MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

A Single Processor Realtime Edge-Line Extraction System for Feature Tracking

Gunter Magin and Christian Robl *
Laboratory of Process Control and Real Time Systems
Prof. Dr.-Ing. G. Färber
Technische Universität München, Germany

Abstract

For robust feature tracking on a mobile robot moving with about 0.5 m/s in an office environment, it is necessary to keep up with a frame rate of at least 10Hz. For this case we introduce an unpublished method for real-time line segment extraction from grey level images by contour tracing. A special emphasis is laid on efficient algorithms for edge finding, contour tracing and symbolizing, to achieve the demanded frame rate for 512x512 8bit grey level images on a single processor system. All algorithms are geared to real time operation, avoiding recursions, enabling to priorize the processing order and aborting in case the deadline has been reached.

1 Introduction

Feature extraction is an essential first step in image interpretation. It is important for three reasons: data reduction, transformation to a symbolic description and noise limitation. First, the raw data, coming in from the sensor with a high rate, must be reduced, as the high-level interpretation process is computational expensive. Second, sensor data must be transformed into a symbolic and parametric form, which is suitable for higher level interpretation e.g. feature tracking. Third, every sensor reading contains noise. Feature extraction is needed to limit the influence of noise by averaging over space.

In this paper, we limit the term 'feature extraction' to the extraction of line segments in grey level video images. The scope of this paper is within the framework of autonomous mobile robots operating in indoor office environments. In such human made environments, linear structures preponderate, generally leading to straight edge lines in the resulting video image. Edges, which are not necessarily straight lines, are places in the image plane with intensity discontinuities in the grey level.

Feature extraction in the mobile robotics domain requires real-time capability. The cyclic deadlines are imposed by application demands, e.g. by localization with feature tracking. Controlling robot motion requires update cycles, depending on egomotion speed, of at-least 10Hz in our case. To gain a better controller stability, a higher rate is desirable. Real-time capability does not just mean speed, but also predictability of the response time. Since scene complexity and the resulting computational amount for feature extraction varies, special precautions in the algorithms must ensure the keeping of deadlines.

Some groups try to keep real-time limits using massive parallel computers, but the costs and programming complexity did not favor the diffusion of parallel image processing and pattern recognition in real applications [1].

After introducing the basic contour tracing algorithm in section 2, we show in section 3 how to obtain a starting point for each contour. The detection of corners and the way the symbolic description is gained from pixel chains is described in section 4. In section 5 we treat real-time aspects and in the last section we present two implementations and give experimental results.

2 Contour tracing

Contour tracing is used to overcome the time consuming multistage process of smoothing, labeling, thinning and symbolizing the image in classical edge line extraction schemes, like Burns et. al. [2], Canny [3], Laplacian-of-Gaussian [4], and others. In those schemes all pixels undergo processing with sometimes huge convolution matrices for each coordinate (2x2 up to 32x32). Being more selective in applying expensive computations in image processing is not a new idea [5], however, we go further and combine most steps needed for extracting a contour segment into three small operators, consisting of 5x5 matrices, whose convolution results are used to determine the next contour spot. A huge timing gain is reached by applying convolutions on only about 5% of the pixels of an image.

The matrices have been originally derived by an electrodynamic analogon, however, we found out that classical gradient and LoG matrices have delivered similar results. Nevertheless, with this mo-

*Address: Arcisstr. 21, 80290 Munich, Germany. E-mail: magin,robl@lpr.e-technik.tu-muenchen.de del one can easier understand how contour tracing works.

2.1 The electro-dynamical analogon

In our analogy, an electron or a 'positron' is driven along a contour by forces caused by electrostatic and magnetic field. These fields are no fields in a physical sense, and they have only those properties in common with them, which are explicitly mentioned. However, for simplicity, we continue to call them electrostatic and magnetic field. The path of the electron or positron consists of a sequence of positions, which are termed 'contour spots'.

Each pixel forms a horizontal and a vertical dipole, whose charges are derived by the local horizontal and local vertical gradient. The resulting electric field affecting the electron is composed by the contributions of the dipoles in the vicinity. An area of 5x5 has been found to be a good compromise between the precision of the result, and the required computation workload. Beside this, the size of the area influences the resolution of parallel edge lines. Choosing it too big contains the risk of being unable to distinguish close parallel contours. Choosing it too small may result in cross jumping due to local noise.

There is a restriction, that the electron's position is aligned to the pixel grid for calculating the distances to the single charges. This means, we approximate the field being constant within the pixel's area. Though this leads to satisfying results, there is an option of being less restrictive while spending the same computational effort, using different sets of matrices in an sub-pixel grid. Besides the electrostatic forces there is a magnetic force (Lorentz-Force) affecting the way the electron moves within the grey level image. The magnetic field is formed by a current flow in the conductor between the inexhaustible charges of each dipole. The balance between the two kinds of fields is adjustable with the parameter R (in our case R=11). This parameter is comparable to the resistance of each conductor. The magnetic field at a grid position in the image plain is composed of the contributions of all horizontal and vertical conductors in the 5x5 vicinity, similar to the electric field.

The three 5x5 matrices obtained by the described way can be seen as follows:

$$H = \begin{pmatrix} -1.5631 & -2.3419 & -1.5609 & -2.3419 & -1.5631 \\ -2.3419 & 0.0000 & 4.6838 & 0.0000 & -2.3419 \\ -1.5609 & 4.6838 & 12.4938 & 4.6838 & -1.5609 \\ -2.3419 & 0.0000 & 4.6838 & 0.0000 & -2.3419 \\ -1.5631 & -2.3419 & -1.5609 & -2.3419 & -1.5631 \end{pmatrix} * \frac{1}{R}$$

$$E_h = \begin{pmatrix} 0.4253 & 0.4617 & 0.0000 & -0.4617 & -0.4253 \\ 0.6900 & 0.9895 & 0.0000 & -0.9895 & -0.6900 \\ 0.1521 & 0.4880 & 0.0000 & -0.4880 & -0.1521 \\ 0.6900 & 0.9895 & 0.0000 & -0.9895 & -0.6900 \\ 0.4253 & 0.4617 & 0.0000 & -0.4617 & -0.4253 \end{pmatrix}$$

$$E_v = -E_h^T$$

2.2 Movement equations

Given a contour spot $\vec{x}_i$ at which the electron has the 'speed' $\vec{v}_i$ ($\vec{v}_i = 0$ if the given contour spot is a starting point), the next spot $\vec{x}_{i+1}$ has to be determined. For simplification all physical constants (e.g. $\mu_0, \epsilon_0, ...$) in the following equations have been set to 1 and then left out or have been accumulated to the parameter q, which is comparable to the electrical charge. The change of velocity being caused by the mentioned forces is:

$$\vec{dv}_i = q*(\vec{E}_i - \vec{H}_i * \begin{pmatrix} v_{y,i} \\ v_{x,i} \end{pmatrix}); \text{with } q = \begin{cases} +T & \text{positron} \\ -T & \text{electron} \end{cases}$$

Thus the sign of q is responsible for the tracing direction. The velocity gets normalized to

$$\vec{v}_{i+1} = \frac{(\vec{dv}_i + \vec{v}_i) * v_s}{\mid \vec{dv}_i + \vec{v}_i \mid}; \text{with } v_s = 1;$$

The value of the fixed step width and distance between the contour spots results from $v_s * T$. A step width of 2 has been found as a good compromise between accuracy and computational load. The next contour spot is determined to

$$\vec{x}_{i+1} = \vec{x}_i + T * \vec{v}_{i+1};$$

2.3 Algorithm interpretation

The Figure 1b shows the intermediate contour spots resulting from the contour tracer algorithm.

As mentioned above the electric field matrices are comparable to the classical gradient matrices (in this case Kirsch-Matrices [6]) and cause the electron to follow a straight and even a curved contour. But these fields are not capable of holding the electron on the ridge of the contour, therefore the magnetic field is necessary to stabilize the electron's motion. In addition, the magnetic field avoids too strong divergences when the contour is strongly curved. It is comparable to a second order derivative (in this case the Marr-Hildreth-Matrix [7]).

3 Starting points

The basic algorithm for contour tracing requires a starting point, which has to lie in the vicinity of a contour. For each isolated contour a separate starting point has to be found. Therefore, the image is divided into small rectangular commensurate grid elements, with application dependent side length (in our case 32 pixels). A given grid element is scanned horizontally and then vertically through the center with a step width of 2. This procedure runs as long as a pixel fulfills the following conditions. Otherwise no starting point was found for this grid element.

- $\mid E \mid > q$, where q is a given threshold. This relation checks whether the examined pixel belongs to a contour.

- | E | is local maximum referring to search direction. This condition tells if the ridge of the contour has been found.
- the found pixel must not belong to an already extracted contour

The third condition can be checked with a label image, in which the pixel values contain either the ID of a contour or zero. After a contour has been extracted, it is entered into the label image as a three pixel wide line with its ID. The third condition is confirmed if all pixels in a 3x3 rhomb around the examined pixel have not yet an ID entry. It is necessary to check such an area due to the strong influence of the 5x5 matrices, that tend to drag the contour tracer back to an already extracted contour.

After having found a suitable starting point, the whole contour is extracted in both directions, even if the contour leaves the actual grid element. Supposing the grid elements are smaller than the important contours in an image, no more starting points are required for this grid element, which is then marked 'ready'. Besides that, grid elements are also marked 'ready', as soon as a certain number of contour spots has been found within that grid element. This uses the fact, that edge segments have a minimum distance between each other.

4 Symbolic description

While tracing the contour, every new determined contour spot is checked to find out if it is a corner or an invalid contour spot (e. g. a contour spot lies not within the image). It is important, that, in spite of noisy pixel chains of contour spots, the resulting polygons have small divergences to the original contour. Additionally the contour should not be divided into too many short straight lines.

The new polygon approximation algorithm is based on calculating the sliding mean value of the secant slope angles between the corner point and the actual contour point. A corner is detected, when the difference between the actual secant slope angle and the sliding mean value is bigger than a dynamic threshold angle adjustment. This threshold is fading with the increase of the length of the contour segment as follows:

$$| \overline{\phi}_i - \phi_{i+1} | > \frac{\pi}{4i};$$

This dynamic threshold takes into account, that the influence of an outlier to the mean value is decreasing while the length of the contour segment is growing. This ensures, that a corner is also detected, when the curvature of a contour is very small.

After a corner point was detected, the symbolic line segment description is gained from the pixel chains by standard least square regression analysis. This further limits the influence of image noise and oscillations caused by the stabilizing effect of the magnetic force. The real corner point from the point of intersection of two successive regression lines.

In order to reduce the number of wrong detected corner points, caused mainly by the aforementionened oscillations, the angle between adjacent line segments is checked. If it falls below a threshold, the line segments are merged using standard least square regression analysis again.

The algorithm described above leads to a symbolic description of each extracted line segment. It contains not only geometrical attributes like length, angle and coordinates of the terminal points, but also additional properties delivered by the contour tracer as usable side effects like information about adjacent line segments, averaged magnetic and electric fields. Thus, there are more dimensions in feature space available for feature tracking, reducing false correspondences in case of ambiguities. Furthermore, the usual search time to classify junctions in L, V, T, Y or Lambda junctions is reduced using the explicit neighbor informations.

Comparing this algorithm to classical split & merge algorithms (e.g. [8]), which require the whole contour before recursive detection of corners is possible, computational load is avoided. The performance is comparable to so called scan-along-algorithms e.g [9]. Recursive approximations usually show better results than the sequential ones. However the computational load referring to real-time limits is only predictable with sequential algorithms.

5 Real-time aspects

The search for starting points described above enables to influence the order in which single parts of the image are processed, when using a translation table for the grid elements. This table can be filled using an a-priori knowledge of the position of edges or parts of the image which should be priorized (e.g expected lines, regions of interest) or generating a simple sequence (e.g. spiral, random or vertical). Less promising parts of the image will not be treated when the referring grid elements are always marked 'ready'. This is not a restriction for feature extraction, because for these grid elements no starting points are required. However a contour crossing such a grid element or leaving a treated grid element is of course extracted.

Supposing that the consumed time for extraction of a single contour is small referring to the given cycle time, after each contour the time left is checked and if necessary the extraction for the current image is aborted. In this case, it is necessary that the processing order determined with the translation table priorizes these parts of the image which presumably contain the relevant contours. This enables a suboptimal extraction result, when the real-time limit has forced to abort the extraction. Remembering the above mentioned treatment of 'ready' marked grid elements, it can be shown, that valuable time is gained and is available for a more complete feature extraction.

In principle it is possible to check the real-time limit after each contour spot,but this increases the

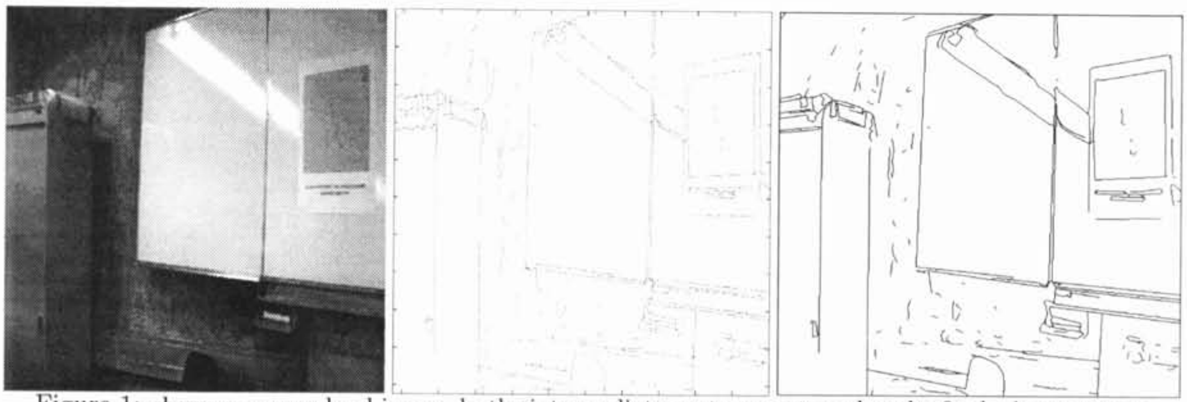
Figure 1: shows a; a grey level image, b; the intermediate contour spots and c; the final edge segments.

time resolution as well as the computational overhead. The reached time resolution however is sufficient for our application.

6 Results and future work

The contour tracer was implemented on two different target systems, a DEC alpha 3000/600 workstation at 175MHz and an Intel i860XR processor board at 40MHz. The averaged times for feature extraction using the presented algorithm on 512x512 and 736x287 (half frames) images of different indoor scenes (see figure 1a for a sample) without abortion can be seen in table 1 for both target systems. In the last line the average time consumed by the standard image processing package VISTA[10] for the same scenes was added to this table. The feature extraction with VISTA, using the Canny-operator [3] delivers comparable results. However, the Canny-operator is more sensitive to short edges (e. g. texture) and the contour tracer merges as much straight lines as possible. This explains the different number of edges found for the same scene. As it can be seen, we cut processing time by a factor of about 30 compared to classical algorithms. This result is not surprising, because according to [1] a single convolution for a whole 512x512 image with a 5x5 matrix needs more than 220ms on high performance RISC workstations (e.g DEC alpha 3000/800 at 200MHz) at highest optimization. The extracted straight

target	algorithm	ø edges	ø time
alpha	contour tracer	434	82.9ms
i860	contour tracer	434	194.6ms
alpha	canny	612	2800ms

Table 1: averaged processing times

lines with deviations of less than one pixel referring to the real contour can be seen in figure 1c for the sample image. Note, that there are all extracted line segments displayed, however, the shorter ones can be easily filtered off to focus on trackable features.

With processing under real time limit we reached a rate of 10Hz for feature tracking with 736x287 8bit grey level images on a processor, which is no longer state of the art (Intel i860 at 40 MHz). Processing in video rate (50Hz) can expected to be achieved with modern signal processors, like TMS320C80 MVP.

Future work will focus on improving the accuracy of the contour tracer without time penalty. Thus enabling to use these algorithms for the detection of soldering pads and pins in high precision 3D-Molded-Interconnection-Devices (MID) mounting.

Acknowledgment

The work presented in this paper was supported by the *Deutsche Forschungsgemeinschaft* as a part of an interdisciplinary research project on "Information Processing in Autonomous Mobile Robots"(SFB331).

References

[1] P. Baglietto, M. Maresca, M. Migliardi, N. Zingirian, "Image Processing on High-Performance RISC Systems", Proc. 84 IEEE Symp. On Parallel Architecture for Image Processing, July 1996.

[2] B. Burns, A. Hanson,E. Riseman, "Extracting Straight Lines", IEEE Trans. PAMI-8,Nr. 4, 1986.

[3] J. Canny, "A Computational Approach to Edge Detection" IEEE Trans. PAMI-8,Nr. 6, 1986.

[4] R. M. Haralick: "Digital step edges from zero crossings of second directional derivatives", IEEE Trans. PAMI-6, Nr. 1, 1984.

[5] P. Kahn, L. Kitchen, E. Riseman, "A Fast Line Finder for Vision-Guided Robot Navigation", IEEE Trans. PAMI-12, Nr. 11, 1990.

[6] D. H. Ballard, C. M. Brown, "Computer Vision", Prentice-Hall, Englewood Cliffs, 1982.

[7] O. Faugeras, "Three-Dimensional Computer Vision. A geometric Viewpoint", MIT Press, Cambridge, 1993.

[8] R. O. Duda, P. E. Hart, "Pattern Classification and Scene Analysis", Wiley, New York, 1973.

[9] K. Wall, P. E. Danielsson, "A Fast Sequential Method for Polygonal Approximation of Digitized Curves", Computer Vision, Graphics, Image Processing, 1984.

[10] A. R. Pope, D. G. Lowe, "Vista: A software environment for computer vision research", Proc. Int. Conf. on Computer Vision and Pattern Recognition, 1994.