# SOME APPROACHES TO LINE DETECTION ON A TASK OF LINE-FOLLOWING

*Dzmitry Kaliukhovich, Brest State Technical University, Brest*
*Vladimir Golovko, Brest State Technical University, Brest*
*Andreas Paczynski, Hochschule Ravensburg-Weingarten, Germany*

Keywords: task of line-following, line detection, Gaussian function, K-means algorithm, Hough transform.

The task of line-following consists in the robot's motion control to take a defined route which represented with one color line e.g. one color insulation tape. In spite of good contemporary investigation into this task nowadays developed and implemented solutions rarely deal with image processing techniques for line detection. Considering possible use of line detection algorithms in automated guided vehicles which can reduce cost of manufacturing and increase efficiency in a flexible manufacturing system [1] their development is still actual. In this paper some approaches to line detection based on Gaussian function [2], K-means algorithm [3] and Hough transform [4] were researched and tested .

As input data for the algorithms there were color images containing the line gotten from a web-camera. The image dimensions are 640x480 pixels. For the algorithms based on Gaussian function and Hough transform only the RGB colorspace is in use and for K-means one additionally the CIELAB colorspace is.

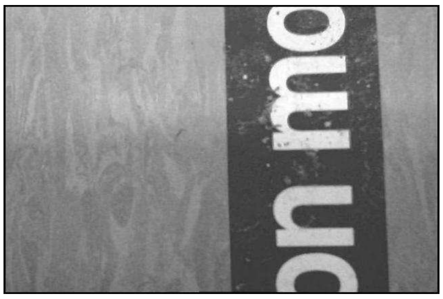
***Fig. 1. Initial image***

The algorithm based on Gaussian function:

1. Set initial values of means and standard deviations for all color channels *mred*, *sred*, *mgreen*, *sgreen*, *mblue*, *sblue*.
2. Set a threshold value *level*.
3. For each pixel to calculate values of belonging to the line according to:

$$belonging = red * green * blue, \tag{1}$$

*where:*

$$red = exp(-(r - mred)^2 / (2 * sred^2)), \tag{2}$$

$$green = exp(-(g - mgreen)^2 / (2 * sgreen^2)), \tag{3}$$

$$blue = exp(-(b - mblue)^2 / (2 * sblue^2)), \tag{4}$$

*(r, g, b)* – corresponding red, green and blue values of each pixel.

4. For each pixel to use the rule:

```
if belonging ≥ level
    then a pixel lies on the line
    else a pixel lies on the background
```

5. Repeat steps 1-4 for other values of means and standard deviations and then combine the separate results into one common result image.

While testing *(mred, sred, mgreen, sgreen, mblue, sblue)* = *{(*144.4, 10.6, 36.9, 7.5, 15.0, 7.8*), (*181.0, 6.7, 50.0, 7.6, 25.0, 7.6*)}*, *level* = 0.1 were used.

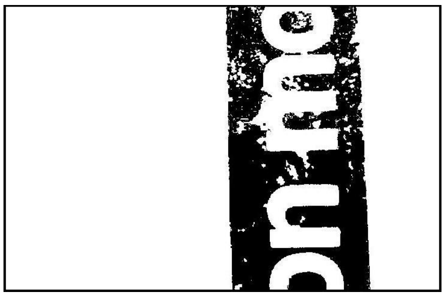

***Fig. 2. Processed with the algorithm based on Gaussian function***

K-means algorithm is an algorithm to cluster objects based on attributes into *k* partitions. A priori the amount of partitions is equal to *k* = 2 – for the line and the background. The dimension of the attributes space is equal to 3 for the RGB colorspace and to 2 for the CIELAB colorspace (the chromaticity layers 'a*' and 'b*' are in use). The algorithm based on K-means:

1. Randomly to generate initial centers of partitions (centroids).
2. Associate points with the nearest centroid according to Euclidean distance.
3. Move the centroids to the center of their respective clusters.
4. Repeat steps 1-3 until a suitable level of convergence will be reached.
5. Estimate pixels related to the line considering that their amount is less than the amount of the background pixels.

The maximum number of iterations is 100 but the real number strongly depends on initial centroid distribution.

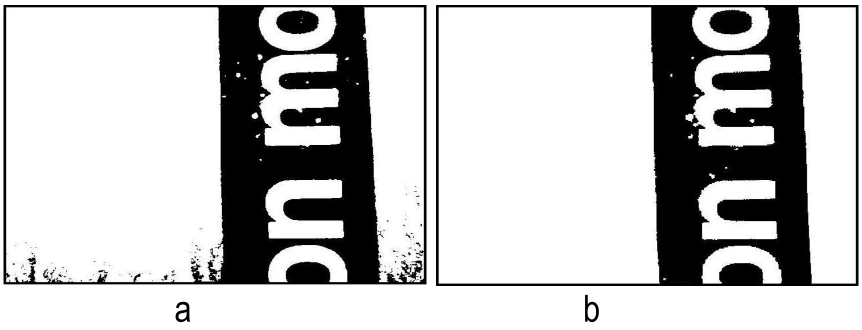

***Fig. 3. Processed with the algorithm based on K-means algorithm:***
***a – the RGB colorspace,***
***b – the CIELAB colorspace***

Taking into consideration that any line has borders with the background we can find left and right ones of it and then to approximate with straight lines. The first step have to be done is to convert a color RGB image into a grayscale image according to:

*grayscale* = $0.3 * r + 0.59 * g + 0.11 * b$, (5)

where see above the *(r, g, b)* parameters.

After that edge detection is going. There a lot of algorithms for it but one of the simple and rather efficient is Sobel operator. For the further processing both derivatives of the horizontal and vertical directions were calculated, the threshold value is equal to 100.

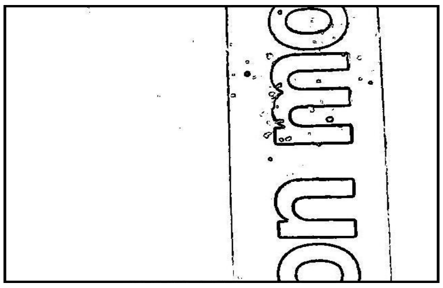

***Fig. 4. Edge detection with Sobel operator***

Line detection with Hough transform requires using of the normal parametrization of a straight line according to:

$$x * \cos\theta + y * \sin\theta = \rho, \quad (6)$$

where *(x, y)* is a point lying on a some edge, $(\theta, \rho)$ is the normal parameters specifying a straight line passing through the point *(x, y)*, $-\pi/2 \le \theta \le \pi/2$, $-(w2+h2)0.5 \le \rho \le (w2+h2)0.5$, *(w, h)* is the image dimensions i.e. the width and the height respectively.

For the program implementation Hough transfrom algorithm uses an array called accumulator to detect the existence of a line. The dimension of the accumulator is equal to the number of unknown parameters e.g. for Hough linear transform it's equal to *d* = 2. The two dimensions of the accumulator array were quantized for the $\theta$ and $\rho$ parameters. In tests the accumulator dimensions were 300x600 cells (the $\theta$x$\rho$ accumulator).

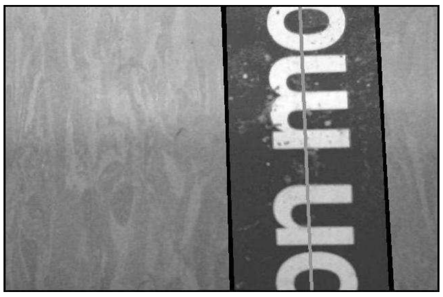

*Fig. 5. Processed with the algorithm based on Hough transform*

More comprehensive information about the line can be estimated with the „mean points of the line” representation. The representation lies in a division of the detected line pixels into some groups and then finding the mean coordinates of these groups of pixels. The least-squares method can be used to get only one straight line fully describing the observed part of the defined route.

*Fig. 6. „Mean points of the line” representation (marked with „+”) and a straight line gotten with the least-squares method*

All described above algorithms were realized and tested in MatLab v.7 for a set of images containing an insulation tape with the symbols „maxon motors” as a line. The algorithm based on Gaussian function was implemented in C++ and used for the motion control algorithms of the robot „MAX” [5] on a task of line-following. Visually the most precise detection was brought with K-means algorithm and the CIELAB colorspace but it coudn't be used in real time systems because of the time of his work isn't determinate. The solution is to get at the beginning the high quality line color characteristics with K-means algorithm and then to use the nearest neighborhood classification at the same time recalculating initial characteristics . As the conclusions the combination of K-means algorithm and Hough transform can be used for the robust line detection in real time.

This work has done in cooperation of the Intelligent Information Technologies Department, Brest State Technical University and the System Engineering Laboratory, Hochschule Ravensburg-Weingarten and was supported by the Landesstiftung Baden-Württemberg gGmbH.

## References

**1.** http://en.wikipedia.org/wiki/Automated_Guided_Vehicle (15-10-2007).
**2.** http://en.wikipedia.org/wiki/Gaussian_function (15-10-2007).
**3.** J. B. MacQueen. Some methods for classification and analysis of multivariate observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, Berkeley, USA. 1967.
**4.** Richard O. Duda, Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. Published in the Comm. ACM, Vol. 15, No. 1, pp. 11-15. January, 1972.
**5.** Zajac M., Paczynski A., Stetter R., Ucinski D. Sensor/Actuator network concept for robust control of a four-wheel-steering mobile robot. Paper. Hochschule Ravensburg-Weingarten, Weingarten, Germany. 2006.

## ОРГАНИЗАЦИЯ РАСПРЕДЕЛЁННОЙ ОБРАБОТКИ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ СРЕДСТВАМИ ПАКЕТА R

***Курочка К.С.***
*Гомельский государственный технический университет им. П. О. Сухого, г. Гомель*

Задачи анализа и распознования цифровых изображений приводят к обработке сотен гигабайт информации, что накладывает значительные требования на используемую вычислительную технику, от которой требуется не только высокая суммарная производительность обрабатывающих процессоров, но и большая оперативная память, достаточная для одновременного хранения сотен и тысяч изображений [1]. Даже самые быстродействующие современные ЭВМ последовательной архитектуры оказываются неспособными решать задачи такой сложности [2]. Таким образом, наиболее эффективным способом решения данных задач оказалось их распараллеливание и распределённая обработка.