PISALA project. Intelligent Sensorization for Line tracking with Artificial Vision

Vicent Girbés, Leopoldo Armesto, Josep Tornero
(Instituto de Diseño y Fabricación, Universidad Politécnica de Valencia, Spain)

Abstract

This paper presents an artificial vision embedded solution for detecting painted lines on the floor, in the context of the Research Project PISALA (Industrial Prototype for Automatic Line Tracking with Artificial Vision), whose main objective is to provide vision-based solutions for Automated Guided Vehicles (AGVs). In addition the paper proposes a procedure for evaluating robustness and accuracy of line-detection algorithms including qualitative and quantitative tests for detecting specific situations such as images with multiple lines or lines with spurious objects, lines with different colors, etc. The paper also considers the algorithm's sensibility with respect to extrinsic camera calibration errors, in order to define the bounds of the detection area. As a result, low cost robust algorithms for line detection can be evaluated. The examples for validation have been implemented on the embedded vision system CMUCAM3.

1 Introduction

Nowadays, there are several commercial solutions with self-guided vehicles, or AGVs (Self-Guided Vehicles), see for example [1, 2, 3] among others. These vehicles have traditionally been guided by magnets, wires or laser located in specific areas of the store. Another type of guidance is based on inertial navigation, in which accelerometers and gyroscopes, along with transponders embedded in the ground to find the position of the vehicle. The major disadvantage of traditional AGVs is its limited flexibility to adapt to the changing environment, the cost associated with the infrastructure required, sometimes can't modify its route, etc.

To increase the degree of autonomy of AGVs, researchers have developed in recent decades, many tools to solve problems that traditionally occur in mobile robotics, such as simulation [4], localization and map construction (SLAM) [5], navigation and path planning [6, 7, 8], the coordination between robots, and so on. In this regard, increasingly are required more sensors with high processing power and abstraction, such as vision systems, which are becoming increasingly important due to the continuous improvement of technology. Take for example the implementation ROBOLIFT © [9], which uses a vision system to detect artificial marks on the floor. In [10] and [11], the problem for load and unload pallets is solved using a vision system.

In [12, 13, 14], were developed several industrial applications with AGVs, where various proposals were made primarily automation, defining hardware and software architectures, oriented to teleoperation of vehicles, autonomous navigation or line tracking by artificial vision.

In [15], system in which a vehicle is capable of following a white line on the floor and a state transition algorithm that allows them to detect bifurcations through pattern recognition was developed. In [16] used a vision system with a LADAR range images to produce a military vehicle. The aim is to detect the edges of the road for the vehicle to move independently. In the field of agriculture, we find various solutions of autonomous agricultural vehicles or semi-autonomous with the ability to detect bias own lines or lines of culture, see [17] as an example. Similarly in [18] describes a robot capable of removing paint from ships following their own lines.

This article is framed within the context of AGVs, describing a procedure for validating line detection algorithms with vision systems. This procedure considers the validation of line detection on images through qualitative and quantitative analysis. Qualitative analysis is based on a set of images; while quantitative analysis provides line detection robustness and accuracy as well as parameter sensibility, defining the limits of extrinsic parameter calibration errors that can be affordable without significant detriment on the robustness of the line detection process. The line detection algorithm that has shown high robust performance and it has been specially designed for processors with limited computational and memory resources. The proposed algorithm has been implemented and validated on the CMUCAM3 [19], a low-cost embedded vision system commonly used in robotics.

The paper presents some results of the project PISALA (Spanish acronym for Industrial Prototype for Automatic Line Tracking with Artificial Vision), whose main objective is to provide vision-based solutions for Automated Guided Vehicles (AGVs). An industrial forklift, as shown in Figure 1, has been automated for autonomous applications. In particular, vision-based tracking of painted lines on the floor has been implemented (see Figure 2) but also fully autonomous navigation without artificial landmarks, intelligent manual-assisted driving in which the drivers co-exists with and intelligent system designed to reduce the risk of collisions and vehicle remote teleoperation. Section 2 describes the algorithm for detecting lines on images. Section 3 provides a qualitative validation of the line detection algorithm, while Section 4 provides quantitative data to analysis of

robustness and accuracy of the proposed algorithm and parameter sensibility to calibration errors. Finally, Section 5 draws conclusions and discusses future work.

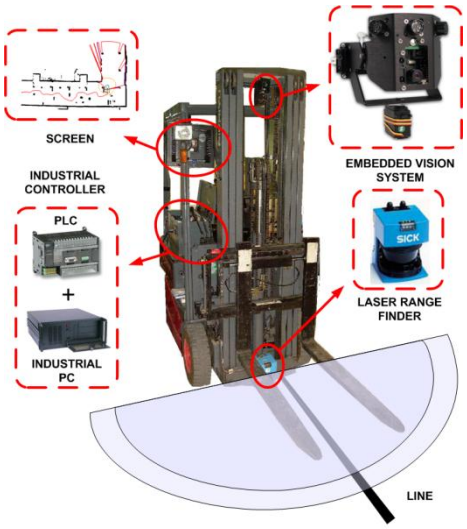


Figure 1 Automated industrial forklift for line-tracking applications.

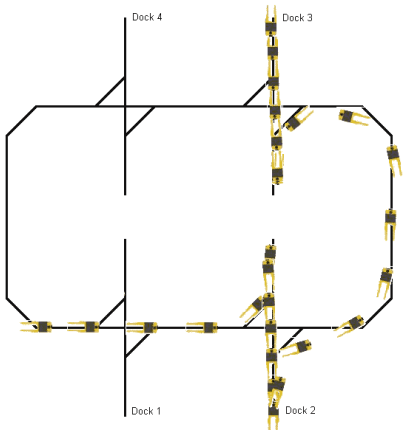


Figure 2 Vision-based line tracking application.

2 Robust Line Detection on Images

This section describes an algorithm for detecting lines with their parameters. The algorithm has been specifically designed to run on processors with low computational and memory resources like the CMUCAM3 embedded vision system. Therefore, the algorithm must be as simple as possible while robustly detect lines on images. The input data is a grayscale image of 176x144 pixels, which corresponds to low-resolution mode with YCrCb format (we only use the channel Y).

The following assumptions have been taken into account:

- The orientation of the line is usually vertical, and therefore the detection algorithm will work better the more vertical the line in the image is. Although, in practice, the algorithm can detect lines close to horizontal axis, failures on such as cases might be affordable.
- The contrast on the line and the image background is not necessary high. Usually the line is assumed to be black colored, while no specific background color is assumed. For that purpose, our algorithm uses a background compensation process.
- The background of the image can contain soil (for example an oil slick) or objects that might disrupt the perception of the line.
- The image can contain several lines and only the most significant line from the left and right sides are considered.

Algorithm 1 shows the pseudocode and a detailed description of algorithms steps. The first step of the algorithm corresponds to a simple background compensation step that uses an operation that computes for each pixel the maximum and minimum of its neighbors with 4-connectivity. In this sense, the background compensation performs first a minimum operation followed with a maximum operation with the neighbor pixels at N=1 pixel distance to the central pixel to filter out image noise. Then, repeats the maximum operation several times followed with the minimum operation, also repeated several times, with neighbor pixels at N=10 pixel distance to the central pixel. Minimum and maximum operations are defined as follows:

$$I_{x,y} = \min\{I_{x,y}, I_{x-N,y}, I_{x,y-N}, I_{x+N,y}, I_{x,y+N}\}$$
$$I_{x,y} = \max\{I_{x,y}, I_{x-N,y}, I_{x,y-N}, I_{x+N,y}, I_{x,y+N}\}$$

where $I_{x,y}$ is the pixel intensity value at x and y coordinates.

The purpose is to obtain a uniform image with the background color that will be subtracted from the original image. Figure 3 shows two image examples and the image background compensated.

The next algorithm step, in line 2, performs a edge detection algorithm with potential line points. For this purpose, we search for changes in contrast per row. Candidate points require rising and falling transitions of the intensity values, but also the expected thickness of the line on the image plane taken into account the line perspective, so that spurious point pairs that do not match this criterion can be easily filtered out.

Our edge detection approach is a simplistic solution to avoid standard edge detection algorithms that will require more computational time and resources. Since this procedure is simple and prone to detect a large number of spurious edges, for that purpose lines from 5 to 12 of the algorithm represent the implementation of the RANSAC algorithm [20], [21], [22]. As a result of the algorithm provides a subset of points that can be robustly fitted to a line model defined by the well-known angle-distance model $\rho = x\cos\phi + y\sin\phi$. Figure 4 shows parameter definitions, where (x_i,y_i) *and* (x_f,y_f) are the Cartesian coordinates with respect to the reference frame of the image of the starting and ending points, respectively; ρ_i distance to the reference point the camera initial distance from the system ρ_f reference chamber end; Φ_i is the angle about the x axis of the starting point; Φ_f is the angle about the x axis of the end point, and ρ and Φ are the line model parameters.

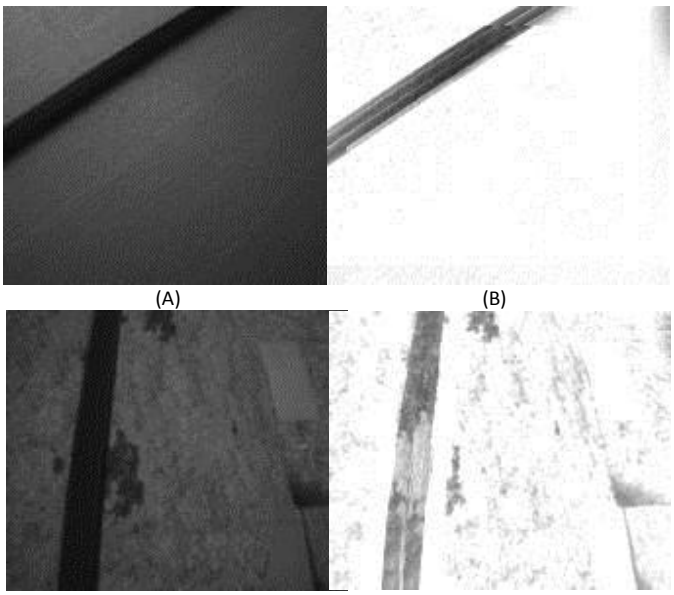


Figure 3 Examples of background compensation. (A) and (C) original images and (B) and (D) are the corresponding background compensated image.

1. **Compensates the background of the image from consecutive operations of maximum and minimum with neighboring pixels.**
2. **For each row, determine the edges of the line based on the difference of intensity of neighboring pixels (search on the left and right of image).**
3. **Repeat the process steps 3 through 12 separately for the edges of the left and right edges.**
4. **Initialize "counter" to zero and "N" (the number of iterations to be executed) to 1.**
5. **While (N> counter).**
6. **Randomly select two rows.**
7. **Calculate the line parameters "rho" and "phi" passing through selected points.**
8. **For the rest of the points, calculate the distance orthogonal to the line calculated.**
9. **Sort, Inliers points and those whose distance is smaller than a certain threshold.**
10. **Inliers If the number exceeds the number of inliers found so far, then save the current solution and calculate the remaining number of iterations N = log (p) / log (1-portion of inliers2), where p = 0.01 the expected probability to have an undetected spurious.**
11. **Increase "counter" on a drive and if they have reached the maximum number of iterations allowed, exit with failure.**
12. **End Repeat.**
13. **Calculate the line parameters only with Orthogonal Regression Inliers points.**
14. **Calculate the intermediate line.**

Algorithm 1. Algorithm used to obtain the characteristic parameters that define the line detected.

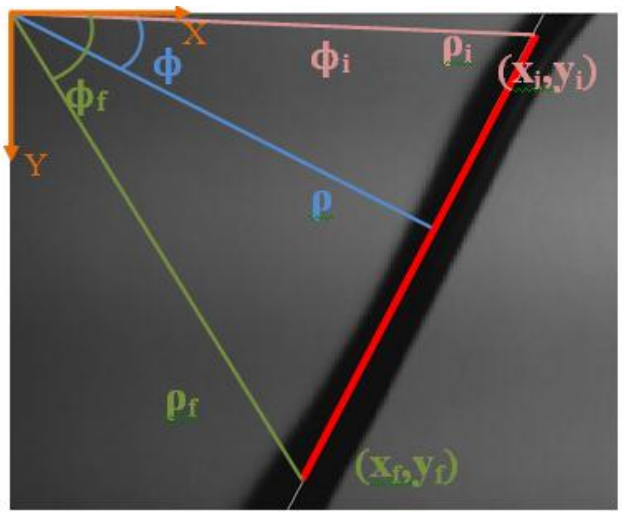


Figure 4 Representation of the line in the distance-angle space

Line parameter estimation, in line 13, is straight forward [23]:

$$\hat{\rho} = \bar{x} \cdot cos\varphi + \bar{y} \cdot sin\varphi, \hat{\varphi} = \frac{1}{2} \cdot arctan\left(\frac{-2 \cdot \sigma_{xy}^2}{\sigma_y^2 - \sigma_x^2}\right) \quad (1)$$

where, $\bar{x}\ and\ \bar{y}$ are mean values of points and $\sigma_{xy}^2, \sigma_y^2\ and\ \sigma_x^2$ are point covariances.

Line extrema are computed based on the first and last inliers points projected onto the line model:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} \rho \\ \rho \cdot \tan(\phi_i - \phi) \end{bmatrix},$$

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} \rho \\ \rho \cdot \tan(\phi_f - \phi) \end{bmatrix} \quad (2)$$

3 Line Detection Qualitative Validation

The algorithm has been tested with different lighthing conditions and line colours. Figure 5 shows a set image examples used to qualitatively validate the line detection algorithm. In the first two columns we can see a set of images with different line shapes, orientations and colours. In particular, the first column depicts detected edge points, while in the second column only inliers points and detected lines are shown. In third and fourth columns we can see a set of images with lines together with some objects in order to validate the spurious filtering capabilities of RANSAC. It can be appreciated from selected images that the algorithm performs successfully even with images with is low contrast between the line and the background. The algorithm provides as estimation the line with the highest number of points rejecting spurious data.

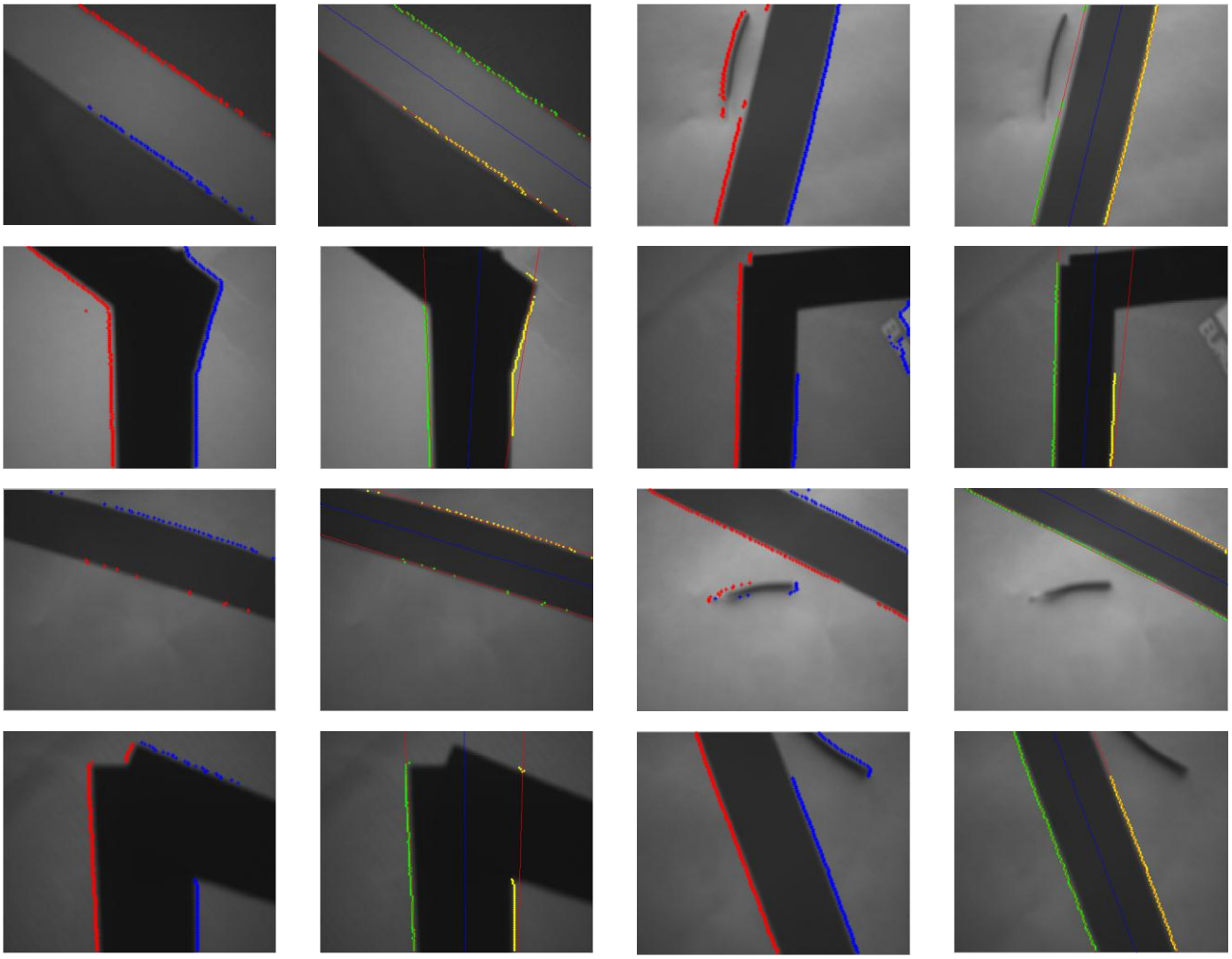


Figure 5 Qualitative analysis of line detection algorithm. Images on even columns show the results of the edge detection process while images on even columns show the filtered points and the estimated lines.

4 Line Detection Quantitative Validation

In order to provide quantitative data a robot set-up is used to validate the algorithm. The set-up includes a KR/15 KUKA industrial robot arm holding the camera, as shown in Figure 6, together with a turntable for moving lines. This allows us to generate combinations of camera positions and line rotations with ground truth data. The robot moves perpendicular to the line (Rx) with increments of 30mm and the turn table rotates (ψ) with increments of 10º.

Based on these experiments, we have obtained statistic results of the robustness and accuracy of the algorithm. It is possible to map sensibility parameters with respect to extrinsic calibration errors of the camera.

4.1 Robustness and Accuracy Analysis

The goal is to analyze the **robustness** and **accuracy** of the algorithm for different movement combinations, which have been classified into small motions ($\psi \in$ [-10°, 10°] and $Rx \in$ [-30 mm, 30 mm]), medium motions ($\psi \in$ [-40°, 40°] and $Rx \in$ [-90 mm, 90 mm]) and large motions ($\psi \in$ [-70°, 70°] and $Rx \in$ [-150 mm, 150 mm]).

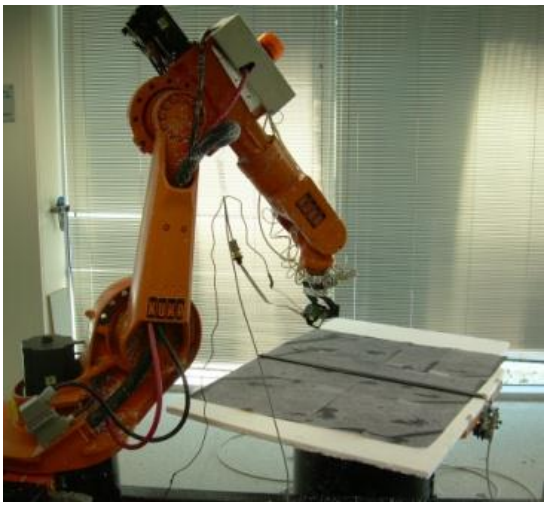


Figure 6 Set-up consisting of a Vision System CMUCAM3 together with KR/15 KUKA robot arm and a turntable for "ground-truth" data.

Therefore, 9 experiments are classified as small rotations and translations, 63 experiments as medium and 165 experiments (all of them) are included in the large set. The experimentation has been done with two different line boards: one board contains a line with a white background (see Figure 3 (A)) while the other the background is grey stained, to represent a more realistic situation (see Figure 3 (B)).

Based on the estimated line parameters, the error with respect to ground truth data is computed. We classify each experiment as true positive (TP) if the estimation error in distance is $\epsilon_\rho \leq 50mm$ and in angle $\epsilon_\phi \leq 6°$. On the other hand, a false positive case (FP) satisfies $\epsilon_\rho > 50mm$ or $\epsilon_\phi > 6°$. Finally, negative results (N) are those cases where the algorithm was not able to detect a line due to lack of detected edge points. Based on this classification, we evaluate the robustness of our algorithm with respect to the type of experiment (small, medium or large) and the type of background (white or gray stained). In addition to this, we also evaluate the accuracy of TP cases based on the following criteria: estimation with high accuracy requires $\epsilon_\rho \leq 50mm$ and $\epsilon_\phi \leq 6°$, medium accuracy results implies $50mm < \epsilon_\rho \leq 100mm$ and $6° < \epsilon_\phi \leq 12°$, and otherwise the accuracy is low.

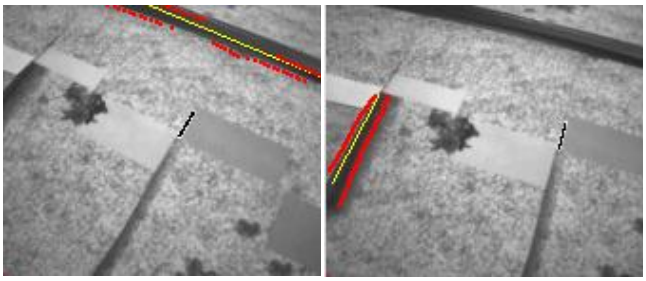
Table 1 shows robustness and accuracy results obtained for detecting lines with white background while Table 2 shows robustness and accuracy results for the case with the gray stained background.

WHITE BACKGROUND							
# Images	Rotation and translation	ROBUSTNESS			ACCURACY		
		TP	FP	N	H	M	L
9	Small	100	0	0	100	0	0
63	Medium	100	0	0	74.6	25.4	0
165	Large	100	0	0	73.33	24.85	1.82

Table 1 Robustness and Accuracy of Line Detection Algorithm with white background.

GREY STAINED BACKGROUND							
# Images	Rotation and translation	ROBUSTNESS			ACCURACY		
		TP	FP	N	H	M	L
9	Small	100	0	0	88.88	0	11.11
63	Medium	98.41	1.59	0	58.06	38.71	3.23
165	Large	91.52	8.48	0	45.63	48.12	6.25

Table 2 Robustness and Accuracy of Line Detection Algorithm with grey stained background.



(A) (B)

Figure 7. TP and FP selected cases for discussion.

It can be seen that all experiments with white background perform successfully and has been classified as TP. For the gray stained case, the percentage of TP case is very high over 91.5% of experiments (14 over 165). Figure 7(B) shows one of the FP cases, where it can be appreciated that due to line orientation, very few points were considered as edges and as a consequence an spurious line is detected instead. Despite of that, figure 7(A) shows a TP case with similar complexity.

4.2 Parameter Sensibility to Extrinsic Calibration Errors

Using the previously described set-up, artificial motions on the robot wirst have been generated to reproduce angle extrinsic calibration errors on yaw and pitch angles, see Figure 8 for the definition of these angles of the camera mounted on a PTU. The goal is to evaluate how these calibration errors affect to the estimation and in particular to robustness and accuracy.

In particular, we are generating a sweep of yaw (α) and pitch (β) angles, with the following set of values, α={-10º,-5º,0º,5º,10º} and β=[25º,30º,35º,40º,45º}. For each angle combination, we are reproducing all the 165 experiments for the grey stained background case.

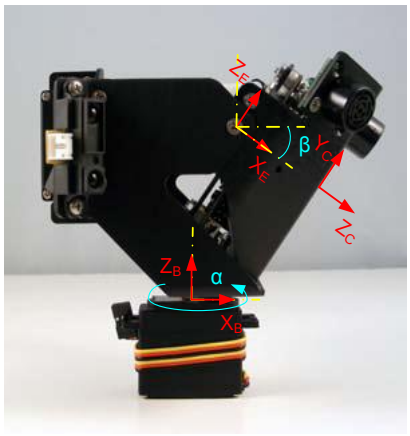


Figure 8. Coordinate systems of the camera turret.

Figure 9 shows the robustness sensibility map to calibration errors, where the color bar values represent percentage of TP cases. As expected, robustness is affected simetrically with respect to the yaw angle with a ±4º bounds without detriment. On the other hand, pitch angle calibration errors seriously affect to robustness, specially for cases where the camera points to horizont due to image perspective.

Moreover, Figure 10 shows the results of accuracy sensibility map to calibration errors, where the color bar represents percentage of high accuracy cases. In this case, the accuracy is affected simetrically by both angles, altough the pitch angle has more influence, obtaining poor accuracy results if this angles is not properly estimated.

Therefore, appropiate calibration of pitch angle is crucial for this kind of applications, since robustness and accuracy are seriously affected by camera perspective.

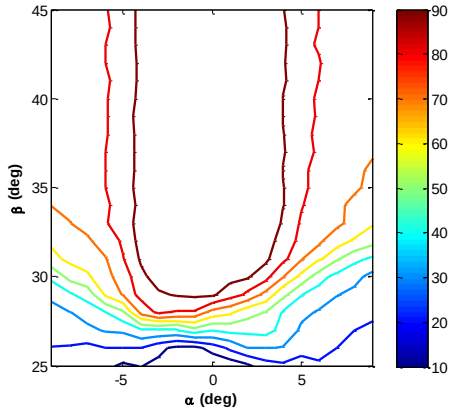


Figure 9 Robustness sensitivity to calibration errors, where α is yaw angle and β is pitch angle.

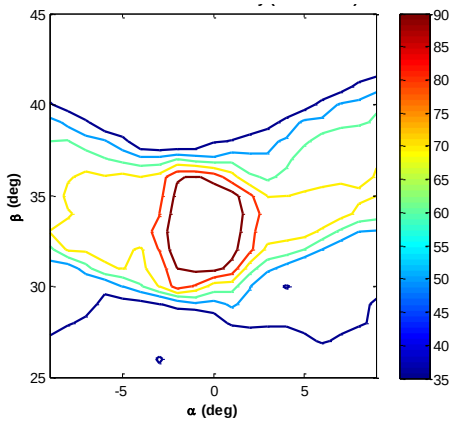


Figure 10 Accuracy sensitivity to calibration errors, where α is yaw angle and β is pitch angle.

5 Conclusions and Future Work

The paper has presented a robust line detection algorithm with application to line tracking with AGVs. The algorithm has been validated using both, qualitative and quantitative results: as qualitative evaluation, we have provided several examples of line detection under different conditions (colors, shapes, etc.); as quantitative evaluation, we have provided robustness and accuracy tables showing very high performance. Therefore the paper has described a procedure for evaluating robustness and accuracy of line-detection algorithms including qualitative and quantitative tests.

In addition to this, the study on parameter sensibility bring us to the conclusion that calibration errors on pitch angle affect to image perspective with higher detriment on robustness and accuracy than errors in yaw angle.

As future work, we plan to integrate visual-servoing techniques to control PTU including the vehicle motion. In addition, it will be combine line tracking and obstacle avoidance algorithms for more complex applications.

Acknowledgments

This work has been partially funded by research projects GVPRE/2008/034, PAID-06-08-3246, DPI2001-2689-C03-02 and DPI2009-14744-C03-01.

Literature

[1] C. Lafuente, "AGV: automatizar el transporte y los flujos internos: flexibilidad, precisión y seguridad," Manutención y almacenaje, n. 42, pp. 71–77, 2005.

[2] www.egemin.com.

[3] www.robocoaster.com.

[4] M.B. Duinkerken, J.A. Ottjes, G. Lodewijks, "Comparison of routing strategies for AGV systems using simulation", Proceedings of Winter Simulation, pp. 1523-1530, 2006.

[5] S. Thrun, W. Burgard, D. Fox, "Probabilistic Robotics", MIT Press, 2005.

[6] K.R.S. Kodagoda, W.S. Wijesoma and E.K. Teoh, "Fuzzy speed and steering control of an AGV", The IEEE transactions on control systems technology, Vol.10, No.1, pp112-120, 2002.

[7] W.S. Wijesoma, K.R.S. Kodagoda and E.K. Teoh, "Stable Fuzzy State Space Controller for Lateral Control of an AGV", Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Vol. 32, pp189-201, 2002.

[8] M. Brady, H. Durrant-Whyte, H. Hu; J. Leonard, P. Probert;, B.S.Y. Rao, "Sensor-based control of AGVs", vol. 1, nº 2, pp. 64-70, 1990.

[9] G. Garibotto, S. Masciangelo, P. Bassino, C. Coelho, A. Pavan, and M. Marson, "Industrial exploitation of computer vision in logistic automation: autonomous control of an intelligent forklift truck," in Proc. Int. Conf, on Robotics and Automation, vol. 2, 1998, pp. 1459 – 1464.

[10] J. Pagés, X. Armangué, J. Salvi, J. Freixenet, and J. Martí, "A computer vision system for autonomous forklift vehicles in industrial environments," in 9th. Mediterranean Conf. on Control and Automation, 2001, pp. 379–384.

[11] M. S. J. Yoder, "Automatic pallet engagment by a vision guided forklift," in IEEE Conference on Robotics and Automation, 2005.

[12] M. Mora, V. Suesta, L. Armesto, and J. Tornero, "Factory management and transport automation," in IEEE Conference on Emerging Technologies and Factory Automation, vol. 2, 2003, pp. 508–515.

[13] L. Armesto, M. Mora, and J. Tornero, "Supervisión, teleoperación y navegación de vehículos industriales y su integración en el sistema de gestión," Revista Iberoamericana de Automática e Informática Industrial, vol. 2, pp. 55–63, 2005.

[14] L. Armesto, J. Tornero, AutoTrans: Management and transport automation in warehouses", Industrial Simulation Conference, vol. 1, pp. 236-241, 2005.

[15] Li. W;Xu C.;Xiad, Q; and Xu, X. "Visual Navigation of an autonomous robot using white line recognition", Proceedings of the 2003 IEEE Iternational Conference on Robotics and Automation (ICRA), 2003.

[16] T.H. Hong, T. Chang, C. Rasmussen and M. Shneier, "Road Detection and Tracking for Autonomous Mobile Robots", Proceedings of SPIE Aerosense Conference, Vol. 4715, 2002.

[17] M. Ollis and A. Stentz, "Vision based perception for an automated harverster", In proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), vol. 2, pages 1838-1844, 1997.

[18] B. Ross, J. Bares and C. Fromme "A semi-autonomous robot for stripping paint from large vessels", The International Journal of Robotics Research, 22 (7-8), pp. 617-626, 2003.

[19] www.cmucam.org.

[20] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Comm. ACM, vol. 24, no. 6, pp. 381- 395, June 1981.

[21] J. Matas and O. Chum, "Randomized RANSAC with Td;d Test," Image and Vision Computing, vol. 22, no. 10, pp. 837-842, Sept. 2004.

[22] "Randomized RANSAC with Sequential Probability Ratio Test," Proc. Int'l Conf. Computer Vision, vol. 2, pp. 1727-1732, Oct. 2005.

[23] G. Araujo and M. Aldon, "Line extraction in 2d range images for mobile robotics," Journal of Intelligent and Robotic Systems, no. 40, pp. 267–297, 2004.