Contour tracking and corner detection in a logic programming environment

Bell, Benjamin; Pau, L. F.

Published in:
I E E E Transactions on Pattern Analysis and Machine Intelligence

Link to article, DOI:
10.1109/34.57685

Publication date:
1990

Document Version
Publisher's PDF, also known as Version of record

it is required to prove that $\Phi'_{n+1} = P\Phi_{n+1}$:

$$a'_n = \frac{\langle \bar{r}' \mid \Phi'_n, \bar{\Phi}'_n \rangle}{\langle \Phi'_n, \Phi'_n \rangle}$$
$$= \frac{\langle \bar{P}(\bar{r} \mid \bar{\Phi}_n), P\Phi_n \rangle}{\langle P\Phi_n, P\Phi_n \rangle}$$
$$= \frac{\langle \bar{r} \mid \bar{\Phi}_n, \bar{P}^T P\Phi_n \rangle}{\langle \Phi_n, P^T P\Phi_n \rangle}$$
$$= \frac{\langle \bar{r} \mid \bar{\Phi}_n, P^T P\Phi_n \rangle}{\langle \Phi_n, P^T P\Phi_n \rangle}$$
$$= a_n, \quad \text{for } n = 1, 2, \cdots, N-1$$

$$b'_n = \frac{\langle \Phi'_n, \Phi'_n \rangle}{\langle \Phi'_{n-1}, \Phi'_{n-1} \rangle}$$
$$= \frac{\langle P\Phi_n, P\Phi_n \rangle}{\langle P\Phi_{n-1}, P\Phi_{n-1} \rangle}$$
$$= \frac{\langle \Phi_n, \bar{P}^T P\Phi_n \rangle}{\langle \Phi_{n-1}, P^T P\Phi_{n-1} \rangle}$$
$$= \frac{\langle \Phi_n, \Phi_n \rangle}{\langle \Phi_{n-1}, \Phi_{n-1} \rangle}$$
$$= b_n, \quad \text{for } n = 2, 3, \cdots, N-1.$$

When $n = 1$,

$$\Phi'_2 = \bar{r}' \mid \bar{\Phi}'_1 - a'_1\Phi'_1$$
$$= \bar{P}(\bar{r} \mid \bar{\Phi}_1) - P(a_1\Phi_1)$$
$$= P(\bar{r} \mid \bar{\Phi}_1 - a_1\Phi_1)$$
$$= P\Phi_2.$$

When $1 < n < N$,

$$\Phi'_{n+1} = \bar{r}' \mid \bar{\Phi}'_n - a'_n\Phi'_n - b'_n\Phi'_{n-1}$$
$$= \bar{P}(\bar{r} \mid \bar{\Phi}_n) - P(a_n\Phi_n) - P(b_n\Phi_{n-1})$$
$$= \bar{P}(\bar{r} \mid \bar{\Phi}_n - a_n\Phi_n - b_n\Phi_{n-1})$$
$$= P\Phi_{n+1}, \quad \text{for } n = 2, 3, \cdots, N-1.$$

So,

$$\alpha'_n = \langle r', \Phi'_n \rangle$$
$$= \langle Pr, P\Phi_n \rangle$$
$$= \langle r, \bar{P}^T P\Phi_n \rangle$$
$$= \langle r, P^T P\Phi_n \rangle$$
$$= \langle r, \Phi_n \rangle$$
$$= \alpha_n, \quad \text{for } n = 2, 3, \cdots, N. \qquad \text{Q.E.D.}$$

ACKNOWLEDGMENT

The authors would like to thank the referees for their valuable comments in improving the quality and significance of this paper.

REFERENCES

[1] Y. S. Abu-Mostafa and D. Psaltis, "Recognitive aspects of moment invariants," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 698–706, 1984.
[2] ——, "Image normalization by complex moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 46–55, 1985.
[3] F. Alt, "Digital pattern recognition by moments," in *Optical Character Recognition*, G. L. Fisher *et al.*, Eds. Washington, DC: Spartan, 1962, pp. 153–179.
[4] R. Askey, in *Orthogonal Polynomials and Special Functions*. Soc. Indust. Appl. Math., 1975.
[5] R. L. Cosgriff, "Identification of shape," Res. Foundation, Ohio State Univ., Columbus, 1960.
[6] R. J. Borel, "A mathematical pattern recognition technique based on contour shape properties," Res. Foundation, Ohio State Univ., Columbus, 1965.
[7] E. L. Brill, "Character recognition via Fourier descriptors," presented in WESCON, session 25, Qualitative Pattern Recognition Through Image Shaping, Los Angeles, CA, 1968.
[8] S. R. Dubois and F. H. Glanz, "An autoregressive model approach to two dimensional shape classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 55–56, 1986.
[9] S. A. Dudani, "Moment methods for the identification of three-dimensional objects from optical images," M.S. thesis, Ohio State Univ., Columbus, 1971.
[10] ——, "An experimental study of moment methods for automatic identification of three-dimensional objects from television images," Ph.D. dissertation, Ohio State Univ., Columbus, 1973.
[11] S. A. Dudani, K. J. Breeding, and R. B. McGhee, "Aircraft identification by moment invariants," *IEEE Trans. Comput.*, vol. C-26, pp. 39–45, 1977.
[12] G. F. D. Duff, in *Differential Equations of Applied Mathematics*. New York, Wiley, 1966.
[13] D. L. Fritzsche, "Systematic method for character recognition," Res. Foundation, Ohio State Univ., Columbus, 1961.
[14] E. L. Hall, W. O. Crawford, and F. E. Roberts, "Computer classification of pneumoconiosis from radiographs of coal workers," *IEEE Trans. Biomed. Eng.*, vol. BME-22, pp. 518–527, 1975.
[15] R. M. Haralick, "Digital step edges from zero crossing of second directional derivatives," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 58–68, 1984.
[16] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179–187, 1962.
[17] M. Jeffreys, *Methods in Mathematical Physics*. Cambridge, England: Cambridge Univ. Press, 1972.
[18] R. L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 627–637, 1981.
[19] C. C. Lin and R. Chellappa, "Classification of partial 2-D shapes using Fourier descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 686–690, 1987.
[20] K. T. Malhathappa and W. E. Brittin, *Mathematical Methods in Theoretical Physics*. New York: Gordon and Breach, 1969.
[21] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 170–179, 1977.
[22] J. G. Raudseps, "Some aspects of the tangent-angle vs. arc length representation of contours," Res. Foundation, Ohio State Univ., Columbus, 1965.
[23] C. W. Richard and H. Hemami, "Identification of three-dimensional objects using Fourier descriptors and the boundary curve," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, pp. 371–378, 1974.
[24] F. A. Sadjadi and E. L. Hall, "Numerical computation of moment invariants for scene analysis," in *Proc. IEEE Conf. Pattern Recognition Image Processing*, Chicago, IL, 1978, pp. 127–136.
[25] ——, "Three-dimensional moment invariants," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 127–136, 1980.
[26] G. Szego, *Orthogonal Polynomials*. New York: Amer. Math. Soc., 1959.
[27] C. H. Teh and R. T. Chin, "On image analysis by the methods of moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 496–512, 1988.
[28] J. Xu and Y. H. Yang, "A new technique for shape analysis using orthogonal polynomials," *Pattern Recognition Lett.*, vol. 7, pp. 191–197, 1988.
[29] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Comput.*, vol. C-21, pp. 269–281, 1972.

Contour Tracking and Corner Detection in a Logic Programming Environment

BENJAMIN BELL AND L. F. PAU

***Abstract*—The importance of shape information in object recognition has led to the emergence of numerous algorithms for contour trac-**

Manuscript received March 20, 1989; revised March 1, 1990.

The authors are with the AI and Vision Group, Bldg. 348/EMI, Technical University of Denmark, 2800 Lyngby, Denmark.

IEEE Log Number 9036260.

ing and corner identification. A common characteristic of these implementations is a parameter selection process. We present an integrated logic programming environment which provides a variety of image process operations. The capability of parameter selection through backtracking and unification is described. Extensions to the existing implementation with respect to higher level image understanding are discussed.

***Index Terms*—Contour tracking, corner detection, image processing, Prolog, truth maintenance, unification.**

I. Introduction

Information extracted from images may be stored in a multilevel knowledge representation structure. Image operations at each level within this structure use knowledge derived from operators at the levels below [14]. Object identification, for example, involves knowledge-based shape attribute operators, which in turn use differential edge detectors.

Of particular interest in this correspondence are contour tracking and corner detection. These operations are quite useful in most vision and image understanding tasks, providing shape information with attributes to higher level operators [15]–[17], [19]. Corner detection may be parameterized to provide different behaviors, depending on the understanding task. By performing these operations within a logic programming environment, it is possible to calibrate the parameters of the lower level operations with backtracking. Logic programming additionally offers the opportunity to combine image processing functions by AND/OR pruning predicates.

This correspondence deals primarily with corner detection and contour tracking, in that higher level scene understanding or classification implemented within the same environment are described elsewhere in detail [20].

II. Environment

To integrate image processing with logic programming, an interface is used between an Imaging Technology/ITEX 200 Image Processor and a MicroVAX based Quintus Prolog package. Using this interface, basic image operators are defined as predicates, which are then combined to form composite predicates. To investigate corner identification, predicates are defined at various levels for edge detection, object labeling, contour tracking, and corner detection.

A perspective view of a car is used as sample data. This image shows three windows which were the objects from which the contours and corners were extracted [Fig. 1(a)]. Applying predicates to perform low-pass filtering and binary thresholding results in the image shown in Fig. 1(b).

III. Lower Level Operators

Extraction of useful information begins with low-level operators for object labeling and edge detection [5], [18]. Here, we do not deal in detail with the selection of a specific thresholding or edge detection algorithm, in that several standard procedures are available, and choices are specified through suitable logic OR predicates. Object labeling is done on a line-by-line basis, first identifying all subobjects, then clustering the neighboring regions. The line-labeling procedure assigns unique pixel values to each object, and provides information about each object's area and centroid; it is nonetheless fast because of its line-by-line decomposition. The line-labeling predicate, augmented with object size filters, generates the image in Fig. 1(c). A variety of edge detection algorithms is provided in the ITEX library; these were included as predicates in the Prolog interface. Fig. 1(d) shows the result of applying a Sobel edge detection predicate to the labeled image.

IV. Contour Tracking

Object contours form a basis for shape description. Contour tracking from a set of edge points is typically tackled by a combination of edge linking, polygonal approximation, thinning, and neighborhoods to a contour element [16]–[19]. A contour is stored

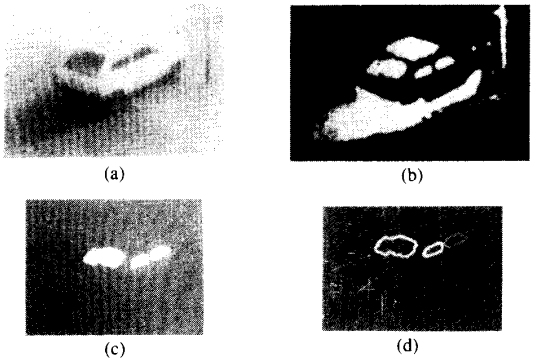

Fig. 1. (a) Original image of car. (b) After low-pass filter and binary threshold. (c) Line-labeled image. (d) After edge detection.

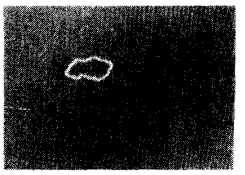

Fig. 2. Contour of one object generated from car image by Prolog predicate.

by encoding the direction from one edge element to its neighbor, using an eight-valued Freeman chain code scheme. To generate object contours, an image which has already been processed by one of the edge detection predicates is recursively processed as follows. The current edge element is surrounded in a three-element × three-element window, and these neighboring pixels are checked to locate one which is part of the detected edge. The search for a neighbor in this matrix of edges is guided by the straight line assumption, meaning that the first location checked is the one which would continue the direction of the edge in a straight line. If that location does not lie on the edge being traced, the location on either side, one chain code value away, is checked; if necessary, locations two and then three chain code values away are checked.

Recursion continues until the contour is closed or until all pixels on the edge have been checked. In the latter case, if the contour is not closed, the routine has the capability of closing it by recursively measuring the direction from the current pixel to the contour origin and choosing the neighbor closest to the measured direction. The distance over which the routine is permitted to close a contour may be adjusted, although the current perimeter length should be considered (e.g., Perimeter/10). Fig. 2 shows an example of a contour generated by this predicate.

V. Corner Detection

Shape attributes are analyzed by scanning a contour for pronounced changes in direction. When such a change is encountered, a corner is identified, and the search resumes for the next corner. The general procedure is to scan the entire contour, using a scanning window, the width of which is one pixel (since it contains segments of the contour) and the length of which is specified as an input parameter (SCAN). Corners are considered to be line endpoints, and the number of lines (corners) found is returned (LINECOUNT), as well as the position and slope of each line.

The algorithm implemented applies the difference of slopes approach to the chain-coded contour [1]. Corners are detected by moving the scanning window around the contour, one pixel at a time. The scan window is split in half, and each pixel in the first half is compared to the corresponding pixel in the second: the comparison in this case is the difference of the two chain codes (Fig. 3). These differences are summed over all pixel pairs in both half windows, and the sum, normalized for scan window length, is compared to a threshold. A turn in the contour is found when this

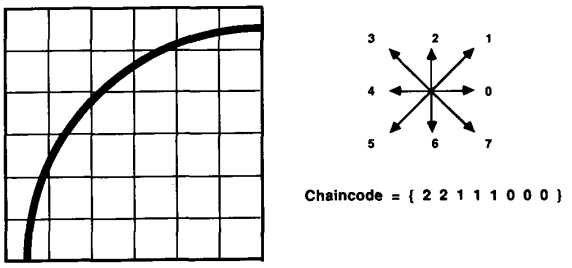

Fig. 3. Determination of curvature for contour segment within scan window.

threshold is exceeded, expressed as follows:

$$\frac{1}{\text{scan}/2} \sum_{i=1}^{\text{scan}/2} \left|\text{slope}[i] - \text{slope}[i + \text{scan}/2]\right| > \text{thresh}.$$

The threshold is dependent on two expressions. The first is the desired "target angle" (angle of the corners being searched for), and is expressed by the target angle in radians divided by $2\pi/8$ (since $2\pi/8$ represents the quantity "radians per chain code interval"). The second is an input parameter k ($0 < k \leq 1$), a coefficient determining how close to this target angle a scan segment must be in order to be classified as a corner. The threshold value is then

$$\text{threshold} = k * \frac{\text{target angle}}{2\pi/8}.$$

Once a turn is detected in a scan window, a pixel within the window must be selected as a reasonable choice for the precise location of the corner. The procedure employed checks each pixel by computing the average of the chain codes preceding it and the average of those following it (in the scan window). The pixel for which these two averages differ most is selected as the corner. The search begins in the middle of the scan window and moves outward. The computed difference in the two averages may be weighted in favor of the more central pixels by a coefficient set to $1 - (d * w)$, where d is the distance from the scan window center and w is a weight factor. The procedure thus may be expressed as follows. Given a scan window with elements $1 \cdots \text{scan}$, find position i which maximizes

$$\left|\frac{1}{i}\sum_{j=1}^{i} \text{chain}[j] - \frac{1}{\text{scan} - i}\sum_{k=i+1}^{\text{scan}} \text{chain}[k]\right| * (1 - d * w).$$

This corner detection capability is provided as a new Prolog predicate with arguments. The syntax of this predicate is of the form

EXTRACT_CORNERS

(Image, OBJECTNUM, SCAN, k, LINECOUNT).

To summarize the important parameters, SCAN is the length of the scan window in pixels; k is the coefficient $[0 \cdots 1)$ specifying how close to the target angle of turn a change of direction must come, and LINECOUNT is the number of lines extracted from the coutour (i.e., the lines connecting the corners). Examples of lines generated by the EXTRACT_CORNERS predicate are provided in Fig. 4.

VI. Application of Logic Programming to Image Analysis

The use of logic programming in conjunction with the image operations just described provides significant gains in functionality. This section will be centered around an application voluntarily selected to be simple, to allow for a better understanding of the workings of the Prolog-controlled corner detection of Section V. Other more complex scenes processed in the same way include the following.

- Extraction of cartographic features from airborne imagery where the corners are those on buildings, roads, parking lots, aprons, and runways [21]; this is already known [12] to be an area which depends crucially on top-down refinement and consistency adjustment to allow for feedback.
- Classification of car types in a road scene with several vehicles [20]; this extends the approach below with object-oriented Prolog object class libraries, and by context-sensitive logic predicates implementing the classification by suitable backtracking and constrained parameter unification.

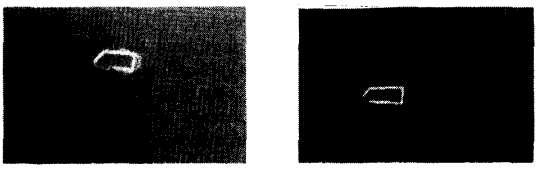

Fig. 4. Examples of lines generated by corner extraction predicate.

A. Unification of Parameters

Consider first Prolog's inherent backtracking capability; other types of constrained search strategies in Prolog are discussed in Section VI-C when higher level image understanding is carried out and embodies corner detection. If an image analysis predicate cannot be satisfied with the given arguments, Prolog can try alternate combinations by varying the (unbound) arguments. This feature is exploited, for example, in the specification of SCAN (the parameter describing scan window length). The length of the scan window is important because of its direct relation to the degree of smoothing it imposes on the contour [2]. Rather than supply SCAN as an argument, the user may instead define a valid set of possible values; Prolog will then try one value at a time, either returning results for each valid scan length or halting when a desired result is obtained. As another example, consider the k parameter (threshold coefficient). Selecting an appropriate threshold is central to effective corner detection [3], [4]. But if a valid set of possible k values is defined, Prolog will try different values from among this set, returning results for each k until backtracking halts. Furthermore, if multiple parameter values are variable predicate inputs, then the Prolog unification will generate them, and return these alternatives for analysis by higher level predicates, e.g., selection of windows for labeling and corner detection. This higher level analysis of intermediate, alternative unifications gives Prolog the capability of incrementally identifying multiple parameter values. Such an approach has been applied in production rule-type image analysis [5].

Even greater generality is possible when another facet of Prolog is considered: predicate arguments are neither strictly inputs nor strictly outputs. Input parameters are simply those which are bound to a value when the predicate is evaluated; output parameters are those which acquire values through unification. The implication of this for corner detection is that the behavior of the predicate depends on which (if any) of the arguments are bound (input) and which are free (output).

For example, in our initial discussion of the corner detection predicate, it was assumed the SCAN and k are input and LINECOUNT an output. This need not be the case. If, for example, LINECOUNT were specified as 4 and SCAN were left unbound, the predicate would determine the proper scan window length to use in order to fit a four-sided polygon to the contour. In fact, both SCAN and LINECOUNT may be left unbound, in which case Prolog's backtracking would find pairs of scan window lengths and the number of lines found. The threshold coefficient k may also be unspecified for that matter. If the aim were to identify all six-sided objects, only LINECOUNT would be fixed, leaving the calibration of SCAN and k to Prolog's internal mechanisms. As another example, with a short scan window and low threshold, the predicate behavior would generate a line segment approximation to the contour [6].

B. Search for Object

Throughout this discussion, one argument to the contour and corner detection predicates was not explained for purposes of clarity. This argument, OBJECTNUM, specifies a pixel value which uniquely identifies one object in the image to be processed (each object, remember, was assigned a unique pixel value by the line-labeling procedure described earlier). By restricting the processing to a single object, an image may be filtered, one object at a time. The filtering may, for example, retain only those objects which can be identified as four-sided polygons: filtering behavior depends, once again, on the configuration of the corner detection predicates arguments. A corner extraction method described in [7] generates widely varying results when different combinations of edge criteria (filters) are applied. Other behaviors are possible, depending on the test; logic programming provides a means of making such operations knowledge-based.

There is also an implication for contour tracing in that the selection of scale factors for smoothing can be based, for example, on curve size [8]. The logic programming environment thus could link the contour parameters (e.g., chain code resolution, pixel sampling interval) to the object shape being searched for.

The notion of searching for *objects* may be extended to *regions*. Cruse *et al.* describe the use of Prolog in selecting regions possessing particular combinations of attribute values [9]. Furthermore, searching may operate simultaneously on both objects and regions. A Prolog-based image analysis system presented in [10], for example, applies target classification (object search) and maneuver detection (region search) to the threat assessment problem.

Predicates easy to implement on the basis of those described in Section VI-A are

- list of open contours, having n corners of target angle θ
- list of closed contours, having n corners of target angle θ
- area of closed contour
- predicates expressing allowed transformations of contours with corner and endpoints, in view of changes of camera-to-scene perspective
- attention predicates, which turn TRUE if a contour is found in a fixed area of interest in the scene, and which return the attribute list and the position of that contour
- ambiguity generation predicates, which transform a contour with a property list into another one with a different property list.

C. Truth Maintenance

When object classification is considered (although it is outside the scope of this paper), the "freeze," "dif" delayed evaluation predicates on class allocation hypotheses will in turn allow for feedback-constrained parameter propagation down to the corner-finding predicates. Feature information derived from corner detection operations may appear in higher level predicates to form hierarchical, recursive object descriptions [10]. And if a more involved inference procedure invokes the image analysis predicates, very powerful tools become easily available. A truth maintenance system (TMS) operating on a selected configuration of objects will maintain/assert attributes of the objects based on required shapes or corners of unspecified items within this configuration [20]. This procedure also offers a way to accommodate incomplete/ambiguous image data through direct modification of the predicates which comprise the object descriptions. Modifications to selected clauses within such object description predicates can be automated to respond to missing data [11], adding a measure of evidential reasoning to the TMS.

Using logic programming as a framework for truth maintenance generalizes goal-driven image segmentation by jointly searching and selecting a segmentation. The consistency enforced by Prolog predicates acts as a natural pruning mechanism, a feature particularly relevant in view of the very large Prolog databases arising from characterizing images in terms of object attributes [9]. Special cases not constrained in this fashion should be handled by special-purpose predicates (goal-generation rules) to avoid a rapid expansion of the search space due to uncontrolled backtracking [12]. The fact that Quintus Prolog is available with a compiler and windowing features adds to the power of this technique.

VII. Conclusion

The added functionality which logic programming lends to standard image operators, specifically such as contour tracking and corner detection, is the focus of this paper. We have described an environment for implementing low-level imaging operations with Prolog predicates. Within this environment, higher level image predicates (contour tracking and corner detection) are constructed. Our emphasis is not to build better corner detectors per se, but to present ways of exploiting the unification and backtracking features of logic programming for these tasks. Corner detection is used as an example of image operation, both because of its usefulness in general image understanding tasks, and because its behavior is strongly influenced by the selected parameters. The performance of this implementation of contour tracking and corner detection has been very good in many more complex images, as it allows for feedback both ways between sensorial input and symbolic models.

More importantly, still, is this parameter selection capability in a dynamic version where background properties change. The search control also allows us to abide by speed constraints in a dynamic way.

We have presented examples of Prolog predicates to perform the contour and corner detection operations. These predicates include numerous arguments, and the roles of unification and backtracking in helping determine appropriate values for these arguments is discussed. Manipulating predicate parameters creates a filtering behavior which can be used to search images for desired objects. Finally, a truth maintenence system may be employed by properly constraining selected predicates and by eliminating objects no longer satisfying those constraints.

Continued work in this area will examine ways to combine logic programming-based image operators with scene understanding. Alternative unifications of parameters, generated by backtracking, may be analyzed by higher level image understanding predicates. Our work thus far suggests the gains achievable by straightforward application of logic programming. Incorporation of knowledge-based inference procedures promises to demonstrate greater improvements at multiple levels of the scene understanding hierarchy.

References

[1] L. O'Gorman, "An analysis of feature detectability from curvature estimation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, Ann Arbor, MI, 1988, pp. 235–240.

[2] H. Freeman and L. S. Davis, "A corner-finding algorithm for chain-coded curves," *IEEE Trans. Comput.*, pp. 297–303, Mar. 1977.

[3] Z. H. Huang and Q. Yu, "A direct corner detecting algorithm," in *Proc. 8th Int. Conf. Pattern Recognition*, Paris, France, 1986, pp. 853–855.

[4] K. E. Price, "Matching closed contours," in *Proc. 7th Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, 1984, pp. 990–992.

[5] A. M. Nazif and M. D. Levine, "Low level image segmentation: An expert system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 555–577, Sept. 1984.

[6] H. S. Tan and J. T. W. Damen, "Shape classification based on dynamic modelling procedure," in *Pattern Recognition in Practice II*, E. S. Gelsema and L. N. Kanal, Ed. Amsterdam: North-Holland, 1986.

[7] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 425–455, July 1986.

[8] D. H. Marimont, "A representation for image curves," in *Proc. 4th Nat. Conf. Artificial Intell.*, AAAI, Aug. 1984, pp. 237–242.

[9] D. Cruse, C. J. Oddy, and A. Wright, "A segmented image data base for image analysis," in *Proc. 7th Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, 1984, pp. 493–496.

[10] L. F. Pau, "Knowledge-based real-time change detection, target image tracking, and threat assessment," in *Machine Intelligence and Knowledge Engineering for Robotic Applications*, A. K. C. Wong and A. Pugh, Ed. Berlin: Springer-Verlag, 1987.

[11] P. Kuner and B. Uebereiter, "Knowledge-based pattern recognition in disturbed line images using graph theory, optimization, and pred-

icate calculus," in *Proc. 8th Int. Conf. Pattern Recognition*, Paris, France, 1986, pp. 240–243.
[12] D. M. McKeown, Jr., W. A. Harvey, Jr., and J. McDermott, "Rule-based interpretation of aerial imagery," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 570–585, Sept. 1985.
[13] F. H. Cheng and W. H. Hsu, "Parallel algorithm for corner finding on digital curves," *Pattern Recognition Lett.*, vol. 8, pp. 47–54, July 1988.
[14] D. T. Lawton and C. C. McConnell, "Image understanding environments," *Proc. IEEE*, vol. 76, pp. 1036–1050, Aug. 1988.
[15] A. Guiducci, "Corner characterization by differential geometry techniques," *Pattern Recognition Lett.*, vol. 8, pp. 311–318, Dec. 1988.
[16] O. Baruch, "Line thinning by line following," *Pattern Recognition Lett.*, vol. 8, pp. 271–276, Nov. 1988.
[17] M. H. Han, D. Jang, and J. Foster, "Identification of cornerpoints of two dimensional images using a line search method," *Pattern Recognition J.*, vol. 22, no. 1, pp. 13–20, 1989.
[18] C. Ronse and P. A. Devijver, *Connected Components in Binary Images*. London: Wiley (Research Studies Press), 1986.
[19] M. Xie and P. Rivers, "Estimation des segments 2 D: Un algorithme Robuste," INRIA, Rennes, France, Res. Rep. 909, Oct. 1988.
[20] B. Bell and L. F. Pau, "Context knowledge and search control issues in object-oriented PROLOG-based image understanding," EMI, Tech. Univ. Denmark, Lyngby, Tech. Rep. R #402, Sept. 1989.
[21] L. F. Pau and L. Kanal, Ed., *Mapping and Spatial Data Structures*. Heidelberg: Springer-Verlag (ASI Series), 1990.

Speaker Adaptation in a Large-Vocabulary Gaussian HMM Recognizer

P. KENNY, M. LENNIG, AND P. MERMELSTEIN

***Abstract*—This correspondence is concerned with the problem of using a small amount of speech data to adapt a set of Gaussian HMM's that have been trained on one speaker to recognize the speech of another. We experimented with a phoneme-dependent spectral mapping for adapting the mean vectors of the multivariate Gaussian distributions, which is analogous to the confusion matrix method that has been used to adapt discrete HMM's and a heuristic for estimating covariance matrices from small amounts of data. Our best results were obtained by training the mean vectors individually from the adaptation data and using the heuristic to estimate distinct covariance matrices for each phoneme.**

***Index Terms*—Confusion matrix method, multivariate Gaussian hidden Markov model, probablistic spectral mapping, smoothing, speaker adaptation.**

I. Introduction

We are studying the design of a speaker-dependent isolated word recognizer with a very large vocabulary. The recognizer is phoneme-based, with each phoneme being represented by a left-to-right HMM [1]. We have found that Markov models with multivariate Gaussian output distributions give better performance than those having discrete distributions defined on a VQ codebook [2]. When phonemes are modeled using unimodal Gaussian HMM's, the performance of our recognizer saturates when the training data are somewhere between 1000 and 2000 words (depending on the speaker) [3]. Speakers without a vested interest in our project usually balk at the prospect of uttering this amount of data in isolated word mode.

This correspondence addresses the problem of transforming a set of Gaussian HMM's that have been trained for one speaker (the "old" speaker) to recognize the speech of another (the "new" speaker) using a much smaller amount of adaptation data. The problem of adapting *discrete* HMM's from one speaker to another has already received considerable attention. The techniques that have proved most successful in this case are essentially sophisticated methods for smoothing the maximum likelihood estimates of the output probability distributions of the HMM's obtained from a small amount of adaptation data. As such, they cannot be carried over directly to the multivariate Gaussian case, and new approaches have to be developed.

In our system, speech is recorded in a quiet sound booth, low-pass filtered at 7 kHz, and sampled at 16 kHz. A Hamming window with a width of 25.6 ms is applied at 10 ms intervals and a set of 8 mel-frequency cepstral coefficients [4] ($c_0, \cdots, c_7$) is calculated together with their differences over an interval of four frames ($\Delta c_0, \cdots, \Delta c_7$). As our feature vector, we use ($c_1, \cdots, c_7, \Delta c_0, \cdots, \Delta c_7$). (Note that the loudness c_0 is not included.) Accordingly, we associate a 15-dimensional Gaussian distribution with each transition in the Markov chains underlying the HMM's. The covariance matrices of these distributions are tied across the transitions in each phoneme model. (The reason for this is to ensure that the covariance matrices can be robustly estimated; it also substantially decreases the amount of computation needed for recognition.)

We ignore the problem of adapting the transition probabilities of the Markov chain, and concentrate on the parameters of the output distributions, namely, the mean vectors and covariance matrices. The most straightforward way of adapting the output distributions is to estimate their parameters directly from the adaptation data using the Baum–Welch algorithm, with the old speaker as the starting point. To compensate for the lack of data, it is necessary to find some way of smoothing these estimates. This approach has been found to work quite well for VQ-based recognizers [5], [6], but a more promising approach seems to be to adapt the output distributions collectively using the probabilistic spectral mapping of BBN [7] (also known as the confusion matrix method [8]). The idea behind this method is to assume that the output distributions in a phoneme model for the new speaker can be obtained from those of the old speaker by a linear transformation:

$$\pi' = \pi M_{\text{phone}}. \tag{1}$$

Our principal contribution is an investigation of an analog of this idea for Gaussian models. Namely, we assume that the means for the new speaker can be obtained from those of the old by translation:

$$\mu' = \mu + v_{\text{phone}}. \tag{2}$$

(Note that these transformations, while similar in spirit, differ substantially in complexity; the vector v is specified by 15 parameters, whereas the number of entries in the matrix M is the square of the codebook size.) We also give a method for estimating covariance matrices for the new speaker's phoneme models from a limited amount of adaptation data. We tested on a single speaker in developing the adaptation algorithm; results for different speakers are presented in the final section.

II. Adapting the Mean Vectors

We first show how the translation vectors v in (2) can be estimated from a small amount of adaptation data by a variant of the Baum–Welch algorithm. Thus, we start out with an initial estimate for v—the zero vector is the obvious choice—and we iteratively apply the reestimation formula derived below. Each iteration increases the likelihood of the adaptation data, and the procedure terminates when the likelihood has converged.

Manuscript received October 18, 1988; revised January 23, 1990. Recommended for acceptance by R. De Mori. This work was supported by the Natural Science and Engineering Research Council of Canada. This paper was presented in part at the IEEE Workshop on Speech Recognition, May 1988.

P. Kenny is with the INRS—Télécommunications, Montréal, P.Q., Canada H3E 1H6.

M. Lennig and P. Mermelstein are with the INRS—Télécommunications, Montréal, P.Q., Canada H3E 1H6 and Bell-Northern Research, Montréal, P.Q., Canada.

IEEE Log Number 9036259.