Grzegorz MATCZAK[1], Przemysław MAZUREK[1]

West–Pomeranian University of Technology, Szczecin,
Department of Signal Processing and Multimedia Engineering,
26. Kwietnia 10 St., 71126 Szczecin, Poland (1)

doi:10.15199/48.2017.01.17

Line Following Robot with Real–Time Viterbi Track–Before–Detect Algorithm

Abstract. *Line following robots are applied in numerous application areas. High reliability of the line estimation could be obtained by the application of Track–Before–Detect algorithms, like Viterbi algorithm. The hardware and software of the robot are shown in the paper. Real–time constraint are discussed in this paper, related to the constructed robot. The obtained results shows the possibilities of tracking the single line using Raspberry Pi v.2 and Linux operating system.*

Streszczenie. *Roboty śledzące linie znajdują zastoswania w wielu miejscach. Dużą niezawodność estymacji można osiągnąć stosując algorytmy TBD w tym Algorytm Viterbiego. W pracy pokazana część sprzętową i programową robota. Ograniczenia czasu rzeczywistego są poruszane w odniesieniu do robota. Pokazano, że można śledzić linię dzięki układowi Raspberry Pi v.2 i systemowi operacyjnemu Linux.* ***(Robot Line following z algorytmem czasu rzeczywistego TBD)***

Keywords: Line following robot, Line estimation, Image processing, Viterbi Algorithm
Słowa kluczowe: Robot śledzący linie, estymacja linii, przetwarzanie obrazów, Algortytm Viterbiego

Introduction

Line following robots are applied in numerous applications [12, 5], because they are simple and reliable. Simplest robots use two light sensors and the line has high contrasts (white on dark floor or black on bright floor). The first line following robots was Stanford Cart [14] with camera for the line following using image processing algorithm. Line following robots are applied in industry [4], for components delivery, also. They are also an entertainment robots [2], and there are numerous robot contests. The line could be established intentionally or not. Road line departure warning systems, as well as vehicle control systems for autonomous work, are special class of the line following robots [17, 5]. The line could be deteriorated and disturbed by the light and weather conditions and should be tracked for the control of vehicle.

Robots are very important with the control system based on the existing lines, not created by the human. Harvesting robots [13, 1] could use line estimated from the positions of trees or another species.

The importance of the line following robots is high, because GPS–based navigation systems do not work in some cases. GPS signal is not available inside buildings and tunnels, and the quality of GPS is reduced due to low visibility of satellites in city canyons.

Related Works

Tracking algorithms are applied for the improving of estimation. The detection of specific features for the control of robots is the solution for controlled environment only. Highly deteriorated lines and natural lines require high reliability algorithms. Track–Before–Detect algorithms [16, 7, 8] could be applied for the estimation of lines on single images or video. The Viterbi algorithm for the line following robots is proposed in [10]. Combined direction filtering and Viterbi algorithm is proposed in [9]. Tracklets are proposed in [6] for the reduction of computation cost. Line could be non–positive signal, and the tracking noise lines are considered in [11].

Content and Contribution of the Paper

The Viterbi algorithm [18] could be applied for the line tracking [15], but the main purpose is the decoding of convolutional codes [3]. The analysis based on Monte Carlo approach was a subject of previous papers [10, 9, 11]. The computation cost of Viterbi algorithm and performance of overall system are important for practical application of the line following robot.

The Viterbi algorithm for image analysis is briefly considered in Section 2. The hardware of robot is presented in Section 3. Software part is considered in Section 4. Obtained results are presented and discussed in Section 5. Final conclusions and the further works are described in Section 6.

Viterbi Algorithm for Line Following Applications

This algorithm uses window approach and the image is analyzed from bottom located window. After each processing step this moving window moves toward top part of image and the process is repeated until end of the image is achieved.

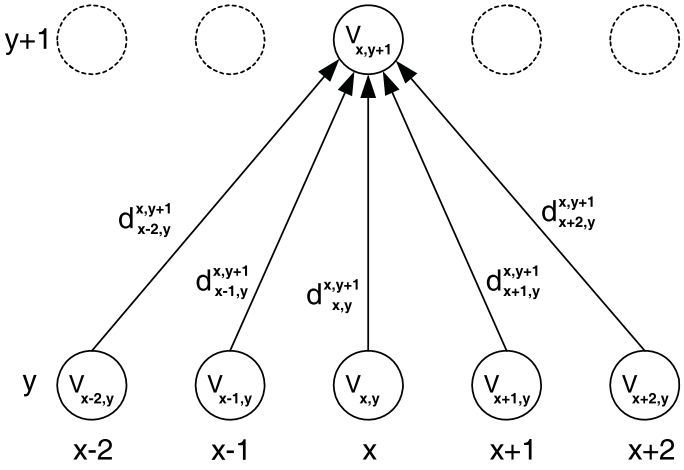

Fig. 1. Local paths of trellis example

The following steps are applied for particular window position. The image analysis starts from first row $y = 0$ of moving window and zero values are assigned to the nodes (node could be a pixel, for example):

$$V_{y=.,1} = 0. \tag{1}$$

The local paths of trellis for five transitions $g \in \{-2, -1, 0, +1, +2\}$ and for two neighborhood rows are shown in Fig. 1. The value of node $(V_{x,y+1})$ is the largest value projected from previous row to the particular node (forward phase):

$$V_{x,y+1} = max\left(V_{x+g,y} + d^{x,y+1}_{x+g,y}\right). \tag{2}$$

Local cost d could be defined as a spectrogram pixel value with optional additional weight (cost). The selection of the

best local path for the particular node is registered using the following formula:

$$L_x^{x+1,y} = \arg\max_g \left(V_{x+g,y} + d_{x+g,y}^{x,y+1} \right), \tag{3}$$

where L stores the local transition value g for further processing. The projection of values start from $y = 1$ to the $y = y_{max}$ row, where y_{max} is the depth of analysis. The solution of forward phase of Viterbi TBD is the position of node x with maximal value obtained for the last row:

$$x_{y=y_{max}} = \max_x \left(V_{x,y=y_{max}} \right). \tag{4}$$

The backward phase allows the selection of optimal solution for particular position of moving window using the following recursive formula:

$$x_{y-1} = x_y + L_{y-1}^{x,y}, \tag{5}$$

for successively decremented row numbers:

$$y = y_{max,\cdots,2}. \tag{6}$$

Obtained position is related to x_1 node. The accumulation process in forward phase formula is responsible for the TBD behavior. The performance depends on the depth of analysis, so y_{max} cannot be small value.

Robot Hardware

The mobile robot (Fig. 2) consist of solid wooden construction, four wheels, power supply and DC motors. It is testing platform for different hardware devices, with a specific constraints. Such flexibility allows the testing autonomous robot with on–board image processing and computation, as well as remote control using specific wireless link. It could be equipped with different type cameras, like RBG, IR, Depth working together or alone. Multiple cameras could be mounted depending on test demands.

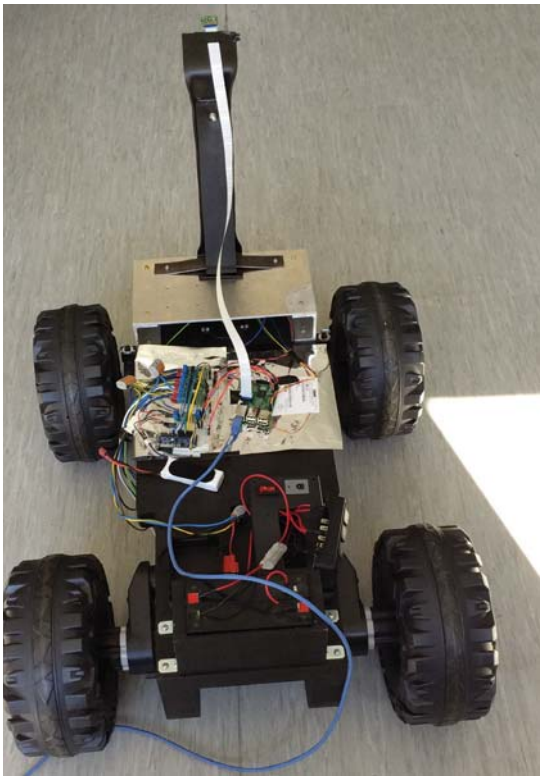

Fig. 2. Test platform – line following robot (rear view)

This is medium size mobile robot ($100 \times 80 \times 50$ cm). There are three brushed motors, powered from 12V batteries. Two motors are used for rear–wheel drive with independent control. The third motor is used for the control of driving direction using front–wheels rotation. The robot sterring is improved by the absolute magnetic encoder mounted to the axis of front wheels. Such angle sensor ensures important information for the application of feedback loop. In the line–following application additional feedback loop is obtained using vision. Solid camera grip, dedicated to PCB camera, is placed at the front of the robot.

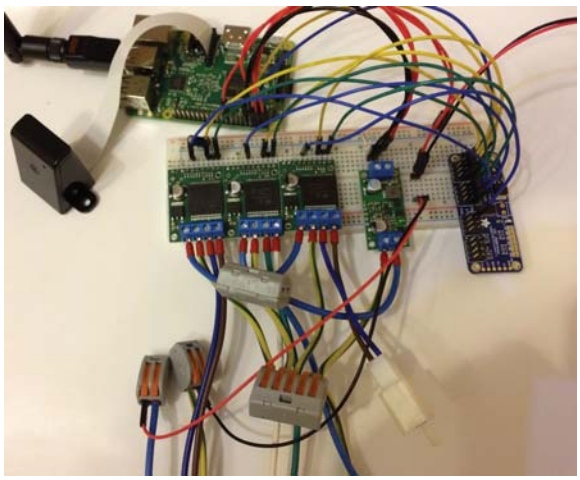

Fig. 3. Hardware used to control the Mobile Robot

The main part of control system is the Raspberry Pi v.2 embedded system board with the latest embedded Linux "Rasbian Jessie".

The dedicated camera for Raspberry Pi version HD C with OV5647 OmniVision sensor is applied in test. This camera supports VGA resolution and 90 fps. Smaller resolution (320×240) and grayscale image scheme is used typically. The camera uses fixed focus with $54^o \times 41^o$ angle view and automatic mode (e.g. brightness) during the image acquisition.

Raspberry Pi and other components of the system drives consumes about 1 A current, that is accepted level in particular system. The flexibility of the robot allows the application of embedded PC with modern GPU.

The robot default control is PC computer with WiFi using UDP (User Datagram Protocol) communication. The Control System is shown in Fig. 3. The secondary control security solution is added and activated after communication lost and lack of the external control.

Remote control of robot is possible using one of three independent communication channels. Main communication channel uses UDP and WiFi adapter or Ethernet cable. Second communication channel uses Bluetooth and software UART (Universal Asynchronous Receiver and Transmitter). The last communication channel uses radio adapter connected to UART. Different radio communication board could be used and high distance (hundreds of meters) could be achieved.

Hardware safety switch could be used in some emergency cases. Additional two analogue distance sensors using ultrasound allows the detection of obstacles.

Robot Software

The Raspberry Pi v.2 board has four processor core 700MHz each. Previous version of Raspberry Pi contains only single core, so hardware supported parallel processing

is not possible in this board.

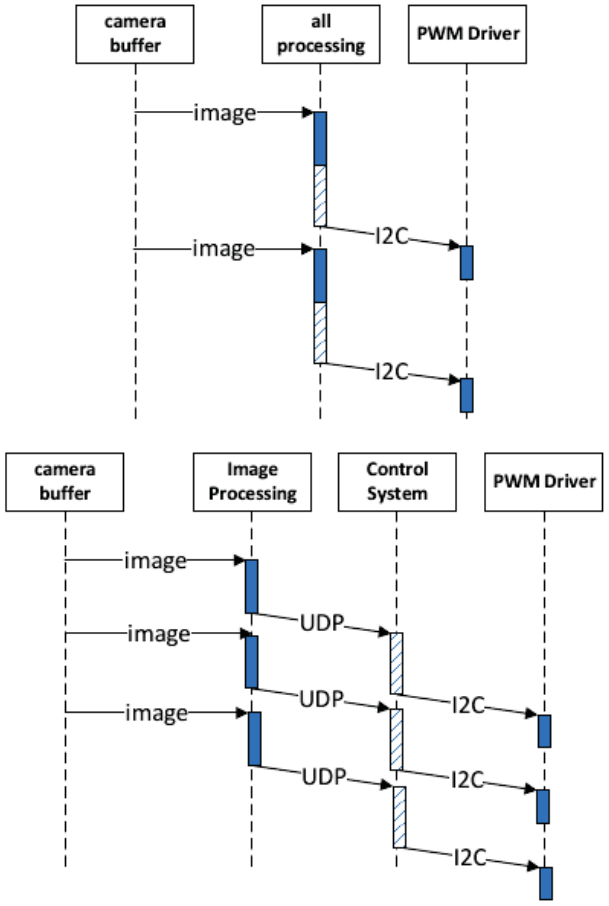

Fig. 4. Two different Main Control System Processing flows

Single core board requires sequential processing (Fig. 4) so real–time processing is harder to achieve. It is important that Linux is not real–time system and the non–critical application requires availability of computation power reserve. Four cores processor allows the parallel processing, so different tasks could be assigned to appropriate cores.

Main program uses two UDP ports for communication. One port is applied for the communication with PC and the second port for debugging purposes. Debug port could be disabled, because there is important influence of the additional communication on the system performance.

Overall scheme of the system in shown in Fig. 6.

The example of a PC Control program was proposed. Screen of this program is shown on Fig. 5. The main purpose of this program is the manual control of robot. Changes of PID (proportional-integral-derivative controller) parameters, motor parameter, etc. is possible using this application, so recompilation of code is not necessary.

Previous works were related to the testing of Viterbi algorithm using Matlab. This robot requires more time efficient code, so C–language implementation is prepared. Image processing part uses OpenCV-2.4.10 library for simple operations and visualizations if an additional monitor is connected to Raspberry Pi board for debugging purposes.

Results

The robot follows the line and estimate line very well. Low contrast line could be estimated but line should be dark for considered test, that is shown in Fig. 7 left. Dark obstacles may disturb (Fig. 7 right) estimation.

Message exchange adds some delays but non–blocking

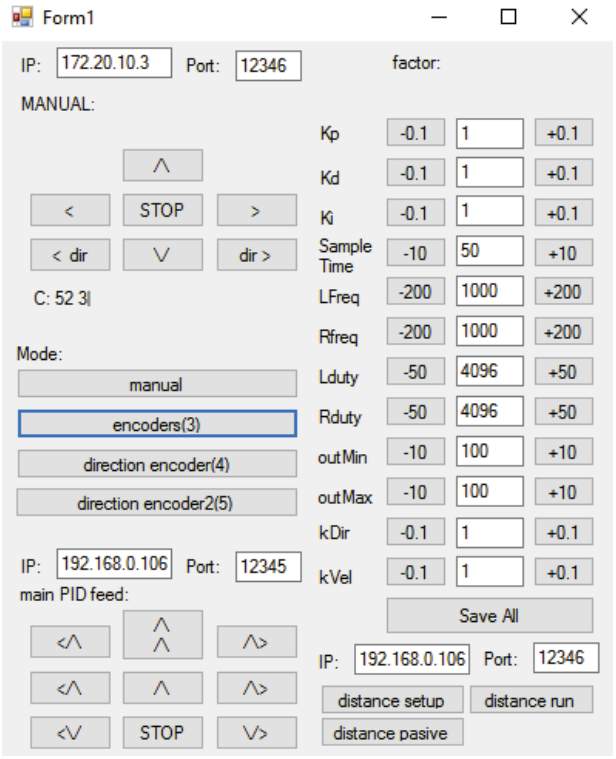

Fig. 5. The prototype of program to control the Mobile Robot manually from PC

mode is used. The processing speed could estimated using achieved frame per seconds (fps) value. Mean value for 0.5 hour period is measured.

Main Control System adapts own processing depending on the frame rate. It is possible by the special design of PID controller. Such behavior is necessary if variable frame rate occurs, but the optimization of the system is necessary. Real–time processing is achieved if the video frame processing rate is equal to the acquisition rate. The desired value is 90 fps due to camera settings. One of the most important factors that influences of fps is optimization level of gcc compiler. Lack of the optimization is obtained for -O0 flag, and aggressive optimization is achieved for -O3 flag.

The improvement especially is significant for the smaller resolutions. The 320×240 resolution is selected as an compromise between processing speed and resolution. The reduction of resolution to 160×120 is possible but the image quality is reduced, so width of tracked line should be larger.

One of the most important problem in the current software is lack of the real–time transmission of video image from camera to the external control computer, because there is lack of computation power in system.

Conclusions and the Further Work

The most important problem (real–time image processing) was solved, by the selection of proper hardware and software implementation. This robot supports image processing with variable frame rate. Fixed frame rate is achieved after the reduction of image size and by the selection of optimization. The most important is not a selection of resolution and optimization flag, but proper design of Viterbi algorithm. This algorithm requires a lot of computation, and some processor design (especially DSP) supports Viterbi processing in hardware.

The optimization of Viterbi algorithm using different software approaches will be considered in further work. The application of Viterbi algorithm gives superior performance of

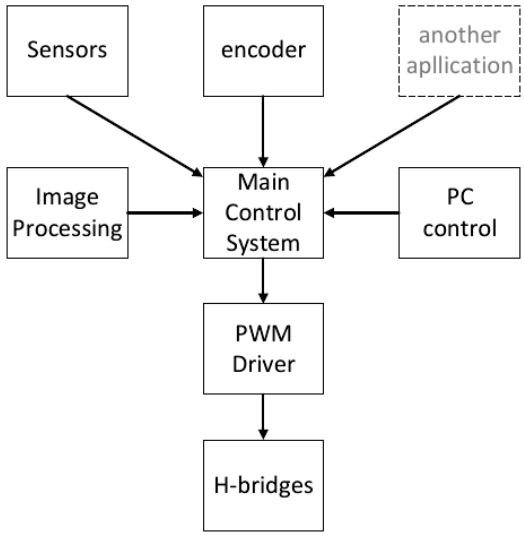

Fig. 6. The Control System scheme

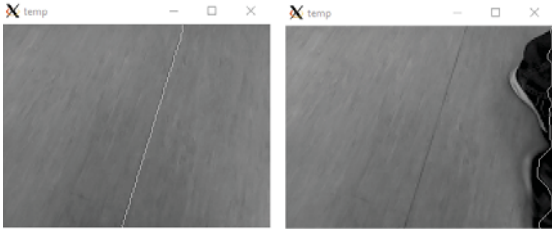

Fig. 7. Line estimation in live view

line estimation, so it is accepted for particular hardware. Low power design requires FPGA implementation.

*

Acknowledgment This work is supported by the UE EFRR ZPORR project Z/2.32/I/1.3.1/267/05 "Szczecin University of Technology – Research and Education Center of Modern Multimedia Technologies" (Poland).

REFERENCES

[1] Astrand, B. and Baerveldt, A.: A vision–based row–following system for agricultural field machinery, Mechatronics, 15 (2), pp. 251–269, 2005.
[2] Colak, I. and Yildirim, D.: Evolving a Line Following Robot to Use in Shopping Centers for Entertainment, 35'th Annual Conference of IEEE Industrial Electronics, 2009. IECON '09, 34(5), pp. 3803–3807, 2009.
[3] Haykin, S. and Moher, M.: Communication Systems, John Wiley & Sons, pp. 251–269, 2009.
[4] Horan, B. and Najdovski, Z. and Black, T. and Nahavandi, S. and Crothers, P.: OzTug Mobile Robot for Manufacturing Transportation, IEEE International Conference on Systems, Man and Cybernetics (SMC 2011), pp. 3554–3560, 2011.
[5] Ismail, A.H. and Ramli, H.R. and Ahmad, M.H. and Marhaban, M.H.: Vision-based System for Line Following Mobile Robot, 2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009), October 4-6, 2009, Kuala Lumpur, Malaysia, pp. 642–645, 2009.
[6] Grzegorz, M. and Mazurek, P.: Tracklet–Based Viterbi Track-Before-Detect Algorithm for Line Following Robots, Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015 Springer International Publishing, pp. 649–658, 2016.
[7] Mazurek, P.: Optimization of Bayesian Track-Before-Detect Algorithms for GPGPUs Implementations, Electrical Review, 86 (7), pp. 187–189, 2010.
[8] Mazurek, P.: Code reordering using local random extraction and insertion (LREI) operator for GPGPU-based track-before-detect systems, Electrical Review, 18 (6), pp. 1095–1106, 2013.
[9] Mazurek, P.: Directional Filter and the Viterbi Algorithm for Line Following Robots, Computer Vision and Graphics, Lecture Notes in Computer Science, Springer, 8671, pp. 428–435, 2014.
[10] Mazurek, P.: Line Estimation using the Viterbi Algorithm and Track–Before–Detect Approach for Line Following Mobile Robots, 19'th International Conference on Methods and Models in Automation and Robotics, pp. 788–793, 2014.
[11] Mazurek, P.: Viterbi Algorithm for Noise Line Following Robots, 19'th International Conference on Methods and Models in Automation and Robotics, Advances in Intelligent Systems and Computing, Springer, 313, pp. 111-118, 2015.
[12] Okarma, K. and Lech, P.: A fast image analysis for the line tracking robots, Artifical Intelligence and Soft Computing (ICAISC 2010), LNCS, 6114, pp. 329–336, 2010.
[13] Ollis, M.: Perception Algorithms for a Harvesting Robot, Carnegie Mellon University, CMU-RI-TR-97-43, 1997.
[14] Schmidt, R.A. Jr.: A study of the real–time control of a computer–driven vehicle, Stanford University, 1971.
[15] Scott, T. A. and Nilanjan, R.: Biomedical Image Analysis: Tracking, Morgan & Claypool, 2005.
[16] Stone, L.D. and Barlow, C.A. and Corwin, T.L.: Bayesian Multiple Target Tracking, Artech House, 1999.
[17] Taubel, G. and Yang, J.-S.: A Lane Departure Warning System Based on the Integration of the Optical Flow and Hough Transform Methods, 2013 10th IEEE International Conference on Control and Automation (ICCA) Hangzhou, China, June 12–14, 2013, pp. 1352–1357, 2013.
[18] Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory, 13 (2), pp. 260–269, 1967.

Resolution	fps	CPU	optimization flag
160×120	72	74%	-O0
320×240	22	89%	-O0
640×480	5	97%	-O0
160×120	89	35%	-O3
320×240	46	65%	-O3
640×480	15	92%	-O3

Fig. 8. Performance of system depending on image resolution and optimization flag

***Authors**: Grzegorz Matczak, West–Pomeranian University of Technology, Szczecin, Department of Signal Processing and Multimedia Engineering, 26. Kwietnia 10 St., 71126 Szczecin, Poland, email: grzegorz.matczak@zut.edu.pl, dr hab. inz. Przemyslaw Mazurek, West–Pomeranian University of Technology, Szczecin, Department of Signal Processing and Multimedia Engineering, 26. Kwietnia 10 St., 71126 Szczecin, Poland, email: przemyslaw.mazurek@zut.edu.pl*