# Localization and Detection of Vector Logo Image Plagiarism

Jong P. Yoon and Zhixiong Chen

Dept of Computer Information Science
Mercy College, Dobbs Ferry, NY
{jyoon,zchen}@mercy.edu

**Abstract.** One of the main research issues in forensic computing is to protect intellectual properties. Logo images, one type of intellectual properties, are posted in the Internet and widely available. Logo image plagiarism and theft are not unusual. Detection and localization of logo image plagiarism are crucial to protect logo intellectual property. In recent years, logo images that are written in Scalable Vector Graphics format are able to be rendered efficiently in the web browser and accessed easily. In this paper, after introducing logo images edited and rendered from scalable vector graphics, we classify all possible types of logo image plagiarism, localize a possible set of logo images being infringed using distance functions, and detect and verify logo plagiarism using reversible transformation. We believe our work is valuable to businesses involving logo creation and development.

**Keywords:** Logo Image Forensics, Intellectual Property Theft, Scalable Vector Graphics.

## 1 Introduction

Logo images, one type of intellectual properties, are posted and widely available in the Internet. Since the evolution of logo as the "identity mark" over the years, plenty of iconic designs have been stolen and reused illegally [11]. To protect intellectual properties of images, a number of techniques have been successfully used. Since one of the biggest sources of image theft is search engines, websites hold a robots.txt file can prevent the images to be searchable because the file could disallow the directory of image files. A popular trick is to disable right-click with JavaScript code [20]. A lesser-known nifty trick is to cloak images behind a transparent gif. The trick is to place a transparent GIF image the same size over the top of a real original image [23]. When right-clicking the image, the transparent GIF is saved, which is an empty image. Another safe method of protecting an actual image is watermarking. The watermarking technique is to place a semi-transparent line of text (e.g., a business name or domain name) right through the middle of the image, rendering it useless for anyone else. This technique has been developed tremendously to preserve the intellectual property in a specifically interesting spot of component images [14]. Obviously these methods impact the way the image looks, and they are efficient for raster images [22] or vector quantization [13]. If images are vector images, e.g., in scalable vector

S. Goel (Ed.): ICDF2C 2009, LNICST 31, pp. 43–58, 2010.

graphics (SVG) format [21], they are not efficient due to descriptive languages. In this paper, rather than protection, we consider detection, localization and verification methods for logo image plagiarism. There are several programs and efforts developed to check the plagiarism of publications, database contents, internet documents [2,4,5,10,16,18,19]. Detecting image plagiarism is very difficult. Plagiarism checking methods have been developed and executable to the names of image files [9], and images or photos themselves. The techniques developed are for mainly colors and external annotation (like timestamps) [17]. Recently studied are the techniques to detect image copies [24]. Entireties of images are analyzed based on salient point matching (SURF), discrete cosine, or wavelet transforms. It is well known that logo images are a special type of images and they convey more meaningful content in concise format. Also, it is uneasy to claim for sure of logo plagiarism because once logo or logo components are copied, they are not trivial to identify in automation.

SVG is used to describing 2D graphics in XML, which is a text file. SVG has many advantages facing to Web application so that images can be shown dynamically and interactively. As increasing usage of SVG graphics in commercial Web applications for exchanging or publishing data, unauthorized duplication and distribution of SVG data become a mandatory concern for many Web applications.

Especially in SVG images, it is often tempting for artists to take short cuts by copying an original image logo. There are two cases of image logo plagiarism. Logos are infringed by copying the main concepts of image logos or by copying the core segments of elements. We call the former “concept copy”, and the latter “code copy”. It is said that image logos are concept-copied although the code elements are not copied but if the concept of images looks similar. For example, In Fig. 1(a) is concept-copied to (b). It is also said that image logos are code-copied if the code segments are similar. For example, Fig. 1(c) is code-copied to (d). In these two extreme cases of logo plagiarism, we consider the latter in this paper.

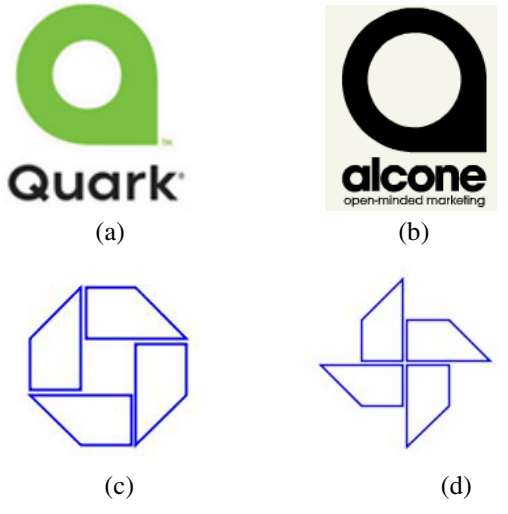



**Fig. 1.** Sample Logos

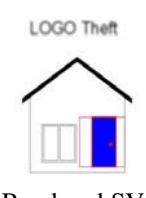
If plagiarized logos were always copied in their entirety, existing technologies would perform adequately. However, this kind of blatant plagiarism is not the most common form. Most violations, indeed, occur when a person uses a small piece or a component image from another logo. If the person borrows from familiar material such as logos within the person's prior work or his or her agency's prior work, the logo plagiarism can sometimes be caught by alert human reviewers. Often, it is modified sufficiently to be considered a unique creation. This paper describes a technique to detect the plagiarism of SVG logos. One of the risks of having the SVG logos available in the Internet is copyright violation.

The remainder of this paper includes the following: Section 2 describes SVG preliminaries. Section 3 describes the descriptors of SVG logo elements, with distance functions. Section 4 describes the taxonomy of logo plagiarism. Section 5 describes the detection method of logo plagiarism. Section 6 concludes our work.

## 2 SVG Preliminaries

SVG defines the way to represent three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composed into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances search ability and accessibility of the SVG graphics. Nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility are some of the more interesting features of SVG. A rich set of event handlers can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page.

Let us look at an example image in Fig. 2 (a), which is represented in and rendered from the SVG file in Fig. 2 (b). SVG document consists of a set of SVG document fragments and each SVG document fragment consists of any number of SVG elements. Line 1 of Figure 2 (b) shows an example of "svg" element. The "width" "height" and "viewBox" attributes specify width, height and container, respectively. The "title" and "desc" element provide a textual description of SVG document fragment.




(a) Rendered SVG

```
(1) <svg width="2.7167in" height="2.9833in" viewBox="0 0 945 999">
(2) <path style="fill:none;stroke:#000;stroke-width:2" d="M 149,556
          l275,-192 274,193 0,397 -548,0 0,-397" />
(3) <path style="fill:#000" d="M 106,588 l318,-222 318,223 0,-28
          -318,-222 -318,222 0,28z" />
(4) <path id="d" style="fill:none;stroke:#f00;stroke-width:3"
          d="M 439,661 l259,0 0,266 -259,0 0,-266" />
(5) <text x="150" y="220" style="font-size:70pt;filter:url
          (#drop-shadow);">LOGO Theft</text>
(6) <path id="d" style="fill:#00f;stroke:#f00;stroke-width:5"
          d="M648,661 l-139,0 0,266 139,0 0,-266" />
(7) <circle id="node" style="fill:#f00;stroke:#f00;stroke-width:2"
          cx="620" cy="807" r="10"/>
(8) <rect x="220" y="700" width="200" height="200" .....>
(9) <rect x="230" y="710" width="85" height="180"  .....>
(10) <rect x="325" y="710" width="85" height="180"  .....>
(11) </svg>
```

(b) SVG code

**Fig. 2.** Example of SVG Path Element

SVG contains the following set of basic shape elements: rectangle, circle, ellipses, lines, polyline, and polygons. For example in Fig. 2, the windows in (a) are described as rectangles in line (8) – (10) in (b). The text "LOGO Theft" in (a) is described as text in line (5) in (b). The doorknob in (a) is described as circle in line (7) in (b). All others are described in path in (b). Mathematically, these shape elements are equivalent to a "path" element that would construct the same shape. An SVG element "path" is defined a sequence of path data attribute, where attribute contains the *moveto*, *line*, *curve* (both cubic and quadratic Béziers), *arc* and *closepath* instructions. Line 3 of Fig. 2 (b) specifies a path element for the shape of the house roof in Fig. 2 (a). (The attribute **M** indicates a *moveto*, **l** indicates *lineto*'s, and **Z** indicates a *closepath*). The attributes are case-sensitive. **M** indicates that absolute coordinates will follow; **m** indicates that relative coordinates will follow. The effect of moveto attribute is as if the "pen" were lifted and moved to a new location. A new path data must begin with a moveto attribute. The lineto (**L** or **l**) attribute draw straight lines from the current point to a new point. **L** uses absolute coordinate system and **l** uses relative coordinate system. A number of coordinate pairs may be used to draw a polyline. The "closepath" (**Z** or **z**) ends the current subpath and causes an automatic straight line to be drawn from the current point to the initial point of the current subpath.

A SVG logo consists of one or more SVG elements. Therefore a sequence of elements can represent a logo. Most of the application programs that support SVG output functionality generate the SVG using path elements. The reason is that Path element can draw all basic shape elements such as rectangle, lines, and etc, as well as all path elements such as Polyline and Polygon. Note that the description of Polyline and Polygon is similar to the Path element. Given two logo images, although they look similar, it is less likely that the Path expressions of those two images are also similar. If they contain similar Path sequences, then it is likely that one is copied to another.

## 3 SVG Element Description

In this section, we investigate an approach to represent the SVG expression. We observe that the SVG expressions can be classified into two groups: Basic Shape Expression, and Path Expression. The basic shape expression consists of basic shape elements of SVG such as `<rect>` `<circ>` `<ellipse>`, while the path expression consists of `<path>` `<polygon>` `<polyline>`. Since SVG is an XML tag language for vector graphics, each element has angle and length components. We first describe the descriptors for the path expression in terms of angle and length. Therefore, a Path expression, which is a sequence of x- and y-coordinates, will be converted to a sequence of (either or both) angles and lengths. Then, we also describe the presentation for the basic shape expression. In the proposed descriptors, we discuss the similarity function. The similarity functions based on angle and distance is then combined to identify the similarity of logo images in Section 5.

### 3.1 Angle Descriptor for Path Expression

We compare two ways of representing angle component: interior angle sequence and directional angle sequence.

**Interior Angle Sequence**

This subsection describes the interior angle approach to handling path data sequences. The definition of interior angle is the angle formed inside a polygon by two adjacent sides. Using interior angles between vectors as transformation function, a path data sequence can be transformed to a sequence of degrees. We define the angle theta between two vectors $v$ and $w$ by the formula

$$\theta = \cos^{-1}\left(\frac{(v \circ w)}{\sqrt{\| v \| \times \| w \|}}\right)$$

Now, we can transform series of vectors to series of angles between vectors by using above formula. The generated sequences have scale, transform, and rotational invariant.

For example in Fig.3 (a), the shape O1 is represented in the interior angle sequence, {90, 270, 90, 90, 270, 90, 90, 270, 90, 90, 270, 90}. If any component logo image is a closed shape, the interior angle sequence is circular, meaning that the sequence can be repeated and there is no starting or ending point. This descriptor is scale, transform, and rotational invariant. In this description, in Fig.3 the shape O1 in (a) and O2 in (b) are exactly the same. Although rotated, this interior angle sequence makes it easier to identify its similarity.

There are some similarity measures in computational geography literatures such as discrete metric[15], Minkowski distance, bottleneck distance [6], hausdroff distance [1], frechet distance [8], template metric [3], transport distance [7], and etc.

In this paper, we give our definition of distance between two components. Given two interior angle sequences, $o_{i1}$,$o_{i2}$, …, and $o_{j1}$, $o_{j2}$, … for shapes $P_i$ and $P_j$, the distance between $P_i$ and $P_j$ is defined as follows.

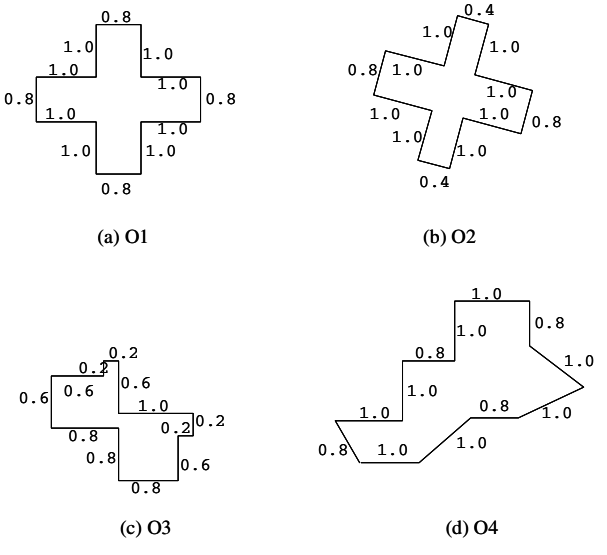



**Fig. 3.** Angle and Length of Description

**Definition 3.1** (*Interior Angle Distance of Two Components*). Suppose $P_i$ and $Pj$ are two path elements from Logo $L_i$ and $L_i$. $O_i$ and $O_i$ are the interior angle sequence corresponding to $P_i$ and $Pj$, respectively. The interior angle distance between the two elements is defined as

$$\left\|P_i - P_j\right\|_{ia} \equiv Dist_{ia}(P_i, P_j) = \frac{\min\left\{\sum_{o_{ik} \in O_i, o_{jk} \in Oj} \left|o_{ik} - o_{jk}\right|\right\}}{180 \bullet d} \tag{1}$$

where the minimum is taken over all possible starting angle from either $P_i$ or $Pj$. For example, take $Pj$ and its angle sequence fixed, let the starting angle of sequence from $P_i$ shift one by one, calculate the sum of their differences and take minimum of all these values. d is the smaller cardinal number of $\left\|O_i\right\|$ and $\left\|O_j\right\|$.

For example, the distance between P1 in (a) and P2 in (b), $\left\|P_1 - P_2\right\|_{ia}$ = 0/(180*12), where O1 = O2={90, 270, 90, 90, 270, 90, 90, 270, 90, 90, 270, 90}.

However, if any logo image is open-ended shape, it may be represented in a directional angle descriptor, which is described below.

**Directional Angle Descriptor**

This subsection describes the approach to handling the path data sequence for open-ended shapes. We can get a desired transform function by substitute one of two vectors from above formula with unit x-axis vector. We define the angle theta between two vector *x* and *w* by the formula

$$\theta = \cos^{-1}\left(\frac{(x \circ w)}{\sqrt{\| x \| \times \| w \|}}\right).$$

Now, we can transform the series of vectors to the series of angles between two vectors by using above formula. The generated sequences have scale and transform invariant properties, but it does not have rotational invariant properties.

We define the directional angle distance similar to formula 1 except that the angle sequences are replace by directional angle sequence.

**Definition 3.2** (Directional *Angle Distance of Two Components*). Suppose $P_i$ and $Pj$ are two path elements from Logo $L_i$ and $L_i$. $D_i$ and $D_i$ are the directional angle sequence corresponding to $P_i$ and $Pj$, respectively. The directional angle distance between the two elements are defined as

$$\left\|P_i - P_j\right\|_{da} \equiv Dist_{da}(P_i, P_j) = \frac{\min\left\{\sum_{D_{ik} \in D_i, D_{jk} \in Dj} \left|D_{ik} - D_{jk}\right|\right\}}{180 \bullet d} \tag{2}$$

where the minimum is taken over all possible starting angle from either $P_i$ or $Pj$.

For example in Fig.3 (a) and (b), they are not similar because their directional angle sequences for O1 and O2 respectively are {0, -90, 0, 90, 0, 90, 180, 90, -180, -90,

-180, -90} and {30, 60, 30, 120, 30, 120, 210, 120, 210, 290, 210, 290}. $\|P_1 - P_2\|_{da}$ = 30*12/180*12 ≈ 0.18.. However, as we can observe that the figure (a) and (c) have the same directional angle sequence descriptor and interior angle sequence descriptor. So angle distance described in Equation (1) and (2) are insufficient. To overcome the shortfall, in the next subsection, we introduce another distance function.

### 3.2 Length Descriptor for Path Expression

The angle description of spatial data has the representative power because the rotation and scales in the description are tolerant. The disadvantages of using the angle sequences are following: 1) it is very sensitive to the noise; 2) shapes which are different may generate same angle sequences (See Fig. 3). First shortcoming can be taken care by a polygon approximation. Second problem can be overcome by considering the length of line segment of path data. Polygon approximation will be discussed in the next section. Instead, the rest of this subsection describes the length of line segments. Let's assume there are vertices $V_1 = (x_1, y_1)$, $V_2 = (x_2, y_2)$, …, $V_n = (x_n, y_n)$. We define the length of the line segment $\overline{V_k V_{k+1}}$ as Euclidean distancetwo points $V_k$ and $V_{k+1}$.. Therefore we can get a length sequence. Clearly, this length of line segment L is dependent on size of shape, so we have to normalize it. Normalized length of line segment $\overline{V_k V_{k+1}}$ can be calculated by following formula:

$$\frac{L_k}{\max\limits_{1 \le k \le n-1} (\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2})}$$

where $L_k$ is length of line segment $\overline{V_k V_{k+1}}$. Normalized length sequence is independent of the size of the shape. The numbers on edges in Fig. 3 indicate the normalized length sequence.

We also define another similar distance function as follows.

**Definition 3.3** (Length *Distance of Two Components*) Suppose $P_i$ and $Pj$ are two path elements from Logo $L_i$ and $L_i$. $L_i$ and $L_i$ are the normalized length sequence corresponding to $P_i$ and $Pj$, respectively. The length distance between the two elements are defined as

$$\|P_i - P_j\|_l \equiv Dist_l(P_i, P_j) = \frac{\min\left\{\sum_{L_{ik} \in L_i, L_{jk} \in Lj} |L_{ik} - L_{jk}|\right\}}{l} \tag{3}$$

where the minimum is taken over all possible starting length from either $P_i$ or $Pj$. L is the minimum of the two maximum distance, that is $\min \max\limits_{1 \le k \le n-1} (\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2})$.

For example, the distance between O1 in (a) and O2 in (b) is {(0.8-0.4)+(0.8-0.4)}/12 = 0.8/12 = 0.067, where O1 = {1.0, 1.0, 0.8, 1.0, 1.0, 0.8, 1.0, 1.0, 0.8, 1.0,

1.0, 0.8} and O2={1.0, 1.0, 0.4, 1.0, 1.0, 0.8, 1.0, 1.0, 0.4, 1.0, 1.0, 0.8}. However, $\left\| P_i - P_j \right\|_l = 0$. It means that O1 is more similar to O4 than O2, in this length sequence descriptor.

Moreover, in the Fig.3, (a) and (c) have the same interior angle sequence descriptor, but they look different. Although (a) and (b) have the same interior angle sequence, they have different normalized length. However, the length sequence is not sufficient to describe such spatial data. Using the distance functions in (1) and (2), together with the distance measure when it comes to the basic logo images, Section 5 describes a detection method of logo plagiarism.

### 3.3 Distance Function for Basic Shapes

The Edit distance functions in Equation (1) and (2) discussed in the previous subjections are measured for the similarity of Path expressions in SVG logo images. Although a Path expression represents more subjective concepts of image design, because a logo image may consist of more than the Path expressions, in this section we discuss how basic shapes are compared. Recall those basic shapes of SVG elements: Rectangle, Circle, and Ellipse. These basic elements specify a couple of pivot points based on which the desired basic shapes are automatically plotted. For example, the Rect element specifies the top left point with width and height. The attributes specified in those basic shape elements are directly compared by the following distance function.

$$\left\| S_i - S_j \right\|_s = \psi_1 MIN\left( \sqrt{\frac{(x_i - x_j)^2}{\sigma_i^2} + \frac{(y_i - y_j)^2}{\sigma_j^2}} \right) + \psi_2 \alpha \tag{4}$$

where the weights $\psi_1$ and $\psi_2$ are in this relation, $\psi_1$=1- $\psi_2$, $\alpha = MIN\left( \sqrt{\frac{(width_i - width_j)^2}{\sigma_i^2} + \frac{(height_i - height_j)^2}{\sigma_j^2}} \right)$ for Rect element, $\alpha = MIN\left( \sqrt{\frac{(radius_i - radius_j)^2}{\sigma_i^2}} \right)$ for circle, $\alpha = MIN\left( \sqrt{\frac{(x2_i - x2_j)^2}{\sigma_i^2} + \frac{(y2_i - y2_j)^2}{\sigma_j^2}} \right)$ for ellipse, and σ is the standard deviation of the values such that width, height, radius, and *x*-, *y*- coordinates.

## 4 Logo Plagiarism

In this section, we analyze the types of logo thefts and describe the taxonomy of logo plagiarism. Logo thefts are classified in terms of size and quality. Logos are copied entirely or partially, in size. Logos are either simply reused or reused by transformation. By transformation, the quality of original logos is changed.

Images are transformed in three operations: translation, scaling, and rotation. In linear algebra, linear transformations can be represented by matrices. If T is a linear transformation mapping $\mathbf{R}^n$ to $\mathbf{R}^m$ and *x* is a column vector with n entries, then

$$T\left(\vec{x}\right) = A\vec{x} \tag{5}$$

for some $m \times n$ matrix **A**, called the transformation matrix of **T**. If one has a linear transformation $T(x)$ in functional form, it is easy to determine the transformation matrix **A** by simply transforming each of the vectors of the standard basis by $T$ and then inserting the results into the columns of a matrix. That is,

$$A = \left[T\left(\vec{e_1}\right) \quad T\left(\vec{e_2}\right) \cdots T\left(\vec{e_n}\right)\right] \tag{6}$$

Given SVG logo $L_i$, a transformation operation $T$, where $T$ can be translate, scale, rotate, and reflection, a component image $e_i$ in an image log $L_i$ can be simply transformed and copied to $L_i$'. Hence, $T(e_i) \in L_i$', where $e_i \in L_i$. We observe that there are two characteristics of element $e_i$: 1) the element $e_i$ in the path expression, and 2) $e_i$ in the basic shape expression. We also observe that there are two characteristics of transformation ***T***: 1) built-in transformation $T_b$, and 2) user-defined transformation $T$. $T_b$ contains the build-in operations such as translate, rotate, and scale [21]. The user-defined transformation is a transformation described by theft, but not using built-in SVG operations, as exemplified below. Having observed these characteristics, logo plagiarism can be classified to the following four cases:

- Built-transformation of basic shape expression
- Built-transformation of path expression
- User-defined transformation of basic shape expression
- User-defined transformation of path expression

Simply, one naïve copy of logos is to apply a SVG built-in transformation operation to logo expressions, either to basic shape or path expression. A logo can be copied quite easily by embedding a built-in transformation operator or a combination of such operators. Of course, logo plagiarism in this case is also easy to detect. For example, in line (8) in Fig. 2 (b),

```
<rect x="220" y="700" width="200" height="200" trans-
form="translate(30) scale(1.5 1.5)">
```

moves the rectangle 30 pixels in x-direction and scale up 50%. However, even in this case, if an original logo consists of basic shape expressions, logo plagiarism is hard to be claimed. For example, in the above SVG statement stating a rectangle, which is a basic shape expression, it is uneasy to claim its plagiarism although a built-in transformation operator is straight forwardly stated.

User-defined transformation is to rewrite a SVG statement manually if not automatically. So, the transformed shape is similar, if not the same as, to the original logo. For example, consider line (6) in Fig. 2(b). If we want to rotate the path expression 45 degree around the point (509, 661), the SVG statement will be

```
<path d="M509,661 l 98.2878,98.2878 -188,188 -98.2878,
-98.2878 188,-188" />.
```

The above statement looks very different from the original statement in line (6), however, it renders the very same shape but only being rotated. The rotated shape is therefore equivalent to the following if you like:

```
<path d="M648,661 l-139,0 0,266 139,0 0,-266" trans-
form="rotate(45 509 661)" />
```

For the similarly shaped logos, if we look into SVG expressions, we may be able to see a clearer clue. This paper and also this section focus on the plagiarism by user-defined transformation of path expression.

What it follows in this section describes the types of logo plagiarism. We investigate that logo plagiarism is performed by translating, scaling, rotating and reflecting part of a Path element.

### 4.1 Logo Plagiarism with Translation

The translate operator performs a geometric transformation which maps the position of each picture element in an input image into a new position in an output image, where the dimensionality of the two images often is, but need not necessarily be, the same. Under translation, an image element, say SVG element, located at $(x_1, y_1)$ in the original is shifted to a new position $(x_2, y_2)$ in the corresponding output image by displacing it through a user-specified translation $(\beta_x,\beta_y)$. Therefore, the following holds: $x_2 = x_1 + \beta_x$, and $y_2 = y_1 + \beta_y$.

For example, four parallelograms in Fig.1 (c) are translated to (d). Consider the path elements of (c):

```
(c1)  <path fill="none" stroke="blue" stroke-width="5" d="m150,50 l100,0
l100,100 l-200,0 z" />
(c2)  <path fill="none" stroke="blue" stroke-width="5" d="m350,160
l0,100 l-100,100 l0,-200 z" />
(c3)  <path fill="none" stroke="blue" stroke-width="5" d="m240,360 l-
100,0 l-100,-100 l200,0 z" />
(c4)  <path fill="none" stroke="blue" stroke-width="5" d="m40,250 l0,-
100 l100,-100 l0,200 z" />
```

and (d):

```
(d1)  <path fill="none" stroke="blue" stroke-width="5" d="m90,250 l0,-
100 l100,-100 l0,200 z" />

(d2)  <path fill="none" stroke="blue" stroke-width="5" d="m200,150
l100,0 l100,100 l-200,0 z" />
(d3)  <path fill="none" stroke="blue" stroke-width="5" d="m300,260
l0,100 l-100,100 l0,-200 z" />
(d4)  <path fill="none" stroke="blue" stroke-width="5" d="m190,360 l-
100,0 l-100,-100 l200,0 z" />
```

The drawing attributes in line (c1) and (d2) above are exactly the same except the initial moveto (m in these cases) expression. (c1) started to draw a parallelogram from the point (150, 50), while (d2) from (200, 150). Similarly, by translating the initial moveto, the entirety of both logo images does not look similar but the code of both logo images is extremely similar if no copied

### 4.2 Logo Plagiarism with Scaling

Image scaling is the process of resizing digital images by interpolation such as jagginess, bilinear interpolation, bicubic and spline interpolation, etc [12]. By scaling, an

image can be enlarged or shrink. Given $x$ and $y$, $x' = s_x x$ and $y'=s_y y$. The matrix form is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

As a simple example, an image is convolved with the expressions x1*width, y1*length, and z1*depth, where width, length and depth are float point.

For example, in Fig. 4(a), the original logo is designed by the following three elements:

```
(1) <path fill="none" stroke="red" stroke-width="5" d="M100,200 C100,100
250,100 250,200 S400,300 400,200" />

(2) <path fill="#0088FF" stroke="green" stroke-width="5" d="M100,200
C100,300 250,300 250,200 S400,100 400,200" />

(3) <path fill="none" stroke="blue" stroke-width="5" d="M100,200
C250,200 250,100 100,100 S400,300 400,200" />
```

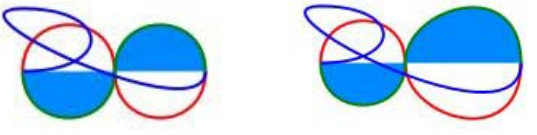
Fig. 3(a) is copied to Fig. 3(b) as follows:

```
(4) <path fill="none" stroke="red" stroke-width="5" d="M100,200 C100,100
250,100 250,200 S460,360 460,200" />

(5) <path fill="#0088FF" stroke="green" stroke-width="5" d="M100,200
C100,300 250,300 250,200 S460,40 460,200" />

(6) <path fill="none" stroke="blue" stroke-width="5" d="M100,200
C250,200 250,100 100,100 S460,360 460,200" />
```

As you can see above, line (1)-(3) is the same as line (4)-(6) except the last four values. Note that one of the path option, S, smoothes the curves. Line (1) S400,300 400,200 is scaled to line (4) S460,360 460,200.



(a) Original (b) (Right Circle and Thread) Scaled

**Fig. 4.** Logo Plagiarism with Scaling

### 4.3 Logo Plagiarism with Rotation

In linear algebra, linear transformations can be represented by matrices. For rotation by an angle θ counterclockwise about the origin, the functional form

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta.$$

Can be written in matrix form,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{8}$$

Similarly, for a rotation clockwise about the origin, the matrix becomes:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

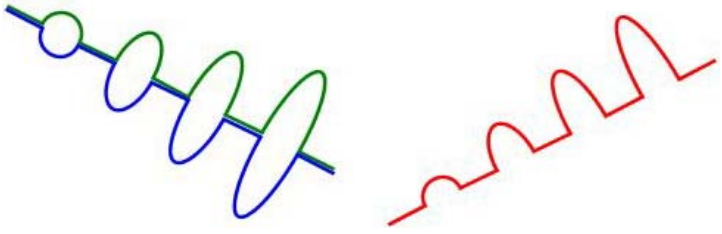
For example, the upper ellipses in Fig. 5(a) are copied to Fig.5(b). The original ellipses are written in SVG such that

```
(7) <path d="M52,45 l 50,25   a25,25 30 0,1 50,25 l 50,25      a25,50 30
0,1 50,25 l 50,25     a25,75 30 0,1 50,25 l 50,25         a25,100 30 0,1
50,25 l 50,25"    fill="none" stroke="green" stroke-width="5"  />
```

The copied ellipses are:

```
(8) <path d="M50,350 l 50,-25 a25,25 -30 0,1 50,-25 l 50,-25  a25,50 -30
0,1 50,-25 l 50,-25   a25,75 -30 0,1 50,-25 l 50,-25    a25,100 -30 0,1
50,-25 l 50,-25"  fill="none" stroke="red" stroke-width="5"  />
```

As compared between line (7) and (8), some of the values are apposite in sign, which is shown in Equation (4) and (5).



(a) Original (b) Top only rotated

**Fig. 5.** Logo Plagiarism by Rotation

### 4.4 Logo Plagiarism with Reflection

To reflect a vector about a line that goes through the origin, let $(l_x,l_y)$ be a vector in the direction of the line:

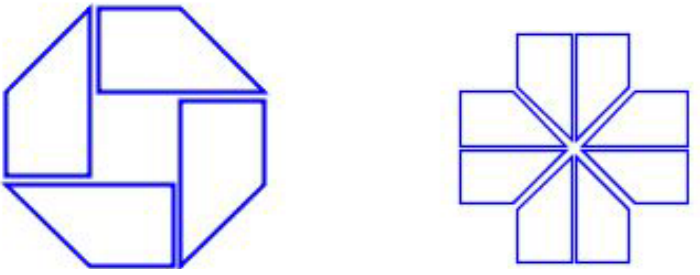
$$A = \frac{1}{l_x^2 + l_y^2} \begin{bmatrix} l_x^2 - l_y^2 & 2l_x l_x \\ 2l_x l_x & l_y^2 - l_x^2 \end{bmatrix} \tag{10}$$

Consider Fig.6, which contains a multiple times of the same shape, a so called parallelogram. In Fig. 6(a), the lower right parallelogram is reflected to the top left parallelogram in Fig. 6(b). The original parallelogram is

```
(9) <path fill="none" stroke="blue" stroke-width="5" d="m350,160 l0,100
l-100,100 l0,-200 z" />
```

The reflected parallelogram is then

```
(10) <path fill="none" stroke="blue" stroke-width="5" d="m200,100 l0,100
l100,100 l0,-200 z" />
```



(a) Original (b) Reflected components

**Fig. 6.** Logo Plagiarism with Reflection

As compared in line (9) and (10) above, the diagonal line has different values: the original has –100,100, while the reflected one has 100,100. In other words, the former tilt angle is –135°, and the later is –45°.

## 5 Detecting Logo Plagiarism

We have analyzed the mechanisms of logo creation. In this section, we derive a framework that could localize logo similarity and identify and detect logo plagiarism. To make a claim of logo plagiarism is not an easy task since logo plagiarism can be represented in various forms like conceptual plagiarism and code mapping plagiarism, or a whole logo plagiarism and only some logo components plagiarism. Therefore our logo plagiarism is in the form of possibility.

In general, our proposed approach is composed of two steps: discovery and verification, as shown in Fig. 7. To render an SVG logo, we need first to build an XML tree from SVG elements. We utilize the parsed XML tree to build logo database in which logo features such as interior and directional angle descriptors, length descriptors, and basic shape descriptors are generated offline.

Given a suspected vector logo, we take the same steps to build an XML parse tree and generate angle descriptors, length descriptors, and basic shape descriptors.

To define a measure of similarity of two logos, we describe a distance function that can indicate the difference between any two SVG image logos by combining Equation (1), (2), and (3) in Section 3.

**Definition 5.1** (*Distance Function between two SVG Logo components*). Consider two SVG logo components $C_i$ and $C_j$, The distance is defined as

$$\begin{aligned}\left\|C_i - C_j\right\| = w_{ia}\left\|P_i - P_j\right\|_{da} + w_{da}\left\|P_i - P_j\right\|_{da} + w_l\left\|P_i - P_j\right\|_l \\ + w_s\left\|S_i - S_j\right\|_s\end{aligned} \tag{11}$$

where $w_{ia}$, $w_{da}$, $w_l$ and $w_s$ are non-negative weight such that $w_{ia} + w_{da} + w_l + w_s = 1$.

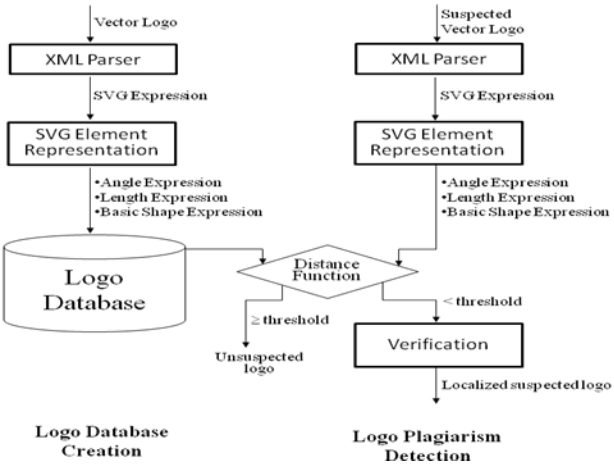



**Fig. 7.** Localization and Detection Architecture

**Definition 5.2** (*Distance Function between two SVG Logos*). Consider two SVG logo $L_i$ and $L_j$, such that $L_i$ and $L_j$ respectively consist $n$ and $m$ SVG elements, where each logo has $n_p$ path expressions and $n_s$ basic shape expressions ($n = n_p + n_s$, and likewise $m = m_p + m_s$ ). Assume that there are $n_c$ conjunctions between path expressions and basic shape expressions such that $n_c << n_p n_s$. Then, the distance between $L_i$ and $L_j$ is defined as

$$\left\| L_i - L_j \right\| = \frac{\min_{C_{ik} \in L_i} \left\{ \sum \left\| C_{ik} - C_{jk} \right\| \right\}}{\min\{n, m\}} \tag{12}$$

where the summation is over all components when a starting component is selected for either one. The minimum is taken over all starting components.

Through the distance functions, we are able to calculate similarity measures of a suspected logo against a logo database. Those logos similar to the suspected logos under pre-defined threshold are localized. They are the candidates for further verification. Let us denote it as $E$.

The verification process is more or less a manual process. For any candidate $e$ from $E$, we try to find a set of maps $\{\boldsymbol{T}\}$: $s \rightarrow e$ where $s$ is the suspected logo. Each mapping $t$ of $\boldsymbol{T}$ is corresponding to a specific component of $s$ to a specific component of $e$. It must be a composite mapping from the basic mappings that include shape-wise mapping like translation, scaling, rotation and reflection; filling and color-wise mapping like filling pattern and filling color; simple path-wise mapping and the identity map. It is easy to show that such mapping is reversible and the revisable mapping is also a composite mapping from those basic mappings. If we are able to identify such mappings, we can claim that there is high possibility of logo component similarity.

We should also mention that this algorithm is not an exhausted search. Some logo component plagiarism cases, especially conceptual plagiarism are not easily identifiable. Table 1 shows that the proposed localization and detection technique can use

**Table 1.** Usage of Distance Functions

| Plagiarism with | Detection by using | | | |
|---|---|---|---|---|
| | Interior angle descriptor | Directional angle descriptor | Length descriptor | Basic shape descriptor |
| Translation | yes | no | no | maybe |
| Scaling | yes | maybe | no | maybe |
| Rotation | yes | no | yes | maybe |
| Reflection | maybe | no | maybe | maybe |

selectively the descriptors depending on the types of plagiarism. For example, Fig. 4 may be localized and detected by the basic shape descriptor, while Fig. 5 and Fig. 6 respectively use interior angle, basic shape descriptors and interior angle, length descriptors.

## 6 Conclusion

This paper described a technique to localize suspected logo images in the presence of a database that contains original logo images. Logo images in SVG format are converted to angle and length expressions. The very conversion technique is applied to both original and suspected logo images. The logo images, which are now in angle and length expressions, are compared by the distance functions according to various types of plagiarism.

The techniques proposed in this paper can be extended and applied not only to vector graphics but also raster, JPEG, GIF images by integrating with contour identification and pattern recognition techniques.

## References

1. Alt, H., Behrends, B., Blomer, J.: Approximate matching of polygonal shapes. Annals of Mathematics and Artificial Intelligence, 251–265 (1995)
2. http://www.articlechecker.com/ (accessed on August 10, 2009)
3. Alt, H., Fuchs, U., Rote, G., Weber, G.: Matching convex shapes with respect to the symmetric difference. In: Díaz, J. (ed.) ESA 1996. LNCS, vol. 1136, pp. 320–333. Springer, Heidelberg (1996)
4. Chang, H., Mockus, A.: Evaluation of Source Code Copy Detection Methods on FreeBSD. In: Proc. of the 2008 Int'l Working Conf. on Mining Software Repositories, pp. 61–65 (2008)
5. http://www.checkforplagiarism.net/ (accessed on August 10, 2009)
6. Efrat, A., Itai, A.: Improvements on bottleneck matching and related problems using geometry. In: Proceedings of the 12th Symposium on Computational Geometry, pp. 301–310 (1996)
7. Fry, D.: Shape Recognition using metrics on the space of shapes. PhD thesis, Harvard University, Department of Mathematics (1993)

8. Godau, M.: A natural metric for curves – computing the distance for polygonal chains and approximation algorithms. In: Jantzen, M., Choffrut, C. (eds.) STACS 1991. LNCS, vol. 480, pp. 127–136. Springer, Heidelberg (1991)
9. http://images.google.com/ (accessed on August 10, 2009)
10. Liu, C., Chen, C., Han, J., Yu, P.: GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis. In: Proc. of ACM Conf. on Knowledge Discovery and Data Mining, pp. 872–881 (2006)
11. http://www.logoblog.org (accessed on August 10, 2009)
12. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999), doi:10.1109/ICCV.1999.790410
13. Lu, Z.-M., Zheng, W.-M., Pan, J.-S., Sun, Z.: Multipurpose Image Watermarking Method Based on Mean-removed Vector Quantization. Journal of Information Assurance and Security, 33–42 (2006)
14. Mohanty, S.P., Bhargava, B.K.: Invisible Wateramrking based on Creation and Robust Insertion-Extraction of Image Adaptive Watermarks. ACM Transactions on Multimedia Computing, Communications, and Applications 5, 12:1–12:22 (2008)
15. Navarro, G.: A Guided Tour to Approximate String Matching. ACM Computing Surveys 33, 31–88 (2001)
16. http://www.plagiarismchecker.net/ (accessed on August 10, 2009)
17. http://www.plagiarismtoday.com/2009/04/24/google-similar-images-poor-copy-detection/
18. Petrakis, E., Voutsakis, E., Milios, E.: Searching for Logo and Trademark Images on the Web. In: Proc. of the 6th ACM Int'l Conf. Image and Video Retrieval (2007)
19. Roy, S., Chang, E.: A Unified Framework for Resolving Ambiguity in Copy Detection. In: Proc. of ACM Conf. on Multimedia, pp. 648–655 (2005)
20. http://rainbow.arch.scriptmania.com/scripts/no_right_click.html (accessed on August 10, 2009)
21. http://www.w3.org/Graphics/SVG/ (accessed on August 10, 2009)
22. Sulehria, H.K., Zhang, Y.: Vehicle Logo Recognition Using Mathematical Morphology. In: Proc. of the 6th WSEAS Int'l Conf. on Telecommunications and Informatics (2007)
23. http://stuff.mit.edu/tweb/map.html (accessed on August 10, 2009)
24. Thomee, B., Huiskes, M., Bakker, E., Lew, M.: Large scale image copy detection evaluation. In: Proceedings of ACM International Conference on Multimedia Information Retrieval, pp. 59–66 (2008)