ROBUST LINE DETECTION USING A WEIGHTED MSE ESTIMATOR

Guido M. Schuster

Abteilung Elektrotechnik
Hochschule Rapperswil, Switzerland
guido.schuster@hsr.ch

Aggelos K. Katsaggelos

Department of Electrical and Computer Engineering
Northwestern University, Evanston, Illinois, USA
aggk@ece.nwu.edu

ABSTRACT

In this paper we introduce a novel line detection algorithm based on a weighted minimum mean square error (MSE) formulation. This algorithm has been developed to enable an autonomous robot to follow a white line drawn on the floor, but is general in nature and widely applicable to line detection problems.

Traditional approaches to line detections consist of two stages, an edge detection stage and a line detection stage using the edge detection result. There are several problems with this approach. First, the initial edge detection stage is sensitive to noise. Second, the second stage does not use all the information available in the image and therefore incorrect decisions made by the first stage cannot be corrected in the second stage.

The proposed algorithm achieves its robustness by operating in one step, using all pixels of the image (correctly weighted) and not using any thresholds. The detected line is the solution of a weighted MSE problem. The following three questions are answered in the paper: (I) what mathematical model should be used for the line? (II) how should the weighted MSE problem be set up so that the optimal solution results in the parameters of the line model? And (III), how should the pixels in the image be weighted such that a weighted MSE optimal solution results in a robust line detection? Experimental results demonstrate the performance of the algorithm in noiseless and noisy conditions.

1. INTRODUCTION

The goal of a line detection algorithm is to find in a given grayscale image I a particular line of interest, as shown for example in white in fig. 1. Note that the line is not of one pixel width and in fact the width is changing with distance and perspective. We are therefore interested in the line in the middle of the white line as a simplified descriptor.

One powerful approach for straight line detection is the Hough transform and its variants [1]. When a curved line needs to be detected, as is the case in this paper, then the Hough transform becomes quickly so high dimensional that it is not feasible.

Another approach is to use an edge image E to find the line, since the line creates two strong edges as can be seen, for example, in fig. 2 which shows the squared norm of the gradient of a Gaussian (σ=1) lowpass filtered image of fig. 1. It is common to first find all the edges in the image, using an edge detection algorithm and then analyze the edge image to find the particular line. The main problems with this approach are that line detection algorithms are sensitive to noise and the detection of the line is now a two step process. If an error is made during the edge detection stage, it cannot be corrected in the line detection stage. Furthermore, most edge detection algorithms require one or more thresholds for selecting which points belong to edges and which not. Whenever a threshold is required, finding a robust value which works under all circumstances is a major problem which typically can not be solved in a satisfactory way. The proposed algorithm does not have these problems; instead it finds the parameters of a line model directly as the optimal solution of a weighted MSE problem.

The paper is organized as follows: In section 2 we introduce notation and assumptions necessary for the remainder of the paper. In section 3 we introduce a parametric model for the line. In section 4 we derive the weighted MSE solution to the line detection problem. In section 5 we propose the weighting matrix that is meaningful in our application. In section 6 we show results of the proposed algorithm for an autonomous robot application and section 7 concludes the paper.

2. NOTATION AND ASSUMPTIONS

The fundamental assumption in this paper is that there is only one dominant line in the image. Since the algorithm is based on a weighted MSE approach, the scheme would select a line as the solution that is the weighted average of the multiple lines, which would be incorrect. This does not mean that no other short line segment is allowed, such as the leg of a chair or a pattern in the floor, it simply means that the line we are trying to detect is the longest and therefore, among all other candidate lines, it has the most points in the original image. A second assumption is that the skeleton of the line we try to detect can be expressed as $c = f(r)$, where c and r are the column and row coordinates of the point and $f(.)$ will be defined in the next section. Although, a circle for example, does not satisfy this assumption, it is however reasonable since we look at the line from the perspective of a robot which is very close to the floor. In fact, points that are closer to the camera are more important since these are the points the robot needs to follow first. We will use this fact in section 5 to make the algorithm pay more attention to these points. We further assume that the grayscale image I is of dimensions $R \times C$, where R is the maximum number of rows and C the maximum number of columns.

3. LINE MODEL

The main purpose of the line model is to introduce the continuity and smoothness of a real line in the detection process. Furthermore, such a mathematical description of the line is a more useful abstraction for a higher level vision algorithm than a simple pixel based description. As discussed in section 2 we assume that the middle of the line we try to detect satisfies this equation: $c = f(r)$. To be exact, we assume that the function $f(.)$ is a low order polynomial, that is, $c = x_1 * r^o + x_2 * r^{o-1} + \ldots + x_{o-1} * r + x_o$, where x_i are the coefficients and o is the order of the polynomial.

It is important to note that we have also investigated other line models. Again, they all introduced some form of continuity and smoothness since these are features of real lines on a floor. One model which is more general than the one used above is that of a Hermite spline, where the column c and the row r are both polynomial functions of a parameter s. While this model allows for more complex lines, the resulting optimization procedure is more complex and will not be discussed in this paper.

Another way to enforce continuity and smoothness of a line is by following a snake based approach [2] and its more recent variants, such as deformable templates [3]. These approaches have the problem that they are not globally optimal but depend on the initial conditions. As we will show in section 4 the proposed solution does not require any initial conditions and finds in fact the globally optimal solution.

4. WEIGHTED MSE SOLUTION

Now that the line model is decided its parameters need to be estimated. We use the well known concept of MSE estimation via the generalized inverse (also known as pseudo-inverse or Moore-Penrose-inverse) of a matrix M which we denote by $(M)^\dagger$. Assume that we have the following equation $y = Ax$ where y is an $N \times 1$ vector, $x = [x_1, x_2, \ldots, x_{o-1}, x_o]^T$ is the line model parameter vector of dimensions $o \times 1$ and A is a matrix of dimensions $N \times o$. Then the MSE solution to this problem is: $\hat{x} = (A^T A)^\dagger A^T y$. In other words the total quadratic error $e^2 = (y - A\hat{x})^T (y - A\hat{x})$ is as small as possible for the selected $\hat{x}$.

Since we would like to be able to weight the individual components of the quadratic error, we introduce an invertible diagonal weight matrix W of dimensions $N \times N$. Hence we are interested in the solution that minimizes the following weighted quadratic error $e_w^2 = (y - A\hat{x})^T W^T W (y - A\hat{x}) = (W(y - A\hat{x}))^T (W(y - A\hat{x}))$. Note that this is an equivalent problem to minimizing the mean squared error of the following equation: $y_w = A_w x_w$, where $y_w = Wy$, $A_w = WA$ and x_w is the parameter vector that needs to be estimated. The solution to this problem is now $\hat{x}_w = (A_w^T A_w)^\dagger A_w^T y_w = (A^T W^T W A)^\dagger A^T W^T W y$.

Now that we have a line model and a method for finding the optimal solution to a weighted MSE problem, we need to set up the problem. In other words we need to define the matrix A, which contains the line model, the vector y which reflects the desired values and the weighting matrix W which reflects the importance of each pixel in the image. This is done in the next section.

5. WEIGHTING SELECTION

The basic idea of this algorithm is to use all available pixels for detecting a line that is given by a line model. In other words, no pixels are eliminated *à priori* but all of them are involved in finding the proper estimate. Clearly some pixels are more important than others and hence we use a weighted MSE scheme discussed in section 4 for estimating the parameters.

The line equation used is $c = x_1 * r^o + x_2 * r^{o-1} + \ldots + x_{o-1} * r + x_o$. This equation must hold for all line pixels (r,c) in the image. Since clearly the line pixels are not known in advance (they are the objective of the estimation process), the line equation is applied to all pixels in the image after they are appropriately weighted. Pixels that have a high intensity value are more likely to belong to the line, hence their weights should be high (in our experiments we used white tape to indicate the line shown in fig. 1). In other words, the weight for a pixel (r_i, c_i) should be proportional to its intensity value $I(r_i, c_i)$.

Furthermore, in the edge image E, which is in our case the squared norm of the gradient of a Gaussian ($\sigma = 1$) lowpass filtered image, high values belong to pixels that are on edges. Clearly the line creates two edges, but the weighted MSE solution results in the middle in the horizontal direction of these two edges, since at this point the average squared error is kept the smallest. Hence the weight of a pixel at (r_i, c_i) should also be proportional to the value in the edge image $E(r_i, c_i)$.

In our particular application, we are more interested in the part of the line closer to the camera than the one in the far distance. Hence we also weight the importance of the error proportionally to the closeness to the camera. To make sure that all these weights are of the same magnitude, we normalize the intensity image, denoted by $\tilde{I}$, such that its standard deviation is 1 and its minimum value is 0. We do the same for the edge image and call it $\tilde{E}$. One row i of the set of equations $Wy = WAx$ (there are $N = RC$ rows of them) has the form: $w_i * c_i = w_i * (x_1 * r_i^o + x_2 * r_i^{o-1} + \ldots + x_{o-1} * r_i + x_o)$, where $w_i = \tilde{I}(r_i, c_i) * \tilde{E}(r_i, c_i) * r_i / R$. Note that the error for a particular parameter vector $\hat{x}$ is now equal to $e_i(\hat{x}) = w_i * (c_i - (\hat{x}_1 * r_i^o + \hat{x}_2 * r_i^{o-1} + \ldots + \hat{x}_{o-1} * r_i + \hat{x}_o))$, which illustrates the idea behind weighting the error of the pixels differently.

In short, this setup is such that the solution will follow the edges and the bright areas closer to the camera as well as the order of the line model allows. Note that for our application it is quite meaningful to weight heavier the pixels closer to the camera. This clearly depends on the particular application, though it shows a nice feature of the algorithm which is the ability to include such prior knowledge into the formulation of the problem. For example in a tracking application, where the previous location of the line is known, one can use a location weighting function that favors solutions that are close to where a motion model would have predicted them.

6. EXPERIMENTAL RESULTS

Figure 1 shows the image acquired by the on-board camera. Only rows 15 through 60 are processed by the proposed algorithm, since from the mounting of the camera this is where we expect the line to be.

We have performed a series of experiments where we visually compare the performance of an automatic Canny edge detector [4] (the first step of a two step line detection process) and our proposed algorithm. First we show the performance in a noise-free environment, which represents images taken with sufficient ambient light and then we show results when the ambient light is low and hence the images are noisy. Polynomials of various orders were tested for describing the line. For the example of fig. 1 orders $o = 2, \ldots, 5$ gave comparable results. The results reported in this section were achieved using an order $o = 3$.

Figure 3 shows the result of the automatic Canny edge detector implemented in the MATLAB image processing toolbox. Figure 4 shows the same image processed by the proposed algorithm.

Clearly the result is at least as good as the Canny edge detector and in addition the proposed algorithm results in a parametric description of the line and not only in an edge image. Note that the pixels closer to the camera (bottom of the image) are better matched than the ones further away, which is the intention of the row-based weighting. Furthermore, in the MATLAB implementation of both algorithms our algorithm is about an order of magnitude faster.

For the next experiment we added zero mean additive white Gaussian noise (AWGN) to the image resulting in a signal to noise ratio (SNR) of 0dB, simulating a low light environment. The resulting grayscale image can be seen in fig. 5, the squared norm of the gradient of a Gaussian ($\sigma = 1$) lowpass filtered version of this noisy image can be seen in fig. 6 and fig. 7 shows the result of the Canny edge detector. Figure 8 shows the image of fig. 5 processed by the proposed algorithm. Clearly this is still a good result even though the SNR is so low that the Canny edge detector fails.

7. SUMMARY AND CONCLUSIONS

In this paper we have presented a novel robust line detection algorithm based on a weighted MSE formulation of the line detection problem. One of the powerful features of the algorithm is that it does not require any thresholds, but finds a solution as the optimal tradeoff between many different sources of information and constraints. The main constraint is the line model which represents our prior knowledge about the continuity and smoothness of the line we are trying to detect. The information sources are the intensity of the image and the edge image (but other ones can be added easily, depending on the particular application). The fusion of this information is achieved via a weighted MSE formulation. In addition, this framework allows for certain information to be trusted more than other, so, for example, we weight the information closer to the camera heavier than the information further away. The experimental results show that the proposed algorithm is quite robust and significantly faster than the Canny edge detector. Especially in a noisy environment the proposed algorithm can still reliably find the line. In comparison, at this SNR the automatic Canny edge detector cannot detect the edges anymore. Even though the algorithm was developed for the line detection stage of an autonomous robot, the scheme is quite general in nature and can be applied to many similar visual detection problems.

8. REFERENCES

[1] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Computer Vision, Graphics and Image Processing*, vol. 44, pp. 87–116, 1988.

[2] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321–331, 1987.

[3] A.K. Jain, Y. Zhong, and S. Lakshmanan, "Object matching using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 267–278, 1996.

[4] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, November 1986.

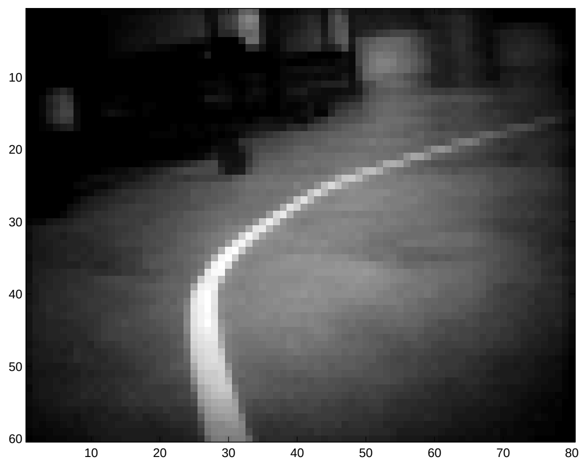

Fig. 1. Original grayscale image

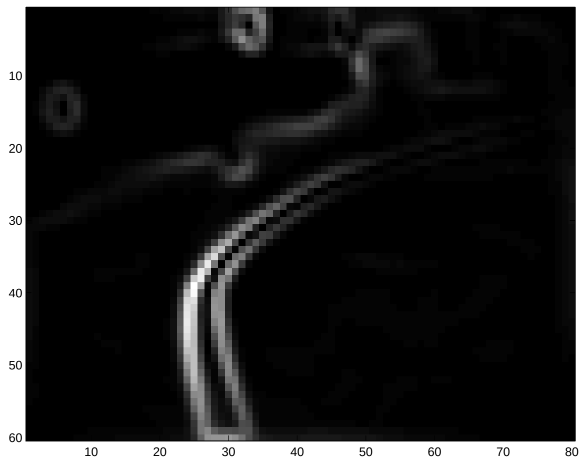

Fig. 2. Squared norm of the gradient of a Gaussian ($\sigma = 1$) lowpass filtered image shown in fig. 1

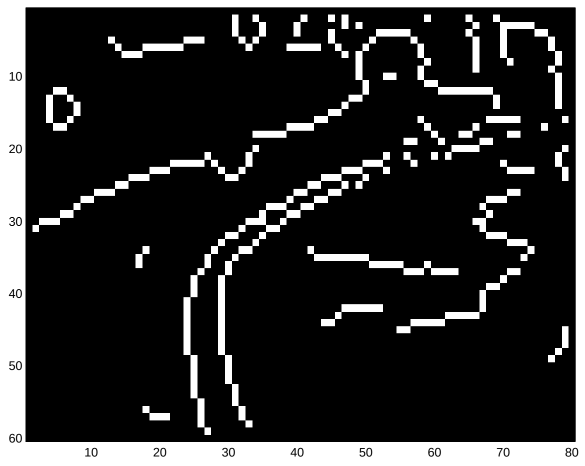

Fig. 3. Canny edge detector processing the image shown in fig. 1

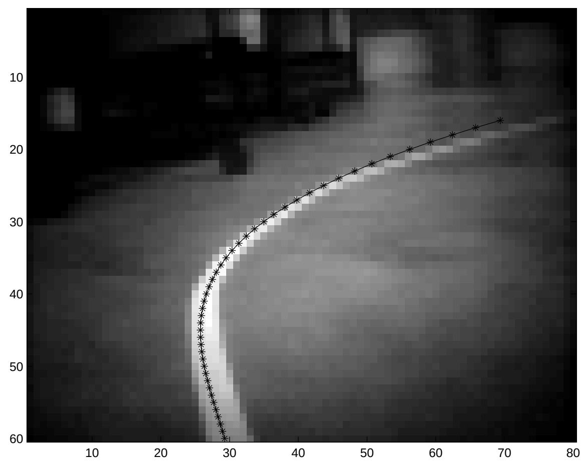

Fig. 4. Detected line in fig. 1 using the proposed algorithm

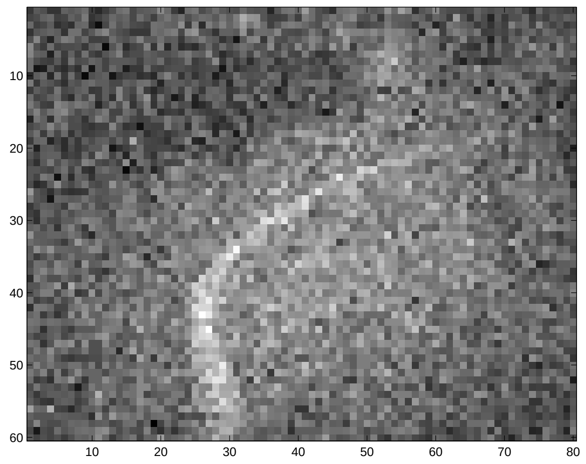

Fig. 5. Original grayscale image with AWGN @ SNR=0dB

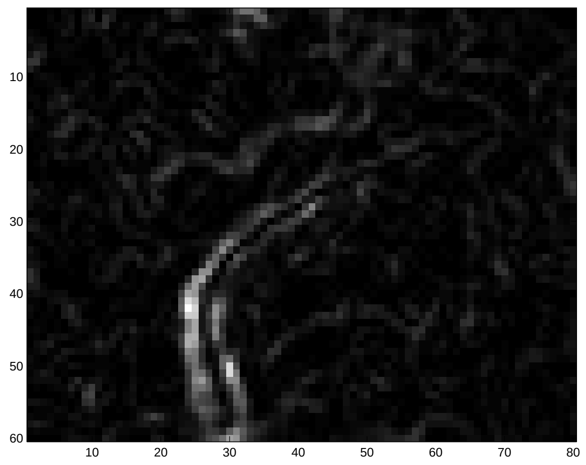

Fig. 6. Squared norm of the gradient of a Gaussian ($\sigma = 1$) low-pass filtered image shown in fig. 5

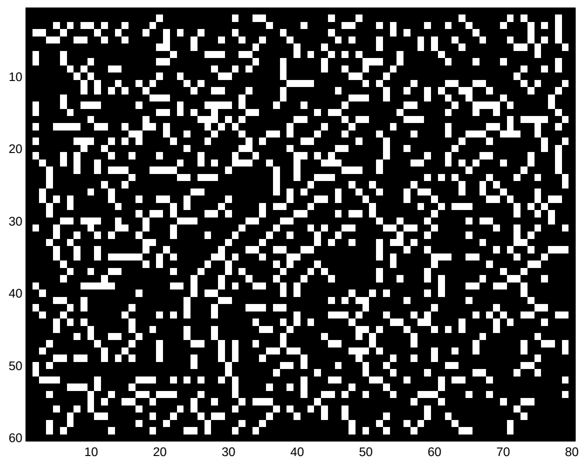

Fig. 7. Canny edge detector processing the image shown in fig. 5

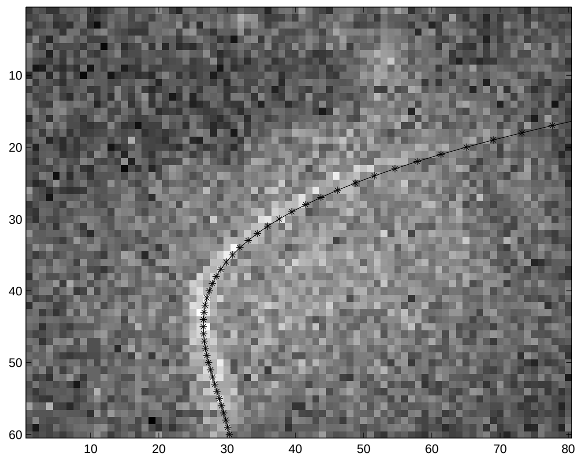

Fig. 8. Detected line in fig. 5 using the proposed algorithm