Applied Soft Computing Journal 89 (2020) 106112

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

An infinite-resolution grid snapping technique based on fuzzy theory

Ten Watanabe, Tomohito Yoshikawa, Tomohiko Ito [*], Yuto Miwa, Takeshi Shibata, Sato Saga

Muroran Institute of Technology, 27-1 Mizumoto-cho, Muroran, Hokkaido, 050-8585, Japan

ARTICLE INFO

Article history:
Received 19 October 2018
Received in revised form 31 October 2019
Accepted 19 January 2020
Available online 27 January 2020

Keywords:
CAD
Sketch-based interface
Fuzzy theory

ABSTRACT

In ordinary CAD systems with pointing devices, users input geometric objects by inputting their feature points one by one through pointing and dragging operations. Thus, each of the feature points is snapped into place interactively, and the geometric objects are aligned on the specified grid as a result. In sketch-based CAD systems with pen input devices, users simply draw a freehand stroke that the system recognizes as a geometric object, and then the system must automatically snap all of the feature points to the grid by a batch process. In this case, it is not easy for the user to set up the grid resolution in advance, because the appropriate resolution changes according to the drawing. Therefore, a multi-resolution fuzzy grid snapping (MFGS) technique has been proposed. In MFGS, the appropriate snapping resolution is dynamically selected in a multi-resolution grid system according to the roughness of the drawing manner. However, the multiplicity of the grid resolutions in MFGS is limited to a finite number, and grid snapping outside this range of resolutions cannot be appropriately handled. In this paper, we propose infinite-resolution fuzzy grid snapping (IFGS), in which there is an infinite number of grid resolutions, and experimentally demonstrate that IFGS effectively resolves the problems of MFGS.

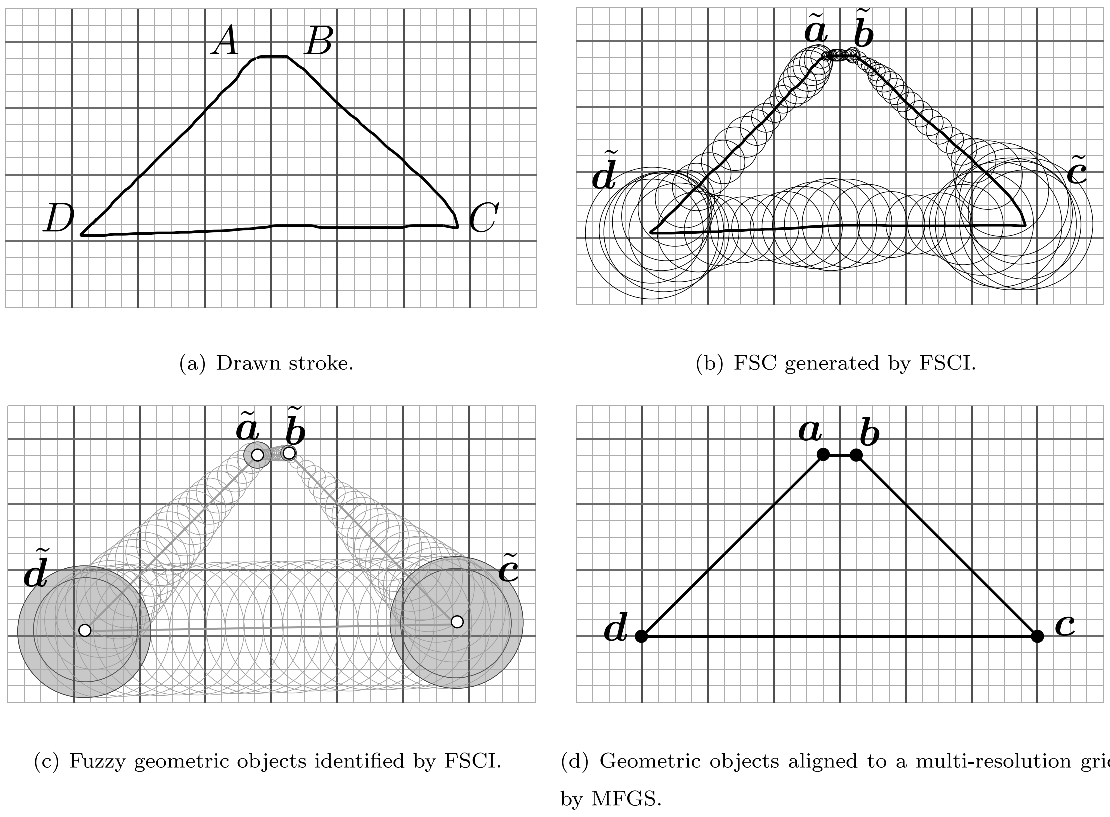
1. Introduction

In recent years, the popularity of touch screens and LCD tablets has resulted in the development of many sketch-based drawing systems, such as Interactive Beautification [1] and Fluid Sketches [2]. However, because the variety of geometric objects recognized by these systems is limited to simple items such as lines or circles, they are not suitable for general use in CAD (Computer-Aided Design or Computer-Aided Drawing) systems. To realize a general-purpose sketch-based CAD system, we proposed a freehand curve identification method called fuzzy spline curve identifier (FSCI) [3–5] that identifies a drawn stroke as a sequence of geometric objects, each of which is one of seven geometric curves (a line, circle, circular arc, ellipse, elliptic arc, closed free curve, or open free curve), based on the shape of the drawn curve and the fuzziness (or coarseness) of the drawing manner. Then, based on the essential idea of FSCI, a freehand drawing system called FFDS was developed [6]. We later developed a sketch-based interface for CAD systems by combining FSCI and some grid snapping methods [7,8].

In CAD systems, snapping processes to align geometric objects are an important part of drawing geometric figures. To enhance the alignment functions, many studies have examined constraint-based drawing systems, such as Sketchpad [9], Briar [10], and Snap-Dragging [11], which increase the variety of constraints. However, these studies have not focused on automatic resolution settings for the constraints. For alternately drawing fine objects and coarse objects, users must switch to an appropriate snapping resolution to ensure proper alignment, and frequent resolution switching can disturb the essential drawing operation. To solve this problem, the techniques of HyperSnapping [12] and Snap-and-go [13] have shown that the dynamic selection of snapping resolutions from a multi-resolution grid system is effective. With these systems, users can point and drag each feature point sequentially to align the geometric objects on the grid. In sketch-based CAD systems where feedback is provided after each pen stroke, the recognition of geometric objects followed by automatic snapping for all of the feature points of the objects must be performed by a batch process immediately after the pen is lifted. For this purpose, we proposed the multi-resolution fuzzy grid snapping (MFGS) method [14,15], which provides multiple snapping resolutions for the sketch-based CAD interfaces of [7] and [8]. MFGS snaps the feature points of geometric objects recognized by FSCI to a multi-resolution grid. This method realized the dynamic selection of an appropriate resolution for each feature point by making use of the fuzziness of the drawing manner involved in [3–8].

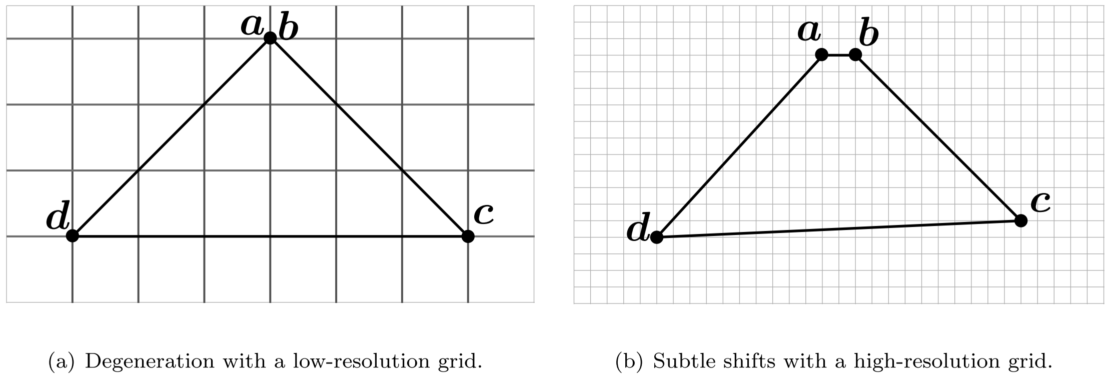
[*] Corresponding author.
E-mail address: 19096003@mmm.muroran-it.ac.jp (T. Ito).

https://doi.org/10.1016/j.asoc.2020.106112
1568-4946/© 2020 Elsevier B.V. All rights reserved.



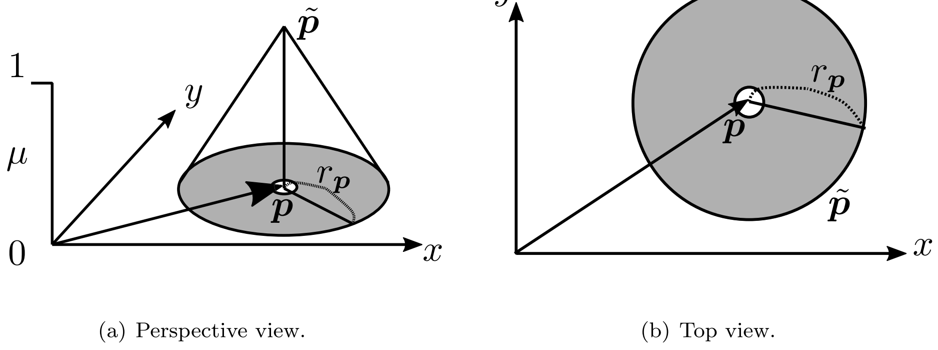
(a) Drawn stroke. (b) FSC generated by FSCI. (c) Fuzzy geometric objects identified by FSCI. (d) Geometric objects aligned to a multi-resolution grid by MFGS.

Fig. 1. Geometric object identification by FSCI and multi-resolution grid snapping by MFGS.



(a) Degeneration with a low-resolution grid. (b) Subtle shifts with a high-resolution grid.

Fig. 2. Problems with single-resolution grid snapping.



(a) Perspective view. (b) Top view.

Fig. 3. Conical fuzzy point $\tilde{p}$.

With MFGS, users can avoid the troublesome operation of switching the grid resolution in wide range of drawing situations. However, because the number of available resolutions is finite, MFGS cannot appropriately handle snapping to resolutions outside the defined range. Accordingly, users must re-set the multi-resolution grid every time their drawing situation becomes incompatible with the settings. The easiest way to deal with this problem might be to increase the number of resolutions of MFGS so that all possible drawing situations are covered with a sufficient margin. However, as the number of resolutions is increased, the computational efficiency of MFGS, which is based on a brute force approach, is greatly reduced.

To solve this problem, we propose the concept of infinite-resolution fuzzy grid snapping (IFGS), which is an extension of MFGS that increases the number of resolutions to infinity. Then, by investigating the properties of fuzzy grids, we propose an efficient search algorithm to select an appropriate resolution from an infinite number of possible resolutions. Furthermore, we experimentally demonstrate that IFGS avoids the need to re-set the multi-resolution grid, even when the drawing situation changes significantly.

2. Overview of multi-resolution fuzzy grid snapping

To explain the proposed technique, we first summarize MFGS, which is the basis of IFGS, with regard to its strengths, grid setting, algorithms, and problems.

2.1. Strengths of MFGS

The MFGS [14,15] method snaps the fuzzy feature points of geometric objects identified by FSCI [4] to a multi-resolution grid. The following text describes the process of geometric object identification by FSCI and geometric object alignment by MFGS with the example shown in Fig. 1, where the multiplicity of the multi-resolution grid is 2.

(1) A drawing stroke is input by the user using a pen. In the example shown in Fig. 1(a), the user, intending to draw a trapezoid, has drawn a stroke that travels through A, B, C, D, and A.

(2) Immediately after the pen is lifted, FSCI generates a fuzzy spline curve (FSC) by adding the fuzziness of the positional information to the drawn stroke according to the roughness of the drawing manner, as shown in Fig. 1(b). Here, the FSC is represented as the locus of a moving conical fuzzy point $\tilde{\boldsymbol{p}} = \langle \boldsymbol{p}, r_{\boldsymbol{p}} \rangle$, which is illustrated in Fig. 3. This $\tilde{\boldsymbol{p}}$ is a fuzzy set characterized by a conical membership function

$$\mu_{\tilde{\boldsymbol{p}}}(\boldsymbol{v}) = \left(1 - \frac{\|\boldsymbol{v} - \boldsymbol{p}\|}{r_{\boldsymbol{p}}}\right) \vee 0, \tag{1}$$

where $\boldsymbol{v}$ is a variable position vector, $\boldsymbol{p}$ is the position vector of $\tilde{\boldsymbol{p}}$, $r_{\boldsymbol{p}}$ is the fuzziness[1] of $\tilde{\boldsymbol{p}}$, and $\vee$ represents the max operator.

(3) FSCI identifies the FSC and generates one or more fuzzy geometric objects. In the example shown in Fig. 1(c), four fuzzy lines $\tilde{\boldsymbol{a}}\tilde{\boldsymbol{b}}$, $\tilde{\boldsymbol{b}}\tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{c}}\tilde{\boldsymbol{d}}$, and $\tilde{\boldsymbol{d}}\tilde{\boldsymbol{a}}$ have been generated.

[1] $r_{\boldsymbol{p}}$ is obtained by the fuzziness generator [16]. The fuzziness generator associates small fuzziness (fine positioning) with carefully drawn portions of the curve, and associates large fuzziness (coarse positioning) with roughly drawn portions, on the basis of the velocity and acceleration of the drawing manner.

(4) MFGS aligns the fuzzy geometric objects by snapping all of their fuzzy feature points simultaneously to the multi-resolution grid by a batch process. In the example shown in Fig. 1(d), four fuzzy feature points $\tilde{\boldsymbol{a}}$, $\tilde{\boldsymbol{b}}$, $\tilde{\boldsymbol{c}}$, and $\tilde{\boldsymbol{d}}$ have been simultaneously snapped to four grid points $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$, respectively. Here, MFGS dynamically selects the resolution of the snapping grid for each feature point according to the strategy whereby fuzzy points with larger fuzziness ($\tilde{\boldsymbol{c}}$ and $\tilde{\boldsymbol{d}}$) are snapped to a lower-resolution grid and fuzzy points with smaller fuzziness ($\tilde{\boldsymbol{a}}$ and $\tilde{\boldsymbol{b}}$) are snapped to a higher-resolution grid.

If the objects are simply snapped to a single-resolution grid without using MFGS, the alignment often does not match the user's intention, as shown in Fig. 2. That is, if we use the low-resolution grid shown in Fig. 2(a), the fine parts of the drawing such as line $\boldsymbol{ab}$ can degenerate into a point. However, if we use the high-resolution grid shown in Fig. 2(b), the coarse parts of the drawing such as line $\boldsymbol{cd}$ are often subtly shifted. In contrast, MFGS dynamically selects the snapping resolution according to the fuzziness of each fuzzy feature point, resulting in appropriate snapping depending on the roughness of the drawing manner at each portion. As a result, the trapezoid shown in Fig. 1(d) was successfully obtained from the drawn stroke shown in Fig. 1(a).

2.2. Grid setting of MFGS

With MFGS, the user sets a multi-resolution fuzzy grid (MFG) of multiplicity n by combining n layers of fuzzy grids G_i ($i = 1, 2, \ldots, n$). Here, each G_i is a fuzzy grid in which fuzzy points are arranged in the pattern shown in Fig. 4. Assuming an arbitrary origin, these grids are represented by a pair of properties (S_{G_i}, r_{G_i}), where S_{G_i} is the grid spacing and r_{G_i} is the fuzziness of the grid points. That is, the MFG is set with the following parameters:

- Multiplicity: n
- Properties of G_1: $(S_{G_1}, r_{\boldsymbol{G_1}})$
- Properties of G_2: $(S_{G_2}, r_{\boldsymbol{G_2}})$

  $\vdots$
- Properties of G_n: $(S_{G_n}, r_{\boldsymbol{G_n}})$

Provided that G_i ($i = 1, 2, \ldots, n$) are arranged in descending order of resolution with regard to grid index i, we always set the parameters such that $S_{G_i} < S_{G_{i+1}}$ ($i = 1, 2, \ldots, n-1$) and $r_{G_i} < r_{G_{i+1}}$ ($i = 1, 2, \ldots, n-1$).

2.3. MFGS algorithms

The following describes the MFGS algorithms with the example shown in Fig. 5, in which a fuzzy feature point $\tilde{\boldsymbol{c}}$ is to be snapped to an MFG of multiplicity 3, which is composed of the three fuzzy grids G_i ($i = 1, 2, 3$).

(1) Snapping Candidate Generation: For each i ($i = 1, 2, \ldots, n$), the system selects one grid point among those of G_i that are closest to the feature point position $\boldsymbol{c}$ and names it $\boldsymbol{g}_i$ ($i = 1, 2, \ldots, n$). The system then generates the snapping candidates $\tilde{\boldsymbol{g}}_i = \langle \boldsymbol{g}_i, r_{\boldsymbol{g}_i} \rangle = \langle \boldsymbol{g}_i, r_{G_i} \rangle$ ($i = 1, 2, \ldots, n$).

(2) Necessity Evaluation: For each snapping candidate $\tilde{\boldsymbol{g}}_i$, the system evaluates the necessity $N^{\tilde{\boldsymbol{g}}_i}$ on the basis of the necessity measure in fuzzy theory [17,18], where $N^{\tilde{\boldsymbol{g}}_i}$ represents the necessity of fuzzy proposition "$\tilde{\boldsymbol{g}}_i$ *is in* $\tilde{\boldsymbol{c}}$", and is calculated as

$$N^{\tilde{\boldsymbol{g}}_i} = \left(\frac{r_{\boldsymbol{c}} - \|\boldsymbol{g}_i - \boldsymbol{c}\|}{r_{\boldsymbol{c}} + r_{\boldsymbol{g}_i}}\right) \vee 0. \tag{2}$$

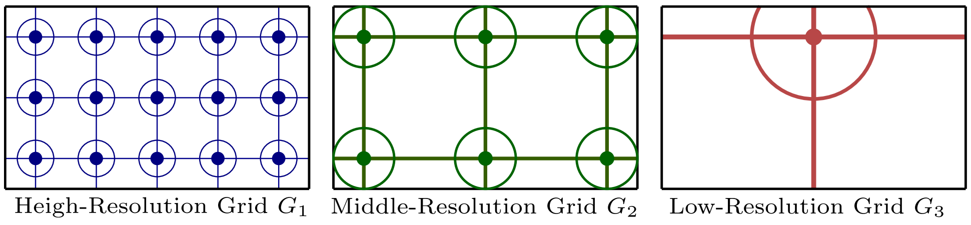

Heigh-Resolution Grid G_1
Middle-Resolution Grid G_2
Low-Resolution Grid G_3

Fig. 4. Fuzzy grids that compose a multi-resolution fuzzy grid (MFG) of multiplicity 3.

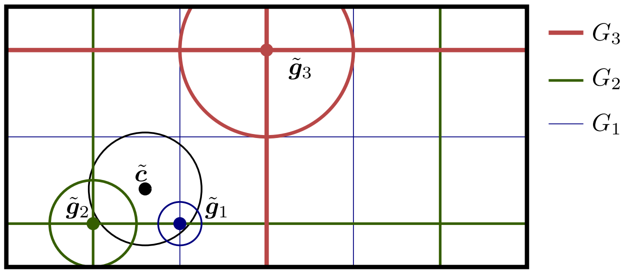


Fig. 5. Fuzzy feature point $\tilde{\boldsymbol{c}}$ on an MFG of multiplicity 3 and snapping candidates $\tilde{\boldsymbol{g}}_i$ $(i = 1, 2, 3)$.

Table 1
Fuzzy rules of MFGS [14].

$\mu(\tilde{\boldsymbol{g}}_3)$	$=$	$N^{\tilde{\boldsymbol{g}}_3}$				
$\mu(\tilde{\boldsymbol{g}}_2)$	$=$	$(1 - N^{\tilde{\boldsymbol{g}}_3})$	$\wedge$	$N^{\tilde{\boldsymbol{g}}_2}$		
$\mu(\tilde{\boldsymbol{g}}_1)$	$=$	$(1 - N^{\tilde{\boldsymbol{g}}_3})$	$\wedge$	$(1 - N^{\tilde{\boldsymbol{g}}_2})$	$\wedge$	$N^{\tilde{\boldsymbol{g}}_1}$
$\mu(\tilde{\boldsymbol{c}})$	$=$	$(1 - N^{\tilde{\boldsymbol{g}}_3})$	$\wedge$	$(1 - N^{\tilde{\boldsymbol{g}}_2})$	$\wedge$	$(1 - N^{\tilde{\boldsymbol{g}}_1})$

(3) Grading: The system calculates the grade $\mu(\tilde{\boldsymbol{g}}_i)$ for each $\tilde{\boldsymbol{g}}_i$ and the grade $\mu(\tilde{\boldsymbol{c}})$ for $\tilde{\boldsymbol{c}}$ by applying the fuzzy logical operations described in Table 1. In this table, note that the symbol $\wedge$ denotes the min operator and is translated as a logical *and*, whereas $(1 - N^{\tilde{\boldsymbol{g}}_i})$ denotes the negation of $N^{\tilde{\boldsymbol{g}}_i}$.

(4) Snapping: The system determines the snapping candidate with the highest grade among $\tilde{\boldsymbol{g}}_i$ $(i = 1, 2, \ldots, n)$, and identifies it as $\tilde{\boldsymbol{g}}_k$. The fuzzy feature point $\tilde{\boldsymbol{c}}$ is then snapped to grid point $\boldsymbol{g}_k$. However, if $\mu(\tilde{\boldsymbol{c}}) > \mu(\tilde{\boldsymbol{g}}_k)$, the system snaps $\tilde{\boldsymbol{c}}$ to its own position $\boldsymbol{c}$, which means that so-called "no-snapping" is selected. For the example shown in Fig. 5, $\tilde{\boldsymbol{g}}_1$ has the highest grade, and so $\tilde{\boldsymbol{c}}$ is snapped to $\boldsymbol{g}_1$.

As a whole, the fuzzy logical operations given in Table 1 function so as to snap the fuzzy feature point to the grid with the lowest resolution as long as the necessity is sufficiently high. Thus, the dynamic snapping of MFGS, whereby fuzzy feature points from coarser drawing are snapped to a grid with lower resolution and feature points from finer drawing are snapped to a grid with higher resolution, is realized. Table 1 shows that MFGS requires $\frac{n(n+1)}{2} + n$ references to necessities $N^{\tilde{\boldsymbol{g}}_i}$ to snap each feature point.

2.4. Problems with MFGS

With an appropriate MFG setting that suits the range of object sizes to be drawn, users can avoid the troublesome operation of switching the grid resolution while drawing. However, inappropriate MFG settings can force users to re-set the parameters controlling the multiplicity and properties of each fuzzy grid. In MFGS, these operations are more onerous than in ordinary single-resolution grid snapping.

Indeed, with sketch-based interfaces, it is not always easy to determine the relative relationships between the grid resolutions and the size distributions of the objects to be drawn in advance. One cause of this difficulty is the fact that the drawn objects may vary greatly in size. When creatively drawing objects by sketch input, users often proceed by changing the size of their drawing strokes over a wide dynamic range to cover small and large objects. In addition, users can input several differently sized geometric objects in a single drawing stroke, such as the four lines we saw in Fig. 1. Therefore, it is difficult to predetermine the size distribution of the objects before the drawing strokes have been made. A second cause of difficulty is the fact that screen sizes and resolutions vary considerably depending on the individual pen-input device.[2] For this reason, even if an appropriate grid resolution is set on a specific device, the physical grid resolution may change significantly on a different device. Hence, it is not possible for users to determine the properties of the grid resolution in pixels with respect to the physical sizes of the drawn objects without specifying the pen-input device.[3] For example, suppose that a user sets a 50 pixel grid spacing and draws a trapezoid of approximately 200 mm in width, as shown in Fig. 6. When using a 52-inch electronic blackboard, the setting of the grid resolution is appropriate, because the physical grid spacing becomes 51.6 mm. In contrast, when using a 13.3-inch LCD tablet, the same grid resolution setting is too high, because the physical grid spacing becomes 5.8 mm.

The above explanation clarifies that, in realistic situations where users sketch creatively using a variety of pen-input devices, it is not easy to set appropriate MFG parameters in advance. Although MFGS is effective in a limited range of situations where the size distribution of the objects to be drawn and pen-input device to be used are defined in advance, it suffers from the frequent need to re-set the MFG, which disturbs the flow of essential drawing tasks in practical use.

To deal with this problem, a user may increase multiplicity n so that all the possible drawing situations are covered. However, as n is increased, the number of references to necessities $N^{\tilde{\boldsymbol{g}}_i}$ increases as $O(n^2)$, as mentioned in 2.3, and the computational efficiency is greatly reduced.

3. Proposal of infinite-resolution fuzzy grid snapping

To solve this problem of MFGS, we now describe IFGS, which extends the multiplicity of possible grid resolutions to an infinite range.

[2] Nowadays, CAD systems are available on a variety of devices, from smartphones and tablets to electric blackboards, and users switch between these devices freely in accordance with their use.

[3] Users may set an appropriate resolution as a physical distance if the interface (as an application program) can obtain the physical resolutions of the pen-input devices from the operating system in a standard way. However, this is not always the case in ordinary operating systems.

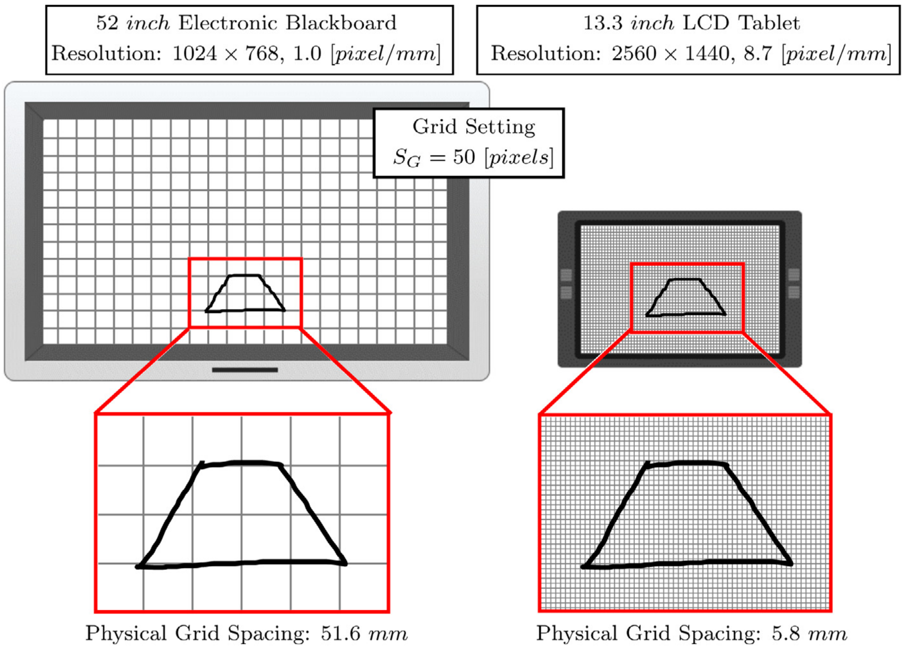

52 inch Electronic Blackboard
Resolution: 1024 × 768, 1.0 [pixel/mm]
13.3 inch LCD Tablet
Resolution: 2560 × 1440, 8.7 [pixel/mm]
Grid Setting
$S_G = 50$ [pixels]
Physical Grid Spacing: 51.6 mm
Physical Grid Spacing: 5.8 mm

Fig. 6. Difference in physical grid spacing between two different pen-input devices.

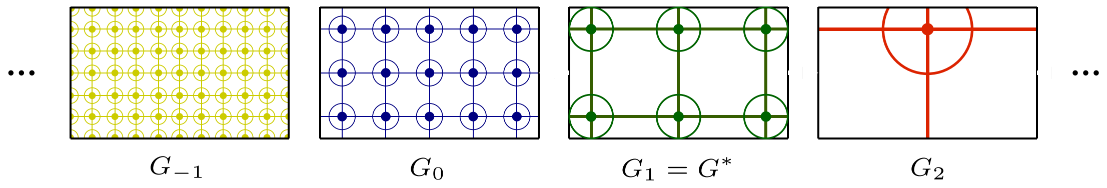


Fig. 7. Fuzzy grids that compose an IFG with magnification 2.

3.1. *Grid setting of IFGS*

For an arbitrary origin, IFGS sets an infinite-resolution fuzzy grid (IFG) using the following two sets of parameters:

- Properties of Base Grid G^*: (S_{G^*}, r_{G^*})
- Magnification of the Grid Resolution: f

With these parameters, IFGS can generate the properties of an infinite number of fuzzy grids G_i ($i \in \mathbb{Z}$, where $\mathbb{Z}$ denotes the set of integers) as

$$(S_{G_i}, r_{G_i}) = (S_{G^*} \times f^{i-1}, r_{G^*} \times f^{i-1}). \tag{3}$$

Provided that G_i ($i \in \mathbb{Z}$) are in descending order of resolution with regard to grid index i, we always set f to satisfy $f > 1$. Fig. 7 illustrates an example of an IFG in which $f = 2$.

3.2. *Algorithms of IFGS*

As described in Table 2, an infinite number of fuzzy rules are applied in IFGS.[4] This is an extension of the finite number of fuzzy rules for MFGS given in Table 1. As in MFGS, we allow IFGS to determine the snapping candidate with the highest grade among $\tilde{\mathbf{g}}_i$ ($i \in \mathbb{Z}$) according to Table 2, and identify this as $\tilde{\mathbf{g}}_k$. Fuzzy feature point $\tilde{\mathbf{c}}$ is then snapped to grid point $\mathbf{g}_k$.

Now, the problem is how to find the snapping candidate with the highest grade from an infinite number of rules, each of which has an infinite number of terms. Considering that $\|\mathbf{g}_i - \mathbf{c}\| \leq \|\mathbf{g}_j - \mathbf{c}\|$ and $r_{\mathbf{g}_i} < r_{\mathbf{g}_j}$ when $i < j$, we know that the necessity $N^{\tilde{\mathbf{g}}_i}$ in (2) is monotonically non-increasing with respect to i. Hence, its negation $1 - N^{\tilde{\mathbf{g}}_i}$ is monotonically non-decreasing. As a result, each grade $\mu(\tilde{\mathbf{g}}_i)$ in Table 2, although expressed by an infinite number of terms, can be simply calculated as

$$\mu(\tilde{\mathbf{g}}_i) = (1 - N^{\tilde{\mathbf{g}}_{i+1}}) \wedge N^{\tilde{\mathbf{g}}_i}. \tag{4}$$

Furthermore, although the number of rules in Table 2 is infinite, we know that it is possible to determine the snapping destination $\mathbf{g}_k$ in a finite number of procedures, because $\mu(\tilde{\mathbf{g}}_i)$ given by (4) shifts in a unimodal manner, as shown in Fig. 8. The following illustrates how the IFGS procedure searches for the snapping destination $\mathbf{g}_k$ using the example shown in Fig. 9.

(1) Search for Higher-Limit Grid G_h: Starting from the base grid G^* ($= G_1$), the system searches for grid G_h such that $\mu(\tilde{\mathbf{g}}_h) \geq \mu(\tilde{\mathbf{g}}_{h-1})$ is satisfied while increasing the resolution according to the following procedure. In the case of Fig. 9, G_h is obtained as G_{-1}.

Table 2
Fuzzy rules of IFGS.

$$\begin{array}{lcl}
\vdots & & \vdots \\
\mu(\tilde{\mathbf{g}}_{i+1}) & = & (1 - N^{\tilde{\mathbf{g}}_\infty}) \wedge \cdots \wedge (1 - N^{\tilde{\mathbf{g}}_{i+2}}) \wedge N^{\tilde{\mathbf{g}}_{i+1}} \\
\mu(\tilde{\mathbf{g}}_{i}) & = & (1 - N^{\tilde{\mathbf{g}}_\infty}) \wedge \cdots \wedge (1 - N^{\tilde{\mathbf{g}}_{i+1}}) \wedge N^{\tilde{\mathbf{g}}_{i}} \\
\mu(\tilde{\mathbf{g}}_{i-1}) & = & (1 - N^{\tilde{\mathbf{g}}_\infty}) \wedge \cdots \wedge (1 - N^{\tilde{\mathbf{g}}_{i}}) \wedge N^{\tilde{\mathbf{g}}_{i-1}} \\
\mu(\tilde{\mathbf{g}}_{i-2}) & = & (1 - N^{\tilde{\mathbf{g}}_\infty}) \wedge \cdots \wedge (1 - N^{\tilde{\mathbf{g}}_{i-1}}) \wedge N^{\tilde{\mathbf{g}}_{i-2}} \\
\vdots & & \vdots
\end{array}$$

[4] Note that the no-snapping grade in MFGS, $\mu(\tilde{\mathbf{c}})$, is meaningless in IFGS, which can utilize grids of infinitely high resolution.

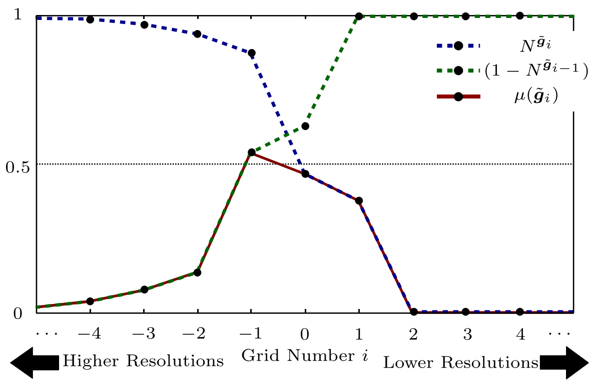


Fig. 8. Example of transitions of $N^{\tilde{g}_i}$, $(1-N^{\tilde{g}_{i+1}})$, and $\mu(\tilde{\boldsymbol{g}}_i)$ in IFGS.

(1-1) Initialization: The system sets $i := 1$.

(1-2) Necessity Evaluation: The system selects the grid point in G_i that is nearest to feature point position $\boldsymbol{c}$ and names it $\boldsymbol{g}_i$. The system then generates a snapping candidate $\tilde{\boldsymbol{g}}_i = \langle \boldsymbol{g}_i, r_{\boldsymbol{g}_i}\rangle = \langle \boldsymbol{g}_i, r_{G_i}\rangle$ and evaluates necessity $N^{\tilde{\boldsymbol{g}}_i}$ by (2).

(1-3) Detection of the Higher Limit: If $N^{\tilde{\boldsymbol{g}}_i} \geq 0.5$, the system determines that $\mu(\tilde{\boldsymbol{g}}_i) \geq \mu(\tilde{\boldsymbol{g}}_{i-1})$ is satisfied based on property (A.1) derived in Appendix A, and proceeds to step (2) with the higher-limit grid setting $h := i$. Otherwise, the system returns to step (1-2) after setting $i := i - 1$.

(2) Search for Snapping Grid G_k: Starting from grid G_h, the system searches for grid G_k with the maximum grade $\mu(\tilde{\boldsymbol{g}}_k)$ while decreasing the resolution according to the following procedure. In the case of Fig. 9, G_k is determined to be G_{-1}.

(2-1) Initialization: The system sets $i := h + 1$.

(2-2) Necessity Evaluation: The system generates $\tilde{\boldsymbol{g}}_i$ in G_i and evaluates $N^{\tilde{\boldsymbol{g}}_i}$ in the same manner as in step (1-2).

(2-3) Detection of the Maximum Grade: If $N^{\tilde{\boldsymbol{g}}_i} < 0.5$, $\mu(\tilde{\boldsymbol{g}}_i) \leq \mu(\tilde{\boldsymbol{g}}_{i-1})$ is satisfied based on property (A.2) in Appendix A, and the system proceeds to step (3) with the snapping grid setting $k := i - 1$. Otherwise, the system returns to step (2-2) after setting $i := i + 1$.

(3) Snapping: The system snaps the fuzzy feature point $\tilde{\boldsymbol{c}}$ to grid point $\boldsymbol{g}_k$, which is the position vector of snapping candidate $\tilde{\boldsymbol{g}}_k$ generated from snapping grid G_k.

With the above algorithm, when a fuzzy feature point $\tilde{\boldsymbol{c}}$ is snapped to grid G_k, the number of references to necessities $N^{\tilde{\boldsymbol{g}}_i}$, denoted as C_{G_k}, is $C^{(1)}_{G_k} + C^{(2)}_{G_k}$, where $C^{(1)}_{G_k}$ is given by

$$C^{(1)}_{G_k} = \begin{cases} 2-k & (k<1) \\ 1 & (k \geq 1) \end{cases}$$

and $C^{(2)}_{G_k}$ is given by

$$C^{(2)}_{G_k} = \begin{cases} 1 & (k<1) \\ k & (k \geq 1). \end{cases}$$

$C^{(1)}_{G_k}$ is the number of references with step (1). $C^{(2)}_{G_k}$ is the number of references with step (2). As a result, C_{G_k} is given by

$$C_{G_k} = |k-1| + 2. \tag{5}$$

3.3. Snapping characteristics of IFGS

To illustrate the snapping characteristics of IFGS, let us consider a fuzzy feature point $\tilde{\boldsymbol{c}} = \langle (c_x, 0), r_{\boldsymbol{c}}\rangle$ on the IFG $(S_{G^*}, r_{G^*}) = (25.00, 6.25)$ pixels and $f = 2$, as shown in Fig. 10. We plot the

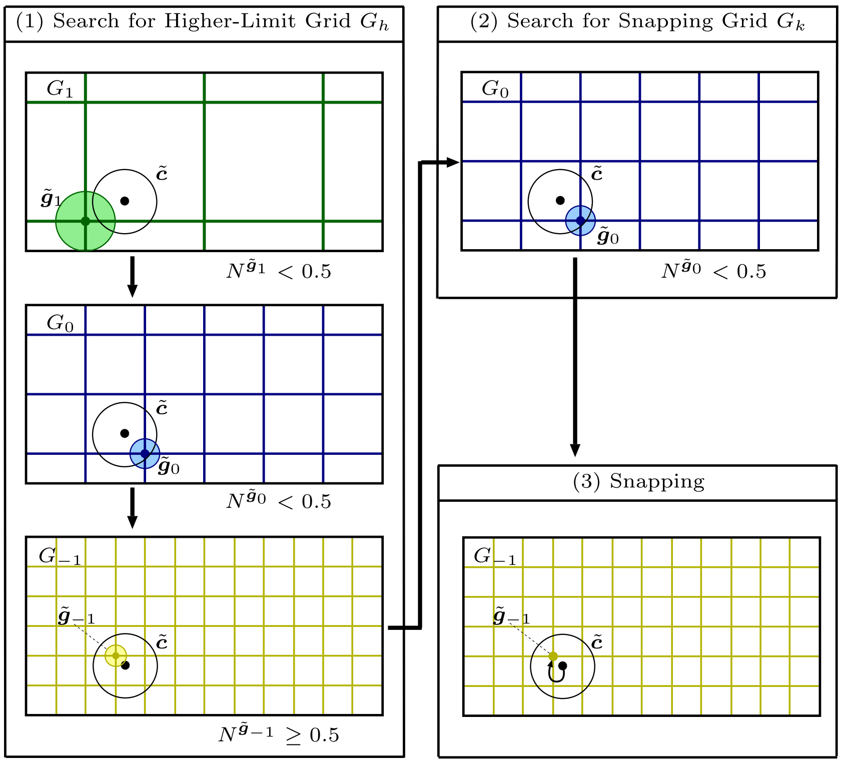


Fig. 9. Process of IFGS.

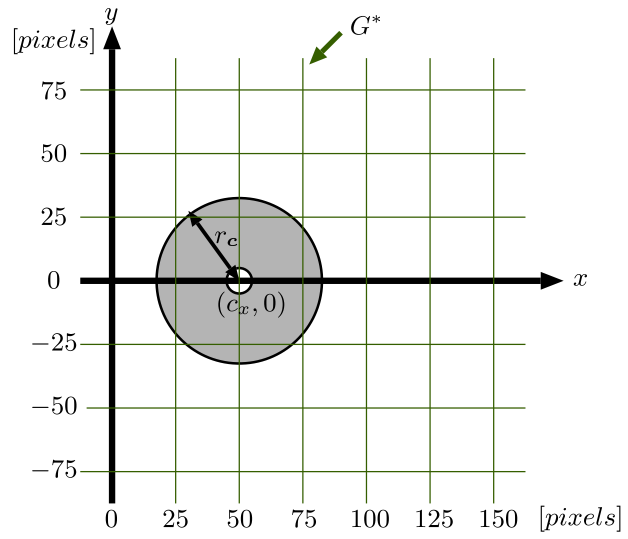


Fig. 10. A fuzzy point $\tilde{\boldsymbol{c}}$ on an IFG.

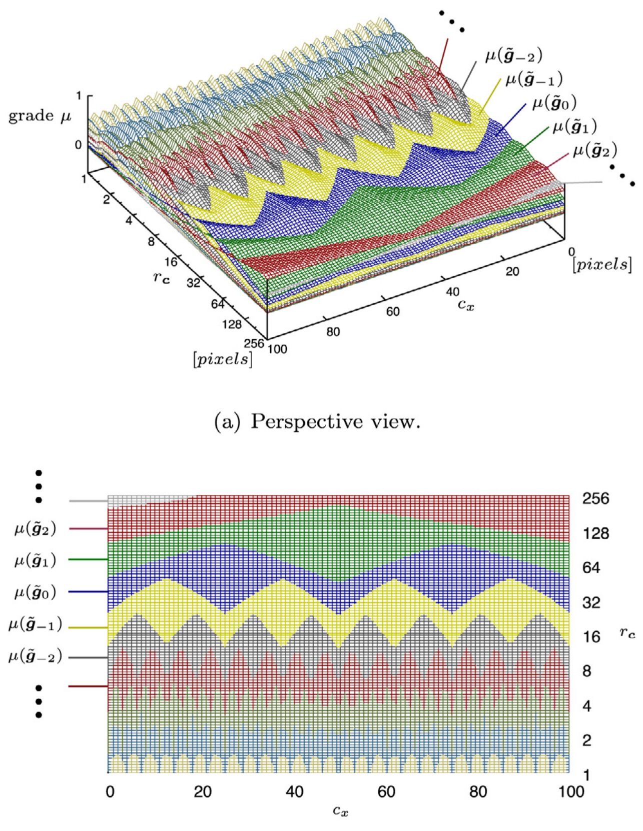


Fig. 11. Grades evaluated by IFGS for the fuzzy point $\tilde{\boldsymbol{c}}$ shown in Fig. 10.

grades $\mu(\tilde{\boldsymbol{g}}_i)$ evaluated by IFGS while changing $\tilde{\boldsymbol{c}}$ in the range $c_x \in [0, 100]$ and $r_c \in [1, 256]$. The result is shown in Fig. 11. Noting that IFGS snaps $\tilde{\boldsymbol{c}}$ to the grid with the maximum grade, Fig. 11 shows that IFGS snaps fuzzy points with larger fuzziness to grids of lower resolution, and vice versa. These characteristics of IFGS can be considered as an extension of those of MFGS, which are illustrated in [14, Fig. 6].

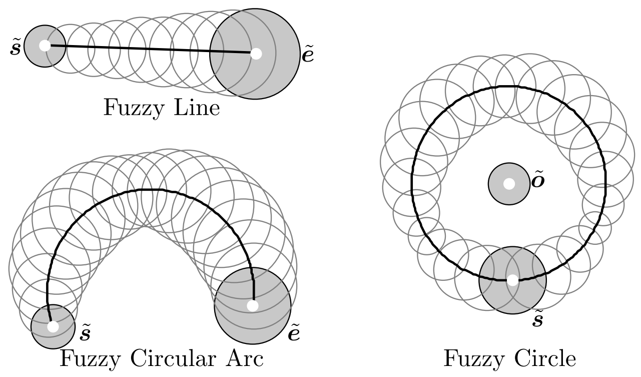


Fig. 12. Fuzzy feature points of fuzzy geometric objects.

4. Experiments comparing MFGS and IFGS

To demonstrate how IFGS overcomes the problems of MFGS, we performed some experiments by drawing geometric objects with a sketch-based CAD system, and compared the results using MFGS and IFGS. For this purpose, we added an object-snapping function based on IFGS to a sketch-based CAD system that uses MFGS [15]. Thus, we constructed a sketch-based CAD system that can switch between MFGS and IFGS. In this experiment, we focused on geometric figures that consist of a combination of lines, circles, and circular arcs. We define the feature points as the start point $\tilde{\boldsymbol{s}}$ and the end point $\tilde{\boldsymbol{e}}$ for fuzzy lines and fuzzy circular arcs, and as the start point $\tilde{\boldsymbol{s}}$ and the center $\tilde{\boldsymbol{o}}$ for fuzzy circles, as shown in Fig. 12. The pen-input device was an LCD tablet with a resolution of 3.6 pixels/mm, which was connected to a computer with a 3.2 GHz Intel Core i5-3470 processor and 4 GB main memory.

4.1. Performance comparison: Drawing time

To compare the performance of MFGS and IFGS, we conducted experiments to measure the time required to draw a given target figure.[5] As the target figure, we set up the compass shape shown in Fig. 13(a), which consists of twelve geometric objects, i.e., different sizes of lines, circles, and circular arcs. As shown in Fig. 14, this drawing requires a high-resolution grid with a spacing of 20 pixels or less so that the pair of small circular arcs that form the loop at the top can be aligned with an offset spacing of 20 pixels. Thus, we prepared the following settings for MFG and IFG, where the origins are set at the center of the display.

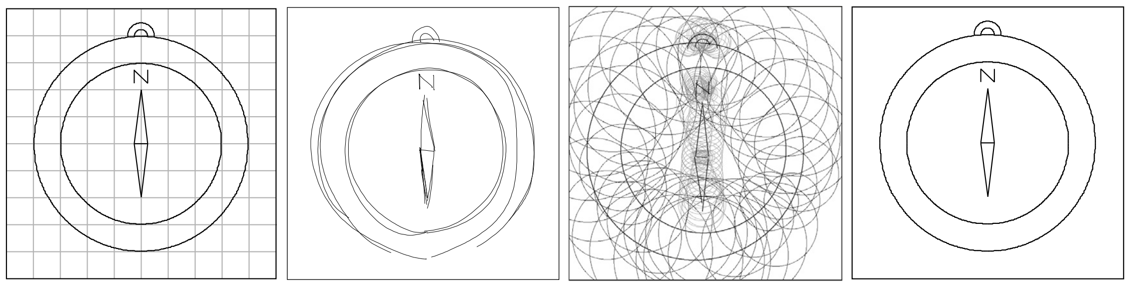
MFG-A Low-Resolution MFG Setting

- $n = 3$
- $(S_{G_1}, r_{G_1}) = (20.0, 5.0)$ pixels
- $(S_{G_2}, r_{G_2}) = (40.0, 10.0)$ pixels
- $(S_{G_3}, r_{G_3}) = (80.0, 20.0)$ pixels

MFG-B High-Resolution MFG Setting

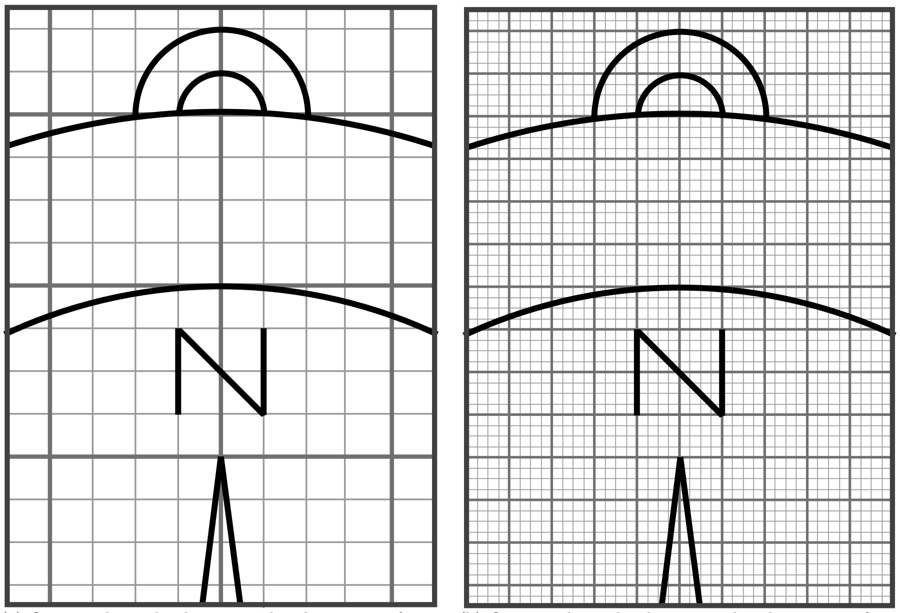
- $n = 3$
- $(S_{G_1}, r_{G_1}) = (5.0, 1.25)$ pixels
- $(S_{G_2}, r_{G_2}) = (10.0, 2.5)$ pixels
- $(S_{G_3}, r_{G_3}) = (20.0, 5.0)$ pixels

[5] Note that the sketch-based CAD system itself could be regarded as a real-time interface, because its reaction time for each stroke (the duration from the time when one stroke is drawn to the time when the snapped geometric objects are displayed) was always less than 0.5 s throughout the experiments.



(a) Target figure on a grid with a spacing of 80 pixels. (b) Example of strokes drawn by a user. (c) Fuzzy geometric objects identified from the strokes. (d) Geometric objects snapped from the fuzzy geometric objects.

Fig. 13. Target compass figure and example of the free-hand input used to complete the target.



(a) On a two-layered indicative grid with spacings of 20.0 pixels and 80.0 pixels. (b) On a two-layered indicative grid with spacings of 5.0 pixels and 20.0 pixels.

Fig. 14. Target compass figure with the indicative grids indicated in the images.

IFG-A Low-Resolution IFG Setting

- $(S_{G^*}, r_{G^*}) = (20.0, 5.0)$ pixels
- $f = 2$

IFG-B High-Resolution IFG Setting

- $(S_{G^*}, r_{G^*}) = (5.0, 1.25)$ pixels
- $f = 2$

The drawing time was measured as follows.

(1) The sketch-based CAD system displays the target figure along with an indicative grid on the LCD tablet. The two-layered grid shown in Fig. 14(a) is displayed for MFG-A or IFG-A, whereas that shown in Fig. 14(b) is given for MFG-B or IFG-B.

(2) When the user begins drawing, the system records the start time of the first drawn stroke.

(3) The user proceeds to complete the target figure by drawing strokes repeatedly (as shown in Fig. 13(b)[6]) while allowing the system to identify geometric objects (as shown in Fig. 13(c)) and snap them to the grid.

(4) A supervisor observing the task stops the user from drawing when the target figure has been completed, as shown in Fig. 13(d).

(5) The system obtains the drawing time by calculating the duration from the start time of the first stroke to the end time of the last stroke.

For each user, the drawing time was measured repeatedly while changing the grid settings. The overall procedure was as follows.

(1) The user practiced completing the task of drawing the target figure twice with each of the four grid settings.

6 Several strokes are overlapped with each other. This is because the sketch-based CAD system in [15] allows users to draw overlapping strokes to modify identified results with the algorithms proposed in [7].

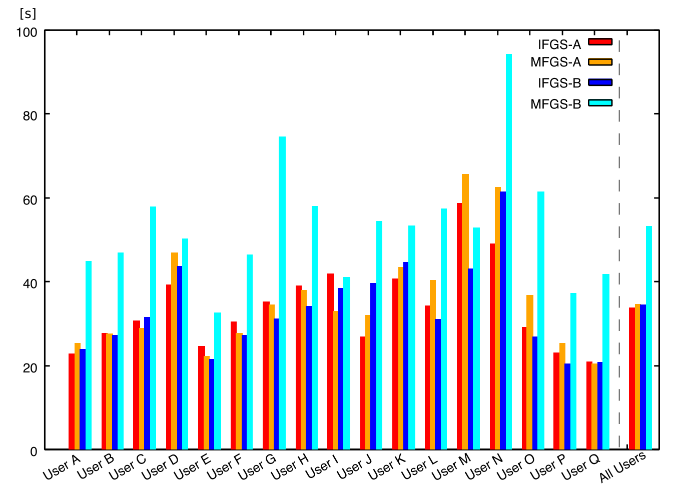


Fig. 15. Average time required to complete the target figure shown in Fig. 13.

(2) The drawing time was measured 10 times and the average drawing times were calculated for IFG-A, MFG-A, IFG-B, and MFG-B.

It took around 40 min for each user to complete the above drawing tasks.

Fig. 15 shows the average drawing times obtained from seventeen users, all of whom were familiar with the sketch-based CAD system [15]. Although there are individual differences, the overall results from all users clearly show that the average drawing time for MFG-B was significantly longer than for the other three grid settings. Thus, MFG-B is an inappropriate setting for the target figure. In fact, in the drawing tasks with MFG-B, we often observed cases in which users were forced to draw many strokes repeatedly to successfully align the big circle 640 pixels in diameter, that is, the body of the compass, because the lowest grid resolution of MFG-B (20 pixels) was still too high for a circle of this size. In contrast, for MFG-A, the lowest grid resolution, with a grid spacing of 80 pixels, was suitable for this circle, and the highest grid resolution with a grid spacing of 20 pixels was suitable for the pair of small circular arcs that form the loop at the top of the compass. Hence, users could complete the task faster with fewer drawing strokes. With respect to IFG-A and IFG-B, because their infinite-resolution fuzzy grids include the multi-resolution fuzzy grid of MFG-A, users were able to complete the task with either grid[7] as fast as with MFG-A.

These results illustrate the problem with MFGS, as discussed in Section 2.4, whereby it performs poorly when the MFG setting is not suited to the range of sizes of the objects to be drawn. In contrast, IFGS works well regardless of the IFG setting, thus overcoming this problem.

4.2. Comparison of characteristics with a complex target figure

Next, we compared the snapping characteristics of MFG-A and IFG-A, both of which worked well in the previous experiment with a simple target figure, in a situation involving a more complex figure that consists of approximately 300 geometric objects. Fig. 16 shows the freehand strokes drawn by the user, as well as the fuzzy geometric objects identified from these strokes. The strokes range from a large rainbow shape with a diameter of about 1500 pixels to a fine fence with column intervals of 10 pixels. Thus, the dynamic range of the object sizes in this case is much wider than for the target figure in Fig. 13.

Fig. 17(a) shows the result obtained from the input of Fig. 16 with MFG-A. In this case, we can see that the large figures that form the rainbow were not properly aligned. This is due to subtle shifts that occurred because the lowest grid resolution, with a grid spacing of 80 pixels, was still too fine for such large objects. We can also see that the fine figures in parts of the fence are not aligned. This is because the highest grid resolution with a grid spacing of 20 pixels was too coarse to align such fine objects. In this situation, to make MFGS snap all the objects properly, the user must manually increase the multiplicity n and set the properties of additional fuzzy grids.

Fig. 17(b) shows the output obtained with IFG-A.[8] We can see that all objects have been properly aligned, regardless of the dynamic range of sizes. This is because IFGS can utilize an infinite number of resolutions. From the results, we can see that users of IFGS can avoid the tedious manual operations of re-setting the grid, even if the sizes of the objects vary widely.

In the experiments, we set the grid origin to the center of the display, but we note that changing this origin will not give the same results for both IFGS and MFGS. For example, Fig. 18 shows the different results given by IFGS with IFG-A when the origin is changed. The results show that the snapping of the large rainbow is affected by the change in origin while the other smaller objects are kept at the same snapping locations. This is because the layouts of the grids whose resolutions are lower than the indicative grid change because of the change in origin. This means that the users should be conscious of the layouts of the grids at these lower resolutions. A remaining challenge in IFGS is how to enable users to perceive the layouts of these grids when they draw figures. In our experience, one of the simplest solutions but effective one is to set the origin at the center of the display.

4.3. Comparison of computational efficiency

It is possible for MFGS to achieve the same results as IFGS, i.e., the results shown in Fig. 17(b), by increasing the multiplicity n. However, IFGS has lower computational costs than MFGS. To demonstrate this, we compare the computational efficiency of IFGS and MFGS with respect to the number of references to necessities $N^{\tilde{g}_i}$.

The total number of fuzzy feature points was 588 when we obtained the results shown in Fig. 17(b). The frequencies of the feature points snapped to the grids G_k of IFG-A, denoted as F_{G_k}, are shown in Fig. 19. To achieve the same results using MFGS, the following minimum MFG setting with a multiplicity 10 is required.

MFG-C Minimum MFG Setting to Obtain Fig. 17(b)

- $n = 10$
- $(S_{G_i}, r_{G_i}) = (20 \times 2^{i-6}, 5 \times 2^{i-6})$ pixels ($i = 1, \ldots, 10$)

The computational cost of MFGS with MFG-C, denoted as $C_{\text{MFG-C}}$, is $588 \times \left(\frac{n(n+1)}{2} + n\right) = 38{,}220$ according to Section 2.3. In contrast, the computational cost of IFGS with IFG-A, denoted as $C_{\text{IFG-A}}$, is $\sum_{k\in\mathbb{Z}} F_{G_k} C_{G_k} = \sum_{k\in\mathbb{Z}} F_{G_k}(|k-1|+2) = 2,041$ according to (5). Thus, the relative computational cost of IFG-A with respect to MFG-C is $\frac{C_{\text{IFG-A}}}{C_{\text{MFG-C}}} = \frac{2{,}041}{38{,}220} \fallingdotseq 0.053$. For IFGS, any of the following IFG settings gives snapping characteristics that are equivalent to those of IFG-A.

7 Note that, although IFG-A and IFG-B have different IFG settings and indicative grids, their snapping characteristics are equivalent, and they produced almost equal average drawing times.

8 Note that IFG-B gives the same output as IFG-A because of their substantial equivalence.

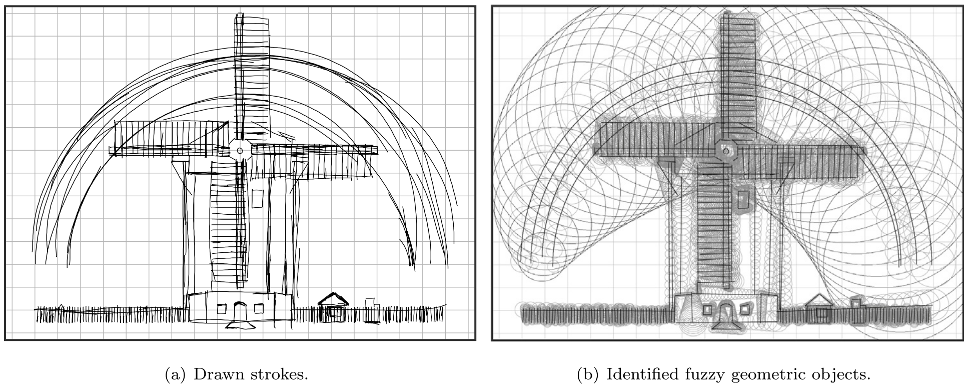

(a) Drawn strokes. (b) Identified fuzzy geometric objects.

Fig. 16. Freehand input strokes to complete a complex figure and fuzzy geometric objects identified from the strokes, shown on a grid with a spacing of 80 pixels.

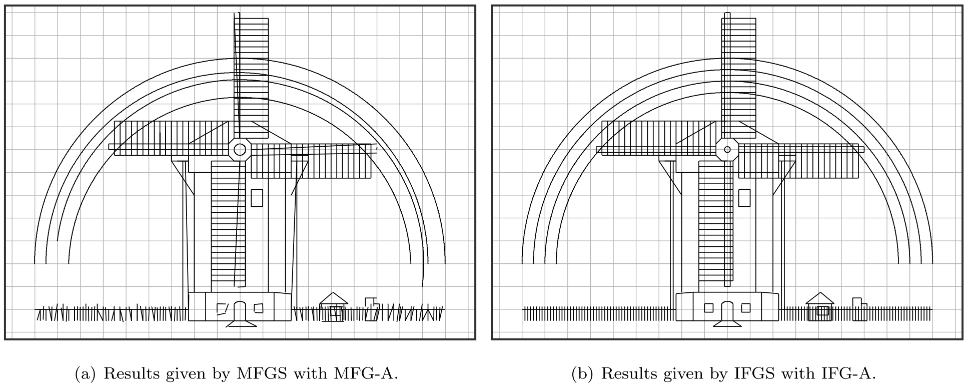
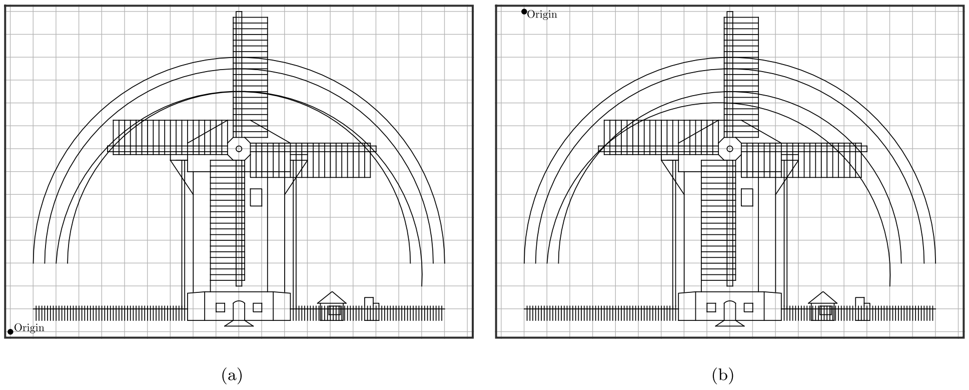

(a) Results given by MFGS with MFG-A. (b) Results given by IFGS with IFG-A.

Fig. 17. Output results snapped from the input shown in Fig. 16.



(a) (b)

Fig. 18. Different results given by IFGS with IFG-A for different origins.

IFG-(i) ($i \in \mathbb{Z}$) IFG Settings Equivalent to IFG-A

- $(S_{G^*}, r_{G^*}) = (20 \times 2^{i-1}, 5 \times 2^{i-1})$ pixels
- $f = 2$

However, their computational cost, denoted as $C_{\text{IFG-}(i)}$, varies as $\sum_{k\in\mathbb{Z}} F_{G_k}(|k-i|+2)$ according to i. We show the relative computational cost of IFG-(i) with respect to MFG-C, that is, $\frac{C_{\text{IFG-}(i)}}{C_{MFG-C}}$, in Fig. 20. Here, it is necessary to increase the multiplicity n to meet various device resolutions. For example, to support both a high resolution tablet computer whose resolution is four times that of the LCD tablet used in this experiment and a low resolution electronic blackboard whose resolution is one fourth that of the LCD tablet, the following MFG setting with a multiplicity of 14 is required.

MFG-D MFG Setting to Achieve Fig. 17(b) for Various Device Resolutions

- $n = 14$
- $(S_{G_i}, r_{G_i}) = (20 \times 2^{i-8}, 5 \times 2^{i-8})$ pixels ($i = 1, \ldots, 14$)

We also show the relative computational cost of IFG-(i) with respect to MFG-D, that is $\frac{C_{\text{IFG-}(i)}}{C_{MFG-D}}$, in Fig. 20.

The above results illustrate that IFGS has lower computational costs (i.e., higher computational efficiency) with respect to MFGS unless its base grid deviates significantly from the range of snapping grids that are actually used. They also illustrate that the relative computational cost of IFGS decreases as the multiplicity n of MFGS increases. The results are based on the best situation, in which all of the objects to be drawn are known. In practical situations, where the sizes of the objects to be drawn are unknown,

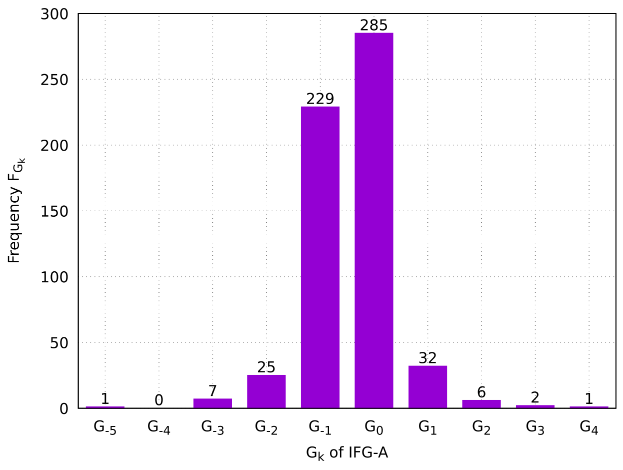


Fig. 19. Frequencies at which G_k were selected as snapping grids.

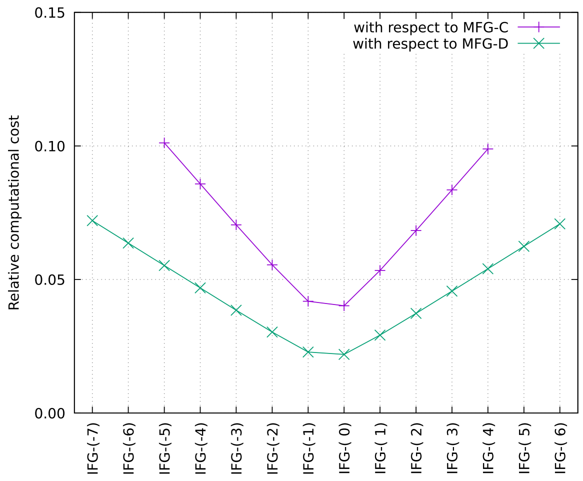


Fig. 20. Computational costs of IFGS with respect to MFGS.

n must be further increased, reducing the relative computational cost of IFGS.

5. Conclusions

To overcome the problematic grid setting requirements in MFGS, we proposed an extension known as IFGS, which increases the number of possible resolutions to infinity. We described the algorithm of IFGS, which can be completed in a finite number of steps. In addition, we added IFGS to a sketch-based CAD system equipped with MFGS, and performed experiments to examine the effectiveness and efficiency of IFGS compared with that of MFGS. The experimental results clearly show that the performance of IFGS remained high level at different grid settings, whereas that of MFGS deteriorated depending on the grid spacing. An experimental comparison of the output characteristics when a user drew a complex figure with a wide range of object sizes showed that IFGS, unlike MFGS, did not require tedious manual operations to modify the grid setting, even when the sizes of the objects varied widely.

For simplicity, the evaluation experiments in this paper have focused on three types of fuzzy geometric objects that have only two fuzzy feature points. In contrast, in [19], we have already used MFGS to snap five types of fuzzy geometric curves (lines, circles, circular arcs, ellipses, and elliptical arcs), which are all the curves identified by FSCI except for closed and open free curves. We intend to conduct a thorough evaluation of IFGS in more practical and complex situations, using all the fuzzy geometric objects listed above, which may have many feature points.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106112.

Appendix A. Properties of $\mu(\tilde{g}_i)$, $\mu(\tilde{g}_{i-1})$, and $N^{\tilde{g}_i}$

Because necessity $N^{\tilde{g}_i}$ is monotonically non-increasing when the G_i are arranged in descending order of resolution with regard to grid index i, we derive the following two properties:

$$N^{\tilde{g}_i} \geq 0.5 \Rightarrow \mu(\tilde{g}_i) \geq \mu(\tilde{g}_{i-1}) \tag{A.1}$$

$$N^{\tilde{g}_i} < 0.5 \Rightarrow \mu(\tilde{g}_i) \leq \mu(\tilde{g}_{i-1}) \tag{A.2}$$

Proposition (A.1) is proved as follows. If $N^{\tilde{g}_i} \geq 0.5$, then $N^{\tilde{g}_i} \geq (1 - N^{\tilde{g}_i})$. Because necessity $N^{\tilde{g}_i}$ is monotonically non-increasing, $N^{\tilde{g}_{i-1}} \geq N^{\tilde{g}_i}$ is established. Thus, when $N^{\tilde{g}_i} \geq (1 - N^{\tilde{g}_i})$, $N^{\tilde{g}_{i-1}} \geq (1 - N^{\tilde{g}_i})$. Hence,

$$\mu(\tilde{g}_{i-1}) = (1 - N^{\tilde{g}_i}) \wedge N^{\tilde{g}_{i-1}} = (1 - N^{\tilde{g}_i}). \tag{A.3}$$

In contrast, because $(1 - N^{\tilde{g}_i})$ is monotonically non-decreasing, $(1-N^{\tilde{g}_i}) \leq (1-N^{\tilde{g}_{i+1}})$ is established. Thus, when $N^{\tilde{g}_i} \geq (1-N^{\tilde{g}_i})$,

$$\begin{aligned}\mu(\tilde{g}_i) &= (1 - N^{\tilde{g}_{i+1}}) \wedge N^{\tilde{g}_i} \\ &\geq (1 - N^{\tilde{g}_i}) \wedge N^{\tilde{g}_i} = (1 - N^{\tilde{g}_i}).\end{aligned} \tag{A.4}$$

From Eqs. (A.3) and (A.4), we obtain

$$\mu(\tilde{g}_i) \geq \mu(\tilde{g}_{i-1}), \tag{A.5}$$

and proposition (A.1) is established. Proposition (A.2) can be proved in a similar manner.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2020.106112.

References

[1] T. Igarashi, S. Matsuoka, S. Kawachiya, H. Tanaka, Interactive beautification: A technique for rapid geometric design, in: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, UIST '97, ACM, New York, NY, USA, 1997, pp. 105–114, http://dx.doi.org/10.1145/263407.263525.

[2] J. Arvo, K. Novins, Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes, in: Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, UIST '00, ACM, New York, NY, USA, 2000, pp. 73–80, http://dx.doi.org/10.1145/354401.354413.

[3] S. Saga, H. Makino, Fuzzy spline interpolation and its application to online freehand curve identification, in: Proceedings 1993 Second IEEE International Conference on Fuzzy Systems, Vol. 2, 1993, pp. 1183–1190, http://dx.doi.org/10.1109/FUZZY.1993.327560.

[4] S. Saga, H. Makino, J. Sasaki, Method for modeling freehand curves-the fuzzy spline interpolation, IEICE Trans. J77-D-II (8) (1994) 1610–1619 (in Japanese).

[5] S. Saga, H. Makino, J. Sasaki, The fuzzy spline curve identifer, IEICE Trans. J77-D-II (8) (1994) 1620–1629 (in Japanese).

[6] C.P. Chen, S. Xie, Freehand drawing system using a fuzzy logic concept, Comput. Aided Des. 28 (2) (1996) 77–89, http://dx.doi.org/10.1016/0010-4485(95)00026-7.

[7] Y. Sato, N. Yasufuku, S. Saga, Sequential fuzzy spline curve generator for drawing interface by sketch, IEICE Trans. J86-D-II (2) (2003) 242–251 (in Japanese).

[8] R. Kawai, A. Nishikawa, S. Saga, A freehand sketch input front-end processor: SKIT, IEICE Trans. J88-D-II (5) (2005) 897–905 (in Japanese).

[9] S.I. Edward, Sketchpad, in: A Man-Machine Graphical Communication System (Ph.D. thesis), Massachusetts Institute of Technology, 1963.
[10] M. Gleicher, A. Witkin, Drawing with constraints, Vis. Comput. 11 (1) (1994) 39–51, http://dx.doi.org/10.1007/BF01900698.
[11] E.A. Bier, M.C. Stone, Snap-dragging, SIGGRAPH Comput. Graph. 20 (4) (1986) 233–240, http://dx.doi.org/10.1145/15886.15912.
[12] T. Masui, Hyper snapping, in: Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments, HCC '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 188–194.
[13] P. Baudisch, E. Cutrell, K. Hinckley, A. Eversole, Snap-and-go: Helping users align objects without the modality of traditional snapping, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05, ACM, New York, NY, USA, 2005, pp. 301–310, http://dx.doi.org/10.1145/1054972.1055014.
[14] Q.U. Khand, S. Dematapitiya, S. Saga, J. Maeda, A multi-resolution grid snapping technique based on fuzzy theory, IPSJ Digit. Courier 3 (2007) 198–206, http://dx.doi.org/10.2197/ipsjdc.3.198.
[15] S. Dematapitiya, M. Kawazoe, A. Nishikawa, M. Sakurai, S. Saga, Snapping of fuzzy objects using the multi-resolution fuzzy grid snapping technique, J. Inform. Process. 17 (2009) 47–58, http://dx.doi.org/10.2197/ipsjjip.17.47.
[16] T. Ohkawa, S. Saga, Refinement of fuzziness generator in freehand curve identifier FSCI, IEICE Trans. J82-D-I (5) (1999) 634–643 (in Japanese).
[17] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems 1 (1978) 3–28.
[18] G.J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
[19] K. Nakajima, N. Hatamoto, T. Ito, T. Shibata, S. Saga, Identification of n-quarter circular arcs and n-quarter elliptic arcs based on freehand curve identifier FSCI, J. Japan Soc. Fuzzy Theory Intell. Inform. 31 (3) (2019) 701–711, http://dx.doi.org/10.3156/jsoft.31.3_701 (in Japanese).

Ten Watanabe received his B.E. and M.E. degrees in computational intelligence from the Muroran Institute of Technology, Hokkaido, Japan, in 2016 and 2018, respectively. His research interests centered on freehand curve identification algorithms for sketch-based CAD systems based on fuzzy logic. He now works at Hokuden Information Technology as a Software Developer.

Tomohito Yoshikawa received his B.E. and M.E. degrees in computational intelligence from the Muroran Institute of Technology, Hokkaido, Japan, in 2014 and 2016, respectively. His research interests centered on intelligent object snapping algorithms for sketch-based CAD systems based on fuzzy logic. He now works at Hokuden Information Technology as a Software Developer.

Tomohiko Ito received his B.E. and M.E. degrees from the Muroran Institute of Technology, Hokkaido, Japan, in 2017 and 2019. He is currently a Ph.D. student in the Division of Engineering at Muroran Institute of Technology. His research interests center on sketch-based interfaces based on fuzzy logic for CAD systems.

Yuto Miwa received his B.E. and M.E. degrees in computational intelligence from the Muroran Institute of Technology, Hokkaido, Japan, in 2012 and 2014, respectively. His research interests centered on intelligent object snapping algorithms for sketch-based CAD systems based on fuzzy logic. He now works at NTT COMWARE Corporation as a Software Developer.

Takeshi Shibata received his D.Eng. degree in electronic and computer system engineering from Akita University in 2012. In 2013, he was employed as Technical Staff at Akita University. From 2014 to 2015, he was a Postdoctoral Fellow with Akita University. From 2015 to 2019 he was an Assistant Professor with Muroran Institute of Technology. Currently he is an Assistant Professor with Ibaraki University. His research interests include virtual reality techniques for archiving and handing-down traditional culture. He is a member of IPSJ, VRSJ, and IEICE.

Sato Saga received his M.E. and Ph.D. degrees from Hokkaido University, Hokkaido, Japan. From 1987 to 1989, he was engaged in computer science education in the Philippines as a member of the Japan Overseas Cooperation Volunteers. From 1989 to 1994, he was a Researcher at Tecnova Corporation. In 1994, he joined the Muroran Institute of Technology, where he is currently a Professor in the Department of Sciences and Informatics. His research interests include human interfaces based on soft computing.