Artificial Intelligence Review (2020) 53:653–670
https://doi.org/10.1007/s10462-018-9672-9

Robust visual line-following navigation system for humanoid robots

Li-Hong Juang[1] · **Jian-Sen Zhang**[2]

Published online: 8 December 2018

Abstract
This paper implements a novel line-following system for humanoid robots. Camera embedded on the robot's head captures the image and then extracts the line using a high-speed and high-accuracy rectangular search method. This method divides the search location into three sides of rectangle and performs image convolution by edge detection matrix. The extracted line is used to calculate relative parameters, including forward velocity, lateral velocity and angular velocity that drive line-following walking. A proposed path curvature estimation method generates the forward velocity and guidance reference point of the robot. A classical PID controller and a PID controller with angle compensation are then used to set the lateral velocity and angular velocity of the robot, improving the performance in tracking a curved line. Line-following experiments for various shapes were conducted using humanoid robot NAO. Experimental results demonstrate the robot can follow different line shapes with the tracking error remaining at a low level. This is a significant improvement from existing biped robot visual navigation systems.

Keywords Line-following · Humanoid robot · Computer vision · Rectangle search · Angle compensation · PID controller · Visual navigation

1 Introduction

Navigation is now an important research direction for the humanoid robot. Most mobile robots in the industry are using line-following technologies. A line-following algorithm can perceive orientation and location based on the locations of the robot and the line. Because this method can provide absolute location information, it is very useful in robot navigation.

There are three commonly used methods in line following. One of them uses the principle of electro-magnetic induction to navigate the electromagnetic wires laid on the ground.

✉ Li-Hong Juang
lipuu@qq.com

[1] School of Electrical Engineering and Automation, Xiamen University of Technology, No. 600, Ligong Road, Jimei, Xiamen 361024, People's Republic of China

[2] Engineering College, HuaQiao University, No. 269,Chenghua North Road, Fengze District, Quanzhou, Fujian 362021, People's Republic of China

Another employs a camera to capture the image of black tape to navigate. Another uses infrared sensors based on photoelectric conversion. Although the sensors used in robots are different, the objective is the same: make the robot accurately follow the established path.

The control algorithms of line following are the main work of many reasearchers. Abhishek Roy and Mathew Mithra Noel (2016) presented an infrared sensor matrix-based line-following robot that can smoothly follow a random curved path using hybrid artificial neural network (ANN) based control strategy. Gianluca Antonelli et al. (2007) proposed a fuzzy-logic-based approach for mobile robot line following. They fuzzify a curve to smooth, appropriate and narrow sets; and fuzzify a distance to close, medium and far sets. They put the emphasis on the if–then rules and calculation of line (angular) velocity; there is no algorithm for the recognition of the path pattern. KH Ng et al. (2012) proposed an adaptive phototransistor sensor for line finding. Modulation and demodulation are used to solve the environment problem. K-means clustering is used to recognize the difference in the color of the line and the floor, while fuzzy logic is used to find the line relative to the sensor. A method based on map building relies on the global map for navigation decisions (Bazylev et al. 2017; Balaji et al. 2015; Oriolo et al. 1995; Saitoh et al. 2009). This will be problematic when the environment changes. Saitoh et al. proposed a method for wheeled mobile robot tracking in the middle of the corridor with a single USB camera and laptops (Saitoh et al. 2009; Martinez et al. 2007). The method uses the Hough transform to detect corridor boundaries and walls, then the robot goes along the center of corridors.

The above-mentioned methods cannot meet the robots in complex environments for complementary tasks. Automatic guided vehicle navigation systems use a guideline-based technology (Morrison et al. 2016; Reyes and Tanner Jul. 2015; Pakdaman and Mehdi 2009; Xiafu and Yong 2009). In one practical application, a mobile robot moves along a pre-designed geometry of completed search-and-rescue missions (Armesto and Tornero 2009; Rahman et al. 2005; Zhang et al. 2010; Alexiadis and Zarpalas 2013). Many researchers have proposed the application of the visual system in autonomous mobile vehicles (Du et al. 2015; Głowicki et al. 2013; Brandão et al. 2015; Luo 2015) to capture and analyze visual images of the guide lines laid on the ground to overcome the limitations of other sensors.

In the humanoid robot navigation system, the camera naturally becomes the most important sensor to obtain information of the surrounding environment (Tessier et al. 2010; Bernabe et al. 2017; Lu et al. 2017; Oriolo 2013; Faragasso 2016; Aldana-Murillo et al. 2015). The uncertainty of the family environment the variability of the placement of furniture increase the difficulty of the robot self-location. At the same time, the rich indoor information greatly reduces the speed of image processing, and puts higher requirements on the self-location algorithm (Hornung et al. 2010; Wu et al. 2014; Lobos-Tsunekawa et al. 2018; Delfin et al. 2014).

The NAO robot is a humanoid robot not exceeding a meter in height (NAO-technical overview) with the visual, auditory and tactile capabilities, able to walk, greet, dance and do some other human-like actions. Owing to its body and movements, it is attracting more and more attention (Ariffin 2015; George and Mazel 2013; Yussof et al. 2015; Okarma and Lech 2010; Ismail et al. 2009; Thuy and Cuong 2016; Shiao et al. 2013; Delgado-Galvan J 2015; Zhang et al. 2004). Combining with its lovely appearance, it can be a companion robot for a child in the family. Its indoor navigation can be utilized to grasp a tiny toy, drop litter and even walk into the room to wake up the baby.

In order to simplify the navigation and enable better adaptability to the environment, so this paper applies the line-following technology to humanoid robot by laying indoor guide lines. The humanoid robot with visual-line navigation system can help complete some relatively fixed-position tasks, in a certain sense, liberate the labor force. The main difficulty of this

research is how to generate real-time appropriate output according to the obtained image to control the humanoid robot walking and minimizing the deviation between the robot and the center of the line.

To solve this problem, we propose a high-speed line extraction method. This method not only can quickly find the path of interest from the image, but also can accurately find the forward path when encountering the intersection.

And using the proposed curvature and slope estimation parameters as input to the PID controller can reduce tracking error.

The purpose of this research is to apply visual line-following technology to a biped humanoid robot NAO and propose a high-speed line extraction method and an angle compensation proportional–integral–derivative (PID) controller to generate step control parameters to make the robot walk along the center of the black line.

This paper is organized as follows: Sect. 2 discusses the visual navigation principle of robot tracking, the line extraction method and a novel method to estimate curvature and slope. Section 3 describes the humanoid robot and the framework of line following. Section 4 presents experimental data and results of the robot visual tracking.

2 Methods

2.1 Image preprocessing and line extraction

The robot determines the direction of walking through the image captured by the camera located on its head. The desired line is extracted through a series of image processing methods.

First, the image is converted from RGB to gray using the formulas:

$$Gray = \frac{(R * 30 + G * 59 + B * 11 + 50)}{100} \tag{1}$$

Then, the Gaussian kernel is used to smooth out the grayscale image by eliminating Gaussian noise. This completes the image preprocessing operation. We now need to extract the line from the image.

When the robot walks in real-time, the image must be processed in real-time, it must guarantee on the image sampling period is completed before the arrival of a next sampled image processing.

To reduce the image processing time as much as possible and ensure the accuracy of the path extraction, this paper proposes a rectangle search method.

Assuming the length of the rectangle is l,the width is w, the center point is $P_{center}(x_0, y_0)$, and the convoluting kernel is [1,0,− 1]:

1. Determine the center point of the bottom edge of the rectangle, which is the center of the edge of the path.
2. Search the desired line in the image in the order of the upper, left, and right side of the rectangle.
3. $I(x,y)$ is a one-dimensional matrix of gray values in the image corresponding to the upper side of the rectangle

$$x = x_0 - w \tag{2}$$

$$y_0 - l/2 \leq y \leq y_0 + l/2 \tag{3}$$

$$Edge = I * Kernel > Threshold \tag{4}$$

When the value obtained by the convolution is greater than the given threshold, we consider that point as the edge of the path. If no edge is detected on the upper side, separately search the left side and right side of the rectangle separately. When the edge of the path is detected, the center point is updated to *Pcenter* (x_1, y_1). If the edge of the line is found on the right or left sides, the next rectangular search needs to rotate 90 degrees in that direction.

4. Connect all center points in sequence with a straight line. The route extracted is considered as the target path.

Under ideal conditions, the image after Gaussian filtering and edge extraction contains only the edge information of the path. However, under actual conditions, including the light intensity as well as the edge information of surrounding objects will interfere with it. Therefore, the edge information must be obtained. The matrix data is processed twice. The pseudocode of this algorithm for this data processing is as follows:

---

for index in I_i (I_i is an array of index values of the edges in the i^{th} row of the image)
  if $index_{j+1} - index_j < Threshold_{Min}$
    delete $index_{j+1}$ in I_i
then **for** index in I_{new}
  Avg ($index_k$, $index_{k+1}$) -> $center_i$
if $center_i$ is more than one vlaue
  for value in $center_i$
    Fabs (value - previous point value)
    - > Difference
  Choose value with minimun Difference as reliable point in I_i

---

Explanation: This method greatly reduces the amount of data for image processing. At the same time, by setting the length and width of the rectangle (at the first statement to the fifth statement in the pseudocode), the extraction of the path can achieve different effects (at the sixth statement to the ninth statement in the pseudocode). Properly reducing the long side of the rectangle provides good noise immunity from the noise outside the target path (at the tenth statement to the eleventh statement in the pseudocode). Decreasing the short side of the rectangle, the path restored by connecting the center point is closer to the original path, but it will increase the processing time loss in response. We should choose the appropriate rectangle length and width according to the actual environment.

Since the order of the rectangular search is in the order of the upper, left and right side, when the change of the target path in the forward direction is small, the gradient of the path in the X-axis direction is large. Therefore, search using the upper side of the rectangle can be performed. Find the center point of the path. When the direction of the target path changes greatly, then the gradient of the path in the Y-axis direction is large, and the center point of the path can be easily found by searching the sides of the rectangle. When a crossover path is encountered, the rectangular search first looks for a path location point that has a small change in the path direction. Therefore it is more robust to correctly extract the forward path when the crossover path is encountered.

The path extracted through the rectangular search is shown in the Fig. 1. The green rectangle is the search line. The red point is the center position of the found path. The center

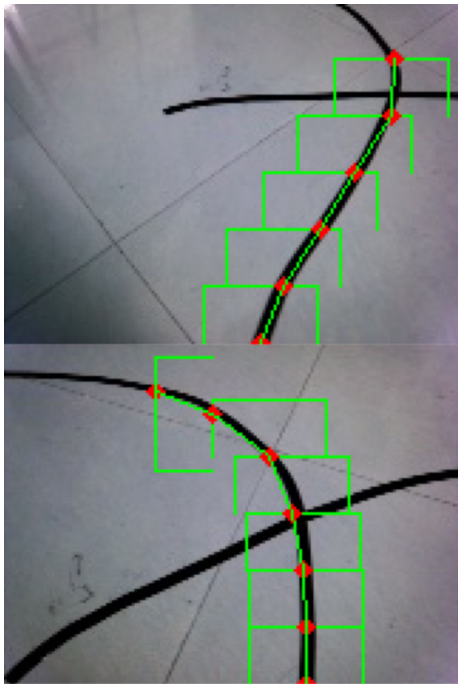


Fig. 1 Extract line with rectangle search

of the path and target path extracted by this method can be clearly seen. The line is basically the same. The maximum value of the data processing amount of this method is only $(2 * w + l) * n$ pixels, n is the number of rectangular searches, assuming $w = 20, l = 40, n = 6$, then $Max = 480$ pixels. For a 160*120 image, the amount of data processed is equivalent to 1/40 of the entire image processing, which greatly increases the processing speed and ensures the robot's real-time control during the movement.

2.2 Forward velocity generator

When walking along the trajectory, the robot should minimize the deviation between its center of mass and the centerline of the path, so when the curvature of the trajectory changes, it is able to automatically adjust the speed. The tracking effect reduces the error. The curvature can be used to represent the degree of curvature at a point on the curve, since the path we extracted is formed by the sequential connection of the discrete points of the path obtained by the rectangular search.

Therefore, using traditional methods to calculate the curvature of a point on the path must first fit a curve according to these discrete points, and then calculate the curvature of the point using the formula of curvature. Although the curvature calculated by this method is more accurate, the time loss is greater. Therefore, this paper proposes a simplified path curvature estimation method to predict the curvature of the current visual path. This method defines the curvature of the path by calculating the angle of rotation between the starting point and the ending point obtained by rectangular search:

$$C = \sum_{i=1}^{n} (\theta_{(i,i+1)}) \tag{5}$$

where θ is the rotation angle value between two adjacent sampling points. The greater the value of C, the greater the degree of curvature of the path. The robot needs to maintain a large angular velocity when walking so that it does not deviate from the path. However, there is a maximum value of the rotational angular velocity of the robot's footsteps. If the robot veers away from the target path, the deviation from the target path must be reduced by reducing the speed of the robot.The formula for generating the forward velocity is as follows:

$$V_{forward} = V_{Max} - \alpha * C - \beta * \sqrt{e_{offset}^2 + e_{theta}^2} \tag{6}$$

where V_{Max} is the maximum forward speed that the robot can set, and α and β are the sensitivity coefficients. The larger the value of the coefficient, the higher the sensitivity of the speed to the path curvature and the robot position deviation. The e_{offset} and e_{theta} indicate the positional and angle deviation between the robot and the target path.

2.3 Focus on

The faster the robot walks along the path, the faster the robot needs to take the position of the path in the field of view as the reference for the movement. Therefore, the Focus On module adaptively adjusts the reference point of the path according to the real-time speed of the robot. This reference point is the path the robot will walk to next. Then take this reference point position and the path starting point position as the controller's input parameters. The coordinates of the reference point are calculated as follows:

$$X_f = I_{width} - V_{forward} * k \tag{7}$$

$$Y_f = \{I(x, y) = L_{line}, x = X_f\} \tag{8}$$

where I_{width} represents the width of the image, $V_{forward}$ is the forward speed of the robot, k is the approximate conversion factor of the pixel distance and the actual distance, $I(x, y)$ represents the gray value of the image, and Lline represents the target path. The formula (8) obtains the column coordinate value of the intersection of the target path and the row where the reference point is located.

2.4 Classical control method

The Basic visual-based autonomous tracking method of a biped robot is as follows: Assume that the humanoid robot can walk on a flat surface and relies on the difference of gray value between black lines and the background to identify the navigation paths by image processing. We can find the path's center line by extracting the edge point. On the basis of the center's location determines the robot's attitude and the relative position of the path to control robot autonomous walking forward in the right direction. The NAO robot visual navigation principle is shown in Fig. 2. The navigation parameters angle deviation and position deviation are the primary inputs to the controller. The controller then generates the control parameters to make the robot change its attitude; the angle deviation and position deviation are updated accordingly. The updated value obtained by the vision sensor and image processing are sent to the controller again.

The simple straight line path-tracing model is shown in Fig. 3. The NAO robot camera gets the navigation path of the model as a straight line, through the edges of image identification algorithms for the path and the path's center line. The distance between the centerline and

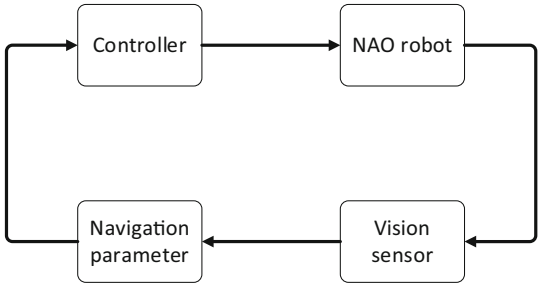


Fig. 2 Robot navigation schematics

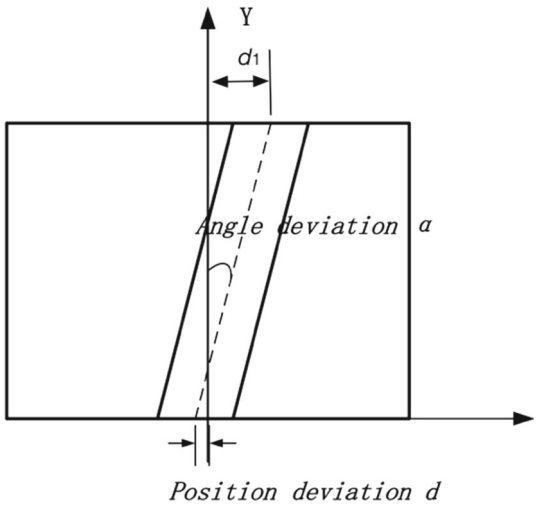


Fig. 3 Linear model of robot navigation

the center of the image bottom is the position deviation d. The angle deviation between the center line and the Y-axis is the angle α.

$$\tan\alpha = \frac{d_1 - d}{I_{height}} \tag{9}$$

I_{height} is the height(pixel) of the image.

When walking a curved path, the best track is to make the robot's direction tangent to the curved path. This is different from the straight path tracking model in that the angular deviation α cannot be obtained directly through the edge. In the curved path model shown in Fig. 4, at the mid-point of the connection of the curve directions V_1 and V_0 in the curve tangent direction, the linear model methods will make the β-α bias.

In order to compensate for this angle, this paper proposes a tangent slope estimation method because the slope depends on the magnitude of the lateral error of the upper and lower center points in the path. When the curvature of the path point increases, the method of compensating the lateral deviation is used for the reference point slope estimation.

In Fig. 5, assume that Start is the starting point of the path, point A is the position that the robot will reach in the next footstep, and dotted line b is the point A point tangent. The horizontal deviation of this position from the path starting point is d1, which corresponds

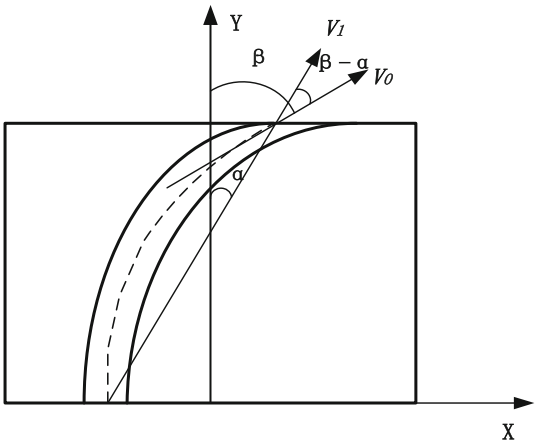


Fig. 4 Curved path model of robot navigation

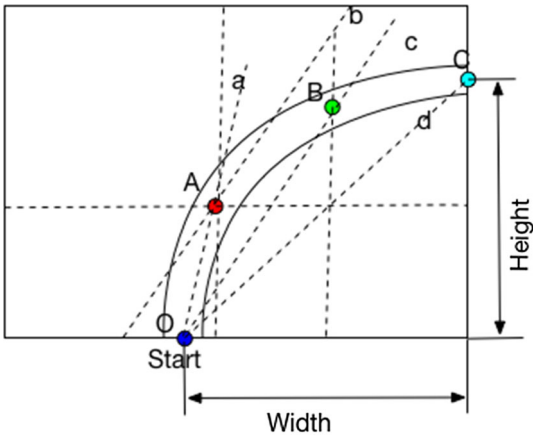


Fig. 5 Slope approximate

to the steering angle θ_1. Obviously, this steering angle cannot make the robot move forward tangent to the path. The θ_2 calculated by the offset compensation is as follows:

$$\tan\theta_2 = \frac{width * l_A}{height * l_B} \tag{10}$$

where l_A, l_B are the distances from point A, B to the bottom of the image in the Y direction respectively: *width* and *height* are the distances from the starting point and the end point of the path to the sides of the image and the bottom direction. The steering angle obtained by this method corresponds to a point B on the path, and the point B is determined by the point A. The point where the dotted line in the X-axis direction intersects the OC is determined in the Y-axis direction intersecting the path centerline, and OB is two points. The slope of the line can be approximately equivalent to the tangent slope at point A. The method can quickly determine the angle of rotation of the robot during walking. It does not need to obtain the curve equation by the binomial fitting of the sampling point to calculate the tangent slope of the point. The method is obtained by fitting with the binomial formula. The steering angle error is compared with the method of uncompensated lateral deviation and approximate tangent estimation. The results are shown in Fig. 6.

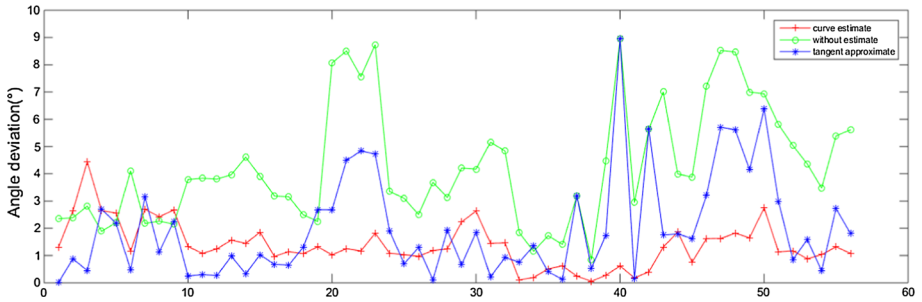


Fig. 6 Compare angle deviation with three different methods

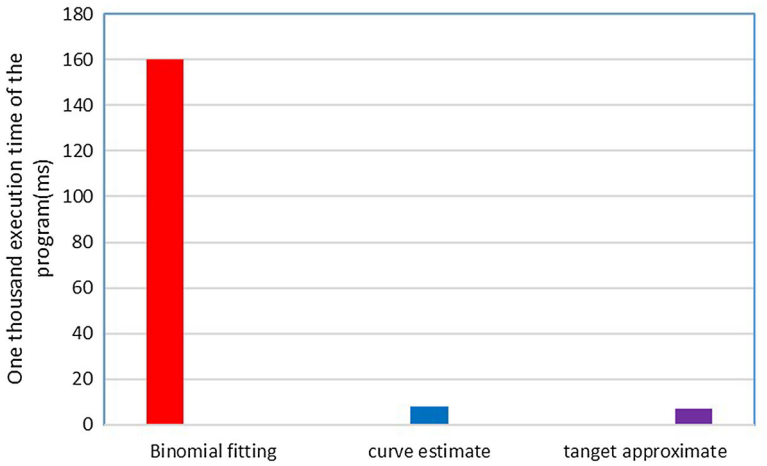


Fig. 7 Time of one thousand program execution

It can be seen from the Fig. 7 that the deviation of the steering angle value obtained by this method from the angle obtained by binomial fitting is basically around 1 degree, enabling the robot to move forward during the course of walking along the arc. Better to stay tangential to the route, and at the same time nearly 25 times faster than the binomial fit in the running time of the program.

2.5 Structure of PID controller

The PID controller is a common feedback loop component in industrial control applications and consists of a proportional element P (which provides proportional control), an integral element I (which can eliminate steady-state errors), and a derivative element D (which can speed up the response of large inertial systems and weaken the overshoot trend).

In this paper, two PID controllers are used: one to control the lateral speed and another to control the angular velocity of the humanoid robot.

Figure 8 is a block diagram of a path tracking control system for a humanoid robot. Input signals to the control system are expected values of positions and expected angles. Position and angle errors are used as inputs to the PID controller, while the outputs are V_x, V_y and w parameters for driving the robot.

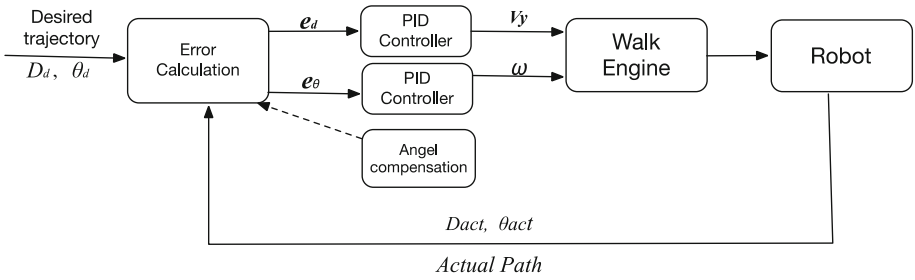


Fig. 8 Block diagram for humanoid robot trajectory tracking control

The general expression of the discretized PID controller is:

$$c(n) = K_p * e(n) + K_t * \sum_{0}^{n} e(n) + K_D * (e(n) - e(n-1)) \tag{11}$$

The first PID controller controls the positional deviation of the robot so that the distance between the center of gravity and the target path is minimized as it advances. The second PID controller controls the rotational angular velocity of the robot's footsteps, so that the robot keeps the forward direction tangent to the path, regardless of whether it is a curve or a straight line. If the angle of deviation between the reference point and the starting point is directly used as the input parameter of the controller, the robot will have a large direction deviation and position deviation when tracking the curved path. Worse, it may lose the path information in the field of vision, resulting in tracking failure. Therefore, when calculating the angle error, the angle of the tangent direction of the reference point and the difference of the target angle are estimated using the angle compensation method as follows:

$$e_\theta(n)^* = e_\theta(n) + c \tag{12}$$

where c is the angle value of the compensation.

Take the sin curve as the target path and use traditional PID controllers and PID controllers with angle compensation for comparison. The tracking results are shown in the Fig. 9.

There is a significant lag between the path trajectory of the classical PID controller and the target path. The lag of the PID controller with angle compensation for the path tracking of the curve is very small, and the error value is smaller.

3 Robot description and framework for visual line-follower

This paper is based on visual robot autonomous tracking for the humanoid robot NAO, which developed by Aldebaran as a hardware platform. The NAO hardware was designed and manufactured to ensure smooth actions. It is also equipped with a variety of sensors and can be programmed using multiple languages (such as C++, Python, Java, etc.) under different operating systems (such as Linux, Windows or Mac OS).

The NAO has two 1280×960 resolution cameras with up to 30 frames per second. Its position and field-of-view are shown in Fig. 10. The path information on the ground is the main source of information for advancement. There is no significant value for image information at a remote location. Therefore, the camera at the bottom of the NAO robot is selected, and the head of the NAO robot is turned down at a fixed angle. In order to obtain

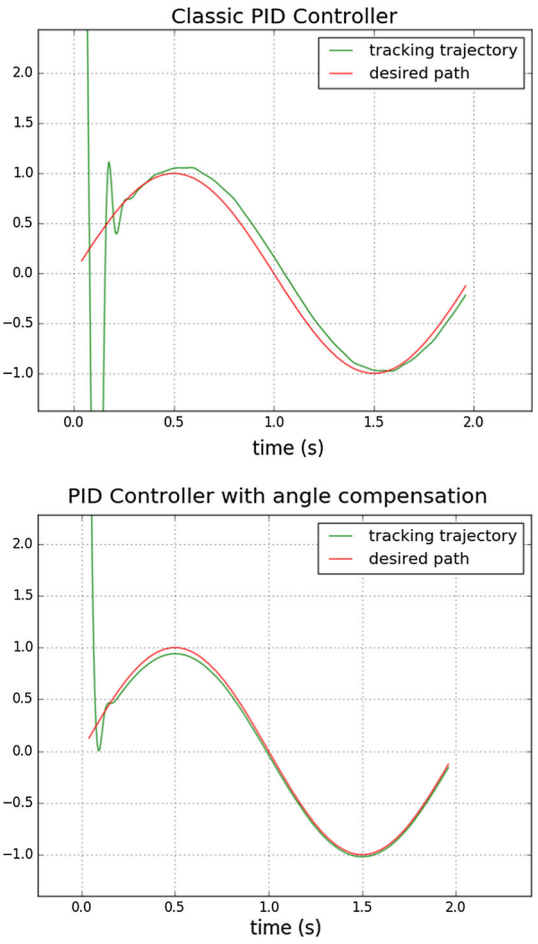


Fig. 9 Comparison PID controller with and without angle compensation

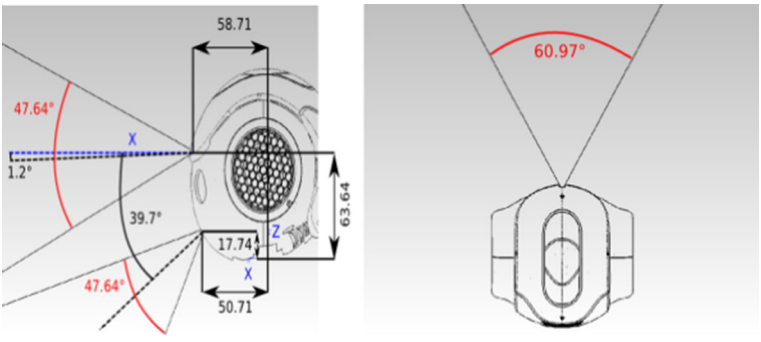


Fig. 10 NAO robot head camera field of view

the path information of the foot and not cover the visual field with its own feet, the initial standing posture and the camera's field of vision are shown in Fig. 11.

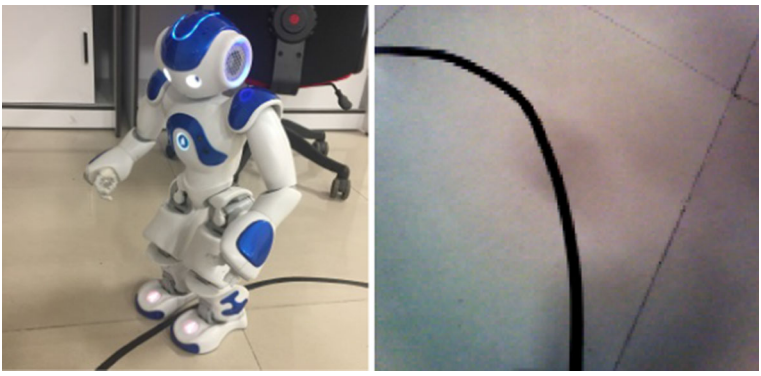


Fig. 11 NAO robot stands initial attitude and its vision

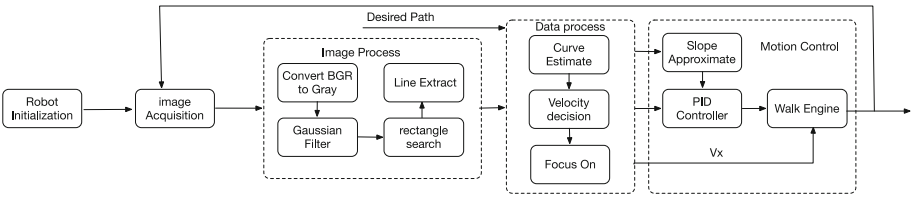


Fig. 12 Framework for humanoid robot line following

Python is now a very popular programming language. It has a rich and powerful library and one can quickly and easily write application software with application value. OpenCV is a cross-platform computer vision library based on a BSD license (open source) distribution that can run on Linux, Windows, Android and Mac OS operating systems. It is lightweight and efficient. It consists of a series of C functions and a small number of C++ classes. It also provides interfaces for languages such as Python, Ruby, MATLAB, and many common algorithms for image processing and computer vision. This paper uses the 32-bit Python2.7 and OpenCV2.4 to perform visual line-follower application development for NAO robots in the Windows environment. The overall program design architecture is shown in Fig. 12. It is divided into four modules: initialization, image processing, data processing, and motion control. The initialization module includes the initialization of internal parameters of the robot, such as resolution, color space, camera selection, etc. The image processing module mainly preprocesses the image acquired by the camera, including binarization, Gaussian deconvolution and edge extraction. Then the preprocessed data is passed through a data processing module to obtain parameters (*Vx, Vy, w*) for driving the robot walking. The module can estimate the curvature of the path in the image to generate the forward velocity parameter *Vx*, and calculate the position and angular deviation from the target path. In the motion control module, the deviation from the target path will be the input of the PID controller which gives the other two parameters *w* and *Vy* that control the robot following the line.

The biped robot locomotion model is shown in Fig. 13. When the robot wants to walk along the curve, its both feet must rotate a suitable angle while moving forward. In this case, only the forward velocity *Vx* and angular velocity *w* are required. If the center of mass of the robot has a large deviation from the line, then a lateral movement (lateral velocity *Vy*) is needed to reduce the error.

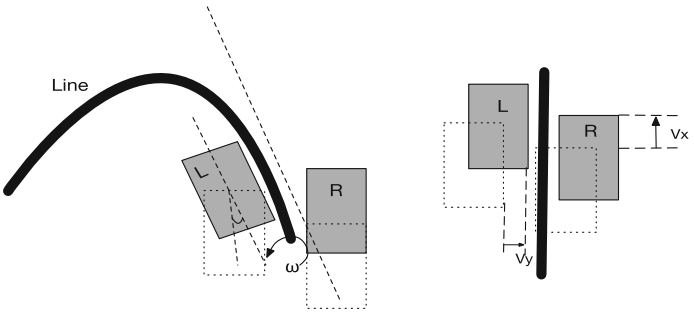
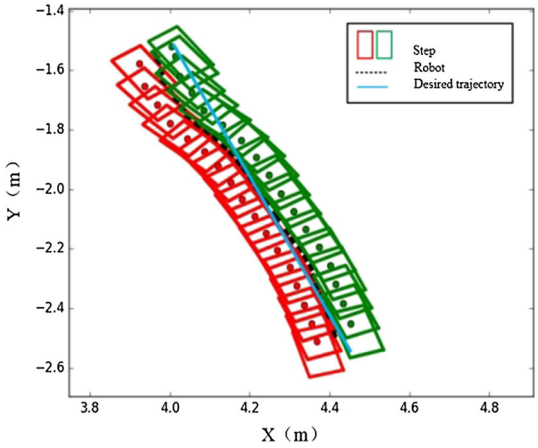


Fig. 13 Biped robot locomotion model

Fig. 14 Straight line footsteps



For a biped humanoid robot NAO motion engine, we invoke *move()* method in AlMotion-Proxy class provided by the Aldebaran software.This function contains three parameters *x, y, theta*. *x* is the velocity along X-axis, in meters per second. Negative values are used for backward motion. *y* is the velocity along Y-axis, in meters per second. Positive values are used to go to the left. *theta* is the velocity around Z-axis, in radians per second. Negative values are used to turn clockwise. The parameter *x* is set with forward speed generator module. The PID controller is used to generate a suitable value of *y* and *theta* to drive the robot walking along a guide line.

4 Experiments

This paper lays out the accuracy and stability of the straight line, S-curve, circular, and octagonal verification theory on the ground. The black tape with a width of about 15 mm was used in this article and was stuck on a light marble floor. Figures 14, 15, 16 and 17 show the target path and the robot's actual tracking step. The red and green rectangles are the footprints traveled by the NAO robot. The blue solid line is the target trajectory, and the black dotted line is the actual trajectory. Figures 18, 19, 20 and 21 gives the tracking deviation curve.

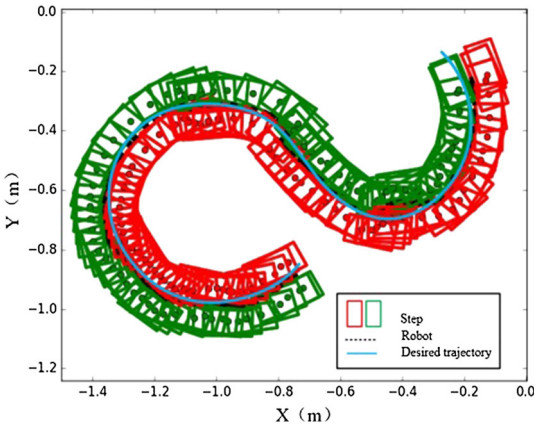


Fig. 15 S-line footsteps

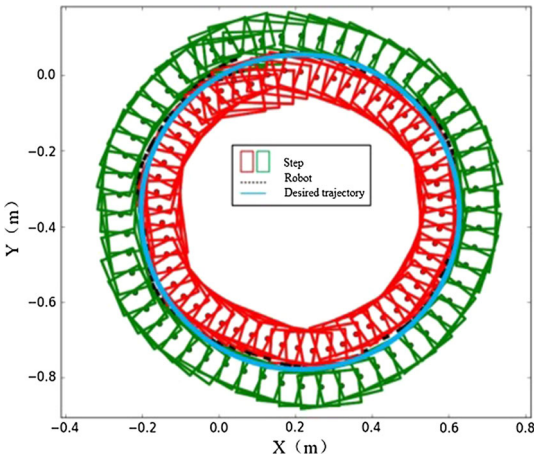


Fig. 16 Circle footsteps

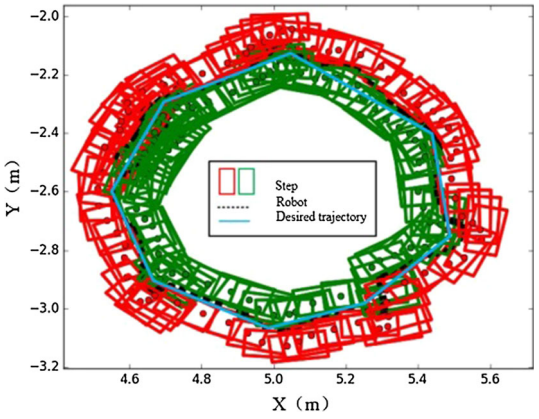


Fig. 17 Octagon footsteps

As they can be seen in Figs. 14, 15, 16, 17, 18, 19, 20 and 21, the robot can be quickly placed toward the center of the straight line with the robot initially placed at a position away from the straight lead, and kept in a straight line at a position where the deviation is small.

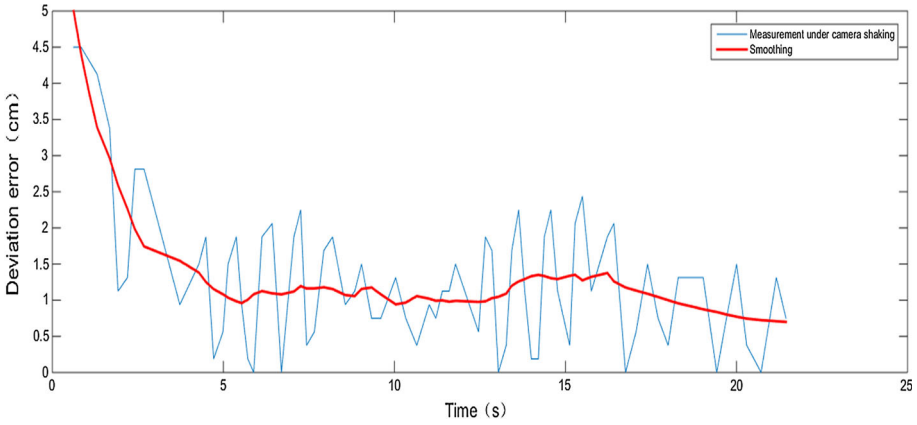


Fig. 18 Deviation for straight line

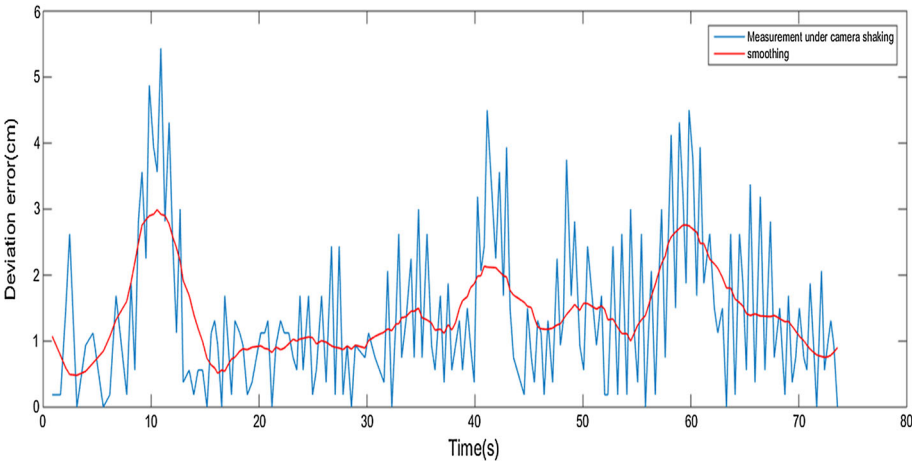


Fig. 19 Deviation for shape S line

The corresponding error curve shows that the robot reduces the deviation from 5 cm to about 1 cm in the five steps, and it can be stabilized at this level. Similarly, when the robot walks along a circle, there is a large fluctuation in the deviation at first, but it can quickly adjust and remain at an error of about 1 cm. As the robot moves along the s-line, it is apparent from the error curve that three relatively large error values appear. As the robot progresses along the s-shaped path, it is apparent from the error curve that three large error values appear. Because the path slope changes and the robot's motion still has a slight delay, so the large deviation occurs when the robot enters and outs the bend. As they can be seen from Figs. 18, 19, 20 and 21, when the robot moves along the octagon, there will be a large deviation at the corner. The reason is that the robot will select a point at an uncertain distance in front of the forward as a reference according to the curvature of line without paying attention to subtly change at the corner.

Since the robot has left and right shaking during walking, the error is smooth filtered. It can be seen from the figure that for the straight path and the circular path, the robot. The

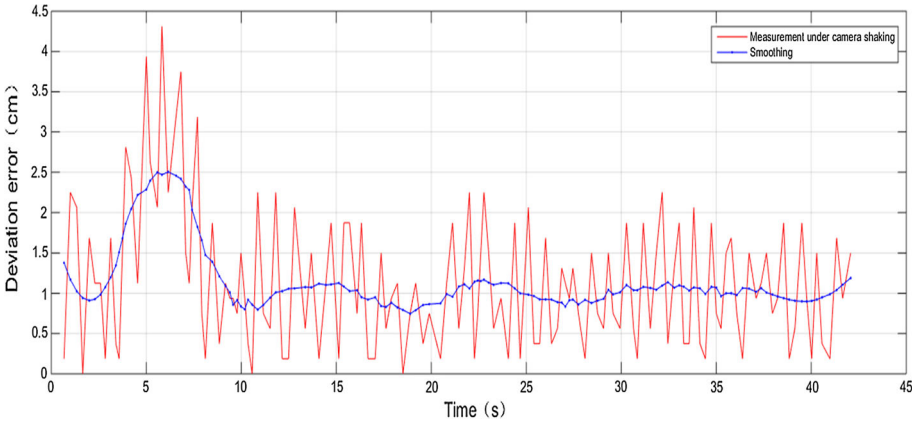


Fig. 20 Deviation for circle

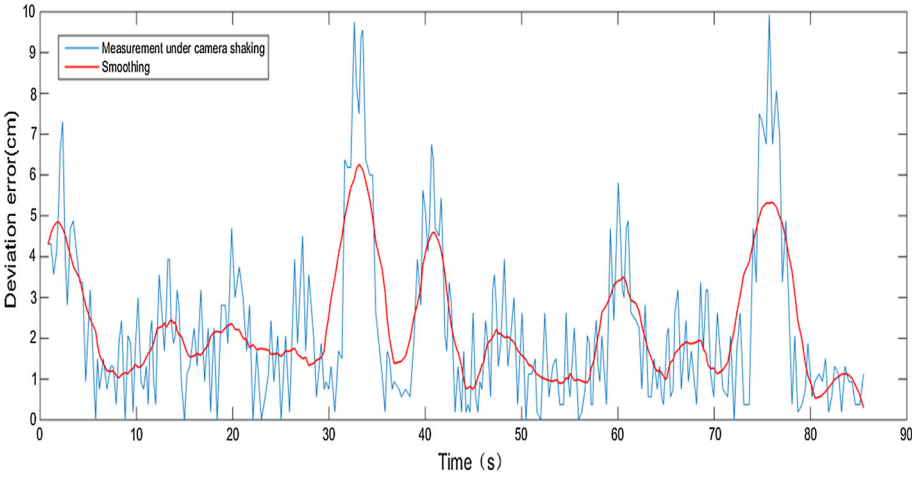


Fig. 21 Deviation for octagon

tracking error is small, about 1 cm. For the sigmoid curve, a large distance deviation appears at the change of curvature, but the overall error is below 2 cm. For the octagon, there is a large deviation at the corner of each side.

In summary, it can be seen from the above four experiments that the humanoid robot with visual line-following navigation system exhibits the better tracking ability and can make timely adjustments when there is a large deviation between the robot position and the desired line. This allows the robot to help these tasks that require the visual navigation in a home or indoor environment.

5 Conclusion

This paper presents a visual line-following navigation system for a humanoid robot. In our system, we use the humanoid robot NAO and its own vision sensor to realize the navigation. The rectangular search method proposed in this paper can quickly and accurately extract the

path in the image, which not only improves the real-time performance of the robot visual navigation, but also lays a foundation for the robot to accurately follow the line.

The proposed forward speed generator and PID controller with the angle compensation allows the robot to select the appropriate forward speed according to the curvature of the path to ensure that the robot does not lose the path in the field of view when the walking speed is too fast and the path is too curved.

The experimental tests show that the humanoid robot NAO can generally walk along the line of different shape with low deviation. Although the position of the robot and the desired line will have a relatively large deviation when the curvature of the path changes or a corner is appeared, the robot can quickly adjust the direction of advancement to ensure that the error with the desired path stays low. However, the camera jitter causes interference on the track, which needs to be improved in later studies.

Acknowledgement The authors deeply acknowledge the financial support from Xiamen University of Technology, Fujian, P.R. China under the Xiamen University of Technology Scientific Research Foundation for Talents plan.

Authors' contribution The contributions of this paper are: (1) a line-following method that uses the Gaussian filter and a simple convolution kernel to extract line information and improves the processing speed by using rectangle search; (2) A curvature approximation method to generate forward velocity; and (3) a PID controller with angle compensation to adapt to different curves. When applied collectively, the humanoid robot NAO's navigation performance was comparatively robust and accurate.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Aldana-Murillo NG, Hayet JB, Becerra HM (2015) Evaluation of local descriptors for vision-based localization of humanoid robots. In: Mexican conference on pattern recognition. Springer International Publishing, pp 179–189

Alexiadis DS, Zarpalas D (2013) Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras. IEEE Trans Multimed 15:339–358

Antonelli G, Chiaverini S, Fusco G (2007) A fuzzy-logic based approach for mobile robot path tracking. IEEE Trans Fuzzy Syst 15(2):211–221

Ariffin IM et al. (2015) Vision tracking application for mobile navigation using Humanoid robot Nao. In: International symposium on micro-nanomechatronics and human science IEEE, pp 1–7

Armesto L, Tornero J (2009) Automation of industrial vehicles: a visionbased line tracking application. In: Proceedings of IEEE conference on emerging technologies & factory automation, pp 1–7

Balaji V, Balaji M, Chandrasekaran M et al. (2015) Optimization of PID control for high speed line tracking robots. In: IEEE international symposium on robotics and intelligent sensors. IEEE, pp 147–154

Bernabe A, De San Dios MD, Ollero A (2017) Efficient integration of RSSI for tracking using wireless camera networks. Inf Fusion 36:296–312

Brandão AS, Martins FN, Soneguetti HB (2015) A vision-based line following strategy for an autonomous UAV. In: International conference on informatics in control, automation and robotics IEEE, pp 314–319

Tessier C et al (2010) Map aided localization and vehicle guidance using an active landmark search. Inf Fusion 11(3):283–296

Delfin JH, Becerra M, Arechavaleta G (2014) Visual path following using a sequence of target images and smooth robot velocities for humanoid navigation. In: IEEE-RAS international conference on humanoid robots IEEE, pp 354–359

Delgado-Galvan J et al. (2015) Vision-based humanoid robot navigation in a featureless environment. In: Mexican Conference on pattern recognition. Springer, Cham, pp 169–178

Du X, Tan KK, Htet KKK (2015) Vision-based lane line detection for autonomous vehicle navigation and guidance. In: Control conference IEEE, pp 1–5
Faragasso A et al. (2016) Vision-based corridor navigation for humanoid robots. In: IEEE international conference on robotics and automation. IEEE, pp 3190–3195
George L, Mazel A (2013) Humanoid robot indoor navigation based on 2D bar codes: application to the NAO robot. In: IEEE-RAS international conference on humanoid robots IEEE, pp 329–335
Głowicki M, Butkiewicz BS (2013) Autonomous line-follower with fuzzy control. In: Signal processing symposium IEEE, pp 1–6
Hornung A, Bennewitz M, Strasdat H (2010) Efficient vision-based navigation. Auton Robots 29(2):137–149
Bazylev D, Popchenko F, Ibraev D, et al. (2017) Humanoid robot walking on track using computer vision. In: Control and automation, IEEE, pp 1310–1315
Ismail AH et al. (2009) Vision-based system for line following mobile robot. In: IEEE symposium on industrial electronics & applications. ISIEA 2009, pp 642–645
Lobos-Tsunekawa K, Leiva F, Ruiz-Del-Solar J (2018) Visual navigation for biped humanoid robots using deep reinforcement learning. IEEE Robot Autom Lett 3:3247–3254
Luo B et al. (2015) Research on mobile robot path tracking based on color vision. In: Chinese automation congress IEEE, pp 371–375
Martinez S, Cortes J, Bullo F (2007) Motion coordination with distributed information. IEEE Control Syst 27(4):75–88
Morrison JG, Gavez-Lopez D, Sibley G (2016) Scalable multirobot localization and mapping with relative maps: introducing MOARSLAM. IEEE Control Syst 36(2):75–85
Ng KH, Che FY, Su ELM et al (2012) Adaptive Phototransistor Sensor for Line Finding. Procedia Engineering 41(41):237–243
Okarma K, Lech P (2010) A fast image analysis technique for the line tracking robots. In: Artifical intelligence and soft computing. Springer Berlin Heidelberg, pp 329–336
Oriolo G, Ulivi G, Vendittelli M (1995) On-line map building and navigation for autonomous mobile. In: Proceedings - IEEE international conference on robotics and automation, vol 3. pp 2900–2906
Oriolo G et al. (2013) Vision-based trajectory control for humanoid navigation. In: IEEE-RAS international conference on humanoid robots IEEE, pp 118–123
Pakdaman M, Mehdi M (2009) Design and implementation of line follower robot. In: Second international conference on computer and electrical engineering, pp 585–590
Rahman M, Rahman MHR, Haque AL, Islam MT (2005) Architecture of the vision system of a line following mobile robot operating is static environment. In: 9th IEEE international multitopic conference, pp 1–8
Reyes LAV, Tanner HG (2015) Flocking, formation control, and path following for a group of mobile robots. IEEE Trans Control Syst Technol 23(4):1268–1282
Roy A, Noel MM (2016) Design of a high-speed line following robot that smoothly follows tight curves. Comput Electr Eng 56:732–747
Saitoh T, Tada N, Konishi R (2009) Indoor mobile robot navigation by center following based on monocular vision. In: Computer Vision, In-the Publishers, pp 352–366
Shiao YS, Yang JL, Su DT (2013) Path tracking laws and implementation of a vision-based wheeled mobile robot. Proc Inst Mech Eng Part I J Syst Control Eng 223(6):847–862
Thuy PX, Cuong NT (2016) Vision Based autonomous path/line following of a mobile robot using a hybrid fuzzy pid controller. In: Hnkh Toàn Quốc Lần Thứ 3 Về Điều Khiển & Tự Động Ho
Wu JJ et al (2014) A real-time method for motion blur detection in visual navigation with a humanoid robot. Acta Autom Sin 40(2):267–276
Xiafu L, Yong C (2009) A design of autonomous tracing in Intelligent vehicle based on infrared photoelectric sensor. In: International conference on information engineering & computer science, pp 1–4
NAO-technical overview http://doc.aldebaran.com/2-1/family/robots/dimensions_robot.html. Accessed 5 May 2014
Yan Lu, Song D (2017) Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. IEEE Trans Robot 31(3):736–749
Yussof H et al (2015) Sensor based mobile navigation using humanoid robot Nao. Procedia Comput Sci 76:474–479
Zhang J, Wang N, Wang S (2004) A developed method of tuning PID controllers with fuzzy rules for integrating processes. In: Proceedings of the IEEE American control conference 2004, pp 1109–1114
Zhang L, Zhuan X, Gao X (2010) Design and implementation of a Vision based 4-wheeled line track robot. In: 2010 WASE international conference on information engineering, pp 3–6