Chapter 13
Mobile Robots

A mobile robot is a device that is capable of locomotion. It has the ability to move around its environment using wheels, tracks, legs, or a combination of them. It may also fly, swim, crawl, or roll. Mobile robots are used for various applications in factories (automated guided vehicles), homes (floor cleaning devices), hospitals (transportation of food and medications), in agriculture (fruit and vegetable picking, fertilization, planting), for military as well as search and rescue operations. They address the demand for flexible material handling, the desire for robots to be able to operate on large structures, and the need for rapid reconfiguration of work areas.

Though mobile robots move in different ways, the focus in this chapter will be on devices that use wheels for locomotion (walking robots are presented in Chap. 14). In industrial applications automated guided vehicles (AGVs) are of special interest to move materials around a manufacturing facility or a warehouse. Tuggers typically pull carts (Fig. 13.1a), unit loaders use a flat platform to transport a unit load stacked on the platform (Fig. 13.1b), and mobile forklifts are used to automatically pickup and drop loads off from various heights (Fig. 13.1c). AGVs typically follow markers or wires in the floor, or use vision, magnets, or lasers for moving around the facility. This organized movement is called navigation; a process or activity to plan and direct a robot along a route or path to move safely from one location to another without getting lost or colliding with other objects.

Navigation is typically a complex task consisting of localization, path planning and motion control. Localization denotes robot's ability to establish its own position and orientation within the global coordinate frame. Autonomous path planning represents determination of a collision-free path for a robot between start and goal positions between obstacles cluttered in a workspace. This also includes interactions between mobile robots and humans and between groups of mobile robots. Motion control must guarantee execution of movement along the planned path with simultaneous obstacle avoidance.

In collaborative settings humans and robots share a workspace resulting in a need for improved human-robot communication and for robot awareness of people around it. The robot must typically keep a safe distance from people. However, devices like

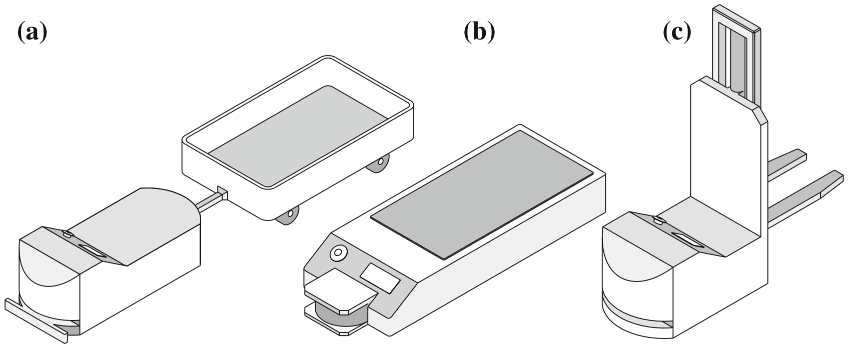

Fig. 13.1 Automated guided vehicles: **a** Tugger, **b** unit loader, and **c** mobile forklift

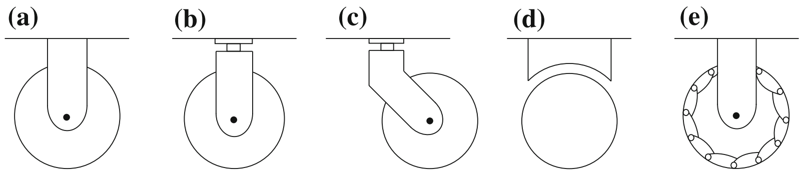

Fig. 13.2 Wheel designs: **a** Standard fixed wheel, **b** standard steered wheel, **c** castor wheel, **d** spherical wheel, and **e** Swedish wheel

personal care robots, require close proximity between the human and the robot and these machines are examples of advanced human-robot interactive systems.

13.1 Mobile Robot Kinematics

With its simple mechanical design, the wheel is the most popular locomotion mechanism in mobile robotics. Wheels provide traction and three wheels guarantee stable robot balance. Wheels can be designed in different forms as shown in Fig. 13.2.

The fixed wheel, the standard steered wheel and the castor wheel have a primary axis of rotation and are directional. Movement in different direction is not possible without first steering the wheel around the vertical axis. The spherical wheel is omnidirectional as it enables movement in all directions without steering first. The Swedish wheel tries to achieve omnidirectional behavior with passive rollers attached around the circumference of the wheel. Thus, the wheel can move along different trajectories, as well as forwards and backwards.

Selection of wheel type, number of wheels, as well as their attachment to the robot chassis significantly affect mobile robot kinematics. Examples of kinematic designs are shown in Fig. 13.3. They range from two-wheel to four-wheel configurations. The two platforms in the righthand column are omnidirectional.

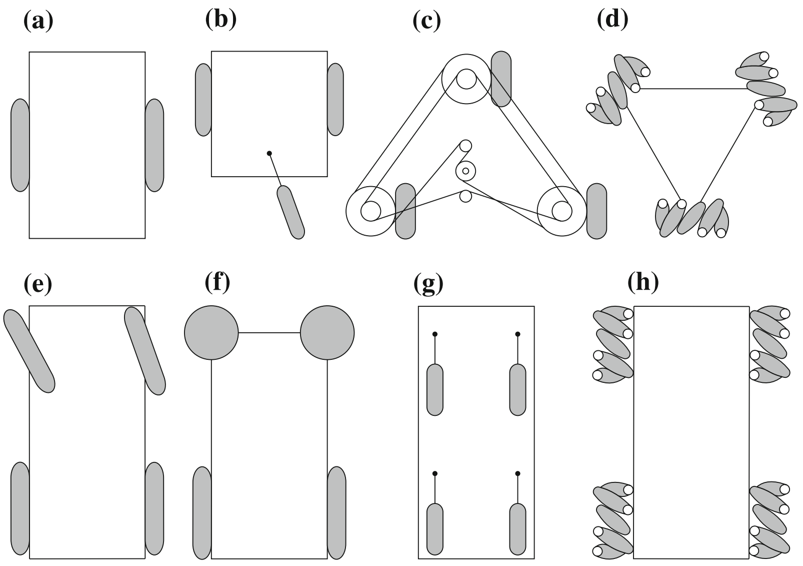

Fig. 13.3 Mobile robot configuration examples: **a** two-wheel differential drive, **b** differential drive with castor wheel, **c** three synchronously motorized and steered wheels, **d** three omnidirectional wheels in triangle, **e** four wheels with car-like steering, **f** two differential traction wheels and two omnidirectional wheels, **g** four motorized and steered castor wheels, and **h** four omnidirectional wheels in rectangular configuration

For the purpose of analysis, a mobile robot will be represented as a rigid body on wheels that can move only in a horizontal plane. With these assumptions the pose of the robot can be defined with three coordinates, two representing position in the horizontal plane and one describing orientation around the vertical axis. Relations are presented in Fig. 13.4 for a simple differential drive mechanism. Axes x_G and y_G define the global coordinate frame. The robot local coordinate frame is defined with axes x_m and y_m. The x_m axis points in the robot forward direction.

Robot position and orientation are defined with the following vector

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix}, \tag{13.1}$$

where x and y coordinates define robot position relative to the global coordinate frame and angle φ determines its orientation (rotation around vertical axis). Robot orientation can be described also in the form of a rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{13.2}$$

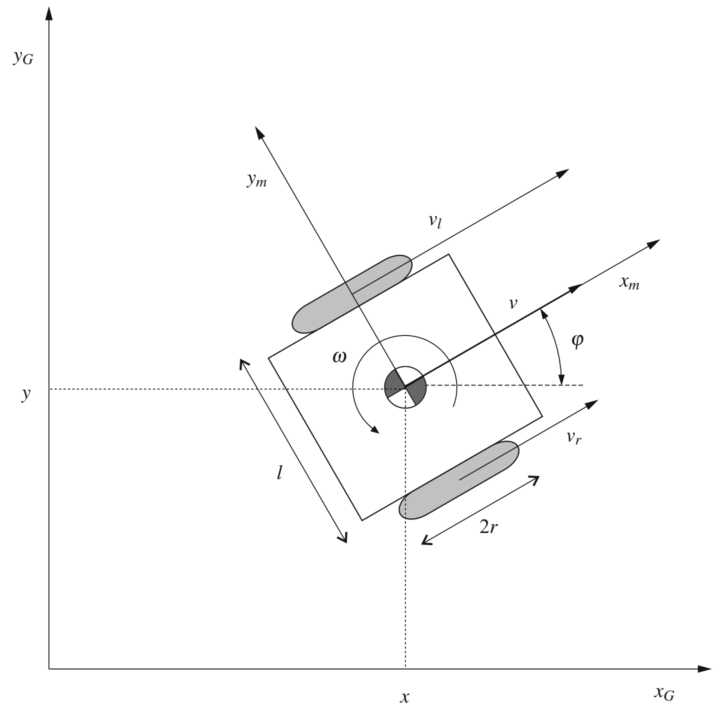

Fig. 13.4 Position and orientation of a mobile robot—differential drive robot example

Homogenous transformation matrix describing the pose of the mobile robot is then

$$\mathbf{T} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & x \\ \sin\varphi & \cos\varphi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{13.3}$$

The differential drive robot presented in Fig. 13.4 has a simple mechanical structure. Its movement is based on two separately driven wheels attached on either side of the robot body. The robot changes its direction by varying the relative speed of rotation of its wheels. Thus, it does not require an additional steering motion. If wheels are driven in the same direction and with equal speed, the robot will follow a straight line. If wheels are turned with equal speed in opposite directions, the robot will rotate about the middle point between the wheels. In general, the center of robot rotation may lay anywhere on the line through wheel axes and will depend on each wheel speed of rotation and its direction.

With its simple kinematics it is an ideal model for studying robot movement. By representing robot width (distance between tire contact points with the ground) with l and wheel radius with r the robot motion can be analyzed. The wheels rotate with

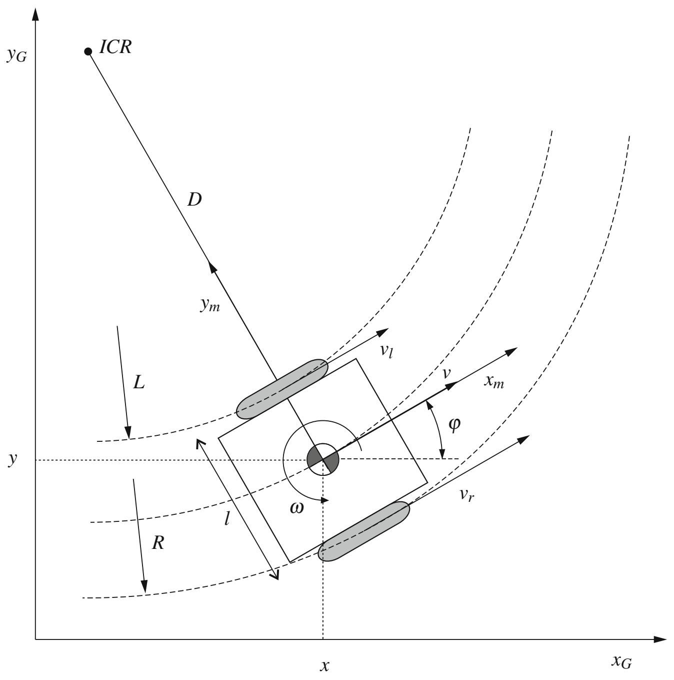

Fig. 13.5 Differential drive robot kinematics

angular rates ω_r (right wheel) and ω_l (left wheel), resulting in wheel speeds v_r and v_l of the right and left wheel, respectively

$$\begin{aligned} v_r &= \omega_r r, \\ v_l &= \omega_l r. \end{aligned} \tag{13.4}$$

The two wheel rotations result in the robot translational speed along robot x_m axis and angular rate around its vertical axis. With reference to Fig. 13.5 the angular rate can be defined as

$$\omega = \frac{v_l}{D - \frac{l}{2}} = \frac{v_r}{D + \frac{l}{2}}, \tag{13.5}$$

where D is the distance between the middle point on the robot (in this case the origin of the frame x_m–y_m) and the point that defines the instantaneous center of rotation (*ICR*). The *ICR* is the point in the horizontal plane around which the robot rotates at a specific instant of time. From equality in (13.5) the following relation can be derived

$$\omega = \frac{v_r - v_l}{l} = \frac{r}{l}(\omega_r - \omega_l). \tag{13.6}$$

Translational speed along the x_m axis can then be determined as

$$v = \omega D = \frac{v_r + v_l}{2} = \frac{r}{2}(\omega_r + \omega_l). \tag{13.7}$$

Equations (13.6) and (13.7) define relations between wheels' angular rates and mobile robot velocity. However, from the control perspective it is the more relevant inverse relation that defines wheels' angular rates from the desired robot velocity. By combining (13.6) and (13.7) the following relations are obtained

$$\begin{aligned} \omega_r &= \frac{2v + \omega l}{2r}, \\ \omega_l &= \frac{2v - \omega l}{2r}. \end{aligned} \tag{13.8}$$

Robot velocity determined as a pair $[v,\ \omega]$ is defined relative to the local coordinate frame of the mobile robot x_m–y_m. Robot velocity in the global coordinate frame x_G–y_G defined as time derivative of robot pose vector $\mathbf{x}$ (13.1) can be computed by rotating the locally expressed velocity using the rotation matrix $\mathbf{R}$ (13.2) as

$$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} v\cos\varphi \\ v\sin\varphi \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix}. \tag{13.9}$$

By combining translational and rotation parts of the above equations and omitting elements that are zero, the mobile robot velocity in the global coordinate frame can be written as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} v\cos\varphi \\ v\sin\varphi \\ \omega \end{bmatrix}. \tag{13.10}$$

From Eq. (13.10) it is clear that relevant quantities for describing mobile robot movement are translational velocity along robot x_m axis v, rotational velocity around vertical axis ω, and robot orientation with respect to the global coordinate frame φ. With this in mind we may further simplify the differential drive robot into a unicycle model (as shown in Fig. 13.6). Now the above-mentioned three quantities describe the movement of the unicycle represented as a single wheel with marked forward direction in the middle of the differential drive robot in Fig. 13.6. The unicycle can be easily transformed back to the differential drive robot based on Eq. (13.8).

The attractive property of the unicycle model is its simplicity. Therefore, it will be used throughout this chapter for analysis. However, the model can be in general converted back to any other kinematically more complex mobile robot. As an example, we review a mobile platform based on the car steering principle shown in Fig. 13.7.

The car steering geometry solves the problem of wheels on the inside and outside of a turn needing to trace circles of different radii. Therefore, steering angles of left and right front wheels are different. In the unicycle model the orientation of the unicycle is defined with angle φ, the same as the orientation of the differential drive robot.

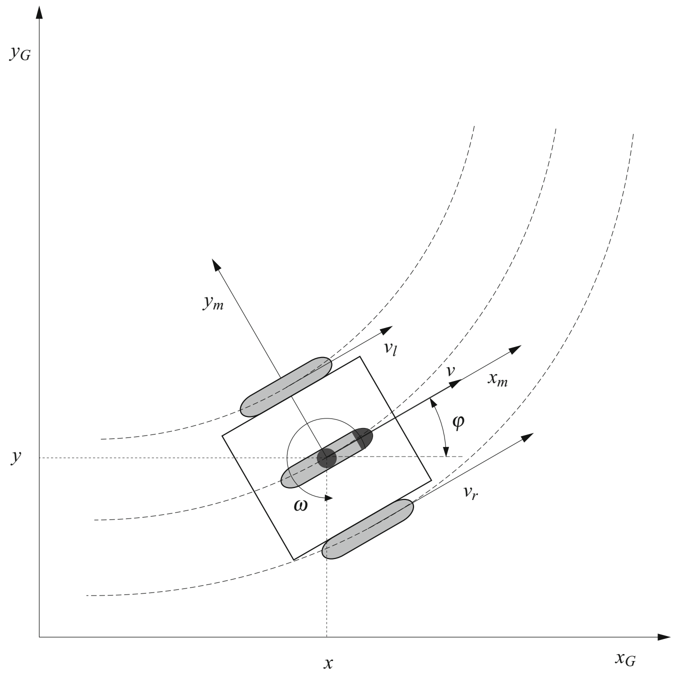

Fig. 13.6 Unicycle model of a differential drive mobile robot

In the car-like problem the orientation of the mobile robot is defined by angle φ. The unicycle model is positioned in the middle of the front wheels and its orientation is defined such to achieve the same instantaneous center of rotation as defined by the orientation of the car's left and right wheels. The unicycle is now the third front wheel and the *ICR* is positioned at the intersection point of all the three lines perpendicular to the front wheels. Angle ψ is now defined as the deviation of the unicycle orientation from the robot x_m axis (as shown in Fig. 13.7). By computing angle ψ the relation between the car-like robot and the unicycle will be established.

By following the same principle as in (13.7), translational velocity of the unicycle can be defined as

$$v = D\omega, \tag{13.11}$$

where D is the distance between the unicycle and the *ICR*. Distance D can then be computed as

$$D = \frac{v}{\omega}. \tag{13.12}$$

Path curvature for the unicycle $\mathscr{K}_u$ can be defined as the inverse of the instantaneous radius of rotation as

$$\mathscr{K}_u = \frac{1}{D} = \frac{\omega}{v}. \tag{13.13}$$

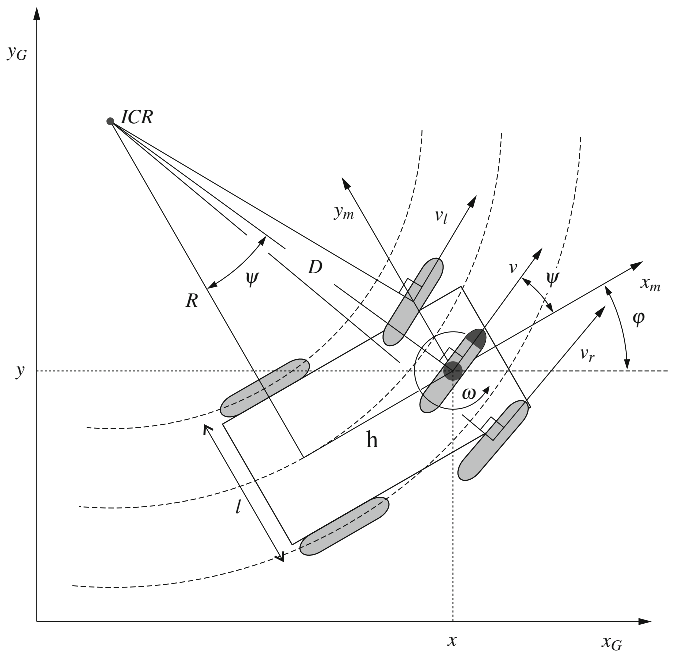

Fig. 13.7 Unicycle model of the car-like steering mobile robot

By considering car kinematics, the following relation can be written from Fig. 13.7.

$$h = D \sin \psi, \tag{13.14}$$

where angle ψ is also the angle between lines D and R (the distance between ICR and the middle point between the rear wheels of the vehicle) and h is the distance between the center of the unicycle and the middle point between the rear wheels of the robot. Distance D can then be computed as

$$D = \frac{h}{\sin \psi} \tag{13.15}$$

and the curvature for the car $\mathscr{K}_c$ is then defined as

$$\mathscr{K}_c = \frac{1}{D} = \frac{\sin \psi}{h}. \tag{13.16}$$

With equal $\mathscr{K}_c$ and $\mathscr{K}_u$ the following relation can be obtained

$$\mathscr{K}_c = \mathscr{K}_u \quad \Rightarrow \quad \sin \psi = \frac{\omega l}{v}. \tag{13.17}$$

Finally, the angle ψ equals

$$\psi = \arcsin \frac{\omega l}{v}. \tag{13.18}$$

Angle ψ is the desired steering angle for the car and it can be computed from the known speed v, angular rate ω, and width of the car l.

With the defined relation between the unicycle and a mobile robot with other kinematics the analysis can be based on a simple unicycle model and generalized to the other robot.

13.2 Navigation

Mobile robots often operate in unknown and unstructured environments and need to self-localize, plan a path to a goal, build and interpret the map of the environment, and then control their motion through that environment.

13.2.1 Localization

An important difference between a manipulator and a mobile robot is in position estimation. A manipulator has a fixed base and by measuring robot joint positions and knowing its kinematic model it is possible to determine the pose of its end-effector. A mobile robot can move as one unit through the environment and there is no direct way for measuring its position and orientation. A general solution is to estimate the robot position and orientation through integration of motion (velocity) over time.

However, more accurate and often also more complex approaches are typically required. If the map of the environment is known in advance mobile robot paths can be preplanned. This is specifically useful when the environment is relatively static and robust operation is required, such as in industrial applications. More complex approaches are based on dynamic path planning based on sensor information and recognition of features in the environment. The robot first determines its own position and plans its movement through traversable areas. When the workspace or the tasks change frequently it is typically better to plan dynamically. Often a trade-off is required between preplanning and dynamic generation of plans. In order to simplify the task, markers may be placed in the environment. These markers can be easily recognized by sensors on the robot and provide accurate localization.

Automated guided vehicles in industrial environments make use of various navigation/guidance technologies: magnetic tape, wire, magnetic spot, laser, and natural.

Localization and path planning are often based on electrified wires embedded in the floor using inductive guidance. A guide path sensor is mounted on the vehicle. The wire can be replaced by magnetic tape or a painted line (Fig. 13.8a). In the latter

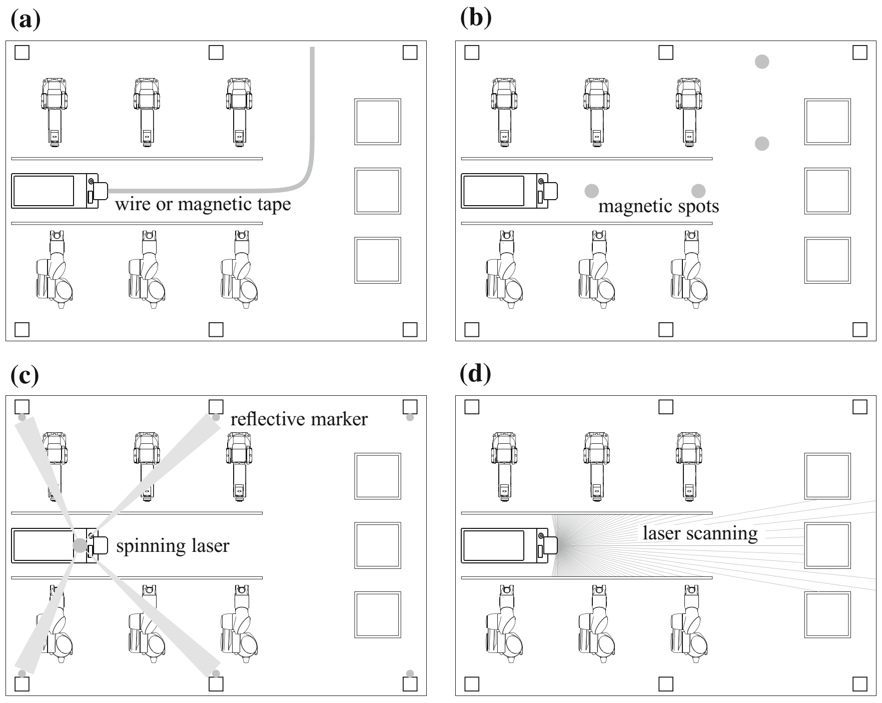

Fig. 13.8 Sensor abstraction disk from the suit of sensors on board the robot

case the robot uses a camera to determine its relative position to the floor line. Paths are fixed and continuous. Unique markers may be placed along the line to indicate specific positions. Instead of placing lines and markers on the floor, markers (two-dimensional patterns) can also be put on the ceiling to be identified by an onboard camera. Magnetic spot guidance uses path marked with magnetic pucks (Fig. 13.8b). Paths are open and changeable.

Floor-based localization techniques are often replaced by laser-based methods. Laser triangulation methods, in which a spinning laser senses range and azimuth to wall-mounted reflectors, provide accurate localization information without the need to follow specific lines on the floor. Laser guidance technology uses multiple, fixed reference points (reflective strips) located within the operating area that can be detected by a laser head mounted on the vehicle (Fig. 13.8c). As the facility is mapped in advance, paths can be easily changed and expanded.

Natural navigation is based on information of the existing environment scanned by laser scanners, with the aid of a few fixed reference points (Fig. 13.8d). Area is mapped in advance. Natural navigation is flexible and expendable. It is suitable for environments that change frequently but not significantly. In confined spaces the robot may follow the wall through the environment range-basing from the wall.

Radio-based indoor positioning systems are also being introduced that enable robot localization in a similar manner as the outdoor global positioning system. Localization is based on triangulation with fixed beacons mounted in the facility and the sensor mounted on the robot. Distances are computed by measuring travel time of radio waves from the beacon to the sensor.

13.2.1.1 Odometry

A simple and commonly-used approach for robot localization is to rely on odometry, which uses information from motion sensors (typically wheel encoders) to estimate change in position over time. These position changes are accumulated using integration principles providing the robot position relative to a starting location. The method is sensitive to errors due to integration of velocity measurements over time to give position estimates.

Analysis of robot motion starts with the understanding of the contribution of each wheel to the velocity of the robot. For the specific case of a differential drive robot these relations are defined in (13.6) and (13.7). Wheel speed may be directly measured using a tachometer. If such a sensor is not available, the speed can be estimated through numerical differentiation of the position obtained from encoders. In such case speeds for the right and left wheel can be computed as

$$\begin{aligned} v_r &= 2\pi r \frac{n_r(t) - n_r(t-\Delta t)}{N\Delta t}, \\ v_l &= 2\pi r \frac{n_l(t) - n_l(t-\Delta t)}{N\Delta t}, \end{aligned} \tag{13.19}$$

where r is the wheel radius, N is the encoder resolution in terms of counts per revolution, n_r and n_l are encoder counts of the right and left wheel at time t, respectively, and $n_r(t-\Delta t)$ and $n_l(t-\Delta t)$ are the same quantities at the previous sampling time.

Robot position and orientation can then be estimated with numerical integration of Eq. (13.10) and consideration of (13.6) and (13.7) as

$$\begin{aligned} x(t) &= x(t-\Delta t) + v\cos\varphi\Delta t = x(t-\Delta t) + \frac{v_r + v_l}{2}\cos\varphi\Delta t, \\ y(t) &= y(t-\Delta t) + v\sin\varphi\Delta t = x(t-\Delta t) + \frac{v_r + v_l}{2}\sin\varphi\Delta t, \\ \varphi(t) &= \varphi(t-\Delta t) + \omega\Delta t = \varphi(t-\Delta t) + \frac{v_r - v_l}{l}\Delta t. \end{aligned} \tag{13.20}$$

Different factors reduce the effectiveness of odometry-based methods for robot position estimation. A very important factor is wheel slippage that significantly reduces precision of position estimation. Performance may be improved by using models of the errors and of the vehicle. Floor spots or magnets may be used to correct for odometry errors that accumulate between these points. Odometry can also be augmented by sensor-based measurements from lasers, cameras, radiofrequency identification systems, and beacons.

13.2.1.2 Simultaneous Localization and Mapping

More advanced systems make use of algorithms that accomplish the navigation subtasks (localization, path planning) simultaneously. The approach that is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map is called simultaneous localization and mapping (SLAM). By observing the same features in multiple views using sensors that move with the vehicle, the SLAM algorithm accumulates and combines together the sensor information. By combining the robot position estimation with the gathered information, a local map can be constructed by stitching together available data. Over time, the complete environment can be mapped and the maps can be used to plan the robot paths.

SLAM consists of multiple parts, such as landmark extraction, data association, state estimation, state update and landmark update. There are many ways to solve each of the smaller parts, but they are beyond the scope of this book.

13.2.1.3 Sensor Abstraction Disk

When the mobile robot is moving through the environment it must also observe its surroundings. Sensors on-board the robot look for obstacles or unexpected objects in the path of the vehicle and the robot may be able to plan a way around them before returning to the pre-planned route. A typical suite of sensors includes infrared proximity sensors, ultrasonic distance sensors, laser scanners, vision, tactile sensing, and global positioning sensors. Sensors are strategically placed onboard the robot and around its circumference. Each sensor provides different information in terms of quantity, quality, range, and resolution. However, typically information from all sensors is combined to provide an accurate image of the robot environment. Without dealing specifically with analysis of individual sensors and integration of sensory information it is possible to assume that distance and direction to all obstacles from the robot's perspective can be obtained from the sensor suite. The sensor abstraction disk presented in Fig. 13.9 is an example of sensory integration providing information about obstacles within the radius of the disk around the robot.

From the known position d_o and orientation φ_o of the obstacle and the known pose of the robot $[x,\ y,\ \varphi]^T$, it is possible to determine the obstacle position $(x_o,\ y_o)$ in the global coordinate frame as

$$\begin{aligned} x_o &= x + d_o \cos(\varphi + \varphi_o), \\ y_o &= y + d_o \sin(\varphi + \varphi_o). \end{aligned} \tag{13.21}$$

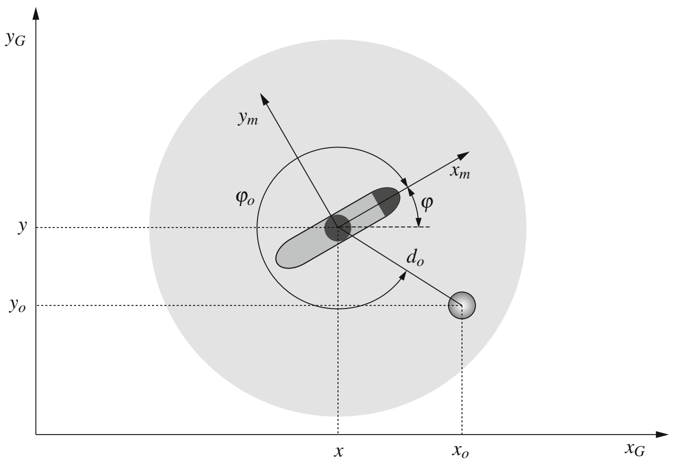

Fig. 13.9 Sensor abstraction disk from the suit of sensors on board the robot

The following analysis will be based on the assumptions of a unicycle robot model and the information about objects obtained from the sensor abstraction disk.

13.2.2 Path Planning

Path planning enables autonomous mobile robots to track an optimal collision free path from the starting position to the goal without colliding with obstacles in the workspace area. An ideal path planner must be able to handle uncertainties in the sensed world model, to minimize the impact of objects to the robot, and to find the optimum path in minimum time especially if the path is to be negotiated regularly. In general, the path planning should result in the path with the lowest possible cost, it should be fast and robust as well as generic with respect to different maps.

Different algorithms are available for (real-time) path planning. A simple method consists of combining straight-line segments connected with vertices. Another standard search method for finding the optimal path is the A* algorithm with its modifications. The algorithm finds a directed path between multiple points, called nodes. The robot environment represented with a map can be decomposed into free and occupied spaces. Then A* search can be performed to find a piecewise linear path through the free nodes.

An artificial potential field algorithm can be used for obstacle avoidance. The algorithm uses repulsive potential fields around the obstacles to force away the robot subjected to this potential and use an attractive potential field around the goal to attract the robot to go to the goal. Repulsive and attractive fields modify the robot's path. The algorithm enables real-time operations of a mobile robot in a complex environment.

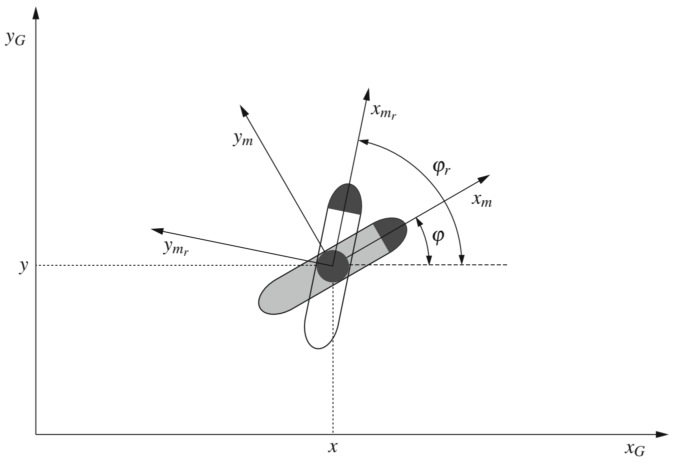

Fig. 13.10 Unicycle orientation control; grey unicycle represents actual robot and white unicycle represents desired orientation

13.2.3 Path Control

In order to complete the task, the mobile robot needs to move from its initial location to the desired final position and orientation. A control system is required to control the vehicle along its path.

13.2.3.1 Control of Orientation

Based on the unicycle model presented in Fig. 13.10 control of orientation will first be considered. A similar approach would be valid for mobile robots that can change orientation without changing their position (a differential drive robot is such a vehicle, but the car is not).

The control goal is to minimize the orientation error

$$\tilde{\varphi} = \varphi_r - \varphi, \tag{13.22}$$

where φ_r is the desired orientation and φ is the actual orientation. We assume that the control is based on proportional-integral-derivative (PID) control approach

$$PID(\tilde{\varphi}) = K_p\tilde{\varphi} + K_i \int \tilde{\varphi}dt + K_d\dot{\tilde{\varphi}} \tag{13.23}$$

or one of its subversions, such as proportional-derivative controller. Then the desired angular velocity of the mobile robot can be computed as

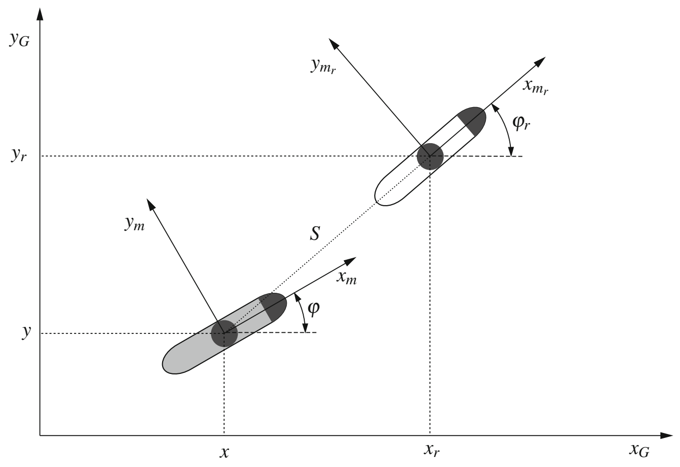

Fig. 13.11 Unicycle position and orientation control; grey unicycle represents actual robot and white unicycle represents goal location

$$\omega = K_p\tilde{\varphi} + K_i \int \tilde{\varphi} dt + K_d \dot{\tilde{\varphi}}. \tag{13.24}$$

It should be noted that angles are periodic functions and if we assume configuration

$$\varphi_r = 0 \quad \wedge \quad \varphi = 2\pi \quad \Rightarrow \quad \tilde{\varphi} = -2\pi, \tag{13.25}$$

the robot will spin once before it will reach the final orientation. This is usually not desirable robot behavior. Therefore, orientation error must be limited such to require at maximum π radians rotation in either direction

$$\tilde{\varphi} \in [-\pi, \pi]. \tag{13.26}$$

A simple solution is to use a four-quadrant arctan function as

$$\tilde{\varphi} = \arctan(\sin\tilde{\varphi}, \cos\tilde{\varphi}) \in [-\pi, \pi]. \tag{13.27}$$

With the combination of (13.27) and (13.24) the robot will reach the desired orientation without rotating more than half circle in positive or negative direction.

13.2.3.2 Control of Position and Orientation

The mobile robot typically moves from its initial location to its final (goal) location which requires change of position and orientation. Since the robot needs to move to its goal location we will refer to this task as *go-to-goal*. Figure 13.11 represents such conditions. Coordinate frame x_m–y_m defines the robot current pose and frame x_{m_r}–y_{m_r} defines the goal pose. Line segment S represents the shortest path for completing the task.

The desired robot orientation for completing the task can be defined as the angle between line segment S and the horizontal axis of the global coordinate frame. With the known desired position $(x_r,\ y_r)$ and robot current position $(x,\ y)$, angle φ_r can be computed at every time instant during robot motion as

$$\varphi_r = \arctan \frac{y_r - y}{x_r - x}. \tag{13.28}$$

By assuming that the robot is moving at constant forward speed v_0, robot movement in the global coordinate frame can be described with the following set of equations

$$\begin{aligned} \dot{x} &= v_0 \cos \varphi, \\ \dot{y} &= v_0 \sin \varphi, \\ \dot{\varphi} &= \omega = PID(\tilde{\varphi}). \end{aligned} \tag{13.29}$$

With this approach the control goal is to maintain constant speed v_0 and track the desired angle φ_r computed from (13.28). If we assume a differential drive robot, wheel angular rates can then be computed from (13.8) as

$$\begin{aligned} \omega_r &= \frac{2v_0 + \omega l}{2r}, \\ \omega_l &= \frac{2v_0 - \omega l}{2r}. \end{aligned} \tag{13.30}$$

When moving with constant velocity v_0 the robot would overshoot its goal location. Therefore, it is reasonable to define robot forward speed based on the distance to the goal

$$G = \sqrt{(x_r - x)^2 + (y_r - y)^2}. \tag{13.31}$$

With a proportional controller, the desired speed can be defined as

$$v_G = K_v G, \tag{13.32}$$

where K_v is the velocity gain. Equations (13.29) can then be rewritten as

$$\begin{aligned} \dot{x} &= v_G \cos \varphi, \\ \dot{y} &= v_G \sin \varphi, \\ \dot{\varphi} &= \omega = PID(\tilde{\varphi}) \end{aligned} \tag{13.33}$$

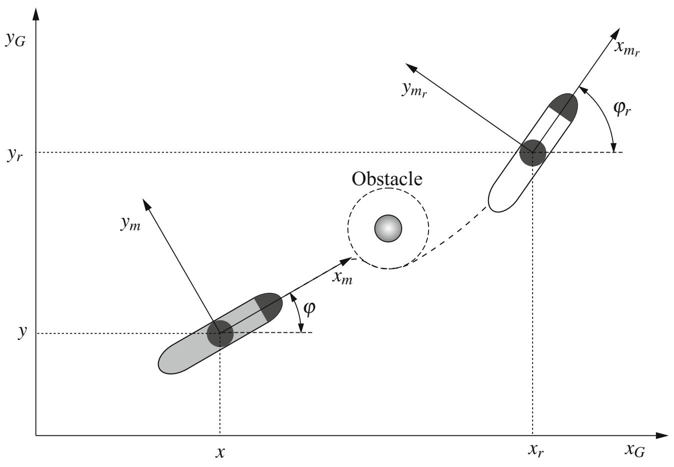

Fig. 13.12 Unicycle position and orientation control with obstacle avoidance; grey unicycle represents actual robot and white unicycle represents goal location; gray circle is the obstacle and dashed circular line is safe zone around the obstacle

and in (13.30) v_0 must be replaced by v_G. With this approach the robot will decelerate when approaching the goal location. Since desired speed increases with the distance to the goal, a maximum limit can be set on $v_G \in [0, v_{G_{max}}]$.

13.2.3.3 Obstacle Avoidance

Figure 13.12 shows conditions with an obstacle in the robot's path to the goal position. The robot cannot proceed directly to the target location without first avoiding the obstacle. Based on the concept of the sensor abstraction disk we assume that the robot is capable of detecting and locating the obstacle from a safe distance and using this information, can plan avoidance activities. The obstacle in Fig. 13.12 is represented by a gray circle and the dashed circular line around the obstacle represents a safe zone around the obstacle. The robot would not be allowed to enter the dashed circle.

With this in mind, we now have two control objectives. The first is *go-to-goal* and the second is *avoid-obstacle*. A more detailed representation of the two control objectives is shown in Fig. 13.13, where d_o indicates distance from the robot to the obstacle, u_g is the control variable associated with the *go-to-goal* objective and u_o is the control variable associated with *avoid-obstacle* objective. In order to successfully complete the task, the u_g needs to point to the goal while the u_o needs to point away from the obstacle. The actual control variable u is the result of blending u_g and u_o.

The *go-to-goal* control part can be defined based on the distance to the goal position as

$$\begin{bmatrix} u_{g_x} \\ u_{g_y} \end{bmatrix} = K_g \begin{bmatrix} x_r - x \\ y_r - y \end{bmatrix}. \tag{13.34}$$

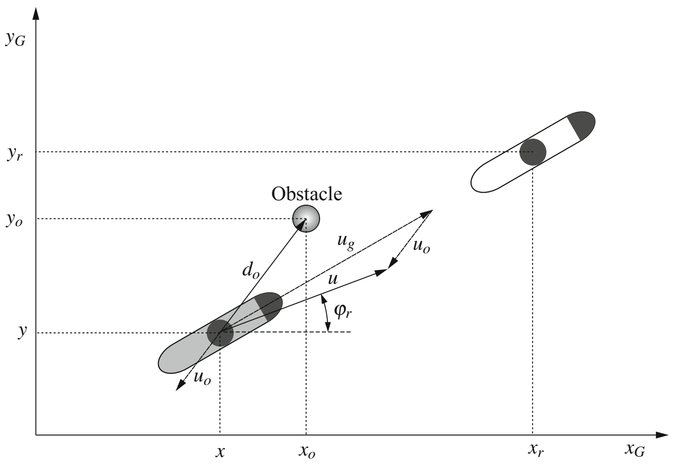

Fig. 13.13 Unicycle obstacle avoidance; grey unicycle represents actual robot, white unicycle represents goal location and grey circle is obstacle

Similarly, the *avoid-obstacle* control variable can be defined based on the distance to the obstacle

$$\begin{bmatrix} u_{o_x} \\ u_{o_y} \end{bmatrix} = K_o \begin{bmatrix} x - x_o \\ y - y_o \end{bmatrix}. \tag{13.35}$$

It should be noted that u_g points to the goal and u_o points away from the obstacle as seen by the definition of distances in the above two equations. Blending of the two control variables must be made based on the distance to the obstacle, which is defined as

$$\|d_o\| = \sqrt{(x_o - x)^2 + (y_o - y)^2}. \tag{13.36}$$

When the robot is far away from the obstacle, it only needs to proceed directly to the goal. However, in the vicinity of the obstacle the primary task becomes obstacle avoidance. Consecutively, blending can be implemented as

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \lambda(\|d_o\|) \begin{bmatrix} u_{g_x} \\ u_{g_y} \end{bmatrix} + (1 - \lambda(\|d_o\|)) \begin{bmatrix} u_{o_x} \\ u_{o_y} \end{bmatrix}, \quad \lambda(\|d_o\|) \in [0, 1]. \tag{13.37}$$

Parameter λ can, for example, be defined as an exponential function based on distance to the obstacle $\lambda = 1 - e^{-\kappa \|d_o\|}$ and parameter κ defines convergence rate of the function toward 1. As seen from Fig. 13.13 control variable u defines desired robot velocities in the global coordinate frame

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}. \tag{13.38}$$

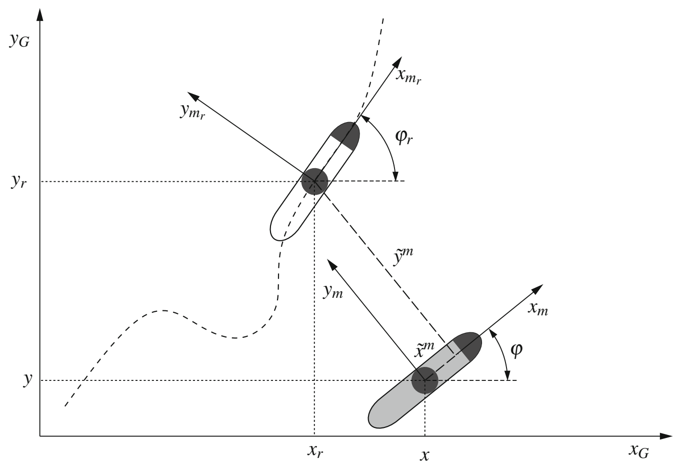

Fig. 13.14 Unicycle path following control; grey unicycle represents actual robot and white unicycle represents virtual vehicle on the path

Desired robot orientation can then be computed as

$$\varphi_r = \arctan \frac{u_y}{u_x}, \tag{13.39}$$

resulting in angular rate

$$\dot{\varphi} = \omega = PID(\tilde{\varphi}). \tag{13.40}$$

The forward robot speed can be computed as

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{v^2 \cos^2 \varphi + v^2 \sin^2 \varphi} = \sqrt{u_x^2 + u_y^2}. \tag{13.41}$$

Again, by assuming a differential drive robot, wheel angular rates can be computed from (13.8).

13.2.3.4 Path Following

Often the robot cannot just take the shortest path to the goal and it must follow a predefined path. In this case the control goal is to stay on the path. The task can be simplified by considering a virtual vehicle that moves along the path with a predefined speed. Then the control goal becomes tracking the virtual vehicle as shown in Fig. 13.14.

The tracking error can be defined as

$$\tilde{\mathbf{x}} = \mathbf{x}_r - \mathbf{x}, \tag{13.42}$$

where $\mathbf{x}_r$ and $\mathbf{x}$ represent position and orientation of the virtual vehicle and the mobile robot, respectively. All quantities are expressed in the global coordinate frame and can be transformed into the robot coordinate frame as

$$\tilde{\mathbf{x}}^m = \begin{bmatrix} \tilde{x}^m \\ \tilde{y}^m \\ \tilde{\varphi}^m \end{bmatrix} = \mathbf{R}^T \tilde{\mathbf{x}}, \tag{13.43}$$

where $\mathbf{R}$ is defined as in (13.2). The robot forward speed can be computed from the tracking error along x_m axis as

$$v = K_x \tilde{x}^m, \tag{13.44}$$

where K_x is the controller proportional gain. The angular rate must take into account the angle tracking error $\tilde{\varphi}^m = \tilde{\varphi}$ as well as distance to the path $\tilde{y}^m$. Namely, when the robot is away from the path it must steer toward the path. Thus, the control algorithm becomes

$$\omega = K_y \tilde{y}^m + K_\varphi \tilde{\varphi}^m, \tag{13.45}$$

where K_y and K_φ are controller proportional gains. Since velocity of the virtual vehicle is known (angular rate can be computed as the change of tangential direction along the path when the virtual vehicle moves forward), it can be taken into account as a feedforward control term. If v_r is the forward speed of the virtual vehicle and ω_r its angular rate, Eqs. (13.44) and (13.45) can be rewritten with the feedforward term as

$$v = v_r \cos \tilde{\varphi} + K_x \tilde{x}^m \tag{13.46}$$

and

$$\omega = \omega_r + K_y \tilde{y}^m + K_\varphi \tilde{\varphi}^m. \tag{13.47}$$