

FUNDAMENTALS OF GEOMETRIC LEARNING

by
Stephen M. Omohundro

February 1988



REPORT NO. UIUCDCS-R-88-1408

FUNDAMENTALS OF GEOMETRIC LEARNING

by

Stephen M. Omohundro

February 1988

DEPARTMENT OF COMPUTER SCIENCE
1304 W. SPRINGFIELD AVENUE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, IL 61801

Fundamentals of Geometric Learning

Stephen M. Omohundro

*Department of Computer Science and
Center for Complex Systems Research,
University of Illinois at Urbana-Champaign,
508 South Sixth Street, Champaign, IL 61820, USA.
217-244-4250*

January 15, 1988

Abstract.

A large number of important technological problems in machine vision, speech, computer graphics, and robot control would be solved by systems capable of learning in geometric domains. We present eight fundamental geometric induction tasks in the context of a simple system for vision-based robotics and discuss their analogs in several other domains. We then present an efficient geometric data structure and learning algorithm for implementing these tasks and discuss several theorems analyzing their performance.

Problem area: Geometric learning, General approach: Computational geometry data structures, Evaluation criterion: theoretical analysis.

1. Introduction

It is now almost an AI *cliché* that knowledge is power and that lots of it is needed to build intelligent systems. We would clearly like our systems to acquire this knowledge through learning, especially in domains where human introspection is difficult. Many of today's most problematic domains, such as speech, vision, and robotics, lie at the interface between an intelligent system and the real world. These domains require knowledge of a geometric nature which is notoriously difficult to encode in propositional logic or even in the natural language that experts must use in introspection. We have found several common geometric learning structures arising in a variety of different application areas which appear to provide the foundations for a general theory of induction and learning in geometric domains. The complete description of these primitives is in terms of the well developed formalism of differential geometry, but recent developments in computational geometry allow us to give these mathematical structures a highly efficient computational form.

There are an enormous number of situations in which geometric learning is relevant. To fix ideas in this short paper, we will describe eight successively more difficult geometric induction tasks in the context of a simple vision-based robot manipulator. This setting has the rudiments of many of the important problems in geometric learning and is currently being studied in our laboratory in the context of neural network learning[9]. After presenting the tasks in this context, we discuss a variety of other situations in which the same kind of geometric induction arises. We then describe computationally and statistically efficient algorithms for implementing these tasks based on a unified data structure and present several theorems analyzing their performance.

2. Eight geometric learning tasks in Vision-based robotics.

The system we will discuss receives a visual image of a robot manipulator which is under its control. The system does not have an *a priori* model of the kinematics of the arm or the imaging geometry of the camera and must build up an internal model by learning from experience. The system has a visual preprocessing stage which can reliably extract the x and y coordinates in the image of a set of distinguishable features of the arm which are sufficient to unambiguously identify the kinematic state of the arm. The visual state is then a point in an N dimensional space V where N is twice the number of extracted features. The kinematic state of the arm is a point in an M dimensional space K . In our current laboratory setup, $N = 12$ and $M = 3$. As is usual, the visual state contains much redundant information and not every point in V is physically realizable.

Task 1. Decide whether a hypothetical visual state in V is physically possible or not. Geometric constraints force the actual visual states to lie on a submanifold S (i.e. a smoothly embedded surface [1], here with dimension typically much less than N) of V . The inductive nature of this task arises because we would like the system to sometimes respond positively to points which it has never seen. This task is that of generalizing a set of data points to a whole submanifold which naturally contains them. Notice that this submanifold S is isomorphic (diffeomorphic) to the arm's kinematic state space K . In many situations an analogous state space is not initially known (often even its dimension is unknown) and this task can create it.

Task 2. Given a kinematic state for the arm (e.g. a specification of the joint angles) predict the corresponding visual state. This is useful for predicting the visual effect of a manipulator action. This is the problem of learning a smooth nonlinear mapping from K to V given as data a set of point pairs from $K \times V$.

Task 3. Given a legal visual state (i.e. a point in V which lies on S), estimate the corresponding kinematic state in K . This is useful for planning a manipulator action

which will result in a desired visual state. If we restrict attention to the submanifold of possible visual states S , this task is to learn the mapping which is the inverse of the mapping in task 2. We will describe a framework in which this kind of partial mapping (i.e. undefined on parts of the domain) is natural to represent.

Task 4. Given only partial information about the visual state (e.g. the locations of only a subset of the set of features), determine the complete visual state (and using task 3 the kinematic state as well). This task is only possible because of the redundancy of the visual state. This kind of completion task suggests that we view the induced submanifold of states S in task 1 as a constraint surface in the visual state space V . Given partial visual information and this constraint we can sometimes obtain complete information.

Task 5. Alternatively we may have partial kinematic information as well as partial visual information and wish to extract the complete visual and kinematic states. For example some of the joint angles and some of the feature locations may be specified. This task is a model for the general task of combining different sources of information, generalizing on the basis of their past performance.

Task 6. Given a set of points along a path through the visual space generate points interpolating the rest of the path. Knowing the induced submanifold S from task 1, we can often do a much better job than direct interpolation in the visual space V . In the robot context, this direct interpolation would consist of moving the arm features along straight lines joining the specified locations. If a limb moves in a circular arc, straightforward interpolation would predict motion along a polygonal path (violating the rigidity of the limb), whereas with knowledge of the visual submanifold S it would predict a circular arc. This task requires the system to see many fewer points to achieve a given level of predictive accuracy than if it had not induced S .

The final two tasks have to do with parameterized versions of the above tasks. We will phrase these in terms of a camera which occasionally changes its location and orientation

relative to the robot (a realistic occurrence in the laboratory). For simplicity we assume that the arm moves about for a long time before the camera is moved, and that the system is told whenever the shift occurs (ultimately a system should be able to determine this as well). A camera state may be specified by a translation and a rotation from an initial state and so the camera state space C may be identified with the three dimensional Euclidean group.

Task 7. After having learned the arm visual submanifold S in one or more camera states, the system should be able to induce the submanifold in a new camera state given only one or a few sample points in it. This kind of task is very important in many circumstances because it allows us to use information gained at one parameter value in making predictions and inferences at another value.

Task 8. In this task we ask the system to induce the dimension and form of the set of camera states C (i.e. the parameter space for the set of visual submanifolds). This should be done in such a way that given a visual state in V , the system can determine the value of the camera parameter in C and given a parameter value it can determine the whole submanifold of realizable visual states S for the arm. Notice that this task requires the system to induce a whole new state space without any a priori knowledge of its form.

3. Other settings for geometric learning

Tasks quite analogous to those listed above arise in a variety of situations in machine vision, computer graphics, robotics, and speech recognition and generation. When learning is involved, there is an intricate interplay between recognition, generation, and compression tasks. The best parameterization for understanding a visual scene often consists of the information needed to generate it with a graphics system; a realistic graphics system should be able to use visual input in addition to a priori models. We can get a large

compression of the data required to specify an instance of a domain if the system has learned the range of possible structures.

Task 1 has the essence of extracting a parameterized model from experience, task 2 represents learning a deterministic mapping from one domain to another (often a generation task like graphics or speech generation), task 3 is its inverse (often a recognition task), task 4 represents pattern completion (such as image enhancement or reconstruction of occluded data), task 5 models combining partial information from different sources (eg. multi-sensor fusion or combining the various "shape-from" processes). Task 6 shows compression in the amount of information needed to store or observe constrained sequences (eg. speech compression or video "in-betweening"). Tasks 7 and 8 have the rudiments of learning to apply the same map or constraint in different situations often with great savings in storage and in number of examples needed (eg. accounting for room lighting, observer motion or room acoustics are processes which apply over a wide variety of tasks).

Many opportunities for geometric induction arise in recognition and prediction tasks. Machine vision uses a large number of physical constraints and relationships which are currently mostly input by hand but which could be learned using the techniques for the first four tasks above. One example of this kind of learning appears in [8] in which learning is applied to the problem of separating illumination color from scene color. A particularly important problem in vision is combining different sources of information influencing an estimate. For example one wants to estimate object motion from two temporally successive images or to estimate disparity from two images from different viewpoints. In both of these processes one may either try to match features and see how they moved, or measure intensity or color change at the same location in the two images and compare with the gradient there [2]. These two partial sources of information need to be combined in the final estimate. [6] uses task 2 in predicting the future behavior of a physical system based on past observations.

The generative processes of computer graphics, robotics and speech generation also have many similarities. Task 7 is a particularly useful form of induction which arises whenever we have a parameterized family of entities with a range of behavior and would like to generalize the behaviors seen at one parameter value to other values. As an example from computer graphics, consider the problem of the realistic representation of human faces. Present day techniques require detailed muscle models and painstaking input of a three-dimensional polygonal approximation of real faces [15]. The results, while very impressive, would not be mistaken for real faces and the whole procedure must be repeated to represent different people. One would really like to just show the system a new face and have it induce the complete set of facial expressions. Induction of the "face space" of possible states for a given individual corresponds to task 1 and applying learned expressions to a new face corresponds to task 7. Exactly the same induced structures are required for the dual tasks of face recognition (independent of expression) and facial expression recognition (independent of face). Similarly, we would like systems which generate and recognize realistic human movement over a range of body types. In the auditory domain we have the dual tasks of speech recognition independent of speaker and speaker recognition independent of what is spoken. We would like a system which could generate speech as spoken by different speakers (with different accents or dialects) using training on only a small portion of the corpus of phrases.

4. Algorithms and data structures for geometric learning

Proposed algorithms for solving the eight tasks presented above can be characterized by two performance metrics representing their computational and statistical efficiencies. By computational efficiency we mean the expected algorithmic complexity for a given number of samples. By statistical efficiency we mean how few samples are needed for a given expected accuracy of induction. As in the theoretical setting proposed by Valiant

[14] for learning in discrete contexts, we would like to develop algorithms with good expected performance with arbitrary underlying probability distributions for the samples. This type of analysis is based on recent developments in nonparametric statistics (see for example [4]).

4.1 Learning Nonlinear Mappings: Task 2

In previous work [12] we analyzed the general setting for task 2. We proposed a data structure and learning algorithm which were shown to implement this task much more efficiently than comparable neural network proposals. In that work the setting was restricted to learning nonlinear mappings from a domain space to a range space given a set of input/output pairs. This is essentially a problem of multidimensional interpolation. High accuracy may be achieved at greater computational expense and complexity by using higher order piecewise polynomial representations (such as splines) but here we will restrict attention to piecewise linear representations. The underlying computational ideas come from recent developments in computational geometry (see [5]).

It is perhaps simplest to picture the graph of the function which is a submanifold in the product of the domain and range spaces (in the robot setting the mapping is from K to V and its graph is a submanifold in the product space $K \times V$). We are given as data a set of sample points lying on this graph and would like to use them to approximate the entire graph. To linearly interpolate, we need to choose a triangulation of the domain space (i.e. a decomposition into M -dimensional tetrahedra) with vertices given by the (projection into the domain of the) sample points. If we choose such a triangulation, then the estimated value of the mapping on a new point in the domain is found by determining which tetrahedron the point lies in and linearly interpolating the mapping values at its vertices to the point of interest.

To carry out this procedure we must choose a triangulation of the domain and store

the samples in such a way that the appropriate tetrahedron may be retrieved given a desired point in the domain. Under certain conditions, the Delaunay triangulation [11] is an optimal triangulation for piecewise linear approximation. If the source space is M dimensional, a set of $M + 1$ sample points form the vertices of a tetrahedron in the Delaunay triangulation if and only if the $M - 1$ sphere which they determine doesn't contain any other sample points. If we structure the sample points using a $k - d$ tree we may use a variant of the analysis in [7] to show that we may retrieve the vertices of the tetrahedron containing a new point in an expected time which is only logarithmic in the number of sample points.

A $k - d$ tree is a binary tree which represents a hierarchical decomposition of a space into hyper-rectangles. Each node of the tree represents a hyper-rectangular region of the space and its two children are formed by splitting this hyper-rectangle along one of its dimensions at a specified value. The nodes of the tree need only store the index of the split dimension and the value at which it is split. The leaves of the tree together partition the space. To determine which leaf hyper-rectangle a given point lies in, we need only make one pass from root to leaf, choosing to go left or right according to whether the splitting coordinate of the point is less than or greater than the stored value in the node. If we form the tree from a set of sample points by always cutting on the most spread out dimension and always choosing the cutting value to split the sample points in half, then the depth of the tree will be logarithmic in the number of points. If the sample points are drawn from an arbitrary smooth probability distribution that asymptotically this procedure produces a leaf partition in which the expected shape of a leaf's hyper-rectangle is cubical and all leaves are expected to contain the same amount of probability (so leaves are large in low density regions and small in high density regions). [7] shows that by using a branch and bound procedure, the n nearest neighbors of a given point can be found in only logarithmic expected time. Similarly, one can show that the desired

vertices of the Delaunay tetrahedron can also be found in only logarithmic time.

In [12] we gave explicit estimates for the expected number of samples for a given accuracy of approximation in terms of bounds on the second derivatives of the mapping. Using the $k - d$ tree representation for the data samples and the piecewise linear interpolation scheme described above for the induction, one achieves efficient and relatively simple to implement geometric mapping induction. The approach is easily adapted to noisy data by using least squares linear approximators instead of interpolants. However, this structure does not deal well with partial information queries as required in the other tasks above. We therefore extend this structure to representing arbitrary submanifolds instead of just graphs of mappings.

4.2 Learning and Representing Submanifolds

Many of the tasks above can be formulated as learning a smooth submanifold in a space from a representative set of sample points lying in the submanifold and representing it in such a way as to efficiently support queries represented as intersections with submanifolds. Two submanifolds intersect transversally at a point if together their tangent spaces span the tangent space to the embedding space. It is an important result of differential topology that non-transversal intersections may be made transversal by arbitrarily small perturbations while transversal intersections remain so under small enough perturbations [3]. In some settings this leads to a principle of “structurally stability” which says that only transversal intersections are likely to occur in real situations.

Task 1 directly asks us to form the submanifold S from samples in V . Task 2 may be formulated in these terms as learning the graph of the kinematic to visual mapping in $K \times V$ and treating function evaluation as intersection with the submanifold determined by fixing a point in K and letting the V component be arbitrary. Notice that function inversion and composition may be formulated similarly (the graph of the composition

may be obtained from the intersection of the product of the graphs of the two functions with the linear submanifold constraining the point in the range of the first function to be equal to the point in the domain of the second). Task 3 asks for the intersection of this same graph with a linear submanifold given by fixing a point in V and letting the K component vary. Only realizable visual states will give rise to an intersection. Both this task and task 1 suggest the desirability of finding the point of closest approach of two submanifolds if they don't intersect. Tasks 4 and 5 are again intersections with linear submanifolds in this formulation. Task 6 may be accomplished by projecting the linearly interpolated path in V onto S and generalizes to families with higher dimensional parameterizations. Tasks 7 and 8 are accomplished by learning a transformation from V to itself which interpolates the given samples and generalizing the original submanifold S to its image under these transformations. The expected performance of each of these task can be analyzed in detail and will appear elsewhere [13].

Let us now describe an efficient data structure and query algorithm to support these operations. As above we would like to triangularize the submanifold but now we don't have the domain space to parameterize the submanifold and so cannot directly use the Delaunay construction. If there are sufficiently many samples measured in terms dependent on the curvature of the submanifold, then the samples in the neighborhood of a point approximate samples drawn from a linear submanifold. We can use this to determine with extremely high probability the dimension and approximate tangent space to the submanifold. The basic idea is to consider clouds of successively nearest neighbors (quickly obtainable using a k - d tree) and to look at the eigenvalues of the matrix of second moments. If enough samples have been drawn, the tangent space directions correspond to eigenvectors with rapidly growing eigenvalues compared to the transversal directions (see [13] for details). Once the dimension and tangent space are determined, we may use the Delaunay construction by projecting out the normal to the tangent space.

Once we have triangularized a submanifold we wish to represent it with a data structure that efficiently supports the queries listed above. We again use a binary tree in which the nodes correspond to hyper-rectangular regions of the ambient space, but now the nodes at a given level will not partition the space but will completely contain the submanifold. As we move down the tree the set represented will better and better approximate the submanifold. Each node in the tree explicitly stores the extents of the corresponding hyper-rectangle along each dimension. To create the children of a node, we again cut it along its longest dimension at the median but now shrink the resulting two pieces as small as they can be while still containing the same chunk of the triangulated submanifold. For example, if the submanifold is a curve, it will be represented as the diagonals a string of hyper-rectangles which get smaller and smaller as we move down the tree.

To find the submanifold tetrahedron containing a point we can proceed as with the k -d tree, starting at the root and working down to a leaf going left or right according to which child hyper-rectangle contains the point. Now, however, we can also go down the tree with only partial information. If only some of the coordinates are specified (but at least as many as the dimension of the submanifold so that a set of points rather than surfaces are defined) we may proceed down the tree pursuing all nodes (using depth first search for example) which intersect the specified query surface. This test is easy since we merely check if each specified coordinate is in the range of the corresponding hyper-rectangle. Because the system sometimes has to continue down both children of a node, one might worry that a query might take linear expected time. Clearly it cannot take less than the number of returned intersections n , but one can show that asymptotically it takes an additional time only logarithmic in the number of samples. This may be proven independently of the sample probability distribution as long as it is non-vanishing over the entire submanifold (see [13]).

To intersect two transversal submanifolds with complementary dimensions, we traverse both trees keeping track of only those nodes which intersect a node of the other tree. Again the expected search time is the number of returned samples plus a term logarithmic in the number of samples. The same structure supports closest approach queries with the same efficiency. For example, to find the point on a submanifold which is closest to a given point, we apply a branch and bound procedure, working our way down the best path and pruning off any paths inferior to a known solution. Replacing the distance estimator allows us to use linear submanifolds or a variety of other query sets. These queries are useful in best or closest match situations. Again we achieve logarithmic expected performance.

5. Conclusions

We have discussed a number of induction problems with a quite different flavor than those mostly studied in the machine learning literature [10]. In some ways learning in geometrical contexts is easier than in general contexts because the geometry imposes constraints which may be used to inform induction. Perhaps the most fundamental constraining principle is provided by continuity: "Unless explicitly known to be otherwise, nearby perceptions should be taken to correspond to nearby states." It appears that this kind of learning will lead to systems with important applications in a variety of disciplines. We have presented some fairly easily implemented data structures and algorithms which efficiently implement a variety of fundamental geometric learning tasks. In a complete system we envision these operations being controlled and modified by a symbolic learning system. Integrating these approaches is an interesting future task.

6. Acknowledgements

I would like to thank Doyme Farmer, Darrell Hougen, Bartlett Mel, and Norm Packard for discussions of these ideas.

References

- [1] R. Abraham, J. Marsden, and T. Ratiu, *Manifolds, Tensor Analysis, and Applications*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts (1983).
- [2] Peter Blicher and Stephen M. Omohundro, "Unique Recovery of Motion and Optic Flow via Lie Algebras", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (1985).
- [3] Th. Bröcker and K. Jänich, *Introduction to Differential Topology*, tr. by C. B. and M. J. Thomas, Cambridge University Press, Cambridge (1982).
- [4] Luc Devroye, *Lecture Notes on Bucket Algorithms*, Birkhäuser Boston Inc., Boston (1986).
- [5] H. Edelsbrunner and J. van Leewen, "Multidimensional data structures and algorithms. A bibliography," IIG, Technische Univ. Graz, Austria, Rep. 104 (1983).
- [6] Doyne Farmer and John Sidorwich, "Predicting Chaotic Time Series," in preparation.
- [7] Jerome H. Friedman, Jon Louis Bently, and Raphael Ari Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software*, 3:3 (1977) 209-226.
- [8] A. Hurlbert and T. Poggio, "Learning a Color Algorithm from Examples", Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo No. 909.
- [9] B.W. Mel, "MURPHY: A robot that learns by doing", in *AIP Proceedings of the IEEE Conf. on Neural Information Processing Systems*, Denver, CO, November 1987.
- [10] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach, Vols. I and II*, Morgan Kaufmann Publishers, Inc., Los Altos, CA (1986).
- [11] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry. An Introduction*. (Springer-Verlag, 1985).

- [12] Stephen M. Omohundro, "Efficient Algorithms with Neural Network Behavior", *Complex Systems*, 1 (1987) 273-347.
- [13] Stephen M. Omohundro, "Submanifold Intersection for Geometric Learning", in preparation.
- [14] L. G. Valiant, "A Theory of the Learnable," *Communications of the ACM*, 27:11 (1984) 1134-1142.
- [15] Keith Waters, "A Muscle Model for Animating Three-Dimensional Facial Expression", *Computer Graphics*, 21: 4 (1987) 17-24.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-88-1408	2.	3. Recipient's Accession No.
	4. Title and Subtitle FUNDAMENTALS OF GEOMETRIC LEARNING		5. Report Date February 1988
7. Author(s) Stephen M. Omohundro		6.	
9. Performing Organization Name and Address Department of Computer Science 1304 W. Springfield Urbana, Illinois 61801		8. Performing Organization Repr. No.	
12. Sponsoring Organization Name and Address		10. Project/Task/Work Unit No.	
		11. Contract/Grant No.	
15. Supplementary Notes		13. Type of Report & Period Covered Technical	
		14.	
16. Abstracts A large number of important technological problems in machine vision, speech, computer graphics, and robot control would be solved by systems capable of learning in geometric domains. We present eight fundamental geometric induction tasks in the context of a simple system for vision-based robotics and discuss their analogs in several other domains. We then present an efficient geometric data structure and learning algorithm for implementing these tasks and discuss several theorems analyzing their performance. Problem area: Geometric learning, General approach: Computational geometry data structures, Evaluation criterion: theoretical analysis.			
17. Key Words and Document Analysis. 17a. Descriptors geometric learning robotics vision computational geometry submanifold intersection			
17b. Identifiers/Open-Ended Terms			
17c. COSATI Field/Group			
18. Availability Statement unlimited		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 19
		20. Security Class (This Page) UNCLASSIFIED	22. Price