Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

Automatic Selection of Control Points for Deformable-Model-Based Target Tracking

I. Pavlidis N. P. Papanikolopoulos

Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

Abstract

A novel curve segmentation algorithm for determining control points for deformable-model-based target tracking is proposed. The algorithm is parameterless enabling a full-fledged automated tracking regardless of the shape of the object being tracked. Compared with other curve segmentation algorithms, it selects a minimal number of control points that yet deliver a superior shape description. The algorithm is comparatively tested with other curve segmentation algorithms in a variety of characteristic target outlines.

1 Introduction

The need for target tracking arises in a number of different applications in robotics research. Characteristic examples include vision-based control of grasping and manipulation tasks [8, 10] and visual tracking of moving objects [2, 5]. Target tracking is also important in a number of other applications, like the case of pedestrian tracking in an Intelligent Transportation System (ITS) [9]. A well-established and popular technique for target tracking involves the use of deformable models [2, 9, 10]. Deformable models originate from the "snake," a model for representing image contours first developed by Kass *et al.* [4].

A necessary first step in the computation of a deformable model is to determine a set of control points to approximate the tracked object's contour. To date, this is done by hand through a user-interface. However, the possibility of using a curve segmentation algorithm is often indicated. Picking control points manually, renders difficult the automation of the entire tracking task. In addition, since the user is picking the points randomly or at best using some heuristic developed through his/her own experience, he/she tends to pick either too many or too few control points. On the other hand, using some classical curve segmentation algorithm [1, 3, 6, 7] only half-automates the task since the performance of these algorithms depends upon the fine tuning of a number of parameters. Different object shapes may require different parameter settings or otherwise the segmentation algorithm will perform at times either excessive segmentation or sparse segmentation.

In this paper we propose a segmentation algorithm (from now on, we will denote it as P & P) that fills out the existing gap in all the respects. It fully automates the selection of the control points since it does not depend on any parameters and works equally well for all kinds of shapes. Comparatively to other curve segmentation algorithms, it manages to select the minimal number of points that yet deliver a superior description of the original shape.

The organization of the paper is as follows: Section 2 presents some previous curve segmentation algorithms and discusses their shortcomings for the task at hand. Section 3 describes the algorithm we propose. In Section 4, the results from experimental tests are presented. Finally, in Section 5, the paper is summarized and conclusions are drawn.

2 Previous Work

Several interesting techniques that segment continuous lines in various ways have already been proposed in the literature in fields other than target tracking. The criteria against which the various curve segmentation algorithms should be judged for deformable-model-based tracking purposes are the following:

- First, the number of segmentation (control) points should be kept to the minimum, since the

0-7803-2988-4/96 $4.00

speed of deformable-model-based tracking is usually linear in the number of control points.

- Second, the control points chosen should deliver an accurate description of the tracked object's contour. This is important because it allows the deformable model to follow small deformations.

The above two criteria appear at first contradictory and researchers resorted to either choosing too many control points, thus compromising the tracking speed, or too few control points, thus compromising the quality of tracking. Interestingly, these two criteria are not quite met without careful or ad hoc parameter tuning by any of the most prominent curve segmentation algorithms. Three typical curve segmentation approaches are briefly presented in the following paragraphs for illustrative and comparative purposes.

A series of curve segmentation algorithms of the "split and merge" kind have been proposed by T. Pavlidis *et al.* [6, 7]. These algorithms are based on an iterative approximation of a curve by straight segments that drive an error norm under a specified threshold. This method (from now on, we will denote it as P) is suitable for the approximation of a curve by a polygonal line but has a tendency to result in too many segmentation points.

Freeman and Davis [3] proposed another type of segmentation technique (from now on, we will denote it as F & D) that involves the analysis of portions of the curve joining s points. The s points are used to locate the discontinuities along the curve. Three indices are calculated for each successive portion of s points; one index $C(i)$ is related to the severity of the curvature combining these s points, and the other two are related to the length of the discontinuity-free region (backwards ($I_b(i)$) and forward ($I_f(i)$)) from the point i. The importance of a given vertex i is calculated with the formula

$$Imp(i) = C(i)\sqrt{I_b(i) * I_f(i)}. \qquad (1)$$

The segmentation points are the ones located, after a proper filtering, at the maxima of $Imp(i)$. This interesting method, however, fixes arbitrarily the domain where a potential vertex could be found and is not able to detect "long and smooth" corners.

Brault and Plamondon [1] presented yet another segmentation algorithm (from now on, we will denote it as B & P) that could be considered as an improvement over the F & D algorithm. The main idea of their approach is that for each point i of the curve, the algorithm tries to iteratively construct a vertex centered on that point with the help of neighboring points to either sides of it until certain geometric conditions are met. In comparison with the F & D algorithm the B & P algorithm considers that a corner could be made by any number of points, and the algorithm itself must determine the length and the specific domain of every potential vertex. This algorithm, however, still depends on two other parameters that need to be fine tuned. In addition, while the algorithm employs a very powerful technique for detecting corners, it does not have an equally powerful way of detecting key points in round or flat curves.

3 The Algorithm

The algorithm we propose (P & P) takes a step further than the B & P algorithm. It does not depend on any parameters and thus offers a potential for true automation. It features a coherent mechanism for detecting not only corners but also some key points between corners. In this way, it improves its approximating power at the minimum cost. And because of its ability to locate some key points with relatively flat or round surroundings, it also performs satisfactorily in the case of rounded and flat objects.

3.1 Corner Determination

The determination of corners is done in a way similar to the method followed in the B & P algorithm [1]. The notable difference is that there is no need for parameter tuning. The basic idea is that for each point i of this curve, the algorithm tries to iteratively construct a vertex centered on that point with the help of neighboring points to either side of it until some conditions are met.

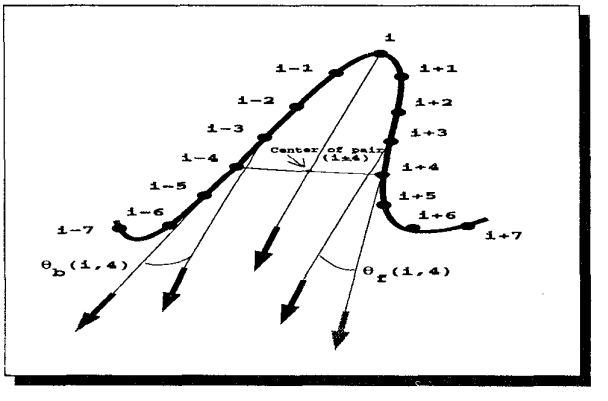

Figure 1: Geometric model for corner determination.

More precisely, the geometric parameters shown in Fig. 1 are calculated for each pair of neighbors $i \pm n$ (for $n = 1, 2...$). Intuitively, the more the two angles $\theta_f(i,n)$, $\theta_b(i,n)$ approach $\frac{\pi}{2}$, the less the corresponding pair of neighboring points contributes to the *cornerness* of point i. As a result, by a suitable analysis of the angles $\theta_f(i,n)$ and $\theta_b(i,n)$, one can determine whether or not the pair of points $i \pm n$ is part of the *corner domain* of i and, in addition, one can estimate the importance of the contribution of these points.

Determining the conditions for which a pair $i \pm n$ belongs to the corner domain of vertex i is straightforward. The angles $\theta_f(i,n)$ and $\theta_b(i,n)$ must satisfy the following inequality:

$$\theta_f(i,n) < \frac{\pi}{2} \quad and \quad \theta_b(i,n) < \frac{\pi}{2}. \tag{2}$$

The importance IMP_c of the contribution of each pair $i \pm n$ to the making of the candidate corner i is calculated by the empirical formula

$$IMP_c(i,n) = \cos(\theta_b(i,n)) * \cos(\theta_f(i,n)). \tag{3}$$

The trigonometric function *cos* was used for its adequate behavior in the range of angles concerned, whereas the multiplication operation takes into account the required simultaneous effect of the pair of points $i \pm n$ to make the vertex i important.

The total contribution of the first $N_{cd}(i)$ points belonging to the corner domain of i (the ones that satisfy the inequality (2)) is calculated by

$$TI_c(i) = \sum_{n=1}^{N_{cd}(i)} IMP_c(i,n). \tag{4}$$

The identification of the corner segmentation points from the function $TI_c(i)$ is very simple because this function usually comes in the form of groups of nonzero values spaced by a group of zero values. Each of the nonzero groups represents a corner, and the maximum value of each group is said to be the corner's apex, where the corner segmentation point is placed.

3.2 Key Flat Point Determination

While corners are the perceptually most important parts in a curve, corners alone provide insufficient data for an accurate reconstruction of the curve. The situation improves dramatically if we provide some key points with rather flat surroundings, that lie between corners, as extra segmentation points. The way we locate these flat points is conjugate to the way we locate the corner points.

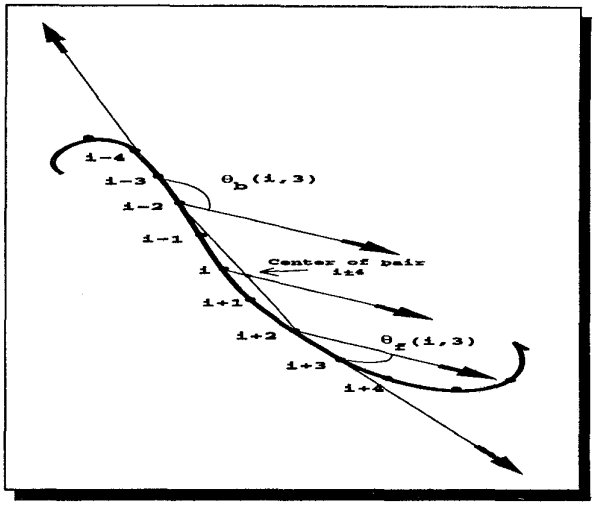

Figure 2: Geometric model for key flat point determination.

More precisely, a separate processing step is taking place for the location of the *key flat points*. The geometric parameters shown in Fig. 2 are the same with these in Fig. 1 and are calculated for each pair of neighbors $i \pm n$ (for $n = 1, 2...$) of every point i of the curve. This time, however, the larger the angles $\theta_f(i,n)$, $\theta_b(i,n)$ are than $\frac{\pi}{2}$, the more the corresponding pair of neighboring points contributes to the *flatness* of point i. As a result, by a suitable analysis of the angles $\theta_f(i,n)$ and $\theta_b(i,n)$, one can determine whether or not the pair of points $i \pm n$ is part of the flat domain of i and, in addition, can estimate the importance of the contribution of these points to the flatness of point i.

The angles $\theta_f(i,n)$ and $\theta_b(i,n)$ must satisfy the following inequality:

$$\theta_f(i,n) > \frac{\pi}{2} \quad or \quad \theta_b(i,n) > \frac{\pi}{2}. \tag{5}$$

The importance IMP_f of the contribution of each pair $i \pm n$ to the making of the candidate key flat point i is calculated by the empirical formula

$$IMP_f(i,n) = \mid \cos(\theta_b(i,n)) \mid * \mid \cos(\theta_f(i,n)) \mid . \tag{6}$$

In contrast to equation (3), equation (6) uses the absolute value of the trigonometric function *cos* since the range of the angles θ_f and/or θ_b features now $\frac{\pi}{2}$ as a lower and not as an upper limit. The total contribution of the first $N_{fd}(i)$ points belonging to the flat domain of i (the ones that satisfy the inequality (5)) is calculated by

$$TI_f(i) = \sum_{n=1}^{N_{fd}(i)} IMP_f(i,n). \quad (7)$$

The identification of the key flat segmentation points from the function $TI_f(i)$ is again very simple because this function, like the conjugate function $TI_c(i)$ in equation (4), usually comes in the form of groups of nonzero values spaced by a group of zero values. Each of the nonzero groups represents a flat or round segment, and the maximum value of each group is said to be the key flat point for the specific flat or round segment.

In the case of a completely round target, like a circle, there will be no corners and there will be one huge continuous run of points with nonzero TI_f values. The point with the maximum TI_f value, will be the only one key flat point of the curve. Actually, it will also be the only segmentation point, since there are no corner segmentation points for round objects. This is obviously a degenerate shape description. To overcome this difficulty, when the algorithm senses that the length of a key flat point run is close to the length of the curve, it partitions the large run to four subruns of equal length and assigns four key flat points instead of one to the original run. The number four was chosen because in the extreme case of a perfect circle, we need at least four equidistant control points for an accurate reconstruction of the original shape by using a spline representation.

4 Experimental Results

The algorithm was tested with the outline curves of two characteristic objects (see Figs. 3 and 7) we use for target tracking in our robotic experimental setup. It was also tested with the outline figure of a pedestrian image (see Fig. 11) taken from our Intelligent Transportation System experimental setup. The outline curve of the object is produced by applying an edge following algorithm on the difference image produced from the original dynamic scene. The same curves were also subjected to segmentation by the B & P and P algorithms for comparison purposes.

The comparative experiments between the three segmentation algorithms were designed in the following way. As it was explained in the previous section, no parameters needed to be fixed in our P & P algorithm. The algorithm produced a set of control points for each curve that accurately described the original figure. The accuracy of description has been tested by using the control points as interpolating points for cardinal splines. The resulting spline curves almost completely coincided with the corresponding original curves. A careful parameter tuning for the other two segmentation algorithms was performed for each curve. This happened in order for the algorithms to produce sets of control points that deliver similarly accurate descriptions of the original figures. Apart from the fact that parameter tuning was needed for the B & P and P algorithms, our algorithm produced substantially fewer control points for the same fine descriptive results.

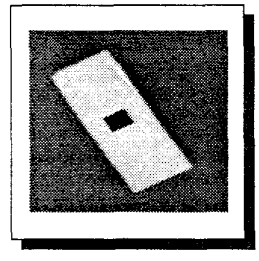

Figure 3: A rectangular target (black blob).

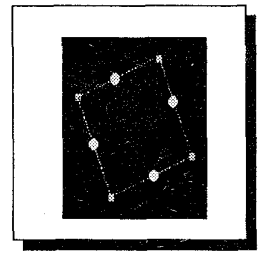

Figure 4: Rectangular outline segmented by P & P.

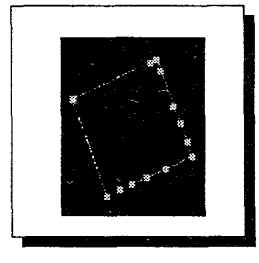

Figure 5: Rectangular outline segmented by B & P.

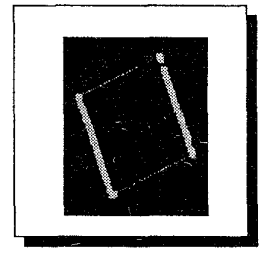

Figure 6: Rectangular outline segmented by P.

In the case of the rectangular target (Figs. 3 through 6), it is apparent the way our P & P algorithm works. The corners have been detected by the corner detection part of the algorithm, while the key flat points fall somewhere between the corners. The working logic is more obscure in the case of the B & P algorithm and even more so in the case of P algorithm, that seems to work simply by brute force.

In the case of the balloon target (Fig. 7), our algorithm (Fig. 8) selects only key flat points since the figure is round and there are no prominent corners. In the parts of the curve where the curvature changes more rapidly, there is a denser concentration of control points. In the parts of the curve where the curvature changes more slowly, the control points are

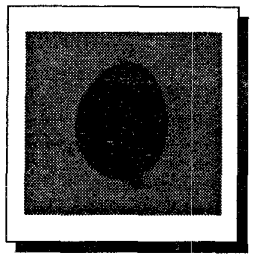

Figure 7: A balloon target.

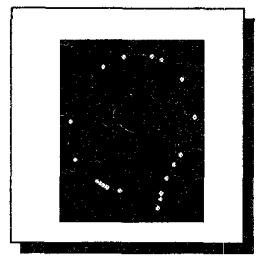

Figure 8: Balloon outline segmented by P & P.

Figure 11: A pedestrian target.

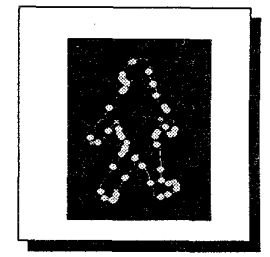

Figure 12: Pedestrian outline segmented by our algorithm.

Figure 9: Balloon outline segmented by B & P.

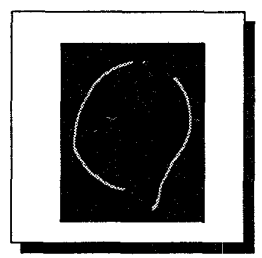

Figure 10: Balloon outline segmented by P.

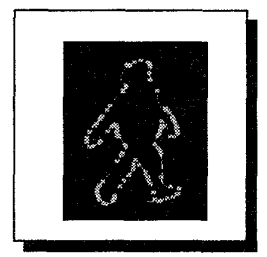

Figure 13: Pedestrian outline segmented by B & P.

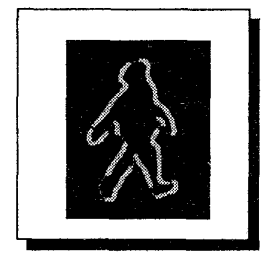

Figure 14: Pedestrian outline segmented by P & H.

more evenly and sparsely distributed. Such a distribution scheme makes both theoretical and intuitive sense. Again, the distribution of control points is less well thought in the other two algorithms.

In the case of pedestrian tracking (Fig. 11), the outline figure is a rather complex true real world figure. Our algorithm still outperforms the other two. It achieves this, by distributing control points more densely in parts of rapid curvature change and more sparsely in parts of slow curvature change. By following such a consistent strategy, the algorithm while economizes in control points it still delivers superior shape description.

In Fig. 15, bar charts that show the performance of the three segmentation algorithms for the various target types are presented. The results correspond to control point sets with highly accurate descriptive power for all three algorithms. It is apparent, that our P & P algorithm delivers the same high descriptive accuracy with the other two algorithms by utilizing a smaller number of control points. Interestingly, our algorithm performs comparatively better as the shape of the targets becomes increasingly complex.

5 Summary

A new curve segmentation algorithm for locating control points for deformable-model-based tracking has been described. Comparatively to other curve segmentation algorithms, the proposed algorithm delivers the minimal amount of control points for the same fine descriptive detail. Moreover, no parameters need to be fixed and the algorithm performs equally well for any type of curve. The algorithm achieves such results by segmenting the curve at its corner and some key flat points only. Its net effect is to distribute control points more densely at parts with high curvature change rate and more sparsely at parts with slow curvature change rate.

On one hand, the proposed algorithm completely automates the deformable-model-based robotic tracking and on the other hand optimally solves the contradictory requirements of using as few control points as possible for the deformable model, that still describe accurately the original figure.

The same algorithm can be used successfully not only in robotic deformable-model-based tracking but also in other deformable-model-based tracking applications like pedestrian tracking. Moreover, it can also be used as a generic segmentation algorithm.

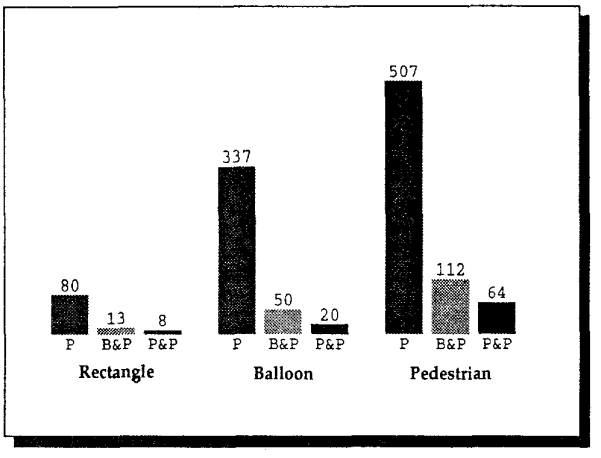

Figure 15: Performance in terms of number of control points for the three segmentation algorithms. The number of control points yielded in each case is shown at the top of the corresponding bar.

At the current implementation, the time complexity of the algorithm is $O(N^2)$, where N is the number of points that comprise the curve. Although, the algorithm still runs in real time for the usual real-world targets, it can easily be made faster. This can be achieved if we exploit the optimal substructure of the way corners and key flat points are found, by using dynamic programming techniques.

Acknowledgements

This work has been supported by the McKnight Land-Grant Professorship Program of the University of Minnesota, the National Science Foundation through Contracts $\#IRI-9410003$ and $\#IRI-9502245$, the Center for Transportation Studies through Contract $\#USDOT/DTRS93-G-0017-1$, the Minnesota Department of Transportation through Contracts $\#71789-72983-169$ and $\#71789-72447-159$, and the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number $\#DAAH04-95-2-0003$/contract number $\#DAAH04-95-C-0008$ (the content of which does not necessarily reflect the position of the policy of the government, and no official endorsement should be inferred).

References

[1] J. J. Brault and R. Plamondon. "Segmenting Handwritten Signatures at Their Perceptually Important Points." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, pp. 953-957, 1993.

[2] P. A. Couvignou, N. P. Papanikolopoulos, and P. K. Khosla. "Hand-Eye Robotic Visual Servoing Around Moving Objects Using Active Deformable Models." In *Proceedings of the 1992 IEEE International Conference on Intelligent Robots and Systems*, pp. 1855-1862, 1992.

[3] H. Freeman and L. Davis. "A Corner-Finding Algorithm for Chain-Coded Curves." *IEEE Transactions on Computers*, Vol. 26, pp. 297-303, 1977.

[4] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: Active Contour Models." *International Journal of Computer Vision*, Vol. 1, pp. 321-331, 1987.

[5] N. P. Papanikolopoulos and P. K. Khosla. "Feature-Based Robotic Visual Tracking of 3-D Translational Motion." In *Proceedings of the 1991 IEEE Conference on Decision and Control*, pp. 1877-1882, 1991.

[6] T. Pavlidis. *Structural Pattern Recognition*, pp. 168-184. Springer-Verlag, 1977.

[7] T. Pavlidis and S. T. Horowitz. "Segmentation of Plane Curves." *IEEE Transactions on Computers*, Vol. 23, pp. 860-870, 1974.

[8] C. Smith and N. P. Papanikolopoulos. "Grasping of Static and Moving Objects Using a Vision-Based Control Approach." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 329-334 1995.

[9] M. J. Sullivan, C. A. Richards, C. E. Smith, O. Masoud, and N. P. Papanikolopoulos. "Pedestrian Tracking from a Stationary Camera Using Active Deformable Models." In *Proceedings of the Intelligent Vehicles '95 Symposium*, pp. 90-95, 1995.

[10] B. H. Yoshimi and P. K. Allen. "Visual Control of Grasping and Manipulation Tasks." In *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 575-582, 1994.