51. G. F. Oster and C. A. Desoer, Tellegen's theorem and thermodynamic inequalities, *J. Theor. Biol.* **32**, 1971, 219–241.
52. G. F. Oster, A. Perelson, and A. Katchalsky, Network thermodynamics, *Nature* **234**, 1971, 393–399.
53. T. Poggio and C. Koch, Analog Networks: A New Approach to Neuronal Computation, Artificial Intelligence Lab. Memo No. 783, MIT, Cambridge, Mass., 1984.
54. T. Poggio and V. Torre, Ill-posed Problems and Regularization Analysis in Early Vision, Artificial Intelligence Lab. Memo No. 773, MIT, Cambridge, Mass., April 1984.
55. T. Poggio, H. Voorhees, and A. Yuille, Regularizing Edge Detection, Artificial Intelligence Lab. Memo No. 776, MIT, Cambridge, Mass., 1984.
56. I. Prigogine, *Thermodynamics of Irreversible Processes*, Wiley–Interscience, New York, 1967.
57. C. H. Reinsch, Smoothing by spline functions, *Numer. Math.* **10**, 1967, 177–183.
58. F. O. Schmitt, P. Dev, and B. H. Smith, Electrotonic processing of information in brain cells, *Science* (Washington, D.C.), **193**, 1976, 114–120.
59. B. G. Schunck and B. K. P. Horn, Constraints on optical flow computation, in *Proc. IEEE Conf. Pattern Recognition and Image Processing*, pp. 205–210, 1981.
60. D. Terzopoulos, Multiresolution Computation of Visible-Surface Representations, Ph.D. thesis, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, Mass., 1984.
61. D. Terzopoulos, Multilevel computational processes for visual surface reconstruction, *Comput. Vision Graphics, Image Process.* **24**, 1983, 52–96.
62. D. Terzopoulos, Multilevel Reconstruction of Visual Surfaces: Variational Principles and Finite Element Representations, Artificial Intelligence Lab. Memo 671, MIT; in *Multiresolution Image Processing and Analysis* (A. Rosenfeld, Ed.), pp. 237–310, Springer-Verlag, New York/Berlin, 1984.
63. D. Terzopoulos, Integrating visual information from multiple sources for the cooperative computation of surface shape, in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, (A. Pentland, Ed.), Ablex, Norwood, N.J., 1985.
64. A. N. Tikhonov, Solution of incorrectly formulated problems and the regularization method, *Soviet Math. Dokl.* 4, 1963, 1035–1038.
65. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Winston & Sons, Washington, D.C., 1977.
66. V. Torre and T. Poggio, On Edge Detection, Artificial Intelligence Lab. Memo No. 768, MIT, Cambridge, Mass., March 1984.
67. S. Ullman, Filling in the gaps: The shape of subjective contours and a model for their generation, *Biol. Cybern.* **25**, 1976, 1–6.
68. S. Ullman, The interpretation of structure from motion, *Proc. R. Soc. London B* **203**, 1979, 405–426.
69. G. Wahba, Ill-Posed Problems: Numerical and Statistical Methods for Mildly, Moderately, and Severely Ill-Posed Problems with Noisy Data, Tech. Report No. 595, Univ. of Wisconsin, Madison, 1980.
70. A. Yuille, The Smoothest Velocity Field and Token Matching Schemes, Artificial Intelligence Lab. Memo No. 724, MIT, Cambridge, Mass., August 1983.

Codon Constraints on Closed 2D Shapes

WHITMAN RICHARDS AND DONALD D. HOFFMAN*

Natural Computation Group, Massachusetts Institute of Technology, Cambridge, Massachusetts

Received September 27, 1984

Codons are simple primitives for describing plane curves. They thus are primarily image-based descriptors. Yet they have the power to capture important information about the 3D world, such as making part boundaries explicit. The codon description is highly redundant (useful for error-correction). This redundancy can be viewed as a constraint on the number of possible codon strings. For smooth closed strings that represent the bounding contour (silhouette) of many smooth 3D objects, the constraints are so strong that sequences containing 6 elements yield only 33 generic shapes as compared with a possible number of 15,625 combinations. © 1985 Academic Press, Inc.

1. INTRODUCTION

An important task for object recognition is the description of the shape of a bounding contour, such as a silhouette that outlines an object. Although recognition need require only partial segments of such contours, the internal canonical description, against which the image contour is compared, is very likely a closed ring. Our concept of most "objects" should lead us to expect such a closed contour. The description of closed, 2D contours thus is an important ingredient of a system for object recognition. First we present such a scheme, described in more detail elsewhere [3, 4] and then show how the scheme leads to a hierarchical taxonomy of closed, 2D shapes.

2. THE REPRESENTATION

When we view shapes such as those in Fig. 1, we immediately see the ellipse and square as being "simpler" (in some psychological sense) than the lemniscate or epicycloid. Why? If we were to "measure" the simplicity of a shape contour by the degree of its polynomial equation, then the cardioid in the middle would have the simplest form, and the square the most complex, being the highest order polynomial. Clearly a polynomial representation seems quite inappropriate for our visual system, because it does not make explicit the meaningful properties of the shapes.

If we asked a child why the ellipse is "simpler" than the lemniscate, he would probably reply "because the latter has two parts, whereas the ellipse has only one." This simple observation is the basis for our representation for shapes: namely a shape should be described in terms of its natural "parts." Fortunately, the rule for finding "parts" is conceptually simple, for when 3D entities are joined to create complex objects, then concavities almost always are created at the join, as indicated by the small arrows in Fig. 2.

This regularity of natural objects follows a principle of transversality treated more fully elsewhere [5]. In the silhouette, these concavities appear as cusps, or as places

*D. D. Hoffman is now at the University of California, Irvine.

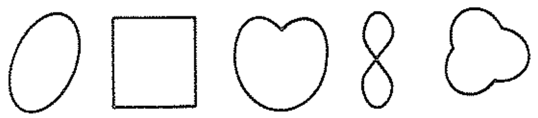


FIG. 1. An ellipse, square, cardioid, lemniscate, and epicycloid. The cardioid has the simplest (lowest order) equation; the square is the highest order polynomial equation.

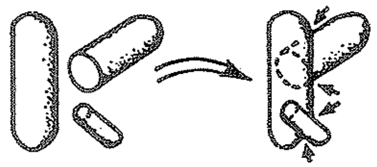


FIG. 2. Joining parts generally provides concavities in the resulting silhouette.

of maximum negative curvature. Natural parts thus lie between concave cusps. In Fig. 1, the rule specifies that the ellipse and the square have no parts, whereas the lemniscate has two and the epicycloid has three. (The cardioid can not be broken simply into two parts, hence must be "simpler" than the two figures on its right.) Our first rule for representing (2D) shapes is thus as follows:

Segment a curve at concave cusps (or minima of negative curvature) in order to break the shape into its "parts."

3. PART DESCRIPTORS: CODONS

Having now broken a curve into "parts" our next task is to describe the part. Again, we wish that our description capture some natural property of shapes, rather than an arbitrary mathematical formula, such as a polynomial equation. For example, at some stage in our representation, we would like to know whether it is round or polygonal. But even before such descriptors, is there a still simpler, more abstract, representation? Perhaps first we should represent the "sides" of the part, or its "top." As a step in this direction, we propose a very primitive representation based upon the singular points of curvature, namely the maxima, minima, and zeroes of curvature along the curve. An important property of these descriptors is that their ordinal relations remain invariant under translations, rotations, and dilations. Thus, regardless of the 3D orientation and size of a part to its whole, a relation between these descriptors is preserved in the 2D image. This property follows because the inflection of a 3D curve is preserved under projection, guaranteeing that at least the ordinal relations between minima, maxima, and zeroes of curvature will be preserved under projection. Our scheme thus provides a very primitive representation for a part, simply in terms of the ordinal relations of the extrema of curvature. This approach yields six different basic primitive shapes, or codons (see Fig. 4).

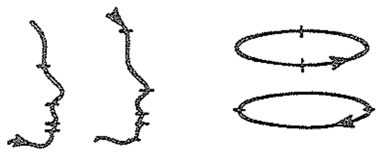


FIG. 3. Minima of curvature are indicated in slashes. Arrows indicate direction of traversal of curve. "Figure" is taken to be to the left of the direction of traversal.

In order to define the codon types, it is first necessary to define maxima and minima of curvature. These definitions require that a convention be adopted for the sign of curvature. Consider Fig. 3. There are two directions along which the profile of the face may be traversed. In the upward direction (left) the minima of curvature (slashes) correspond to the points where the curve rotates at the greatest rate in the clockwise direction. If the same curve is traversed in the opposite direction, however, then the maxima and minima reverse. Our convention thus places "figure" to the left of the direction of traversal. When the figure is on the left, then the profile indeed looks like a face because the minima of curvature divide the curve into the natural parts—namely forehead, nose, mouth, and chin. (Note that the opposite view yields the "vase" of Rubin's famous figure–ground illusion observed as early as 1819 by Turton [14].) Thus, knowing which side is the figure determines the choice of orientation on a curve, or, conversely, choosing an orientation determines which side is the figure by convention. Minima are then typically associated with the concavities of the figure, whereas maxima are convexities.

To define our basic primitive codons, we first note that all curve segments lying between minima of curvature must have zero, one, or two points of zero curvature. If there are no zeroes (i.e., inflections), then the segment is designated as a type 0 codon (see Fig. 4). Those with two zeroes are called type 2 codons. If a segment has exactly one zero, then the zero may be encountered either before (type 1^-) or after (type 1^+) reaching the maximum point of the segment during traversal in the chosen orientation.

The type 0 codons may be further subdivided into 0^+, 0^- and (∞) to yield six basic codon types. Consider Fig. 3 once again. Note that as the ellipse is traversed in different directions, the minima of curvature change as expected. In the lower ellipse, which corresponds to a "hole" with figure outside, the minima have negative curvature, because the direction of rotation is clockwise. (Thus, the slashes suggest a part boundary by our rule, which will be repaired later when we discuss "holes.") In the upper ellipse, however, the minima have positive curvature (the rotation is always counterclockwise). Thus, the type 0 codon can be subdivided into 0^+ and 0^- with the superscript indicating the sign of curvature. Note that the 0^- codon can constitute a part boundary, whereas the type 0^+ codon must appear only as a shape

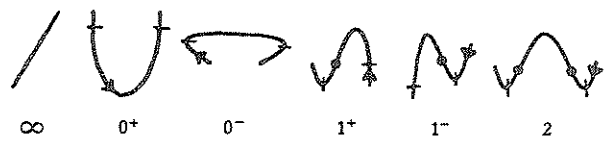


FIG. 4. The primitive codon types. Zeroes of curvature are indicated by dots, minima by slashes. The straight line (∞) is a degenerate case included for completeness, although it is not treated in the text.

descriptor. Finally, the type ∞ codon simply is the degenerate case of a straight line that has an ∞ of zeroes.

4. CONSTRAINTS ON SMOOTH CODON STRINGS

Not all sequences of codons are possible if the curve is smooth. Referring to Fig. 4 once again, note that a 1^- can not follow a 1^- codon unless a cusp is allowed. Similarly, a 1^+ can not follow a 1^+, because if such a join is attempted either a cusp will be created or, if the curve is indeed smooth, the 1^+ codon would have to be transformed into a type 2. To specify all legal smooth codon strings, we will first enumerate all pairs, and then show what pair substitutions are legal for one element in a sequence of pairs, thereby creating all possible triples.

Define the "tail" of a codon as the region about the first minima encountered when traversing the curve. The "head" of the codon is the subsequent minima. A smooth string of two codons is then allowable only if the head of the first codon has the same sign of curvature as the tail of the second codon in the string. Table 1a shows the sign of curvature for each codon type (excluding the degenerate type ∞). Table 1b is constructed simply by multiplying the sign of curvature of the "head" of the first codon (the left-most column) by the "tail" of the second (given in the second row). If the signs agree, then a (+) is entered, indicating a legal smooth join, otherwise the (−) product is an illegal smooth join. Thus, for these five codons, there are 13 legal joins out of a possible 25 combinations.

To enumerate the possible codon triples for a smooth contour, we now require that the curvature of both the head and tail of a middle codon match the tail of its successor or the head of its predecessor in the string. Table 2 provides the signs of the heads (and tails) of the legal pairs (left column) which must match the tail (or head) of the third codon in the string. For each column under the third codon, the legal triplets are indicated by a (+). If two pluses appear in brackets, then the third codon can either precede or follow the pair. Consider first the case where the third codon follows the pair (these are given in the columns headed "tail"). There are 34 legal smooth triplets of this type. Symmetry arguments yield a similar number of triplets when the third codon precedes the pairs (these are given in the columns headed "head"). Thus, there are only 34 legal codon triplets out of a possible $5^3 = 125$.

TABLE 1
Codon Signatures (a) and Legal Smooth Codon Pairs (b)

a			b	LEGAL CODON PAIRS				
CODON SIGNATURES				2nd CODON (tail)				
CODON	TAIL	HEAD	1st CODON (head)	$0^-(-)$	$0^+(+)$	$1^-(-)$	$1^+(+)$	$2(-)$
0^-	−	−	$0^-(-)$	+	−	+	−	+
0^+	+	+	$0^+(+)$	−	+	−	+	−
1^-	−	+	$1^-(+)$	−	+	−	+	−
1^+	+	−	$1^+(-)$	+	−	+	−	+
2	−	−	$2\ (-)$	+	−	+	−	+

TABLE 2
Legal Smooth Codon Triplets

LEGAL CODON PAIRS			THIRD CODON									
			0^-		0^+		1^-		1^{+-}		2	
			−	−	+	+	−	+	+	−	−	−
	TAIL	HEAD	TAIL	HEAD	TAIL	HEAD	TAIL	HEAD	TAIL	HEAD	TAIL	HEAD
0^-0^-	−	−	[+	+]			+			+	[+	+]
0^-1^-	−	+		+	+				[+	+]		+
0^-2	−	−	[+	+]			+			+	[+	+]
0^+0^+	+	+			[+	+]		+	+			
0^+1^+	+	−	+			+	[+	+]			+	
1^-0^+	−	+		+	+				[+	+]		+
1^-1^+	−	−	[+	+]			+			+	[+	+]
1^+0^-	+	−	+			+	[+	+]			+	
1^+1^-	+	+			[+	+]		+	+			
1^+2	+	−	+			+	[+	+]			+	
$2\ 0^-$	−	−	[+	+]			+			+	[+	+]
$2\ 1^-$	−	+		+	+				[+	+]		+
$2\ 2$	−	−	[+	+]			+			+	[+	+]
NUMBER OF LEGAL PAIR SUBSTITUTIONS			5		2		3		3		5	
NUMBER OF PAIR SUBSTITUTIONS			9		4		6		6		9	

Note. The third codon can either follow or precede the pair. A (+) indicates a proper join. Because of symmetry, there are an equal number of total pluses in the head and tail columns.

Of the 34 possible triplets, there are 18 cases where the same codon can be attached to either end of the codon pair, indicated by [+ +]. This subset is particularly useful for establishing legal smooth codon strings of order higher than three. For example, consider a codon sequence C_{j-1}, C_j, C_{j+1}. We now desire to expand the sequence. This can be done simply by replacing C_j by C_iC_k, where C_iC_k is one of the pairs that will accept C_j at either end. To extend the string, we thus have the following rewrite rule:

Any individual codon in a smooth string may be replaced by any pair yielding a [+ +] *for the (third) codon in Table 2.*

Thus, an 0^- codon may be replaced by any one of the following pairs: $0^-0^-, 0^-2, 1^-1^+, 20^-, 22$, etc.

To calculate the number of possible quadruplets, we can determine from Table 2 how many times any given codon type appears in the middle portion of the string.

TABLE 3

A Comparison Showing How the Number of Possible Strings of Codon Elements is Reduced as First Smoothness (Open Strings) and Then Closure Are Imposed as Constraints on a Curve

NUMBER OF CODONS	NUMBER OF:		
IN STRING	COMBINATIONS	OPEN STRINGS	CLOSED STRINGS
1	5	5	(2)
2	25	13	3
3	125	34	5
4	625	89	9
5	3,125	233	17
6	15,625	610	33
7	78,125	1,597	65
8	390,625	4,181	129
9	1,953,125	10,946	257
10	9,765,625	28,657	513

Then we can multiply this number of occurrences by the number of possible pair substitutions. For example, the type 0^- codon appears as the middle codon in rows 1, 8, and 11. In each row, the only legal third codons whose tail curvature matches that of the head of 0^- are 0^-, 1^- and 2 (see Table 1). Thus the O^- codon appears as the middle codon in 9 of all the possible triplets. Similarly we find the following number of occurrences of the other codon types in the middle position of the string:

$$0^- = 9; 0^+ = 4; 1^- = 6; 1^+ = 6; 2 = 9. \Sigma = 34$$

Thus the total number of possible smooth codon quadruples will simply be the sum of each of these numbers times the number of possible pair substitutions for each type (next to last row of Table 2), less any duplicate strings. The total substitutions are 134, but the 0^- and 2's duplicate each other, reducing the total quadruples by 45. The answer is the difference of 89 out of a possible 625. Table 3 shows how the number of possible open strings increases with the number of codon elements. In general there will be less than $5 \cdot 7^{(N-1)/2}$ possible smooth strings of N codons compared with 5^N possible (see Appendix II).

5. CLOSED CODONS

Because most objects have closed bounding contours, matching to closed codon sequences is of greater interest for shape recognition than representing open strings. Clearly this constraint on the codon sequence will further reduce the number of allowable smooth shapes. Indeed, this constraint is so powerful that all closed shapes containing up to four codons will be enumerated shortly.

First, let us examine the generating rules. Closed codon pairs can be noted simply by inspecting Table 2. Here, the signs of the heads and tails of the pairs in the first column must agree. The only cases are 0^-0^-, 0^+0^+, 0^-2, 1^-1^+, and 22. These shapes are depicted in Fig. 5, with figure indicated by cross-hatching. Note that there are only three basic outlines, if figure and ground are ignored. Later, we will address

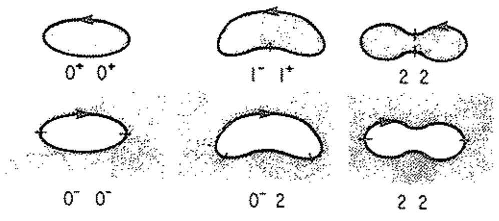


FIG. 5. Legal smooth, closed codon pairs. Figure is indicated by cross hatching. Part boundaries are noted by the slashes.

the problem of indexing identical codon descriptors that have different figure–ground relations (e.g., the 22 pair) and also the observation that the part boundaries (slashes) for the "holes" do not seem appropriate.

From these five legal codon pairs, we can now easily generate the legal closed triples. Simply consider the pair as a string of three elements and then replace each "middle" element by a pair according to Table 2. Thus, the closed pair 1^-1^+ may be rewritten as $1^-1^+1^-$ (or $1^+1^-1^+$) and the 1^+ (or 1^-) can then be replaced by 0^-1^-, 1^-0^+, 21^- (or $0^+1^+, 1^+0^-, 1^+2$). Such substitutions yield a total of 10 different codon triplets, or only 5 different outlines out of a possible 125 if figure and ground are ignored. These shapes are shown in Fig. 6 with their codon labels. Figure 7 shows the result of applying the same rewrite rules to the triples to enumerate all possible codon quadruples. Here there are only 9 outlines out of a total possible combination of 5^4 or 625 sequences. Appendix III shows that the upper bound on the number of closed smooth codons is $2^{N-1} + 1$, where N is the number of codons in the ring. The compression is thus about $2^{N-1}/5^N$, or over 10^4 for a 10-element ring. The reduction comes in part from a propagation of constraints through the closed string, very analogous to the constraint propagation used by Waltz [15] to solve for "blocks-world" shapes using constraints on legal trihedral joins.

6. MIRROR REVERSAL, HOLES, AND FIGURE–GROUND

In Fig. 5, three pairs of codon shapes are possible for a two element ring. For each pair, the outline is the same, but the figure–ground relation is reversed. The situation is similar to a lock–key arrangement, where the shapes in the upper row fit snugly into their "hole" complements in the lower row.

Two problems arise in representing these mirror shapes: First, a lock–key pair may have the same codon description, such as "22" on the right. This is easily rectified by adding an extra index. The second problem is perceptual. For each of

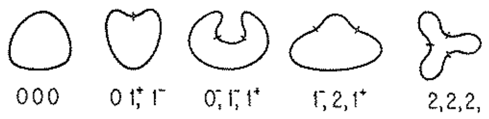


FIG. 6. Legal smooth, closed codon triplets.

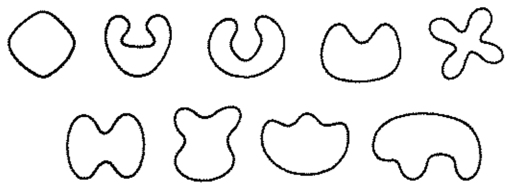


FIG. 7. Legal smooth, closed codon quadruples.

the shapes in the lower row, although "figure" is outside, we strongly prefer the "hole" as the figure. Why?

Some insight into why "holes" are preferred as figure can be obtained by noting that all the shapes in the lower row consist of two parts by our rule. (The slashes indicate the points of maximum negative curvature.) On the other hand, the "ellipse" and "peanut" in the upper row are single entities without parts. Certainly a single entity is "simpler" in some very basic sense than one with parts. Thus, it makes sense to describe the "hole" as the complementary figure if that complement has fewer parts. Note that for the 22 dumbbell shapes, the preference between the hole and its "key" is less strong. In fact, it is not too difficult to regard each "bump" in the hole as a part of the hole, whereas it is almost impossible to view the 0^-0^- elliptical hole as having two parts. Our rule for representing "holes" is thus:

Represent a "hole" by its figure–ground complement if that complement has fewer "parts."

Note that there are many linguistic examples where this rule has been applied. For example, "key-hole," "screw-hole," "oval window," etc., are all descriptions of a hole in terms of the figure–ground complement. (Appendix I shows how a figure–ground complement can be computed easily using a binary representation.)

7. INDEX DEVELOPMENT

The shapes of closed codon rings shown in Figs. 5–7 have all been drawn to preserve symmetry. Furthermore the axes of elongation tend to be straight, and the "parts" are neither too "thin" nor too "thick" for most peoples' taste. What are the rules that underly the canonical representation of these primitive shapes? What regularities of the world are captured here?

Clearly any time a rule is applied to put a codon ring in canonical form, then we have an implicit shape index which is being defaulted. An interesting extension of the abstract codon description is thus to develop indices that are meaningful. The fact that the codon hierarchy is small provides an opportunity for a rational development of such indices—at least for a start. (A more complete and useful set should relate each index to a desired real-world property.)

To give the flavor of this approach, consider the first three primitive codon shapes in Fig. 5: the "ellipse" (= 00), the "peanut" (= 1^-1^+), and the "dumbbell" (= 22). What useful properties can now be assigned to an ellipse outline? We have already mentioned the need to specify figure–ground. What about the orientation of the ellipse, or its eccentricity, or even perhaps its size? These four parameters will

completely specify the elliptical shape relative to a reference frame. However, if we encounter an ovoidal shape having the same 00 codon description, then still another index may be required. For our single, most primitive closed shape we thus have already the following possible indices:

FIGURE–GROUND	0, 1
ORIENTATION (OF AXIS)	ϕ
ECCENTRICITY (ASPECT RATIO)	ρ
SIZE	Σ
SKEW	ν

As we proceed to the next more complex shape, the "peanut," the axis is now curved, and the left and right portions need not have identical size. Two more parameters thus must be added:

RELATION LEFT AXIS TO RIGHT AXIS	α
RELATIVE SIZE, LEFT TO RIGHT "PART"	σ_{lr}

These indices suffice for the dumbbell.

In a similar vein we may proceed up the closed codon hierarchy, adding additional indices. This procedure automatically provides an ordinal order to the indices (whether this order is perceptually appropriate is a separate issue!). Note that left–right "handedness" does not appear until we encounter four element codon rings. For codon rings greater than four or five, the complexity of the shape undoubtedly prohibits practical use. In sum, the codon hierarchy can thus be used to develop an ordered set of indices to the more metrical properties of shapes.[1]

8. MAPPING 3D ⇔ 2D

Codons are descriptors for 2D plane curves, and hence of necessity are an image-based representation. In Marr's [9, 10] terminology, they are part of the data structure of a primal sketch. An important aspect of the motivation for the codon description comes from the nature of the 3D world, however, namely the rule for locating part boundaries at maxima of negative curvature. This rule for partitioning a curve captures the concavity regularity created when two 3D parts are joined (see Fig. 2), as seen in the 2D image. Thus, the presence of a concavity in silhouette is used to infer a part boundary in the 3D world. (See [5] for a more rigorous treatment of this inference.) Can other inferences about the properties of 3D objects also be made from the codon descriptors?

To explore the kinds of inferences possible about 3D shape from 2D contours, we will consider some of the canonical, primitive shapes generated by codons shown in Figs. 5–7. Our aim is not to exhaust all possible inferences, but rather to indicate a profitable direction for future study.

There are two kinds of inferences to consider: (1) those that lead to the acceptance of a particular 3D shape and (2) those that reject a possible 3D shape [12, 11].

[1] It is expected that a mapping between our codon-based and axial-based (or "grassfire") representations for shapes can be made with a suitable list of indexed parameters. We see the advantage of the codon scheme being that crude part descriptions appear at the top level, allowing immediate access.

Consider the first three primitive outlines given in Fig 5: the ellipse, the peanut, and the dumbbell. An example of the rejection strategy is that the 2D peanut contour can not arise from a surface of revolution about a straight axis, for if it did, then the concavity in the outline would be eliminated. Similarly, the dumbbell can not be the projection of a 3D surface of revolution about a vertical axis, although it could be a 3D shape created by revolving the outline about the horizontal axis (i.e., a dumbbell in 3D).

An example of the accept strategy is the inference that the elliptical outline represents a 3D ellipsoid. But of course although it is true that a 3D ellipsoid will generate a 2D elliptical contour, so will any planar 2D ellipse or more awkwardly any shape whatsoever that has at least one elliptical cross section.

To infer the 3D shape from the 2D contour thus requires assumptions about (1) what the hidden 3D surface looks like, (2) whether the shape is 2D or 3D, and (3) whether the silhouette arises from a plane curve, etc. [13]. Clearly, then, the accept mode of inference is much more fragile than the rejection strategy. In one case the inference rests on assumptions, whereas in the other, possible assumptions are rejected.

What kinds of 3D properties then can be tested from 2D codons? We have already mentioned the surface of revolution constraint, which is a particularly popular basis for modelling shapes [2, 1, 8]. But still deeper insights into 3D shape can be obtained if the part boundaries are reinforced by spines and cusps which appear in the image [7]. For example, in Fig. 8 the silhouette of all three figures is the same. However, their interpretation is quite different. Assuming general position, outline (A) is seen as planar, (B) as three dimensional, whereas (C) suggests a 2D fin on a 3D ball, otherwise it would be an impossible object [6]. It is clear that as the codon representation is developed, the internal contours must play an important role. Their presence may force (or exclude) a particular 3D interpretation. These differences in interpretation should be reflected in the codon description. For example, the description of (B) should include three separate codon strings rather than just one for the silhouette. Thus the silhouette (A) = ⟨2002⟩ whereas (B) is more correctly depicted as ⟨[2][000][2]⟩, with the two [2] being the "ears" of the head [000]. In this case the transformation from A to B is a simple restructuring of the string. Other cases with similar silhouettes will not be so simple. Yet, just as 3D shapes constrain the 2D codon descriptors and vice versa, so will there be constraints on the transformation of strings of type B into those of type A (and vice versa). Here is an exciting but difficult area for future study.

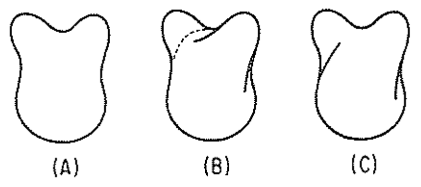


FIG. 8. Internal contours may drastically alter the 3D interpretation as well as the codon description.

9. SUMMARY

Codons are simple primitives for describing planar curves. They thus are principally image-based descriptors. Yet they have the power to capture important information about the 3D world, such as making part boundaries explicit. The codon description is highly redundant (useful for error-correction). This redundancy can be viewed as a constraint on the number of possible codon strings. For smooth closed strings that represent the bounding contour (silhouette) of many smooth 3D objects, the constraints are so strong that sequences containing 6 elements yield only 33 generic shapes as compared with a possible number of 15,625 combinations. An intriguing and important question for image understanding is to explore the constraints on the possible 3D configurations that can project into these 33 generic 2D shapes.

APPENDIX I: A BINARY MAPPING FOR CODON STRINGS[2]

1. Mapping Rule

Five basic codon types would normally require at least three bits for a binary encoding. However, there are sufficient constraints on codon joins that the 0^- and 0^+ codons can be distinguished, provided at least one member of the string is not a type 0 codon. By inspecting Tables 1 or 2, we see that if a type 0 codon follows a 1^- codon, then it must be an 0^+, whereas if the type 0 follows a 1^+ or 2 type codon, then it must be an 0^-. Similarly if a type 0 precedes a 1^- or 2, it must be type 0^-, but if it precedes a 1^+ codon, then it is type 0^+. Because adjacent type 0 codons must have the same sign, the designation of the 0 codon type will be completely specified by its neighbors. This redundancy is also reflected in the number of legal codon pairs, which is thirteen and can be mapped into 4 bits or 2 bits per codon.

Our mapping scheme utilizes the constraint that between every minima there is at least one maxima (provided the positive minima of 0^+ are noted). Thus, given the location of a minima in a binary string we only need to encode the position and number of inflections in relation to the maxima. (This was the basis for the codon definitions.) Let "1" represent an inflection and "ϕ" represent no inflection. Then the mapping rule will be as follows:

CODON TYPE		BINARY CODE
0^-	→	$\phi\phi$
1^+	→	$\phi 1$
1^-	→	1ϕ
2	→	11

Note that in this mapping, the position of both the maximum and minimum is implicit. Namely the minima lie at the beginning and end of a binary pair whereas the maximum lies between the pair. This property will be seen to be useful in depicting figure-ground reversals where maxima and minima exchange places. The

[2] This mapping and its properties were first noted by Chris Fitch and Steve Schad in WR's class, "Natural Computation" in the Fall of 1983.

exchange can be brought about simply by phase shifting the starting point on a string by one element.

2. Mirror Transform

A mirror transform in a codon string occurs most frequently when a shape is symmetric. The rule for effecting a mirror transform is to read the string backwards and change the signs of the 1^+ and 1^- codons (see [4]). The binary rule is simpler: "Read the string backwards."

EXAMPLE.

ORIGINAL STRING	1^+	2	1^-	1^+	0	1^-
BINARY MAPPING	01	11	10	01	00	10
REVERSAL BINARY	01	00	10	01	11	10
MIRROR STRING	1^+	0	1^-	1^+	2	1^-

Note that the mirror codon is simply the original read backwards with the signs changed.

3. Lock-Key or Figure-Ground Transform

The basic idea underlying a figure-ground reversal is that the maxima and minima must be exchanged. This can be accomplished by rotating the binary string by one element. However, because a figure-ground exchange also entails a reversal in the direction of traversing the curve, the order of the binary string must also be reversed. Hence the figure-ground rule is: "Rotate the binary string by one element and read the string backwards."

EXAMPLE.

ORIGINAL STRING (FIGURE)	1 +	2	1^-	1^+	0	1^-
BINARY MAPPING	01	11	10	01	00	10
ROTATE ONE ELEMENT	00	11	11	00	10	01
REVERSED BINARY	10	01	00	11	11	00
COMPLEMENTARY STRING (GROUND)	1^-	1^+	0	2	2	0

APPENDIX II: NUMBERS OF POSSIBLE OPEN, SMOOTH CODON STRINGS

1. Strategy of Proof

An upper bound on the number of legal codon strings with smooth joins, but which are not closed, will be obtained by induction. First we will determine the number of possible strings of length $N + 1$ given the number of strings of length N. This relation will then lead to an obvious sequential pattern for even and odd Ns of low numbers. The sequence will be bounded by $5 \cdot 7^{(N-1)/2}$.

2. Two Types of Strings

Referring to Table 1, we note that a codon string may be extended in only two ways: (a) by adding either a 0^-, 1^-, or 2 or (b) by adding either a 0^+ or 1^+. If the string ends in a 0^-, 1^+, or 2, then there are three choices, namely (a), for the

addition, but if the string ends in a 0^+ or 1^-, then there are only two choices, namely (b). Let us designate the 0^-, 1^+, and 2 codons as type A and the 0^+, 1^- as type B. (A and B simply specify whether the head of the codon has positive or negative curvature, respectively.) Then if there are A_N strings of length N ending in 0^-, 1^+, or 2, there will be $3A_N$ possible strings of length $N + 1$ that are constructed from these A_N codons. Similarly there will be $2B_N$ possible strings of length $N + 1$ constructed from these B_N codons. The total number $\Sigma(N + 1)$ of strings of length $N + 1$ will then be

$$\Sigma(N+1) = 3A_N + 2B_N. \tag{1}$$

Let us now consider a string of length N ending in a type A codon $(0^-, 1^+, 2)$. How did each of these codon types arise from the string of length $N - 1$? Again referring to Table 1, we see that the next-to-last codon must have been either an 0^-, 1^+, or 2. Thus we have two instances of type A (namely $0^-, 2$) and one instance of type B (namely 1^+). The number of A_N codon strings is then simply

$$A_N = 2A_{N-1} + B_{N-1}. \tag{2}$$

Similarly, we find that the last type B codon in a string (namely $0^+, 1^-$) must be preceded by either an 0^+, which is type B, or a 1^-, which is type A. Hence we have

$$B_N = A_{N-1} + B_{N-1} \tag{3}$$

We now can solve for A_N and B_N in terms of Σ. But first note that adding (2) and (3) gives us the relation

$$A_N + B_N = 3A_{N-1} + 2B_{N-1} = \Sigma(N) \tag{4a}$$

or

$$B_N = \Sigma(N) - A_N. \tag{4b}$$

Now if the right-hand terms of (4b) are replaced by the appropriate forms of Eqs. (1) and (2), we find that

$$B_N = \Sigma(N-1). \tag{5a}$$

Thus from (4a),

$$A_N = \Sigma(N) - \Sigma(N-1). \tag{5b}$$

Finally, by substitution in (1) and changing the index by one, we obtain

$$\Sigma(N) = 3\Sigma(N-1) - \Sigma(N-2). \tag{6}$$

These totals for N are given in column 3 of Table 3.

3. Upper Bound

An upper bound on these totals can be set by making the negative right-hand term of Eq. (6) smaller and then approximating the sums by the positive term. For example, by algebraic manipulation, the negative term can be reduced to $\Sigma(N - 4)$, giving

$$\Sigma(N) = 7 \cdot \Sigma(N-2) - \Sigma(N-4). \tag{7}$$

An upper bound on the possible number of open codon strings is thus $7 \cdot \Sigma(N-2)$. We then observe the following pattern:

	CORRECT
$\Sigma(3) = 7 \cdot 5 = 35$	34
$\Sigma(4) = 7 \cdot 13 = 91$	89
$\Sigma(5) = 7 \cdot 7 \cdot 5 = 245$	233
$\Sigma(6) = 7 \cdot 7 \cdot 13 = 637$	610
$\vdots$	

Note that for all odd sums the factor other than 7 is 5, whereas for the even sums, the factor is 13. Thus if N is odd, $\Sigma_{\text{odd}} < 5 \cdot 7^{(N-1)/2}$ whereas if N is even, $\Sigma_{\text{even}} < 13 \cdot 7^{(N-2)/2}$. Now note that 13 may be approximated by $5 \cdot 7^{1/2}$, or more especially $13 < 5 \cdot 7^{1/2}$. Thus the factor 13 may be replaced to yield the single equation for the upper bound on Σ,

$$\Sigma(N) < 5 \cdot 7^{(N-1)/2}. \tag{8}$$

APPENDIX III: Number of Smooth Closed Codon Strings

1. Strategy of Proof

We will use the binary representation presented in Appendix I. Furthermore, we will count only the basic closed shapes without regard to whether figure or ground is specified. The counting proceeds by constructing a binomial tree, subject to three constraining rules:

(i) The sum of the 1s must be even.

(ii) There must be an even number of adjacent binary ϕs in any sequence.

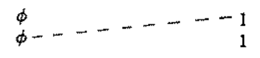
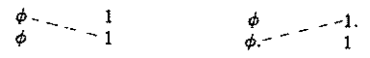
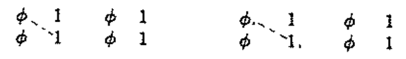
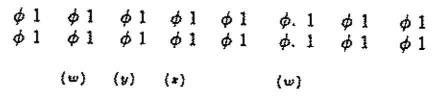
(iii) Only the 1111..., string can end in a 1.

The first rule says that the total number of inflections in a closed string must be even (or zero). If there were an odd number of inflections then the string cannot close on itself because the sign of curvature at the beginning and end of the string would be different.

The second rule follows indirectly from the first. A type 2 codon is represented by "11" because it has two inflections. Thus taking all "2"s from the string will still leave an even number of "1"s. For each remaining "1" there will be an equal number of ϕs. This number will be even. The total number of ϕs in the binary string remains even because all other ϕs in the binary string will come from the type 0 codon, which is represented by a pair of binary ϕs. The requirement that there must be an even number of adjacent ϕs follows from the legal joins shown in Table 1. For example, a $1\phi11$ sequence, corresponding to a 1^-2 string is illegal. The third digit must be a ϕ, changing the string to a 1^-1^+, etc.

TABLE 4

Tree Structure of Binary Codon Strings of Length N

STRING POSTION (or LEVEL, N)	BINARY TREE	LEGAL POSSIBILITIES
0	$\binom{1}{1}$	—
1		—
2		3
3		5
4		9

The third rule simply forces the end position in the string to be a zero to prevent duplication of strings. The exception is a codon string made up only of type "2" codons, which must be included in the count. Otherwise the string is rotated to remove a final binary "1".

2. The Construction

We wish to count all possibilities for the binary ϕ and 1 in any position of the string of codon length N. This will be accomplished by constructing a tree, with two binary positions added at each new level, N, of the tree. The binary positions, of course, represent the five basic codons plus the constraint that allows both the 0^- and 0^+ codons to be represented by the binary pair $\phi\phi$ (see Appendix I).

Table 4 shows the construction of the binary tree. The tree is initiated by the pair of 1s in the first row at level 0. These pairs of 1s then branch to a pair of 0s and 1s at level 1, corresponding to a single codon.

At this level the possible legal binary pairs according to our rules would be $\phi\phi$ and 11, corresponding to the type 0 and 2 codons. However neither of these single codons can close on itself without introducing a maxima. Hence the first legal *closed* binary sequences begin at level 2.

Reading down from the top through level 2 (and ignoring the initializing 1s), we have only three possibilities: $\phi\phi\phi\phi$, $11\phi,\phi$, and 1111 (see Fig. 5). The sequences $\phi\phi11$ and $1\phi\phi1$ are illegal because a sequence cannot end in a "1" unless all members of the sequence are "1" (rule (iii)). Hence there are only three possible closed strings made of two codons.

Moving to the third level, we now need explore only those branchings (and cross-branchings) that end in ϕ. There are five legal strings obtained by directly moving down the branches without crossing over. These are $\phi\phi\phi\phi\phi\phi$, $\phi\phi11\phi\phi$,

$11\phi\phi\phi\phi$, $1111\phi\phi$, and 111111. However, note that the second and third strings are simply a rotation of one another and hence are duplicates. Indeed, any right-branching sequence from a $\phi\phi$ to a 11 on the left side of the tree must be duplicated on the right side of the tree, because a simple leftward rotation of the binary string can move the 11 pair into the first position. The initial pair 11 will now correspond to the 11 pair at level one on the right side of the tree, and the preceding pair of binary $\phi\phi$s will have moved into the last position, corresponding to the $\phi\phi$s at the highest level being considered. Hence all direct strings on the left side of the tree can be ignored in the direct string count, with the exception of the sequence $\phi\phi\phi\phi\ldots$, consisting solely of binary ϕs. Thus, the number of legal "direct" strings constructed by moving down the tree without "crossovers" (dashed lines) will be

$$\text{Number of direct strings:} = 2 + 2^{N-2}. \tag{9}$$

3. *Crossover Strings*

The above count does not include "crossings" between branches similar to those indicated by the dashed lines. Given that a "crossing" is made at one level, we are then required by rule (ii) to cross back at the next or later level in such a manner that the number of adjacent ϕs is even, and such that the string ends in ϕ. Thus the first crossing $1\phi\phi1$ is illegal because it ends in a "1." However this sequence can be extended at level 3 into a legal string, namely $1\phi\phi1\phi\phi$. (Note that the other crossing $1\phi11\phi\phi$ is illegal because there is a single "ϕ.") We thus see that there are no legal crossover strings of codon length two, and only one of codon length three, namely $1\phi\phi1\phi\phi$.

Moving to level four, we may continue to use the $1\phi\phi1$ crossover as a "header." Now a 11 at level three may be used to extend the string, for we have a pair of binary ϕs available at level four. This gives us the two crossover strings ending in columns "y" and "z," as indicated in the last row of Table 4. In addition, we may now also create a new crossover header, namely $1\phi\phi\phi\phi1$, which becomes legal because of the binary ϕs available at level four. This new sequence, $1\phi\phi\phi\phi1\phi\phi$ ends in column "x." By considering crossovers on the left side of the tree we have thus added three more strings of codon length four.

Now consider the right-half of the tree. At level two, we may add another crossover at the "1" followed by a period. Crossing over here and back again at level three, we obtain the following sequence ending in column (w): $111\phi\phi1\phi\phi$. But this sequence is the mirror transform of a previous string found by an earlier crossover to the left side of the tree, namely $1\phi\phi111\phi\phi$. (The mirror transform is the same string read in reverse, and in this case rotated by two to place the two binary ϕs at the end.) It should be obvious that the symmetry of the crossover header, namely $1\phi\ldots\phi1$, allows one to read it backwards to get the same result. Hence all crossovers on the right half of the tree will be the "mirrors" of strings obtained on the left half.

We are now in a position to count the crossover strings. At each new level, we add one string of the form $1\phi\phi\ldots\phi1\phi\phi$. This sequence doubles at each successive level, giving us 2^{N-3} possible strings from the first possible crossover, 2^{N-4} from the second, etc.:

$$\text{Number of crossover strings} := \sum_{J=3}^{N} 2^{N-J}. \tag{10}$$

4. *Final Count*

Adding the counts for the direct (9) and crossover (10) strings, we obtain

$$\text{Number of closed strings:} = 2 + 2^{N-2} + \sum_{J=3}^{N} 2^{N-J}. \tag{11}$$

By algebraic manipulation, it can be shown that the above reduces to

$$\text{Number of closed strings:} = 2^{N-1} + 1 \qquad (N \geq 2). \tag{12}$$

ACKNOWLEDGMENTS

This report describes research done at the Department of Psychology and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this work is provided by AFOSR under a grant for Image Understanding, contract F49620-83-C-0135. William Gilson helped with the preparation of the manuscript. The comments of Aaron Bobick and John Rubin were appreciated. Chris Fitch and Steve Schad broke the combination that made Appendix III possible.

REFERENCES

1. G. Agin, *Representation and Description of Curved Objects*, Stanford A.I. Memo No. 173, Stanford Univ., 1974.
2. T. O. Binford, Unusual perception by computer, in *IEEE Conference on Systems and Control*, December, Miami, 1972.
3. D. Hoffman, *Representing Shapes for Visual Recognition*, Ph.D. thesis, MIT, Cambridge, Mass., 1983.
4. D. Hoffman and W. Richards, Representing smooth plane curves for visual recognition: Implications for Figure–Ground Reversal, in *Proceedings of the American Association for Artificial Intelligence*, pp. 5–8, 1982.
5. D. D. Hoffman and W. Richards, Parts of recognition, *Cognition*, in press; MIT A.I. Memo No. 714, 1984.
6. D. A. Huffman, Impossible objects as nonsense sentences, in *Machine Intelligence*, Vol. 6 (B. Meltzer and D. Michie, Eds.), Edinburgh Univ. Press, Edinburgh, 1971.
7. J. Koenderink and A. Van Doorn, The shape of smooth objects and the way contours end, *Perception*, **11** (1982), 129–137.
8. D. Marr, Analysis of occluding contour, *Proc. R. Soc. London Ser. B*, **197**, 1977, 441–475.
9. D. Marr, Early processing of visual information, *Philos. Trans. R. Soc. London*, **275**, 1976, 483–524.
10. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Freeman, San Francisco, 1982.
11. W. Richards, J. Rubin and D. D. Hoffman, Equation counting and the interpretation of sensory data, *Perception*, **11**, 1983, 557–576.
12. J. M. Rubin and W. A. Richards, Color vision and image intensities: When are changes material? *Biol. Cybern.*, **45**, 1982, 215–226.
13. K. A. Stevens, Visual interpretation of surface contours, *Artif. Intell.*, **17**, 1983, 47–73.
14. W. Turton, *A Conchological Dictionary of the British Islands*, (frontispiece), printed for John Booth, London, (1819). [This early reference was kindly pointed out to us by J. F. W. McOmic.]
15. D. Waltz, Understanding line drawings of scenes with shadows, in *The Psychology of Computer Vision* (P. Winston, Ed.), McGraw–Hill, New York, 1975.