Curve Finder Combining Perceptual Grouping and a Kalman Like Fitting

Frédéric Guichard Jean-Philippe Tarel
INRETS
2, ave. du Gen. Malleret-Joinville
BP 34-94114 Arcueil-Cedex, France

Abstract

We present an algorithm that extracts curves from a set of edgels within a specific class in a decreasing order of their "length". The algorithm inherits the perceptual grouping approaches. But, instead of using only local cues, a global constraint is imposed to each extracted subset of edgels, that the underlying curve belongs to a specific class. In order to reduce the complexity of the solution, we work with a linearly parameterized class of curves, function of one image coordinate. This allows, first, to use a recursive Kalman based fitting and, second, to cast the problem as an optimal path search in an directed graph. Experiments on finding lane-markings on roads demonstrate that real-time processing is achievable.

1 Introduction

Our study is motivated by the detection - via on-board camera - of road lane markings for automatic vehicle guidance. The difficulty of finding road markings and boundaries stems from two main facts. First, such "features" often suffer from low image contrast. They may also be masked by shadows, light spots, be partially occluded, or even be physically fragmented. Second, the extraction of markings must be performed in a reasonable time on a standard on-board computer.

To alleviate these adverse conditions, we assume that markings are ideally embedded into a family of smooth curves, $\mathcal{S}$. A direct consequence of this modeling, is that we can now select edges based on their curve fitting performance, rather than a more traditional - and blind - gradient magnitude thresholding.

We favor the longest curves that belongs to $\mathcal{S}$ as characteristic of the features we seek to retrieve. Indeed, the existence of such curves in a typical image of a road, has a high probability to correspond to road boundaries or lane-markings.

When the family of curves $\mathcal{S}$ is of dimension 2 or 3, the Hough transform can be used to find the longest elements of that family. Nowadays, the most widely used curve finder involves, first, linking edgels via connectivity properties and, second, partitioning the result into line segments [7]. After such a partitioning the problem remains to aggregate such segments into curves [13]. In [4, 5] are proposed methods that follow the edgels and recursively fits a curve until the fitting error is large. But, important difficulties remain: the extracted curves are highly dependent on the selected starting points, as well as the order of the edgel linking. The approach we have explored tackles such problem.

Other approaches have been motivated by the idea of perceptual grouping. The edgels are organized as nodes of a graph, and linked to each other through arcs. The grouping relies on evaluating "perceptual cues", which are stored in each arc. Generally such cues correspond to some intuitive measure of the local geometrical consistency as evaluated for each pair of nodes. Measures such as alignment, co-circularity, and saliency have been proposed [14]. In [2] the grouping is modulated through statistical properties. Different algorithms have been proposed to find curves from such graphs, for instance: dynamic programming and relaxation (see [2, 1]). All such methods propose cues based on pair-wise interactions between edgels, and seem difficult to extend when a more global constraint on the curve is needed.

In [10], the author proposes a method that finds the longest convex subgraph. Convexity proves to be a strong enough constraint such that the computation can be performed by an exhaustive search.

We propose here to combine the above approaches tuned to the particular case of our road following application. The main difficulties are in designing (i) a global grouping technique that may result in a high combinatorial complexity, and (ii) a fitting technique (for the family $\mathcal{S}$) that involves a large amount of computations. However, the problem can be drastically simplified thanks to two strong assumptions made on $\mathcal{S}$. (1) We consider only parametric curves $x = f(y)$, where x is the horizontal coordinate, and y the vertical one. This is an acceptable assumption in the case of a vehicle well-aligned with the road. In turn this implies finding directed arcs between edgels, leading to a connected acyclic directed graph (i.e., a network). (2) We assume that $\mathcal{S}$ is a linear subspace of finite dimension, which allows us to use a recursive curve fitting.

The paper is organized as follow. First, we describe the variational statement of our problem. Second, we define a simple edgel detector based on level-lines. Third, we describe an efficient recursive implementation of the curve fitting using Kalman filtering. We then we show, how we can use it for finding curves described by $x = f(y)$ in an edgel set. Finally, we apply the designed technique to the retrieval of lane-markings on road images.

2 Geometric "Best-First Segmentation" of Edges

We want to select edges based on geometrical aspects. More precisely, we want to select a set of edgels corresponding to a shape approximatively in $\mathcal{S}$. We define the fitting error e^{fit} of a set of edgels, $\{E_0, \dots, E_m\}$, as the sum of the Least-Squared distances between the edgels and the best fitting shape s in $\mathcal{S}$:

$$e^{fit}(E_0, \dots, E_m) = \min_{s \in \mathcal{S}} \sum_{i=0}^{m} d^2(E_i, s)$$

A large error indicates that the edgel set cannot be well represented by a curve in $\mathcal{S}$. We obviously have

$$e^{fit}(E_0, \dots, E_m) \leq e^{fit}(E_0, \dots, E_m, E_{m+1})$$

which means that adding an edgel to a set of edgels will increase the fitting error. Therefore, in order to perform a grouping of edgels, we need to balance this increase of error. Hence we introduce a measure $e^{over}(\{E_0, \dots, E_m\})$ based on the sum of the edgel lengths and on their density within the fitted curve. To balance the fitting error, it is sufficient that e^{over} satisfies:

$$e^{over}(E_0, \dots, E_{m+1}) \geq e^{over}(E_0, \dots, E_m) + e^{over}(E_{m+1})$$

A simple example of such a measure is the squared sum of the edgel lengths.

An "energy", that indicates how consistent the edgels are with respect to the best curve in $\mathcal{S}$, can be defined by the weighted differences of e^{over} and e^{fit}:

$$G(E_0, \dots, E_m) = \lambda e^{over}(E_0, \dots, E_m) - e^{fit}(E_0, \dots, E_m),$$

where λ controls the tradeoff between e^{fit} and e^{over}. We then derive the energy gain of grouping an edgel E_{m+1} with a set of edgels $\{E_0, \dots, E_m\}$ by:

$$\Delta G = G(E_0, \dots, E_m, E_{m+1}) - G(E_0, \dots, E_m) - G(E_{m+1})$$

A positive ΔG represent a likely good grouping of edgels. A set of edgels having a large G is then clearly an important geometrical structure. Keeping only subsets, of the edgel set, that have a large enough G becomes a valid alternative over selecting edgels with respect to their contrast amplitudes. The problem is now how to find such subsets. Ideally, this involves finding the partition $\mathcal{P}$, of the edgel sets, that maximizes a "Mumford and Shah" like energy [12]:

$$E(\mathcal{P}) = \sum_{P \in \mathcal{P}} \left(\lambda e^{over}(\{E_i\} \in P) - e^{fit}(\{E_i\} \in P) \right) \quad (1)$$

Unfortunately, this problem is computationally difficult to solve for two reasons. First, the maximization is not local and its complexity is similar to the "salesman" problem. Second, this maximization of (1) requires the computation of e^{fit} for all subsets of the edgel set, which is highly computationally expensive. With respect to perceptual grouping techniques, the difference implied in our formulation is that global constraints are computed. Therefore, it is not obvious to directly apply one of the proposed algorithm that would computes an approximate solution. In addition, any iteration, with such algorithms, involves many fits of edgel subsets, which might not be computationally tractable.

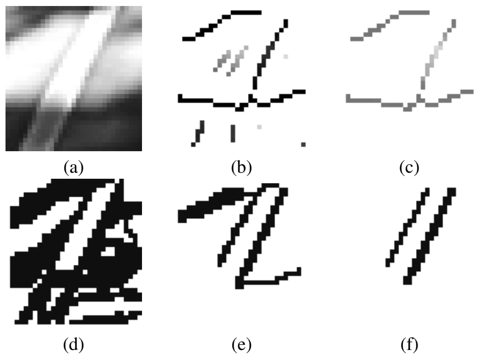


Figure 1. *(a) original image of a white lane perturbed by a spot light, results of Canny-Deriche edge detector with a 1 pixel size smoothing: (b) no threshold on the gradient magnitude and (c) 40 gray levels threshold. On the second line, results of the line segment detector for different values of the minimal length: (d) 4 pixels, (e) 15 pixels and (f) 20 pixels.*

Nevertheless, an approximative solution of (1) can be obtained in reasonable time under the following assumptions:

- Connected straight line edgels can be grouped together. Thus, edgels can be defined as straight line segments.

- The family of the shapes $\mathcal{S}$ is a linearly parameterizable subset of curves.
- The edgel set can be ordered. Therefore, the edgel graph is a connected acyclic directed graph.

Under these assumptions, we propose a new approach for an efficient partitioning which approximates the best solution $\mathcal{P}$ maximizing (1). Our approach consists in finding the longest edgel subset first. Then, to remove the found subset of the edgel set, and to iterate the optimal search for the next longest edgel subsets. With this partitioning approach, that we named *Best-First Segmentation*, the resulting subsets are ordered in decreasing energy.

3 Straight Line Segment Detector

3.1 Edgels as Straight Line Segments

Most of edge-detector algorithms involve (at least) a smoothing and a threshold steps [9]. Both steps decrease the number of resulting edgels, and in fact may remove useful information, as we explain below.

Smoothing. It removes from the image "small" details created by noise. Since, the chosen filtering is often linear, "small" detail means a "small" mix of spatial size and gray-level amplitude. Therefore the selection is harder on low-contrast zone. As example, we show in Fig. 1 (b) or (c), a Canny-Deriche edge detector applied on an image of a light spot on a white lane-markings. The magnitude of the gradient along the light spot is so strong that the smoothing removes the edge of the white lane-markings we want to detect.

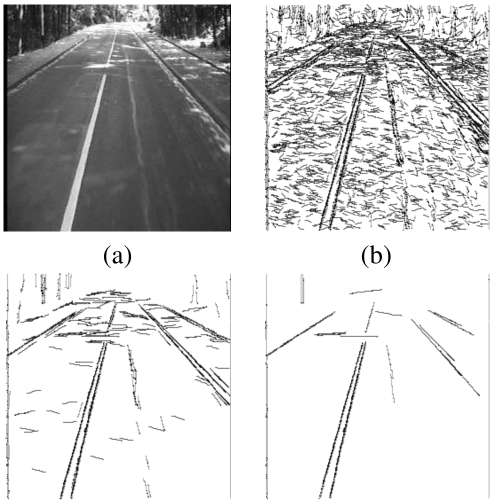
Thresholding. It usually discards low contrast candidate edgels.

If we reduce as much as possible the effects of the smoothing and threshold steps, edgels in images are numerous, and a criterion for selecting these becomes mandatory. We believe that a selection based on geometrical considerations is a better alternative than one based on intensity contrast, as illustrated in Fig. 1.

3.2 Extracting Edgels

We start with an edge map defined as the set of all the *level lines* of the image. (As defined in [8], we call "level line" the boundary of a level set L_μ, i.e., the set of pixels having an intensity larger or equal to μ). Note that, at this point, no selection is performed. Of course, others edge map definitions could work (e.g. lines given by the zero crossings of the Laplacian). The important point here, is to reduce as much as possible the use of contrast-based selections, given the problems outlined above.

We then define an edgel as a straight segment embedded in the edge map, or equivalently, as part of a level line. Due to the use of an image grid, there exists only 8 possible local directions for any pixel. These directions are coded by



(a) (b) (c) (d)

Figure 2. *(a) the original image and the result of the line segment detector for different values of the minimal length (b) 8 pixels, (c) 16 pixels and (d) 32 pixels.*

a number between 0 and 7, alike the well known Freeman codes. The list of directions on these connected edgels is thus equivalent to the chain-code of an edge.

Different algorithms have been proposed for recognizing when a chain code of a list of connected edgels is a straight line or not [11, 16]. Using such algorithms allows us to construct a complete tree of possible straight line chain-codes given a pre-specified target length. Due to the $\frac{\pi}{4}$ symmetry of the process, Freeman [16] proves that at most two basic directions are present in the chain code and these can differ only by unity, modulo 8. Therefore, this tree is a *binary tree*. Note that the size of such a tree remains relatively small.

Once the tree has been constructed, a fast algorithm for following connected straight segments of the edge map is used. Given a starting edgel, the tree of chain codes is traversed until a leaf is reached, i.e., until an end-point feature pixel is reached. We end up with a list of straight segments, denoted in the following by "edgels".

4 Curve Detector

In this section we explain how to group edgels lying on a certain shape and how to then find the principal curves from an image. As specified in Section 2, we first restrict our attention to a linearly parameterizable subset of underlying curves $\mathcal{S}$. Indeed as pointed out by [4], linear subspaces of curves allow recursive estimates of the curve parameters when a new edgel is provided. Noticeably, this is also the main property upon which Kalman filtering is based. Most common subsets such as straight lines, conics, cubics are examples of subspaces of curves. More complex curves may be approximated by higher degree algebraic curves [15].

We then consider the underlying curves explicitly described as a function of one of the image coordinates:

$$x = \sum_{i=0}^{d} f_i(y) a_i = F(y)^t A \qquad (2)$$

where (x, y) are the image coordinates of a point on the curve, $A = (a_i)_{0 \leq i \leq d}$ is the coefficient vector of the curve parameters, and $F(y) = (f_i(y))_{0 \leq i \leq d}$ is a vector of functions of the vertical coordinate y.

4.1 Recursive Fitting of a Curve in $\mathcal{S}$

As explained in Section 3, the used edgels are straight line segments with pixels as extrema. Thus, an edgel may be described by two pixel positions, i.e, by two points. Keeping in mind that we are still working on edgels, we will consider from now-on that the dataset comprises points only, for the sake of clarity.

The simplest way to fit a curve to data is to minimize distance over the set of given data points $(x_j, y_j)_{1 \leq j \leq m}$ with the least-squares criterion:

$$e_m^{fit} = \sum_{1 \leq j \leq m} (F(y_j)^t A_m - x_j)^2 \qquad (3)$$

The minimization of the previous fitting error gives the well-known normal equations:

$$M M^t A_m = M X_m \qquad (4)$$

where $X_m = (x_j)_{1 \leq j \leq m}$ is the vector of x coordinates, the matrix $M = (F(y_j))_{1 \leq j \leq m}$ is the *design matrix*, and $S_m = M M^t$ is the *scatter matrix*. Let $G_m = M X_m$, thus (4) is rewritten as $S_m A_m = G_m$. The computation of the best fit consists in solving the previous linear system.

Since the tasks of edgel grouping and curve fitting are not separable, a recursive algorithm is required. Given a new data point (x_{m+1}, y_{m+1}), we need to update the solution A_m to A_{m+1}. Therefore, an updated inverse of S_m is needed. We first compute the vector $F_{m+1} = F(y_{m+1})$. The updated scatter matrix is then given by $S_{m+1} = S_m + F_{m+1} F_{m+1}^t$, and we have $G_{m+1} = G_m + x_{m+1} F_{m+1}$.

In comparison to [4], where the recursive fitting is based on a QR decomposition of the design matrix M, we recast the curve fitting in the framework of Kalman filtering because less computer memory and power are then required. Kalman filtering is based on the following property for the updating:

$$(S + F F^t)^{-1} = S^{-1} - \gamma s^{-1} S^{-1} F F^t S^{-1} \qquad (5)$$

with $\gamma = (1 + F^t S^{-1} F)^{-1}$. Equation (5) assumes that the $p \times p$ matrix S can be inverted and is further based on the fact that F is a vector of size p. From (5), we deduce the recursive computation of $K_{m+1} = S_{m+1}^{-1}$:

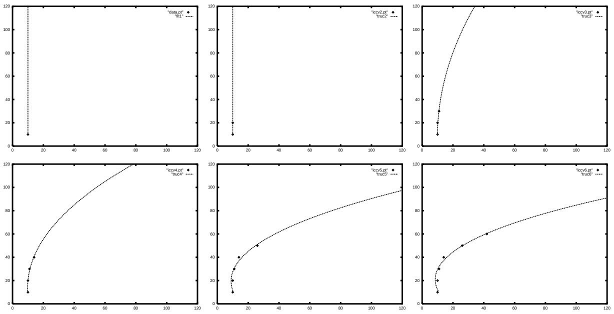


Figure 3. *Recursive fit with an increasing number of points (from 1 to 6) by a polynomial of degree 4. There is 5 parameters for this kind of curve. When the number of point is lower than 5, the recursive fitting algorithm gives curves of lower degree.*

$$K_{m+1} = K_m - \gamma_{m+1} K_m F_{m+1} F_{m+1}^t K_m \qquad (6)$$

with $\gamma_{m+1} = (1 + F_{m+1}^t K_m F_{m+1})^{-1}$. The previous equation gives the so-called *covariance* matrix K_{m+1} of A_{m+1} as a function of the previous covariance matrix K_m and the vector F_{m+1}.

The updated curve parameters are then obtained via:

$$A_{m+1} = A_m + K_{m+1} F_{m+1} (x_{m+1} - A_m^t F_{m+1}) \qquad (7)$$

The recursive fitting algorithm consists in:

- Select an edgel and initialize the recursive fitting by setting K_0 to k times the identity matrix, and A_0 to zero. Then compute the covariance matrix K_1 using (6) and the curve parameters A_1 using (7).
- Given a new data point (x_{m+1}, y_{m+1}), the covariance matrix K_m is updated using (6) and the curve parameter vector A_m is updated using (7).

The previous choice of K_0 insures that (5) can be applied at each step without any problem, even if the number of points is not sufficient for constraining well-enough the least-squares minimization of (3). This is equivalent to the *Ridge Regression* regularization investigated in the context of non-recursive fitting of algebraic curves [15]. As shown in Fig. 3, when the data does not contain enough information for the accurate estimation of curves of degree d, the algorithm fits the data set by a lower degree curve.

The fitting error can be recursively updated, without requiring the updated curve parameters A_{m+1} and the updated covariance parameters K_{m+1}, using:

$$e_{m+1}^{fit} = e_m^{fit} + \frac{(x_{m+1} - A_m^t F(y_{m+1}))^2}{1 + F^t(y_{m+1}) K_m F(y_{m+1})} \qquad (8)$$

which is obtained by substituting (6) and (7) in (3). This is of practical importance for optimizing the speed of the curve finder described in the next section.

4.2 Search for the Best Curve

As explained in Section 2, our algorithm is not finding the curves in the image in a random order, but, rather, it first find the longest, and then the others ones by decreasing energy G.

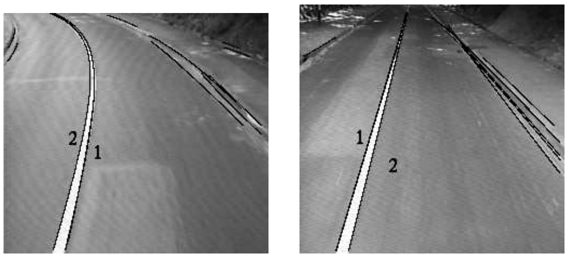


Figure 4. *Main curves detected. Both longest curves are numbered. All curves have 3 parameters.*

The assumption of explicit curves allows us to order the edgels, e.g. in a decreasing order of the explicit coordinate (here y). Then, starting from the bottom of a curve (in the image), that curve is always grown upward toward smaller y.

We organize the edgels as nodes in an acyclic directed graph, where every edgel is linked to all other consistent edgels with smaller y coordinates. Let E_1 and E_2 be two edgels, we say that $E_1 \rightarrow E_2$ if there is a direct link in the graph, from E_1 to E_2. We associate to each edgel E: $E.c$ its coordinates, and the b best curves arriving at E. Each curve is specified by its energy $E.e$ (i.e G), its parameters $E.A$, its covariance matrix $E.K$, and its length $E.L$.

The Moore-Dijkstra algorithm performs optimal search when arc weights are fixed. Contrary to the use of local cues, these weights are unknown with fitted curves. Therefore, we propose a variation of the classical Moore-Dijkstra algorithm to find the longest path in the graph with positive but unknown arc weights:

`For each ordered node` E`:`

1. `Compute the edgel energy` $G(\emptyset, E) = \lambda e^{over}(E.c)$.
2. `For the` b `best curves of every nodes` E' `such that` $E' \rightarrow E$ `compute:` $G(E', E) = E'.e + \lambda e^{over}(E'.L, E.c) - e^{fit}(E'.\{A, K\}, E.c)$.
3. `Let` $E_i.e$ `be the` $1 \leq i \leq b$ `best` G `in step 2 and 1. Then compute the` b `best curves associated to the best energies:` $E_i.\{A, K, L\} = RecFit(E'_i.\{A, K, L\}, E.c)$.

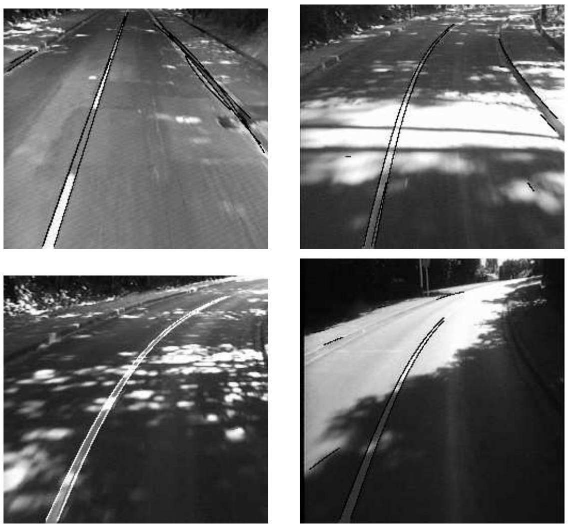


Figure 5. *Examples of main curves detected in images with complicated lightening conditions or with holes in the white lane-markings.*

At the end of the loop, the edgel with the curve of largest energy is the lowest coordinate edgel of the best curve. Finding the best curve is then straightforward. Let us described the ingredients of the algorithm:

e^{fit} is recursively computed as the distance of the extrema points of E to the chosen b best curve E'_i of E'. It uses (8) replacing x_{m+1}, y_{m+1} by the coordinates in $E.c$, and A_m, K_m by $E'_i.A$, $E'_i.K$.

e^{over} is the squared length, so that $e^{over}(E'_i, E) = (E'_i.L + E.L)^2 = e^{over}(E'_i) + e^{over}(E) + 2E.L\ E'_i.L$.

$RecFit$ denotes the recursive fitting where the edgel E is added to the fitted curve stored in E'_i. It follows formula (6) and (7), replacing A_{m+1}, K_{m+1} by $E.A$, $E.K$, and A_m, K_m by $E'_i.A$, $E'_i.K$. $RecFit$ may be reduced to initialize the fitting for the single edgel E to the straight line passing through it, as described in Section 4.1.

As we see, the algorithm involves two related loops on the edgels. Without, the proposed recursive process, a fitting step would have been needed *within* these two loops. Denoting by n the number of edgels, and considering an average of $n^{\frac{1}{2}}$ of edgels per curve, such a process would have yielded an average complexity of $bn^{\frac{5}{2}}$ (worse case is bn^3). The recursive fitting allows us to bring the fitting out of one loop as well as reducing its associated complexity. Inside the loops, it remains to estimate errors, which is a simple computation with fixed and small cost. The resulting complexity is therefore at worse bn^2.

The proposed algorithm represents a trade-off between optimality and efficiency. When b, the number of considered fitted curves for each node increases to the maximum path number, our search algorithm becomes optimal, to the detriment of processing speed.

5 Application to Real-time Video Analysis

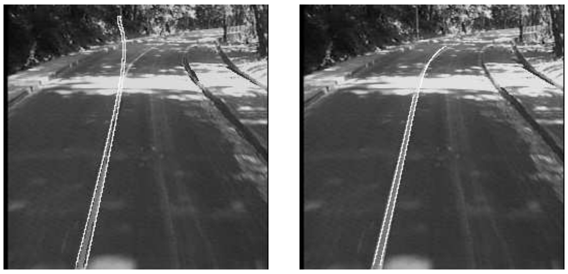


Figure 6. *Left: Example of longest curves finding using cues based on only pair of edgels. (A fitting is performed afterwards on the found edgel subsets). We see that since no global geometrical constraints is asked, the edgels of the white marks are linked to a telegraphic post located out of the road. Right: two longest lines found by the algorithm.*

In this section we present experiments for curve detection in the context of lane-markings recognition for automatic control of vehicles [3, 6]. We assume here that the road is planar and that its shape may be approximated by a polynomial: $x^* = \sum_{i=0}^{d} a_i \; {y^*}^i$. The transformation between the road plane (x^*, y^*) and the image plane (x, y) is: $x = l_x \frac{x^*}{y^*}$ and $y = l_y \frac{1}{y^*}$, where l_x and l_y are only functions of the camera calibration parameters. We set the origin of coordinate system to a point on the *line of horizon* - the position of this line can be computed from the camera calibration. We can then compute how the road is projected in the image as the curve:

$$x = \sum_{i=0}^{d} a_i y^{1-i} \tag{9}$$

We have found experimentally that in most cases, $d = 2$ or 3 is sufficient for a correct approximation of the road shape. In Fig. 6, we compute the two best curves using only local cues (a), and the two best curves that stands in $\mathcal{S}$ (b). The first algorithm links the white lane markings to a telegraphic post, which constitutes the best curve having a small mean curvature. Whereas the second follows the lane markings which is better represented by a function of $\mathcal{S}$. Figures 4 and 5 show the best found curves where d is respectively 2 and 3. Size of the images is 256×256. Typical computation time (in seconds) on a Pentium 200Mhz, 32Mo are: edgels set computation (keeping only those that are at least 8 pixels long): $0.04s$, and for best curve finding, when based on local cues: $0.06s$, when embedded in a family: between 0.1 and 0.5 second depending on d and on the image complexity (only the best fitted curve is saved $b = 1$).

6 Conclusion

We have described an algorithm for finding subsets of edgels that are embedded in a specific family of curves. Thanks to two assumptions made on the family of curves - i.e., linear parameterization, and functions of one coordinate - we derived a process based on a classical graph algorithm combined with a Kalman based recursive fitting. This allows the process to run in a reasonable time. We are currently working on optimizing this algorithm, and extending the technique for finding more generic curves.

References

[1] T. Alter and R. Basri. Extracting salient curves from images: An analysis of the saliency network. *IJCV*, 27(1):51–69, March 1998.

[2] A. Amir and M. Lindenbaum. A generic grouping algorithm and its quantitative analysis. *PAMI*, 20(2):168–185, February 1998.

[3] T. K. C. Thorpe, M. Herbert and S. Shafer. Vision an navigation for the canegie-mellon navlab. *PAMI*, 10(3), 1988.

[4] D. Chen. A data-driven intermediate level feature extraction algorithm. *PAMI*, 11(7):749–758, July 1989.

[5] I. Cox, J. Rehg, and S. Hingorani. A bayesian multiple-hypothesis approach to edge grouping and contour segmentation. *IJCV*, 11(1):5–24, August 1993.

[6] E. Dickmanns and A. Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *Proceedings of SPIE Conference on Mobile Robots S.161-16*, volume 727, 1986.

[7] M. Fischler and R. Bolles. Perceptual organization and curve partitioning. *PAMI*, 8(1):100–105, January 1986.

[8] F. Guichard. *Axiomatization of images and movies scale-space*. PhD thesis, University of Paris IX-Dauphine, 1994.

[9] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer. A robust visual method for assessing the relative performance of edge detection algorithms. *PAMI*, 19(12):1338–1359, December 1997.

[10] D. Jacobs. Robust and efficient detection of salient convex groups. *PAMI*, 18(1):23–37, January 1996.

[11] W. G. Kropatsch and H. Tockner. Detecting the straightness of digital curves in O(N) steps. *Computer Vision, Graphics, and Image Processing*, 45(1):1–21, Jan. 1989.

[12] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *CVPR*, pages 22–26, 1985.

[13] P. Rosin and G. West. Nonparametric segmentation of curves into various representations. *PAMI*, 17(12):1140–1153, December 1995.

[14] A. Shashua and S. Ullman. Grouping contours by iterated pairing network. *Neural Info*, 3:335–341, 1991.

[15] T. Tasdizen, J.-P. Tarel, and D. Cooper. Improving the stability of algebraic curves for applications. *accepted in IEEE Transactions on Image Processing*, 1999. also as LEMS Tech. Report 176, Brown University.

[16] L. Wu. On the chain code of a line. *PAMI*, 4(3):347–353, May 1982.