COMPUTER-AIDED DESIGN & APPLICATIONS, 2016
VOL. 13, NO. 5, 571–586
http://dx.doi.org/10.1080/16864360.2016.1150702

Semi-fragile watermarking for automatic detection of engineering drawing modifications in collaborative design

Yong-Fu Chen, Cheng Sun, Jia-Qi Shen, Zheng-Dong Huang and Li-Ping Chen

Huazhong University of Science and Technology, PR China

ABSTRACT
The automatic detection of engineering drawing modifications made by other designers is central to improving CAD/CAPP working efficiency levels. Engineering drawings include numerous entities, but only a few entities are modified. It is difficult for designers to identify modified entities in a complex drawing. This paper presents a novel means of detecting entities in engineering drawings created by other designers that employs semi-fragile watermark technologies. Modifications will change embedded watermarks. Consequently, detection is realized by analyzing changes. The approach was implemented, and the experimental results demonstrate that the method is reliable and can accurately detect entities that have been modified, deleted and added.

KEYWORDS
Semi-fragile watermarking; modification detection; collaborative CAD

1. Introduction

With the rapid development of manufacturing and Internet technologies, the fields of globally distributed design, analysis, natural resource-based manufacturing, and human resources have become increasingly competitive around the world [26]. Meanwhile, roughly 3 million people in 90 countries worldwide use AutoCAD software to create various types of drawings and a large number of computer drawings [25]. Studies show that roughly 70–80% of the total cost of a product is determined at the design stage, and product design has increasingly emerged as the soul of the manufacturing industry [8, 10, 21]. Apart from such costs, a foreign-based survey showed that engineering designers spend roughly 30% of their working time checking drawings [4]. All of the above findings show that several problems impede the improvement of CAD/CAPP design efficiency levels. A first problem relates to the fact that product design constitutes a long and complicated process that involves revisions and checks, and in turn designs must typically be revised and improved [5, 14]. Second, engineering design methods employed in mechanical industries are inherently multi-step, complex processes, whereby designers from various fields always cooperate based on integrated system planning and management tools [3, 7]. Thus, differences in terms of professional background and engineering knowledge and a lack of software tools that facilitate communication constitute another issue. Third, differences between two drawings are too minor to identify. All of these issues render modification detection and quick version checks a key issue during the drawing communication and design approval stages of integrated design process management [2].

A number of related studies have already been conducted. Non-blind engineering drawing comparisons based on original drawings have been a widely employed to examine this issue. Tan and Chen [20] presented a means of identifying drawing entities using an ObjectARX platform and compared original drawings with modified ones based on single entity comparisons and two-way search technologies. Furthermore, Liu et al. [9] improved on this method by pre-processing complex entities and by introducing a divide-and-conquer comparison strategy that helps achieve higher levels of comparison accuracy. However, both of these methods are inefficient. In addition, original drawings are needed for comparisons. Using a large number of drawings makes it difficult to manage drawings for comparison purposes. In turn, the non-blind method is not widely used. Therefore, it is our intention in this work to propose a semi-fragile watermarking scheme for quickly detecting and locating modifications in revised drawings without the use of original drawings.

CONTACT Yong-Fu Chen chenyf@hust.edu.cn

© 2016 CAD Solutions, LLC, http://www.cadanda.com

The reminder of this paper is organized as follows. Related work is presented in Section 2. In Section 3 explains the principle of tampering detection and localization. In Section 4, the detection process are described. The analyses of experimental results and performance are conducted in Section 5. Finally, Section 6 concludes the work.

2. Related work

Watermark technologies [18] have been applied to image, video, audio and mesh tools since they were first developed [11, 17, 23, 24]. It is widely used for data hiding, copyright protection and integrity verification. Apart from that, watermark has also been applied into tampering detection and localization. Several studies have been conducted on watermarking tampering detection for image [13, 15, 16]. Meanwhile many watermarking schemes are also proposed for watermark tampering detection on 2D vector graphics.

Wang and Men proposed reversible watermark scheme for 2D vector map [22]. The scheme involves calculating watermarks for each spatial feature group and embedding watermarks in a reversible manner to mark the original location of each feature using interpolated vertices. While the mark of each feature ensures more accurate tamper localization results, the reversible data-hiding method can be used to accurately recover original content. However, the method can only be used to detect groups that have changed. Zheng and You [29] have also studied watermark for vector map. They proposed a fragile watermark algorithm for vector map based on that the vector map was divided into a series of blocks, and watermark was embedded into these blocks. Although it can verify the integrity of the watermarked vector map and detect modification and label them with rectangles, the detection precision still needs to be improved. After that, they proposed another scheme [28]. The linear entities of vector graphics were divided into different groups and watermark was embedded into these groups. The linear entities groups which are not modified and the original data are accurately recovered. As for the linear entities groups where modification happened, the algorithm can accurately detect them and label them with the dashed line. However, the scheme can still locate the modification to certain group. Zhang and Gao [27] presented a different approach whereby drawing vertices are grouped, an 8×8 transformation matrix is constructed via DCT (Discrete Cosine Transform), and then watermark information is embedded within the middle frequency region of the DCT coefficients. This approach generates an authentication watermark based on DCT relationships. However, this method, which is not especially accurate, can only be used to detect regions. Also, some researches on watermarking are focused on 2D CAD drawings. Peng et al. [12] proposed a different approach based on logarithmic coordinate transformation whereby vertices that carry a watermark are mapped to a log-polar coordinate system. The watermark is then embedded into the mantissa of the real-valued log-polar coordinates via bit substitution. However, this method is less accurate, as it can only detect changed groups and not single entities. Guo and Peng [6] proposed an embedding watermark approach based on the improved odd-even quantization method. The entity set is first acquired from 2D engineering drawings that are arranged in groups, and then the watermark is embedded into each group via odd even quantization. This method is robust against some incidental operations (e.g., rotation, translation, and scaling), and it is sensitive to malicious attacks. However, entity grouping is required, and the method is not particularly accurate.

3. Drawing modification detection principles

Preliminary knowledge

Watermark unit *The watermark unit* is the smallest unit of *watermark information* and is denoted by W_i. In this paper, a *watermark unit* is a character embedded into an engineering drawing.

Watermark information *Watermark information* is composed of several *watermark units* and is denoted by $w = \{w_1, \ldots, w_i, \ldots, w_m\}$. It has several real meanings. It may denote the name of a designer or his personal information (e.g., the name 'Jack') and is denoted by $w = \{j, a, c, k\}$.

Watermark sequence The *watermark sequence,* denoted by W, is created by extending *watermark information* w, separated by '*' and ended by "#". This denotes that $W = \{w, ^*, w, ^*, \ldots, w, \#\}$. Therefore, W can be expressed as $W = \{w_1, \ldots, w_i, \ldots, w_m, ^*, w_1, \ldots, w_i, \ldots, w_m, ^*, \ldots, w_1, \ldots, w_i, \ldots, w_m, \#\}$. The size of W is determined by the total number of entities in an engineering drawing. Due to size limitations, the last w may be incomplete. However, the last # is required.

New watermark sequence The *new watermark sequence*, denoted by W', is the watermark sequence extracted from a revised engineering drawing.

Virtual point The *virtual point* is introduced to illustrate the algorithm used in this paper. The algorithm declares that every entity has a *virtual point* that calculated based on certain rules. This point separates entities into two parts and is called the *virtual point,* as it does not exist.

Virtual line The *virtual line* is created by joining *virtual points* of two adjacent entities. However, as the *virtual line* does not exist in reality, it is merely introduced for algorithm realization purposes.

Virtual length This is the length of a *virtual line*.

Splitting ratio k The *virtual point* divides an entity into two parts, and the length ratio of the two parts is k, denoting that the *virtual point* is calculated using the splitting ratio k. Fig. 3 shows how it is used to obtain a *virtual point* from a line, arc and circle.

3.1. Scheme overview

The semi-fragile watermark scheme proposed in this paper is based on ratio information between geometric entities in engineering drawings. The overall procedure of the proposed approach involves the following steps (see Fig. 1):

(1) Watermarks are generated and embedded by designer 1. The *watermark sequence* W for the drawing G is generated. Traverse drawing entities and embed the watermarks according to entity types and relationships. The drawing with watermarks is denoted by G_w.
(2) Revise the drawing. Designer 2 revises the drawing G_w and produces a revised drawing G'_w, which is sent to the previous designer for further approval.
(3) Extract the *new watermark sequence* W' and restore *watermark information* w. The *new watermark sequence* W', which may be changed through drawing revisions such as entity modifications, additions and deletions is extracted from G'_w using the private *splitting ratio* k, and the original *watermark information* w is restored using statistical methods.
(4) Modified entities are detected and marked with colors by analyzing the *new watermark sequence* based on the restored *watermark information*.

3.2. Embedding algorithm based on virtual points

In engineering drawings, the coordinates of vertices are presented using the IEEE-754 double-precision format [1], which is presented in Fig. 2. An IEEE-754 floating point number has three parts: a *sign* (Bit 63), *exponent* (Bits 52 to 62) and *mantissa* (Bits 0 to 51). Maxenchuk et al. [19] noted that, based on human visual capabilities, individuals cannot detect a change that is smaller than 0.0067 *in* (roughly 0.167 mm). The double-precision floating-point number mantissa is selected as

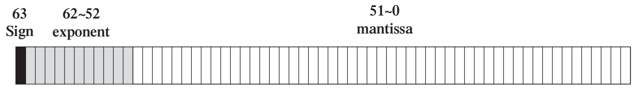


Figure 2. IEEE-754 double-precision floating-point format.

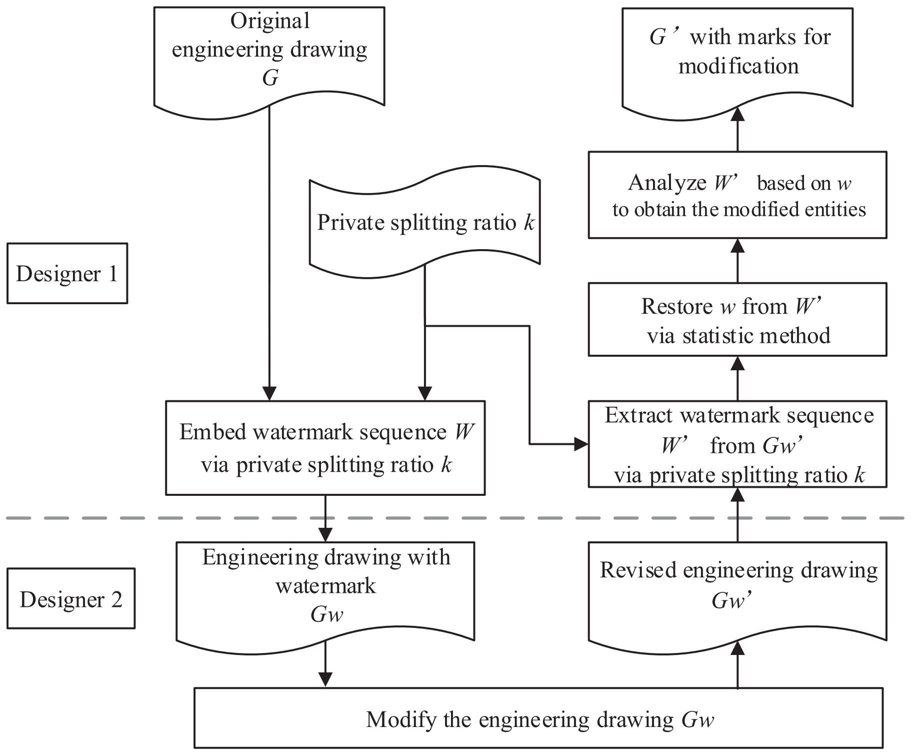

Original engineering drawing G
Private splitting ratio k
Designer 1
Embed watermark sequence W via private splitting ratio k
G' with marks for modification
Analyze W' based on w to obtain the modified entities
Restore w from W' via statistic method
Extract watermark sequence W' from Gw' via private splitting ratio k
Designer 2
Engineering drawing with watermark Gw
Revised engineering drawing Gw'
Modify the engineering drawing Gw

Figure 1. Flow chart of the proposed method.

the watermark carrier for the floating-point number, which not only meets accuracy demands but also capacity requirements.

Engineering drawings are mainly composed of lines, arcs and circles. This paper focuses on these three entities. Other entities can be processed in a similar way. *Virtual point* calculation for lines, arcs and circles is shown as follows:

1. Virtual points for lines

For the line AB shown in Fig. 3(a), $A(x_a, y_a)$ is the start point, $B(x_b, y_b)$ is the end point and $V(x_v, y_v)$ is the *virtual point*. According to the *virtual point* definition:

$\frac{AV}{VB} = k$, and based on this, the coordinate of point V is:

$$\begin{cases} x_v = \dfrac{x_a + k \times x_b}{1+k} \\ y_v = \dfrac{y_a + k \times y_b}{1+k} \end{cases}$$

2. *Virtual points* for arcs

Fig. 3(b) shows the arc AB, wherein the center is $O(x_o, y_o)$, the radius is R, $A(x_a, y_a)$ is the start point, $B(x_b, y_b)$ is the end point, and the virtual point of the arc is $V(x_v, y_v)$. Assume that the angle between line BO and the horizontal line is α, that β is the angle between line AO and the horizontal line, and that σ is the angle between line VO and the horizontal line. According to this definition:

$\frac{\overset{\frown}{AV}}{\overset{\frown}{VB}} = k$, and the corresponding calculation is:

$$\begin{cases} \sigma = \dfrac{\beta + k\alpha}{1+k} \\ x_v = R\cos\sigma + x_o \\ y_v = R\sin\sigma + x_o \end{cases}$$

3. *Virtual points* for circles

A circle is presented in Fig. 3(c), wherein the center is $O(x_o, y_o)$ and the radius is R. Suppose that the intersection between the circle and horizontal line is the start point $A(x_a, y_a)$ and that the end point $B(x_b, y_b)$ is the same intersection. $V(x_v, y_v)$ is the *virtual point* of the circle, and the angle between line VO and horizontal line is σ.

$\frac{\overset{\frown}{AV}}{\overset{\frown}{VOB}} = k$, The coordinate of point V is calculated using:

$$\begin{cases} \sigma = 360^\circ \dfrac{k}{k+1} \\ x_v = R\cos\sigma + x_o \\ y_v = R\sin\sigma + y_o \end{cases}$$

Each *watermark unit* is embedded into the ratio of every two adjacent entities. We take the line-line pair as an example to illustrate the embedding process. First, determine the start and end points of the lines, and then input the user splitting ratio $k(0 < k < 1)$. Here, points A and C are the start points of lines AB and CD, respectively, and points B and D are their end points, respectively.

Fig. 4 shows that *virtual points* V_{p1} and V_{p2} are calculated using k. p and q denote the lengths of line AB and $V_{p1}V_{p2}$, respectively. Assume that the end points of the lines are $A(x_a, y_a)$, $B(x_b, y_b)$, $C(x_c, y_c)$ and $D(x_d, y_d)$ and that the coordinates of the *virtual point* V_{p1} can be obtained using the formula presented below:

$$\begin{cases} x_{V_{p1}} = \dfrac{x_a + k \cdot x_b}{1+k} \\ y_{V_{p1}} = \frac{y_a + k \cdot y_b}{1+k} \end{cases} \tag{3.1}$$

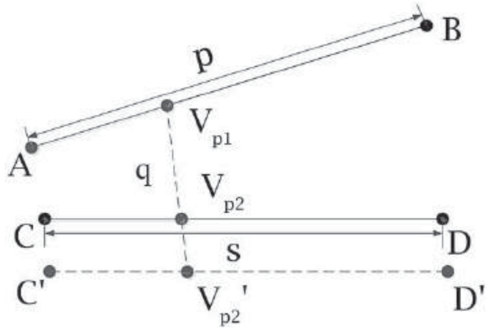


Figure 4. Embedding process principle.

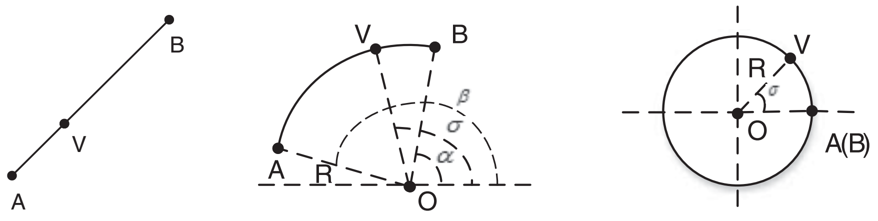


Figure 3. Virtual point calculation.

V_{p2} can be determined using the same method. The *watermark unit* w_i is embedded into the ratio of AB and $V_{p1}V_{p2}$. Line $V_{p1}V_{p2}$ is the *virtual line*. The ratio of AB and $V_{p1}V_{p2}$ is $r$based on Eqn. (3.2)

$$r = \frac{AB}{V_{p1}V_{p2}} = \frac{p}{q} \tag{3.2}$$

r is expressed using a double-precision floating-point format. The embedding algorithm is shown below, and EMB is the embed function.

$$r^w = EMB(C(w_i), pos, r) \tag{3.3}$$

r^w represents the ratio after the watermark has been embedded. $C(w_j)$ is the code value of w_i, which is discussed at length in Section 3.3. After coding has been conducted, only five bits are required. pos is the embed position. An appropriate embed position helps control embed error levels and will be discussed later. The EMB is shown in Fig. 5.

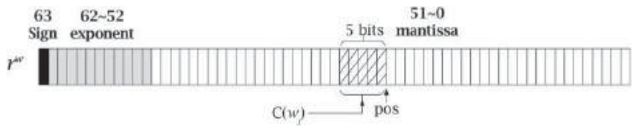


Figure 5. Ratio embed function.

After w_i is embedded, r is changed to r^w, and the *virtual length* $V_{p1}V_{p2}{}'$ can be calculated as formulated in Eqn. (3.4).

$$V_{p1}V_{p2}{}' = \frac{AB}{r^w} = \frac{p}{r^w} \tag{3.4}$$

r and r^wrepresent the ratio before and after the *watermark unit* is embedded, respectively. To adjust to changes in r, the algorithm moves the second entity along the vector $\overrightarrow{V_{p2}V_{p2}{}'}$. $\overrightarrow{V_{p2}V_{p2}{}'} = (\Delta x, \Delta y), \begin{cases} \Delta x = x_{v_{p2}} - x_{v_{p2}{}'} \\ \Delta y = y_{v_{p2}} - y_{v_{p2}{}'} \end{cases}$. Its translation matrix is T.

$$C' = C \cdot T = \begin{bmatrix} x_c & y_c & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \Delta x & \Delta y & 1 \end{bmatrix} \tag{3.5}$$

After the translation has been completed, the *watermark unit* is embedded, and the positioning of lines AB and CD change somewhat while their lengths remain invariable. Unless line AB rotates around point V_{p1}, any other operations will change the ratio r. After the embed algorithm is introduced, the embedding position and error control values are determined as follows. According to IEEE-754, the structure of N is presented below:

$$N = (-1)^S \times 2^{E-1023} \times M \tag{3.6}$$

where n, s, and m are the values of parameters N, S, and M, respectively. Eqn. (3.2) and Eqn. (3.4) show the changes that result after the *watermark unit* has been embedded:

$$\Delta q = |q - q'| \tag{3.7}$$

Apply Eqn. (3.2) and Eqn. (3.4) to Eqn. (3.7)

$$\Delta q = \left| \frac{p}{r} - \frac{p}{r^w} \right| \tag{3.8}$$

According to Eqn. (3.6)

$$r = (-1)^0 \times 2^{e_r} \times m_r,\ r^w = (-1)^0 \times 2^{e_r} \times m_{r^w} \tag{3.9}$$

Apply these to Eqn. (3.8)

$$\Delta q = \frac{p}{2^{e_r}} \cdot \left| \frac{1}{m_r} - \frac{1}{m_{r^w}} \right|$$

After creating the transformation, we get: $\Delta q = \frac{p}{(-1)^0 2^{e_r} \cdot m_r} \cdot \left| \frac{1}{m_r} - \frac{1}{m_{r^w}} \right| \cdot m_r$
And this can be simplified to obtain Eqn. (3.10):

$$\Delta q = q \cdot \left| \frac{m_{r^w} - m_r}{m_{r^w}} \right| \tag{3.10}$$

According to IEEE-754, $m_{r^w} \geq 1$:

$$\Delta q = q \cdot \left| \frac{m_{r^w} - m_r}{m_{r^w}} \right| \leq q \cdot |m_{r^w} - m_r| \tag{3.11}$$

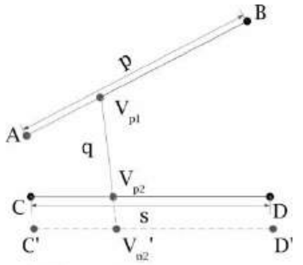
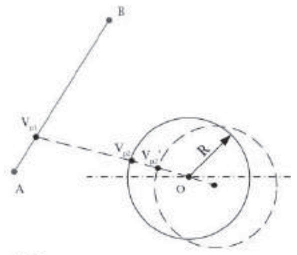
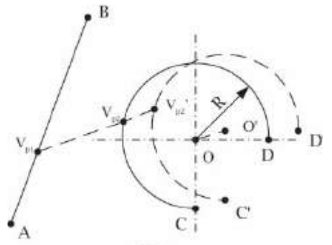
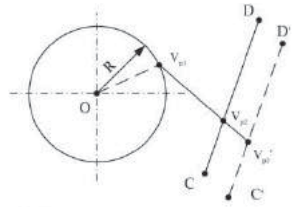
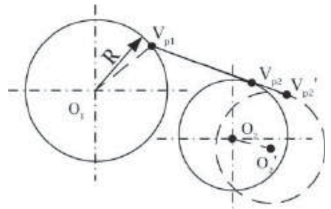
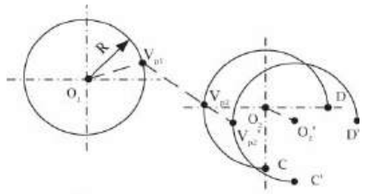
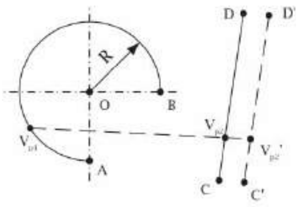
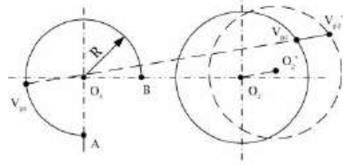
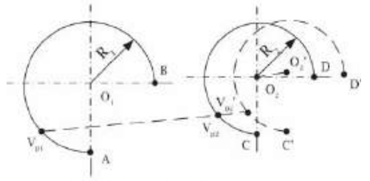
We know that $0 \leq |m_{r^w} - m_r| \leq 2^{pos-51}$ and $q = (-1)^0 \times 2^{e_q} \times m_q < 2^{e_q+1}$:

$$\Delta q < 2^{e_q+pos-50} \tag{3.12}$$

Where pos represents the embedding position. It can be concluded from Eqn. (3.12) that pos and e_q affect the embedding error level. When q is large, this means that e_q is large, and so pos must be as small as possible for the sake of reducing embed error level. However, when q is small, if pos is too small, the *watermark unit* will be lost due to computer calculation accuracy levels. In turn, the embedding position must be dynamically changed based on e_q. Based on errors and calculation accuracy levels, we let the embedding position be $pos = \delta - e_q, \delta = 25$. Here, $\delta = 25$ is obtained from experimental data that are discussed in Section 5.3. When $\delta = 25$ and e_q is different according to entity length, pos becomes a dynamic value.

According to this rule, there are nine line, arc and circle combinations. The other eight types are processed using a similar method as that used for the line-line type. More details are shown in Tab. 1.

Table 1. Embedding watermark unit for nine types of entity pairs.

	Line	Circle	Arc
Line	 $\frac{AV_{p1}}{V_{p1}B}=\frac{CV_{p2}}{V_{p2}D}=k$ $r=\frac{AB}{V_{p1}V_{p2}},r^{w}=\frac{AB}{V_{p1}V_{p2}'}$	 $\frac{AV_{p1}}{V_{p1}B}=k$ $r=\frac{AB}{V_{p1}V_{p2}},r^{w}=\frac{AB}{V_{p1}V_{p2}'}$	 $\frac{AV_{p1}}{V_{p1}B}=\frac{\overset{\frown}{CV_{p2}}}{\overset{\frown}{V_{p2}D}}=k$ $r=\frac{AB}{V_{p1}V_{p2}},r^{w}=\frac{AB}{V_{p1}V_{p2}'}$
Circle	 $\frac{CV_{p2}}{V_{p2}D}=k$ $r=\frac{R}{V_{p1}V_{p2}},r^{w}=\frac{R}{V_{p1}V_{p2}'}$	 $r=\frac{R}{V_{p1}V_{p2}},r^{w}=\frac{R}{V_{p1}V_{p2}'}$	 $\frac{\overset{\frown}{CV_{p2}}}{\overset{\frown}{V_{p2}O_2D}}=k$ $r=\frac{R}{V_{p1}V_{p2}},r^{w}=\frac{R}{V_{p1}V_{p2}'}$
Arc	 $\frac{\overset{\frown}{AV_{p1}}}{\overset{\frown}{V_{p1}OB}}=\frac{CV_{p2}}{V_{p2}D}=k$ $r=\frac{R}{V_{p1}V_{p2}},r^{w}=\frac{R}{V_{p1}V_{p2}'}$	 $\frac{\overset{\frown}{AV_{p1}}}{\overset{\frown}{V_{p1}O_1B}}=k$ $r=\frac{R}{V_{p1}V_{p2}},r^{w}=\frac{R}{V_{p1}V_{p2}'}$	 $\frac{\overset{\frown}{AV_{p1}}}{V_{p1}O_1B}=\frac{\overset{\frown}{CV_{p2}}}{V_{p2}O_2D}=k$ $r=\frac{R_1}{V_{p1}V_{p2}},r^{w}=\frac{R_1}{V_{p1}V_{p2}'}$

3.3. Embedding watermark

To adjust to the embed algorithm described above, the *watermark information* must also meet some requirements. For a given set of *watermark information w* and for a drawing with *n* entities, *watermark information* should meet the following demands:

(1) The length *m* of the *watermark information* should be shorter than *n*. This will allow one to restore *watermark information* using statistical methods after the *new watermark sequence* has been extracted from the revised drawing.
(2) However, *m* should not be too small, for this could decrease the detection accuracy level.
(3) Units of *watermark information* should vary to ensure accuracy and reliability.

Watermark information w may include the names of designers or their personal information (e.g., the name $\{j, a, c, k\}$). After expanding this, the *watermark sequence* is W. Meanwhile, character '$*$' is inserted between two w as a mark to separate them. Finally, a character '#' is added to the end of the sequence, and in turn $W = \{*, j, a, c, k, *, j, a, c, k \ldots \#\}$. The character '#' is used to distinguish between additions and modifications, which are analyzed in Section 4.3. The length of the expended *watermark sequence W* should be equal to *n* - 1 (here, *n* denotes the number of entities in engineering drawings).

The ASCII codes of characters ranging from ‘a’ to ‘z’ required seven bits of storage space in a computer program. To use less storage space and to reduce error levels resulting from bit replacement tasks, the approach presented here involves redefining the code for the 26 characters. We use the ASCII code of each character, subtract 96 from this value, and in turn obtain the new code. We define the code of the special mark ‘$*$’ as 0 and the code of the end mark ‘#’ are 27. In turn, each *watermark unit* value ranges from zero to 27, with five bits required.

The *watermark sequence W* is embedded into the engineering drawing. As discussed in Section 3.1, the embedding process is presented as follows:

(1) Go through the CAD database and obtain the entity sequence $E = \{e_1, e_2 \ldots e_n\}$. The embedding process traverses from the first entity to the last.
(2) Obtain the adjacent entities. Every two adjacent entities are a pair. The pairs are $E_i = \{e_i, e_{i+1}\}$ $(1 \le i \le n-1)$.
(3) Determine the pair type and choose the corresponding embedding method.
(4) Embed the corresponding *watermark unit* using the equation shown in Section 3.2.
(5) Repeat steps (2) $\sim$ (4) until all *watermark units* have been embedded.

3.4. Extracting watermark

The watermark extraction process is the inverse of the watermark embedding process. To extract the watermark, the private *splitting ratio k* that was used to embed the *watermark unit* is needed.

(1) First, obtain the entity sequence $E = \{e_1, e_2 \ldots e_{n'}\}$. It should be noted that entity additions and deletions may change the total number of entities, and the new number is n'.
(2) Obtain the two adjacent entities. The pairs are $E_i = \{e_i, e_{i+1}\}(1 \le i \le n-1)$. Every two adjacent entities is a pair.
(3) Determine the pair type, select the corresponding extracting method using Tab. 1.
(4) Extract the *watermark unit* from the position calculated in Section 3.2.
(5) Repeat steps (2) $\sim$ (4) until all *watermark units* have been extracted.

Add 96 to the extracted character code values to obtain the standard ASCII codes. As the drawing may be revised, the extracted new *watermark sequence W′* differs from the original one.

3.5. Restoring watermark

After the *new watermark sequence W′* is extracted from the revised drawing, the original *watermark information w* is restored fromW'. According to the algorithm described above, the original *watermark unit* w_i is more prominent than the others in the revised drawing, as the *watermark unit* extracted from the revised entities is a random character, while the *watermark unit* extracted from the unrevised drawing has not been changed. Therefore, occurrence times are used to determine the original *watermark unit*. The steps for restoring *watermark information* are as follows:

(1) The extracted *new watermark sequence W′* is stored in array $W[n]$, where n is the length of W';
(2) The two-dimensional array $T[s, u]$ is used to record the adjacent relationship, $u \in [0,2]$. $T[j, 0]$ and $T[j, 1]$ denote the adjacent *watermark unit* in the sequence, and $T[j,1]$ is behind $T[j, 0]$. $T[j, 2]$ is used to record the occurrence times of the relationship between $T[j, 0]$ and $T[j, 1]$.
(3) The s is set to zero, i is the circular variable of $W[n]$, j is the circular variable of $T[s, u]$, *i and j* are set to zero at the beginning.
(4) If $i > n$-1, then proceed to step (8); if $i < \; = n$ -1, then proceed to step (5);
(5) If $j = = s$ && $T[j, 0] = = W[i]$ && $T[j, 1] = = W[i+1]$, then $T[s, 2] = 1$ and $s = s + 1$. Proceed to step (7); if $j < s$, then proceed to step (6);
(6) If $j < s$ && if $T[j, 0] = = W[i]$ && $T[j,1] = = W[i+1]$, then make $T[j,2] = T[j,2] + 1$, and proceed to step(7); otherwise, if $j = j + 1$, proceed to step (6); if $j = = s$, proceed to step (5);
(7) If $i = i + 1$, use $j = 0$, and proceed to step (4);

After steps (4) $\sim$ (7) have been completed, adjacent *watermark unit* relationships are determined, and $T[s, u]$ is generated. We then restore the *watermark information* $w[m]$ from $T[s, u]$, where m is the number of *watermark units* and where m is set to zero at the beginning.

(8) Determine the maximum $T[j,2]$ when $T[j,0] = =$ ‘*’, record $w[m] = T[j, 1]$, and let $m = m + 1$;
(9) If $T[j,1] \neq$ ‘*’, determine the maximum $T[j,2]$ when $T[j,0] = w[m\text{-}1]$, and then record $w[m] = T[j,1]$, $m = m + 1$. Repeat step (9) until $T[j,1] = =$ ‘*’.

After steps (8) $\sim$ (9) have been completed, the original *watermark information w* is restored. This can be used for further modification detection.

4. Detection realization

4.1. Modification detection

Among various types of revisions used for design purposes, the most common operations used for entities are modifications. This section describes entity modifications as an example, assuming that the pair of entities is $\{e_i, e_{i+1}\}$ and that both are lines. Entity modifications can be divided into three categories:

(1) Move line CD along the vector $\vec{v} = (\Delta x, \Delta y)$ to obtain a new line $C'D'$, where the corresponding coordinates are $(x_c + \Delta x, y_c + \Delta y)$ and $(x_d + \Delta x, y_d + \Delta y)$. The *virtual point* V_{p2}'' can be calculated using Eqn. (3.1). Evidently, V_{p2} and V_{p2}'' are not the same point, and so $V_{p1}V_{p2} \neq V_{p1}V_{p2}''$, which means that q is changed, and r is changed too. As a result, the original *watermark unit* cannot be extracted from the ratio.

(2) Rotation is similar to translation. The ratio r is changed along with the entity rotation. Only when the entity rotates round the *virtual point* V_{p2} will the ratio r maintain invariable. However, the *virtual point* V_{p2} is calculated using the private *splitting ratio k* defined by the designer. Therefore, this special situation rarely occurs.

(3) Assume that the modified entity is e_{i+1}, i.e., CD. After the modification, the endpoint coordinates are (x_c', y_c') and (x_d', y_d'). Apply these to Eqn. (3.1) to obtain the *virtual point* V_{p2}'. Evidently, this will change ratio r, and the original watermark cannot be extracted.

The modified entity e_i may produce extracted *watermark units* of between e_{i-1} and e_i, where e_i and e_{i+1} are not the original ones. The original *watermark sequence* W is shown in Fig. 6(a). When e_i is modified, a *new watermark sequence* W' is extracted from the entities, as shown in Fig. 6(b).

The *watermark unit* extracted from e_{i-1} and e_i' is w_{i-1}', w_i', which is extracted from e_i' and e_{i+1}. Compared to the embedded *watermark information*, w_{i-1}' and w_i' do not belong to the original w. If entities $\{e_i, e_{i+1} \ldots e_j\}$ are modified, the related extracted watermark unit $\{w_i, w_{i+1} \ldots w_{j-1}\}$ does not belong to the original w. Using this feature, modified entities can be detected.

4.2. Deletion detection

As is shown in Fig. 6(c), when entity e_i is deleted, e_{i-1} moves next to e_{i+1}. Fig. 6(c) shows that a *watermark unit* is lost and that the extracted *watermark unit* from e_{i-1} and e_{i+1} does not belong to the original *watermark information w*. Clearly, one *watermark unit* is lost in the *new watermark sequence,* and an incorrect *watermark unit* is extracted accordingly. If several continuous entities are deleted, only one wrong *watermark unit* is extracted from prior and future entities. We conclude that some entities are deleted when a *watermark unit* is incorrect and that prior and future units are correct. However, the number of continuous entities that are deleted remains unknown.

The process becomes more complicated when deletion and modification tasks are performed on adjacent entities. For example, when entity e_i is modified and e_{i+1} is deleted, $w_{i-2}' == w_{i-2}$, .., $w_i' \neq w_i$, $w_{i+1}' == w_{i+1}$. The algorithm noted above assumes that the entity e_i is modified according to Section 4.1 and that deletions cannot be found. While this differs from the actual situation, the entity e_i is still regarded as modified and still has practical significance.

4.3. Addition detection

The method for entity additions differs from that for modifications and deletions described above. CAD databases store entities in chronological order. This means that added entities are placed at the end of an entity list. Therefore, entity additions do not affect

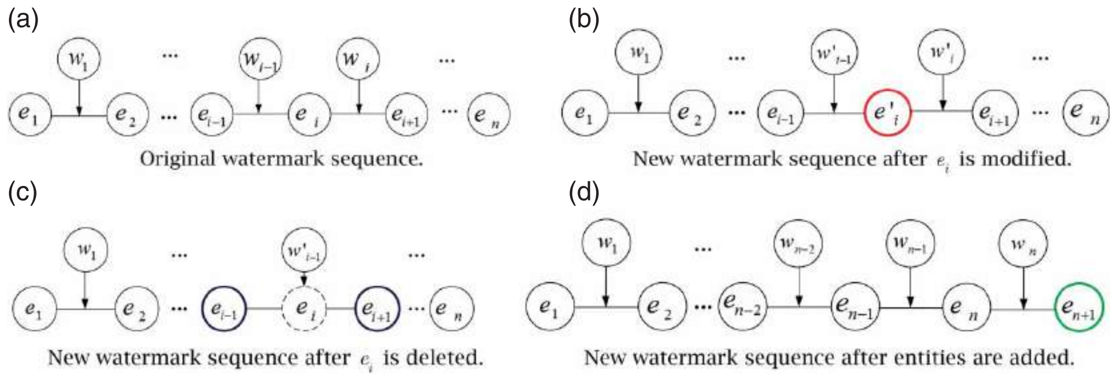

(a)
Original watermark sequence.
(b)
New watermark sequence after e_i is modified.
(c)
New watermark sequence after e_i is deleted.
(d)
New watermark sequence after entities are added.

Figure 6. New watermark sequence after entity modification, deletion and addition.

previous *watermark sequences*. If a random *watermark sequence* appears at the end of an extracted *watermark sequence*, this means that entities have been added.

In Fig. 6(d), entity e_{n+1} has been added. An incorrect *watermark unit* w_n will be extracted when the entity e_{n+1} is added. According to the embedding watermark algorithm shown in Section 3.3, w_{n-1} must be the end mark '#'. Therefore, entities are added when corresponding watermark units are behind the end mark '#'.

4.4. Detection procedure

Based on the conditions described above, the detection procedure is presented. First, a *new watermark sequence* W' is extracted from a revised drawing. Second, original *watermark information* w is restored from W'. Third, a double linked list H is built using the separator '*' and the original *watermark information* w. A pointer p points to an element of the list H. Finally, a detecting algorithm is used for W' and H. Here, we traverse the *new watermark sequence* W' and check each *watermark unit* w_i of W' with H. Generally speaking, if w_i is contained in H, the corresponding two entities remain unrevised. However, to limit chances of misjudgment, the algorithm will check not only w_i but also w_{i-1} orw_{i+1}. Reasons for this will be discussed at length in Section 5.4. The detecting procedure is outlined in Fig. 7.

Modified entities are marked in red, deleted entities are marked in blue, and added entities are marked in green. It should be noted that rather than marking deleted entities, we mark prior and future entities.

4.5. Detection example

A simple drawing of axis with all entities marked with numbers is presented in Fig. 8(a). All the entities are numbered according to their storage order. In this example, the *watermark information* w is $\{j, a, c, k\}$, and the *watermark sequence* W, expanded with w, is $\{*, j, a, c, k, *, j, a, c, k, *, j, a, c, k, *, j, \#\}$. Fig. 8(c) shows the watermark embedding process. Line 5 and line 6 in Fig. 8(a) are chosen to illustrate the process. First the

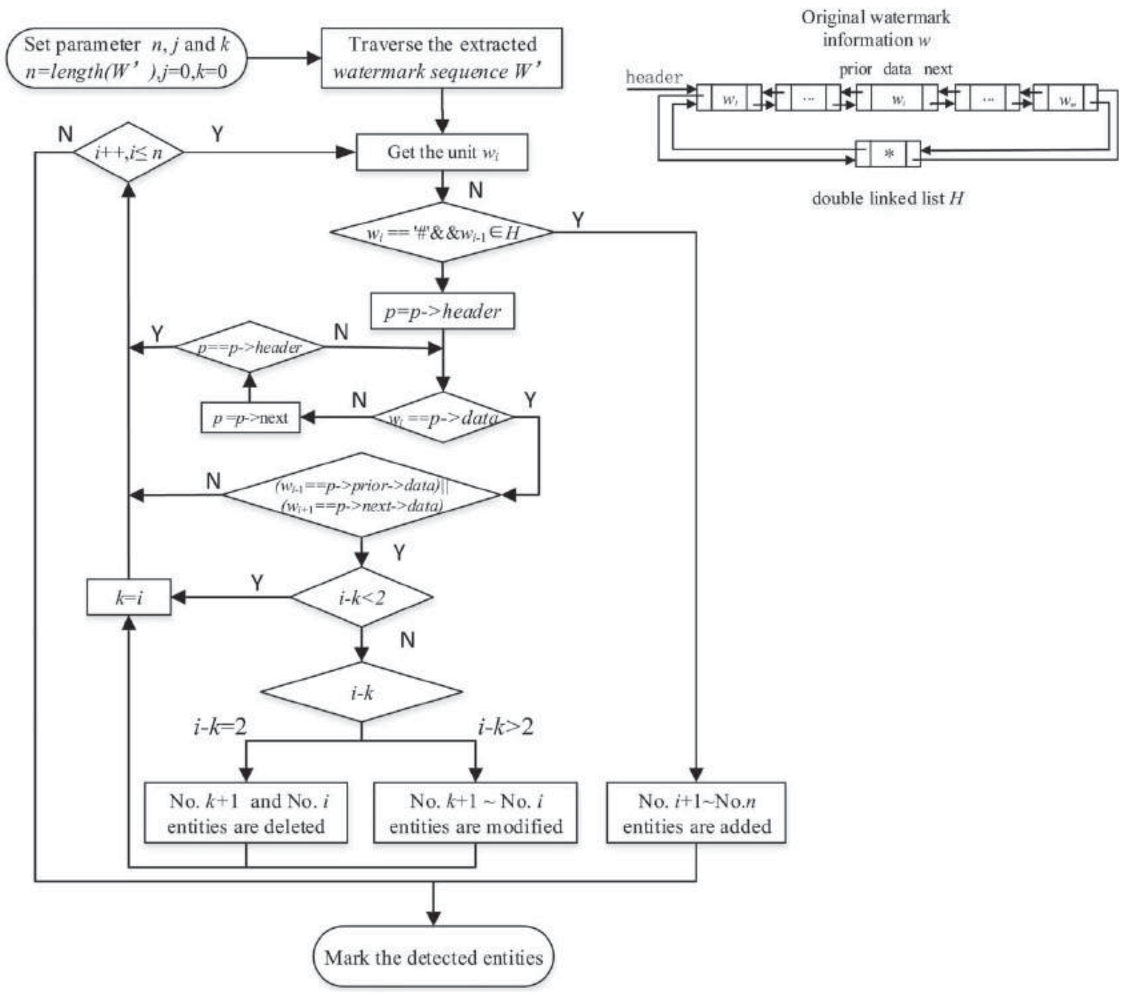

Set parameter n, j and k
n=length(W'),j=0,k=0
Traverse the extracted watermark sequence W'
Get the unit w_i
i++,i≤n
N
Y
w_i == '#'&&w_{i-1}∈H
p=p->header
p==p->header
p=p->next
w_i ==p->data
(w_{i-1}==p->prior->data)||(w_{i+1}==p->next->data)
i-k<2
k=i
i-k
i-k=2
i-k>2
No. k+1 and No. i entities are deleted
No. k+1 ~ No. i entities are modified
No. i+1~No.n entities are added
Mark the detected entities
Original watermark information w
header
prior data next
w_1
w_i
w_n
*
double linked list H

Figure 7. The proposed detection procedure.

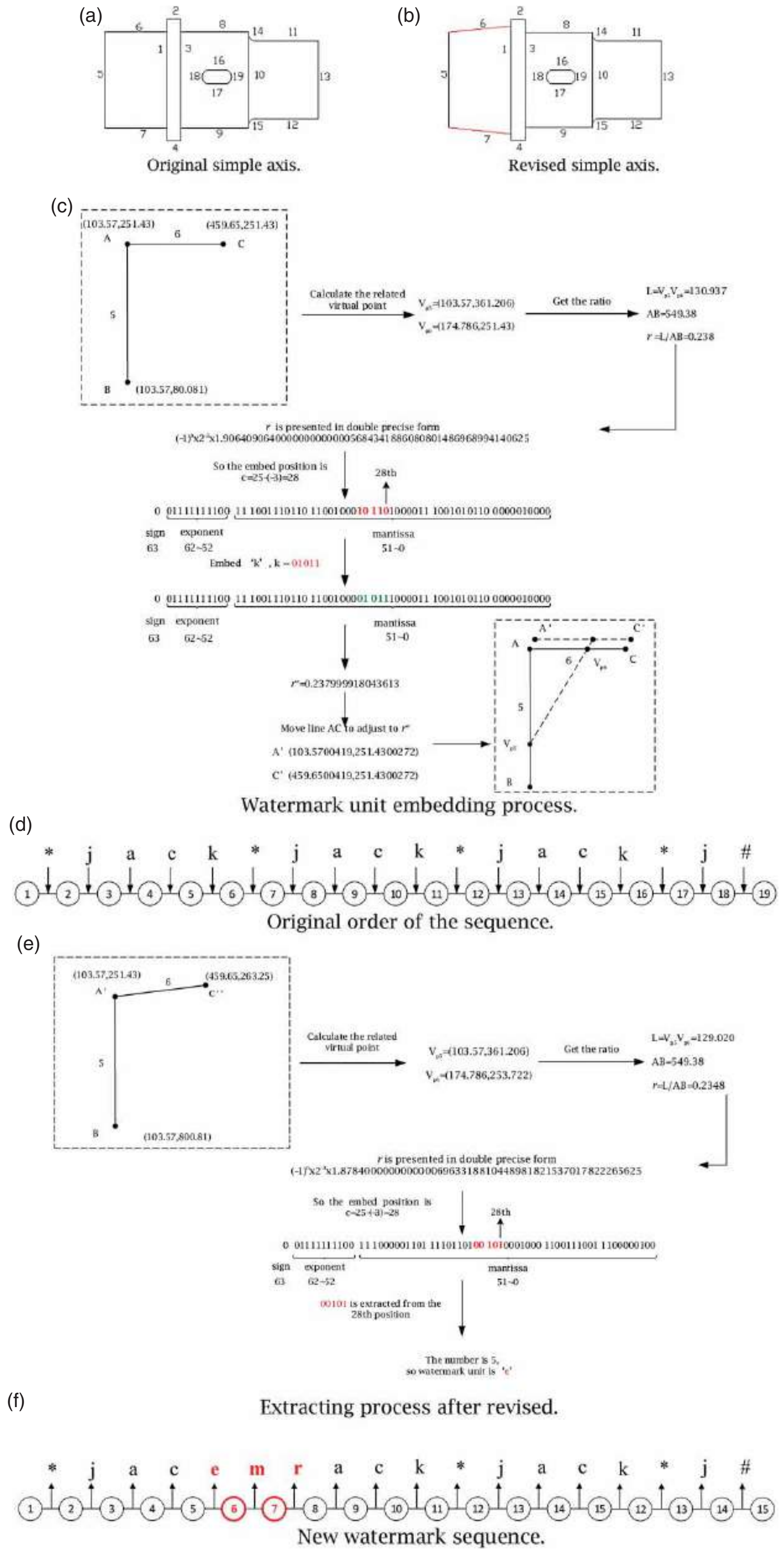
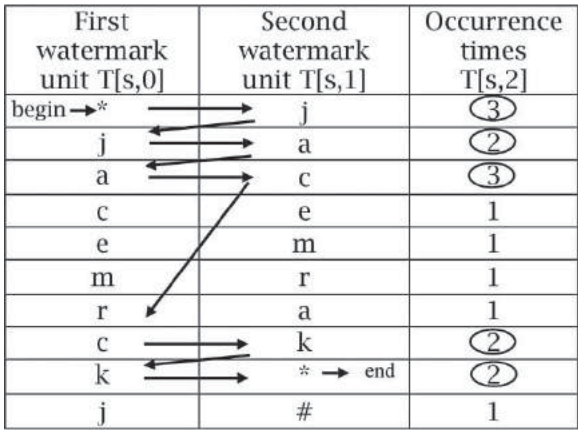

(a)
Original simple axis.
(b)
Revised simple axis.
(c)
(103.57,251.43)
(459.65,251.43)
(103.57,80.081)
Calculate the related virtual point
V_p5=(103.57,361.206)
V_p6=(174.786,251.43)
Get the ratio
L=V_p5V_p6=130.937
AB=549.38
r=L/AB=0.238
r is presented in double precise form
(-1)^0x2^-3x1.9064090640000000000005684341886080801486968994140625
So the embed position is c=25-(-3)=28
28th
0 01111111100 11 1001110110 1100100010 1101000011 1001010110 0000010000
sign 63
exponent 62~52
mantissa 51~0
Embed 'k', k=01011
0 01111111100 11 1001110110 1100100001 0111000011 1001010110 0000010000
r'=0.237999918043613
Move line AC to adjust to r'
A' (103.5700419,251.4300272)
C' (459.6500419,251.4300272)
Watermark unit embedding process.
(d)
* j a c k * j a c k * j a c k * j #
Original order of the sequence.
(e)
(103.57,251.43)
(459.65,263.25)
(103.57,800.81)
Calculate the related virtual point
V_p5=(103.57,361.206)
V_p6=(174.786,253.722)
Get the ratio
L=V_p5V_p6=129.020
AB=549.38
r=L/AB=0.2348
r is presented in double precise form
(-1)^0x2^-3x1.8784000000000000696331881044898182153701782265625
So the embed position is c=25-(-3)=28
28th
0 01111111100 11 1000001101 1110110100 1010001000 1100111001 1100000100
00101 is extracted from the 28th position
The number is 5, so watermark unit is 'e'
(f)
Extracting process after revised.
* j a c e m r a c k * j a c k * j #
New watermark sequence.

Figure 8. Watermark sequences before and after revision.

virtual point and *virtual length* is obtained. Then the ratio r is presented in double precise form. According to Eqn. (3.12) and $\delta = e_q + c(\delta = 25, e_q = -3)$, the embed position is 28. The red part in Fig. 8(c) is replaced by the *watermark unit* ‘k’. As a result, r is changed to r^w. To adjust to r^w, line 6 is moved from AC to A'C'. The original *watermark sequence* is embedded into the drawing and is shown in Fig. 8(d). After that, the drawing is sent to another designer. However, the designer changes the first segment into the cone axis, shown in Fig. 8(b).

The *new watermark sequence* is extracted from the revised drawing and is presented in Fig. 8(e). The extracting process is very similar to the embedding process. First, r is calculated. Then the red part is obtained and *watermark unit* ‘e’ is extracted. Other *watermark units* are extracted in the same way. The *new watermark sequence* is shown in Fig. 8(f).

Occurrence times are used to determine the original *watermark unit* and the process is shown in Fig. 9. The *watermark information* is restored according to Section 3.5. We begin with ‘*’ from T [s,0], and end with ‘*’ from T [s,1]. It is easy to get $\{j, a, c, k\}$ and the original *watermark information w* is restored.



First watermark unit T[s,0]	Second watermark unit T[s,1]	Occurrence times T[s,2]
begin→*	j	③
j	a	②
a	c	③
c	e	1
e	m	1
m	r	1
r	a	1
c	k	②
k	* → end	②
j	#	1

Figure 9. Statistic of occurrence frequency.

It is obvious to find that ‘e’, ‘m’, ‘r’, shown in Fig. 8(f), do not belong to $\{j, a, c, k\}$ and $\{*, \#\}$. According to the algorithm in Section 4.4, entity 6 and 7 are modified, and marked in red in Fig. 8(b).

5. Result and analysis

5.1. Experimental results

The proposed method is implemented for a PC with CPU Core i5 1.8 GHz, RAM 4 GB, Win7 Professional, AutoCAD 2007 and ObjectARX 2008 specifications. A mechanical drawing that includes 1,135 entities was used. The drawing shown in Fig. 10(a) is a practical drawing. Designer 1 embeds *watermark information* using his private splitting ratio k after completing the drawing. He sends the drawing to designer 2 to evaluate manufacturability levels. Designer 2 revises the drawing according to technical requirements. The revised drawing is then returned to designer 1 for approval. Designer 1 receives the revised drawing and detects modifications made using the proposed approach. First, the program extracts the *new watermark sequence* and then restores the original *watermark information*. Then, the program analyzes the *new watermark sequence* using the original *watermark information* and highlights the modifications in different colors.

As is shown in Fig. 10(b), entities in place I are created by designer 2 to increase the diameter of the hole and the radius of the arc. They are judged as modifications by the program, and so they are marked in red. The diameter of the shaft in place II is increased to facilitate its manufacture, and these entities are marked in red as well. In accordance with assembly requirements, the width of the bearing in place III is increased. The program detects this modification and marks it in red. The thread in place IV is not required and is thus deleted, and corresponding prior and future entities are marked in blue by the program. Two holes in place V are required for installation purposes, and so they are added by designer 2. The program detects this as an addition, and marks it in green.

As embedding distortion, detection reliability, and detection accuracy are important properties of semi-fragile watermarking, we will discuss them in detail in the follow sections.

5.2. Discussion of reliability

To evaluate reliability levels, the *modification detection rate* (*MDR*) is defined as follows:

$$MDR = \frac{N_D}{N} \tag{5.1}$$

Where N is the total number of entities found in engineering drawings and where N_D is the number of revised entities that are detected. Experiments were carried out on 50 engineering drawings to examine *MDR* values when watermarked engineering drawings are rotated or scaled. In the experiments, the *watermark information* length and separate mark ‘*’ take a value of five, denoted by b = 5. The results are shown in Fig. 11 and Fig. 12.

MDR is closely related to the *pos*, where *pos* denotes the embed position. The detection rate is acceptable after rotation and scaling when $pos \geq 20$, while the *MDR* is unsatisfactory when $pos \leq 20$. These two figures only

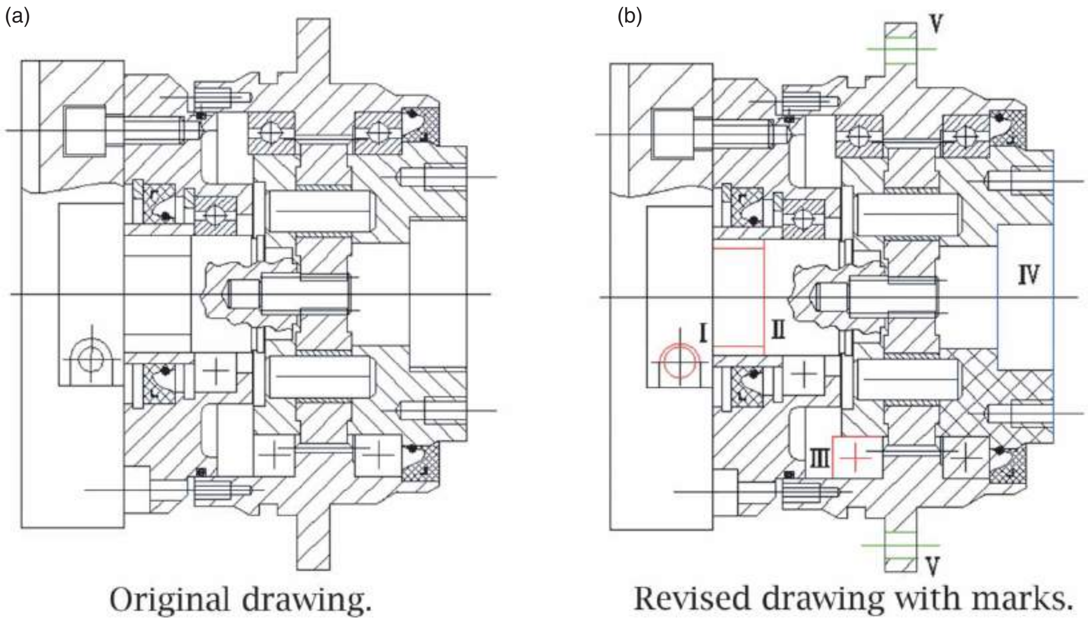


Figure 10. Example of modification detection in an engineering drawing.

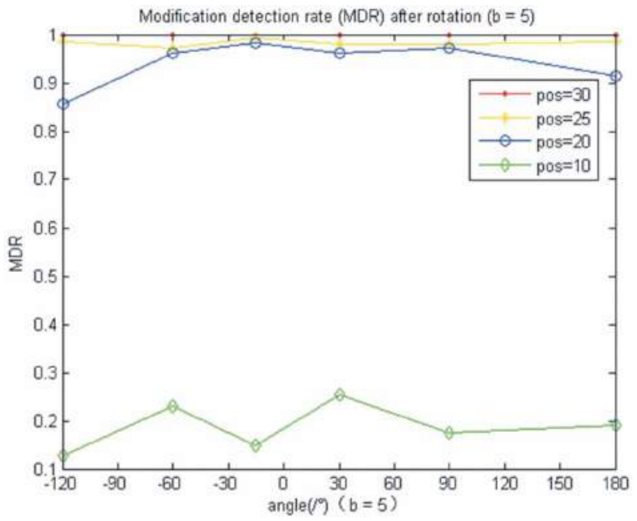


Figure 11. Modification detection rate after rotation.

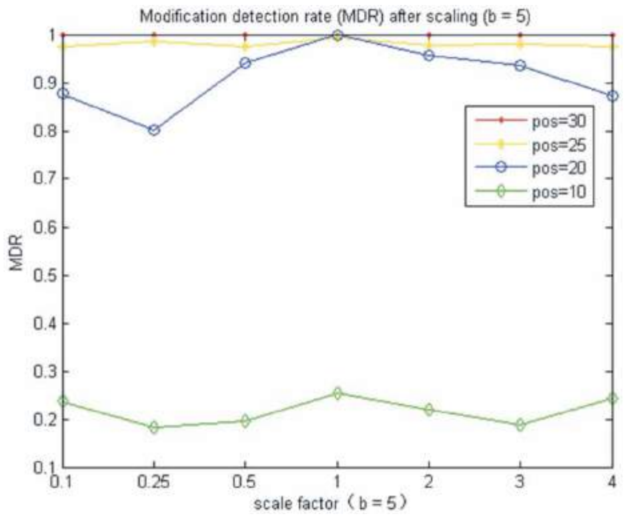


Figure 12. Modification detection rate after scaling.

show the dynamic trend of *MDR*, how to control *pos* will be discussed in the next section.

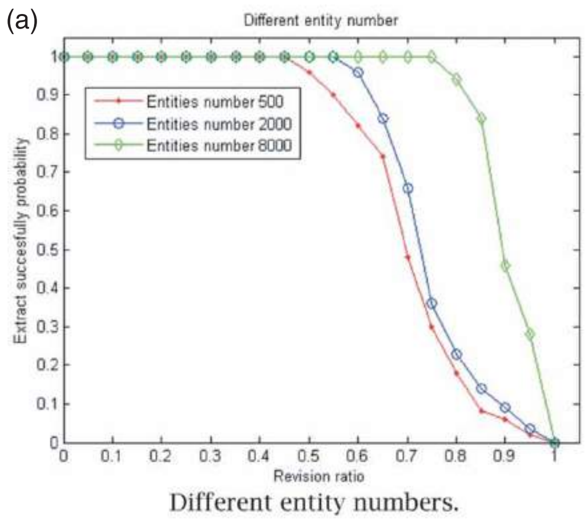
The scheme can detect not only single operations (e.g., modification, deletion and addition) but also mixed operations. According to the detecting algorithm, if mixed operations are performed on discontinuous entities, we can also detect the positions wherein entities are modified. To verify this, engineering drawings with roughly 500 entities, 2,000 entities and 8,000 entities are used to conduct the experiment. Every drawing is experimented on 50 times. The *revision rate (R)* is defined as follows:

$$R = \frac{N_R}{N} \tag{5.2}$$

Where N is the total number of entities in engineering drawings and where N_R is the number of revised entities. Fig. 13(a) and Fig. 13(b) show the experimental results for three types of revisions. When applied for modification detection purposes, performance levels are acceptable. However, the algorithm is not suitable if most of entities have been revised.

5.3. Discussion of distortion

Theoretically speaking, distortion levels should meet design error requirements if the watermark is embedded in the end of the mantissa, but they may also reduce



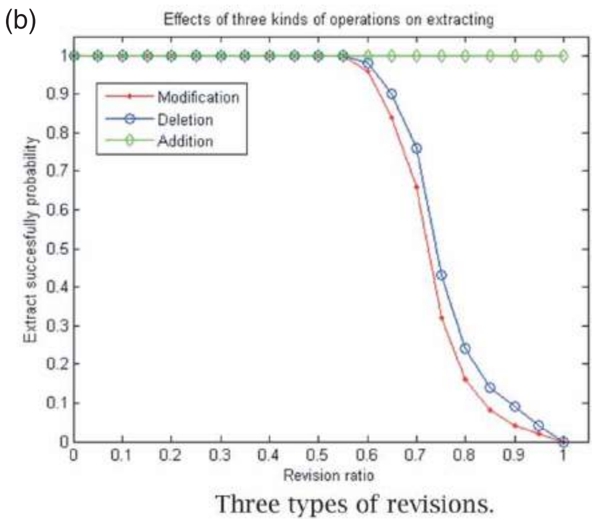


Figure 13. Experiments on different drawings.

semi-fragility levels when detecting normal revisions, as a floating number is not an accurate number (especially the value of the end of the mantissa). However, embedding watermarks in the beginning of the mantissa may also visibly distort engineering drawings, and so the middle sections of the mantissa serve as a reasonable compromise between false-negative rate and distortion levels. We also use *RMS* (root mean square) values to evaluate distortion levels. The *RMS* is formulated in Eqn. (5.3).

$$RMS = \frac{1}{N}||l - l'|| \tag{5.3}$$

l and l' denote the virtual length between two entities before and after the watermark is embedded, respectively, and N is the number of entities. Data are then collected via experimentation. The relationship between *RMS* and the embedding position, given a fixed *watermark information* length of b, is shown in Fig. 14. When the embedding position extends beyond 25, the *RMS* increases rapidly. Thus, *pos* should be less than 25.

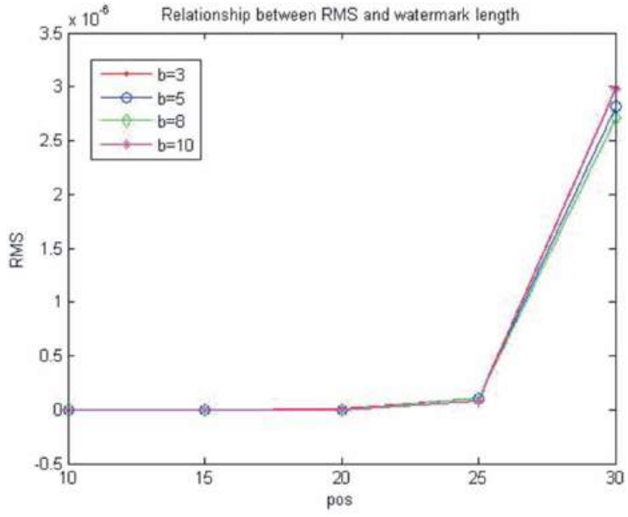


Figure 14. Relationship between *RMS* and *b*.

As noted in Section 3.1, the error is controlled by the watermark carrier index and by the embedding position, i.e., $\delta = e_q + pos$. It is important to select an appropriate δ to control the error level. According to Eqn. (3.12), $\delta_{\max} < 50 + \log_2 \Delta q$. Different δ values are used to conduct the experiment to select the best δ value with the lowest error level. Every experiment is repeated 10 times for different engineering drawings. The partial results shown in Tab. 2 are optimal. The experiment results show that smaller δ values can be preferable; in fact, the integrity of an extracted watermark cannot be ensured when δ is too small. Based on the error level and the engineering drawings, $\delta = 25$ is used to maintain a balance between the experiments conducted in this study.

In addition, the decimal digits of the splitting ratio k also affect the calculation during watermark embedding. In the experiment, different values of k ranging from one to five decimal digits are used to determine effects on watermark extraction outcomes. The test results for different decimal digits are shown in Tab. 3.

Table 2. Relationship between δ and Δq.

δ	28	25	22	19
Δq	2.38e^{-7}	2.98e^{-8}	3.72e^{-9}	4.65e^{-10}

Table 3. Decimal digits on extraction.

Decimal digit of k	1	2	3	4	5
Effects on extraction	Pass	Pass	Pass	Pass	Fail

When a decimal digit meets design requirements, it is less able to reduce error accumulation levels. The relationship between *RMS* and decimal digit values is shown in Fig. 15. The order of magnitude for error values is 10^{-8}; Δ represents variation range in the figure, which means the *RMS* becomes stable in a range when decimal digit of k increases; and this is more accurate and suitable for use with engineering design drawings than the *RMS* described in [12]. In taking safety and precision issues into consideration, four decimal digits are recommended when selecting the splitting ratio k value, according to Tab. 3 and Fig. 15.

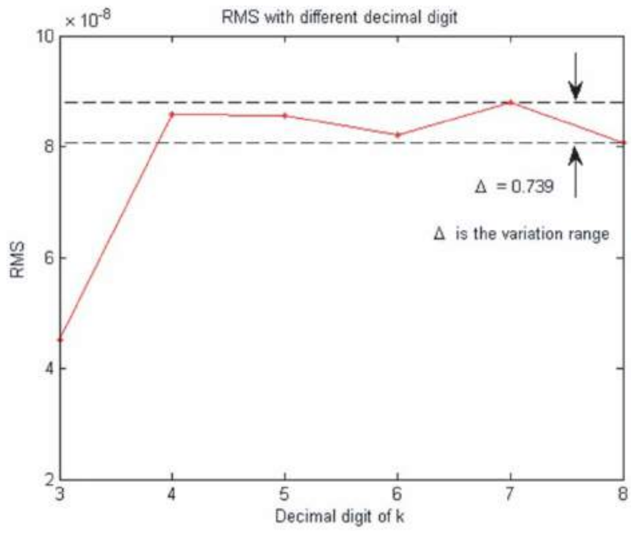


Figure 15. *RMS* with different decimal digits of k.

5.4. Discussion of accuracy

In fact, a certain issue will affect performance levels. When certain entities are modified, the corresponding *watermark unit* that is extracted should not be equal to the original one. In this paper, the code of a *watermark unit* is denoted by five bits, as noted in Section 3.3. When one entity is modified, the corresponding five bits are extracted from the embed position, and the bits are random as a result of modifications. In turn, the probability of extracting the original code is $P_{error} = 1/2^5$, and the related accuracy level is $P_{accuracy} = 1 - P_{error} = 96.875\%$. For example, the sequence between entities e_i and e_{i+5} is $\{w_i, w_{i+1}, w_{i+2}, w_{i+3}, w_{i+4}\}$. When e_{i+3} is modified, the *watermark units* w_{i+2} and w_{i+3} are changed according to the scheme. In actually, the original *watermark unit* w_{i+2} can be extracted instead of the incorrect *watermark unit* w_{i+2}'. In turn, the sequence is $\{w_i, w_{i+1}, w_{i+2}, w_{i+3}{}', w_{i+4}\}$. This is not consistent with the theoretical results, which will lead to inaccurate judgments and which will seriously lower detection accuracy levels.

Table 4. Detection accuracy of different operations.

Operations	Modifications	Deletions	Additions	Mixing operations
Accuracy	99.319%	97.793%	99.127%	98.351%

Table 5. The comparison of embedding distortion between the proposed method and the one in [12].

RMS with method in [12]			*RMS* of the proposed method		
N	*RMS*	Δq	δ	*RMS*	Δq
4	5.65E-4	1.38E-2	28	2.03E-8	8.07E-8
6	1.62E-4	1.46E-2	25	2.64E-9	1.39E-8
8	8.28E-5	1.27E-2	22	3.29E-10	1.41E-9
10	3.62E-5	1.32E-2	19	3.99E-11	1.92E-10

Probability levels, which result in randomness due to revisions, are assumed to be P'_{error}. To reduce probability levels, we check not only the watermark units but also prior and future watermark units. As a result, $P'_{error} = (1/2^5)^2$ and $P'_{accuracy} = 1 - P'_{error} = 99.90234375\%$. Given 50 different 2D CAD engineering graphics, after semi-fragile watermarks have been embedded into them, various malicious operations such as modifications, additions, deletions and mixed operations are carried out to create 50 tampered 2D CAD engineering graphics. The proposed method is used for tampering detection. We assume that the number of malicious operations in a tampered graphic is N_T and the number of detected malicious operations isN_D. The detection accuracy is N_D/N_T. 100 tests were conducted and results are presented in Tab. 4.

5.5. Performance comparison between the proposed method and the method in [12]

Some comparative experiments have been carried out and we analyze on the result produced with our method and the method in paper [12].

The embedding distortions of the two methods is compared by conducting experiments on 50 2D CAD graphics, and the average results are presented in Tab. 5. From the results in Tab. 5 , we find that the proposed method causes less distortion than that of the method in [12].

Apart from the embedding distortion, our method can detect and locate modifications to single entity. The precision is higher than the methods in [6, 12, 22, 27–29] for they can only detect and locate the modifications to certain groups or regions. Based on this feature, our method is more practical in collaborative design.

6. Conclusions

In this paper, we propose a ratio-based semi-fragile watermark scheme for modifications detection. Compared with past approaches of watermarking, our method has the following advantages. First, the method can accurately locate the modification to one single entity while the past approaches can only locate the modification to certain group. It is more appropriate for checking drawing modifications in collaborative design. Second, ratios of *virtual lines* are used as watermark carrier, making it tolerant to common incidental operations such as rotation, translation and scaling, but sensitive to modifications. In addition, the splitting ratio k keep watermark information safe and can be used for drawing authentication.

In our future work, we intend to explore cases with irregular entities such as splines (e.g., B splines), Bézier curves. As vector drawings have the same data structures as CAD drawings, more applications for other vector drawings (e.g., vector maps) will be examined in long-term studies.

Acknowledgments

This project is supported by National Natural Science Foundation of China (Grant No. #51475186).

ORCID

Yong-Fu Chen http://orcid.org/0000-0002-2601-0042

References

[1] IEEE standard for binary floating-point arithmetic.ANSI/IEEE Standard 754–1985. 1985.

[2] CAM, B.; Bruno, F.; Marcelo, D.; Rubens, M.; Sérgio, S.: Distributed object model for collaborative CAD environments based on design history, *Advances in Engineering Software*, 34(10), 2003, 621–631. http://dx.doi.org/10.1016/S0965-9978(03)00095-4

[3] Chin, K.; Duan, G.; Tang, X.: A computer-integrated framework for global quality chain management, *International Journal of Advanced Manufacturing Technology*, 27, 2006, 547–560. http://dx.doi.org/10.1007/s00170-004-2202-8

[4] Ding, J.; Zhang, T.; Wang, S.; Wang, W.: Electronic Document Management System Oriented Collaborative Design, *Applied Mechanics and Materials*, 44, 2011, 2036–2039.

[5] Dinga, J.; Zhang, T.; Yu, T.; Wang, W.: Study and Development of the Document Manament System Oriented Collaborative Design, *Advanced Materials Research*, 433, 2012, 2047–2052. http://dx.doi.org/10.4028/www.scientific.net/AMR.433-440.2047

[6] Guo, R.; Peng, F.: Semi-fragile watermarking algorithm for 2D engineering graphics based on improved odd-Even quantization, *Journal of Chinese Computer Systems*, 10, 2010, 038.

[7] Kumar, M.; Rajotia, S.: Integration of process planning and scheduling in a job shop environment, *International Journal of Advanced Manufacturing Technology*, 28, 2006, 109–116. http://dx.doi.org/10.1007/s00170-004-2317-y

[8] Li, W.; Lu, W.F.; Fuh, J.Y.; Wong, Y.: Collaborative computer-aided design—research and development status, *Computer-Aided Design and Applications*, 37(9), 2005, 931–940. http://dx.doi.org/10.1016/j.cad.2004.09.020

[9] Liu, J.; Duan, N.; Peng, X.: The realization of homework intelligence correction arithmetic, *Journal of Engineering Graphics*, 26(5), 2006, 137–143.

[10] MS, H.: Rules and models for low cost design, *Des Eng Des Manufacturability ASME*, 52, 1993, 75–84.

[11] Patel, H.; Patoliya, J.; Panchal, P.; Patel, R.: Digital Robust Video Watermarking using 4-level DWT, *International Journal of Advanced Engineering Technology*, 1(3), 2010, 101–113.

[12] Peng, F.; Guo, R.; Li, C.; Long, M.: A semi-fragile watermarking algorithm for authenticating 2D CAD engineering graphics based on log-polar transformation, *Computer-Aided Design*, 42(12), 2010, 1207–1216. http://dx.doi.org/10.1016/j.cad.2010.08.004

[13] Preda, R.O.: Semi-fragile watermarking for image authentication with sensitive tamper localization in the wavelet domain, *Measurement*, 46(1), 2013, 367–373. http://dx.doi.org/10.1016/j.measurement.2012.07.010

[14] Quanling, M.: Computer network based analysis and design of drawing document management system, in: World Automation Congress (WAC), 2012. IEEE, 2012, pp. 1–4.

[15] Sandipbhai, M.; Ketki, P.: Tamper localization in wavelet domain using semi-fragile watermarking, International Journal of Engineering Development and Research, 2014.

[16] Sanjay, R.; Balasubramanian, R.: A chaotic system based fragile watermarking scheme for image tamper detection, *AEU-International Journal of Electronics and Communications*, 65(10), 2011, 840–847. http://dx.doi.org/10.1016/j.aeue.2011.01.016

[17] Sengul, D.; Turker, T.; Engin, A.; Arif, G.: A robust color image watermarking with singular value decomposition method, *Advances in Engineering Software*, 42(6), 2011, 336–346. http://dx.doi.org/10.1016/j.advengsoft.2011.02.012

[18] Singh, P.; Chadha, R.: A survey of digital watermarking techniques, applications and attacks, *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(9), 2013, 165–175.

[19] T, B.J.; Steven, L.; F, M.N.: Copyright protection for the electronic distribution of text documents, *Proceedings of the IEEE*, 87(7), 1999, 1181–1196. http://dx.doi.org/10.1109/5.771071

[20] Tan, Q.; Chen, G.-X.: The method of mechanical vector graphics comparison AutoCAD environment, *Journal of Engineering Graphics*, 24(4), 2004, 133–137.

[21] Tseng, K.C.; El-Ganzoury, W.; Abdalla, H.: Integration of Non-networked Software Agents for Collaborative Product Development, *Computer-Aided Design and Applications*, 2(1–4), 2005, 377–386. http://dx.doi.org/10.1080/16864360.2005.10738386

[22] Wang, N.; Men, C.: Reversible fragile watermarking for 2-D vector map authentication with localization, *Computer-Aided Design*, 44(4), 2012, 320–330. http://dx.doi.org/10.1016/j.cad.2011.11.001

[23] Wang, W.-B.; Zheng, G.-Q.; Yong, J.-H.; Gu, H.-J.: A numerically stable fragile watermarking scheme for

authenticating 3D models, *Computer-Aided Design*, 40(5), 2008, 634–645. http://dx.doi.org/10.1016/j.cad.2008.03.001

[24] Wu, S.; Huang, J.; Huang, D.; Shi, Y.Q.: Efficiently self-synchronized audio watermarking for assured audio data transmission, *Broadcasting, IEEE Transactions on*, 51(1), 2005, 69–76.

[25] Yin, Z.; Xin, S.: Development of AutoCAD Graphic Database Management System Based on Component Technology, *Advanced Materials Research*, 998, 2014, 1394–1297.

[26] Yu, J.; Cha, J.; Lu, Y.; Zhuang, Y.: A remote CAE collaborative design system for complex product based on design resource unit, *International Journal of Advanced Manufacturing Technology*, 53, 2011, 855–866. http://dx.doi.org/10.1007/s00170-010-2908-8

[27] Zhang, H.; Gao, M.: A semi-fragile digital watermarking algorithm for 2D vector graphics tamper localization, 2009 International Conference on Multimedia Information Networking and Security, 2009. http://dx.doi.org/10.1109/mines.2009.224

[28] Zheng, L.; Li, Y.; Feng, L.; Liu, H.: Research and implementation of fragile watermark for vector graphics, in: Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, IEEE, 2010, 522–525.

[29] Zheng, L.; You, F.: A fragile digital watermark used to verify the integrity of vector map, in: E-Business and Information System Security, 2009. EBISS'09. *International Conference on, IEEE*, 2009, 1–4. http://dx.doi.org/10.1109/ebiss.2009.5137869