IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 23, 2017, accepted July 20, 2017, date of publication July 31, 2017, date of current version August 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2733722

Sparse Parallel Algorithms for Recognizing Touch Topology on Curved Interactive Screens

KEIJI MIURA[1] AND KAZUKI NAKADA[2], (Member, IEEE)
[1]School of Science and Technology, Kwansei Gakuin University, Sanda 669-1337, Japan
[2]Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan

Corresponding author: Keiji Miura (miura@kwansei.ac.jp)

This work was supported by the Grants-in-Aid for Scientific Research, Japan Society for the Promotion of Science, under Grant 26520202 and Grant 16H01556.

ABSTRACT The advent of topological data analysis enabled us to extract topological invariants under object deformations and transformations, such as the number of loops in an image, which was conventionally unachievable with the filters of analog nature. However, the existing algorithms are mostly off-line or non-parallel, and therefore not suitable for interactive applications, such as touch sensors. Therefore, we previously proposed a real-time, distributed algorithm to compute the Euler characteristics as a touch invariant by summing up the Poincare–Hopf index for each pixel, which is fortunately sparse being zero for most pixels. However, the previous algorithm was specialized and restricted to plane screens with triangulated meshes. The explosive growth of the tablet or touch interface technology with super-thin screens or virtual realities is creating a new demand for software, for example, to detect if or not a touch wraps plastic bottles, coffee cup handles, or balls. In this paper, we extended the scope of our previous algorithm from plane to curved screens, including cylinders, toruses, and regular polyhedra, for solving truly topological sensing problems. We demonstrate that our implementation in processing, in the form of solely local logical operations and without any time-consuming iterative operations, returns and updates the correct topological invariants of touches on curved screens in real time.

INDEX TERMS Poincare-Hopf index, topological data analysis, sparse representation, touch screen, sensor networks, surface mesh, torus, icosahedron, neuromorphic engineering.

I. INTRODUCTION

The advent of topological data analysis [1]–[5] enabled us to extract topological invariants under object deformations and translations such as the number of islands (connected components) or holes in an image [6]–[8]. This recognition of invariants, conventionally unachievable by ''analogue'' filters [9]–[12], can provide valuable information in applications such as hand-written digit discrimination [13]–[18].

However, the inventive algorithms of computational topology mostly aimed at off-line computation on serial [19]–[24] or parallel system computers [25]–[27]. Therefore, the real-time algorithms, which is possibly distributed to achieve scalability, for computing topology remain to be developed for interactive applications such as touch sensors.

In a previous paper [28], we proposed a real-time algorithm to compute the Euler characteristics of a binary touch image via the graph theoretical or discretized Poincare-Hopf index [29]. Validated by topology, the proposed algorithm can accurately count the number of islands or holes in a touch image, irrespectively of the shapes and positions of the touches. The key idea is to compute the Poincare-Hopf index for each point and sum them for all the points to obtain the Euler characteristics. Among many algorithmic variants for Euler characteristics [7], [8], [24], [27], we believe that our fast algorithm uses the least resources. This is because the indices are non-zero only for (half of) the sparse ''critical'' points for a Morse or height function based on the topological as well as combinatorial formulation of the Poincare-Hopf index [29].

However, the previous algorithm was specialized and restricted to plane screens [28], although the explosive growth of the tablet or touch interface technology may generate new demands. Specifically, there is a potential demand for computing touch topology on curved screens such as plastic bottles, swim rings and balls, which are realizable by latest super-thin touch screens or virtual realities. For example, it is useful to know if a touch wraps a coffee cup handle or not. From a mathematical viewpoint, a coffee cup is topologically

equivalent to a torus. Thus it might suffice for many applications to extend our previous algorithm from planes to toruses or, possibly, other typical curved surfaces such as cylinders or balls.

This type of applications can truly take an advantage of the topological nature of Poincare-Hopf indices. The background mathematics of graph theoretical Poincare-Hopf indices [29] is not necessarily restricted to the triangulated plane and can actually extend rather straightforwardly to general graphs (clique complexes) including curved surfaces. Specifically, the sensor network can easily extend to any complicated shape in three dimensional space, say, with periodic boundary conditions like cylinders or balls.

In this paper, we propose a real-time algorithm to compute the Euler characteristics of a binary touch image on curved screens via the graph theoretical or discretized Poincare-Hopf index [28], [29]. Specifically, we extended our previous algorithm to cylinder-, torus- and ball-type screens with triangulated meshes. We demonstrate that our implementation in Processing, in the form of solely local logical operations and without any time-consuming iterative operations, returns and updates the correct topological invariants of touches in real time.

Although the background mathematics here is rather advanced, we tried to keep the paper as accessible as possible by engineers and designers. One of the goals of this paper is to import the state-of-the-art idea from mathematics and explain it as plainly as possible, as in our previous paper [28]. Our substantial contribution resides in the simplification of Poincare-Hopf indices for a binary touch image, especially on triangulated curved surfaces in this paper, by enumerating all possibilities and its circuit implementation solely as reduced logical operations to enable real-time tracking.

In Section II, we introduce the problem setting of a binary touch image and sketch a fast algorithm to compute the topology of touch shapes via the Poincare Hopf index. In Section III-A, we briefly review the mathematical backgrounds for the Poincare-Hopf index which is commonly used for plane-, cylinder-, torus-, and ball-type screens. In Section III-B–III-E, we implement plane-, cylinder-, torus- and ball-type screens, respectively. We demonstrate that our simple algorithm, that can be implemented in logical gates, returns and updates in real time the correct topological invariants of touches on each curved screen. In Section IV, we present a summary and discussions.

II. PROBLEM SETTINGS AND PREVIOUS RESULTS FOR COMPUTING TOUCH TOPOLOGY VIA SPARSE INDICES

Let us begin with a simple example of the eight-shape touch in Fig. 1 (left) for ease of comprehension. We assume that the tactile sensors are densely located on a two dimensional triangular lattice to sense tactile stimulations at each point. Note that a touch can have an arbitrary shape of finite size while each point sensor can sense and keep whether each point is touched so far or not.

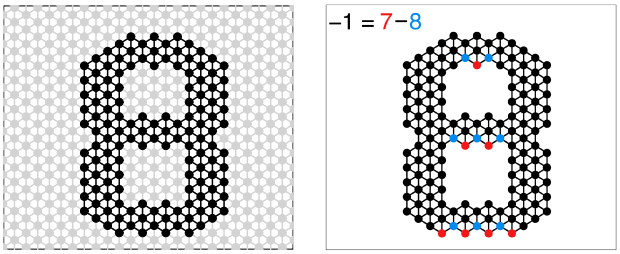

FIGURE 1. (left) Problem setting exemplified by a binary touch image (black) on a two dimensional triangular lattice (gray). We want the Euler characteristics, #islands − #holes (= 1 − 2 = −1, for this example), of an arbitrarily shaped touch as a topological invariant. The touched point is colored black. When the neighboring two points are both touched, they are connected by an edge (black line). When the neighboring three points are all touched, they are assumed to be "filled", that is, we do not count local triangles as a hole. (right) Poincare-Hopf index for each lattice point by color. The red and blue indicate the nonzero indices +1 and −1, respectively. Otherwise, the indices are zero. The Euler characteristics for this touch image can be computed as the sum of all the indices (=#red points − #blue points = 7 − 8 = −1).

The goal is to count the topological invariants, such as the number of islands (=1) or the number of holes (=2) in this binary image, irrespective of the shapes and positions of the stimulating touch. Specifically, we are interested in the Euler characteristics, which is defined as their difference,

$$\chi = \#\text{islands} - \#\text{holes}, \tag{1}$$

because the alternating sum or difference is generally easier to compute in a real time. It can be useful, for example, to count the number of marbles (without holes) in a binary image or to count the number of holes in a single connected touch, as a clue for pattern recognition. However, it is not easy to directly compute the numbers of islands and holes for general touches in an automatic way.

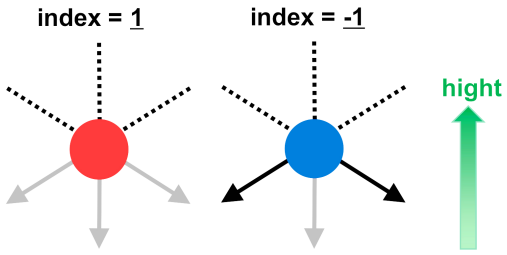

FIGURE 2. Rule for assigning Poincare-Hopf index to each point on plane screens. It is +1 (red), −1 (blue) or 0 (otherwise), depending on the downward patterns of connections. That is, a blue point is a point that is connected to only two side points out of three downward points, and a red point is not connected to any of three downward points.

Here we overview how to compute the Euler characteristics (=#islands − #holes) via the Poincare-Hopf indices. First, we compute the Poincare-Hopf index for each lattice point, as exemplified by color in Fig. 1 (right). The coloring rule (for a plane screen) will be explained later using Fig. 2, but we briefly mention that a blue point is a saddle point that is connected to only two side points out of three downward points, and a red point is a minimum point that

is not connected to any of three downward points. Finally, the Euler characteristics is obtained as the sum of Poincare-Hopf indices for all the points [28], [29]. In the case of Fig. 1, the Euler characteristics via Poincare-Hopf indices is given by

$$\chi = \#\text{red} - \#\text{blue} = 7 - 8 = -1. \tag{2}$$

As proven for general touches or graphs [29], this is equal to the ''topological'' Euler characteristics given as

$$\chi = \#\text{islands} - \#\text{holes} = 1 - 2 = -1. \tag{3}$$

Fig. 3 demonstrates that this algorithm also works for other examples on the plane screen.

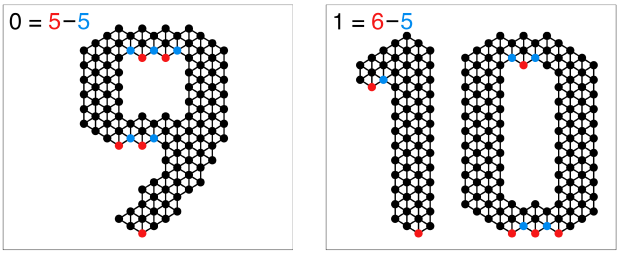

FIGURE 3. Additional two examples of Poincare-Hopf indices for plane screens. The red and blue colors indicate the nonzero indices +1 and −1, respectively. The Euler characteristics via the Poincare-Hopf index (=#red points − #blue points) indicated at topleft in each figure is equal to the "topological" Euler characteristics (=#islands − #holes), which is 0 = 1 − 1 (left) or 1 = 2 − 1 (right).

III. RESULTS

A. COMMON MATHEMATICAL BACKGROUNDS FOR PLANE-, CYLINDER-, TORUS- AND BALL-TYPE SCREENS

Non-curved plane screens were actually assumed in the previous results of Figs. 1 and 3 with Fig. 2 as a coloring rule. In order to consider general curved surfaces such as (triangulated) cylinders, toruses and balls as screens, we need a first principle definition of the Poincare Hopf index, instead of Fig. 2. Note that it is a common rule that the Euler characteristics of the touch shape on any triangulated curved surface can be computed by summing the Poincare-Hopf indices for all the lattice points. Only the coloring rule differs across different types of curved screens (eqs. 8, 19 and 20) as we will explain later subsection by subsection.

The Poincare-Hopf index for a given (i-th) vertex v_i in a GENERAL graph is defined as

$$m_i = 1 - \chi(\text{exit set for } v_i), \tag{4}$$

where the exit set for v_i is defined as the ''downstream'' neighbors of the vertex v_i on which a Morse function f_{Morse}, typically representing a height, takes smaller values than that on v_i [28], [29]. The condition for v_j to be included in the exit set for v_i can be mathematically written as

$$f_{Morse}(j) < f_{Morse}(i) \text{ for } i \text{ and } j \text{ neighbors}. \tag{5}$$

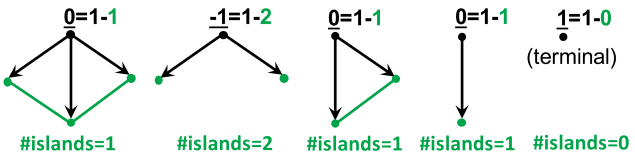

FIGURE 4. The rule for Poincare-Hopf index (=1− #islands) for all eight patterns of downward connections for plane screens. As there are three downward edges, the existence or absence of each edge results in eight patterns in total. However, to save space, the symmetric and equivalent three patterns were omitted. The green points and edges represent the exit set for each pattern. The green integers denote the numbers of isolated islands in the exit set. The underlined integers denote the Poincare-Hopf indices for each black top point. Note that the number of islands is one and thus the index is zero except for the two patterns shown in Fig. 2.

To be precise, the exit set is a subgraph. That is, in addition to the above vertices v_j's in the exit set, the edges connecting them, if they exist in the original graph, are also included in the exit set. The examples of exit sets (for the triangulated plane screen case) are shown in Fig. 4.

At this moment, it is still impossible to derive a general coloring rule from the above abstract definition of eq. 4. Instead, we will restrict ourselves to specific types of screens: plane-, cylinder-, torus- and ball-types. In those implementations, an exit set is obtained as a subgraph for each lattice point on a triangulated screen and you can automatically compute the Euler characteristic for that exit set as

$$\chi(\text{exit set}) = (\#\text{vertices in exit set}) - (\#\text{edges in exit set}), \tag{6}$$

where we only count the touched vertices and the edges with two terminal vertices both touched. Note that this value is equal to the ''topological'' definition: $\chi = \#\text{islands} - \#\text{holes}$.

Regarding the Morse function, which determines an exit set, arbitrarily-defined any Morse function actually works. That is, although different Morse functions result in different exit sets or Poincare-Hopf indices, the sum of the indices for all lattice points or the Euler characteristics does not depends on the choice of the Morse function. Nonetheless, we solely use the height (essentially y- or z-coordinate) as a Morse function, because it is not only geometrically simple enough for deriving coloring rules but also the derived indices are sparse being non-zero only for a few critical points. Thus our algorithmic contribution is to choose an appropriate Morse function and write down the resulting coloring rule for each type of curved screens. We find different but simple coloring rules for computing the Poincare-Hopf indices for each type of curved screens, which will be shown in the following subsections.

B. IMPLEMENTING POINCARE-HOPF INDICES FOR PLANE-TYPE SCREENS WITHOUT PERIODIC BOUNDARY CONDITION

To obtain the Poincare-Hopf indices for a plane-type screen as in Fig. 1, we simply need to count the number of islands (as there is no hole) in the exit set for each lattice point in eq. 4. Note that you can equivalently compute the number of

vertices and edges in the exit set as in eq. 6, leading to the same results. In Fig. 2, the red case has zero downward island and the blue case has two downward islands. The cases other than Fig. 2, the entire exit set is connected (#islands $=$ 1) as enumerated in Fig. 4. Note that although any generic Morse function actually works for defining exit sets, here we solely used the y-axis itself or height as a Morse function for simplicity.

Then, the Poincare-Hopf index for a lattice point, (i, j), on a plane screen is given as

$$m_{[i,j]} = 1 - (\#\text{islands in exit set for } (i, j)). \tag{7}$$

Therefore it is 1 for the red point, -1 for the blue point, and 0 otherwise. As is evident in Figs. 1 and 3, the index detects the saddle-like (blue) and the minimum points (red) and they correctly amount to the Euler characteristics (=#islands $-$ #holes). Note that in the field of topology it is well known that the difference of the numbers of red and blue points is an invariant under morphing, because adding an extra bunch to a camel back only increases one saddle and one minima together, keeping the difference. That is, even if we prolong a shape, it simply adds the equal numbers of saddles (blue points) and minima (red points). Although the above explanation is just intuitive, it is proven that this definition correctly computes the Euler characteristics [29]. Although the conventional Morse theory count both the minima and maxima, here we count only minima utilizing the symmetry to save half computational resources.

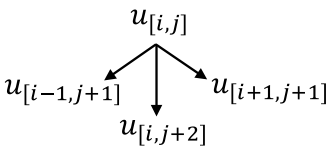

FIGURE 5. Notation to derive a logical operation for Poincare-Hopf index for downsampled plane screens. As we downsample from a square lattice $u_{[i,j]}$ to obtain a triangular lattice, we only consider the points where $i + j$ is even and let $u_{[i,j]} = 0$ for $i + j$ odd.

Here we show a concrete algorithm to compute the Poincare Hopf index for each point in a plane screen. As the image is often given in the form of a square lattice, we first assume an image consists of tiny square pixels indexed as $u_{[i,j]}$. In that case, you can downsample the lattice points in order to obtain a triangular lattice as in Fig. 5. To be specific, you can only consider the points where $i + j$ is even or let $u_{[i,j]} = 0$ if $i + j$ odd. In the style shown in Fig. 5, the downsampled graph has the vertical adjacency and the vertices for $u_{[i,j]}$ and $u_{[i,j+2]}$ are connected. Note that in the Processing implementation we used logical operations instead of ''if then'' branch to gain speed.

Then by using the notation in Fig. 5, the rule to compute the Poincare-Hopf index in Fig. 2 at the point $(x, y) = (i, j)$ can be described as

$$m_{[i,j]} = u_{[i,j]}\{(1 - u_{[i-1,j+1]})(1 - u_{[i,j+2]})(1 - u_{[i+1,j+1]}) - u_{[i-1,j+1]}(1 - u_{[i,j+2]})u_{[i+1,j+1]}\}. \tag{8}$$

Note that $m_{[i,j]}$ can be nonzero only when either of the two terms in the right hand side is nonzero, corresponding to the two cases in Fig. 2. That is, the first term corresponds to the red point while the second to the blue point. Again, the Euler characteristics of an image is the sum of the Poincare-Hopf indices for all the points.

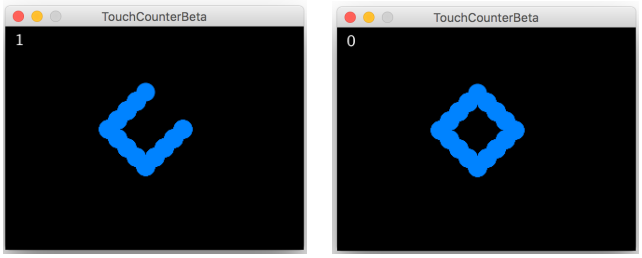

FIGURE 6. Real-time implementation of plane screens in Processing. The number at topleft of each figure indicates the Euler characteristics. The implementation successfully computes the Euler characteristics of touches automatically and updates it for newly added points in real time. For the left example, χ = #islands $-$ #holes $= 1 - 0 = 1$. For the right example, $\chi = 1 - 1 = 0$. Note that the Euler characteristics decreases by one when the touch forms a loop.

In order to demonstrate the proposed algorithm, we concretely implemented it in Processing, which simulated an interactive touch screen. Fig. 6 shows that it successfully computes the Euler characteristics of touches automatically and updates it for newly added points in real time. Note that the Euler characteristic change when the touch forms a loop. To achieve the speed for interactive responsiveness, the implementation was solely designed as logical operations as evident in eq. 8. Thus our algorithm is ready to be translated, say, into Raspberry Pi or Arduino.

C. IMPLEMENTING POINCARE-HOPF INDICES FOR CYLINDER-TYPE SCREENS WITH PERIODIC BOUNDARY CONDITION FOR x-AXIS

Here we implement a cylinder-type screen by simply imposing the periodic boundary condition for x-axis to the previous plane screen.

To obtain the Poincare-Hopf indices for a cylinder-type screen, we again simply need to count the number of islands (as there is no hole) in the exit set in eq. 4. Note that although any generic Morse function works for defining exit sets, here we again used the y-axis itself or height as a Morse function for simplicity. Even if you impose the periodic boundary condition on x-axis as in Fig. 7, the rule in Fig. 2 essentially does not change at all. That is, the red case has zero downward island and the blue case has two downward islands. The cases other than Fig. 2, the entire exit set is connected (#islands $=$ 1) as enumerated in Fig. 4. Then, the Poincare-Hopf index at a lattice point, (i, j), is again given as

$$m_{[i,j]} = 1 - (\#\text{islands in exit set for } (i, j)). \tag{9}$$

Therefore it is 1 for the red point, -1 for the blue point, and 0 otherwise. The examples in Fig. 8 demonstrate how the

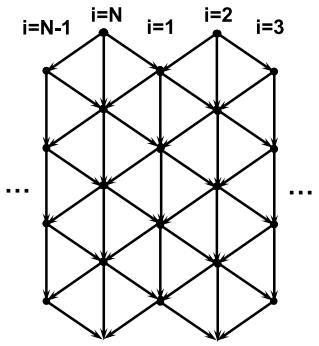

FIGURE 7. *x*-axis periodic boundary condition for cylinder-type screens, where $u_{[N+1,j]}$ and $u_{[0,j]}$ that can appear in Fig. 5 or eq. 8, can be practically read as $u_{[1,j]}$ and $u_{[N,j]}$, respectively. The *x*-width of the screen, *N*, is assumed to be even. Again, As we downsample from a square lattice $u_{[i,j]}$ to obtain a triangular lattice, we only consider the points where $i+j$ is even and let $u_{[i,j]}=0$ for $i+j$ odd.

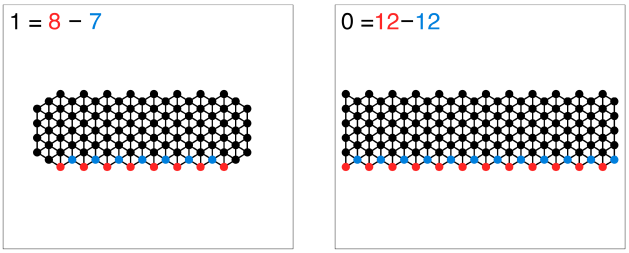

FIGURE 8. Examples of Poincare-Hopf indices for lattice points for cylinder-type screens. The nonzero indices +1 and −1 were indicated by red and blue colors, respectively. The Euler characteristics via the Poincare-Hopf index (=#red points − #blue points) indicated at topleft in each figure is equal to the "topological" Euler characteristics (=#islands − #holes), which is 1 = 1 − 0 (left) or 0 = 1 − 1 (right). Note that the Euler characteristics decreases by one when the touch wraps the cylinder and adds an extra loop.

difference of the numbers of red and blue points balances to amount to the "topological" Euler characteristics under the periodic boundary condition.

In practice, you can actually reuse eq. 8 (as well as Figs. 5 and 7) by reading values as

$$u_{[0,j]} := u_{[N,j]}, \quad \text{and} \\ u_{[N+1,j]} := u_{[1,j]}, \tag{10}$$

for "virtual" external points, whose values are set in order to satisfy the periodic boundary condition. (We also utilized these virtual external points in Processing implementation, when we plot two points on both sides of boundaries if either of them is touched, emphasizing the periodic boundary condition for *x*-axis.) Fig. 9 demonstrates that the implementation of cylinder-type screen in Processing successfully computes the Euler characteristics of touches automatically and updates it for newly added points in real time. Note that the Euler characteristic changes when the touch wraps the cylinder and forms a hole.

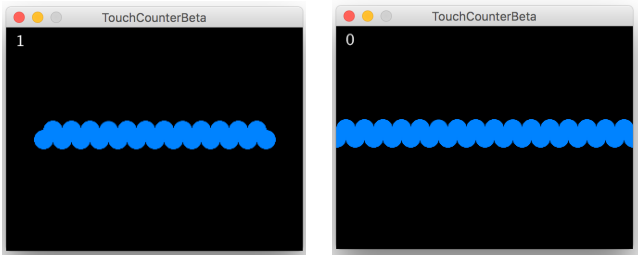

FIGURE 9. Real-time implementation of cylinder-type screens in Processing. The top left figures indicate the Euler characteristics. For the left example, χ = #islands − #holes = 1 − 0 = 1. For the right example, $\chi = 1-1=0$. Note that the Euler characteristics decreases by one when the touch wraps the cylinder and adds an extra loop.

D. IMPLEMENTING POINCARE-HOPF INDICES FOR TORUS-TYPE SCREENS WITH PERIODIC BOUNDARY CONDITIONS FOR x- AND y-AXES

Here we implement a torus-type dual-periodic screen by adding the virtual *x*-boundary points as before as well as the exceptional rules in the *y*-axis boundary area to the previous plane screen. Note that although any generic Morse function works for defining exit sets, here we again used the (inverted) *y*-axis itself or height as a Morse function for simplicity, that is,

$$f_{Morse}(i,j) = -j. \tag{11}$$

Although physically we have periodic boundary conditions for both *x*- and *y*-axes, only *x*-axis boundary condition can be solved by setting "virtual" boundary points as in eq. 10. This is because the height function is asymmetric for *x* and *y*. So we need to manage *y*-axis boundary condition with the exceptional coloring rules there.

Let us begin with the interior points of a rectangular screen, to which we impose periodic boundary conditions later. To obtain the Poincare-Hopf index of an interior point, we again simply need to count the number of islands (as there is no hole) in the exit set in eq. 4. Even if you impose the periodic boundary conditions, the rule in Fig. 2 essentially does not change at all for the interior points. This is because the Morse function is simply the *y*-coordinate and continuous as far as you only look at the neighbors of an interior point locally. Then, the Poincare-Hopf index for an interior point, (i,j), is again given as

$$m_{[i,j]} = 1 - (\#\text{islands in exit set for } (i,j)), \tag{12}$$

which is 1 for the red point, −1 for the blue point, and 0 otherwise. So we can again reuse eq. 8 for an interior point.

Next we consider the *x*-axis boundary points. We simply impose the periodic boundary condition on *x*-axis and avoid the complicated rules. That is, we can again use the virtual boundary points as in eq. 10, when reading values in eq. 8.

Then the only complication for the coloring rule comes from the *y*-axis boundary points. Here we must go back to the original definition of the Poincare-Hopf index in eq. 4

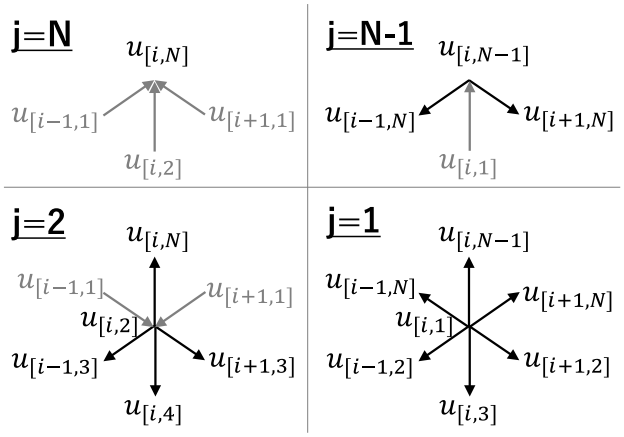

FIGURE 10. Possible exit sets (black) for $u[i, j]$ for boundary j's to derive exceptional rules for torus-type screen. Note that we only consider the points where $i + j$ is even and let $u_{[i,j]} = 0$ for $i + j$ odd as we downsampled from a square lattice $u_{[i,j]}$ to obtain a triangular lattice.

and consider all possible cases of exit sets as in Fig. 10. Note that as the same rule holds for all i's due to the x-axis periodic boundary condition, we only need to consider the four cases for $j = N, N-1, 2$ and 1 separately.

For $j = N$, there is no exit set because

$$f_{Morse}(i, N) = -N \tag{13}$$

is the lowest, meaning that there is no downstream (or lower) point for the boundary point, (i, N), as in Fig. 10. This results in

$$m_{[i,N]} = u_{[i,N]}. \tag{14}$$

Equivalently, you can also set the "virtual" points as

$$\begin{aligned} u_{[*,N+1]} &= 0, \quad \text{and} \\ u_{[*,N+2]} &= 0, \quad \text{for any } *(= 1, 2, 3, \ldots), \end{aligned} \tag{15}$$

when reading values in eq. 8. The latter way allows you to consistently use eq. 8 for evaluation.

For $j = N - 1$, there are only two possible exit points because

$$f_{Morse}(i, 1) = -1 \tag{16}$$

is actually the highest and the point $(i, 1)$ cannot be in the exit set for the boundary point $(i, N-1)$, as in Fig. 10. This results in

$$\begin{aligned} m_{[i,N-1]} &= u_{[i,N-1]}\{(1 - u_{[i-1,N]})(1 - u_{[i+1,N]}) \\ &\qquad - u_{[i-1,N]}u_{[i+1,N]}\}. \\ &= u_{[i,N-1]}\{1 - u_{[i-1,N]} - u_{[i+1,N]}\}. \end{aligned} \tag{17}$$

Equivalently, you can set the "virtual" points as

$$u_{[*,N+1]} = 0, \quad \text{for any } *(= 1, 2, 3, \ldots), \tag{18}$$

when reading values in eq. 8. Note that these virtual points are already included in and consistent with eq. 15. Thus, you can use solely eq. 8 for all the cases so far.

For $j = 2$, eq. 4 becomes,

$$\begin{aligned} m_{[i,2]} = u_{[i,2]}\{&(1 - u_{[i-1,3]})(1 - u_{[i,4]})(1 - u_{[i+1,3]}) \\ &- u_{[i-1,3]}(1 - u_{[i,4]})u_{[i+1,3]} - u_{[i,N]}\}. \end{aligned} \tag{19}$$

This is because only the last term $-u[i, N]$ was added to the original eq. 8, depending on if the extra exit set should be counted as an island or not in the bottom left of Fig. 10.

For $j = 1$, all the neighbors can be potentially included in the exit set. To compute eq. 4, here we used eq. 6 as it is almighty and obtained,

$$\begin{aligned} m_{[i,1]} = u_{[i,1]}\{1 &- u_{[i+1,N]}(1 - u_{[i+1,2]}) \\ &- u_{[i+1,2]}(1 - u_{[i,3]}) \\ &- u_{[i,3]}(1 - u_{[i-1,2]}) \\ &- u_{[i-1,2]}(1 - u_{[i-1,N]}) \\ &- u_{[i-1,N]}(1 - u_{[i,N-1]}) \\ &- u_{[i,N-1]}(1 - u_{[i+1,N]})\}. \end{aligned} \tag{20}$$

Note that u counts the number of vertices and uu counts the number of edges essentially. Another way to interpret is that this equation detects the difference of neighboring u's to count the beginnings of the islands clockwise. This equation also happens to work when there is a hole or when all neighboring u's are 1.

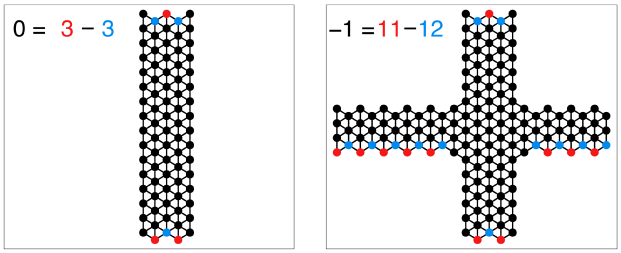

FIGURE 11. Example of Poincare-Hopf indices assigned on lattice points for torus-type screen. The nonzero indices +1 and −1 were indicated by red and blue colors, respectively. The Euler characteristics via the Poincare-Hopf index (=#red points − #blue points) indicated at topleft in each figure is equal to the "topological" Euler characteristics (=#islands − #holes), which is 0 = 1 − 1 (left) or −1 = 1 − 2 (right). Note that the Euler characteristics decreases by one when the touch wraps the torus and adds an extra loop.

The examples in Fig. 11 demonstrate how the difference of the numbers of red and blue points balances to amount to the "topological" Euler characteristics even in the torus-type screen. Practically, we consistently used eq. 8 (with the help of virtual points) as much as possible, except for eq. 19 when $j = 2$ and eq. 20 when $j = 1$. Fig. 12 demonstrates that the real-time implementation of torus-type screens in Processing successfully computes the Euler characteristics of touches automatically and updates it for newly added points in real time. Note that the Euler characteristic changes when the touch wraps the torus and adds an extra loop.

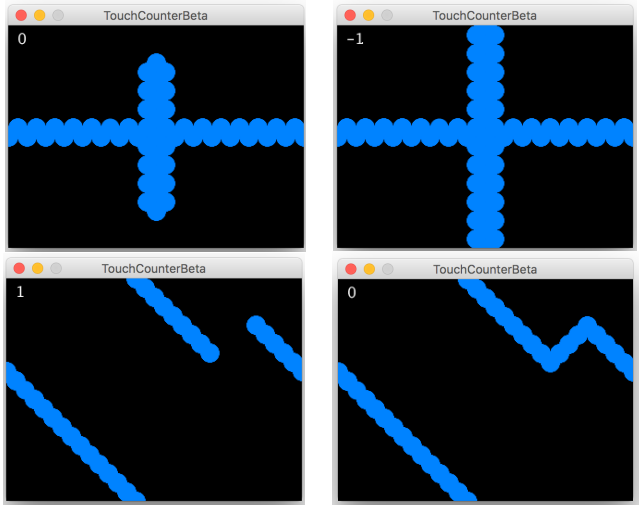

FIGURE 12. Real-time Implementation of torus-type screen in Processing. The top left figures indicate the Euler characteristics, which decreases by one when the touch wraps the torus and adds an extra loop. For the topleft example, χ = #islands − #holes = 1 − 1 = 0. For the topright example, $\chi = 1 - 2 = -1$. For the bottomleft example, χ = #islands − #holes = 1 − 0 = 1. For the bottomright example, $\chi = 1 - 1 = 0$.

E. IMPLEMENTING POINCARE-HOPF INDICES FOR BALL-TYPE SCREENS AS ICOSAHEDRON

Here we implement a ball-type screen as an icosahedron. First we assign coordinates for the twelve points by using only 0, $\frac{1}{2}$ and the golden ratio, $\phi = \pm\frac{1+\sqrt{5}}{2}$ as

$$(x, y, z) = (0, \pm\frac{1}{2}, \pm\frac{\phi}{2}), (\pm\frac{\phi}{2}, 0, \pm\frac{1}{2}), (\pm\frac{1}{2}, \pm\frac{\phi}{2}, 0). \quad (21)$$

This is based on the fact that the three rectangles with the golden ration can form an icosahedron [30], [31]. We refer each point by the number assigned in Fig. 13.

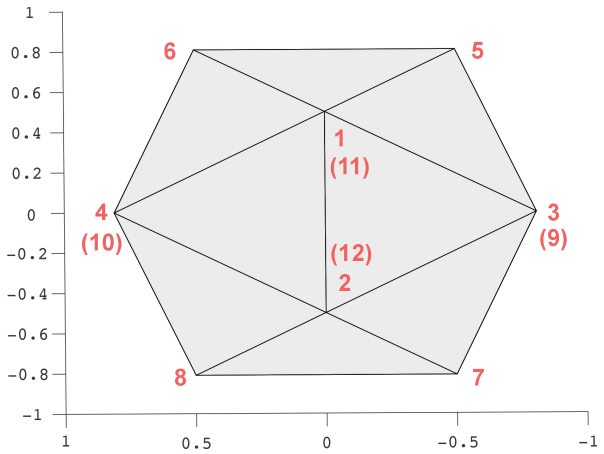

FIGURE 13. The labels and *xy*-coordinates for each point in icosahedron. The parentheses indicate the label for the point on the back.

Then, although any generic Morse function works for defining exit sets, here we use the slightly tilted z-axis or height as a Morse function for simplicity:

$$f(x, y, z) = 100z + 10y + x. \quad (22)$$

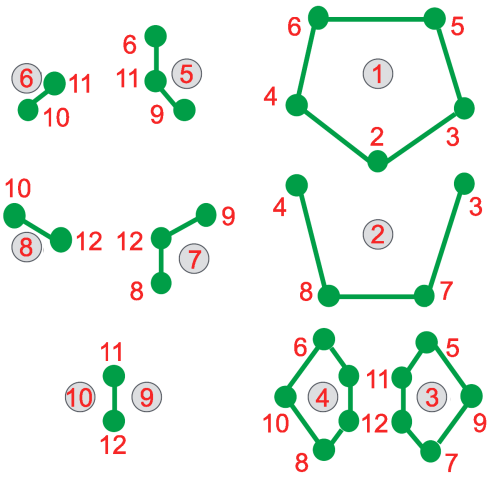

FIGURE 14. Possible exit sets (green) for each point (black, isolated). The red numbers indicate the labels assigned to each point in Fig. 13.

This means that the height is basically determined or dominated by z. However, when two points share the same z-coordinate, the y-coordinate dominates. Similarly, when two points share the same z- and y-coordinates, the x-coordinate matters finally. That is, the tilt is necessary to break the symmetry in positions.

Here we must go back to the original definition of the Poincare-Hopf index eq. 4 and consider all possible cases of exit sets as in Fig. 14. We set $u_i = 0, 1$ depending on if the i-th point was touched or not. Then the Poincare-Hopf indices for the twelve points end up with

$$\begin{aligned}
m_1 &= u_1\{1 - u_2(1 - u_3) - u_3(1 - u_5) \\
&\quad - u_5(1 - u_6) - u_6(1 - u_4) - u_4(1 - u_2)\}, \\
m_2 &= u_2\{1 - u_3(1 - u_7) - u_7(1 - u_8) - u_8(1 - u_4) - u_4\}, \\
m_3 &= u_3\{1 - u_5(1 - u_{11}) - u_{11}(1 - u_{12}) \\
&\quad - u_{12}(1 - u_7) - u_7(1 - u_9) - u_9(1 - u_5)\}, \\
m_4 &= u_4\{1 - u_6(1 - u_{11}) - u_{11}(1 - u_{12}) \\
&\quad - u_{12}(1 - u_8) - u_8(1 - u_{10}) - u_{10}(1 - u_6)\}, \\
m_5 &= u_5\{1 - u_6(1 - u_{11}) - u_{11}(1 - u_9) - u_9\}, \\
m_6 &= u_6\{1 - u_{10}(1 - u_{11}) - u_{11}\}, \\
m_7 &= u_7\{1 - u_9(1 - u_{12}) - u_{12}(1 - u_8) - u_8\}, \\
m_8 &= u_8(1 - u_{12})(1 - u_{10}), \\
m_9 &= u_9(1 - u_{11})(1 - u_{12}), \\
m_{10} &= u_{10}(1 - u_{11})(1 - u_{12}), \\
m_{11} &= u_{11}(1 - u_{12}), \\
m_{12} &= u_{12}. \qquad (23)
\end{aligned}$$

One interpretation of these equations is that all the rules are just specific cases of m_1, where the possible exit set forms a loop. That is, fixing some u_i's in the equation for m_1 to be zero can recover the rules for other m_i. Note that here we ignore specific labels assigned for the points as they are arbitrary.

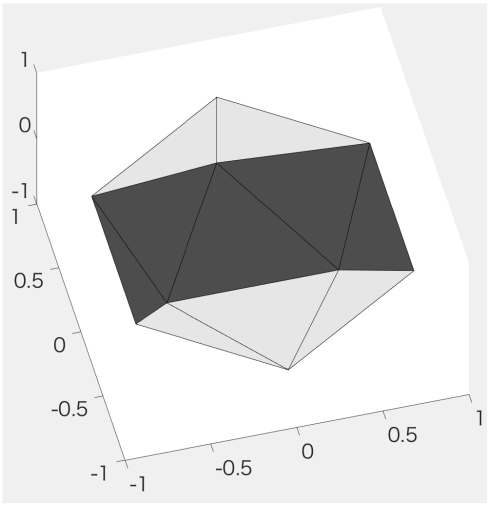

FIGURE 15. An example touch (filled by black) that wraps a ball-type screen, where $\boldsymbol{u} = (1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1)$ and $\boldsymbol{m} = (0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$. The Euler characteristics for the touch, $\chi = \sum_{i=1}^{12} m_i = 0$, indicates the existence of a nontrivial loop.

The most general rule for m_1 or ring case is already given for $j = 1$ of torus-type screens as in Fig. 10.

The Matlab simulation successfully detected the hole in the touch in Fig. 15 by correctly returning $\chi = 0$.

IV. SUMMARY AND DISCUSSIONS

In this paper, we proposed a real-time algorithm to compute the Euler characteristics of a binary touch image on curved surface via Poincare-Hopf indices. Specifically we extended our previous algorithms to cylinder-, torus- and ball-type screens. The proposed algorithm accurately computes the Euler characteristics (=#islands − #holes) for a touch image, irrespectively of the shapes and positions of the touches. To quickly obtain the Euler characteristics, you simply sum the Poincare-Hopf indices only for a few active points. That is, the advantage of our implementation resides in the real-time computation supported by the sparse Poincare-Hopf indices, which enables interactive applications with (possibly super-thin) curved touchscreens. Our implementation in Processing (or Matlab simulations for ball-type screens) returned and updated the correct topological invariants of touches in real time, although its solely local logical form representation, never specialized to our Processing code, is implementable in any circuits such as FPGAs, Arduino, or Raspberry Pi.

Although we focused on the typical surfaces like cylinders, toruses and balls in this paper as concrete examples, the algorithm actually extends to general graphs (clique complexes) rather straightforwardly, if you do not mind the assistance of numerical methods. To be precise, with a randomly defined Morse function, you can automatically compute an exit set and the numbers of vertices and edges therein in eq. 6, leading to the Poincare-Hopf indices in eq. 4. Note that although different Morse functions result in different exit sets, the sum of the Poincare-Hopf indices or the Euler characteristics does not change [29]. Specifically, the Morse function need not to be a height function. The reason that we set our Morse function as real height in three dimensional space in this paper was just for geometrical interpretability to avoid complications. You can generally compute the Poincare-Hopf indices for any graphs numerically, even if you can not write down a general rule for the Poincare-Hopf index explicitly. Thus, with the assistance of numerical computation, you can extend the scope of the Poincare-Hopf index approach to any graphs, which may be comfortable for some applications.

Regarding the computational speed, the algorithm proposed in the current paper is much faster than our previous implementation [27]. This is because the previous one needed an iterative matrix multiplication, although our previous one and other offline algorithms can compute not only the Euler characteristics (=#islands − #holes) but also individual Betti numbers such as #islands or #holes separately, at the expense of the computational time. Therefore the previous offline algorithms can be complementary to the proposed algorithm. To be concrete, you can utilize the proposed algorithm if you only need the Euler characteristics but not each component of its alternating sum. An alternative algorithm for the Euler characteristics without iterations could be to use the Euler's formula: χ = #points − #edges + #faces. However, the counting the number of triangular faces costs resources as it requires to confirm whether all the three points for each triangle (triplets) exist or not. In general, the algorithm that uses as little (active) points as possible such as the proposed algorithm that uses the critical points may be resource efficient. Thus our algorithm, which counts only half of the critical points, can be the fastest among the algorithmic variants for computing Euler characteristics.

REFERENCES

[1] Z. Arai, H. Kokubu, and P. Pilarczyk, "Recent development in rigorous computational methods in dynamical systems," *Jpn. J. Ind. Appl. Math.*, vol. 26, no. 2, pp. 393–417, Oct. 2009.

[2] Y. Baryshnikov and R. Ghrist, "Target enumeration via euler characteristic integrals," *SIAM J. Appl. Math.*, vol. 70, no. 3, pp. 825–844, 2009.

[3] Y. Baryshnikov and R. Ghrist, "Euler integration over definable functions," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 21, pp. 9525–9530, 2010.

[4] J. Curry, R. Ghrist, and M. Robinson, "Euler calculus with applications to signals and sensing," in *Proc. Symp. Appl. Math.*, vol. 70. 2012, pp. 75–146.

[5] M. Gameiro, Y. Hiraoka, S. Izumi, M. Kramar, K. Mischaikow, and V. Nanda, "A topological measurement of protein compressibility," *Jpn. J. Ind. Appl. Math.*, vol. 32, no. 1, pp. 1–17, Mar. 2015.

[6] W. Fulton, *Algebraic Topology: A First Course*. New York, NY, USA: Springer-Verlag, 1995.

[7] Y. Matsumoto, *An Introduction to Morse Theory*. Providence, RI, USA: AMS, 2001.

[8] R. Ghrist, *Elementary Applied Topology*. North Charleston, SC, USA: CreateSpace, 2014.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer-Verlag, 2006.

[10] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3642–3649.

[11] M. Ishikawa, "Learning of modular structured networks," *Artif. Intell.*, vol. 75, no. 1, pp. 51–62, May 1995.

[12] M. Ishikawa, "Structural learning with forgetting," *Neural Netw.*, vol. 9, no. 3, pp. 509–521, Apr. 1996.
[13] A. Hirata *et al.*, "Geometric frustration of icosahedron in metallic glasses," *Science*, vol. 341, no. 6144, pp. 376–379, 2013.
[14] K. Miura and T. Aoki, "Hodge–Kodaira decomposition of evolving neural networks," *Neural Netw.*, vol. 62, pp. 20–24, Feb. 2015.
[15] K. Miura and T. Aoki, "Scaling of Hodge–Kodaira decomposition distinguishes learning rules of neural networks," in *Proc. IFAC CHAOS*, 2015, pp. 175–180.
[16] Z. Chen, S. N. Gomperts, J. Yamamoto, and M. A. Wilson, "Neural representation of spatial topology in the rodent hippocampus," *Neural Comput.*, vol. 26, no. 1, pp. 1–39, 2014.
[17] C. Giusti, E. Pastalkova, C. Curto, and V. Itskov, "Clique topology reveals intrinsic geometric structure in neural correlations," *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 4, pp. 13455–13460, 2015.
[18] M. W. Reimann *et al.*, "Cliques of neurons bound into cavities provide a missing link between structure and function," *Front. Comput. Neurosci.*, vol. 11, p. 48, Jun. 2017.
[19] H. Edelsbrunner and J. L. Harer, *Computational Topology*. Providence, RI, USA: AMS, 2009.
[20] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*. New York, NY, USA: Springer-Verlag, 2010.
[21] G. Carlsson and V. de Silva, "Zigzag persistence," *Found. Comput. Math.*, vol. 10, no. 4, pp. 367–405, 2010.
[22] K. Mischaikow and V. Nanda, "Morse theory for filtrations and efficient computation of persistent homology," *Discrete Comput. Geometry*, vol. 50, no. 2, pp. 330–353, Sep. 2013.
[23] H. Edelsbrunner, *A Short Course in Computational Geometry and Topology*. Berlin, Germany: Springer-Verlag, 2014.
[24] O. Delgado-Friedrichs, V. Robins, and A. Sheppard, "Skeletonization and partitioning of digital images using discrete morse theory," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 654–666, Mar. 2015.
[25] A. Tahbaz-Salehi and A. Jadbabaie, "Distributed coverage verification in sensor networks without location information," in *Proc. 47th IEEE Conf. Decision Control*, Dec. 2008, pp. 4170–4176.
[26] Z. Arai, K. Hayashi, and Y. Hiraoka, "Mayer–Vietoris sequences and coverage problems in sensor networks," *Jpn. J. Ind. Appl. Math.*, vol. 28, no. 2, pp. 237–250, 2011.
[27] K. Miura and K. Nakada, "Neural implementation of shape-invariant touch counter based on Euler calculus," *IEEE Access*, vol. 2, pp. 960–970, Aug. 2014.
[28] K. Miura and K. Nakada, "Real-time computing of touch topology via Poincare–Hopf index," *IEEE Access*, vol. 3, pp. 2566–2571, 2015.
[29] O. Knill. (2012). "A graph theoretical Poincare–Hopf theorem." [Online]. Available: https://arxiv.org/abs/1201.1162
[30] P. R. Cromwell, *Polyhedra*. Cambridge, MA, USA: Cambridge Univ. Press, 1997.
[31] D. Sutton, *Platonic & Archimedean Solids*. New York, NY, USA: Bloomsbury, 2002.

KEIJI MIURA received the Ph.D. degree in science from Kyoto University, Kyoto, Japan, in 2006. From 2006 to 2008, he was a JSPS Research Fellow with the University of Tokyo. From 2008 to 2011, he was a JST PRESTO Researcher with Harvard University. From 2011 to 2015, he was an Assistant Professor with Tohoku University. Since 2015, he has been an Associate Professor with Kwansei Gakuin University. His research area is mathematical neuroscience.

KAZUKI NAKADA received the B.Sc. degree from the Department of Mathematics, Hokkaido University, Sapporo, Japan, in 1999, and the M.E. and Ph.D. degrees from the Department of Electrical Engineering, Hokkaido University, in 2002 and 2005, respectively. From 2005 to 2011, he was an Assistant Professor with the Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Kitakyushu, Japan. In 2011, he joined the Advanced Electronics Research Division, INAMORI Frontier Research Center, Kyushu University, Fukuoka, Japan, as a Research Associate. From 2013 to 2015, he was with the Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan, as a Research Associate. From 2015, he is currently with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan, as a Contract Assistant Professor. His main research interests are the mathematical analysis and physical implementation of unconventional computing, including cognitive neuromorphic computing. Toward the research aim, he has been currently exploring novel design and control principles exploiting cooperative phenomena in silicon electronics and spintronics.