Sensor Data Fusion for the Localization and Position Control of One Kind of Omnidirectional Mobile Robots

Peter Eberhard and Qirong Tang

Abstract This contribution deals with the problem of sensors and sensor data fusion for mobile robots localization and position control. For this, the robot internal odometry is first corrected and used for position control. Then, an extended Kalman filter based on the corrected odometry and a novel North Star navigation system is designed in a distributed manner. The estimated information from the extended Kalman filter is feed back to the desired poses for further accelerating and precising the position control process. Finally, after the analysis of data flows and uncertainties, the whole developed scheme is verified by experiments on an omnidirectional mobile robot.

Key words: Data fusion, Odometry and correction, North Star, Extended Kalman filter, Omnidirectional mobile robot

1.1 Introduction

Localization is a fundamental and key issue for mobile robots since it is the prerequisite for many abilities, e.g., path planning, navigation and execution of tasks. Besides traditional sensor based methods, there are also many new theoretical approaches for solving this problem such as, using probabilistic [24], [28], topology [6], [18], [20] and fuzzy logic [3], [16]. No matter what kind of methods are used, when facing real robots one must deal with the used sensors and the resulting sensors data. Although some methods are very

Peter Eberhard
Institute of Engineering and Computational Mechanics, Pfaffenwaldring 9, 70569 Stuttgart, Germany, e-mail: peter.eberhard@itm.uni-stuttgart.de

Qirong Tang
Institute of Engineering and Computational Mechanics, Pfaffenwaldring 9, 70569 Stuttgart, Germany, e-mail: qirong.tang@itm.uni-stuttgart.de

mature in theory, it is still difficult to utilize and validate them under realistic conditions or with combinations of other strategies. Sources of difficulties come, e.g., from restrictions like sensors precision, environment noise, uncertainties of the robot system. For these reasons, researchers rarely use just one method or sensor for robot localization, but instead hybrid strategies which are based on several kinds of methods based on hardware and software.

Usually the localization is mobile robots oriented, typically for differentially driven mobile robots. However, in recent years omnidirectional mobile robots attracted researchers interest due to their unique features. This study presents methods for the localization of one kind of omnidirectional mobile robots, the Festo Robotino, and the scheme is valid for all of the individuals in the robot group which is used for our purpose of searching a target in an environment. Some researches focus on similar robots and localization questions, e.g., [15] and [27], but they pay more attention to the path planning issue. The investigations described in [2], [7], [10], [17] and [21] focus on multi-robot collaborative localization but unfortunately they neither consider to improve the accuracy of sensors nor individual measurements before used for localization. Thus, they have to pay more efforts on data post-processing. The robot trajectories in these studies are neither optimal nor under an unified movement guidance. To the contrary, they directly use the available sensors even with high noises or uncertainties. The study [21] contributes its decentralized architecture and [2] uses a minimum entropy approach to minimize sensor uncertainty. The work in [17] is oriented for an outdoor environment where the global positioning system (GPS) is applied. However, GPS doesn't effectively work inside of the buildings. The research in [7] and [10] performs localization by relying on the probabilistic approaches of Markov localization and maximum likelihood estimation, respectively. As a result, these studies for multi-robot localization greatly increase the burden for positioning and also involve more interferences. Other researches study differentially driven robots and there are only few researches that concern the localization for omnidirectional mobile robots with consideration of robot sensors and swarm behavior first.

Section 1.2 introduces some common concepts and states the reason of localization, then the overall position control scheme and the used sensors are described in Section 1.3. After the introduction of Kalman filters the detailed sensor data fusion and localization processes are shown in Section 1.4. Experiments with a real robot and results analysis are performed in Section 1.5 while Section 1.6 gives conclusions.

1.2 Some Common Concepts

For a better understanding, it is necessary to introduce some common concepts such as, e.g., control points and measurements, localization and nav-

igation, controller and robots position control. A classical feedback control diagram is illustrated in Fig. 1.1 which is used in the following.

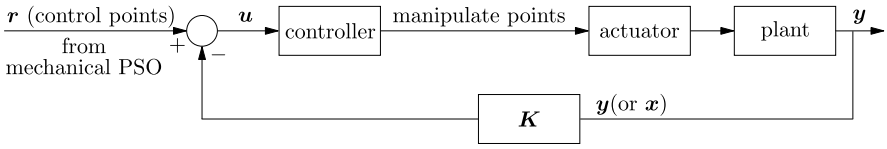


Fig. 1.1 Classical feedback control

1.2.1 Control Points and Measurement

The control points in our case are referred to the desired position points which are generated by robot swarm under mechanical PSO algorithm. The controller manipulate points sometimes are also called control variables. They are generated by the system controller and act on the controlled plant directly or through an actuator to manipulate the concerned variable to approach the reference variable.

The controlled plant show certain behavior, and the system should have some sensors to measure it. Thus, the system needs some devices to perform measurements, then compare the obtained information to the references. In this study, the sensors are the robots internal odometer and the external North Star system.

1.2.2 Navigation Strategies and Classification

For mobile robot research one can not avoid the topic of localization and navigation. The former one answers the question 'Where am I?' to the robot, i.e., provides the position, orientation and additional information like the environment and info about other robots. The latter one is a comprehensive strategy which guides the robot (hopefully in an optimal way) and includes also environment detection and obstacle avoidance yielding a collision free path. Table 1.1 lists some basic methods for localization and navigation. Details can be found in the mentioned publications or in [5].

Robot motion planning belongs to robot navigation which includes location, path planning, and obstacle avoidance. However, these three different topics are often considered at the same time and are summarized in the term 'motion planning' in practical applications.

Table 1.1 Robot navigation strategies and motion planning

localization	relative	dead reckoning	
	absolute	imaging, laser, GPS	
path planning	local	APF, genetic algorithms, fuzzy logic	
	global	environment modeling	graph methods, free-space methods, grid methods
		path search	A* algorithms [9], D* optimal algorithms [22]
obstacle avoidance	VFH [1], APF [13], VFH+ [25], VFH* [26]		

1.2.3 Controller and Robot Position Control

According to the measurements by sensors and the reference values, see Fig. 1.1, the controller adjusts values of the actuator so as to alter the status of the controlled plant.

Robot position control is based on the fact that a controller is considered which focuses on robot positioning, trajectory tracking and so on. Please distinguish this to the general position control concept which usually takes into account a servo position control with emphasize on controlling the pulse inputs.

1.2.4 Why Do We Need Localization?

One of the project goals is to use a group of omnidirectional mobile robots to search a target in an environment and the main guidance mechanism is based on the mechanical Particle Swarm Optimization (PSO) which is an extension from the basic PSO and includes some of robots mechanical properties [23]. This scheme uses the mechanical PSO to generate the search trajectory. Then the robots further perform fine tuning for obstacle avoidance and mutual avoidance locally. A main demand of this method is, that it requires the robots current velocities, the self-best positions as well as the swarm-best positions, i.e., it heavily relies on the pose of each robot. Thus, the localization with acceptable accuracy becomes important.

1.3 Position Measurement and Control

To successfully perform some motions, e.g., the trajectory tracking on mobile robots where the trajectory is generated by a mechanical PSO algorithm, one must also consider the robots localization abilities. Unfortunately, in reality

a controller can not arbitrarily accurate command the robots to the positions and usually the robots lack accurate positioning functionality. In some cases it is even very difficult to obtain a precise position from robots.

Originally Robotinos can only roughly obtain their position information by using motor encoders if used without help from external positioning equipment. Such a measurement, however, can not get meet the requirement. Therefore, this study builds a systematic hybrid strategy for the localization. Combined with control techniques, it enables to drive robot to a desired position within an acceptable accuracy. This section first of all shows the overall scheme for robot position control and then goes on with a detail research of the sensors involved in the measurements for localization.

1.3.1 Overall Position Control Scheme

The overall scheme for the localization and position control of a single Robotino is shown in Fig. 1.2.

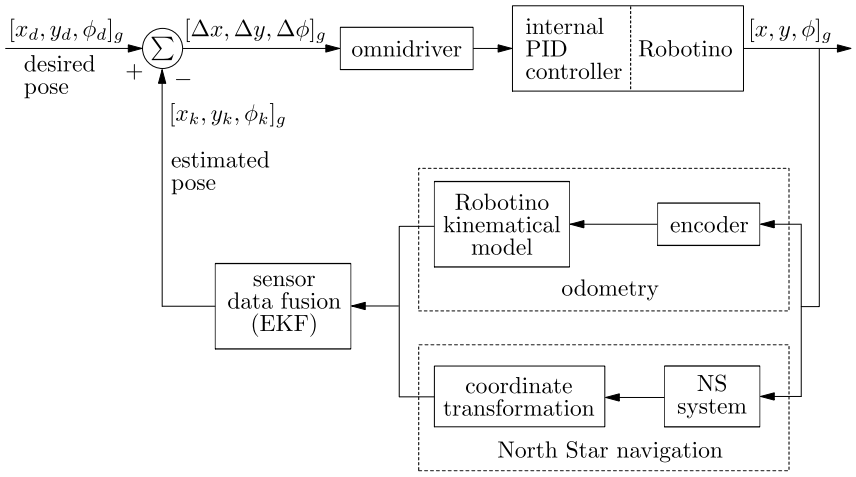


Fig. 1.2 Scheme of accurate localization

The scheme consists of one internal controller and one external controller. First of all, a partially fixed PID controller which can directly control the DC motor is adjusted and used as an internal controller. The robot's 'actual' position and orientation are measured by not only on-body encoders but also an external North Star system. The measured pose is then processed in an extended Kalman filter (EKF). Data fusion in the EKF gives an estimated position and orientation which are compared to the desired. This forms the closed-loop feedback control which contains here also the external controller.

The resulting deviation $[\Delta x \ \Delta y \ \Delta\phi]_g$ goes on to drive the robot. By using such a hybrid strategy the localization accuracy of the robots can be improved. The scheme is valid for all the robots in the group since each robot runs on its own.

1.3.2 State Variable Feedback

From Fig. 1.1 one can see that a proportional feedback control law

$$\boldsymbol{u} = -\boldsymbol{K} \cdot \boldsymbol{y} + \boldsymbol{r} \tag{1.1}$$

is used. This scheme is also known as output feedback. However, the output feedback design has many difficulties for pole placement, see details, e.g., in [4] and [14]. Another basic control scheme is to use all the states as outputs, then it becomes state variable feedback with the control law

$$\boldsymbol{u} = -\boldsymbol{K} \cdot \boldsymbol{x} + \boldsymbol{r}. \tag{1.2}$$

Benefiting from high quality sensors and techniques like Kalman filters and observers, to measure all of the state variables directly or indirectly becomes possible. Most important, the state variable feedback is simple for pole placement, see also in [4] and [14]. In this study, it considers the robots pose (2D position, one orientation) as state variables. Actually, by a reasonable observer, it also can include pose rates. In the following this study will focus on the sensors which are used for state measuring.

1.3.3 Internal Odometer and Its Enhancement

Odometry is the measurement of wheel rotation with use of many different methods that are integrated in the drive system and continually updates with incremental wheel information. The position and orientation then can be determined easily by time integration added to the previously known position.

1.3.3.1 Odometry Mechanism

The Festo Robotino is a holonomic mobile robot which contains three omni drive units with 120° between each. The robot and its main components are shown in Fig. 1.3, the structure of Robotino base and coordinates can be seen in Fig. 1.4.

After simple derivation one can get the kinematic relation between wheel speed and global velocity by

Fig. 1.3 Festo Robotino and its main components

$$\begin{bmatrix} \dot{x}_g(t) \\ \dot{y}_g(t) \\ \dot{\phi}_g(t) \end{bmatrix} = r \begin{bmatrix} -\sin(\phi_g(t)+\alpha_1) & \cos(\phi_g(t)+\alpha_1) & R \\ -\sin(\phi_g(t)+\alpha_2) & \cos(\phi_g(t)+\alpha_2) & R \\ -\sin(\phi_g(t)+\alpha_3) & \cos(\phi_g(t)+\alpha_3) & R \end{bmatrix}^{-1} \cdot \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} \tag{1.3}$$

where r is the wheel's radius, R is the distance from the wheel's center to the center of the robot, and $\omega_i(t)$ $(i = 1, 2, 3)$ are the wheels angular velocities. The geometry angle ϕ_g and α_i $(i = 1, 2, 3)$ are illustrated in Fig. 1.4, and $[\dot{x}_g \ \dot{y}_g \ \dot{\phi}_g]^T$ represents the global velocity of the robot. Odometry can estimate the robots position and orientation over time by

$$\begin{bmatrix} x_g(t_1) \\ y_g(t_1) \\ \phi_g(t_1) \end{bmatrix} = \begin{bmatrix} x_g(t_0) \\ y_g(t_0) \\ \phi_g(t_0) \end{bmatrix} + \int_{t=t_0}^{t_1} \begin{bmatrix} \dot{x}_g(t_0) \\ \dot{y}_g(t_0) \\ \dot{\phi}_g(t_0) \end{bmatrix} dt. \tag{1.4}$$

The so called 'odometry' in this study is based on the robots motor encoders (see the drivetrain of the Robotino in Fig. 1.5) and the kinematics relationships (1.3) and (1.4). The rotation of robot wheels is measured with the highest possible resolution. In each small time step the distance driven by a robot is calculated by (1.4). Thus, the current position of the controlled robot relative to its starting point can be calculated. The time integration is actually calculated as counting the number of beam interruptions caused from the toothed interrupter disc. So, if one considers this in a discrete way together with first order Taylor series expansion, it results in

$$\begin{bmatrix} x_g(k) \\ y_g(k) \\ \phi_g(k) \end{bmatrix} = \begin{bmatrix} x_g(k-1) \\ y_g(k-1) \\ \phi_g(k-1) \end{bmatrix} + \Delta t \, r\boldsymbol{T}(k-1) \cdot \begin{bmatrix} \omega_1(k-1) \\ \omega_2(k-1) \\ \omega_3(k-1) \end{bmatrix} \tag{1.5}$$

where Δt is the size of time step, typically in our case $\Delta t = 0.01$s. The matrix $\boldsymbol{T}(k-1)$ includes the structure matrix and the rotation matrix from robot

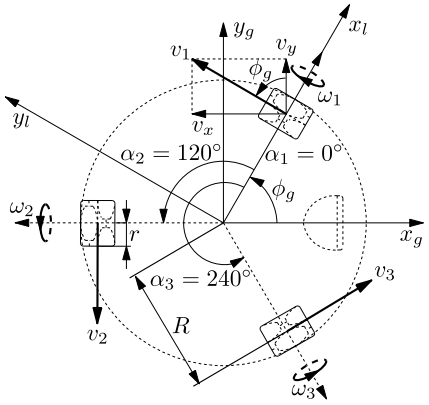


Fig. 1.4 Geometry diagram and kinematic relations of Robotino base

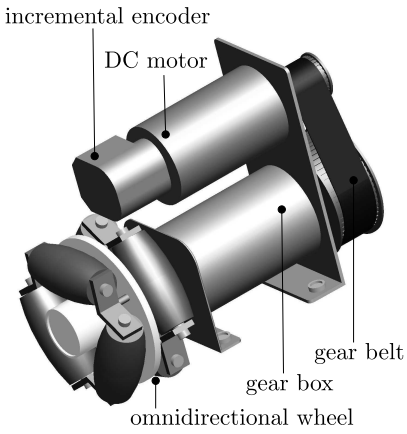


Fig. 1.5 Robotino drivetrain

local coordinates to global coordinates which can be governed by

$$
\begin{aligned}
\boldsymbol{T}(k-1) &= \begin{bmatrix} -\sin(\phi_g(k-1)+\alpha_1) & \cos(\phi_g(k-1)+\alpha_1) & R \\ -\sin(\phi_g(k-1)+\alpha_2) & \cos(\phi_g(k-1)+\alpha_2) & R \\ -\sin(\phi_g(k-1)+\alpha_3) & \cos(\phi_g(k-1)+\alpha_3) & R \end{bmatrix}^{-1} \\
&= \left(\begin{bmatrix} -\sin(\alpha_1) & \cos(\alpha_1) & R \\ -\sin(\alpha_2) & \cos(\alpha_2) & R \\ -\sin(\alpha_3) & \cos(\alpha_3) & R \end{bmatrix} \right. \\
&\quad \left. \cdot \begin{bmatrix} \cos(\phi_g(k-1)) & \sin(\phi_g(k-1)) & 0 \\ -\sin(\phi_g(k-1)) & \cos(\phi_g(k-1)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} \cos(\phi_g(k-1)) & -\sin(\phi_g(k-1)) & 0 \\ \sin(\phi_g(k-1)) & \cos(\phi_g(k-1)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\sin(\alpha_1) & \cos(\alpha_1) & R \\ -\sin(\alpha_2) & \cos(\alpha_2) & R \\ -\sin(\alpha_3) & \cos(\alpha_3) & R \end{bmatrix}^{-1} .
\end{aligned}
$$

1.3.3.2 Odometry Error Correction

Robot odometry is used for pose measurement since it is an internal 'sensor' system and is convenient for practical utilization. However, this method only yields good performance for short distance motions. Errors and especially disturbances will be accumulated and affect the robot's odometry results. This is obvious for long distance travels or under adverse conditions.

There are mainly two kinds of error sources. One kind comes from the robot itself and another one is from outside. The former one includes the errors of robot's unequal wheel radii, unequal wheel distances, misalignment

of wheel angles and so on. These errors can be corrected since one can just measure the actual data from the real robot as precisely as possible and use them for the odometry calculation. Thus, Eq. (1.4) is extended to

$$\begin{bmatrix} x_g(t_1) \\ y_g(t_1) \\ \phi_g(t_1) \end{bmatrix} = \begin{bmatrix} x_g(t_0) \\ y_g(t_0) \\ \phi_g(t_0) \end{bmatrix} + \int_{t=t_0}^{t_1} \left(\boldsymbol{T}_{ex}(t_0) \cdot \begin{bmatrix} (r+e_{r_1})\omega_1(t_0) \\ (r+e_{r_2})\omega_2(t_0) \\ (r+e_{r_3})\omega_3(t_0) \end{bmatrix} \right) dt \quad (1.6)$$

with

$$\boldsymbol{T}_{ex}(t_0) = \begin{bmatrix} -\sin(\phi_g(t_0)+\alpha_1+e_{m_1}) & \cos(\phi_g(t_0)+\alpha_1+e_{m_1}) & R+e_{d_1} \\ -\sin(\phi_g(t_0)+\alpha_2+e_{m_2}) & \cos(\phi_g(t_0)+\alpha_2+e_{m_2}) & R+e_{d_2} \\ -\sin(\phi_g(t_0)+\alpha_3+e_{m_3}) & \cos(\phi_g(t_0)+\alpha_3+e_{m_3}) & R+e_{d_3} \end{bmatrix}^{-1} .$$

Here e_{r_i}, e_{d_i} and e_{m_i} $(i = 1, 2, 3)$, represent small deviations because of unequal wheel radii, unequal wheel distances and misalignment of wheel angles, respectively. A further investigation focusing on this is done at the institute which can be found in [19]. By this way, the error sources from robot itself can be reduced. This kind of odometry correction actually is a fine step of model modification which provides a more accurate odometry calculation.

However, during tests and experiments we noticed that the error sources from outside play a more important role, among them the wheel slippage is a key aspect. This means the robot moves less than the odometer counted because of slippage. Thus, this study also attempts to construct a slippage correction to the odometry calculation. One should know that there are many different kinds of global motions that can be performed by the used robot. Most important, its omnidirectional feature makes it possible to generate coupled motions, e.g., moving in x direction in the global frame probably due to an actuation in y direction together with a spinning in the robot's body frame. So, it is a troublesome task to build a uniform correction factor. However, if one changes the idea to work in the robot's local frame the question becomes easier. This is so, because either in the global or local frame all the movements are the combinations of two basic motion forms. These are the translational move (x and y directions) and the rotation. The important difference is if considered in a local frame, that one specific basic motion form always invokes the same wheel combination (also the same motors combination). Based on this principle, this research focuses on robot's body frame and for each of calculated steps a correction is made before projecting the motion to the global frame.

With this idea, a large number of experiments under different ground conditions need to be done since different kinds of grounds keep different slippery extents, although all of them only allow the same basic motion types. During one motion tests the other types of motion are isolated. After testing, one group of correction factors can be recommended for a specific ground, which in a discrete way can be governed by

$$\begin{bmatrix} \dot{x}_l^c(k) \\ \dot{y}_l^c(k) \\ \dot{\phi}_l^c(k) \end{bmatrix} = \boldsymbol{F}_c \cdot \boldsymbol{S}_l^{-1} \cdot \begin{bmatrix} (r+e_{r_1})\omega_1(k) \\ (r+e_{r_2})\omega_2(k) \\ (r+e_{r_3})\omega_3(k) \end{bmatrix} dt \tag{1.7}$$

where

$$\boldsymbol{F}_c = \begin{bmatrix} f_{c_1} & 0 & 0 \\ 0 & f_{c_2} & 0 \\ 0 & 0 & f_{c_3} \end{bmatrix}, \quad \boldsymbol{S}_l = \begin{bmatrix} -\sin(\alpha_1+e_{m_1}) & \cos(\alpha_1+e_{m_1}) & R+e_{d_1} \\ -\sin(\alpha_2+e_{m_2}) & \cos(\alpha_2+e_{m_2}) & R+e_{d_2} \\ -\sin(\alpha_3+e_{m_3}) & \cos(\alpha_3+e_{m_3}) & R+e_{d_3} \end{bmatrix}.$$

Here $\dot{x}_l^c(k), \dot{y}_l^c(k)$ and $\dot{\phi}_l^c(k)$ are the corrected local frame velocities of robot. From (1.7) one can see that the correction factors neither depend on step size nor the angular velocity of wheels. It is a diagonalized constant correction matrix without coupling from different dimensions. Such form of correction is what we pursue since it then only needs some simple tests to get the correction factors for a specific robot on a specific ground. This study uses a group of correction factors on a carpeted ground, in [19] a relatively smooth ground was also investigated.

Although this slippage correction improves the accuracy of robots odometry, one can not completely avoid all of the error sources. Therefore, one goes on to add other solutions, e.g., the following North Star measurements.

1.3.4 External North Star Measurements

It is advantageous to add another sensor, typically an odometry independent sensor, to attain the robot pose. This kind of standalone data is also used for later performing data fusion in the Kalman filter.

1.3.4.1 The North Star Navigation System

This study uses a novel external positioning sensor, the North Star system. It enables position and heading awareness in mobile robots, and can be applied in many position tracking applications in a variety of indoor situations. It breaks the 'line-of-sight' barrier of previous light beacon systems with simplicity and reliability. The concept of how this system does its measurements is illustrated in Fig. 1.6.

The North Star system is an infrared (IR) light based positioning device. When a projector (see Fig. 1.7) emits IR light which is reflected by the ceiling, the IR detector (see Fig. 1.8) which is mounted on the Robotino can receive the light signals and takes such signals as uniquely identifiable landmarks. By this, the robot can determine its relative position and direction. Of course the robot requires calibration and coordination before starting. This North

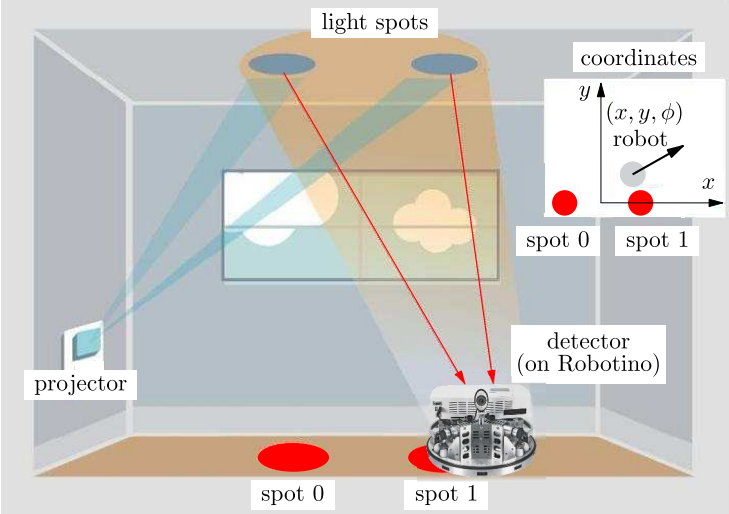


Fig. 1.6 Illustration of North Star pose measurement

Star positioning system is valid for several robots simultaneously by using different channels identified by varied frequency ranges. This also improves the positioning accuracy since it reduces the coordination errors compared to the situation of where each robot has its own global frame.

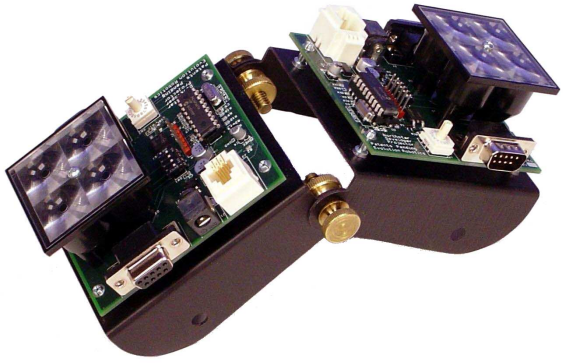


Fig. 1.7 North Star projector (two emitter clusters)

Fig. 1.8 North Star detector

1.3.4.2 North Star Calibration

For odometry, there is no special calibration process since it only needs to refresh fast enough. However, the North Star needs calibration including building the coordinates before it can be used for measuring the robot pose. If several robots are used, they can share the same global coordinates. They need to gain the relationship between the changed pose and the changed infrared lights. Therefore, the calibration is done by performing a desired motion with exactly known position and orientation changing traces. Meanwhile the North Star detector which is mounted on the Robotino records the infrared light changes corresponding to the robot motion. With triangulation calculus, the calibration purpose is achieved. The recording and calculus processes can be done by the North Star system itself, so one only needs to design a specific calibration motion and tell to the North Star.

1.4 Localization through Sensor Data Fusion in Extended Kalman Filters

Up to now, this study uses two methods to measure robot pose. Now the questions are, e.g, how to use the gained pose information for robot position control, how to weight the measurements from different sensors? From another side, we want to improve the position accuracy of robots while not putting too much burden on the external hardware. Thus, using methods to perform sensor data fusion will be a good choice.

1.4.1 Sensor Data Fusion

Both the odometry and NS system provide position and orientation measurements of the robot. However, their accuracies are not very satisfactory even with further correction. Besides of the errors and uncertainties from odometry, the NS system will inevitably involve noises, too. For example, the North Star detector will exhibit nonlinear absolute localization responses and this nonlinearity will increase as the IR projector's light spots move away from the center of the detectors view field. Additionally, position and orientation errors will also be produced when the detector is tilted from the plane of localization environment, e.g., the floor.

Due to above reasons, it is necessary to perform a pose correction based on the information from odometry and NS system. The most commonly used method in robotics to cope with this problem is the Kalman filter.

1.4.2 Kalman Filter and Extended Kalman Filter

In 1960, Rudolph E. Kalman published his famous paper [12] in which a recursive solution to the discrete-data filtering problem was described. Kalman filtering is a recursive process, rather than an electronic filter. In most cases, the Kalman filter is used for estimating the state of processes. A Kalman filter is an estimator for what is called linear-quadratic-problem, which is the problem of estimating the instantaneous 'state' of a linear dynamic system perturbed by white noise [8]. Kalman filters have been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. This kind of estimation fits very well to the localization and position control for robot where the pose state of robot is a critical requirement. The robot obtains its poses through continually measuring by internal odometry and external North Star system, but neither the odometry nor North Star alone has a satisfactory accuracy since both of them contain noises. Therefore, it needs a method to weight the measurements and tries to fuse the information from both sides of measurements including the reduction of the noises affection.

Kalman filters address the general problem of trying to estimate the state $\boldsymbol{x} \in \Re^n$ of a discrete time controlled process that is governed by the linear difference equation

$$\boldsymbol{x}(k) = \boldsymbol{A} \cdot \boldsymbol{x}(k-1) + \boldsymbol{B} \cdot \boldsymbol{u}(k-1) + \boldsymbol{w}(k-1) \tag{1.8}$$

with a current measurement $\boldsymbol{z} \in \Re^m$ which is

$$\boldsymbol{z}(k) = \boldsymbol{H} \cdot \boldsymbol{x}(k) + \boldsymbol{v}(k). \tag{1.9}$$

Here k is the time step, the $n \times n$ matrix $\boldsymbol{A}$ relates the state at the previous step $k-1$ to the state of at current step k and depends on the odometry integration equation, see Eq. (1.4). Here n is the dimensions of the system. In practical implementation $\boldsymbol{A}$ might change with each time step, e.g., the robot's odometry update in this study since the orientation ϕ_g in matrix $\boldsymbol{T}$ in Eq. (1.5) might change. The formulation of $\boldsymbol{A}$ for the Robotinos can be seen in the following nonlinear part. The $n \times l$ matrix $\boldsymbol{B}$ relates the optional control input $\boldsymbol{u} \in \Re^l$ to the state $\boldsymbol{x}$ while the $m \times n$ matrix $\boldsymbol{H}$ relates the state $\boldsymbol{x}$ to the measurement $\boldsymbol{z}$. For this study the state $\boldsymbol{x}$ includes the x position, y position and orientation of the robot which are the states concerned by our investigation. Please distinguish to the mechanical systems state vectors in which usually the velocities are also included. Furthermore, here the odometry uses the internal encoders information to update, thus, there is no outside control input, i.e., $\boldsymbol{B}$ can vanish. The form of matrix $\boldsymbol{H}$ depends on the external sensor used for measurement, usually it contains the information like signal conversion, scaling, and so on. In this study, the external measurement is performed by the North Star system which directly

provides the pose result thus $\boldsymbol{H}$ turns to be an unit matrix. The random variables $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ represent the process and measurement white noise, respectively. These noises satisfy the conditions of independent and according to normal probability distributions

$$\begin{aligned} p(\boldsymbol{w}) &\sim N(0, \boldsymbol{Q}), \\ p(\boldsymbol{v}) &\sim N(0, \boldsymbol{R}). \end{aligned} \tag{1.10}$$

Matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the process noise covariance and measurement noise covariance, respectively, and both of them might change. In our case, $\boldsymbol{Q}$ essentially depends on the noise of reading the motor speeds and $\boldsymbol{R}$ depends on the measurement noise from North Star. Robot's odometry contains an integration function. Qualifying Kalman filter to be used for the robot pose estimation needs to make a discretization of the odometry update as shown in Eq. (1.5).

The Kalman filter's recursive processes include not only the update of the differential equation (1.8) but also of the measurements and noises. This is in line with the concept of sensor data fusion since it needs to consider the information from different sensors. For our robot pose estimate case, one sensor is the internal odometry and another is the external North Star system. So, if the robot's pose update (estimate) is according to the Kalman recursive processes, it can fuse the useful information from both odometry and North Star. This is also the main reason why we use Kalman filters for the robot pose estimate.

The recursive processes of Kalman filters mainly consist of two parts, one part is the prediction and another part is the correction. The former one projects the current state estimate ahead over time and estimates the error covariance to obtain the a-priori estimate, noted as $\boldsymbol{x}(k)^-$ for step k with the help of a 'super minus'. The correction part is responsible for incorporating a new measurement into the a-priori estimate so as to correct the projected estimate and obtain the a-posteriori estimate, noted as $\hat{\boldsymbol{x}}(k)$. A set of specific mathematical equations can be built to describe the prediction process by

$$\text{prediction} \begin{cases} \boldsymbol{x}(k)^- = \boldsymbol{A} \cdot \hat{\boldsymbol{x}}(k-1) + \boldsymbol{B} \cdot \boldsymbol{u}(k-1) \\ \boldsymbol{P}(k)^- = \boldsymbol{A} \cdot \boldsymbol{P}(k-1) \cdot \boldsymbol{A}^T + \boldsymbol{Q}(k-1) \end{cases} \tag{1.11}$$

and the correction by

$$\text{correction} \begin{cases} \boldsymbol{K}(k) = \boldsymbol{P}(k)^- \cdot \boldsymbol{H}^T \cdot (\boldsymbol{H} \cdot \boldsymbol{P}(k)^- \cdot \boldsymbol{H}^T + \boldsymbol{R}(k))^{-1} \\ \boldsymbol{P}(k) = (\boldsymbol{I} - \boldsymbol{K}(k) \cdot \boldsymbol{H}) \cdot \boldsymbol{P}(k)^- \\ \hat{\boldsymbol{x}}(k) = \boldsymbol{x}(k)^- + \boldsymbol{K}(k) \cdot (\boldsymbol{z}(k) - \boldsymbol{H} \cdot \boldsymbol{x}(k)^-) \end{cases} . \tag{1.12}$$

Except the notations of $\boldsymbol{P}, \boldsymbol{K}$ and $\boldsymbol{I}$, all others used in (1.11) and (1.12) have the same definitions as in (1.8), (1.9) and (1.10). Here $\boldsymbol{I}$ is an unit matrix, $\boldsymbol{P}(k)^-$ is the a-priori estimate error covariance which is defined by

$$\boldsymbol{P}(k)^{-} = \boldsymbol{E}\left[\boldsymbol{e}(k)^{-}(\boldsymbol{e}(k)^{-})^{T}\right] \tag{1.13}$$

where $\boldsymbol{E}$ is the operation symbol for mathematical expectation and $\boldsymbol{e}(k)^{-} \equiv \boldsymbol{x}(k) - \boldsymbol{x}(k)^{-}$. The similar definition is given to $\boldsymbol{P}(k)$. However, the prediction and correction of $\boldsymbol{P}$ during the Kalman recursive processes are according to the respective equations in (1.11) and (1.12). The expressions in Eqs. (1.11) and (1.12) for $\boldsymbol{P}$ are equivalent to the definition in Eq. (1.13) when the filter uses the optimal Kalman gain which is nearly always the case in practice, but they are in a clearer recursive form. Another important item in (1.12) is the so called Kalman gain or Kalman blending factor $\boldsymbol{K}$ which is used for guaranteeing to minimize the a-posteriori error covariance of $\boldsymbol{P}(k)$. Furthermore, the Kalman gain is also used for weighting the a-priori estimate and the measurement as shown in the a-posteriori equation. Specifically for this study, $\boldsymbol{K}$ is used for weighting the pose estimates from odometry and North Star. One form for the iteration of $\boldsymbol{K}$ is shown in (1.12), for more details see [11].

Originally, Kalman filters are designed for linear processes and the external measurement also should be linear. However, most practical problems are nonlinear, i.e., the system differential equation (1.8) becomes

$$\boldsymbol{x}(k) = \boldsymbol{f}\left(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), \boldsymbol{w}(k-1)\right). \tag{1.14}$$

The measurement equation (1.9) is then governed by

$$\boldsymbol{z}(k) = \boldsymbol{h}\left(\boldsymbol{x}(k), \boldsymbol{v}(k)\right). \tag{1.15}$$

Here $\boldsymbol{f}$ is the nonlinear function which relates the state at step $k-1$ to step k while $\boldsymbol{h}$ is the nonlinear measurement function. Something akin to the Taylor series expansion, researchers linearize the estimation by using partial derivatives (Jacobian matrices) of the process and measurement. Thus, the extended Kalman filter (EKF) was developed, see details in [8]. Finally, a set of mathematical equations for the EKF can be described as

$$\text{prediction} \begin{cases} \boldsymbol{x}(k)^{-} = \boldsymbol{f}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), \boldsymbol{0}) \\ \boldsymbol{P}(k)^{-} = \bar{\boldsymbol{A}}(k) \cdot \boldsymbol{P}(k-1) \cdot \bar{\boldsymbol{A}}^{T}(k) \\ \qquad\quad + \boldsymbol{W}(k) \cdot \boldsymbol{Q}(k-1) \cdot \boldsymbol{W}^{T}(k) \end{cases}, \tag{1.16}$$

$$\text{correction} \begin{cases} \boldsymbol{K}(k) = \boldsymbol{P}(k)^{-} \cdot \bar{\boldsymbol{H}}^{T}(k) \cdot (\bar{\boldsymbol{H}}(k) \cdot \boldsymbol{P}(k)^{-} \cdot \bar{\boldsymbol{H}}^{T}(k) \\ \qquad\quad + \boldsymbol{V}(k) \cdot \boldsymbol{R}(k) \cdot \boldsymbol{V}^{T}(k))^{-1} \\ \boldsymbol{P}(k) = (\boldsymbol{I} - \boldsymbol{K}(k) \cdot \bar{\boldsymbol{H}}(k)) \cdot \boldsymbol{P}(k)^{-} \\ \hat{\boldsymbol{x}}(k) = \boldsymbol{x}(k)^{-} + \boldsymbol{K}(k) \cdot (\boldsymbol{z}(k) - \boldsymbol{h}(\boldsymbol{x}(k)^{-}, \boldsymbol{0})) \end{cases}. \tag{1.17}$$

Here $\bar{\boldsymbol{A}}(k), \bar{\boldsymbol{H}}(k), \boldsymbol{W}(k)$ and $\boldsymbol{V}(k)$ are all Jacobian matrices containing partial derivatives

$$\bar{A}_{i,j}(k) = \frac{\partial f_i}{\partial x_j}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), \boldsymbol{0}), \tag{1.18}$$

$$\bar{H}_{i,j}(k) = \frac{\partial h_i}{\partial x_j}(\boldsymbol{x}(k)^-, \boldsymbol{0}), \tag{1.19}$$

$$W_{i,j}(k) = \frac{\partial f_i}{\partial w_j}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), \boldsymbol{0}), \tag{1.20}$$

$$V_{i,j}(k) = \frac{\partial h_i}{\partial v_j}(\boldsymbol{x}(k)^-, \boldsymbol{0}). \tag{1.21}$$

Usually the nonlinear function $\boldsymbol{f}$ is computationally expensive. Thus, researchers use approximation to the nonlinear processes and measurements.

1.4.3 Robotino Sensor Data Fusion for Localization

The investigated robot pose estimation is a nonlinear process thus it uses an extended Kalman filter. Using Taylor series expansion, we can perform the a-priori estimate according to Eq. (1.5). Additionally, if one writes the wheels angular velocities together with wheels radii and geometric structure of Robotino, then the a-priori estimate can be rearranged to

$$\begin{bmatrix} x_g(k)^- \\ y_g(k)^- \\ \phi_g(k)^- \end{bmatrix} = \begin{bmatrix} \hat{x}_g(k-1) \\ \hat{y}_g(k-1) \\ \hat{\phi}_g(k-1) \end{bmatrix} + \Delta t \begin{bmatrix} \cos(\hat{\phi}_g(k-1)) & -\sin(\hat{\phi}_g(k-1)) & 0 \\ \sin(\hat{\phi}_g(k-1)) & \cos(\hat{\phi}_g(k-1)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_l^c(k-1) \\ \dot{y}_l^c(k-1) \\ \dot{\phi}_l^c(k-1) \end{bmatrix} \tag{1.22}$$

where $\dot{x}_l^c(k-1), \dot{y}_l^c(k-1)$ and $\dot{\phi}_l^c(k-1)$ are the last step's velocities given in the robot body frame. It is necessary to emphasize, here the used velocities in the local frame are already corrected according to Eq. (1.7). For the extended Kalman filter to perform the recursion for robot pose estimate one needs to know the functions $\boldsymbol{f}$ and $\boldsymbol{h}$ explicitly. Based on Eq. (1.22), one can write the three dimensions individually which yields

$$f_x = x_g(k)^- = \hat{x}_g(k-1) + \Delta t \cos(\hat{\phi}_g(k-1))\dot{x}_l^c(k-1) \tag{1.23}$$
$$-\Delta t \sin(\hat{\phi}_g(k-1))\dot{y}_l^c(k-1) + \cancelto{0}{0 \cdot \Delta t \dot{\phi}_l^c(k-1)},$$

$$f_y = y_g(k)^- = \hat{y}_g(k-1) + \Delta t \sin(\hat{\phi}_g(k-1))\dot{x}_l^c(k-1) \tag{1.24}$$
$$+\Delta t \cos(\hat{\phi}_g(k-1))\dot{y}_l^c(k-1) + \cancelto{0}{0 \cdot \Delta t \dot{\phi}_l^c(k-1)},$$

$$f_\phi = \phi_g(k)^- = \hat{\phi}_g(k-1) + \cancelto{0}{0 \cdot \Delta t \dot{x}_l^c(k-1)} \tag{1.25}$$
$$+\cancelto{0}{0 \cdot \Delta t \dot{y}_l^c(k-1)} + \Delta t \dot{\phi}_l^c(k-1).$$

Next, for obtaining the a-priori estimate of error covariance $\boldsymbol{P}(k)^-$ one first needs to calculate Jacobian matrices $\bar{\boldsymbol{A}}(k)$ and $\boldsymbol{W}(k)$ according to Eqs. (1.18) and (1.20) combined with the process functions of Eqs. (1.23)-(1.25). In our application this yields

$$\bar{\boldsymbol{A}}(k) = \left(\bar{A}_{i,j}(k)\right)_{3\times 3} = \begin{bmatrix} \frac{\partial f_x}{\partial \hat{x}_g(k-1)} & \frac{\partial f_x}{\partial \hat{y}_g(k-1)} & \frac{\partial f_x}{\partial \hat{\phi}_g(k-1)} \\ \frac{\partial f_y}{\partial \hat{x}_g(k-1)} & \frac{\partial f_y}{\partial \hat{y}_g(k-1)} & \frac{\partial f_y}{\partial \hat{\phi}_g(k-1)} \\ \frac{\partial f_\phi}{\partial \hat{x}_g(k-1)} & \frac{\partial f_\phi}{\partial \hat{y}_g(k-1)} & \frac{\partial f_\phi}{\partial \hat{\phi}_g(k-1)} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & -\Delta t \sin(\hat{\phi}_g(k-1))\dot{x}_l^c(k-1) - \Delta t \cos(\hat{\phi}_g(k-1))\dot{y}_l^c(k-1) \\ 0 & 1 & \Delta t \cos(\hat{\phi}_g(k-1))\dot{x}_l^c(k-1) - \Delta t \sin(\hat{\phi}_g(k-1))\dot{y}_l^c(k-1) \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.26}$$

$$\boldsymbol{W}(k) = (W_{i,j}(k))_{3\times 3} = \begin{bmatrix} \frac{\partial f_x}{\partial \dot{x}_l^c(k-1)} & \frac{\partial f_x}{\partial \dot{y}_l^c(k-1)} & \frac{\partial f_x}{\partial \dot{\phi}_l^c(k-1)} \\ \frac{\partial f_y}{\partial \dot{x}_l^c(k-1)} & \frac{\partial f_y}{\partial \dot{y}_l^c(k-1)} & \frac{\partial f_y}{\partial \dot{\phi}_l^c(k-1)} \\ \frac{\partial f_\phi}{\partial \dot{x}_l^c(k-1)} & \frac{\partial f_\phi}{\partial \dot{y}_l^c(k-1)} & \frac{\partial f_\phi}{\partial \dot{\phi}_l^c(k-1)} \end{bmatrix} \tag{1.27}$$
$$= \begin{bmatrix} \Delta t \cos(\hat{\phi}_g(k-1)) & -\Delta t \sin(\hat{\phi}_g(k-1)) & 0 \\ \Delta t \sin(\hat{\phi}_g(k-1)) & \Delta t \cos(\hat{\phi}_g(k-1)) & 0 \\ 0 & 0 & \Delta t \end{bmatrix}.$$

According to Eq. (1.20) the entries of $\boldsymbol{W}$ are the partial derivatives of $\boldsymbol{f}$ with respect to $\boldsymbol{w}$. However, practically $\boldsymbol{w}$ is always calculated through the variables of the noise sources. In robot navigation, if odometry is used as the internal pose sensor, then usually the considered noises are contained in

robot's body velocities. In our case, originally the noise is mainly from robot's motor speed, however it is transferred and reflected on robot velocity. Thus, in our applications the entries of matrix $\boldsymbol{W}$ become the partial derivatives of $\boldsymbol{f}$ with respect to $\dot{x}_l^c(k-1), \dot{y}_l^c(k-1)$ and $\dot{\phi}_l^c(k-1)$. Of course here the noises don't include the noises which are eliminated by odometry correction.

With $\bar{\boldsymbol{A}}(k)$ and $\boldsymbol{W}(k)$ one can calculate the prediction covariance

$$\boldsymbol{P}(k)^- = \bar{\boldsymbol{A}}(k) \cdot \boldsymbol{P}(k-1) \cdot \bar{\boldsymbol{A}}^T(k) + \boldsymbol{W}(k) \cdot \boldsymbol{Q}(k-1) \cdot \boldsymbol{W}^T(k). \tag{1.28}$$

Here $\boldsymbol{Q}(k-1)$ is the process noise covariance, see Section 1.4.5 for details. After the calculation of $\boldsymbol{P}(k)^-$ the extended Kalman filter gets into the processes of correction, i.e., the correction of the robot pose will be performed. For this the Kalman gain matrix must be calculated resulting in

$$\boldsymbol{K}(k) = \boldsymbol{P}(k)^- \cdot (\boldsymbol{P}(k)^- + \boldsymbol{R}(k))^{-1} \tag{1.29}$$

since both $\bar{\boldsymbol{H}}(k)$ and $\boldsymbol{V}(k)$ are unit matrices in this application, see Section 1.4.2 and Eq. (1.21). Here, the matrix $\boldsymbol{R}(k)$ is the measure noise covariance and it is one of the uncertainty sources. In Section 1.4.5 it will be determined.

Finally it performs the a-posteriori pose correction which will in each step be feed back to the desired pose for position control

$$\begin{bmatrix} \hat{x}_g(k) \\ \hat{y}_g(k) \\ \hat{\phi}_g(k) \end{bmatrix} = \begin{bmatrix} x_g(k)^- \\ y_g(k)^- \\ \phi_g(k)^- \end{bmatrix} + \boldsymbol{K}(k) \cdot \left(\begin{bmatrix} z_x(k) \\ z_y(k) \\ z_\phi(k) \end{bmatrix} - \begin{bmatrix} x_g(k)^- \\ y_g(k)^- \\ \phi_g(k)^- \end{bmatrix} \right). \tag{1.30}$$

For the next step it updates the prediction covariance, here simplified to

$$\boldsymbol{P}(k) = (\boldsymbol{I} - \boldsymbol{K}(k)) \cdot \boldsymbol{P}(k)^-. \tag{1.31}$$

Here z_x, z_y and z_ϕ are the three dimensions of pose information measured by the North Star system and $\boldsymbol{I}$ is a 3×3 unit matrix. From a mathematical point of view, in Eq. (1.30) $\boldsymbol{K}(k)$ is just a scaling factor to the two items, the $\boldsymbol{x}(k)^-$ and $\boldsymbol{z}(k)$. However, physically the Kalman gain weights the two sensors measurements, the odometry and North Star, by considering both of the sensors noises and uncertainties. Through this method, the sensors data are fused into the a-posteriori estimate of $\hat{\boldsymbol{x}}_g(k)$ having information from both odometry and North Star.

1.4.4 Data Flow Analysis

It is necessary to further clarify the data flows during this robot position control which is based on sensor data fusion in EKF. When the robot receives motion command for driving along one desired trajectory it immediately calls

the algorithm which is listed in Algorithm 1. Here the trajectory consists of many control points $\boldsymbol{x}_d(i)$ in which i is the index for the position points on the desired trajectory. This study tries to design the algorithm for general purpose. Specifically speaking, if the position control purpose is only for the robot to reach a final position and it doesn't care about the robot's trajectory, then one can directly give the final desired point $\boldsymbol{x}_d$(final) to the robot. Otherwise, give the position points on the desired trajectory one by one to the robot which means the robot should traverse all of these points so as to track this trajectory. The more position points are given, the more accurate trajectory will be performed by the robot, but of course within the restrictions of computational power and hardware.

Algorithm 1 Robot position control based on sensor data fusion in EKF

/* initialize: get system information and initial conditions (including robot initial pose $\hat{\boldsymbol{x}}_g(0)$, read in initial motor speed $[\omega_{m_1}(0), \omega_{m_2}(0), \omega_{m_3}(0)]$ and EKF start up conditions $\boldsymbol{P}(0), \boldsymbol{Q}(0), \boldsymbol{R}(0)$), specify time step Δt, step $k = 0$, control point notation $i = 1$, calculate initial deviation $\Delta\boldsymbol{x}_g(0)$, define tolerance threshold ϵ for position error*/
while $\Delta\boldsymbol{x}_g(k) > \epsilon$ or $i <$ final **do**
  if $\Delta\boldsymbol{x}_g(k) \leq \epsilon$ **then**
    $i = i + 1$
    input : the new control point $\boldsymbol{x}_d(i)$ from the desired trajectory
    calculate the new pose deviation $\Delta\boldsymbol{x}_g(k) = \boldsymbol{x}_d(i) - \hat{\boldsymbol{x}}_g(k)$
  end if
  drive (or compensate) the robot pose due to $\Delta\boldsymbol{x}_g(k)$
  update EKF step index $k = k + 1$
  convert last step's motor speeds to wheels speeds $[\omega_1(k-1), \omega_2(k-1), \omega_3(k-1)]$
  calculate robot velocity in body coordinates, including odometry correction due to Eq. (1.7)
  perform the a-priori pose estimation $\boldsymbol{x}_g(k)^-$ by odometry, Eq. (1.22), and store the current robot velocity $\dot{\boldsymbol{x}}_g(k)$
  calculate Jacobian matrices $\boldsymbol{A}(k)$ and $\boldsymbol{W}(k)$ by Eqs. (1.26) and (1.27)
  if $k = 1$ **then**
    use $\boldsymbol{Q}(0)$ as the covariance of process noise
  else
    read in last step's motor speeds, update the covariance of process noise $\boldsymbol{Q}(k-1)$ by Eqs. (1.32) and (1.33)
  end if
  calculate prediction covariance $\boldsymbol{P}(k)^-$, Eq. (1.28)
  update the covariance of measurement noise $\boldsymbol{R}(k)$, Eq. (1.34)
  calculate Kalman gain matrix $\boldsymbol{K}(k)$, Eq. (1.29)
  read in the external measurement $\boldsymbol{z}(k)$ from North Star system
  perform the a-posteriori pose correction, get $\hat{\boldsymbol{x}}_g(k)$ by Eq. (1.30)
  compare the EKF estimated pose $\hat{\boldsymbol{x}}_g(k)$ with the control point $\boldsymbol{x}_d(i)$, refresh deviation $\Delta\boldsymbol{x}_g(k)$
  update prediction covariance $\boldsymbol{P}(k)$ for next step, Eq. (1.31)
  get the three motors current speeds $[\omega_{m_1}(k), \omega_{m_2}(k), \omega_{m_3}(k)]$
end while
output : final pose $\hat{\boldsymbol{x}}_g(k)$ and velocity $\dot{\boldsymbol{x}}_g(k)$ (the information provided for mechanical PSO's iteration in our upper level algorithm)

Algorithm 1 is the pseudo-code for our experiments. In addition to the EKF algorithm itself, here the processes for position control are also shown which are directly oriented to the implementation and involved hardware.

1.4.5 Uncertainty Analysis

There are many kinds of uncertainties during the data fusion for robot localization and position control, some of them are known and some are not. In this study, basically the uncertainties come from two parts. One part is the robot and its motion, i.e., the values of physical parameters, uneven terrain, slippery floor and so on. Another kind of uncertainty is from the sensor measurement data. There are no specific methods (formulae or procedures) to determine the covariance matrices $\boldsymbol{Q}(k)$ and $\boldsymbol{R}(k)$. The relatively effective way is by trials and relying on experience.

First, it is important to choose a reasonable initial value. Here, the process noise covariance $\boldsymbol{Q}(k)$ is mainly due to the robot's velocity noise which is essentially from the robot's motors. As such, it is possible to get the basic noises information from the DC motors data sheet and some simple tests. Such tests can be performed by giving a planned constant motor speed value to one specific motor, and let the motor run for a long time until we consider it is running stable. Then, we let the encoder read back the current actual value of motor speed which will be used for odometry calculation. By comparing the read in actual value and the planned value one can get an idea about the noise range of the motor. With the drivetrain relationships, the noise from the motors can be converted to the global pose noise which is formulated as

$$\boldsymbol{Q}(k) = c_p \hat{\boldsymbol{Q}}(k) \tag{1.32}$$

with

$$\begin{aligned}\hat{\boldsymbol{Q}}(k) &= \boldsymbol{T}_{\text{trans}} \\ &\quad \cdot \begin{bmatrix} \operatorname{cov}(\boldsymbol{N}_1(k), \boldsymbol{N}_1(k)) & 0 & 0 \\ 0 & \operatorname{cov}(\boldsymbol{N}_2(k), \boldsymbol{N}_2(k)) & 0 \\ 0 & 0 & \operatorname{cov}(\boldsymbol{N}_3(k), \boldsymbol{N}_3(k)) \end{bmatrix} \\ &= \boldsymbol{T}_{\text{trans}} \cdot \begin{bmatrix} \mathrm{D}(\boldsymbol{N}_1(k)) & 0 & 0 \\ 0 & \mathrm{D}(\boldsymbol{N}_2(k)) & 0 \\ 0 & 0 & \mathrm{D}(\boldsymbol{N}_3(k)) \end{bmatrix}\end{aligned} \tag{1.33}$$

where c_p is a scaling factor, $\boldsymbol{N}_i(k)(i = 1, 2, 3)$ is a vector which contains the i-th motor's process noises sequence up to the current step while $\operatorname{cov}(\boldsymbol{N}_i(k), \boldsymbol{N}_i(k))$ is the i-th motor's noise covariance. The scalers $\mathrm{D}(\boldsymbol{N}_i(k))$ represent the variance and mathematically $\operatorname{cov}(\boldsymbol{N}_i(k), \boldsymbol{N}_i(k)) = \mathrm{D}(\boldsymbol{N}_i(k))$.

Here $\boldsymbol{T}_{\text{trans}}$ is a transformation matrix which includes the information of time step Δt and the conversion from motor speeds to robot's global velocities.

We still left the question of how to get the noises vector $\boldsymbol{N}_i(k)$. By simple tests one already knows the range of motor noise, this can be used for calculating the initial value $\boldsymbol{Q}(k)$. However, the motor noise (velocity noise) counted into the robot odometry calculation is changing during the robot's movement. So the more precise way is to update it in each step. This is done by comparing the real time read in motor speed with the nominal control point motor speed which is computed by the current pose deviation $\Delta \boldsymbol{x}_g(k)$ and inverse kinematics. Here the nominal motor speed actually is the command value one gives to the motor controller. By this the motor process noises can be estimated. Worth to be noticed here it only obtains the execution process noises of the motors, and the encoders are not possible to count the uncertainties like the wheel slippage, so this part of noise without consideration of slippage noise. Wheels slippage noise partially is handled (eliminated) by the odometry correction, and the residual part is transmitted finally to the robot velocity noise together with motor execution noise. Other noises and uncertainties can be considered in the scaling factor c_p which is convenient to be adjusted.

From Eqs. (1.32) and (1.33) one can see that the motor covariances are assumed to be diagonal, however its contribution to the EKF pose estimation is not in a diagonal way. One reason is its transformation matrix $\boldsymbol{T}_{\text{trans}}$, and another reason is from the second item of Eq. (1.28) where the Jacobian matrix $\boldsymbol{W}$ and its transpose couple the different dimensions of $\boldsymbol{Q}(k)$. This is actually physically reasonable.

Akin to the process noise covariance, the measure noise covariance in this application is governed by

$$\boldsymbol{R}(k) = c_m \begin{bmatrix} \operatorname{cov}(\boldsymbol{X}_1(k), \boldsymbol{X}_1(k)) & \operatorname{cov}(\boldsymbol{X}_1(k), \boldsymbol{X}_2(k)) & \operatorname{cov}(\boldsymbol{X}_1(k), \boldsymbol{X}_3(k)) \\ \operatorname{cov}(\boldsymbol{X}_2(k), \boldsymbol{X}_1(k)) & \operatorname{cov}(\boldsymbol{X}_2(k), \boldsymbol{X}_2(k)) & \operatorname{cov}(\boldsymbol{X}_2(k), \boldsymbol{X}_3(k)) \\ \operatorname{cov}(\boldsymbol{X}_3(k), \boldsymbol{X}_1(k)) & \operatorname{cov}(\boldsymbol{X}_3(k), \boldsymbol{X}_2(k)) & \operatorname{cov}(\boldsymbol{X}_3(k), \boldsymbol{X}_3(k)) \end{bmatrix} \tag{1.34}$$

where

$$\begin{aligned} \boldsymbol{X}_1(k) &= \{\Delta x_g(1), \Delta x_g(2), \cdots, \Delta x_g(k)\}, \\ \boldsymbol{X}_2(k) &= \{\Delta y_g(1), \Delta y_g(2), \cdots, \Delta y_g(k)\}, \\ \boldsymbol{X}_3(k) &= \{\Delta \phi_g(1), \Delta \phi_g(2), \cdots, \Delta \phi_g(k)\}, \end{aligned}$$

are the corresponding noise sequences of North Star measured x, y positions and ϕ orientations, c_m is a scaling factor. Here $\boldsymbol{R}(k)$ directly contributes in a coupled way (non-diagonal). Now the question is left how to get the noise (error) sequences $[\Delta x_g(i), \Delta y_g(i), \Delta \phi_g(i)], (i = 1, 2, \cdots, k)$, which can basically describe the noise during the North Star measurements. One gets

help from the North Star data sheets. Here one takes the orientation noise as an example, see Fig. 1.9.

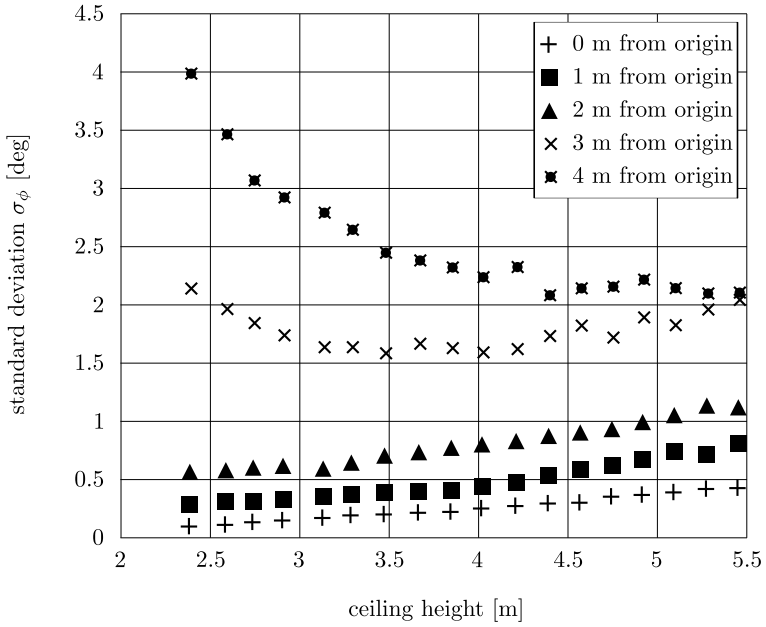


Fig. 1.9 The North Star orientation noise when used 4 LEDs in projector (data from information of Evolution Robotics, Inc.)

Figure 1.9 shows the standard deviation of the orientation errors of the North Star system with different ceiling height and different distances from the origin. Usually, in a specific experiment room with a fixed ceiling height, the robot moving in different locations gains different position and orientation noises, so the measure noise covariance is changing. As we can see from Fig. 1.9, the standard deviation of orientation error changes approximately from 0.15° to 2.8° as the distance changes from 0 to 4 meters in a room with a ceiling around 3 meters high. Obviously, from Fig. 1.9 one can see that this is a nonlinear relationship. However, such trends can be represented by a polynomial interpolated curve. The input of this curve is the current distance from the robot to the origin, in this case we get it from $\boldsymbol{x}_g(k)^-$. By this the sequence of measure noise for robot orientation is obtained. With the current noises sequence, one can calculate the noises covariance easily. Similarly, the covariances for position noises can be obtained, too.

Both $\boldsymbol{Q}(k)$ and $\boldsymbol{R}(k)$ are updated in each step during the EKF recursion. This is different to other traditional EKF applications where usually the covariance matrices for process noise and measure noise are fixed. Refreshing them in each step gives us a relatively closer value to the true noises.

1.5 Experiments and Results Analysis

In this section the previously described method is verified on a real physical Robotino robot, and then the obtained results are compared to the results from motions without performing sensor data fusion, including no feedback and only odometry feedback position control. Additionally, the method presented by this study is also briefly compared to the traditional navigation method, the simultaneously localization and mapping (SLAM).

1.5.1 Experimental Setup

1.5.1.1 Environment Requirements

The experiments are performed in an indoor environment which has a carpeted, flat ground. The room ceiling is around 3 meters high and it's better to keep few objects in the room because we want to reduce the reflecting interferences from objects. Furthermore, it is also necessary to keep away as good as possible light sources including the sun light. A representative experimental environment is illustrated in Fig. 1.10.

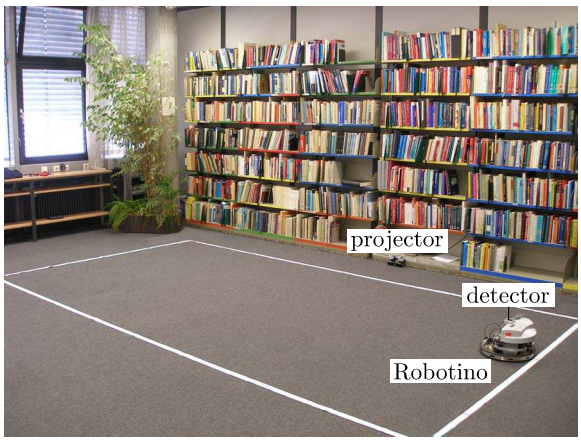


Fig. 1.10 Environment for localization and position control experiments

1.5.1.2 Experiment Arrangement

Three groups of experiments are designed in which one group is done without position feedback control and the other two groups are all performed with closed loop feedback control. The open loop group uses an input-state

equations-output style. The second group of experiments only uses the corrected odometry for measurement, and the measured results are feed back for position control. For comparisons, the third group is performed with sensor data fusion using an extended Kalman filter where the pose information is not only from the odometry but also from the North Star system. Then, the estimated pose is used for the position feedback control. Each group of experiments is organized with 6 kinds of motions and each motion gets 20 runs. Complex motions are combined from such basic types. The motion types and experiment dimensions are illustrated in Fig. 1.11.

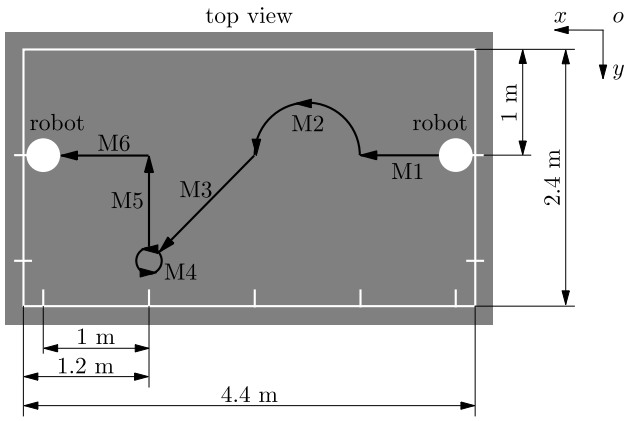


Fig. 1.11 Motion types

The 6 kinds of motions are: M1 move forward 1 meter in global x direction, M2 rotate with a half circle, M3 move diagonal with $(\Delta 1m, \Delta 1m)$, M4 spin at the original place 360°, M5 move in $-y$ direction with 1 meter and M6 again move forward 1 meter in x direction.

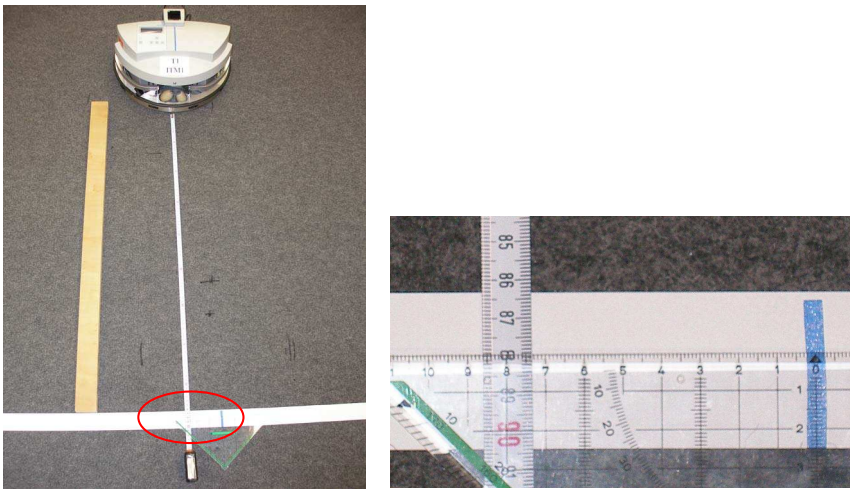
1.5.2 Localization and Position Control Experiments

Through the designed three groups of experiments this study aims to investigate how much the described approaches can improve localization and position control quality.

1.5.2.1 Open Loop Position Control

Open loop position control means the robots are moving by assigning specific velocities and running time, there are no actual pose measurement and no

state feedback. One typical run for moving forward 1 meter in x direction of the open loop position control is shown in Fig. 1.12.



(a) full view (b) partially zoom view

Fig. 1.12 Open loop position control for motion M1

1.5.2.2 Odometry Based Feedback Position Control

The second group of experiments uses only the measurements from the corrected odometry. This is a single sensor relied position control method. The parameters used in the odometry calculation for error sources are

$$\begin{aligned} \boldsymbol{e}_r &= [e_{r_1}, e_{r_2}, e_{r_3}]^T &&= [-0.16\text{mm}, 0.12\text{mm}, -0.03\text{mm}]^T, \\ \boldsymbol{e}_d &= [e_{d_1}, e_{d_2}, e_{d_3}]^T &&= [-2\text{mm}, 2\text{mm}, -10\text{mm}]^T, \\ \boldsymbol{e}_m &= [e_{m_1}, e_{m_2}, e_{m_3}]^T &&= [3^\circ, 2^\circ, 2^\circ]^T, \end{aligned}$$

the nominal values are

$$r = 40\text{mm}, \qquad R = 135\text{mm},$$

and for the odometry correction matrix we use $\boldsymbol{F}_c = \boldsymbol{diag}(0.77\ 0.77\ 1.01)$. A more complete description of the corrected odometry can be found in [19].

1.5.2.3 Localization and Closed Loop Position Control with Odometry and North Star Data Fusion in EKF

In the third group of experiments one needs information from both odometry and North Star. The obtained results are then fused by EKF. Through this the improved pose information can be obtained. The used initial covariances of process and measure noise are $\boldsymbol{Q}(0) = \boldsymbol{diag}(10^{-4}\ 10^{-4}\ (3 \times \pi/180)^2)$ and $\boldsymbol{R}(0) = \boldsymbol{diag}(6.76 \times 10^{-4}\ 6.76 \times 10^{-4}\ (7 \times \pi/180)^2)$, respectively. They all have the units $[\mathrm{m}^2\ \mathrm{m}^2\ \mathrm{rad}^2]$. The odometry has a start up error of 1 cm for the x and y directions and 3° for the orientation error. Similarly, the North Star initial error is estimated as [2.60cm 2.60cm 7°] where a bigger orientation error is used since the North Star system has a worse measurement for the orientation at the beginning although the robot is near to the origin at that time and according to Fig. 1.9 this value is only around 0.2°. The initial estimate error covariance is $\boldsymbol{P}(0) = \boldsymbol{diag}(10^{-4}\mathrm{m}^2\ 10^{-4}\mathrm{m}^2\ (\pi/60)^2\mathrm{rad}^2)$ and the time step is $\Delta t = 0.01$s. For the North Star noises fitting curves we tried 3-th order, 4-th order and 5-th order polynomials, and we find the 4-th order polynomial is reasonable which for x, y directions and orientation noises are

$$\begin{aligned}
\sigma_x &= -0.0010dis^4 + 0.0128dis^3 - 0.0285dis^2 + 0.0265dis + 0.0003,\\
\sigma_y &= -0.0007dis^4 + 0.0054dis^3 - 0.0098dis^2 + 0.0066dis + 0.0025,\\
\sigma_\phi &= -0.0729dis^4 + 0.5625dis^3 - 1.1271dis^2 + 0.8375dis + 0.1500.
\end{aligned}$$

Here dis represents the current distance from the origin to the robot. With above North Star noise expressions, one can update the noise covariance $\boldsymbol{R}(k)$ for each step. The odometry related values used in this group of experiments are exactly the same as in Section 1.5.2.2 for odometry feedback position control.

1.5.3 Result Analysis and Comparisons

Each of the 6 motions is performed with 20 runs and we compare their average values for the three different groups. The statistical results can be seen in Fig. 1.13.

One can see that the positioning accuracy is improved by using odometry or EKF based localization. Taking motion M1 as an example, in the odometry feedback case, the final position error is 0.3cm in average. This is further improved by the EKF based closed loop position control where the localization error is only 0.1cm in average. It is assumed that the controller can drive exactly according to the feedback localized poses which means the final 0.3cm or 0.1cm errors come from the localization, rather than from the controller. Thus, the experiment results can directly represent the localization quality.

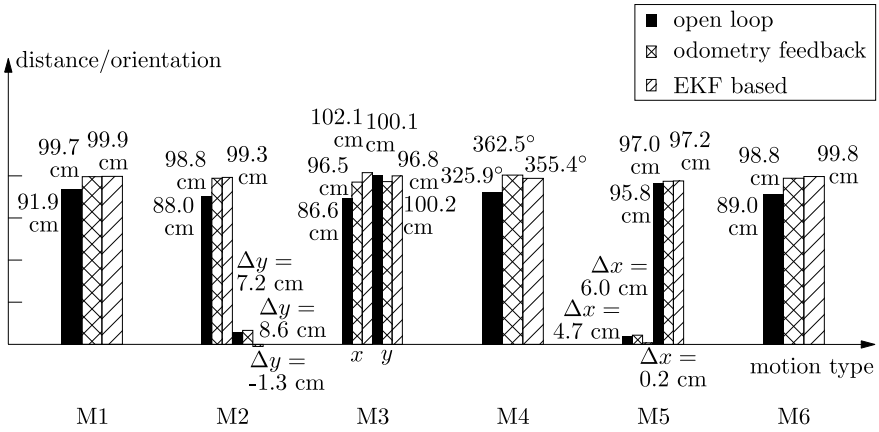


Fig. 1.13 Statistical results of open loop, odometry based closed loop and EKF based closed loop position control experiments

The errors are increasing as the robot is moving away from the origin, i.e., this leads to bigger deviations for long distance movements. This can be seen from motions M1 and M6 in Fig. 1.13. For these two motions, although they perform the same behavior, the errors from all of the three groups of experiments are increased. However, the EKF based one shows the highest robustness since it keeps the smallest deterioration rate.

The robot orientation result is still not satisfactory since the orientation error is more sensitive compared to the translational error because of the omnidirectional feature, see e.g., the results of M4. From motion M4 one also can see that the 'odometry only' feedback control there even obtains a better result than the EKF based one. This is because the odometry, especially our corrected odometry, has a good localization quality in the spinning motion, whereas the North Star system has a worse orientation measurement especially when the robot moves away from its origin. Basically, the EKF based feedback control can further improve the translational positioning accuracy compared to the odometry based case, however the orientation accuracy is slightly worse than in the case where only odometry is used.

In the following some comparisons to a traditional method are performed. In most of the cases, researchers use simultaneously localization and mapping (SLAM) method. It actually has become a de-facto standard for robots navigation and related tasks. The so called SLAM is a technique used by robots and autonomous vehicles to build up a map within an unknown environment (without a priori knowledge) or to update a map within a known environment (with a priori knowledge from a given map) while at the same time keeping track of their current locations. However, this method is computational costly, especially for the unknown environment mapping. It usually needs many vision techniques and image processing knowledge which are not

suitable for our application since the calculation ability on the robot itself is very restricted.

As such, this investigation avoids the SLAM method, at least avoids the mapping part. With PSO's powerful search ability this becomes possible. The mechanical PSO is guiding the robot (generating trajectories) and the robot is handling some things locally such as, e.g., obstacle avoidance and localization. By this method, it is neither necessary to obtain the environment information precisely nor fully. With the help of the corrected odometry or the corrected odometry and North Star data fusion in the extended Kalman filter, the localization quality is improved.

1.6 Conclusions

This contribution investigates the robot localization. For this purpose, methods based on corrected odometry and North Star measurements are developed. Then, improve positioning accuracy by sensor data fusion in the extended Kalman filters. Two measurement channels for EKF data fusion are the corrected odometer and the external North Star system. The data flows and uncertainties during the localization process are analyzed. Finally, the developed methods are verified by robot position control experiments with one group of open loop runs, one group of 'odometry only' closed loop feedback runs and one group of EKF based closed loop feedback runs. The experimental results carried out by an omnidirectional robot on a carpet ground under 6 kinds of motions demonstrate the feasibility of odometry based or EKF based closed loop feedback control for improving robot positioning accuracy. Due to the localized information, the robot pose control can be performed successfully. For an application where the pose accuracy is not a strict requirement, the control based on the corrected odometry is a good and sufficient choice. For the case where higher accuracy is a necessity, and the environment allows to perform sensor data fusion, then the EKF based pose control will be a nicer choice although with limitation for improving the orientation accuracy. From the comparisons, one can see that the proposed techniques show promising results when compared to the no localization case.

Acknowledgements The authors gratefully acknowledge the Cluster of Excellence Simulation Technology (SimTech) in Stuttgart which is funded by German Research Foundation (DFG),

References

1. Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Transactions on Robotics and Automation **7**(3), 278–288 (1991)
2. Caglioti, V., Citterio, A., Fossati, A.: Cooperative, distributed localization in multi-robot systems: a minimum-entropy approach. In: 2006 IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, pp. 25–30. Prague (2006)
3. Cao, M.M.: Dynamic behavior path planning of mobile robot based on fuzzy logic control (in Chinese). Master thesis, Beijing Jiaotong University (2009)
4. Chen, C.T.: Linear System Theory and Design, 3 edn. Oxford University Press, Cary (1999)
5. Eberhard, P., Tang, Q.: Particle Swarm Optimization used for mechanism design and guidance of swarm mobile robots. In: A.E. Olsson (ed.) Particle Swarm Optimization: Theory, Techniques, and Applications, pp. 193–225. Nova Science Publishers, New York (2011)
6. Ferdaus, S.N.: A topological approach to online autonomous map building for mobile robot navigation. Master thesis, Memorial University of Newfoundland (2008)
7. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. Autonomous Robots **8**(3), 325–344 (2000)
8. Grewal, M.S., Andrews, A.P.: Kalman Filtering: Theory and Practice Using MATLAB, 3 edn. John Wiley & Sons, Hoboken (2008)
9. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
10. Howard, A., Matark, M., Sukhatme, G.: Localization for mobile robot teams using maximum likelihood estimation. In: 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 434–439. Lausanne (2002)
11. Jacobs, O.L.R.: Introduction to Control Theory, 2 edn. Oxford University Press, Cary (1993)
12. Kalman, R.E.: A new approach to linear filtering and prediction problems. Basic Engineering **82**(1), 35–45 (1960)
13. Khatib, O.: Real time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research **5**(1), 90–98 (1986)
14. Kwakernaak, H., Sivan, R.: Linear Optimal Control Systems. John Wiley & Sons, Hoboken (1972)
15. Liu, Y., Zhu, J.J., Williams, R.L., Wu, J.H.: Omni-directional mobile robot controller based on trajectory linearization. Robotics and Autonomous Systems **56**(5), 461–479 (2008)
16. Maaref, H., Barret, C.: Sensor-based navigation of a mobile robot in an indoor environment. Robotics and Autonomous Systems **38**(1), 1–18 (2002)
17. Madhavan, R., Fregene, K., Parker, L.E.: Distributed cooperative outdoor multirobot localization and mapping. Autonomous Robots **17**(1), 23–39 (2004)
18. McLurkin, J.: Measuring the accuracy of distributed algorithms on multi-robot systems with dynamic network topologies. In: Distributed Autonomous Robotic Systems 8, pp. 15–26. Springer (2009)
19. Munir, R.: Odometry error propagation and correction for one kind of omnidirectional mobile robots. Student Thesis STUD-369, Institute of Engineering and Computational Mechanics, University of Stuttgart (2012)
20. Murillo, A.C., Guerrero, J.J., Sagüés, C.: Topological and metric robot localization through computer vision techniques. In: 2007 IEEE Workshop From Features to Actions: Unifying Perspectives in Computational and Robot Vision Held with

IEEE International Conference on Robotics and Automation, pp. 79–85. Rome (2007)
21. Roumeliotis, S.I., Bekey, G.A.: Distributed multirobot localization. IEEE Transactions on Robotics and Automation **18**(5), 781–795 (2002)
22. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, pp. 3310–3317. San Diego (1994)
23. Tang, Q., Eberhard, P.: Cooperative motion of swarm mobile robots based on Particle Swarm Optimization and multibody system dynamics. Mechanics Based Design of Structures and Machines **39**(2), 179–193 (2011)
24. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
25. Ulrich, I., Borenstein, J.: VFH+: reliable obstacle avoidance for fast mobile robots. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1572–1577. Leuven (1998)
26. Ulrich, I., Borenstein, J.: VFH*: local obstacle avoidance with look-ahead verification. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, pp. 2505–2511. San Francisco (2000)
27. Wu, J.H.: Dynamic path planning of an omni-directional robot in a dynamic environment. Ph.D. thesis, Ohio University (2005)
28. Yang, Y.: Probabilistic path planning with extended local planners. Ph.D. thesis, Purdue University (2008)