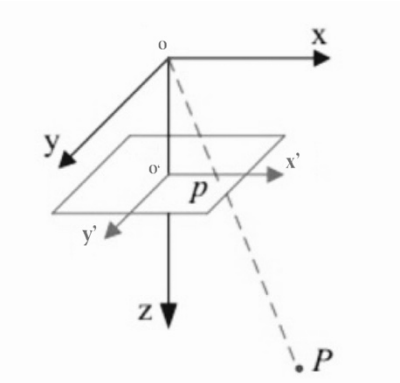


Fig. 3. Pinhole imaging principle

Figure 3 illustrates the relationship between any point P in space and its image point p. The line between P and the camera optical center o is oP, and the intersection of oP and the image plane p is the projection of the point P in space on the image plane. This process is perspective projection, as represented by the following matrix:

$$s\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3}$$

where s is the scale factor (s is not zero), and f is the effective focal length (the distance from the optical center to the image plane). $(x, y, z, 1)^T$ is the homogeneous coordinates of the spatial point P in the camera coordinate system xoy, and $(X, Y, 1)^T$ is the homogeneous coordinates of the image point p in the image coordinate system XOY. Combining Equations (1) to (3) we can get the intrinsic and extrinsic parameters of the camera.

$$s\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1/dX & 0 & u_0 \\ 0 & 1/dY & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & 0 & u_0 & 0 \\ 0 & a_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \boldsymbol{M}_1 \boldsymbol{M}_2 \boldsymbol{X}_w \tag{4}$$

where, $a_x = f/dX$, $a_y = f/dY$, are called the scale factors of the u and v axes. M_1 and M_2 are the intrinsic and extrinsic parameters of the camera, respectively.

2.3. YOLO Detection

The second module uses YOLO to detect markers. As we mentioned earlier, YOLO is a single-stage detector, which is fast and ideal for applications in real-time systems. The following will describe how YOLOV3 is applied to this project.

When a frame is passed into YOLOV3, this image is first resized to 416x416 grids, and a gray bar is added around the image to prevent distortion. YOLOV3 then splits the images into 13x13, 26x26, and 52x52 grids, which are used to detect large, medium, and small objects, respectively. Each grid point is responsible for

detecting its lower right corner area, and if the object's center point falls in a grid, then the object's position will be determined by that grid point. In the example given in Figure 4, an image containing the object to be detected, i.e., the ArUco marker, is input into the YOLOV3 neural network and then surrounded by gray bars. In this image, the ArUco belongs to a large object, so a 13×13 grid image will detect the result. For the training part of YOLO, we choose to train on Google Colab.

2.4. Camera Pose Estimation

2.4.1. Principle&Conditions

The two-dimensional coordinates obtained from the detection of the markers will be used for the estimation of the camera pose. The camera poses estimation is mainly based on the PnP (Perspective-n-point) algorithm. General conditions for the PnP problem [9]:

- Coordinates of the n 3D reference points in the world coordinate system.
- Corresponding to these n 3D points, the coordinates of the 2D reference point are projected on the image.
- The intrinsic parameters of the Camera are denoted by M_1.

Based on our experimental results (for details, see Section IV), the EPnP algorithm is of the highest accuracy among the existing PnP algorithms.

Most non-iterative PnP algorithms will first solve for the depth of the feature point to obtain its 3D coordinates of it in the camera coordinate system. The EPnP algorithm, on the other hand, represents the 3D coordinates in the world coordinate system as a weighted sum of a set of virtual control points. For the general case, the EPnP algorithm requires the number of control points to be four, and these four control points cannot be coplanar. Because the camera's extrinsic parameters are unknown, the coordinates of these four control points under the camera reference coordinate system are unknown. Furthermore, if we can solve the coordinates of these four control points under the camera reference coordinate system, we can calculate the camera's pose [10].

2.4.2. Control Points and Barycentric Coordinates

Using the EPnP algorithm as a basis, the coordinates of the control points and are represented. This can help us to estimate homogeneous barycentric coordinates. Further, we can calculate barycentric coordinates from isometric coordinates. The positions of these points can help us to derive the camera angle and thus further estimate the camera pose.

In this paper, the superscripts ^{w} and ^{c} are used to denote coordinates in the world and camera coordinate systems, respectively. Then, the coordinates of the 3D reference points in the real-world frame are $p^w{}_i, i = 1,...,n$, the coordinates in the camera frame are $p^c{}_i, i = 1,...,n$. The four control points in the world coordinate system are $c^w{}_j, j = 1,...,4$, the coordinates in the camera reference coordinate system are $c^c{}_j, j = 1,...,4$.

$$\begin{bmatrix} \boldsymbol{p}_i^w \\ 1 \end{bmatrix} = \boldsymbol{C} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1^w & \boldsymbol{c}_2^w & \boldsymbol{c}_3^w & \boldsymbol{c}_4^w \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} \quad (5)$$

$[p^{W_i T}1]^T$ and $[c^{W_j T}1]^T$ are both isometric coordinates. Thus, we also get barycentric coordinates computed as follows: (See Appendix for detailed derivation)

$$\begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} = \boldsymbol{C}^{-1} \begin{bmatrix} p_i^w \\ 1 \end{bmatrix} \tag{6}$$

2.4.3. Selection of Control Points

A specific method for determining the control points is given in here. The set of 3D reference points is $P_i^W, i = 1,...,n$ and the barycentric coordinates of the 3D reference points are chosen as the first control point:

$$\boldsymbol{c}_1^w = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i^w \tag{7}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{p}_1^{w^T} - \boldsymbol{c}_1^{w^T} \\ \cdots \\ \boldsymbol{p}_n^{w^T} - \boldsymbol{c}_1^{w^T} \end{bmatrix} \tag{8}$$

Donating the characteristic value of A^TA as $\lambda_{C,i,i=1,2,3}$, the corresponding feature vector is $v_{c,i,i=1,2,3}$. Thus, the remaining three control points can then be determined by the following formula:

$$\boldsymbol{c}_j^w = \boldsymbol{c}_1^w + \lambda_{c,j-1}^{\frac{1}{2}} v_{c,j-1}, j = 2,3,4 \tag{9}$$

2.4.4. Solve for Coordinates of the Control Point in Camera Coordinates:

$u_i, i = 1,...,n$ is the 2D projection of the reference point $p_i, i = 1,...,n$, then,

$$\forall i, w_i \begin{bmatrix} \boldsymbol{u}_i \\ 1 \end{bmatrix} = K\boldsymbol{p}_i^C = K\sum_{j=1}^{4} a_{ij}\boldsymbol{c}_j^C \tag{10}$$

Substitute $c^C{}_j = [x^C{}_j, y_j^C, z_j^C]^T$ into the above equation and write K in the form of focal length f_u, f_v and optical center (u_c, v_c), then,

$$\forall i, w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} a_{ij} \begin{bmatrix} x_j^C \\ y_j^C \\ z_j^C \end{bmatrix} \tag{11}$$

Two linear equations can be obtained from Equation 11:

$$\sum_{j=1}^{4} a_{ij} f_u x_j^C + a_{ij}(u_c - u_i) z_j^C = 0 \tag{12}$$

$$\sum_{j=1}^{4} a_{ij} f_v y_j^C + a_{ij}(v_c - v_j) z_j^C = 0 \tag{13}$$

Concatenating all n reference points, we can obtain a linear system of equations:

$$Mx = 0 \tag{14}$$

where M is a $2n \times 12$ matrix, $x = [\boldsymbol{c}_1^{c^T}, \boldsymbol{c}_2^{c^T}, \boldsymbol{c}_3^{c^T}, \boldsymbol{c}_4^{c^T}]$, x is the coordinate of the control point in the camera coordinate system, which is a 12×1 vector and x is in the right null space of M, or $x \in ker(M)$. Hence,

$$\boldsymbol{x} = \sum_{i=1}^{N} \beta_i \boldsymbol{v}_i \tag{15}$$

In the equation above, v_i is the N eigenvector corresponding to the N null eigenvalues of M. For the i-th control point:

$$\boldsymbol{c}_j^C = \sum_{k=1}^{N} \beta_k \boldsymbol{v}_K^{[i]} \tag{16}$$

where v_k is i-th 3×1 sub-vector of eigenvector v_k. Then we can obtain v_i by computing the eigenvectors of M^TM.

The next step is to calculate $\beta_{i,i=1,\ldots,N}$. Because the extrinsic parameters of the camera describe only coordinate transformations and do not change the distance between control points, thus:

$$\left\|\boldsymbol{c}_i^C - \boldsymbol{c}_j^C\right\|^2 \tag{17}$$

$$\left\|\sum_{k=1}^{N} \beta_k \boldsymbol{v}_K^{[i]} - \sum_{k=1}^{N} \beta_k \boldsymbol{v}_K^{[j]}\right\|^2 = \left\|\boldsymbol{c}_i^C - \boldsymbol{c}_j^C\right\|^2 \tag{18}$$

This is a linear equation for $\beta_{ij,i,j=1,\ldots,N}$. In EPnP algorithm
[14], four cases N = 1,2,3,4 is discussed. When N takes different values, the number of unknowns of the linear equation is:

- N = 1, the unknown number is 1 • N = 2, the unknown number is 3
- N = 3, the unknown number is 6
- N = 4, the unknown number is 10

When N = 4, the number of equations is 6 and the number of unknowns is more than the number of equations. By commutativity of the multiplication, we have

$$\beta a b \beta c d = \beta a \beta b \beta c \beta d = \beta a 0 b 0 \beta c 0 d 0 \tag{19}$$

where $\{a^0,b^0,c^0,d^0\}$ represents any permutation of the integers $\{a,b,c,d\}$. Then we can reduce the number of unknowns. For example, if we solve for $\beta_{11},\beta_{12},\beta_{13}$, then we get $\beta_{23} = \frac{\beta_{12}\beta_{13}}{\beta_{11}}$.

2.4.5. Gauss-Newton Optimization

The objective function of the optimization is:

$$Error(\beta) = \sum_{(i,j s.t. i<j)} \left(\left\|\boldsymbol{c}_i^C - \boldsymbol{c}_j^C\right\|^2 - \left\|\boldsymbol{c}_i^W - \boldsymbol{c}_j^W\right\|^2\right)^2 \tag{20}$$

2.4.6. Calculating the Camera's Pose

The calculation of the pose of the camera in the EPnP algorithm is as follows.

1) Calculate the coordinates of the control point in the camera reference coordinate system.

$$c_i^c = \sum^{N} \beta_k v_k^{[i]}, i = 1,2,3,4 \tag{21}$$

j=1

2) Calculate the coordinates of the 3D reference point in the camera reference coordinate system.

$$\boldsymbol{p}_i^c = \sum_{j=1}^{4} a_{ij} \boldsymbol{c}_j^c, i = 1, ..., n \quad (22)$$

3) Calculate the barycentric coordinates $p^w{}_0$ of $p^w{}_i, i = 1,...,n$ and matrix A:

$$\boldsymbol{p}_0^w = \frac{1}{n} \sum_{i=1}^{n} p_i^w \quad (23), \qquad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{p}_1^{w^T} - \boldsymbol{p}_0^{w^T} \\ \cdots \\ \boldsymbol{p}_n^{w^T} - \boldsymbol{p}_0^{w^T} \end{bmatrix} \quad (24)$$

4) Calculate the barycentric coordinates $p^c{}_0$ of $p^c{}_i, i = 1,...,n$ and matrix B:

$$\boldsymbol{p}_0^c = \frac{1}{n} \sum_{i=1}^{n} p_i^c \quad (25), \qquad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{p}_1^{c^T} - \boldsymbol{p}_0^{c^T} \\ \cdots \\ \boldsymbol{p}_n^{c^T} - \boldsymbol{p}_0^{c^T} \end{bmatrix} \quad (26)$$

5) Calculate H:

$$H = B^T A \quad (27)$$

6) Calculating the singular value decomposition (SVD) of H:

$$H = U^X V^T \quad (28)$$

7) Calculate the rotation R in the pose:

$$R = UV^T \quad (29)$$

8) Calculate the translation t in the pose:

$$t = p_0^c - Rp^w{}_0 \quad (30)$$

l represents the camera's position in the real-world coordinate system:

$$l = -R^{-1}t \quad (31)$$

2.5. Sensor Fusion

2.5.1. Role of IMU in Our System

The third module is the IMU-based localization method. This method is used as a replacement when visual localization is not available. As can be seen in Figure 1, when the marker is not detected, the system uses

the IMU measurements to calculate the 3D coordinates of the mobile device. The sensors used in this system are mainly an accelerometer, gyroscope, and magnetometer, all of which are currently equipped in smartphones. Before we can use the sensor for localization, we need to convert the phone's coordinate system.

2.5.2. Coordinate System Conversion

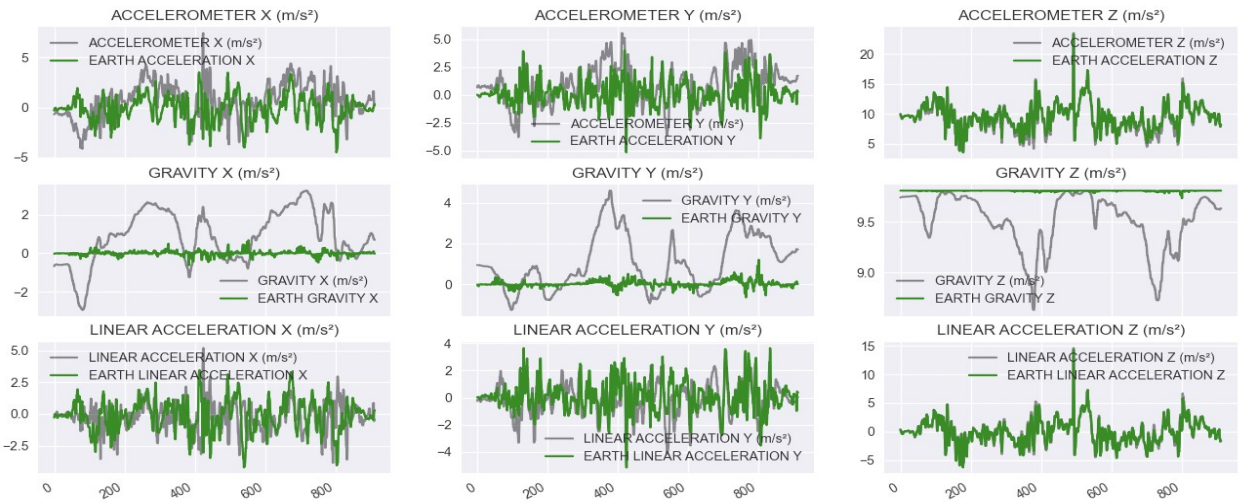


Fig. 4. Change in sensor values before(orange) and after(green) coordinate system conversion

The acceleration sensor of an Android phone refers to its coordinate system when measuring acceleration. Therefore, we need to convert the phone's coordinate system to an inertial, non-rotating coordinate system, which is the Earth coordinate system. This conversion will make it possible to hold the Android phone in any orientation and measure the correct acceleration vectors to calculate the phone's trajectory in the earth coordinate system. The transformation from the Phone coordinate to Earth's transformation [11] is done with formula (32) as shown below

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_z(\psi)R_y(\theta)R_x(\phi)\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (32)$$

Where $[X, Y, Z]$ are the phone's coordinate system linear accelerations, and R_z, R_y, R_x are the rotation matrices for each axis in order to rotate the [X, Y, Z] over to earth's $[x,y,x]$ axes. The Euler angles (ψ,θ,φ) correspond to the angles about the pitch, roll, and yaw. The difference between the acceleration in the earth coordinate system (The green line) and the Android phone coordinate system (The gray line) can be seen in Figure 4. The vertical coordinate in the figure represents the value of the sensor, and the horizontal coordinate represents the sampling times (we used a sampling rate of 100 Hz). After the conversion, the Z-axis's gravity stays near 9.8 meters per second, while the gravity in the X-axis and Y-axis stays near 0 meters per second. This result is the same as what we know as common sense.

2.5.3. Displacement Estimation

The last step of the sensor module is to use the acceleration to calculate the displacement and thus estimate the distance. Acceleration is the rate of change of an object's velocity. At the same time, velocity is the rate of change in the position of the same object. In other words, velocity is the derivative of position, and acceleration is the derivative of velocity. Therefore, the following equation is available, (See Appendix for derivation)

$$\vec{s_x} = \int (\int (\vec{a_x})\, \mathrm{d}t)\, \mathrm{d}t \tag{33}$$

A similar expression for displacement in the y-axis and z-axis.

2.6. Kalman Filter Correction

After the initial value is given by the IMU, the error of the sensor itself will accumulate in the continuous optimization. The role of Kalman filtering here is to correct this error using linear iterations. Kalman filtering [12] is mainly divided into two steps, prediction, and correction. Prediction is the estimation of the current state based on the state of the previous moment, and correction is the integrated analysis based on the observation of the current state and the estimation of the previous moment to estimate the system's optimal state value. Then the process is repeated the next moment. The Kalman filter iterates continuously, it does not require many-particle state inputs, only process quantities, so it is fast and well-suited for state estimation of linear systems. Applying Kalman filtering to this project uses the position information obtained from vision-based localization to update the position information obtained from sensor-based localization.

3. SYSTEM TESTING EXPERIMENT

3.1. Design and Preparation

To demonstrate that our proposed method is superior to both sensor-based localization only and vision-based localization only, we first tested the effect of applying only one of these methods for localization and tracking. Lastly, we tested our complete system, i.e., applying the IMU to support visual localization.

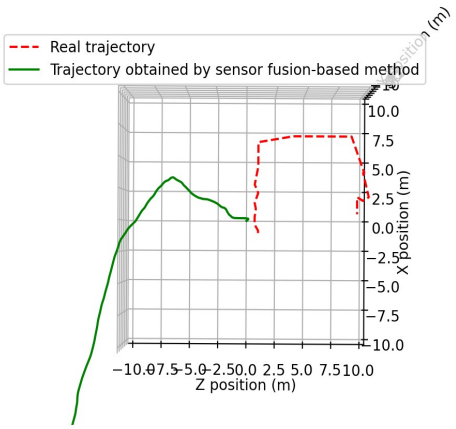
Our experimental setup is as follows:

- Location: The laboratory in the Prairie Springs Science Center
- Device: Nokia 7.2
- Measuring tool: Tape measure
- Conditions: Sufficient and insufficient light; shade and no shade on markers

We choose a point in the lab as the origin of the world coordinates, and then affix ArUco markers at different heights
and on different surfaces. Each marker's location information was recorded to observe the difference between the system test results and the actual data results. We take the center point of the first marker as the origin of the world coordinate system.

In our indoor localization and tracking experiment, fifteen trails are conducted. Figure 5 is the actual 3D moving trajectory. The actual path shows the route of our experiment. The true path contains two corners, as well as a movement in the vertical direction. The maximum distance of motion in the x-axis direction reached 7 meters, and the maximum length of motion in the z-axis direction was 10 meters.

3.2. Testing of IMU-based Indoor Tracking & Vision-based Tracking



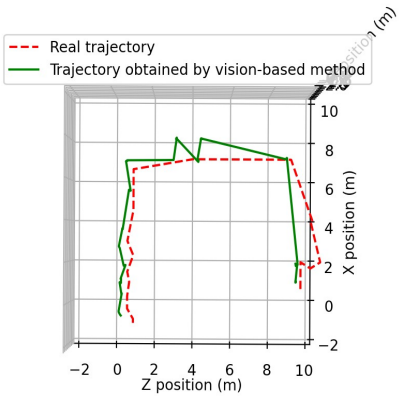


Fig. 5(left). Comparison of real trajectory and trajectory obtained by sensor fusion-based method
Fig. 6(right). Comparison of real trajectory and trajectory obtained by vision fusion-based method

We first conducted 15 experiments on the IMU-based indoor localization method to evaluate its effectiveness of the method. Figure 5 shows the results of the sensor fusion-based method through 15 experiments. As can be seen from the figure, the sensor-based method, which is also known as the IMU-based method, has a significant drift in obtaining motion trajectories. The reason why the IMU-based method has such a large error is that the interference of the gravitational acceleration in the vertical direction cannot be eliminated and there are residuals. In addition, the double integration of acceleration leads to the accumulation of errors. Another reason is the drift of the sensor during the measurement. Specifically, the reading of the inertial sensor is not zero when a moving device goes from rest to motion and back to rest. The second part of the experiment is for the vision-based localization method. Again, this part of the experiment was repeated fifteen times. Figure 6 shows the comparison between the trajectory obtained by the vision-based localization method and the real trajectory. It can be seen from the figure that the vision-based approach provides trajectories that are not smooth because it calculates the absolute position of the mobile device. This is caused by the fact that the vision-based localization method relies on markers as reference positions. When no marker is captured in FoV, the pose, and position of the camera cannot be estimated. In this case, there will be a gap in the trajectory, and the localization will continue once at least one marker is detected. As a result, the estimated motion trajectory will suddenly move from the previous position to the current one.

3.3. Testing of Our Optimized Method

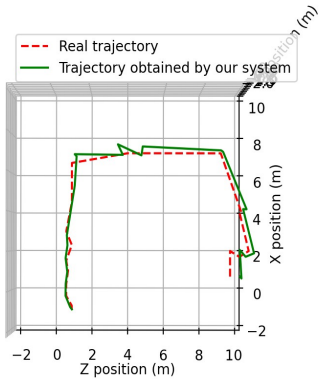


Fig. 7. Comparison of real trajectory and trajectory obtained by our system

The third part of the experiment is to evaluate our proposed method and test the reliability and effectiveness of our system. The experiment has also been repeated 15 times. Figure 7 points out that the obtained trajectories have no big jumps, which means that sensor-based localization successfully fills this gap when vision-based localization cannot be used. This largely indicates that this method combines the advantages of the two methods mentioned above and achieves optimization. The new method achieves improvements of 93.6%, 80%, and 84.4% in the x-axis, y-axis, and z-axis, respectively, with respect to the sensor-based method.

3.4. Experiment Results in Comparison

The experimental results show that the IMU-based indoor localization method has the worst performance. The average error of this method is about 8 meters in our 15 iterations of experiments. The vision-based localization method performs better, with an average error of less than 1 meter. Our system performed the best, with the highest accuracy of localization, with an average error of less than 0.5 m. Compared with the other two axes, the motion trajectory of the phone obtained by our system has a relatively large error in the Y-axis. The reason for this phenomenon may be that the markers are closer together in the vertical direction, causing repeated detection. Overall, this greatly enhances the accuracy of indoor positioning.

4. CONCLUSION AND FUTURE WORKS

Our proposed neural network-based indoor localization method improves the stability of the indoor tracking system. It is improved by introducing inertial sensors to assist vision-based localization. Compared to vision-based localization without the assistance of inertial sensors, our system avoids the inability to localize due to missing markers. The proposed method does not rely on any expensive depth camera and can be easily planted on a mobile device. We evaluate and validate our method with the prototype implementation on the

smartphone platform. The experimental results show that the system has strong robustness to the complex indoor environment, strong anti-interference ability, high accuracy, and fast processing speed, which meets the demand for indoor localization. The neural network model we trained explicitly for detecting ArUco code has a breakneck detection speed, taking only 0.164 seconds to detect a single frame. Overall, the proposed method demonstrates a tracking accuracy of under 0.5 meters.

In terms of future work, we plan to increase further the number and diversity of datasets, which will significantly improve the accuracy of the neural network in detecting markers. This improves the accuracy of vision-based location detection of markers, which in turn improves the accuracy of our system's location. We also plan to include hardware devices like depth cameras in the approach. This will allow our visual localization method to no longer rely on markers and use objects already present in the indoor environment for localization.

5. APPENDIX

5.1. Derivation for barycentric coordinates

The EPnP algorithm expresses the coordinates of the reference point as a weighted sum of the coordinates of the control point:

$$\boldsymbol{p}_i^w = \sum_{j=1}^{4} a_{ij} \boldsymbol{c}_j^w, with \sum_{j=1}^{4} a_{ij} = 1 \tag{1}$$

where the a_{ij} are homogeneous barycentric coordinates. They are unique and can easily be estimated. In the camera coordinate system, the same relationship exists:

$$\boldsymbol{p}_i^c = \sum_{j=1}^{4} a_{ij} \boldsymbol{c}_j^c \tag{2}$$

Assuming that the extrinsic parameters (rotation matrix R and translation vector t) of the camera are $[R,t]$ then a relationship exists between the virtual control points $c^w{}_j$ and $c^c{}_j$:

$$\boldsymbol{c}_j^c = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_j^w \\ 1 \end{bmatrix} \tag{3}$$

Considering that the EPnP algorithm expresses the reference point coordinates as a weighted sum of the control point coordinates, then can get:

$$\boldsymbol{p}_i^c = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} p_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{4} a_{ij} c_j^w \\ 1 \end{bmatrix} \quad (4)$$

$$\boldsymbol{p}_i^c = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{4} a_{ij} \boldsymbol{c}_j^w \\ 1 \end{bmatrix} = \sum_{j=1}^{4} a_{ij} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} c_j^w \\ 1 \end{bmatrix} = \sum_{j=1}^{4} a_{ij} \boldsymbol{c}_j^c \quad (5)$$

In the above derivation, the important constraint $\sum_{j=1}^{4} a_{ij} = 1$ of EPnP on the weight a_{ij} is used. Without this constraint, the above derivation will not hold. Putting the four control point constraints together yields the following equation:

$$\begin{bmatrix} \boldsymbol{p}_i^w \\ 1 \end{bmatrix} = \boldsymbol{C} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1^w & \boldsymbol{c}_2^w & \boldsymbol{c}_3^w & \boldsymbol{c}_4^w \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} \quad (6)$$

Obviously, $[p^{W}{}_{i}{}^{T}1]^T$ and $[c^{W}{}_{j}{}^{T}1]^T$ are both isometric coordinates. The equation (1) in the reference paper, however, is essentially a linear combination of the isometric coordinates of 3D reference points with the isometric coordinates of the control points. Thus, we also get barycentric coordinates computed as follows:

$$\begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} = \boldsymbol{C}^{-1} \begin{bmatrix} p_i^w \\ 1 \end{bmatrix} \quad (7)$$

5.2. Derivation for Displacement Expression

The relation between acceleration/ displacement and velocity: $\vec{a} = \dfrac{d\vec{v}}{dt}$, $\vec{v} = \dfrac{d\vec{s}}{dt}$

The relation between displacement and velocity: $\vec{a} = \dfrac{d(d\vec{s})}{dt^2}$

The integral is the opposite of the derivative. If the acceleration of an object is known, then we can obtain the position of the object using the double integral. Assuming that the initial condition is 0, then there is the following equation.

$$\vec{v} = \int (\vec{a})\, \mathrm{d}t$$

$$\vec{s} = \int (\vec{v})\, \mathrm{d}t$$

6. REFERENCES

[1] T. Lindner, L. Fritsch, K. Plank, and K. Rannenberg, "Exploitation of public and private wifi coverage for new business models," in *Building the E-Service Society*. Springer, pp. 131–148, 2004.

[2] G. E. Violettas, T. L. Theodorou, and C. K. Georgiadis, "Netargus: A snmp monitor & wi-fi positioning, 3-tier application suite," in *2009 Fifth International Conference on Wireless and Mobile Communications*. IEEE, pp. 346–351, 2009.

[3] J. Kunhoth, A. Karkar, S. Al-Maadeed, and A. Al-Ali, "Indoor positioning and wayfinding systems: a survey," *Human-centric Computing and Information Sciences*, vol. 10, pp. 1–41, 2020.

[4] G. MAPS, "Google indoor maps," Website, [Online]. Available from: https://www.google.com/maps/about/partners/indoormaps/, 2021(Accessed 22-December-2022).

[5] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[6] P. Roy and C. Chowdhury, "A survey of machine learning techniques for indoor localization and navigation systems," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–34, 2021.

[7] N. Piasco, D. Sidibe, C. Demonceaux, and V. Gouet-Brunet, "A survey´ on visual-based localization: On the benefit of heterogeneous data," *Pattern Recognition*, vol. 74, pp. 90–109, 2018.

[8] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[9] A. A. B. Pritsker, *Introduction to Simulation and SLAM II*. Halsted Press, 1984.

[10] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[11] Wikipedia contributors, "Conversion between quaternions and Euler angles — Wikipedia, the free encyclopedia," [Online]. Available from: https://en.wikipedia.org/w/index.php?title=Conversion between quaternions and Eulerangles&oldid=998442809, 2021(Accessed 12-January-2023).

[12] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.