STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM - 199

STAN-CS-73-398

IMAGE CONTOURING AND COMPARING

BY

BRUCE G. BAUMGART

SUPPORTED BY

ADVANCED RESEARCH PROJECTS AGENCY
ARPA ORDER NO. 2494
PROJECT CODE 3D30

OCTOBER 1973

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM-1 99

OCTOBER 1973.

COMPUTER SCIENCE DEPARTMENT REPORT NO. CS-398

IMAGE CONTOURING AND COMPARING

Bruce G. Baumgart

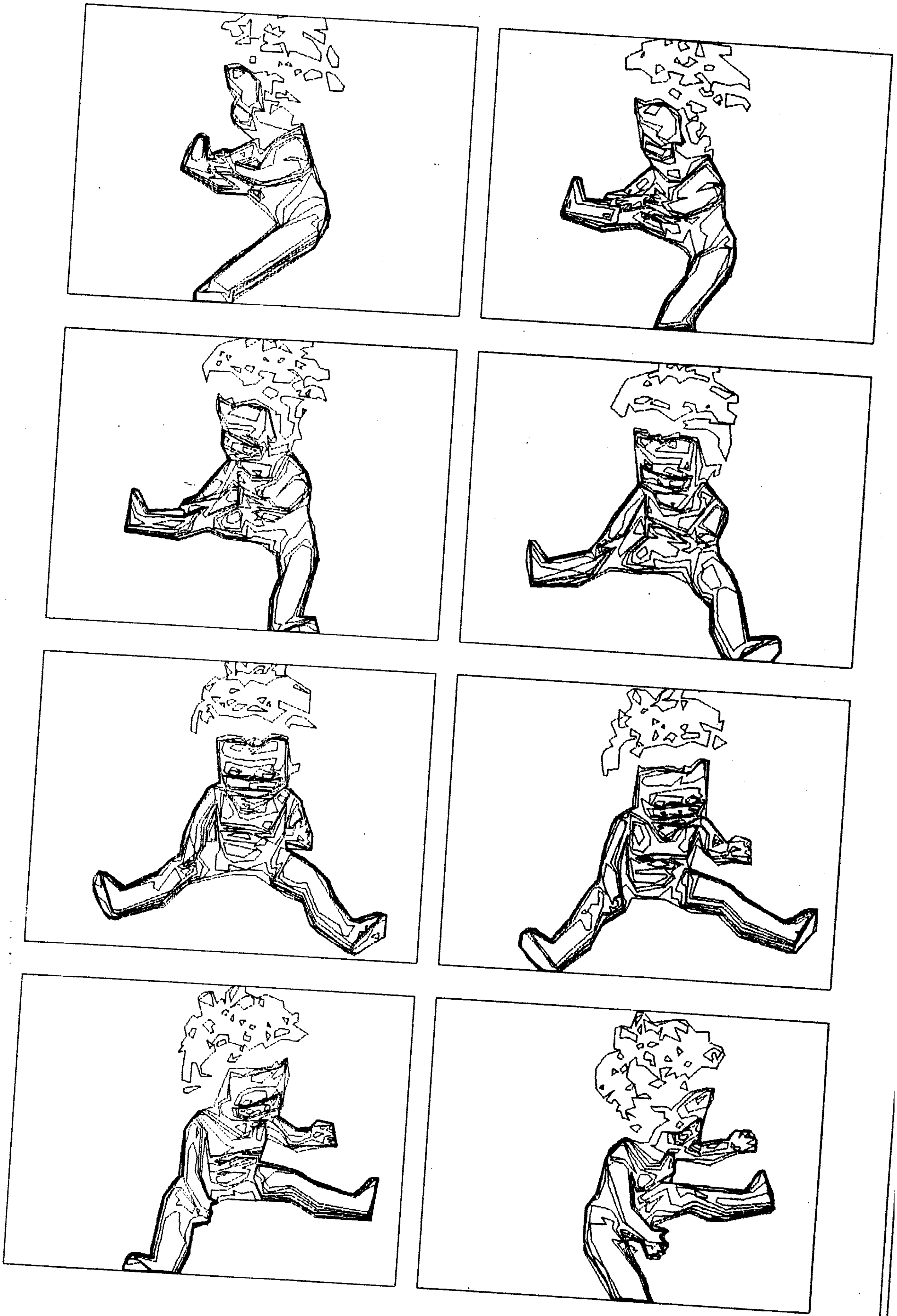
ABSTRACT:

A contour image representation is stated and an algorithm for converting a set of digital television irnages into this representation is explained. The algorithm consists of five steps: digital image thresholding, binary image contouring, polygon nesting, polygon smoothing, and polygon comparing. An implementation of the algorithm is the main routine of a program called CRE; auxiliary routines provide cart and turn table control, TV camera input, image display, and xerox printer output. A serendip application of CRE to type font construction is explained. Details about the intended application of CRE to the perception of physical objects will appear in sequels to this paper.

CONTENTS:

This research was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under Contract No. DAHC 15-73-C-0435.

FIGURE 1- BABYDOLL ON A TURN TABLE.



INTRODUCTION.

The acronym CRE stands both for "Contour, Region, Edge". CRE is a solution to the problem of finding contour edges in a set of television pictures and of linking corresponding edges from one picture to the next. The process is automatic and is intended to run without human intervention. Furthermore, the process is bottom up; there are no significant inputs other than the given television images. The output of CRE is a 2D contour map data structure which is suitable input to a 3D geometric modeling program.

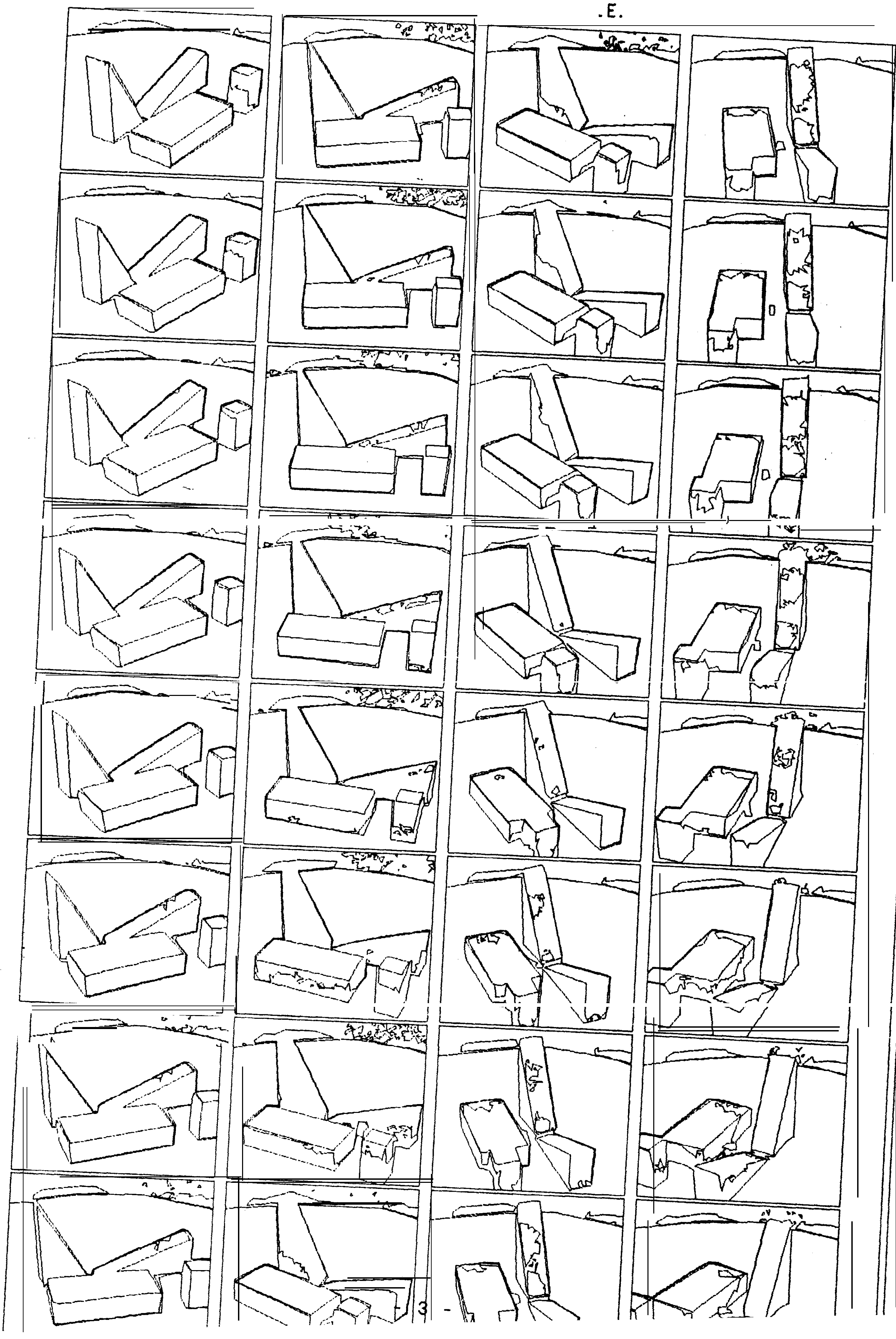
The overall design goal for CRE was to build a region-edge finding program that could be applied to a sequence of television pictures and that would output a sequence of line drawings without having to know anything about the content of the images. Furthermore it was desired that the line drawings be structured. The six design choices that determined the character of CRE are:

1. Dumb vision rather than model driven vision.
2. Multi image analysis rather than single image analysis.
3. Total image structure imposed on edge finding; rather than separate edge finder and image analyzer.
4. Automatic rather than interactive.
5. Fixed image window size rather than variable window size.
6. Machine language rather than higher level language.

The design choices are ordered from the more strategic to the more tactical; the first three choices being research strategies, the latter three choices being programming tactics. Adopting these design choices lead to image contouring and contour map structures similar to that of Krakauer[3] and Zahn[4].

The first design choice does not refer to the issue of how model dependent a finished general vision system will be (it will be quite model dependent), but rather to the issue of how one should begin building such a system. I believe that the best starting points are at the two apparent extremes of nearly total knowledge of a particular visual world or nearly total ignorance. The first extreme involves synthesis (by computer graphics) of a predicted 2D image, followed by comparing the predicted and a perceived image for slight differences which are expected but not yet measured. The second extreme involves analysing perceived images into structures which can be readily compared for near equality and measured for slight differences; followed by the construction of a 3D geometric model of the perceived world. The point is that in both cases images are compared, and in both cases the 3D model initially {or finally} contains specific numerical data on the geometry and physics of the particular world being looked at.

FIGURE 2 - BLOCK SCENE ON A TURNTABLE.



INTRODUCTION.

The second design choice, of multi image anaylsis rather than single image analysis, provides a basis for solving for camera positions and feature depths. The third design choice solves (or rather avoids) the problem of integrating an edge finder's results into an image. By using a very simple edge finder, and by accepting all the edges found, the image structure is never lost. This design postpones the problem of interpreting photometric edges as physical edges.

The fourth choice is a resolution to write an image processor that does not requires operator assistance or parameter tuning. The fifth choice of the 216 by 288 fixed window size is a sin that proved surprisingly expedient, it is explained later. A variable window version of CRE at halves, thirds and other simple fractions of its present window size will be made at some future date.

The final design choice of using machine language was for the sake of implementing node link data structures that are proccssed 100 faster than LEAP, 10 times faster than compiled LISP and that require significantly less memory than similar structures in either LISP or LEAP. Furthermore machine code assembles and loads faster than higher level languages; and machine code can be extensively fixed and altered without recompiling.

It is my impression that CRE does not- raise any new scientific problems; nor does it have any really new solutions to the old problems; rather CRE is another competent video region edge finding program with its own set of tricks. However, it is further my impression that the particular tricks for smoothing, nesting and comparing polygons in CRE are original as programming t cchniques.

The intended use of CRE is illustrated by the sequences of turn table pictures on pages 1 and 3. The figures on page 5 illustrate the quality of contoured images over a range of subject matter. Finally the application of CRE to typography is illustrated below:

Christmas Font

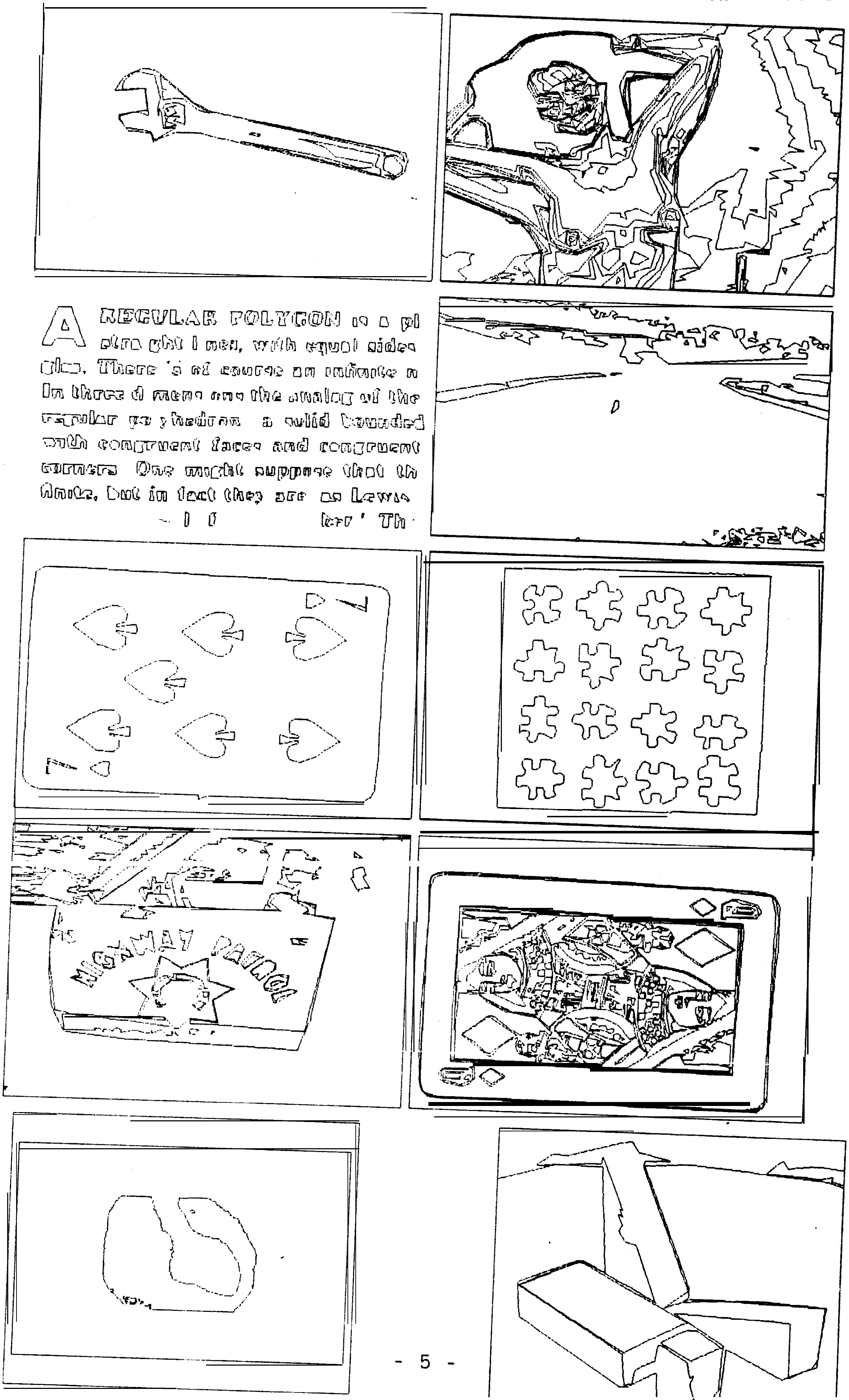
A B C D E F G H I J K L M N O P Q R S T U V W X

a b c d e f g h i j k l m n o p q r s t u v w x y z

ACKNOWLEDGEMENT.

Tovar Mock assisted me with the development of the type font making program, TVFGNT.

FIGURE 3 - INTENSITY CONTOURING ON A VARIETY OF STIMULI



DA-T-A STRUCTURE: BASICS.

The two generic data structures of CRE are arrays and nodes; there are five kinds of arrays and eight kinds of nodes. The node structures to be discussed are implemented as seven word fixed sized blocks in a fashion usual to graphics and simulation; an introduction to this technology can be found in Knuth [4]. The language of implementation is PDP- 1 0 machine code via the FAIL assembler.

The whole nodal structure in CRE represents a sequence in time of video intensity contour maps. Such contour maps are like topographical elevation contour maps, in that no two contour lines should ever cross and in that all the contour lines should close. Consequently, the loops of contours enclose regions; and these regions overlap in a nested fashion forming a tree like data structure.

As the general examples of contoured images on page 5 illustrate, a notion that is emphatically not in CRE, is that of a schematic line drawing. Although the CRE output can be viewed as a collection of lines on a display screen, people expecting a line drawing rendition of the given television picture will be disappointed. A CRE picture is a simple transformation of the photometry, geometry and topology of the original video image; whereas the typical line drawing from a human illustrator is a representation of the scene without photometric inforrnation. On the other hand, the work of an artist such as Peter Max; or a paint-by-the-numbers grid does resemble CRE output. This is not an idle coincidence but rat her a consequence of whether or not the artist is trying to represent photometric data by quantum lines.

The explanation of CRE node structures will be presented in three parts: first, the several kinds of nodes will be briefly explained; second, the sub structures such as rings, trees and lists will be described; and third, the node formats and their contents will be explained in detail. Following that will be an explanation of the five at-rays in CRE. The reader is warned that this whole sub section (on data structure) is an elaborate shaggy dog st ory o f naming names and defining things; all the action is to be found in the following sub section (on the algori t hm).

DATA STRUCTURE: KINDS OF NODES.

The are eight kinds of CRE nodes: Vector, Arc, Polygon, Shape, Image, Level, Film and Empty:

1. At the top of the structure is the film node, the film node is unique and serves as an OBLIST from which all other node's may be reached. The film node embodies the idea of a piece of celluloid film or a length of magnetic video tape. A film is a sequence of images taken by the sarne camera of the same scene with only a small amount of action between images.

2. An image node represents the familiar two dimensional idea of a photograph or an oil painting or to be exact a digital video image of 216 rows by 288 columns of numbers ranging from 0 for dark to 63 for bright. The image is formed by a thin lens and is projected on a flat image plane. The idea of an image is so common that it is easy to overlook the wonder of sun light scattering off of surfaces, refracting thru a lens, and forming a complex pattern called a real image.

3. Below the image node are the intensity contour levels, A contour level is a binary image that results from thresholding a gray scaled image. So an image is composed of levels and, in turn, a level is composed of polygons.

4. A Polygon node represents the idea of a contour loop which always closes upon it self and does not cross it self or any other contour, Contour loops are approximated by a ring of vectors; hence, the term “polygon”. The contour polygons always have at least three sides and are simply connected.

5. Shape nodes contain data about one or two polygons. The data in a shape node is not a positive representation of the notion of shape; but is rather the parameters of alignment that must be normalized out before shapes can be compared.

6. Vector nodes contain the locus of an image vertex; however since vectors always belong to a polygon and always have two neighbors; their counterclockwise neighbor is considered to determine their vector direction,

7. Arc nodes are vectors that are made by the polygon smoothing routine; one arc typically replace several vectors. When both arcs and vectors arc being discussed; vectors are strictly horizontal and vertical, whereas arcs may point in any direction,

8. Empty nodes are an artifact of the fixed node size dynamic storage allocation mechanism used in CRE. Entities are made by taking empty nodes from an AVAIL list and entities are killed by returning their node to the AVAIL list; there is no garbage collector, but there is a space compactor.

DATA STRUCTURE; LINK AND DATUM NAMES.

Nodes contain either numerical data or pointers to other nodes; such node pointers are actual machine addresses and are called links. The positions within a node where a link is stored are named and are reserved for particular uses, In the table below the 11 link names and 1 3 datum names are -introduced. The link names will always appear capitalized.

1 1 LINK NAMES:

CW clockwise
CCW count er clockwise
DAD parent of node up a free structure.
SON descendent of a node down a tree structure.
ENDO Greek for inside, polygon wit hin.
EXO Greek for out side, surrounding polygon.
ALT alternate.
NGON negative polygon.
PGON positive polygon.
NTIME negative in real time, into the past.
PTIME positive in real time, into the future,

13 DATUM NAMES:

Bool ean dat ums.

type	type of node bits.
reloc	rellocation of node bits.

Fixed point datums.

row	row of image locus.
col	column of image locus.
cnt rst	contrast of an edge, vector and arc,
ncnt	number count, various uses.

Floating point datums.

zdepth	z depth from camera lens center.
perm	length of perimeter.
area	area in pixel unit s.
mxx	motnent of inertia about X axis.
myy	moment of inertia about Y axis.
mzz	moment of inertia about Z axis.
pxy	product of inertia with respect to the X and Y axes.

DA-T-A STRUCTURE: THE RINGS, TREES, LISTS AND ARRAYS,

CRE inputs an image into an array called TVBUF; it makes the node structures, some of which are temporary; and it out puts a final version of the structure representing a film of images. The temporary structures are relevant to understanding the process; but only the final structure is relevant. to using CRE output. In summary, the important structures are:

FOUR RINGS.

1. Image ring of the film.
2. Level ring of an image.
3. Polygon ring of i-1 level.
4. Vector ring of a polygon.

TWO TREES.

1. The tree of rings.
2. The tree of nested polygons.

TWO LISTS.

1. Time line lists.
2. The empty node list.

TEMPORARY STRUCTURES.

1. Arc rings of polygons.
2. Fusion shape rings of levels.

FIVE ARRAYS.

1. TVBUF - 216 rows, 288 columns of 6 bit bytes,
2. PAC - 216 rows, 288 columns of 1 bit bytes.
3. VSEG - 216 rows, 289, columns of 1 bit bytes.
4. HSEG - 217 rows, 288 columns of 1 bit bytes.
5. SKY - 216 rows, 289 columns of 18 bit bytes.

There is one film node. The film is composed of a ring of images. Each image is composed of a ring of levels. Each level is composed of a ring of polygons. Each polygon is composed of a ring of vectors. The ring structures are implemented with the four links named DAD, SON, CW and CCW. The rings are headless only in the sense that all the elements of a ring are brothers; a pointer to the head of a ring is stored in the DAD link of each element. The DAD of the film node is NIL; and NIL is an 18-bit zero. The final SON of all vector nodes is also NIL. The DAD and SON links form a tree of rings.

DATA STRUCTURE: THE RINGS, TREES, LISTS AND ARRAYS.

Besides the tree of rings, there is the tree of nested polygons. The nested polygon tree is implemented with the four links named ENDO, EXO, NGON and PGON. The EXO of a polygon points at its surrounding polygon. The ENDO of a polygon points at one of the polygons that may be enclaved within the given polygon; and the NGON and PGON links form a ring of polygons that have the same EXO polygon,

The time line lists run thru arc and polygon nodes. In the simple case, the time line links of a polygon point to a corresponding polygon in the image previous (NTIME) or subsequent (PTIME) of the current polygon; the correspondence being that the time polygon is exactly the same intensity at nearly the same location, orientation, and size as the given polygon. In the case of polygon fusion, the time line link of a polygon points to a time polygon of which the given polygon becomes a part. In the case of polygon fission, the time line link of a polygon points to only one the pieces into which the given polygon splits.

The time line links of an arc vector point to a corresponding arc vector in the image previous or subsequent of the current arc vector. The polygons of arc vectors mated in time are themselves mated in time; because after polygon time tine links have been made, one polygon is temporarily translated, rotated and dilated so as to have the same lamina inertia tensor as its mate; that is the locus of the arc vectors of one polygon arc temporarily altered; then the corresponding arc vectors are found and their time line linkages are made.

The empty node list is maintained in the CCW link positions; the last empty node contains a zero link. All nodes are explicitly made from and killed to the empty node list by the subroutines MKNODE and KLNODE.

The arc ring of a polygon is just like a vector ring except that the pointer to it is stored in the ALT link of the polygon, white the polygon has both a ring of vectors and a ring of arcs.

The fusion shape ring of a intensity level runs thru the CW and CCW links of shape nodes and is pointed at by the ALT link of the level. Fusion shape nodes are the shapes generated to represent pairs of polygons unmated in time.

DATA STRUCTURE: TYPE BITS.

Each node has a word reserved for a boolean vector of 36 values, or bits. The first eighteen bits are called the type bits and are individually named as follows:

for	0	WESBIT	westward vector.
vectors	1	SOUBIT	southward vector.
only	2	EASBIT	eastward vector.
	3	NORBIT	northward vector.
	4	NFUSE	NTIME polygon fusion.
	5	NFISS	NTIME polygon fission.
for	6	NEXCT	NTIME polygon exact match.
polygons			
only	7	PFUSE	PTIME polygon fusion.
	8	PFISS	PTIME polygon fission.
	9	PEXCT	PTIME polygon exact match.
	10	HOLBIT	Hole polygon bit.
modify	11	ARCBIT	Arc vector bit.
	12	SBIT	Shape node bit.
	13	VBIT	Vertex node bit.
	14	PBIT	Polygon node bit.
kind			
	15	LBIT	Level node bit,
	16	IBIT	Image node bit.
	17	FBIT	Film node bit.

The first four bits WESBIT, SOUBIT, EASBIT and NORBIT apply only to vectors and indicate the direction of the vector. The next six bits NFUSE, NFISS, NEXCT, PFUSE, PFISS, PEXCT are set by the polygon compare routine to indicate the kind of time mating found; where N and P mean negative time or postive time linkage; fusion means that the given polygon and another polygon fuse to form the time polygon, two into one; fission means the given polygon splits, one into two; and exact means that the given polygon matchs one for one with its time polygon.

The next two bits HOLBIT and ARCBIT indicate distinguished polygons and vectors respectively. Only one of the last six bits: SBIT, VBIT, PBIT, LBIT, IBIT and FBIT may be on in a node. These bits indicate the node's type.

DATA STRUCTURE: RELOCATION BITS.

The next eighteen bits are called the reloc bits and indicate whether or not a link is stored in a particular position of the node. The relocation bits are used to compact the CRE node space for output.

18 unused
19 CAR(WORD0)
20 CDR(WORD0)

21 unused
22 CAR(WORD1)
23 CDR(WORD 1)

24 unused
25 CAR(WORD3)
26 CDR(WORD3)

27 unused
28 CAR(WORD4)
29 CDR(WORD4)

30 unused
31 CAR(WORD5)
32 CDR(WORD5)

33 unused
34 CAR(WORD6)
35 CDR(WORD6)

The CAR of a word is the left half. The CDR of a word is the right half. in the node diagrams the relocation of each word is indicated directly to its right as 0, 1, 2 or 3 meaning no relocation, left only, right only, and relocate both halves, respectively,

1. VECTOR & 2. ARC NODE FORMAT.

word 0	CW vector	CCW ring	3
word 1	DAD polygon	SON arc or vector	3
word 2	type	reloc 33 0003	
word 3	row 0000.00	col 0000.00	0
word 4	cntrst	ncnt length	0
word 5	zdepth		0
word 6	NTIME time	line PJIME	3

The format of vectors and arcs is identical. Inside CRE the term "vector" has the connotation of being strictly a horizontal or vertical generated by the cont ouring step; whereas an arc is a vector generated by the smoothing step. Vectors contain the fundamental geometric datum of an image locus. The image locus is stored in the halfword datums named row and cot, which contain the row and column of a point in units 1/64 of a pixel. (A "pixel" is a "picture element"). Vectors and arcs also contain the photometric datum of edge contrast.

Vectors always belong to a polygon node, a pointer to the polygon of each vector is stored in the link named DAD; as members of a polygon the vectors form a loop which is always connected so that each vertex has a neighboring vertex in the clockwise and in the counter clockwise directions about the polygon's perimeter; these perimeter pointers are stored in the link positions named CW and CCW. Vectors never cross, arcs cross on occassions but can be fixed.

The ncnt datum of arcs and vectors contains their length. The time line links, NTIME and PTIME, may point to a corresponding arc or vector in the image previous or subsequent to the current image. (The zdepth datum contains a positive number indicating distance from the camera's image plane; the tdepth computation is not properly implemented as of May 1973).

3. POLYGON NODE FORMAT.

word 0	CW polygo-	CCW ring	3
word 1	DAD level	SON 1st vector	3
word 2	type 10	reloc 33 3233	
word 3	ENDO 1st polygon within	EXO polygon surround me	3
word 4	ALT shape (or 1st arc)	ncnt number of sides	2
word 5	NGON nest bro polygon	PGON nest sis polygon	3
word 6	NTIME time	line PTIME	3

Every polygon belongs to a level pointed at by the DAD link; the ring of polygons of a level is formed in the CW and CCW links; the son of a polygon is its first vector (or arc after the polygon has been smoothed) and that first vector has the upper left most locus of any vector of the polygon.

The ENDO, EXO, NGON, PGON are used to form the nested polygon tree. Every polygon has non-NIL NGON and PGON links; the trivial case being that the polygon points at it self twice. Every polygon except one, the outer border polygon, has a non-NIL EXO link. Every polygon that surrounds one or more other polygons has a non-NIL ENDO link.

The ALT link position of a polygon temporarily points to the first arc of a polygon during smoothing when a polygon has both vectors and arcs. The final contents of the ALT link is a pointer to the shape node of the polygon. The ncnt datum indicates the number of sides of the polygon.

The time line of polygons runs thru the NTIME and PTIME links which point either to a nearly exact match of a polygon; or to a fusion polygon of a two for one match; or to one of the two fission parts of a one for two match; {to find the other fission part, the tjme links of the vectors must be scanned).

4. SHAPE NODE FORMAT.

word 0	CW fusion shape ring	CCW	3
word 1	perm	area	0
word 2	type 10	reloc 30 0030	
word 3	row 0000.00	col 0000.00	0
word 4	pxy product of inertia	mzz Z-axis moment	0
word 5	NGON fusion polygon	PGON main polygon	3
word 6	mxx X-axis moment	myy Y-axis moment	0

The shape node contains the data necessary for normalizing two polygons so that only their shapes remain. In particular, the row and col of a shape node is the center of mass of the polygon; area is the area; perm is the length of perimeter; and mxx, myy, mzz, pxy is the polygons inertia tensor (frorn which the principle angle of orientation can be computed). When given two shapes, the centers of mass may be aligned; the principle angles may be align; and the areas (or perimeter) of the two may be normalized.

There are two kinds of shapes: polygon shapes and fusion shapes. Polygon shapes correspond to a single polygon pointed at by the PGON link. The CW, CCW and NGON links of a polygon shape are NIL. Fusion shapes are temporary nodes belonging to a level as a ring thru CW and CCW. Fusion shapes correspond to the summation of two unmated polygons which are pointed to by the NGON and PGON links. The expressions relating to the inertia tensor and to fusion summation are given in the section on polygon comparing.

The datums named perm, area, pxy, mxx, myy, mzz contain the left half of a PDP-10 floating number. (Technical note: half of a floating number has 9 bits of precision and should be expanded to full word by using the (mirabile dictu!) HLLE instruction in order to avoid an illegal floating zero caused by truncating numbers like -1023.0; in CRE, only the product of inertia will ever be negative).

5. LEVEL NODE FORMAT.

word 0	CW level	CCW ring	3
word 1	DAD image	SON 1st polygon	3
word 2	type	reloc 33 0200	
word 3	---	---	0
word 4	ALT (1st fusion shape)	ncnt threshold cut level	2
word 5	---	---	0
word 6	---	---	0

Every level belongs to an image pointed to by the DAD link; the ring of levels of an image is formed in the CW and CCW links; the son of a level is its first polygon and that first polygon is the upper left most polygon of the level.

The ncnt datum of a level contains its threshold cut value, which is an integer between -1 and 63. The -1 level is always generated, and it contains a single polygon with four sides. The -1 level's polygon is called the border polygon; the fiction being that every point beyond the edges of the television picture has an intensity value of -2, which is blacker than black.

The ALT link of a level contains a temporary pointer to that level's ring of fusion shapes during polygon compare time mating.

6. IMAGE NODE FORMAT.

word 0	CW image	ring CCW	3
word 1	DAD film	SON 1st level	3
word 2	type	reloc 33 0000	
word 3		---	0
word 4	---	---	0
word 5	---	---	0
word 6	---	---	0

Every image belongs to the film pointed to by the DAD link; the ring of images of the film is formed in the CW and CCW links; the son of an image is its first level and that first level is the -1 intensity cut level of the image.

Although an affront to common sense, the counter clockwise direction about the image ring is positive or later in time and the clockwise direction is negative or earlier in time. I achieved this curio by consistently adhereing to the mathematical convention of counter clockwise as. positive; and a day came when counter clockwise around a ring of real time events was represented in the same manner as counter clockwise around a polygonal ring of edges.

All the empty space in the image node is reserved for camera specification data.

7. FILM NODE FORMAT.

word 0		CCW 1st empty	1
word 1	DAD 0	SON 1st image	3
word 2	type	reloc 33 0000	
word 3	---	---	0
word 4	---	---	0
word 5	---	---	0
word 6	---	---	0

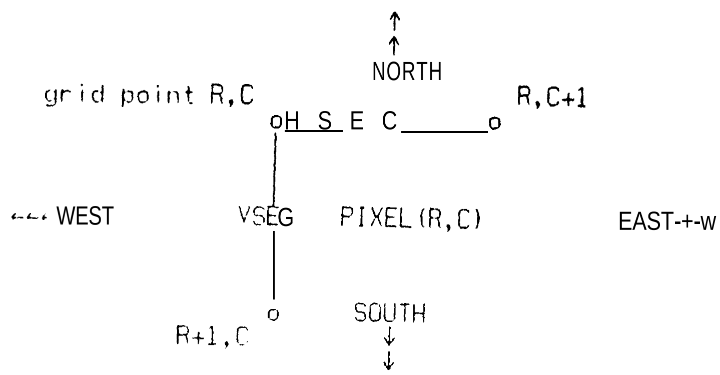
The film -node is unique; it is the first node in a CRE output file; the SON of film is its first image; the DAD of a film is NIL; the CCW of a film is a pointer to the 1st empty node; however, because the nodes are compacted for output and then relocated with respect to the film node; the final empty node pointer indicates the number of words of data in the CRE file.

8. EMPTY NODE FORMAT.

word 0	---	CCW avail	1
word 1	---		0
word 2	type	reloc 00 0000	
word 3	---	---	0
word 4	---	---	0
word 5	---	---	0
word 6	---	---	0

The list--of empty nodes is maintained in the CCW link postion; the last empty node contains a zero or NIL link. At present all the other words of an empty node are zero.

FIGURE SHOWING RASTER STRUCTURE.



DATA STRUCTURE: IMAGE ARRAYS.

As mentioned before, there are five arrays in CRE: TVBUF, Television Buffer; PAC, Picture Accumulator; VSEG, vertical segments; HSEG, horizontal segments; and SKY, background sky blue array. The dimensions are;

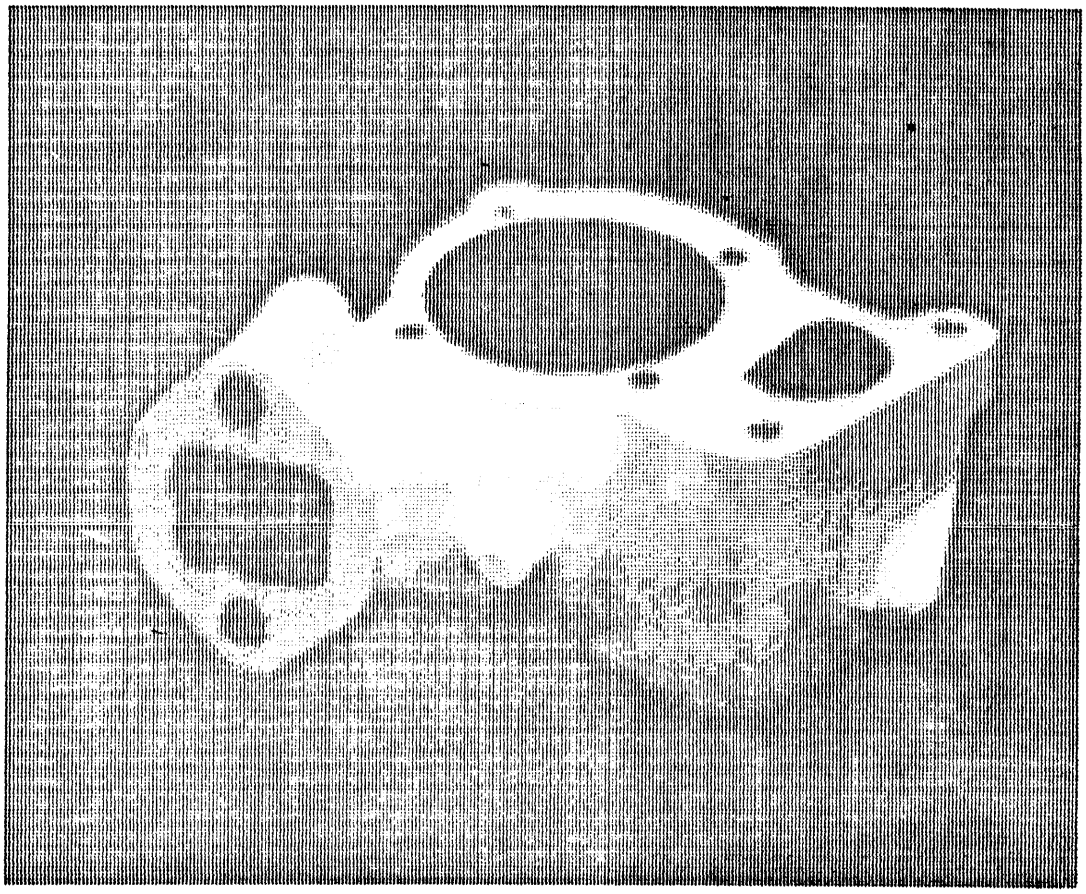
FIVE ARRAYS.

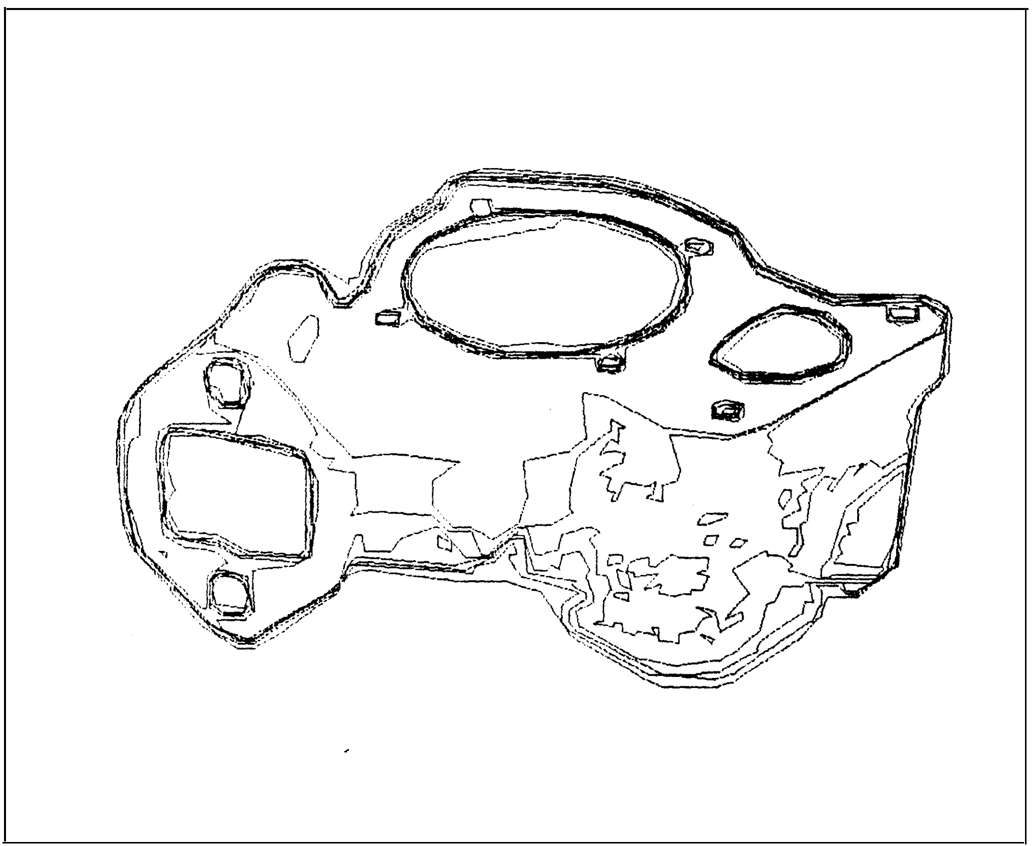
1. TVBUF - 216 rows, 288 columns of 6 bit bytes.
2. PAC - 216 rows, 288 columns of 1 bit bytes.
3. VSEG - 216 rows, 289 columns of 1 bit bytes.
4. HSEG - 217 rows, 288 columns of 1 bit bytes.
5. SKY - 216 rows, 289 columns of 18 bit bytes.

Inside CRE, the video image size was fixed at 216 rows of 288 columns of 6 bits per pixel. My original idea was to write a vision operator that would be applied on a small fixed sized window; so I have had windows 2 by 2; 2 by 3; 4 by 9; 32 by 36; 72 by 96; and 216 by 288. That is 216=2*2*2*3*3*3 and 288=2*2*2*2*2*3*3. Having a fixed window size avoids a morass of word packing, array allocation and window splicing. Having a window size constructed out of powers of 2 and 3 simplifies what word packing is required and allows me to do area and space computations in my head.

The image arrays of CRE are of course two dimensional with the coordinates in row and columns. Row number increases going down image, in the negative Y axis direction, which is also called the direction south. Column numbers increase going right on the image, in the positive X axis direction, which is also called the direction east. Video picture elements, or "pixels" are thought of as expressing the intensity of a square cell; the cells are numbered from 0 to 215 rows, 0 to 287 columns; the number of a ceil is the grid locus of its upper left (northwest) corner; the center locus of a cell is at (row+1/5,col+1/2). A pixel cell is surrounded by four segments; the horizontal segments are numbered 0 to 216 rows, 0 to 287 columns; the number of an HSEG is the grid locus of its left (west) end point. The vertical segments are numbered 0 to 215 rows, 0 to 288 columns; the number of a VSEG is the grid locus of its upper (north) end point. These conventions are suggested in the diagram at the bottom of page 19.

FIGURE 4 - WATER PUMP VIDEO AND SMOOTH CONTOURS.





THE ALGORITHM: INTRODUCTION.

CRE consists of five steps: t hresholding, contouring, nesting, smoothing and comparing. Thresholding, contouring and smoothing perform conversions between two different kinds of images. Nesting and contouring compute topological relationships within a given image representation. In summary the major operations are:

MAJOR OPERATION.	OPERAND.	RESULT.
1. THRESHOLDING:	6-BIT-IMAGE,	1 -BIT-IMAGES.
2. CONTOURING:	1 -BIT-IMAGES,	VIC-IMAGE.
3. NESTING:	VIC-IMAGE,	NESTED-VIC-IMAGE.
4. SMOOTHING:	VIC-IMAGE,	ARC-IMAGE.
5. COMPARING;	IMAGE & FILM,	FILM.

Although the natural order of operations is sequential from image thresholding to image comparing; in order to keep memory size down, the first four steps are applied one intensity level at a time from the darkest cut to the lightest cut (only nesting depends on this sequential cut order); and comparing is applied to whole images.

The illustrations on pages 21 and 23 show an initial video image and its final smoothed contour image; the illustrations immediately below and on page 24 the corresponding intermediate sawtoothed contour images. The illustrated images are each composed of seven intensity levels, and took 16 seconds and 13 seconds to compute respectively (on a PDP-10, 2usec memory). The final CRE data structures contained 680 and 293 nodes respectives, which comes to 2K and 4.5K words respectively; the initial video image requires 10.2K words.

FIGURE: PUMP SAW TOOTHED CONTOURS.

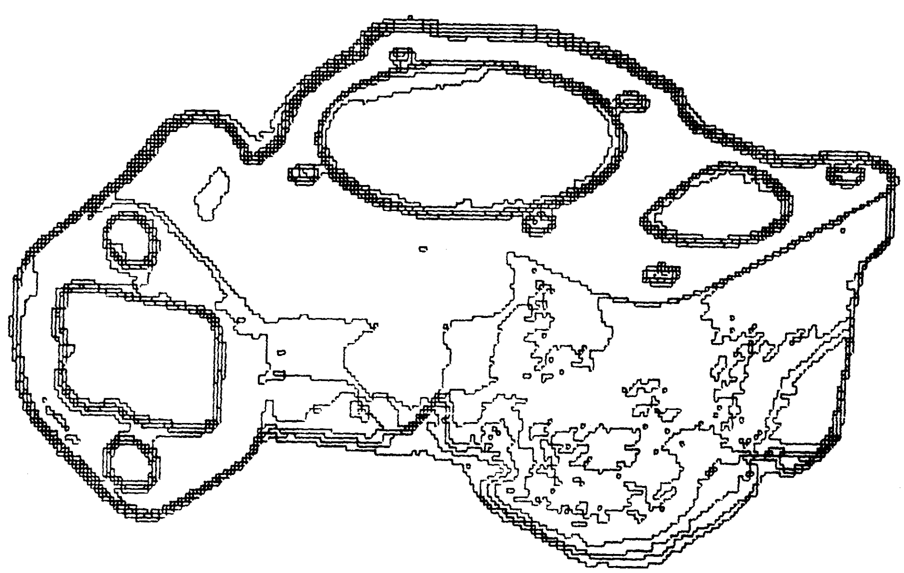
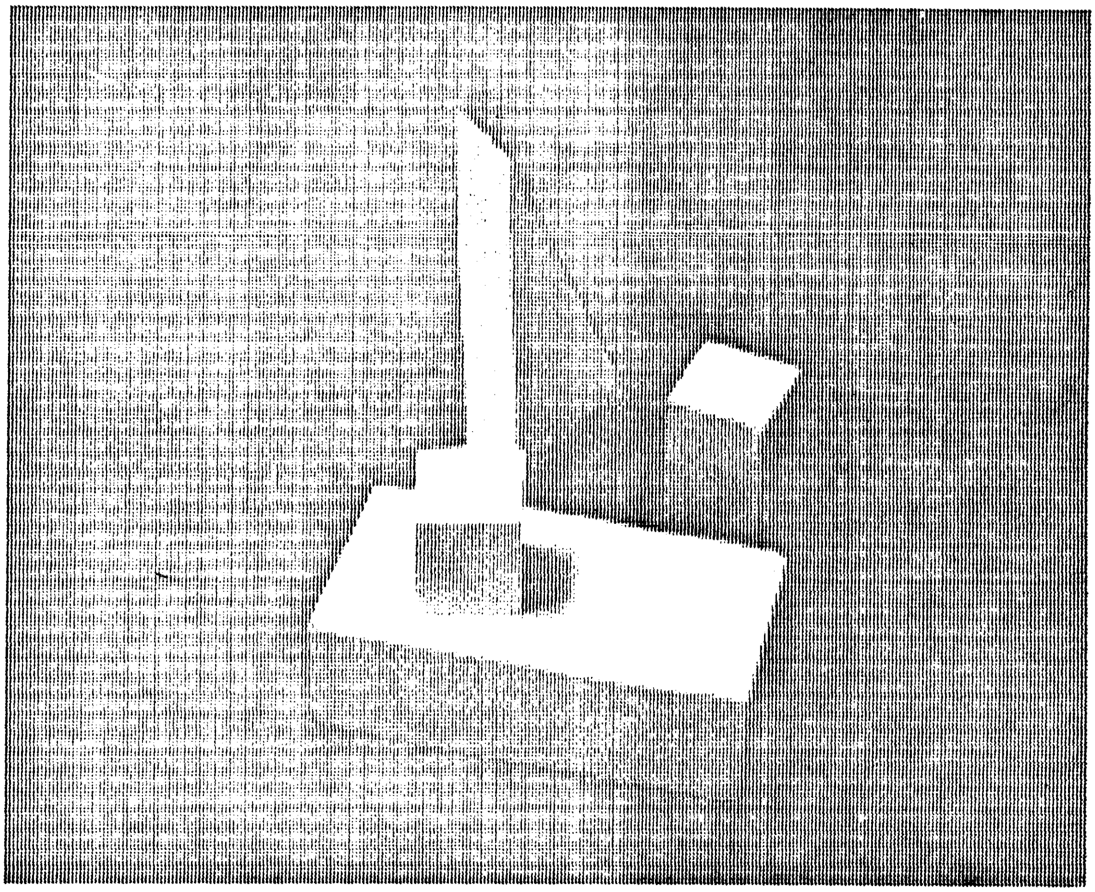
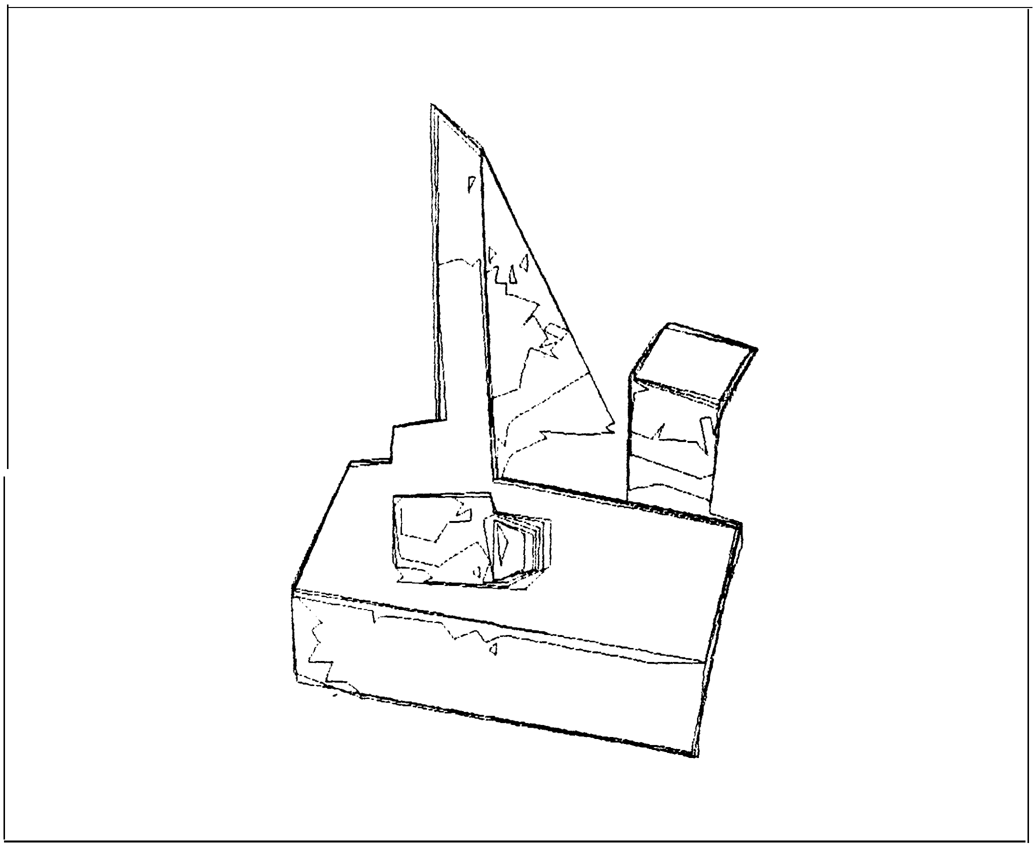


FIGURE 5 - BLOCK SCENE VIDEO AND SMOOTH CONTOURS.





1. THRESHOLDING.

Thresholding, the first and easiest step, consists of two subroutines, called THRESH and PACXOR. THRESH converts a 6-bit image into a l-bit image with respect to a given threshold cut level between zero for black and sixty-three for light. All pixels equal to or greater than the cut, map into a one; all the pixels less than the cut, map into zero. The resulting 1 -bit image is stored in a bit array of 216 rows by 288 columns (1 728 words) called the PAC (picture accumulator) which was named in memory of McCormick's ILLIAC-III. After THRESH, the PAC contains blobs of bits. A blob is defined as "rook's move" simply connected; that is every bit of a blob can be reached by horizontal or vertical moves from any other bit without having to cross a zero bit or having to make a diagonal (bishop's) move. Blobs may of course have holes. Or equivalently a blob always has one outer perimeter polygon, and may have one, several or no inner perimeter polygons. This blob and hole topology is recoverable from the CRE data structure and is built by the nesting step.

Next, PACXOR copies the PAC into two slightly larger bit arrays named HSEG and VSEG. Then the PAC is shifted down one row and exclusive OR'ed into the HSEG array; and the PAC is shifted right one column and exclusive OR'ed into the VSEG array to compute' the horizontal and vertical border bits of the PAC blobs. Notice, that this is the very heart of the "edge finder" of CRE. Namely, PACXOR is the mechanism that converts regions into edges.

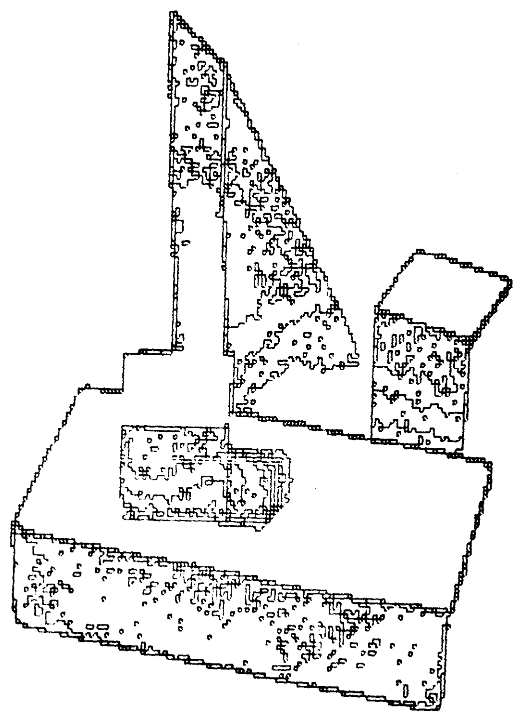
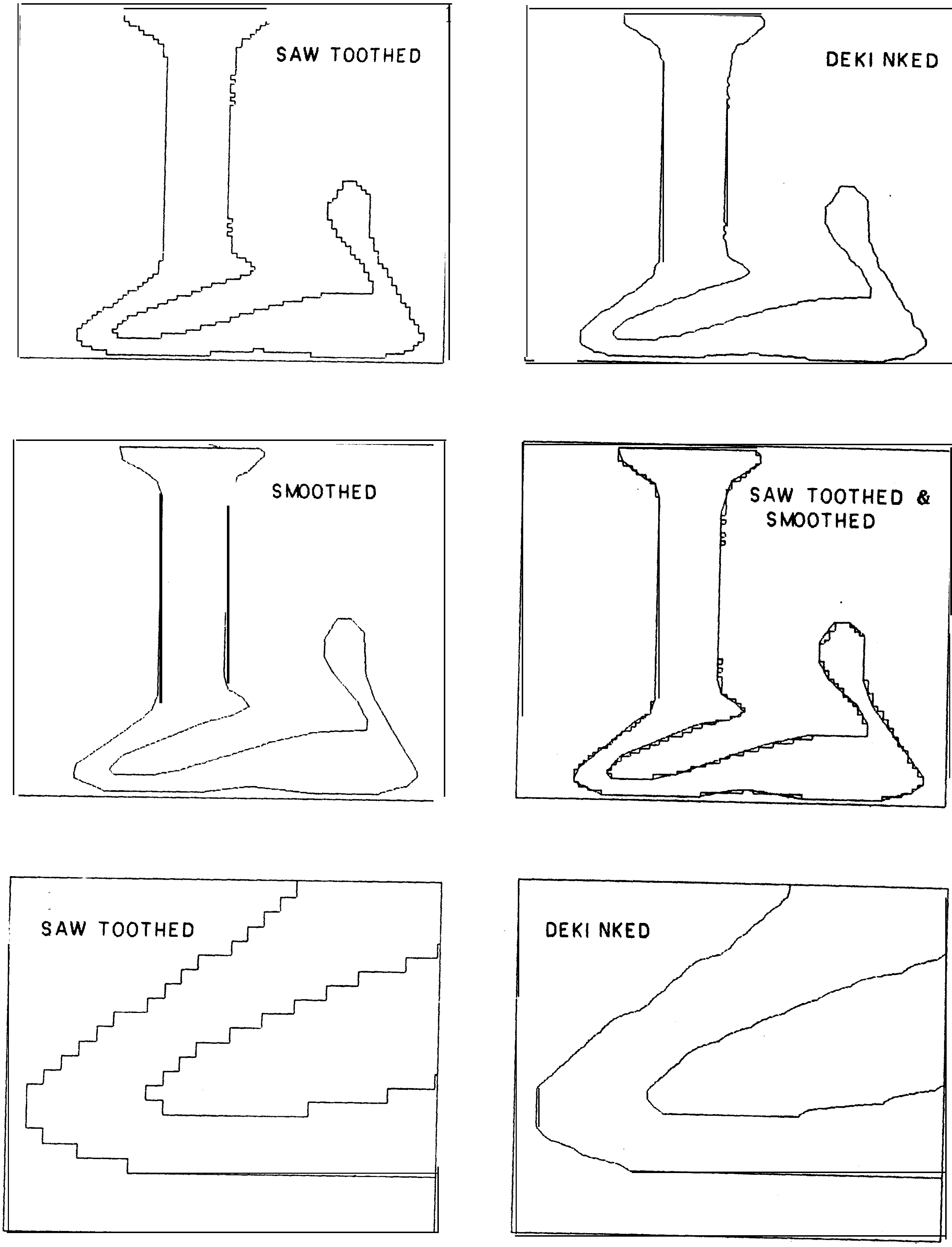


FIGURE 6 - SAW TOOTH DEKINKING ILLUSTRATED.



2. CONTOURING.

Contouring, converts the bit arrays HSEG and VSEG into vectors and polygons, The contouring it self, is done by a single subroutine named MKPGON, make polygon. When MKPGON is called, it looks for the upper most left non-zero bit in the VSEG array. If the VSEG array is empty, MKPGON returns a NIL. However, when the bit is found, MKPGON traces and erases the polygonal outline to which that bit belongs and returns a polygon node with a ring of vectors.

To belabor the details (for the sake of later complexities); the MKGON trace can go in four directions; north and south along vertical columns of bits in the VSEG array, or east and west along horizontal rows of the HSEG array. The trace starts by heading south until it hits a turn; when heading south MKPGON must check for 'first a turn to the east (indicated by a bit in HSEG); next for no turn (continue south); and last for a turn to the west. When a turn is encountered MKPGON creates a vector node representing the run of bits between the previous turn and the present turn. The trace always ends heading west bound. The outline so traced can be either the edge of a blob or a hole, the two cases are distinguished by looking at the VIC-polygon's uppermost left pixel in the PAC bit array.

There are two complexities: contrast accumulation and dekinking. The contrast of a vector is defined as (QUOTIENT (DIFFERENCE (Sum of pixel values on one side of the vector)(Sum of pixel values on the other side of the vector}) (length of the vector in pixels)). Since vectors are always either horizontal or vertical and are construed as being on the cracks between pixels; the specified summations refer to the pixels immediately to either side of the vector. Notice that this definition of constrast will always give a positive contrast for vectors of a blob and negative contrast for the vectors of a hole.

The terms "jaggies", "kinks" and "sawtooth" all are used to express what seems to be wrong about the lowermost left polygon on page 25. The problem involves doing something to a rectilinear quantized set of segments, to make its linear nature more evident. The CRE jaggies solution (in the subroutine MKPGON) merely positions the turning locus diagonally off its grid point alittle in the direction' (northeast, northwest, sout hwcst or sout heast) that bisects the turn's right angle. The distance of dekink vernier positioning is always less than half a pixel; but greater for brighter cuts and less for the darker cuts; in order to preserve the nesting of contours. The saw toothed and the dekinked versions of a polygon have the same number of vectors. I am very fond of t-his dekinking algorithm because of its incredible efficiency; given that you have a north, south, east, west polygon trace routine (which handles image coordinates packed row, column into one accumulator word}; then dekinking requires only one more ADD instruction execution per vector !

3. NESTING.

The nesting problem is to decide whether one contour polygon is within another, Although easy in the two polygon case; solving the nesting of many polygons with respect to each other becomes n-squared expensive in either compute time or in . memory space. The nesting solution in CRE sacrifices memory for the sake of greater speed and requires a 31K array, called the SKY.

CRE's accumulation of a properly nested tree of polygons depends on the order of threshold cutting going from dark to light. For each polygon there are two nesting steps: first, the polygon is placed in the tree of nested polygons by the subroutine INTREE; second, the polygon is placed in the SKY array by the subroutine named INSKY.

The SKY array is 216 rows of 289 columns of 18-bit pointers. The name "SKY" came about because, the array use to represent the furthest away regions or background, which in the case of a robot vehicle is the real sky blue. The sky contains vector pointers;--and would be more efficient on a virtual memory machine that didn't allocate unused pages of its address space. Whereas most computers have more memory containers than address space; computer graphics and vision might be easier to program in a memory with more address space than physical space; i.e. an almost empty virtual memory.

The first part of the INTREE routine finds the surrounder of a given polygon by scanning the SKY due east from the uppermost left pixel of the given polygon. The SON of a polygon is always its uppermost left vector. After INTREE, the INSKY routine places pointers to the vertical vectors of the given polygon into the sky array.

The second part of the INTREE routine checks for and fixes up the case where the new polygon captures a polygon that is already enciaved. This only happens when two or more levels of the image have blobs that have holes. The next paragraph explains the arcane details of fixing up the tree links of multi level hole polygons and the box following that is a quotation from the appendix of Krakauer thesis [3] describing his nesting aigorit hm.

3. NESTING.

Let the given polygon be named Poly; and let the surrounder of Poly be called Exopoly; and assume that Exopoly surrounds several enclaved polygons called "endo's", which are already in the nested polygon tree. Also, there are two kinds of temporary lists named the PLIST and the NLIST. There is one PLIST which is initially a list of all the ENDO polygons on Exopoly's END0 ring. Each endo in turn has an NLIST which is initially empty. The subroutine INTREE re-scans the sky array for the polygon due east of the uppermost left vector of each endo polygon on the PLIST, (Exopoly's ENDO ring). On such re-scanning, (on behalf of say an Endol), there are four cases:

1. No change; the scan returns Exopoly; which is Endol's original EXO.
2. Poly captures Endol; the scan returns Poly indicating that endol has been captured by Poly.
3. My brothers fate; the scan hits an endo2 which is not on the PLIST; which means that endo2's EXO is valid and is the valid EXO of endo1.
4. My fate delayed; the scan hits an endo2 which is still on the PLIST; which means that endo2's EXO is not yet valid but when discovered it will also be Endol's EXO; so Endol is CONS'ed into Endo2's NLIST.

When an endo polygon's EXO has been re-discovered, then all the polygons on that endo's NLIST are also placed into the polygon tree at that place. All of this link crunching machinery takes half a page of code and is not frequently executed.

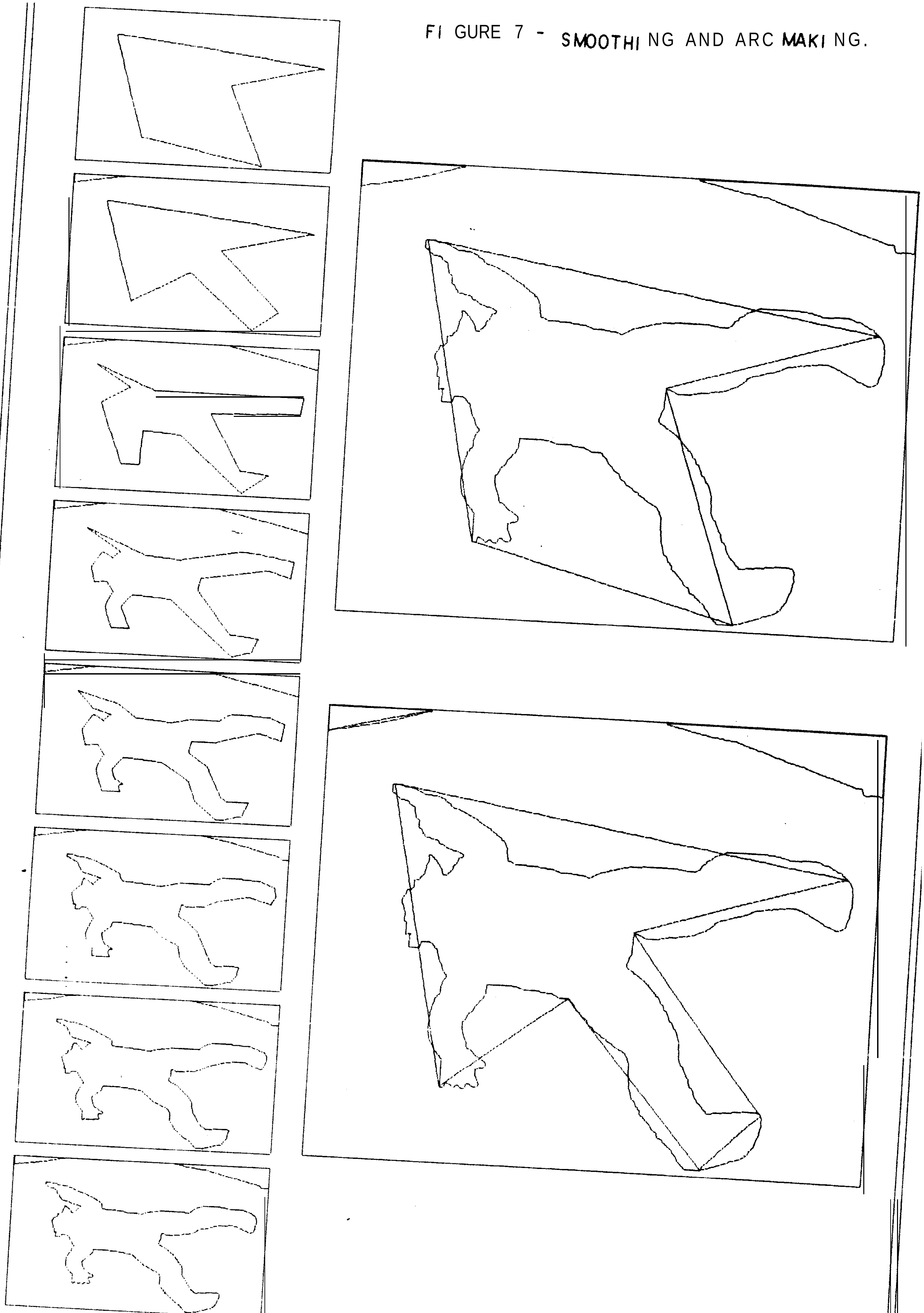
KRAKAUER'S NESTING ALGORITHM.

"The image tree is generated one threshold level at a time, starting at the highest level (branch tips). At each level, the image is scanned, and the points above the threshold are marked in a scratch array. This scatch array is then scanned for marked points. When one is found, a contiguity routine is called, which visits all marked points which can be reached from the start via a connected path. The marks are erased by this routine as it goes, and statistics are kept on the region thus generated, such as the sums of the x and y coordinates of the points, and the sum of the squares of the x and y coordinates (used to compute the center and the eccentricity). A tree node is then made up for the region, and the scan for marked points continues. A special mark is left in the scratch array for each region. When this mark is encountered during the scan at the next level, it is looked up on an association list. This establishes the link between a region and the regions which are a subset of it at the previous level - i.e. between a node and its sub-nodes."

"The contiguity scan is the most complex program. It works by leaving directional pointers in the scratch array. These are three-bit codes denoting one of the eight possible neighboring points. The contiguity scan is always started at a point which is on the bottom edge of the region. It traces along this edge to the right by moving from one marked point to the next, but always keeping an un-marked point to the right side. As it goes, it erases the marks, so that for a region with smooth boundaries, it will follow a spiral path to the center, "eating up" the marks as it goes, like a lathe with the tool continually advancing into the work."

"As the contiguity routine scans, it lays down back pointers in the scratch array which enable it to retrace its path back to the start. If a dead end is reached (no more marked neighbors), it traces back along this path, looking for marked points to the right. There can be no marked points on the left side while backtracking, since this was the right side on the way out, and the outgoing scan stayed as far to the right as possible. If a marked point is found on the backtrace, it is replaced with a pointer to the adjacent path already traced out, and then a new path is traced as if this were a new starting point. When the backtrace reaches the original starting point, the contiguity scan is completed. The effect of this algorithm is to construct a tree of pointers in the scratch array, with the starting point at the root. All points which can be reached via a connected path from the starting point will be a part of this tree."

FIGURE 7 - SMOOTHING AND ARC MAKING.



4. SMOOTHING.

In CRE the term “smoothing” refers more to the problem of breaking a manifold (polygon) into functions (arcs), rather than to the problem of fitting functions to measured data. The smoothing step, converts the. polygons of vertical and horizontal vectors into polygons of arcs. For the present the term “arc” means “linear arc” which is a line segment. Fancier arcs: circular and cubic spline were implemented and thrown out mostly because they were of no use to higher processes such as the polygon compare which would break the fancier arcs back down into linear vectors for computing areas, inertia tensors or mere display buffers.

Smoothing is applied to each polygon of a level. To start the smoothing, a ring of two arcs is formed (a bi-gon) with one arc at the uppermost left and the other at the lowermost right of the given vector polygon. Next a recursive make arc operation, MKARC, is is appled to the two initial arcs. Since the arc given to MKARC is in a one to one correspondece with a doubly linked list of vectors; MKARC checks to see whether each point on the list of vectors is close enough to the approximating arc. MKARC returns the given arc as good enough when all the sub vectors fall within a given width; otherwise MKARC splits the arc in two and places a new arc vertex on the vector vertex that was furthest away from the original arc.

The two large images in figure-7, illustrate a polygon smoothed with arc width tolerances set at two different widths in order to show one recursion of MKARC. The eight smaller images illustrate the results of setting the arc width tolerance over a range of values. Because of the dekinking mentioned earlier the arc width tolerance can be equal to or less than 1.0 pixels and still expect a substantial reduction in the number of vectors it takes to describe a contour polygon.

A final important smoothing detail is that the arc width tolerance is actually taken as a function of the highest contrast vector found along the arc; so that high contrast arcs are smoothed with much smaller arc width tolerances than are low contrast arcs. After smoothing, the contrast across each arc is computed and the ring of arcs replaces the ring of vectors of the given polygon. (Polygons that would be expressed as only two arcs are deleted).

FIGURE 9 - POLYGON COMPARE AND LINK.

FIGURE 9A.

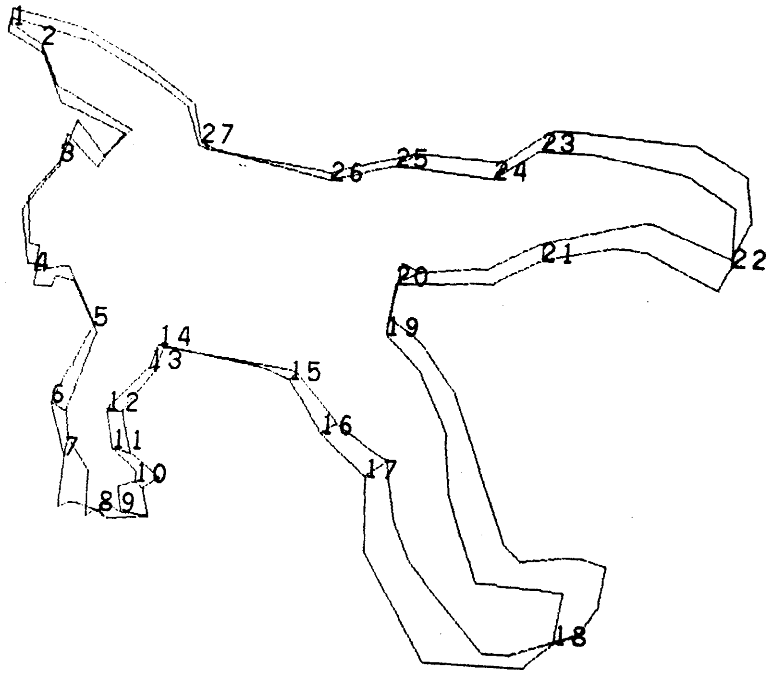


FIGURE 9B.

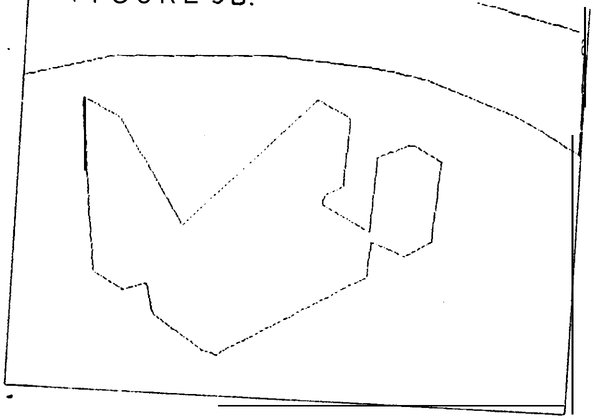
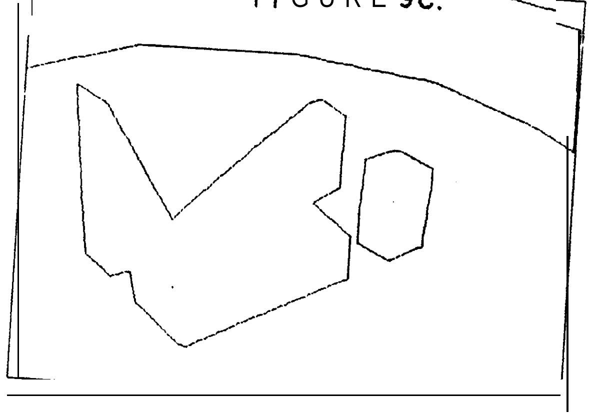


FIGURE 9C.



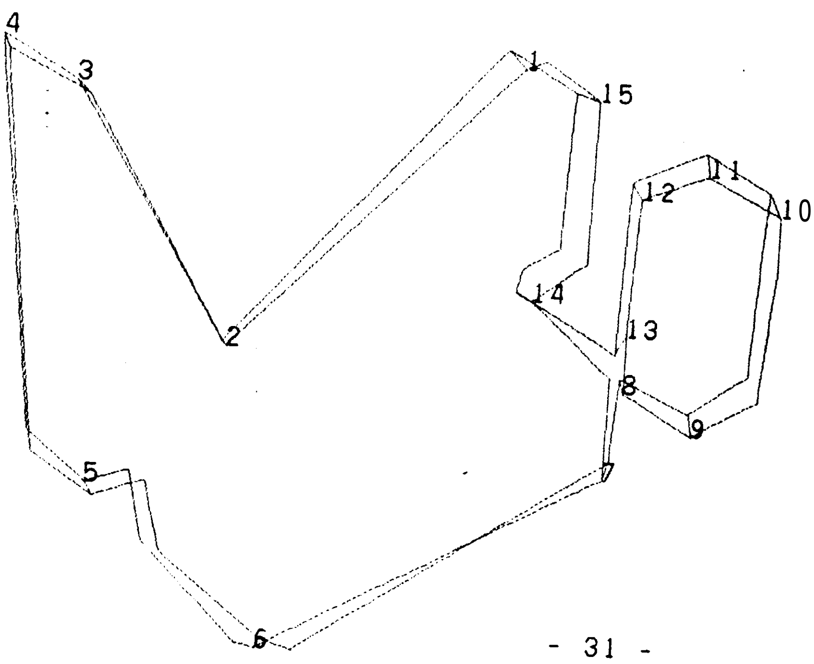


FIGURE 90.

5. COMPARING.

The compare step of CRE, CMPARE, connects the polygons and arcs of the current image with corresponding polygons and arcs of the previous image. CMPARE solves the problem of correlating features between two similar images and is composed four sub sect ions:

1. make shape nodes for polygons.
2. compare and connect polygons one to one.
3. compare and connect polygons two to one.
4. compare and connect vertices of connected polygons,

First, the shape nodes of all the polygons of an image are computed. The shape node contains the center of mass and the lamina inertia tensor of a polygon. The lamina inertia tensor of a polygon with N sides is computed by summation over N trapezoids. The trapezoid corresponding to each side is formed by dropping perpendiculars "up" to the top of the image frame; each such trapezoid consists of a rectangle an a right triangle; since the sides of polygons are directed vectors the areas of the triangles and rectangles can be arranged to take positive and negative values such that a summation will describe the interior region of the polygon as positive. The equations necessary for computing the lamina inertia tensor of a polygon are collected in a table in the postscripts to this paper and were derived by using Goldstein's Classical Mechanics [1] as a reference. The meaning of the inertia tensor is that it characterizes each polygon by a rectangle of a certain length and width at a particular location and oriention; and of further importance such inertia tensors can be "added" to characterize two or more polygons by a single rectangle. It is the lamina inertia tensor rectangles that are actually compared by CRE.

Second, all the shapes of the polygons of one level of the first image are compared with all the shapes of the polygons of the corresponding level of the second image for nearly exact match. The potentially (M*N/2) compares is avoided by sorting on the center of mass locations. In CRE, which is intended for comparing sequences of pictures of natural scenes; match for center of mass location is tested first and most strictly, followed by match for inertia. Pointers between matching polygons are placed in the time link positions of the polygon nodes and the polygons are considered to be mated in time.

5. COMPARING.

Third, all the unmated polygons of a level are considered two at a time and a fusion shape node for each pair is made. The potentially (N*N/2-N) fusion shapes are avoided because there is a maximum possible unmated inertia in the other image; if there are no unmated polygons in one image then the extra polygons of the first image can be ignored. In the event where there are unmated polygons in corresponding levels of the two images, the fusion shapes of one are compared with the polygon shapes of the other. The fusion (fission) compare solves the rather nasty problem, illustrated in figures 9A and 9B of linking two contour polygons of one image with a single contour polygon in the next image.

Fourth, the vertices of polygons mated in time are compared and mated. To start a vertex compare, the vertices of one polygon are translated, rotated and dilated to get that polygon's lamina inertia tensor coincident with its mate (or mates). Conceptually, each vertex of one polygon is compared with each vertex of the other polygon(s) and the mutually closest vertices (closer than an epsilon) are considered to be mated. Actually the potential (N*M) compares is avoided by a window splitting scheme similiar to that used in hidden line elimination algorithms (like Warnock's),

The results of vertex compare and mate are illustrated in figures 9A and 9D; the compare execution takes less than a second on images such as the pump, blocks, and dolls that have appeared in this paper. The applications of this compare might include the aiming of a pixel correlation comparator (such as Quam's); recognition and location of an expected object; or the location and extent of an unknown object. It is this latter application that will be described in my forthcoming thesis.

III. USING CRE.

PRIMER ON RUNNING CRE.

Single Image Contouring.

The Stanford copy of CRE is run by typing "R CRE". CRE displays only on a III console, however it will work (without displays) when run from a Data Disc console. The command scanner is a simple character jump table; the command scanner will type an asterisk when it is listening for teletype input. Carriage returns following commands are unnecessary but harmless; most commands signal their completion by displaying something or by typing a carriage return. Some commands require arguments or file names. The question mark, "?", command will display a summary of all the other commands,

Command characters may be modified by the control and meta shift keys; such keying will be indicated in this document by the prefixing the characters "α", "β", and "ε" to indicate control, meta or both rneta-control-shift keying respectively.

The command "T" will take a four bit television picture from camera number one. The command "H" will display a histogram of the television picture. The command character SPACE will refresh the image you had before the histogram display. The command "C" followed by a list of octal numbers followed by a carriage return will make a contour image and display it. Thus the teletype discourse for taking and contouring a single television image should have the following appearance:

```
1C

.R CRE

*T
*H
*C20 40 60
*
```

All the images in this document were made with 3 or 7 equally spaced contours; for which cases the commands "Q" and "αQ" will automatically specify contour cuts are 20, 40, 60 or 10, 20, 30, 40, 50, 60, 70 respectively.

PRIMER

IMAGE INPUT, OUTPUT AND XGP'ing.

After you have an image and its contours; you can save one or the other or both on disk files; or print one or the other. The "O" command will output a video image file, in the new hand-eye 200 octal word header format. The "I" will input a video image from such a hand-eye file; if the file is not 216 by 288, then the center of the image will be placed coincident with the center of a 216 by 288 window and the image will be repacked with undefined pixels set to zero. Both the "I" and the "O" commands will ask for a filename; if an extension is not explicitly given the default extension "TMP" will be used. The "αO" command will output the CRE data structure and the "αI" command will input CRE data structure, naturally the default extension is "CRE".

The "X" command will output a video image to the XGP. The "ϵC" command followed by a list of octal number-s will output the HSEG and VSEG; raw vector contours, to the XGP. The "P" command will output the currently displayed III buffer, the default extension is "III". Finally, the "J" command enhances the contrast of an image for the sake of its appearance on the XGP.

INTERACTIVE (MANUAL) MULTI IMAGE PROCESSING.

Taking or inputing new television images, and contouring them using the "C" command or the "Q" command will form a film data structure. images can be explicitly compared and linked by typing "M" match command which links the latest image with the immediately previous image. The "Z" command will zero the data structure of all images.

AUTOMATIC MULTI IMAGE PROCESSING.

The "A" command is for automatic turn table perception, CRE takes 64 pictures from camera #3 while rotating the turn table, outputs a file and exits (returning control to the 3D geometric editor). The turn table is manually moved small amounts by the four possible "Y" commands: "Y", "αY", "βY", and "ϵY". Numeric absolute and relative positioning of the turntable is under the "U" command; the details of which are still being developed.

CRE TELETYPE COMMANDS

VIDEO COMMANDS

T	Take a 4-bit television picture.
αT	Take a 6-bit television picture.
S	Select camera number, default is camera #1.
αS	Set TCLIP, default is 0.
βS	Set BCLIP, default is 7.
‹S	Shrink node space. Calls node storage compactor.

The two command characters "T" and "S" control live video camera input. The default camera is camera #1 on the Cohu camera on the hand eye table. Camera #0 is the Cart Receiver, camera #2 is the sierra hand eye camera, and camera #3 is one or the other old brown cameras depending on which coax is plugged up, the brown camera near Ill23 is the Font Camera and the brown camera near the turntable is the GCOMED Camera.

INPUT OUTPUT COMMANDS

I	Input TMP file. Television image from disk file.
αI	Input CRE file. Contour film from disk file.
0	Output TMP file. Television image to disk file.
α0	Output CRE file. Contour film to disk file.
X	Output video image to XGP.
P	Output III file. III buffer for calcomp plotter.
‹C	Output VIC contour edges to XGP.
.	This command requires a list of octal numbers.
J	Contrast enhancement for the sake of XGP appearance.
#	Type twenty CRLF's to clear page printer.
?	Display help summary of CRE commands.

IMAGE CONTOURING COMMANDS

C	Cut at given threshold levels.
Q	Cut at equally spaced conttours, three cuts: 20, 40, 60.
αQ	Seven cuts:10, 20, 30, 40, 50, GO, 70.
E	Enable all CRE processing.
D	Disable all steps except contouring.
M	Compare and mate match current image wrth previous.
W	Enter- Arc Width Table alter mode,

CRE TELETYPE COMMANDS

NODE FOLLOWING COMMANDS

+		Fetch film node.
!		Flush node display.
, .	CW,,CCW	Fetch Ring links.
< >	DAD,,SON TYPE,,RELLOC	Fetch Tree links,
u n	ENDO,,EXO	Fetch nested polygon tree links,
≤ ≥	ALT,,NCNT	Fetch alternate shape or arc link.
⊂ ⊃	NGON,,PGON	Fetch nested polygon tree links.
v A	NTIME,,PTIME	Fetch time line links.

These 14 commands allow detailed inspection of the CRE data structure by showing the contents of a node. Data halfwords of a node are displayed in octal; link halfwords are displayed prefixed with a letter indicating the type of node being pointed at; a zero link is displayed as "NIL".

The FILM node, which is the root of the whole data structure is fetched and displayed by the "+" command. From the Film, the ">" command can be used to get SON(FILM) which is always the first image, and ">" command of an image will get a level and ">" of a level will get a polygon. Vectors and polygons are intensified when their contents are being displayed. The exit command is "!", which leaves the screen less cluttered.

WINDOW SCROLLING COMMANDS

;	Move camera left.
.	Move camera right.
(	Move camera down.
)	Move camera up.
	Zoom out, shrink displayed image.
*	Zoom in, expand displayed image.
α.Z	Reset scrolling window to it initial position and size.
/	Halve strength of scrolling delta.
\	Double strength of scrolling delta.
↔	Single step displayed image forwards.
α.↔	Single step displayed image backwards.
β↔	Run film display forwards.
(↔	Run film display backwards.

The first several commands allow minute examination of the image by magnification and window positioning. The command character "↔" allows single stepping thru the film of images or continous display of the film forwards or backwards.

CART DRIVING COMMANDS

F Drive forwards.
B Drive backwards.
L Turn wheels hard left.
R Turn wheels hard right.
αL Pan camera left,
αR Pan camera right.
SPACE Stop the cart.
RETURN Exit cart command mode.

First, and most important is understanding how to stop the cart. The teletype halt command is SPACE; also any character other than "F", "B", "L", or "R" will stop the cart. Cart commands are passed first from a teletype to the PDP-10; then to the PDP-6; then over a citizens band, 27.045 megahertz, radio link to the cart control logic. When communication is lacking between entities in the chain of command the lower entity times out and causes the cart to halt. The cart control logic times out in a fifth of a second if it does not hear from the POP-6; the PDP-6 times out in less than a minute if it has not heard from the PDP-10; the PDP-6 stops broadcasting cart commands if it detects the death of the PDP-10; the PDP-10 job times out after 5 minutes of not hearing from the teletype and kills the PDP-6 spacewar job.

Second, and of occasional interest is understanding how to make the cart go. The command "F" will make the cart go forwards; and the other commands will cause action as mentioned in the table. If the cart fails to move; all its switchs should be check for being in the ON or AUTOMATIC or FAST position; all its plugs should be plugged in; and its batteries should be checked. Recently cart failure had been most often caused by the radio transmitter in the Kludge Bay. Check to see that the transmitter is turned on and that the PDP-6 is running. By the end of the year (1973), a new cart radio controler will be installed by Hans Moravec, and these commands will be updated.

CART HARDWARE DIAGONOSTIC

V Enter diagonostic listen loop.
RETURN Exit diagonostic listen loop.

NUMERALS: 0,1,2,3,4,5,6,7 send direction relay bits.
CHARACTERS: H,A,B,C,D,E,F,G send action relay bits.

The cart diagonostic listen loop simply takes the low order four bits of a non-carriage return ASCII character and broadcasts them to the cart. The cart decodes four bit radio command bytes into six relays; commands 0 thru 7 set the pan, drive, or steering direction relay repective to bits 4, 2 and 1; commands A thru G set the pan, drive, or steering action relays respective to bits 4, 2, and 1.

SAIL INTERFACING TO CRE.

It should be possible to embed the CRE machine code under a SAIL core image; however I do not intend to do this work. For the present, the CRE interface to SAIL is only realized via a disk file transfer of the data structure. A CRE file may be read into an integer array in binary mode as illustrated below.

The first word of a CRE file is the first word of. the film node which contains the size of the file in words. The film node has address 0; the next node has address 7; and so on in multiples of seven. There at-e no empty nodes in a CRE file. The following SAIL program will read in a CRE file named X:

```
COMMENT EXAMPLE OF SAIL INPUT OF A CRE FILE;
BEGIN "TEST"
        INTEGER SIZE;
        OPEN(1,"DSK",8,3,0,0,0,0);
        LOOKUP(1,"X.CRE",0);
        SIZE ←WORDIN(1);
BEGIN
        INTEGER ARRAY NODE[0:SIZE];
        ARRYIN(1,NODE[1],SIZE-1);
        RELEASE(1);
        "MAIN PROGRAM.";
END;
END;
```

After the NODE array is loaded, CRE links and data may be accessed by their document names in a reasonable node-link notation using macros like the following:

```
DEFINE CW(Q) = "(NODE[Q] LSH -18)";
DEFINE CCW(Q) = "(NODE[Q] LAND '777777)";
DEFINE DAD(Q) = "(NODE[Q+1] LSH -18)";
DEFINE SON(Q) = "(NODE[Q+1] LAND '777777)";
```

So that the first vertex of the first polygon of the first level of the first image of the film can be obtained:

```
INTEGER FILM,IMAGE,LEVEL,POLYGON,VERTEX;

FILM ← 0;
LEVEL ←SON(FILM);
POLYGON ←SON(LEVEL);
VERTEX ← SON(POLYGON);
```

The user may note that SAIL will compile three or more instructions for what is known as a PDP-10 halfword operation; also if the user converts the CRE nodes and links into LEAP items and associations then an overhead of from ten to one hundred instructions per "halfword operation" will be incurred.

LISP INTERFACING TO CRE.

It should be possible to embed the CRE machine code under a LISP core image; however I do not intend to do this work. For the present, the CRE Interface to LISP is only realized via a disk file transfer of the data structure. A CRE file may be read into LISP binary program space and accessed using the CRE nomensclature (11 link names and 13 datum names) by means of the S-Expression subroutines provided in the file CRE.LSP[CRE,BGB]. The subroutines work in both the old Stanford LISP 1.6 as welt as the newer UCI LISP and Micro Planner, PLNR. The CRE.LSP[CRE,BGB] can be loaded either by one or the other of the following two LISP statements:

```
(DSKIN(CRE,BGB)(CRE.LSP))
(INC(INPUT(CRE,BGB)(CRE.LSP)))
```

A CRE film file is read into LISP binary program space by one of the three possible INCRE formats:

```
(INCRE filename)
(INCRE filename project)
(INCRE filename project programmer)
```

Filenames should be six characters or less, projects and programmer initials should be three characters or less, the filename extension CRE is assummed and the usual PPPN defaults occur. If the input succeeds INCRE returns a value T; if the input fails INCRE returns a value NIL and prints one or the other of these two messages:

CRE FILE NOT FOUND.
CRE FILE REQUIRES 00000 MORE WORDS OF BINARY PROGRAM SPACE.

After a sucessful INCRE; the film, image, level, polygon, arc and vector nodes are referred to by integer-s using the 11 Link Fetch Subroutines:

```
(CW node)(CCW node)(DAD node)(SON node)(ENDO node)(EXO node)
(ALT node)(NGON node)(PGON node)(NTIME node)(PTIME node)
```

The film node's address is the integer 0, zero. So that the expression (SETQ V3(CCW(CCW(SON(SON(SON(SON 0))))))) will retrieve the lower right hand corner of the border polygon of the -1 level of the first image of the film. The 13 CRE.LSP datutn fetch subroutines are:

```
(ROW node)(COL node)(CRETYPE node)(RELOC node)
(CNTRST node)(NCNT node)(ZDEPTH node)(PERM node)(AREA node)
(MXX node)(MYY node)(MZZ node)(PXY node)
```

This is sample output from the Xerox Graphics Printer.
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0 123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

This is sample output from the Xerox Graphics Printer.
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

This is sample output from the Xerox Graphics Printer.
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0 123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

This is sample output from the Xerox Graphics Printer,
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0 123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

This is sample output from the Xerox **Graphics Printer.**
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0 123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

This is sample output from the **Xerox Graphics** P
rinter,
↓∝/β∧¬∈π λ∞∂⊂⊃∩∪∀∃⊗↔_→~≠≤≥≡∨
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
'abcdefghijklmnopqrstuvwxyz{|fl}

USING TVFONT - draft.

Introduction.

TVFONT is a version of CRE (January 1973) that was specialized to the task of converting television images into type fonts for the XGP, Xerox Graphics Printer. The original idea was to demonstrate the utility of a polygon representation for scaling, smoothing and editing typographical glyphs; the resulting hack (demonstration program) was extended and developed by Tovar Mock into the program called TVFONT. Accordingly, the main idea of TVFONT is to convert video rasters into polygons, to edit and scale the polygons, and to convert the polygons back into bit rasters.

This section IV, will be available as a TVFONT user manual in another six months: it is presented here to give the would be user a start, and the general reader a sample of the design and extent of TVFONT.

The figure on page 41 is an example of expanding and contracting a font without manual touching up. The top sample is the original (BDR40 from CMU). The remainder have been generated by TVFONT. The expansion or contraction was done by converting fonts from bit matrices into a polygonal representation, multiplying by the appropriate constant and reconverting back into a bit representation. The following paragraph is an example of a font made from television pictures:

Как Вам нравится наша новая широко-печать? Она называется *XGP (Xerox Graphics Printer)* и сделана фирмой *Xerox* на основании машины *LDX*. Центр иследования *Xerox* в *Palo Alto* сдала нам машину безплатно, чтобы исследовать ее применения. *XGP* получает из ЦВМ до 1700 разрядов каждый пять миллисекундов. Это *scan line* как по телевидении. Буквы сделаны из точек программой. Страница состоит из 1700x2200 = 3,740,000 разрядов. Из-за этого, ЦВМ должна работать очень быстро. Или она должна иметь в оперативной память около 100,000 слов, или она должна получать из диска довольно большой буфер очен регулярно, потому что, когда бумага начинает двигать в широко-печати, она не может остановиться до 22 дюжиных. Принципе машины - такой же как у обыкновенной машины *Xerox*.

Как Вы видите, машина очень гибка. Возможно ползоваться любым алфавитом, в любом размере, и кроме того, возможно печатать иллустрации. Наверно Вы тоже заметили, хорошая широко-печать не помогает моему плохому русскому языку. Сейчас вернюсь на английский язык.

TVFONT PRIMER - (draft).

TVFONT is on the system, and can be run by typing "R TVFONT" at a III display console. At present, III #23 is next to a camera setup for making fonts. The process of making a new XGP font or altering an old one will be explained in six steps:

1. Raster input: get a video image or an old font,
2. Contouring: make polygons.
3. Polygon editing: delete, scale, position and alter,
4. Polygon I/O: save and restore polygons.
5. Font output: make new font and output font file.

Complexity arises in that there is more than one way to do each step, there are default arguments and switchs which the user may alter, there are ways to save and restore intermediate results, and there are quite a few different display modes and display diagonostics. The TVFONT command scanner resembles that of TVED and E; (as well as CRE and GEOMED); the command scanner types an astcrisk "*" when it is in its top most listen loop waiting for a single command character. The command character may be modified by the META and CONTROL keys which will be abbreviated as "a" "β" and "ε" for CONTROL, META, and META-CONTROL respectively. Many' commands in turn require arguments such as numbers or file names. Finally the "x" command waits for an extended command name of several characters. which is called an extended command.

This first explanation will present a way of making a new font using the fewest commands.

Raster Input and Contouring:

1. "T" take television picture.
2. "H" Display histogram of television picture.
3. "C24" Cut at intensity level 24,

Get the Font Camera looking at a single letter in a font book. Use a black piece of paper with a square cut out as a mask to isolate the letter. The "T" command will take a television picture. The "H" command will display a histogram of the television picture, showing how many points of the image were 0 intensity, (total black) and how many points of the image were 77 intehsity, (total white). A picture of a black glyph on a white background surrounded by a black mask should yield a histogram with two peaks.

Next the "C" command followed by an octal number followed by a carriage return: contours the image at the given octal intensity cut threshold. That is all the points of the image above the threshold are inside of a polygon. The intensity value of the lowest valley between the two peaks of the histogram is probably the best cut value (and is probably the octal number 24 or 30). The cut command, will display the polygons that are made.

Polygoh Killing.

4.	"ϵ+"	Fetch 1st polygon of 1st image of the film.
5.	"K"	Kill a polygon.
6.	"."	ring around the polygons of an image.
7.	"|"	flush node display,

Given an image of polygons corresponding to one letter, undesired polygons can be deleted by using the "K" command and the node link display commands. To start, the "ϵ+" will intensify the first polygon of the image's polygon ring; from there the "." commands will intensify the next polygon of the ring; the "K" command will delete the presently intensified polygon and fetch the next polygon.

A font corresponds to a film. An image corresponds to a letter. After taking a series of iaages, and deleting undesired polygons a font file can be made using:

Making and Outputing a Font File.

8.	"X"CENTER	Center all the images of the film.
9.	"Q"	Make font bit rasters.
10.	"ϵO"	Output font file.

The "X"CENTER command is an extend mode command and requires both hitting "X" and typing out the word "CENTER" followed by a carriage return. The "Q" will cause a bit raster to be made for the interior portions of each image of the film; if an image node does not have an associate ASCII code then the user will be requested to supply one. The "ϵO" will ask for a font filename and will output a font file in the Stanford Format.

Testing a new Font File.

11. .XGP FILE/FONT=NEWFNT.FNT[XGP,BGB]

The above monitor command will print a FILE with a new font. The user must specify his PPPN because the default is [XGP,SYS].

TVFONT COMMAND SUMMARY

A	ASSIGN ASCII CODE TO IMAGE.
B	EXPAND/CONTRACT BY CONSTANT
αB	EXPAND/CONTRACT IN Y DIRECTION
βB	EXPAND/CONTRACT IN X DIRECTION
εB	SLANT CHARACTER (1/2 SLANTS TO 45 DEGREE ANGLE)
C	MAKE THRESHOLD CUT.
cc	MAKE POLYGON IMAGE OUT OF BIT REPRESENTATION OF FONT.
D	ENABLE/DISABLE DELETION OF BABY POLYGONS (DEFAULT IS OFF).
F	LOCATE NEAREST POINT, εF USE LIGHT PEN
G	LEVEL OF CORRESPONDING CHARACTER CODE
H	HISTOGRAM, "αH" "βH" BI-MODAL CUT,
I	INPUT TV PICTURE FROM DISK.
αI	INPUT CRE FILE
K	KILL IMAGE, POLYGON OR VERTEX
L	SHOW LAST BIT IMAGE
αL	SHOW CHARACTER FROM FONT IN FNTSEG
M	MOVE POLYGON TO NEXT IMAGE.
αM	MOVE TO NEW IMAGE
βM	MIDPOINT LINE
εM	MUNG ONTO GRID POINT (AS SEEN IN εY)
N	NEXT IMAGE
αN	PREVIOUS IMAGE
βN	REPEAT NEXT IMAGE UNTIL A CHARACTER IS TYPED
εN	REPEAT PREVIOUS IMAGE UNTIL A CHARACTER IS TYPED

TVFONT COMMAND SUMMARY

O	OUTPUT CAREYE FILE.
αO	OUTPUT CRE FILE
εO	OUTPUT FONT FILE
P	PLOT OUTPUT FILE.
Q	MAKE FONT
αQ	MAKE 1/2 SIZE FONT
R	DISPLAY BIT MATRIX FOR THIS CHARACTER.
αR	ROTATE IMAGE, LEVEL OR POLYGON (ANGLE IN RADIANS)
S	SMOOTH
αS	SMOOTH AND KILL VIDEO INTENSITY CONTOUR
βS	REPEAT 'S' FOR EACH IMAGE
εS	REPEAT 'αS' FOR EACH IMAGE
T	TAKE A TV PICTURE
αT	TAKE A TV PICTURE, SETTING CLIP LEVELS AUTOMATICALLY
V	CREATE VERTEX AT CENTER
αV	CREATE NEW VERTEX AT CURRENT VERTEX
βV	CREATE NEW VERTEX IN NEW IMAGE
W	CENTER IN THE WINDOW.
αW	CENTER Y-POSITION ONLY.
βW	CENTER X-POSITION ONLY.
εW	MOVE POINT SPECIFIED BY LIGHT PEN TO CENTER.
X	XTEND MODE COMMANDS
Y	DISPLAY SMOOTHED FORM
βY	DISPLAY VIDEO INTENSITY CONTOUR
αY	DISPLAY BOTH OF ABOVE
εY	DISPLAY VIDEO INTENSITY CONTOUR MUNGEO ONTO PIXELS
Z	NO-OP
αZ	RESET LOGICAL CAMERA POSITION
βZ	RESET DISPLAY

TVFONT COMMAND SUMMARY

+	Fetch film node.
α+	Fetch first image node from film.
β+	Fetch first level from film.
(+	Fetch first polygon from film.

IF A NODE IS CURRENTLY BEING DISPLAYED, THESE COMMANDS AFFECT THAT NODE, OTHERWISE THEY AFFECT THE CAMERA (VIEWERS) POSITION. <CONTROL> MULTIPLIES BY 2, <META> MULTJPLIES BY 4.

;	MOVE LEFT (←) BY DELTA
:	MOVE RIGHT (→) BY DELTA
(	MOVE UP BY DELTA
)	MOVE DOWN BY DELTA
/	DIVIDE DELTA BY 2
\	MULTIPLY DELTA BY 2

THESE COMMANDS AFFECT THE CAMERA (VIEWERS) POSITION.

*	INCREASE MAGNIFICATION BY DELTA
	DECREASE MAGNIFICATION BY OELTA

THESE COMMANDS CHANGE NODE BEING DISPLAYED.

.	FETCH COUNTER CLOCKWISE NODE IN RING.
.	FETCH CLOCKWISE NODE IN RING.
<	FETCH FATHER OF NODE
>	FETCH SON OF NODE
≤	FETCH ARC [OF POLYGON OR VERTEX3
3	FETCH POLYGON [OF VERTEX1
A	EQUIVALENT TO '<.>'
V	EQUIVALENT TO '<.>'
!	FLUSH NODE DISPLAY

THESE COMMANDS AFFECT THE PUSHDOWN LIST

U	PUSH NODE BEING DISPLAYED ONTO STACK
∩	POP NODE OFF STACK AND DISPLAY IT
↔	SWAP NODE BEING DISPLAYED WITH TOP OF STACK

TVFONT'S EXTENDED COMMANDS.

ARCWID

Set smoothing constant. This is the maximum distance a vertex may from a arc before it i's split into tuo arcs. See description of smoothing algorithm on page XX.

BABYKILL

Toggle flag which causes baby polygons (those consisting of only one pixel) to be killed)

CAMERA

Select a different camera number.

CENTER

Center all images. It is equivalent to the command 'W' applied to each image and uses the same control bits.

DOT

Invoke DOT if present, return with αP..

DISPLAY

Enable display.

-DISPLAY

Disable display. TVFONT spends a significant amount of time putting up the display.

EXIT

Exit to moniter.

GRID

Enable display of grid. Grid is some multiple of pixel size, dependent on camera focal length. It is useful of lining up characters.

-GRID

Disable display of grid.

HELP:

Display help file.

HOLE

Change a polygon into a hole.

KILARC

Kill arcs vectors. This allows several degrees of smoothing to be tried in conjunction with the ARCWID command.

TVFONT'S EXTENDED COMMANDS,

KILVIC
Kill video intensity contours-and replaces them with arcs.

MUNG
Force all vertices of current polygon or level onto pixel boundaries. This has a permanent effect as opposed to 'εY' command which only displays them that way.

ORTHMUNG
ORTHMUNG forces vertices which appear to be form right angles onto pixel boundaries. This is attempt to counter the rounding effect of dekinking on sharp corners as are generated by reading a font.

POLYGON
Change a hole into a polygon

POPJ
Leave TTY loop. Used for debugging.

READFONT
Convert font which has been read into the font segment into polygonal representation, displaying each character as read.

SCALE
Scale all images by constant. Equivalent to the command 'B' applied to each image.

SLANT
Slant all images by constant . Please see command 'εB' for a more complete description.

SORT
Sort images on film according to ASCII code. This is for convienence in looking a fonts sequentially. The 'G' command. is recommended for finding specific characters.

XEROX
OUTPUT TV IMAGE TO XGP

XSCALE
Scale all images by constant in the X direction. Equivalent to the command 'αB' applied to each image.

YSCALE
Scale all images by constant in the Y direction. Equivalent to the command 'βB' applied to each image.

TVFONT NODE FORMATS - JAN 1973.

VERTEX/ARC NODE,		POLYGON/REGION NODE,	
0	VERTEX-RING	0	POLYGON-RING.
1	ROW,,COL	1	DAD,,SON
2	TYPE,,RELOC	2	TYPE,,RELOC
3	- ,, -	3	- ,, -
4	ARC,, -	4	ARC,,NCNT
5	- ,,PGON	5	- ,,PGON
6	RT SEG,,LT SEG	6	- ,, -

IMAGE NODE.		LEVEL NODE,	
0	IMAGE-RING	0	LEVEL-RING
1	- ,, SON	1	- ,,SON
2	TYPE,,RELOC	2	TYPE,,RELOC
3	- ,, -	3	- ,, -
4	- ,, -	4	- ,,NCNT
5	- ,, -	5	- ,, -
6	- ,, -	6	- ,, -

FILM NODE.		EMPTY NODE,	
0	CORESIZE	0	- ,,AVAIL
1	- ,,SON	1	- ,,
2	TYPE,,RELOC	2	TYPE,,RELOC
3	- ,,AVAIL	3	- ,, -
4	BLOCK COUNT	4	- ,, -
5	- ,, -	5	- ,, -
6	- ,, -	6	- ,, -

SEGMENT NODE.	
0	SEGMENT RING
1	- ,, -
2	TYPE,,300003
3	LDEL,,RDEL
4	LCOL,,RCOL
5	LROW,,RROW
6	LT,,RT

SUMMARY OF LAMINA INERTIA TENSOR EXPRESSIONS.

RECTANGLE'S LAMINA INERTIA TENSOR ABOUT ITS CENTER OF MASS.

```
MXX   ←   B*B*AREA/12;      (B HEIGHT IN ROWS).
MYY   ←   A*A*AREA/12;      (A WIDTH IN COLUMNS).
MZZ   ←   MXX + MYY;
FXY   ←   0;
```

ORIENTED RIGHT TRIANGLE'S LAMINA INERTIA TENSOR ABOUT ITS CENTER OF MASS.

```
MXX   ←   B*B*AREA/18;      (B HEIGHT IN ROWS).
MYY   ←   A*A*AREA/18;      (A WIDTH IN COLUMNS).
MZZ   ←   MXX + MYY;
PXY   ←   -A*B*AREA/36;
```

SUMMATION OF LAMINA INERTIA TENSORS.

```
AREA  ←   (AREA1 + AREA2);
XCM   ←   (AREA1 * XCM1 + AREA2 * XCM2) / AREA:
YCM   ←   (AREA1 * YCM1 t AREA2 * YCM2) / AREA:
MXX   ←   MXX1 t YCM1*YCM1*AREA1  t
          MXX2 t YCM2*YCM2*AREA2  -  YCM*YCM*AREA;
MYY   ←   MYY1 t XCM1*XCM1*AREA1  +
          MYY2 t XCM2*XCM2*AREA2  -  XCM*XCM*AREA;
PXY   ←   PXY1 - XCM1*YCM1*AREA1  t
          PXY2 - XCM2*YCM2*AREA2  t  XCM*YCM*AREA;
```

ANGLE OF PRINCIPLE AXIS

```
PHI   ←   0.5*ATAN((MYY-MXX)/(2*PXY))
PXY   ←   0.5*(MYY - MXX)*TAN(2*PHI)
```

TRANSLATION OF LAMINA INERTIA TENSOR AWAY FROM CENTER OF MASS.

```
MXX'  ←   MXX t AREA*DY*DY;
MYY'  ←   MYY t AREA*DX*DX;
PXY'  ←   PXY - AREA*DX*DY;
```

ROTATION OF LAMINA INERTIA TENSOR ABOUT CENTER OF MASS.

```
C     ←   COSINE(PHI);
S     ←   SINE(PHI);
MXX'  ←   C*C*MXX + S*S*MYY - 2*C*S*PXY;
MYY'  ←   C*C*MYY + S*S*MXX + 2*C*S*PXY;
PXY'  ←   (C*C - S*S)*PXY t C*S*(MYY - MXX);
```

REFERENCES.

[1] GOLDSTEIN, H. 1950.
Classical Mechanics,
Addison-Wesley.
Reading, Massachusetts.

[2] KNUTH, D.E. 1968.
The Art of Computer Programing.
Volume 1, Fundamental Algorithms.
Chapter 2, Information Structures.
Addison-Wesley.
Reading, Massachusetts,

[3] KRAKAUER, L. J. 1971.
Computer Analysis of Visual Properties of Curved Objects.
Project MAC, Technical Report TR-82,
Massachusetts Institute of Technology.
Cambridge, Massachusetts 02139.

[4] ZAHN, C. T . 1966.
Tr-JO Dimensional Pattern Description and **Recognition** via curvature points.
SLAC Report 70.
Stanford Linear- Acceleration Center.
Stanford University.
Stanford, Cailfornia.