The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Slope-Based Point Pursuing Maneuvers of Nonholonomic Robots using FPGA

Ying-Hao Yu, S. Kodagoda, and Q.P. Ha

Abstract— **Steering maneuver is essential in robotic motion planning. Despite a lot of steering mechanisms successfully developed in past years, for miniature robots, real-time computation is still a limitation for robot path tracking. The design issues in cooperative control of battery-powered nonholonomic robots rest with the complicacy of the control strategies, the low power consumption and real-time processing capability. Conventionally, the improvement of computing speed mostly relies on the increment of the system clock and often results in some transient loss. Thus, an elaborate control algorithm developed for PC might not work on an embedded system. This paper presents a comprehensive steering algorithm which, via issuing predicaments for computation, will dramatically reduce the resource usage in hardware circuit design. The proposed algorithm is implemented on an embedded system for ubiquitous robotics using the field programmable gate array (FPGA) technology.**

I. INTRODUCTION

THE most popular robotic steering systems of wheeled robots are the differential-drive and Ackermann steering. The motion of the differential-drive robots are controlled by individually controlling the motors driving each wheel. The same speed of motors produces straight line motion whereas different speeds cause the robot to turn. On the other hand, the Ackermann steering vehicles use two separate actuators for driving and turning [1].

Steering control of non-holonomic robots in indoor environments can be modeled by using the space coordinates such as the turning angle, angular turning speed, and orientation of robots due to their relatively slow speed of motion [2]. The lateral sliding problem in an indoor environment is not as serious as in a high speed car [3]. During an autonomous driving scenario, two essential parameters can be considered for steering strategies. The first parameter is called the look-ahead distance, i.e. the specified virtual distance in front of the robot. In real-world applications, the specified look-ahead distance helps a vehicle to decide the deviation from the central line of road [4], or it can also be extended to calculate accordingly the speed with respect to ground [5]. The second critical parameter is the pursued point, which is the destination of a look-ahead distance. Consequently, a long and sweeping route can be composed from sequential pursuing points, and the expected trajectory of robot is achieved by different angles of turning. If we assume a wheeled indoor robot is tracking on planned points, the shortest path tracking will depend on the least swing of every look-ahead path.

There have been many techniques successfully developed in past years to obtain optimal path tracking for differential-drive and car-like vehicle steering [6,7]. Minimizing errors in vehicle path tracking can be achieved via the use of rigorous control techniques [4,5]. However, autonomous steering requires the availability of measurable parameters from the robots and their environment, resulting often in much computing effort for a higher accuracy. In an indoor environment, real-time computation remains a limitation for robot path tracking with miniature robots. The complicacy of the control strategies, the low power consumption and real-time processing capability are design issues for cooperative control of battery-powered nonholonomic robots. Even implementing complicated strategies for vehicle control on an embedded system, the overall motion may also inherit the problem of driving stability. This problem has been discussed in [8], revealing the swing of steering control with short horizons, and is generally ascribed to the requirement of a drastic reaction to a large deviated angle for error correction to maintain the desired path. Reactive tracking for a group of robots is proposed in [9] using the variable structure methodology, but also facing implementation difficulties. The driving stability problem could be alleviated by employing a larger look-ahead distance [10]. Unfortunately, such long look-ahead schemes may not be realistic in an indoor environment where the driving space is limited.

In contrast to control theoretic solutions, the pure-pursuit task executing the shortest path between two points can be realized only with a single turning [10]. Although the pure-pursuit algorithm is an efficient mechanism to reach the expected destination, it does not guarantee the orientation of the robot aligning with the path at the destination [11]. The problem is originated from the curve steering nature which always requires to maintain an incident angle to the path. Towards a valid solution, the use of a behavior-based model for robot steering seems to be feasible with small values of speed and acceleration of a robot navigating in an indoor environment. The steering behavior can be represented by geometric representations which have been used to coin the

This work was supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

The authors are with the ARC Centre of Excellence for Autonomous Systems (CAS), Faculty of Engineering and Information Technology, University of Technology, Sydney, PO Box 123 Broadway NSW 2007 Australia. (E-mail: {YingHao.Yu|sakoda|quangha}@eng.uts.edu.au).

shortest path planning in mobile robotics. Dubins first estimated the shortest path between two points in an obstacle free space by combining clockwise turning (R), anticlockwise turning (L), and a straight line driving (S), e.g. RSL or LSR maneuvers [12]. Although it proposed a good paradigm in behavior-based steering, finding the start and end point of a tangent between two turning arcs is not an easy task for embedded systems [13].

From [7], one can see that the shortest path for nonholonomic robots in an obstacle free environment comprises in general a trajectory combining three labels. Motivated by this framework and Dubins' turning mechanisms, we introduce in this paper a slope-based are-line-arc (SBALA) algorithm, aiming at implementation of robot steering on a programmable chip for a compromised shortest path that is feasible for the chip capacity. Notably, this algorithm involves reduced computing effort, and hence allowing the steering maneuvers to be realized directly on an embedded system with hardware circuit design. This targets directly the rising trend of ubiquitous robotics, which mainly entails embedded systems with limited computational power. Therefore, the possibility that the proposed algorithm can be implemented on an embedded system using the FPGA technology underlines the significance of the idea [14].

The remainder of this paper is organized as following: Section II describes the SBALA geometric algorithm. Section III provides the development of the proposed steering mechanism on an FPGA kit for a differential-drive platform, the Eyebots. Experimental results and discussion are included in Section IV. Finally, Section V concludes the paper.

II. Slope Based Arc-Line-Arc Algorithm

A great deal of research has been devoted to the problem of planning collision-free optimal trajectories for nonholonomic mobile robots that move forward only [12] or move forward and backward [15], whereby the shortest paths comprise straight line segments and arcs subject to bounded turning radius [6].

A. SBALA Algorithm

By considering the least computing effort and resource usage on embedded systems, particularly for hardware circuit design, the angular and trigonometric representations are firstly replaced by the instant tangent slope of the robot trajectory on a 2D plane. Figure 1 shows an example of the slope based arc-line-arc algorithm (SBALA) in 2D representation. Here, we consider only cases when a nonholonomic mobile robot is pursuing a target that is away at least by four minimal turning radii.

The SBALA is mainly composed of seven critical points, from A to F. Point A is the expected pursuit destination, and B is the central point of the robot at its initial location. Slope m represents the trajectory tangent at point A in (x,y)-coordinates, and another slope, denoted m', perpendicular to the trajectory at A in the direction $\overline{AF}$. The look-ahead distance shown in Fig. 1 is the straight distance between point A and B. Point C is the perpendicular intersection point from B and the desired orientation at point A. To provide the turning reference of the robot, $\overline{AC}$ is separated into four equal sections, and R_0 is the turning radius, equal to $1/4AC$ at point B. More division numbers may be used for differential-drive robots while a section length will be limited by the maximum turning curvature of a vehicle-like robot. The orientation of the robot at B shown in Fig. 1 is conveniently set aligned to $\overline{BC}$.

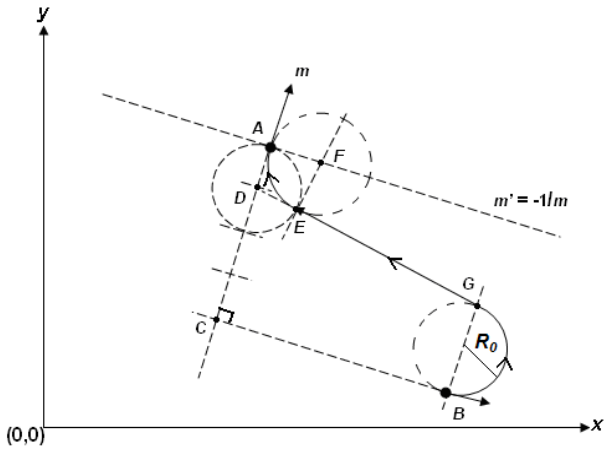


Fig. 1. A representation of SBALA on 2D plane.

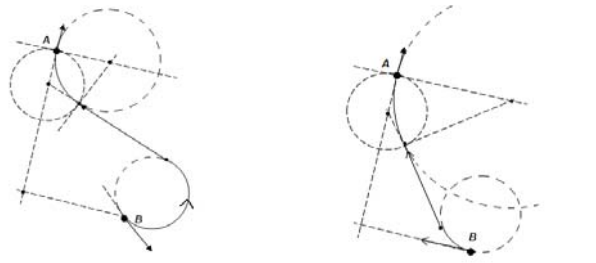
Next, the robot starts to pursue the point A with an initial anticlockwise turning (L). Line $\overline{DG}$ is the tangent of the robot's first turning arc. After the robot has reached point G, it starts a straight line driving (S) toward point D, which is determined by $\overline{AD}=1/4\,\overline{AC}$. Point E is the starting off point of second arc, located on the ring which circles point D with radius $1/4AC$. Once the robot reaches point E, the second arc can be derived with radii $AF=EF$, completing the path B to A with an aligned orientation, where point F is the center of the second arc.

B. General Cases of SBALA Algorithm

Considering cases of robots moving ahead only, Figure 2 shows four examples of SBALA maneuvers using R, S and L combinations. Those diagrams show realistic steering to the pursued point. Different turnings on both sides of $\overline{AC}$ can be represented as LSR, RSR, RSL, and LSL. Thereby, even on the straight path, the curvature of the second arc will be approximated to zero, so the robot nearly keeps a straight line for driving on the look-ahead path.

From Fig. 2, a question may be asked as to which turning path (L or R) can be chosen from SBALA algorithm. As the variant length on the second arc is limited for a small range operation, this problem can be discussed for finding the shortest path for the first arc toward point D, then determining the second turning by R or L maneuver, see Fig. 3. The shortest turning rules for the first arc with arbitrary initial

orientations of the robot are summarized in Table I by comparing the slopes of the straight line $\overline{BD}$ and at the starting point B.



(a) LSR on right side of $\overline{AC}$. (b) RSR on right side of $\overline{AC}$.

(c) RSL on left side of $\overline{AC}$. (d) LSL on left side of $\overline{AC}$.

Fig. 2. Examples of SBALA algorithm.

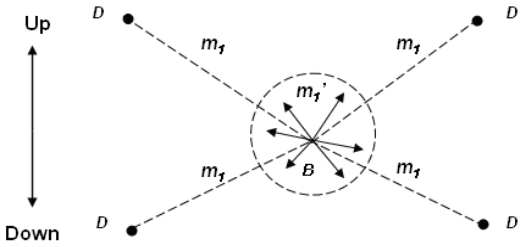


Fig. 3. Point pursuing for different locations of point D.

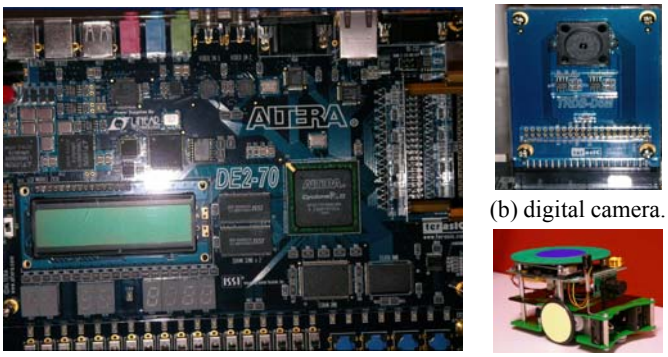
Table I
The shortest turning rules for the first arc toward point D.

m_1	m_1'	$m_1 - m_1$'	$D_{(y)} \geq B_{(y)}$		$D_{(y)} < B_{(y)}$	
			Up	Down	Up	Down
+	+	≥ 0	L	R	R	L
		< 0	R	L	L	R
−	−	> 0	L	R	R	L
		≤ 0	R	L	L	R
−	+	x	L	R	R	L
+	−	x	R	L	L	R

In Table I, m_1 is the slope of $\overline{BD}$, and m_1' is the slope at point B for an arbitrary initial orientation of the robot. Notation "Up" is for the robot is moving forward on 2D plane while "Down" is for the reverse direction. $D_{(y)}$ and $B_{(y)}$ represent the coordinates in y axis at D and B. The turning rules on Table I will reverse with different longitudinal driving directions or when point D crosses the horizontal axis at B corresponding polar angle 0 or π.

III. SBALA Implementation on FPGA

The FPGA platform used in our study is the Altera DE2-70 equipped with Cyclone II FPGA which is shown in Fig. 4(a). The camera utilized is a 5-Mega pixels digital camera module from Terasic shown in Fig. 4(b) setting shutter speed of 34fps and 1024x1280 pixels resolution. In the test scenario, the monocular digital camera with a FPGA platform is used as the environment mounted global camera to track and control two differential-drive robots of Eyebot type shown in Fig. 4(c).



(a) FPGA platform with Cyclone II FPGA.

(b) digital camera.

(c) Eyebot with bull eye label.

Fig. 4. Devices for SBALA implementation.

For FPGA implementation of the proposed SBALA algorithm, we have utilized our previous designs for machine vision. The first design is the color discrimination function [16,17]. Instead of using computationally demanding pattern recognition algorithms, the FPGA detects the dual color of bull eye labels on the top of Eyebots. The outer green ring of the bull eye is used for interference reduction, and the inner blue area is the interested label. The FPGA tracks Eyebots (labels) which start off from a docking area on the bottom of the monitor screen via specified threshold of blue pixel numbers. The second design is the relative distance estimation algorithm based on Perspective Projection Image Ratio (PPIR) [18]. Instead of performing relative distance measurement using the sensors installed on robots, we use a monocular global camera to estimate the relative distance between perspective labels' images on the robots. Further, it provides the ratio between the real distance and the diameter of labels (circles), i.e. distance in ratio equal real distance over circle diameter. Once the color discrimination and PPIR

relative distance estimation functions have been implemented on the FPGA platform, the coordinates of the leader robot, follower robot, and pursued point can be determined.

One of the SBALA applications is the multi-robot formation control. Figure 5 depicts the test scenario requiring the robot located at bottom of the figure (point *B*) to pursue and align with the virtual leader robot on the top of the figure (point *A*). Firstly, we deliberately define the central point of a leader's label at $L(x_0,y_0) = L(40,40)$ as the global coordinates in PPIR distance ratio. This value is decided by the ratio of the maximum view range of the camera over the blue circular label diameter. The instant trajectory tangent (slope) *m* is measured by movement of 1/10 label length. Once the pursuance has started, the Eyebots move forward shortly then stop, and the slopes of trajectories on 2D plane are accordingly recorded by FPGA.

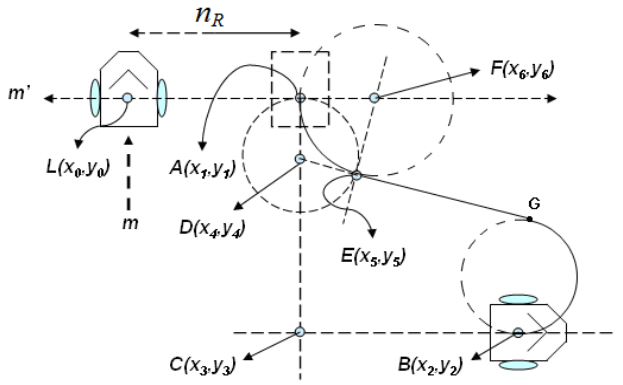


Fig. 5. Test scenario for the SBALA algorithm

In the next step, the follower robot starts to pursue the expected point with a constant speed and driving distance (curve length) in every remote instruction received, the FPGA continuously sends driving angles in radians to Eyebot for turning. The pursued point $A(x_1,y_1)$ can be derived by FPGA as:

$$m' = \frac{-1}{m} = \frac{(y_1 - y_0)}{(x_1 - x_0)}, \tag{1}$$

where its coordinates are obtained from

$$(x_1 - x_0)^2 + (y_1 - y_0)^2 = (n_R)^2 \quad , \tag{2}$$

as

$$x_1 = x_0 \pm \frac{n_R}{\sqrt{1 + (m')^2}} = x_0 \pm X,$$
$$y_1 = y_0 \pm \frac{n_R}{\sqrt{1 + \left(\frac{1}{m'}\right)^2}} = y_0 \pm Y, \tag{3}$$

in which the sign of *X* and *Y* are respectively decided by the location to leader and slope *m*, n_R is the PPIR distance ratio.

The location of follower robot $B(x_2, y_2)$ is decided by the PPIR relative ratio to the global point *L*. Coordinates $C(x_3,y_3)$ in Fig. 5 are derived from slopes *m* and *m'* as:

$$x_3 = \frac{y_2 - y_1 - m'x_2 + mx_1}{m - m'}$$
$$y_3 = y_1 - mx_1 + mx_3. \tag{4}$$

Similar to equation (3), Point $D(x_4,y_4)$ is determined according to the SLABA algorithm from coordinates $C(x_3,y_3)$ minus 3/4 $\overline{AC}$. Moreover, the follower trajectory slope and line $\overline{BD}$ can be simplified by a look-up table $\tan\theta$ in FPGA for angles from 0 to 90 degrees. Due to the control delay, if the flollower trajectory tangent is matched with the $\overline{BD}$ slope, point *G* is reached, then it will start the segment of straight driving (S). As soon as the follower reaches the circle around point *D* at distance of about $1/4AC$, point *E* is determined by $ED = 1/4AC$. The center, point $F(x_6,y_6)$, can be derived via coordinates of *A*, *D*, and *E* and slopes m_2 and *m'* as:

$$m_2 = \frac{(y_5 - y_4)}{(x_5 - x_4)} = \frac{(x_6 - x_5)}{(y_5 - y_6)}, \tag{5}$$

as

$$x_6 = x_5 + m_2 y_5 - m_2 y_6,$$
$$y_6 = \frac{y_1 - m'x_1 + m'x_5 + m'm_2 y_5}{1 + m'm_2}. \tag{6}$$

The robot is then driving along the second arc (R) with radius $AF = EF$. However, in the real applications, the length *AC* might be fixed, so the radii of the second arc can be also simplified by a look-up table which is corresponding with the look-up table of $\overline{BD}$ slope. The real driving SBALA strategy is implemented on the FPGA platform with the following pseudo codes:

```
If |slope(B) - slope(BD)| > tolerance then arc_1
  else
        reach point G and load straight line drive "S"

If |length(BD) – length(AC)| <= tolerance
  reach point E and load arc_2

If arc_2
  If AB = shortest distance then stop or reload arc_1
```

All the other functions including image processing, relative distance estimation, target tracking, remote control, and VGA interface for external image buffer are implemented on a FPGA chip using pure hardware circuit designs. The total FPGA resource usage is 41% of 68,416 LEs. The parallel processing structure guarantees the PPIR and points of SBALA both to be performed in 70 clocks synchronizing with every two pixels at the final image line "1023" (39MHz).

IV. Experimental Results and Discussion

The real-time driving images from the digital camera are shown on a computer monitor through the VGA interface of the FPGA platform. Figure 6(a) shows the initial deployment of SBALA. A rectangle mark on the floor with 24 x 48 cm^2 is used to present the coordinates. The central point of leader is initially located on the left-up side at 42 cm point. Every section of $\overline{AC}$ is 12 cm. In Figs. 6(b) and (c), the leader and follower robot move forward for one blue label length (6cm) firstly, then the follower starts the first arc turning in Figs. (d) and (e) with fixed radius 12 cm. Through the Figs. (f) to (h), the follower performs the straight and second arc driving without interruption then reach destination on the right-up corner.

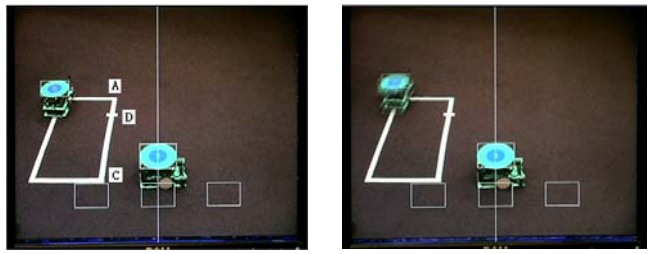
The resolution of the look-up table for $\overline{BD}$ slope and the second turning is chosen as 5 degrees. The remote medium of Eyebots is infrared with 0.3 to 0.5 seconds reaction delay. The driving speed of Eyebot is set at 10 cm per second, and the PPIR distance estimation error is less 5%. With a detected movement of 1/10 label, the deviation between robot's orientation and slope is about 5 degrees. The error becomes critical when the separating distance is increased. Thus by setting the trajectory to D with error ± 5 degrees, the maximum deviation at destination are about 10 degrees and 1/4 label shift if the robot begins the first turning at six labels away from point C.

Notably, since the proposed SBALA algorithm is developed based on Dubins' turning rules with "L", "S", and "R" maneuvers, in order to suit the requirements of FPGA implementation, the start and terminal points of the tangent between arcs are determined by using simply basic arithmetic operations. Another interesting feature is that SBALA compromises the absolutely shortest tracking path of the second arc in calculating with slopes. Also, it makes the final turning less abrupt. Such design has the advantage for easing off the error from sliding motion during the final turning process. In our experiments, the SBALA operations can cope with large incident angles. The unstable driving swing is mitigated since the second arc is aimed to align with the path to keep steering small when the robot is fluctuating a little bit from the path. Contrarily, the swing conditions in control techniques may require a small incident angle to avoid drastic error correction from over steering control.

Finally, the requirements of real-time and low power consumption on embedded systems are also considered in the proposed SBALA algorithm. Although both issues have been overcome via hardware circuit designs, unfortunately, the floating point operations with trigonometric functions consume too many logic gates, so it is infeasible to implement the whole system directly in hardware circuitry with a programmable embedded system [19]. Here, the SBALA is designed suitably to interpret robots' trajectory tangent in 2D slope with the purpose to demonstrate the possibility of realizing the steering control via basic binary operations. For example, by incorporating the PPIR algorithm [18], a smaller space unit can be represented in ratios with respect to the label diameter. With a reasonable error, it accepts numbers which are simply re-scaled by multiplying for 100, and the square rooting problem can be also easily approximated by using the following algorithm [20]:

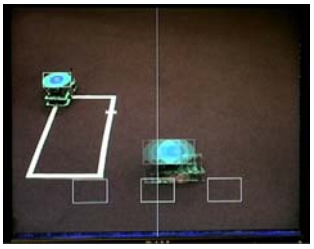
$$\sqrt{x^2+y^2} \cong \max\left(\left(\left(a-0.125a\right)+0.5b\right),a\right), \tag{7}$$

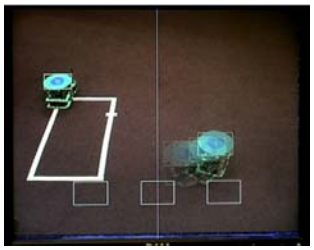
where a = max (|x| , |y|), and b = min (|x| , |y|), with values 0.5 and 0.125 being expressed by right shift operations for 1 and 3 bits.

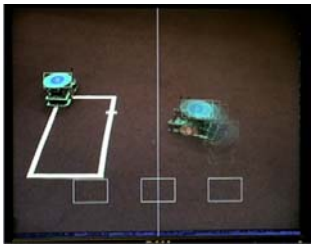


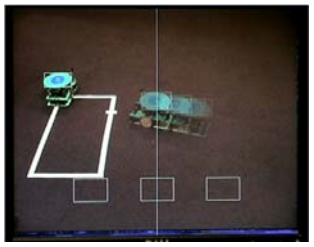
(a) Initial deployment

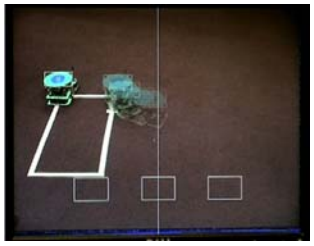
(b) Leader move forward.


(c) Follower move forward.


(d) Running on first arc.


(e) Keeping first arc.


(f) Running on straight.


(g) Running on second arc.

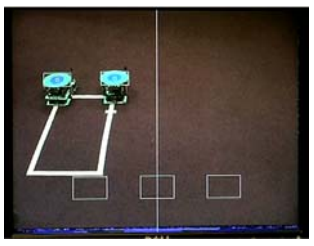

(h) Reaching destination.

Fig. 6. Point pursuing for row formation with two Eyebots.

V. CONCLUSION

We have presented an effective algorithm for robot steering with its implementation on a programmable chip. By improving Dubins' turning maneuvers, we have solved the steering problem for robotic path tracking with minimum design effort. This innovative feature can contribute to the required real-time implementation with low power computation in embedded systems. According to the experimental results, the proposed slope-based arc-line-arc algorithm demonstrates its comparable abilities in stable steering and path tracking. By breaking away from the concept of trigonometric modeling, the algorithm can model the steering maneuvers by trajectory tangents in 2D. Consequently, the operations of floating point become an option in such algorithm. These advantages make the proposed SBALA algorithm feasible and efficient for implementation on embedded systems, particularly for the application of the FPGA technology in ubiquitous robotics.

REFERENCES

[1] T. Bräunl. *Embedded Robotics*. Springer, Australia, 2006, pp. 97-121.

[2] B. Li and C. Zhang," Adaptive fuzzy control for mobile robot obstacle avoidance based on virtual line path tracking," *Proc. IEEE Intl. Conf. Robotics and Biomimetics*, Kunming, China, 2006, pp.1454-1458.

[3] R.N. Jazar, *Vehicle Dynamics: Theory and Application*. Springer, USA, 2008, pp. 98–145.

[4] R. Marion, S. Scalzi, G. Orlando, and M. Netto, "A Nested PID Steering Control for Lane Keeping in Vision Based Autonomous Vehicles," *Proc. American Control Conf.*, Missouri, USA, 2009, PP. 2885-2890.

[5] Y. Hayakawa, R. White, T. Kimura, and G. Naitou," Driver Oriented Path Following in ITS," *Proc. IEEE Intl. Conf. Advanced Intelligent Mechatronics*, Illinois, USA, 2003, vol. 1, pp.558-563.

[6] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson, "Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints," *IEEE Trans. on Robotics*, Vol. 23, pp. 47–59, 2007.

[7] F. Lamiraux and J.-P. Laumond, "Smooth Motion Planning for Car-Like Vehicles," *IEEE Trans. on Robotics*, vol. 17, pp. 498-502, 2001.

[8] S.-B. Wu, H-Y. Chen, and M-Q. Zheng, "Study on Stability for Steering Closed-Loop System of Remote-Operated Tracked Vehicle," *Proc. IEEE Intl. Conf. on Machine Learning and Cybernetics*, Montreal, Canada, vol.6, 2009, pp.3145-3149.

[9] Q.P. Ha and G. Dissanayake, "Robust Formation of Multiple Robots using Reactive Variable Structure Systems," *Int. Trans. Systems Science and Applications*, Vol. 1, No. 2, pp. 183-192, 2006.

[10] J. Morales, J-L. Martínez, A. Martínez, and A. Mandow," Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile," *EURASIP Journal on Advances in Signal Processing*, Article ID 935237, 10 pages, vol. 2009.

[11] Y. Suzuki, S. Kagami, and J. J. Kuffner, "Path Planning with Steering Sets for Car-Like Robots and Finding an Effective Set," *Proc. IEEE Intl. Conf. on Robotic and Biomimetics*, Kunming, China, 2006, pp.1221-1226.

[12] L.E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, Vol. 79, pp. 497-516, 1957.

[13] A. Balluchi, A. Bicchi, A. Balestrino, and G. Casalino, "Path Tracking Control for Dubin's Cars," *Proc. IEEE Intl. Conf. on Robotic and Automation*, Minnesota, USA, 1996, Vol. 4, pp. 3123-3128.

[14] Ying-Hao Yu, S. Kodagoda, and Q. P. Ha, "FPGA-Based Ubiquitous Computing Intelligence for Robotic Formation Control," *Proc. Int. Symp. Automation and Robotics in Construction*, Bratislava, Slovakia, 25-27 June 2010.

[15] J.A. Reeds and L.A. Shepp, "Optimal paths for car that go both forwards and backwards," *Pacific Journal of Mathematics*, Vol. 145, pp. 367-393, 1990.

[16] Ying-Hao Yu, N.M. Kwok, and Q.P. Ha, "FPGA-based Real-time Color Discrimination Design for Ubiquitous Robots" *Proc. Australasian Conference on Robotics and Automation*, Sydney, Australia, 2009.

[17] Ying-Hao Yu, N. M. Kwok, and Q. P. Ha, "Chip-based Design for Real-time Moving Object Detection using Digital Camera Module," *Proc. of the 2nd International Congress on Image and Signal Processing* (CISP'09), Tianjin, China, 17-19 October 2009, Vol. 4, pp. 1993-1997.

[18] Ying-Hao Yu, Chau Vo-Ky, S. Kodagoda, and Q.P. Ha, "FPGA-Based Relative Distance Estimation for Indoor Robot Control Using Monocular Digital Camera", *Journal of Advanced Computational Intelligence & Intelligent Informatics*, Vol. 14, No. 6, 2010.

[19] J. Detrey and F. Dinechin, "Floating-Point Trigonometric Functions for FPGAs," *Proc. IEEE Intl. Conf. on Field Programmable Logic and Application*, Amsterdam, The Netherlands, 2007, pp. 29-34.

[20] P. P. Chu. *RTL Hardware Design Using VHDL*. John Wiley & Sons, Canada, 2006, pp. 460-468.