# Expressing Shape and Composition by LAVA/SILK

Howard Jones
SPOTOPS
tech@spotops.net

## ABSTRACT

This paper describes a method of comparing and sorting vector images in terms of their constituent shapes and composition. A unique numeric field, the SPOT System, generated in a spiral pathway about a reference element in a row and column matrix of elements, provides a compact notational scheme for encoding shape and compositional signatures.

## Categories and Subject Descriptors

H.3.3 **[Information Storage and Retrieval]**: Information Search and Retrieval – *Information Filtering*;

H.5.4 **[Information Interfaces and Presentation]**: Multimedia Information Systems – *Graphics*;

J.5 **[Arts and Humanities]**: Visual Arts – *Technique*;

## General Terms

Algorithms, Human Factors, Theory.

## Keywords

limn, image structure, markup, analysis, archive, XML, SVG, graphic key, similarity, pattern classification, image recognition

## 1. INTRODUCTION

Shape and composition are among the fundamental elements of visual form.

*Shape* is constituted in a configuration of 'limned' linear elements (vectors) that define and bound an assembly of spatial regions. (According to *WordNet*, "The verb 'limn' has two senses: (1) to delineate, outline; or (2) to portray, depict.")[1]

*Composition* denotes a rhythmic deployment of shapes (both 'positive' and 'negative', i.e., 'figure'/'ground') across some finite (i.e., shaped, usually rectangular) spatial context that serves as organizational frame of reference.

Perception/conception of shape and composition seems vested in a largely unconscious complex of rhythmic sentience. ('Rhythm' as used here is taken broadly as 2D specifics of 'accented repetition' within an imaged plane. The lines of a drawing, for instance, rhythmically organize and accent the expanse of their contextual frame.)

Traditional devices and techniques of painting and drawing often employ lattice lines as a frame of reference to assist construing and copying pictorial features. In Figure 1 a woodcut by Albrecht Durer (*Treatise on the Art of Measurement*, 1525) shows the utility of such a grid in helping the artist sustain accurate depiction of proportional (rhythmic) relationships while capturing a complex assembly of features into a limned image. [2]

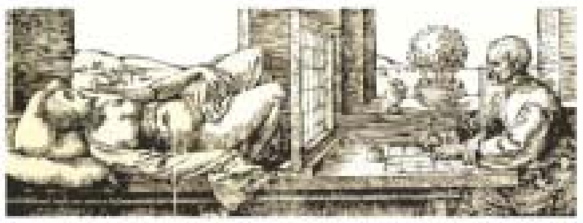


**Figure 1**

The lattice also facilitates an ancient method of copying and scaling, much used by muralists. A grid of regularly spaced lines is superimposed on a source image. Intersections of source image lines and the lattice are methodically copied to corresponding locations of a second lattice (drawn to some scaling factor) on the receiving surface. Working among transposed reference intersections, the artist fills in missing visual features by drawing into the *articulated lattice* to match those of the source image. (Until the advent of optical projecting devices, such as the 'magic lantern', lattice transposition was the standard means for transferring sketched studies onto architectural surfaces designated for murals. During the Renaissance and Enlightenment artists played all sorts of visual games in stretching and distorting an underlying lattice in anamorphic fanatasy.

## 2. EXPRESSING SHAPE BY MARKUP

*Markup* is a term from the printing industry. It denotes a collaborative process wherein drafts of text and illustrations are marked, highlighted, isolated, rearranged, and commented to guide production. Figures 3 and 4 illustrate markup of an example leaf photograph for analysis. Such markup consists of isolating and limning features (and agglomerate features) of interest. It is important to distinguish among purposes that may guide such markup. For the sake of this example, three of the basic image factors discussed below – composition, silhouette, and skeleton – are treated. Each is handled according to its specific role in the image structure. (Depending on the domain of application it may be that only one or two, or even some different class of visual order altogether, guides feature markup.) Context of use determines what features are relevant, level of detail and emphasis, and how best to serve application requirements in setting criteria of comparison and retrieval by similarity.

Figure 2

## 2.1 Fundamental Constituents of Visual Form

Figure 2 shows a simple photograph of a leaf. Simple though it may seem, it exhibits the three major configurative assemblies found in even the most complex visual depictions:

- *composition* of the image as a whole,
- *silhouette* (contour outline), and
- *skeleton* (here, the stem and veins).

Composition, silhouette, and skeletal structure are fundamental factors of image organization and meaning. *Significant shape* (silhouette, or contour) especially is crucial to iconic and symbolic functions of imagery.

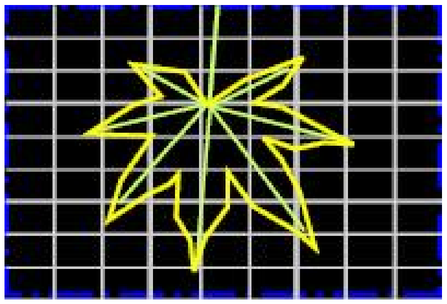


Figure 3

In Figure 3 image composition guides markup, shown within a LAVA lattice frame applied to the entire image. (Note: lattice lines are NOT markup, but are depicted here as conceptual aid.)

Since the leaf silhouette and its stem/vein skeleton are prominent features each is limned separately from the composition (in terms of its own *attentional sub-frame*, a minimum enclosing rectangle with 2-4% *padding* at the edge) as seen in Figure 4, prior to scaling onto the square LAVA matrix.

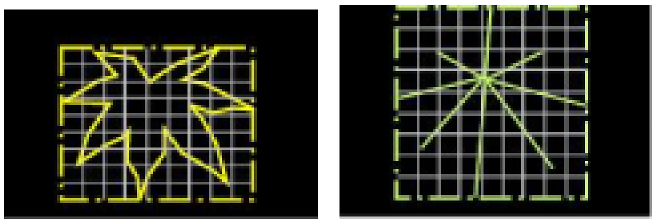


Figure 4

Figure 5 shows both the composition baseframe (a) and silhouette subframe (b) after scaling onto a square LAVA matrix. (The skeleton subframe would be handled similarly to the silhouette but is not shown here.)

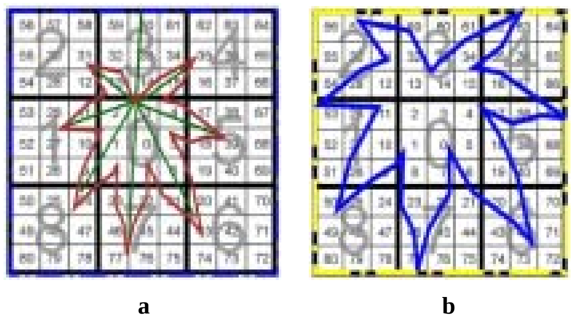


Figure 5

## 3. LAVA/SILK SHAPE SIGNATURE

*Lattice Articulated Vector Analysis* (LAVA) is a method (US Patent #6332040) of analyzing vector image shape (and composition) in terms of a standard rectangular grid of lattice lines. The LAVA engine keeps account of the number of vector/lattice intersections for each lattice cell. It also maintains for each cell a cumulative *clutch value,* which denotes the sum of the angles of inclination (gradients) of all vectors transiting its boundaries. Thus, each lattice cell serves as registry nexus of all line vectors impinging upon it.

A *String Indexed Lattice Key* (SILKey) expresses LAVA results in one or more (for complex images) compact and efficiently coded text strings that can be manipulated and compared by standard text processing. The utility of LAVA/SILK derives from observation that 'similar' linear configurations have 'similar' SILKeys. Furthermore, tolerances in criteria of comparison among SILKeys (and hence among their linked targets) can be set by *fuzzy logic* (or Bayesian filter) to impose variable criteria of compliance (ranging from a strict *exactly like* to a casual *kind of like*) in gathering result candidates for some data process. Genetic algorithms or other adaptive agents can flexibly evaluate targeted subsections among textually encoded configurations to quickly assess and manipulate a wide range of image factors. Image recognition and archival image databases are two intuitively evident applications for LAVA/SILK facilities.

## 4. SPOT DIGITAL VECTORS

The method of LAVA/SILK processing considered here uses numeric notation developed expressly for processing digital vectors, the *Spiral Position and Orientation Translator (SPOT) System*. [3] Understanding SPOT facilitates effective use of the LAVA/SILK method and notation derived from it.

SPOT notation is derived from regularities observed in the numeric field generated by a spiral pathway about a reference element in a row and column matrix of elements, as in Figure 6.

The reference element is 0; each element along the pathway is designated by an integer *tallynumber*, T. Each complete circuit constitutes a *range*, R. R(0) = 0; R(1) = 1...8, R(2) = 9…24, etc. Note that *even* square tallynumbers are aligned along a diagonal up and to the right of the reference element 0. Similarly, *odd* square tallynumbers are aligned along a diagonal down and to the left. A SPOT value, S, is the square root (*tallyroot*) of T. The stepped boundary between odd and even squares constitutes the SPOT *major diagonal*, 0.00. Moving along the spiral pathway, the fractional component of S approaches a limit at each of the axes and diagonals, namely .00, .25, .50, .75, and back again to .00.

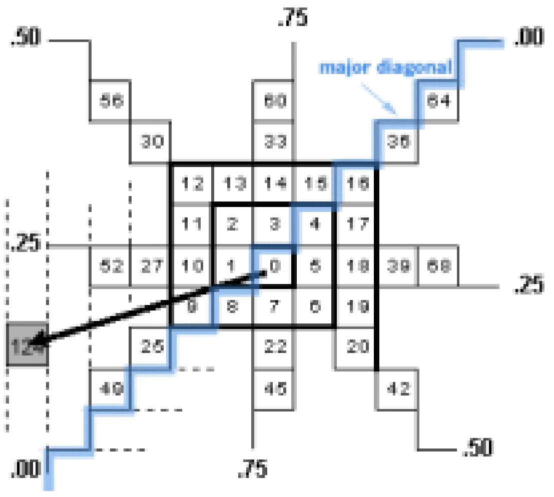


**Figure 6**

SPOT vector S = 11.1355…, (square root of T = 124)
where R = 6; M = 11; G = 0.1355…

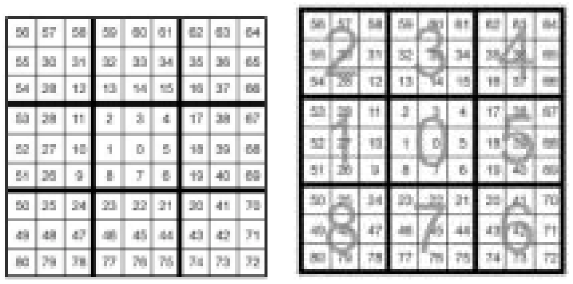
For a line vector on a digital bitmap S expresses its *magnitude and orientation* in the integer portion, M, which denotes both "up-left / down-right" orientation (in relation to the major diagonal) of the vector as well as its distance (range) from the reference element, 0. Distance and orientation of a vector are easily calculated in terms of the number of ranges, R, it traverses. (See Listing 1) If M is odd the vector points up and to the left of the major diagonal and R = (M+1)/2. If M is even the vector points down and to the right and R = M/2. The fractional portion, G, expresses the *vector gradient* (angle of inclination) in relation to the major diagonal at M.00. In the example vector depicted in Figure 6 the tallynumber of the vector displacement is T = 124, at range R = 6. Its SPOT value S = squareRoot(124 ) = 11.1355.... The magnitude M = 11; the gradient G = 0.1355....

## 5. LAVA

### 5.1.1 *Two Layers of the LAVA Matrix*

The method of lattice articulated vector analysis (LAVA) discussed here organizes the analytic matrix into two layers:

- *lava-lattice*, a base 9 x 9 matrix of cells, numbered from 0…80 (Figure 7a)
- *meta-lattice*, each cell of which comprises a 3 x 3 sub-region of the lava-lattice (Figure 7b)



a b

**Figure 7**

### 5.1.2 *Registering vectors in the lava-lattice*

Cartesian coordinates of markup stroke vectors are analyzed and filtered into a set of those that intersect one or more cell boundaries. The SPOT algorithm (Listing 1) converts each pair of Cartesian points (displacement) into an equivalent SPOT vector. (Note: it is important to distinguish between SPOT vector displacement notation and the underlying labeling of LAVA matrix cells along the spiral pathway of their deployment. While related, these are two separate notational considerations.)

### 5.1.3 *Registering vectors in the meta-lattice*

Eight of the lava-lattice lines also bound nine 'super', or *meta*-cells (each comprising a 3 x 3 block of lava-lattice cells) that constitute the meta-lattice. Intersections of any of these eight lattice-lines are registered in both the lava-lattice and the meta-lattice. Vector registration and assessment are handled in the 9-cell meta-lattice in the same manner as for the 81 cell lava-lattice. (Figure 7b) The latter samples finely to generates a longer 81-slot SILKey component termed *laval* (for lava-lattice) while the former samples coarsely to generate a terse 9-slot SILKey termed *metal* (for meta-lattice).

### 5.1.4 *LAVA/SILK Utility*

After all vectors have been processed LAVA examines default mappings of lava-lattice *symmetric twin* cells defined across each orthogonal and diagonal axis to estimate *axial symmetry*.

During LAVA each lattice cell registers and evaluates line vectors impinging upon it. LAVA expresses a rhythmic sampling of planar disposition and deployment of linear elements within the parent limned image. Results among LAVA cells are assessed in relation to both orthogonal and diagonal axes to derive estimates of internal symmetry. Calculated symmetry, meta-lattice, and lava-lattice values are then textually encoded in a triplet of hexadecimal SILKeys that can be used individually or together to sort and compare among constituents of image collections.

The fundamental notion of LAVA/SILK processing is quite simple: configurations with line vectors similarly deployed across the LAVA matrix generate SILKeys whose positional values are correspondingly similar. Such SILKeys become textual tokens of their parent graphic configurations for sorting, comparing, manipulating, and retrieving objects that are similar in shape or composition within a LAVA/SILK markup database.

## 6. SILK

The silhouette subframe depicted in Figure 5b produces the following SILKey components from LAVA.

## 6.1 SILKey Components

### 6.1.1 *Prefix*

A *key prefix* may optionally be used to denote SILKey version and process slot width. In XML/SVG renderings this information is given by SILK namespace attributes.

SILK2.1-1^1-2.1: denotes

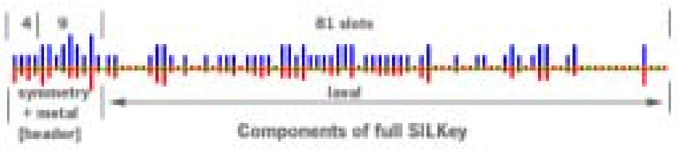
- SILK version 2.1;
- slot widths, here 1-vector + 1-gradient_integer [+ 1-(always given) gradient_fraction]; and
- LAVA version 2.1

### 6.1.2 Symmetry

As described in *5.1.4 LAVA/SILK Utility* (above): 45424442

Each symmetry axis is represented in a pair of hexadecimal digits. The first digit of each pair expresses the relative proportion of vector counts matched across an axis; the second digit expresses relative proportion of cell gradients matched across the same axis. The four axes, in left-to-right order, are vertical and horizontal orthogonal axes, then major and minor diagonal axes. The minimum symmetry value is 0; maximum symmetry value, f.

### 6.1.3 Plotted SILK Values



*Vector count = blue; clutch = red*

**Figure 8**

## 6.2 Meld SILKey

Several SILKey types may be derived from LAVA. Most useful is the *meld SILKey*, which melds, or summarizes, all of the LAVA calculations into a set of triples: an 8-digit symmetry key, 9-digit metal key, and 81-digit laval key. Metal and laval values range from 0..f, with ‘+’ denoting a null (no vectors) cell.

### 6.2.1 MeldSILKey for Figure 5b

meld-metal: c733579b7

meld-laval: bd++++abd8+9++5+67a+78db+77e9226a526+74ab b97+ec+++7++21++777+188++75+++++++++c+++

## 6.3 SILKey Compression

The same meld-laval key for the Figure 5b example, packed:

bd*14+abd8+9++5+67a+78db+77e9226a526+74abb97+ec+++7++ 21++777+188++75*19+c+++

Especially for simple image structures a SILKey can exhibit many empty cells. Long runs of slots filled with a repeating character may be replaced by **awnn…* where * signals beginning of a run, a=the character being repeated (usually ‘+’), w= hex number of following digits needed to denote the hex nn…= count of repeating characters. Since the minimum character count of the compression token is four, only runs of four or more are packed.

All variants of SILKey may use compression, as desired, so long as care is taken to ensure that matching slots are aligned during comparisons and manipulations.

## 7. CARTESIAN TO SPOT CONVERSION

```
#! /usr/bin/env python

import math

# convert Cartesian 2D vector to SPOT digital vector
def carSpotT(dx, dy):      # returns SPOT tallynumber
    x = int(dx)          # x,y displacements must be signed integers!!!
    y = int(dy)
    xneg = x<0
    yneg = y<0
    absx = abs(x)
    absy = abs(y)

    if(absx == absy):   # at corner of SPOT Range square?
        r = absx
        if(xneg and not(yneg)):
            spotT = 4*r*r + 4*r
        elif(xneg and yneg):
            spotT = 4*r*r - 2*r
        elif(not(xneg) and yneg):
            spotT = 4*r*r
        elif(not xneg and not(yneg)):
            spotT = 4*r*r + 2*r
        return spotT

    else:      # not at corner, on which Leg ("side") of Range square?
        r = max(absx, absy)
        if(r == absx):
            if(xneg):
                spotL = 0
            else:
                spotL = 2
        else:
            if(yneg):
                spotL = 1
            else:
                spotL = 3

    if(spotL == 0):      # calculate SPOT TallyNumber this Range, r
        spotT = 4*r*r - 3*r - y
    elif(spotL == 1):
        spotT = 4*r*r - r + x
    elif(spotL == 2):
        spotT = 4*r*r + r + y
    elif(spotL == 3):
        spotT = 4*r*r + 3*r – x
    return spotT


def carSpotV(dx,dy):             # returns SPOT vector
    spotV = math.sqrt(carSpotT(dx,dy))
        return spotV
```

**Listing 1**

## 8. LAVA/SILK in XML/SVG

Listing 2 is an SVG markup of the Figure 5b example.

```
    <?xml version='1.0' encoding='UTF-8'?>
    <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
    <svg xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
xmlns='http://www.w3.org/2000/svg'
xmlns:dc='http://purl.org/dc/elements/1.1/'
xmlns:lava='http://www.spotops.net/protocols/lava21'
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
height='100%' width='100%' version='1.1' y='0' x='0'
xmlns:silk='http://www.spotops.net/protocols/silk21'>
 <metadata>rdf:RDF><rdf:Description about='/data/sweetgum.jpg'
dc:creator='Wade Penny' dc:title='Sweetgum Leaf' dc:language='en'
dc:format='image/svg+xml' dc:date='2005-01-01 19:28:37'
lava:limnarity='L:3' dc:publisher='SPOTOPS'
dc:description='Sweetgum Leaf' dc:subject='sweetgum leaf'/>
 </rdf:RDF></metadata>
<g width='400' lava:id='lavaset' height='268'> <g opacity='1.0'
silk:meldmetal='9957189ae' lava:id='baseframe'
```

```
transform='translate(0,0)' stroke='#0000ff'
silk:meldlaval='be9998abacd36784+76ba9cce+++3++76c11+++6+
++a+dc++++++++++++c+++++++++++++++c+++'
lava:role='muframe' lava:kin='b2' stroke-width='3'
silk:symmetry='77786676' display='visible' fill='none'>
        <rect height='268' width='400' stroke-dasharray='40 10 5 10'
lava:role='muframe' y='0' x='0' stroke-width='6' fill='none'/>
        <g opacity='1.0' silk:meldmetal='8666289c0'
lava:id='frame_1' transform='translate(69,3)' stroke='#00cc00'
silk:meldlaval='b088599cc803668225599++c+00+355++c+++++5+
+99++c+++++33++++++c+++++++5++++++++c+++'
lava:role='muframe'  lava:kin='72' stroke-width='3'
silk:symmetry='65559754' display='visible' fill='none'>
          <rect height='246' width='255' stroke-dasharray='10 10'
lava:role='muframe' y='0' x='0' stroke-width='6' fill='none'/>
          <polyline points='122 3 123 47 120 74 116 89 113 146 107
197 103 238' lava:id='poly_1' lava:role='mulava'/>
          <polyline points='116 91 91 120 62 150 43 168 31 185'
lava:id='poly_2' lava:role='mulava'/>
          <polyline points='115 89 74 97 43 102 15 111'
lava:id='poly_3' lava:role='mulava'/>
          <polyline points='114 87 84 70 67 58 54 54'
lava:id='poly_4' lava:role='mulava'/>
          <polyline points='118 88 151 75 176 63 190 56'
lava:id='poly_5' lava:role='mulava'/>
          <polyline points='120 89 157 100 204 112 231 120'
lava:id='poly_6' lava:role='mulava'/>
          <polyline points='119 89 145 124 168 154 188 180 208
207' lava:id='poly_7' lava:role='mulava'/>
        </g>
        <g opacity='1.0' silk:meldmetal='ab44489af'
lava:id='frame_2' transform='translate(69,43)' stroke='#ffff00'
silk:meldlaval='bd++7+abd8+9+6a367aa78db+fbe99258817+75+bb
97+ec+++f++39++776+352+++6+++++++++c+++'
lava:role='muframe' lava:kin='42' stroke-width='3'
silk:symmetry='77767587' display='visible' fill='none'>
          <rect height='207' width='255' stroke-dasharray='10 10'
lava:role='muframe' y='0' x='0' stroke-width='6' fill='none'/>
          <polyline points='114 46 103 26 82 16 64 12 46 11 59 25
68 32 76 39 61 40 41 50 28 61 8 71 32 74 48 76 66 75 44 103 35
129 25 154 47 135 67 127 79 124 88 112 88 133 95 158 102 174
104 199 112 178 125 158 133 141 135 126 135 110 142 127 159
143 180 155 198 164 217 179 205 153 200 126 190 106 175 88
167 81 189 88 210 85 225 85 245 85 214 65 196 54 177 49 164 47
180 35 191 17 206 8 173 17 147 24 132 34 122 49 113 46'
lava:id='poly_8' lava:role='mulava'/></g></g></g></svg>
```

**Listing 2**

## 9. APPLICATION DOMAINS

LAVA/SILK has utility for searching large image archives of SILK meta-data (established either manually or by some batch-process vectorizer). A graphic interface can let a searcher sketch target features on a drawing pad to generate the search SILKey, which then gathers image links to candidates with similar SILKeys (governed by 'fuzzy logic' or Bayesian filter criteria set per search). Similarly, in ‘machine vision' LAVA/SILK can define template patterns of visual features to guide recognition and manipulation of objects.

LAVA/SILK is applicable wherever configurative relationships are important, either in some native subject class itself or in visualizations derived from it. For example, 3D wireframe top, front, back, side, and bottom views can each be 'flattened' , analyzed and expressed in linked SILKeys; this set of SILKeys can serve as a meta-data key for the total 3D form.

The approach may be generalized to any number of dimensions by flattening defined views into 2D projections of limned features. Even areas not ordinarily regarded as visual, such as semantic comparisons among texts, can be shown to benefit from application of LAVA/SILK processing within a context of visualized (diagrammatic) semantic structures.[1]

## 10. Sketch & Fetch Interface

Figure 9 conceptualizes an interface to a search engine that uses LAVA/SILK to extract likely candidates from vast image archives. An exploratory prototype is available for trial at www.spotops.net .

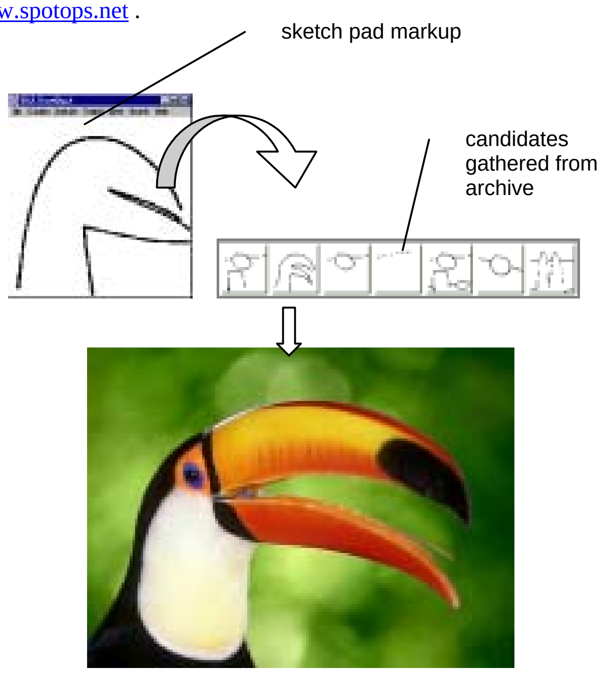


**Figure 9**

## 11. REFERENCES

[1] *WordNet*, http://www.cogsci.princeton.edu/cgi-bin/webwn

[2] Kurth, Willi, ed. *The Complete Woodcuts of Albrecht Durer*. Dover Publications, Inc., Plate 340, 1963.

[3] Jones, Howard. The SPOT System: A Method of Numerically Specifying Spatial Configuration. In *Computer Graphics*, SIGGRAPH/ACM, Vol. 15, No. 1, April 1981, pp. 124-136.