**INDOOR LOCALIZATION FOR MECANUM-WHEELED MOBILE ROBOT: A VISION-BASED LOCALIZATION FRAMEWORK**

**(Định vị trong nhà cho robot di động sử dụng bánh xe Mecanum: Khung định vị dựa trên thị giác máy tính)**

Trinh Dang Bao Duc2,3\_2113239\_Control Engineering and Automation, Bach Ngoc Phu2,3 \_2114409\_Control Engineering and Automation, Bui Minh Duc2,3 \_2110130\_Control Engineering and Automation, Nguyen Trong Tai1,3, \*

1 Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

2 Faculty of Electrical and Electronics Engineering - Undergraduate Student, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

3 Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

\* Corresponding author: nttai@hcmut.edu.vn

**Abstract**

This paper addresses the problem of indoor localization and autonomous navigation for a Mecanum-wheeled mobile robot operating in environments without LiDAR sensing. A vision-based localization framework is developed by fusing information from ArUco marker detections, inertial measurement units, and wheel odometry. Two estimate methods - a regular Extended Kalman Filter (EKF) with χ²-based statistical testing for measurement outlier rejection and an Adaptive Extended Kalman Filter (AEKF) with online noise covariance adaptation - are used and compared methodically. A nonlinear Backstepping controller is created to allow for accurate trajectory tracking while taking into consideration the particular kinematic limitations of the Mecanum platform. The complete system is experimentally validated in a structured indoor environment, demonstrating the comparative performance of the localization algorithms and the effectiveness of the control strategy. The proposed approach provides a low-cost, reliable solution for LiDAR-free indoor navigation of omnidirectional mobile robots.

***Keywords:*** *indoor localization, multi-sensor fusion, ArUco markers, Adaptive Extended Kalman Filter, χ²-testing, mecanum robot, nonlinear control, autonomous navigation*

1. **Introduction**

The use of autonomous mobile robots in indoor environments is expanding, with small-scale platforms playing a key role in prototyping algorithms for larger systems. This paper presents an integrated sensor fusion system for an ADAS model vehicle, offering a comprehensive platform to test autonomous driving algorithms before real-world deployment.

While GPS is widely used for outdoor localization, it is ineffective in constrained indoor environments, necessitating specialized indoor solutions. Accurate localization is essential for autonomous systems, as modules like motion planning and control depend on precise positional data. LiDAR (Light Detection and Ranging) enables high-precision scanning, making it suitable for mapping and relative localization in reflective indoor settings. Platforms such as TurtleBot3 and Clearpath Jackal are equipped with LiDAR sensors to support SLAM and navigation tasks [1]. However, LiDAR systems are often costly and power-intensive,

which limits their suitability for small-scale robots or budget-constrained applications.

Another approach involves acoustic-based indoor positioning systems, where robots determine their position using ultrasonic signals emitted from beacons installed in the environment. Such systems have been implemented in service robots like Keenon T1 and Savioke Relay [2]. Nevertheless, these methods are often susceptible to environmental noise, resonance, and obstructions, which degrade localization accuracy and stability.

Single-modality localization methods have inherent limitations, making multi-sensor fusion essential. Integrating vision, IMU, and wheel odometry leverages the strengths of each sensor, with fusion algorithms like EKF, AEKF, and χ²-enhanced EKF showing strong performance indoors. This study proposes and evaluates a vision-based localization system using ArUco markers fused with IMU and encoder data to provide accurate, robust state estimation for an indoor ADAS model vehicle.

1. **Methods and Implementation**

**2.1 Sensor setup and physical parameters of model robot**

***Table 1.*** *Physical parameters of the mobile robot*

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Meaning |
| m | 6 | Mass of Robot |
| Iα | 0.24328 kg.m2 | Moment of inertia around z axis |
| r | 0.035 m | Radius of robot wheel |
| L1 | 0.11 m | Distance from robot center to wheel along vertical axis |
| L2 | 0.10 m | Distance from robot center to wheel along horizontal axis |

**A machine with wires and lights

AI-generated content may be incorrect.**

***Figure 1.*** *Hardware setup of the Mecanum Robot*

A diagram of a method

AI-generated content may be incorrect.

***Figure 2.*** *Scheme design of the indoor autonomous driving framework Robot*

Our indoor localization system is implemented on a custom Mecanum-wheeled mobile robot, equipped with an onboard IPC for centralized processing, a 1080p RGB camera for ArUco marker detection, and a BNO055 IMU for yaw rate sensing. Each CHR-GM37-520 motor is encoder-equipped and controlled via an Arduino ATmega 2560. Hardware specifications are listed in Tables 1, and the setup is shown in Fig. 1. The localization, trajectory planning, and control algorithms are implemented on the Mecanum-wheeled mobile robot platform described in Sec. 2.1. Fig. 1 illustrates the complete indoor autonomous navigation framework, where localization relies on three sensing sources. As outlined previously, wheel velocities and yaw rate are obtained from the encoders and the IMU, respectively, and serve as inputs for the prediction step in the sensor fusion module. The update step utilizes position and orientation measurements from ArUco markers detected by the onboard RGB camera. The fused state estimate (x, y, φ, v) is then passed to a trajectory planner.

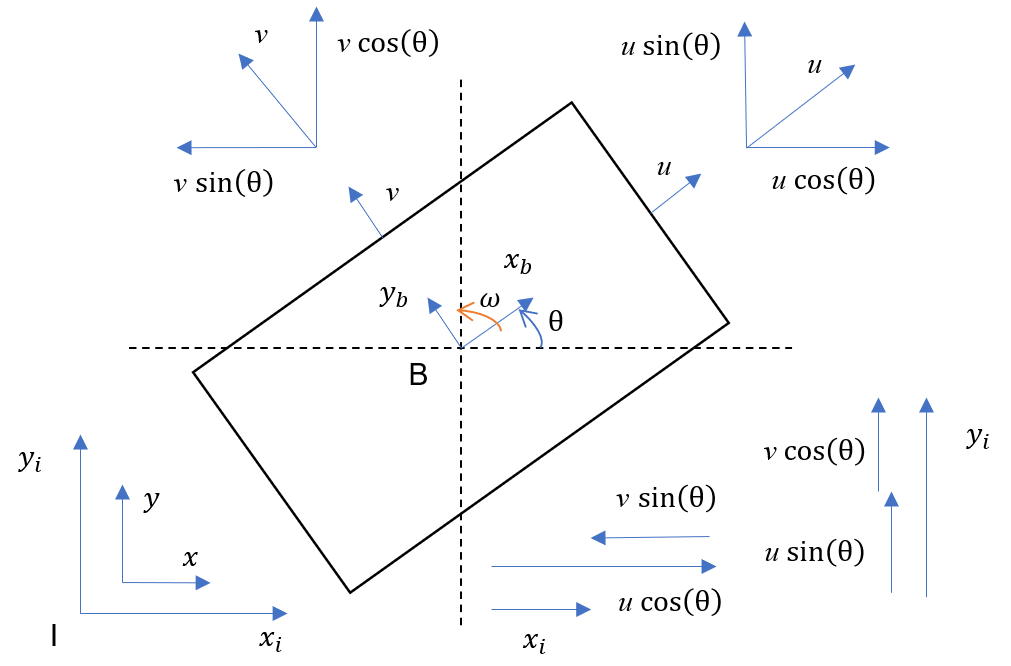
**2.2 Kinematic and dynamic model of the mobile robot system**

**2.2.1 Kinematic model**

Consider a mobile robot with an inertial frame (*xi, yi*) and a body-fixed frame (*xb, yb*), as shown in Fig. 2. The system states include the robot's position (*x, y*) and orientation *θ* in the inertial frame, along with body-frame velocities *u*, *v* (linear) and *ω* (angular). Following [3], the kinematic model relates the body-frame velocities at the center point *B* to their counterparts in the inertial frame:

(1)

Where:



***Figure 2.*** *Kinematic model of the Mecanum Robot*

**2.1.2 Dynamic model**

Consider the dynamic model of a mobile robot with coordinate frames as shown in Fig. 3, where point BBB denotes the robot’s geometric center. The system parameters include mass m, dimensions l (length) and d (width), moment of inertia , angular velocity ω, angular acceleration ​, linear accelerations ​, ​, and wheel diameter *a*. Based on [3], the corresponding dynamic equations are derived.

A diagram of a square with lines and letters

AI-generated content may be incorrect.

***Figure 3.*** *Robot dynamics model*

The dynamic model of the mobile robot is developed using the Lagrangian method:

(2)

Here, is the Lagrangian, where *KE* includes both translational and rotational kinetic energy, and *PE* = 0 as the robot moves on a flat surface. The kinetic energy is given by:

(2)

Using the velocity transformation: *;,* we get:

(3)

Applying the Lagrangian formulation with generalized velocities *u*, *v* and *ω*, the expressions for force and torque components can be derived as shown in [3]. These expressions capture the effects of mass distribution and rotational dynamics. Based on this formulation, the general dynamic model of the mobile robot is expressed as:

(4)

Where: is the velocity vector in the body frame, D is the inertia matrix, represents the nonlinear terms, and is the vector of generalized forces and torques.

**2.2 Backstepping control design**

**2.2.1 Kinematic and dynamic equations**

The robot's system of equations is described as:

(5)

This system can be rewritten in a control-affine form as:

(6)

Where:

* is the Jacobian matrix

Let be the desired reference value, with its first and second derivatives denoted as *,* . Define the tracking error as . The objective control is to ensure that and .

**2.2.2 Backstepping controller design**

To design a stabilizing control law, we adopt the backstepping method as presented in [3]. Let and define a Lyapunov function . Assuming the virtual control ​ is selected to stabilize the first subsystem.

The second step introduces the actual control input *u* to stabilize the error . A composite Lyapunov function is considered. Following the derivation in [3], the control input is chosen to ensure with positive constants , , leading to the final control law:

[4] (7)

This control law satisfies the Lyapunov stability condition:

(8)

**2.4 ArUco marker mearsurement**

AruCo is a widely adopted, low-cost fiducial marker system that can be easily integrated with various sensors for indoor robotic applications. To accurately extract visual information and estimate the robot's pose, camera calibration must be performed. This process yields two essential parameters: the intrinsic matrix of the camera and the distortion coefficients. These parameters are crucial for correcting lens distortion and enabling precise pose estimation from the detected markers.

Ảnh có chứa hàng, biểu đồ, văn bản, Song song

Nội dung do AI tạo ra có thể không chính xác.

***Figure 4.*** *PnP estimates the pose of markers relative to the camera frame*

Let (*XC, YC, ZC*) be the 3D coordinates of marker corners in the camera frame, and (*Xs, Ys*) be their 2D pixel coordinates in the image. The transformation from each 3D point (*xci, yci, zci*) to its corresponding 2D pixel location (*xsi, ysi*) is described by a projection equation:

(9)

Where i {1,2,3,4}corresponds to the four corner points of the marker. The values (*xs0, ys0*)denote the coordinates of the principal point, and *fx*, *fy* are the scale factors in *Xs* and *Ys* axes. These four parameters are components of the camera intrinsic matrix. In addition, *rij* and *ti* represent the elements of the rotation matrix *Rc* and the translation vector *Tc* respectively. Once the *Rc* and *Tc* matrices are obtained [6], the relative orientation angle *ϕ1* and the distance ddd from the ArUco marker to the robot can be computed using the following equation:

) (10)

(11)

At this point, and represent the relative lateral and longitudinal offsets (in meters) between the camera and the robot’s center in the real-world coordinate system. Each ArUco marker is pre-defined with a known pose in the world frame, consisting of (*x0, y0, ϕ0*), where *x0* and *y0*​​ are the global coordinates and *ϕ0*​ is the marker's orientation. The global position of the robot can then be estimated using the relative distance *d*, the relative angle *ϕ1* and the known global pose of the detected ArUco marker. The robot's orientation in the global frame, denoted as , is computed as: . During the setup phase, the camera is assumed to be rigidly mounted and aligned with the robot’s center axis. Therefore, the longitudinal offset is set to zero. The orientation corrections and are calculated as follows:

, (12)

Finally, the position of the robot in the global coordinate frame, expressed in meters, can be computed as:

*,* (13)

Ảnh có chứa biểu đồ, hàng, văn bản

Nội dung do AI tạo ra có thể không chính xác.

***Figure 5.*** *Single-track model with ArUco-based pose detection in the global frame*

Since the acquired image frames are often affected by noise due to lighting conditions, color variations, and other environmental factors, the estimated pose values may become unstable or inaccurate. To enhance the robustness and precision of pose estimation, we propose applying additional filtering techniques. Specifically, a Low-Pass Filter (LPF) [7] is employed with a smoothing factor α = 0.05, applied uniformly to the x, y positions and the orientation angle ϕ.

In scenarios where the robot fails to detect ArUco markers, a dead-reckoning method is used to estimate its pose based on IMU and encoder data. The velocities and are derived from the encoders, while the angular velocity comes from the IMU. The robot's pose at time t+Δt is updated as follows:

,

,

.

This ensures position estimation continues even without visual marker input.

* 1. **Estimation methods**

**2.5.1 EKF**

**A diagram of a algorithm

AI-generated content may be incorrect.**

***Figure 6.*** *Block diagram of EKF family with Adaptation and χ²-based Filtering*

The goal of the EKF is to estimate the system's state from noisy control inputs and measurements, minimizing the estimation error. Each time step *t*, the EKF uses a dynamic model to predict the system's new state and the corresponding covariance matrix.

*,* (14)

Where is the nonlinear system model, which in this study is a kinematic single-track model [8]; is the process noise covariance matrix; is the Jacobian of *,* obtained by first-order Taylor expansion around the previous state .

(15)

Where is the time step, the yaw rate [deg/sec] is measured by the IMU and is used as the control input of the system . The speed [m/sec] is obtained directly from the wheel encoders.

The system receives sensor data (camera) . This data is used in the update step to refine the state estimate and the covariance matrix . First, we compute the innovation , where . Innovation represents the difference between the actual measurement and the predicted measurement. A small suggests good sensor-model agreement, while a large one may indicate noise or model error. However, its significance depends on what is considered a normal range; therefore, it is necessary to compute the residual covariance matrix . It represents the uncertainty of the innovation ​, combining both the predicted state uncertainty *P* and the uncertainty from the sensor *R*. The Kalman Gain acts as a weighting factor that determines how much to trust the new measurement compared to the model prediction. Finally, we update the estimated state and the covariance matrix.

, (16)

The updated state always lies between the prediction and the measurement, weighted by the confidence in each. After updating the covariance matrix, the uncertainty of the state decreases.

**2.5.2 AEKF**

In addition to EKF, we also experimented with the AEKF filter [9], which updates the process and measures noise matrices dynamically rather than using fixed values. Following the method in [10], we compute the residual . This represents the error between the actual measurement and the inferred measurement from the updated state. Therefore, the process noise and measurement noise matrices are adaptively calculated during the update phase*.*

(17)

(18)

**2.5.3 EKF with -testing**

In the EKF with -testing [5], the chi-squared test is used to detect faulty sensor measurements. The value is computed from the innovation error, normalized by the innovation covariance matrix, as follows:

*,* (19)

If the value exceeds an experimentally defined threshold, the measurement is rejected as an outlier and skipped in the EKF update. Otherwise, correction proceeds as usual. This safeguards the localization system from major errors, ensuring stability and accuracy in noisy conditions.

1. **Results and Discussion**

***Table 2.*** *Configuration parameters for EKF family*

|  |  |
| --- | --- |
| Parameters | Values |
| threshold | 0.03 |
|  | [0.2, 0.2, 0.1, 0.02] |
|  | [0.005, 0.005, 0.001] |
|  | [0.1, 0.1, 0.1, 0.1] |

To evaluate trajectory tracking performance, we compare the robot’s estimated and ground-truth values of x, y, and yaw angle across different algorithms. As the Backstepping controller depends on state estimates, we also assess how estimation accuracy influences control performance by analyzing both estimated and ground-truth positions on the global map.

The reference trajectory is a circular path with a 1-meter radius. Localization is achieved using 12 ArUco markers placed at known positions (, , ). The selected process and measurement noise covariance matrices , (shown in Table 3) are applied to all methods. For EKF and EKF with χ² threshold, these matrices remain constant = and = , while AEKF dynamically adjusts them using scaling factors *αQ**and αR.* The best configuration for AEKF was found to be {*αQ*, *αR*} = {0.95, 1}. The χ² threshold was set to 0.03 for the corresponding method.

A diagram of a person's face

AI-generated content may be incorrect.

***Figure 7.*** *EKF path and position error*

Figures 7–9 show trajectory tracking results: ground-truth positions from ArUco markers (green), estimated positions (blue), and the reference trajectory (red). In Figure 7, the EKF enables the robot to follow the reference path but suffers from notable estimation errors, particularly in yaw, due to sensor noise.

A diagram of a circle with different colored lines

AI-generated content may be incorrect.

***Figure 8.*** *AEKF path and position error*

Figure 8 illustrates the benefits of adaptive covariance tuning in AEKF. The estimated trajectory is smoother and more closely aligned with the reference, indicating better noise handling and estimation accuracy. Figure 9 shows that EKF with χ² thresholding also performs well under uncertainty, similar to AEKF.

A diagram of a circle with different colored lines

AI-generated content may be incorrect.

***Figure 9.*** *EKF with χ² threshold path and position error*

Figure 10 highlights the role of an appropriate χ² threshold in rejecting outliers. Initially, floor noise caused large deviations, but once calibrated, the method produced accurate tracking. This confirms that the χ²-based outlier rejection effectively enhances localization in noisy environments.

A graph of a chi-squared value

AI-generated content may be incorrect.

***Figure 10.*** *χ² value and χ² threshold*

To quantify the performance of each estimation method, we calculated the error between the estimated and ground-truth values of the robot’s position (x,y) and orientation θ at each time step. The ground truth is derived from the known global positions of ArUco markers, while the estimated values come from the respective filters. Let (​, ​, ) be the ground-truth pose at timestamp i, and () be the corresponding estimated pose. The instantaneous errors are computed as:

*, ,*

The mean error is the average of the absolute errors over the full trajectory:

Mean error in , similar for y and .

The peak error is defined as the maximum value among all instantaneous errors:

Peak error in  , similar for y and .

***Table 3.*** *Peak localization errors for each filter*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filter | Time(s) | Max error x (m) | Max error y (m) | Max error θ (rad) |
| EKF | 98 | 0.212 | 0.245 | 0.167 |
| AEKF | 96 | 0.094 | 0.089 | 0.074 |
| EKF with *χ²* threshold | 92 | 0.142 | 0.137 | 0.092 |

***Table 4.*** *Mean localization errors of each filter*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filter | Time(s) | Mean error x (m) | Mean error y (m) | Mean error θ (rad) |
| EKF | 98 | 0.12 | 0.134 | 0.167 |
| AEKF | 96 | 0.047 | 0.043 | 0.053 |
| EKF with *χ²* threshold | 92 | 0.076 | 0.071 | 0.062 |

**4. Conclusions**

This paper presents a low-cost, comprehensive indoor localization system for ADAS-scale vehicles in pre-mapped environments. The system integrates ground truth measurement, multi-sensor fusion-based pose estimation, and control output. An enhanced Extended Kalman Filter (EKF) with adaptive noise tuning and Chi-square-based outlier rejection improves robustness in noisy conditions. A backstepping controller was used to assess state estimation accuracy under real-world control. Experimental results show that the system effectively filtered erroneous ArUco measurements and maintained reliable localization, demonstrating strong potential for indoor autonomous vehicle applications where GPS is unavailable.

**Acknowledgement:** This research is funded by Ho Chi Minh City University of Technology – VNU-HCM under grant number SVKSTN-2024-ĐĐT-12. We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

**References**

[1] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 5, pp. 1179–1187, 1989.

[2] Z. Chen, Y. Li, and Q. Zhang, "An Indoor Ultrasonic Positioning System Based on Mobile Robot," IEEE Sensors Journal, vol. 21, no. 3, pp. 3447–3457, 2021.

[3] Z. Hendzel and L. Rykała, "Modelling of dynamics of a wheeled mobile robot with Mecanum wheels with the use of Lagrange equations of the second kind," Int. J. Appl. Mech. Eng., vol. 22, no. 1, pp. 81–99, 2017.

[4] H. H. Giáp and N. Q. Đại, "Nghiên cứu thiết kế bộ điều khiển Backstepping ứng dụng cho rô bốt di động bốn bánh Mecanum," J. Sci. Technol., vol. 59, no. 3, pp. 31–37, Jun. 2023.

[5] B. Brumback and M. Srinath, "A chi-square test for fault-detection in Kalman filters," IEEE Trans. Autom. Control, vol. 32, no. 6, pp. 552–554, 1987.

[6] OpenCV, "Calibration with ArUco and ChArUco." [Online]. [Accessed: May. 4, 2025]. Available: https://docs.opencv.org/4.x/da/d13/tutorial\_aruco\_calibration.html

[7] "Low-pass filter design with microcontroller." [Online]. [Accessed: May. 11, 2025]. Available: <https://www.researchgate.net/publication/365108086_Low-Pass_Filter_Design_with_Microcontroller>

[8] D. Schramm, M. Hiller, and R. Bardini, Single Track Models. Berlin, Heidelberg: Springer, 2014, pp. 223–253.

[9] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation," in Proc. IEEE Power Energy Soc. Gen. Meeting, 2017.

[10] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman filtering for INS/GPS," J. Geodesy, vol. 73, pp. 193–203, 1999.