IEEE Access
The journal for rapid open access publishing

Received October 31, 2015, accepted November 14, 2015, date of publication November 30, 2015,
date of current version December 16, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2504387

Real-Time Computing of Touch Topology via Poincare–Hopf Index

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: K. Miura (miura@kwansei.ac.jp)

This work was supported by the Japan Society for the Promotion of Science through the Grants-in-Aid for Scientific Research under Grant 26520202 and Grant 24700301.

ABSTRACT While visual or tactile image data have been conventionally processed via filters or perceptron-like learning machines, the recent advances of computational topology may make it possible to successfully extract the global features from the local pixelwise data. In fact, some inventive algorithms have succeeded in computing the topological invariants, such as the number of objects or holes and irrespective of the shapes and positions of the touches. However, they are mostly offline algorithms aiming at big data. A real-time algorithm for computing topology is also needed for interactive applications such as touch sensors. Here, we propose a fast algorithm to compute the Euler characteristics of touch shapes by using the Poincare–Hopf index for each pixel. We demonstrate that our simple algorithm, implemented solely as logical operations in Arduino, correctly returns and updates the topological invariants of touches in real time.

INDEX TERMS Poincare-Hopf index, topology, invariance, touch counter, sensor networks.

I. INTRODUCTION

Although our brain can quickly count the number of, say, objects on a screen or touches on a skin, irrespectively of their shapes and positions [1]–[3]. it is still difficult to extract such global or topological features from a pixelwise image in real time pattern recognition. Although various filters including perceptrons have been conventionally used for pattern recognition [4], [5], they do not perfectly achieve topological invariance under object deformations and translations. Generally, their analogue nature of processing makes it difficult to count a number in integer with the infinite precision [6], [7]. Therefore, it is worth examining an alternative algorithm, which can be, hopefully, realized as simple and fast operations with the perfect invariance guaranteed.

To compute the topological invariants such as the number of distinct objects in an image, the algorithms based on topology, a major area of mathematics [8]–[10], can be more effective than *ad hoc* algorithms. Fortunately, recent advances of the field of computational topology made it possible to compute topological invariants in an accessible way [11]–[18]. However, the inventive algorithms of computational topology mostly aimed at off-line computation on serial [19]–[22] or parallel system computers [23]–[25]. Therefore the real-time algorithms and their implementations for computing topology remain to be developed for interactive applications such as touch sensors.

In this paper, we propose a real-time algorithm to compute the Euler characteristics of a binary touch image via the graph theoretical or discretized Poincare-Hopf index [26]. Validated by topology, the proposed algorithm can accurately count the number of islands in a touch image or the number of holes in a touch, 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 [9], [10], [22], [25], we believe that our fast algorithm uses the least resources. This is because it only utilizes (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 [26]. We demonstrate that our implementation in Arduino, 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.

The background mathematics here is rather advanced, but we try 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. Our substantial contribution resides in

IEEE Access
The journal for rapid open access publishing

the simplification of Poincare-Hopf indices for a binary touch image on a two-dimensional triangular lattice by exhausting possibilities and its circuit implementation solely as reduced logical operations for enabling real-time tracking.

In Sec. 2, we introduce the problem setting of a binary touch image and propose a fast algorithm to compute the topology of touch shapes via Poincare Hopf index. After we briefly review the mathematics of the Poincare-Hopf index for computing the Euler characteristics, we demonstrate that our simple algorithm, that can be implemented in logical gates, returns and updates the correct topological invariants of touches in real time. In Sec. 3, we present a summary and discussions.

II. RESULTS

We consider the problem of real-time computing the topological invariants of touches, which are independent of touch shapes and positions. As the alternating sum or difference is generally easier to compute, here we are especially interested in the Euler characteristics,

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

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 touch as we will see.

A. PROBLEM SETTING

For ease of comprehension, let us begin with a simple example of a "six" shape touch in Figure 1. The tactile sensors are densely located on a two dimensional triangular lattice to sense tactile stimulations at each point. The goal is to count the topological invariant, in this example the number of holes (=1) in this binary image (2-clique complex with the graph as 1-skeleton), irrespective of the shapes and positions of the stimulating touch. Note that a touch has an arbitrary shape of finite size while each point sensor can sense and keep whether each point is touched so far or not.

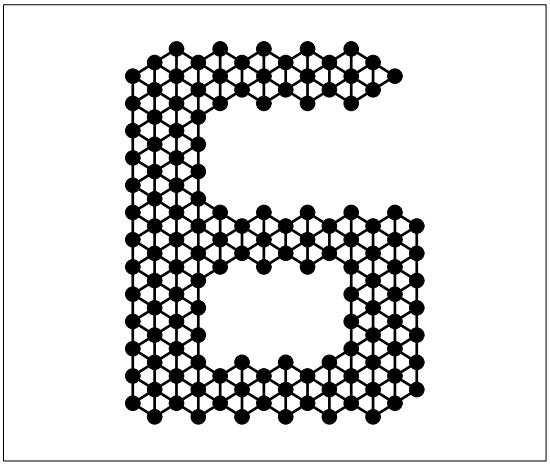

FIGURE 1. Example of a binary touch image on a two dimensional triangular lattice. The touched point is colored black. When the neighboring two points are both touched, they are connected by a black line. In our problem setting, for a touch image of an arbitrary shape, we would like to compute the Euler characteristics, #islands − #holes (= 1 − 1 = 0 for this example), as a topological invariant. Here we do not count local triangles as a hole.

B. COMPUTING THE EULER CHARACTERISTICS OF TOUCHES VIA POINCARE HOPF INDEX

Here we overview how to compute the Euler characteristics (= #islands − #holes) via Poincare-Hopf indices.

First we compute the Poincare-Hopf index for each lattice point as colored in Figure 2. It is +1 (red), −1 (blue) or 0 (otherwise), depending on the exit (or downward) patterns of connections as shown in Figure 3. 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. We will explain why only downward patterns matter for the index in the next subsection.

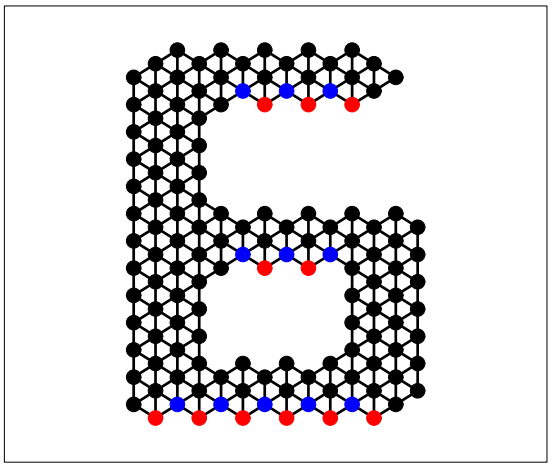

FIGURE 2. Example of Poincare-Hopf indices assigned on lattice points. The nonzero indices +1 and −1 were indicated by red and blue colors, respectively. The Euler characteristics for this touch image can be computed as the sum of all the indices (= #red points− #blue points = 11 − 11 = 0).

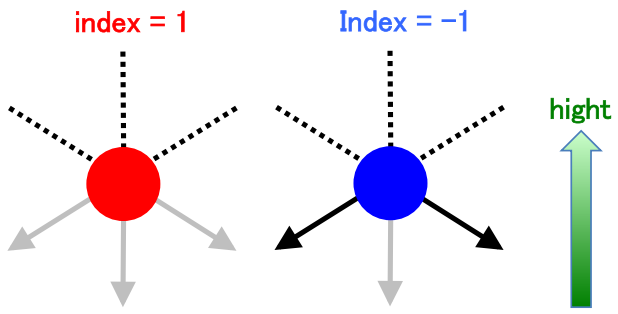

FIGURE 3. The rule for Poincare-Hopf index for each point. It is +1 (red), −1 (blue) or 0 (otherwise), depending on the downward patterns of connections. That is, a blue point is a point that is connected to only two side points out of three downward points, and a red point is not connected to any of three downward points.

Finally, the Euler characteristics is obtained as the sum of Poincare-Hopf indices for all the points. In the case of Figures 1 and 2, the Euler characteristics via Poincare-Hopf

index is given by

$$\chi = \#\text{red} - \#\text{blue} = 11 - 11 = 0. \quad (2)$$

Note that this is always equal to the "topological" Euler characteristics computed as

$$\chi = \#\text{islands} - \#\text{holes} = 1 - 1 = 0, \quad (3)$$

as we will explain in the next subsection. Figure 4 demonstrates that this algorithm also works for other examples.

C. MATHEMATICAL BACKGROUND OF POINCARE-HOPF INDEX TO RIGOROUSLY COMPUTE EULER CHRACTERISTICS

Here we show some backgrounds why the Poincare-Hopf indices rigorously compute the Euler characteristics.

Historically the Euler characteristics and the Poincare-Hopf index were defined in the continuous setting, say, for surfaces like a swim ring. Because triangulated surfaces are more tractable for computations on computers, a discretized version has been recently developed [26]. Although this discretized formulation reproduces the continuous Poincare-Hopf index for the case of a triangulated surface (which we specifically use), it actually works for a general graph (clique complex) for the purpose of computing the Euler characteristics. Note that, for a general graph, the Euler characteristics is combinatorially defined as the alternating sum of the number of k-cliques: $\chi = \sum_{k=0}^{\infty} \#k\text{-cliques}$.

Although the Poincare-Hopf indices can be computed for a given flow (vector field) on a surface, the specific case called the Morse index has been especially useful, where non-zero indices are assigned for the critical points of an arbitrary given height function (or a gradient flow). Therefore we simply considered y-axis as a height function when we specifically focused on a two-dimensional triangular lattice for the application purpose. To be precise, we wrote down and simplified the coloring rule (red or blue) for all possible downward cases in this specific lattice. Note that as we used the graph theoretical index based on the combinatorial formulation, it actually count only half of the critical points, computationally favorably.

To obtain the Poincare-Hopf index for this discretized setting, we simply need to count the number of downward destinations. In Figure 3, the red case has zero downward destination and the blue case has two downward destinations. Note that although any generic hight function works, we solely used y-axis itself as a hight for simplicity. Rigorously speaking, the number of destinations means the number of islands (or connected components) and, thus, the cases other than Figure 3, the destinations are all connected ($= 1$) as shown in Figure 5. (For a general graph, you should compute the Euler characteristics of the subgraph set of destinations instead of simply the number of destinations.)

Then, the Poincare-Hopf index at each point is defined as

$$\text{index} = 1 - \#\text{destinations}. \quad (4)$$

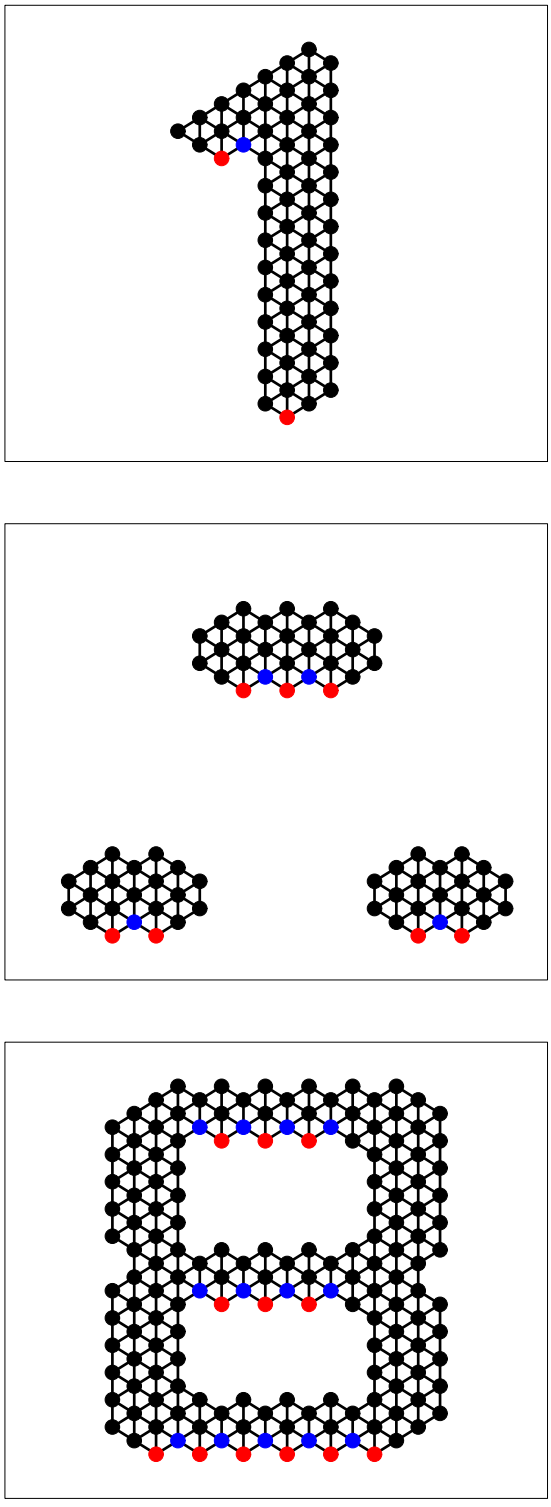

FIGURE 4. Additional examples of Poincare-Hopf indices. The nonzero indices +1 and −1 were indicated by red and blue colors, respectively. The Euler characteristics via the Poincare-Hopf index (=#red points − #blue points) is equal to the topological Euler characteristics (=#islands − #holes) for each case, that is, 1, 3, and −1, respectively.

Therefore it is 1 for the red point, −1 for the blue point, and 0 otherwise. As is evident in Figures 2 and 4, the index detects the saddle-like points (blue) and the minimum (red) and they

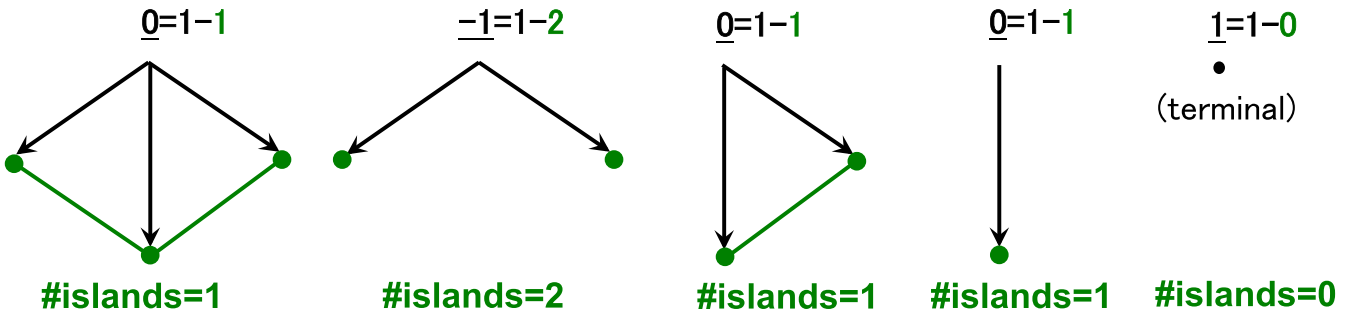

FIGURE 5. The rule for Poincare-Hopf index (=1− #destinations) for all eight patterns of downward connections. As there are three downward edges, the existence or absence of each edge results in eight patterns in total. However, to save space, the symmetric and equivalent three patterns were omitted. The green dots and edges represent the destinations called exit sets. The green integers denote the numbers of isolated destinations. The underlined integers denote the Poincare-Hopf indices for each top point. Note that the number of destinations is one and thus the index is zero except for the two patterns shown in Figure 3.

correctly amount to the Euler characteristics (=#islands − #holes), at least for the given examples. And 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 (or maxima) 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 [26]. Note that, again, although the conventional Morse theory count both the minima and maxima, here we count only minima from the symmetry to save computational resources.

D. A CONCRETE ALGORITHM FOR IMPLEMENTATION

Here we show a concrete algorithm to compute the Poincare Hopf index for each point in a two dimensional binary image. Often, the image is given in the form of a square lattice. So here we assume a square image represented by $u_{[i,j]}$. In that case, you can downsample the lattice points in order to obtain a triangular lattice as in Figure 6. To be specific, you can only consider the points where $i+j$ is even or let $u_{[i,j]} = 0$ if $i+j$ odd. Notice that in the hardware implementation we used logical operations instead of "if then" branch to gain speed.

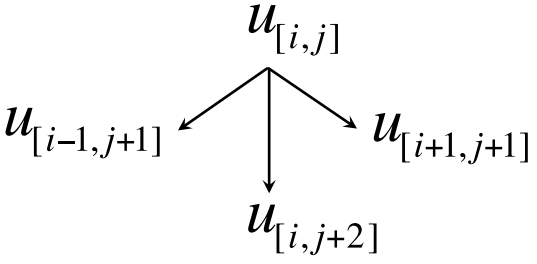

FIGURE 6. Notation to derive a logical operation for Poincare-Hopf index for downsampled case. As we downsample from a square lattice $u_{[i,j]}$ to obtain a triangular lattice, we only consider the points where $i+j$ is even and let $u_{[i,j]} = 0$ for $i+j$ odd.

Then by using the notation in Figure 6, the rule to compute the Poincare-Hopf index in Figure 3 at the point $(x, y) = (i, j)$ can be described as

$$m_{[i,j]} = u_{[i,j]}(1-u_{[i-1,j+1]})(1-u_{[i,j+2]})(1-u_{[i+1,j+1]}) - u_{[i,j]}u_{[i-1,j+1]}(1-u_{[i,j+2]})u_{[i+1,j+1]}. \quad (5)$$

Note that $m_{[i,j]}$ can be nonzero only when either of the above two terms is nonzero, corresponding to the two cases in Figure 3. That is, the first term of the right-hand side of the equation corresponds to the red point while the second term to the blue point. Again, the Euler characteristics of the image is the sum of the Poincare-Hopf indices of all points.

E. DEMONSTRATION BY ARDUINO IMPLEMENTATION

In order to demonstrate the proposed algorithm, we concretely implemented it on a microcontroller board, Arduino (Figure 7), with a TFT touchscreen, which computes the Euler characteristics of touches automatically and updates it in real time.

We successfully computed the Euler characteristics (= #islands − #holes) for the inputs by finger. For example, the top picture in Figure 4 has two islands and no hole, resulting in $\chi = 2-0 = 2$. The next picture has one island and one hole, resulting in $\chi = 1-1 = 0$. The next picture has one island and two holes, resulting in $\chi = 1-2 = -1$. The bottom picture has one island and three holes, resulting in $\chi = 1-3 = -2$. Thus our implementation correctly updated the Euler characteristics in real time when we continued to add points directly on the touch screen.

To achieve the speed for interactive responsiveness, the implementation was designed as logical operations in an Arduino programming language like Wiring. The specification of our implementation is as follows: Arduino Uno, a microcontroller board equipped with the AVR 8bit microcontroller, a 16MHz quartz crystal oscillator, and a 32KB Flash memory, and the Adafruit 2.8″ TFT Touch Shield with a touchscreen display (240×320 pixels) and capacitive touch functionality.

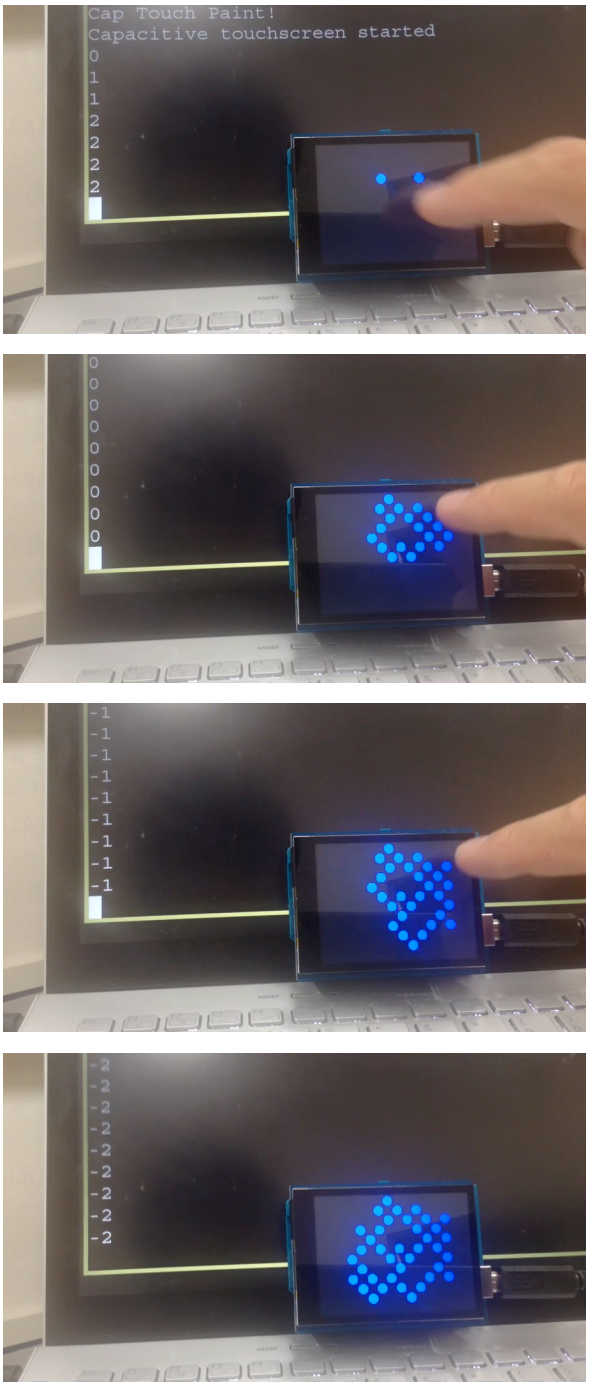

FIGURE 7. Demonstration by real-time Arduino implementation of proposed algorithm. While the sensors keep all the past touched points, the monitor indicates the Euler characteristics (= #islands − #holes) for the touch image displayed on the touchscreen. Note the monitor updates the number whenever a new island or a hole appears. For example, the top picture has two islands and no hole, resulting in $\chi = 2 - 0 = 2$. The second picture has one island and one hole, resulting in $\chi = 1 - 1 = 0$. The third picture has one island and two holes, resulting in $\chi = 1 - 2 = -1$. The bottom picture has one island and three holes, resulting in $\chi = 1 - 3 = -2$.

III. SUMMARY AND DISCUSSIONS

In this paper, we proposed a real-time algorithm to compute the Euler characteristics of a binary touch image via Poincare-Hopf indices. Validated by topology, the proposed algorithm accurately counts the number of islands in a touch image or the number of holes in a touch, irrespectively of the shapes and positions of the touches. The key idea is to compute the Poincare-Hopf indices for a few active points and sum them to obtain the Euler characteristics. We demonstrate that our implementation in Arduino, in the form of solely local logical operations, returns and updates the correct topological invariants of touches in real time. Thus the advantage of our implementation resides in the real-time computation supported by the sparse Poincare-Hopf indices, which enabled interactive applications with touchscreens. The proposed algorithm of the logical form representation can be implemented in any circuits such as FPGAs and is never specialized to our Arduino codes with a microcomputer board.

Regarding the computational speed, the algorithm proposed in the current paper is much faster than our previous implementation [25]. 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.

In addition to the speed, there are possible extensions that truly take an advantage of the Poincare-Hopf index. First, you can improve the real-time algorithm by utilizing the locality of the index. For example, the Poincare-Hopf index needs to be updated only when the surrounding pixels have changed. Second, the sensor network can be easily extended to any complicated shape in three dimensional space. Actually the algorithm extends to general graphs (clique complexes) rather straightforwardly.

REFERENCES

[1] A. Nieder, D. J. Freedman, and E. K. Miller, "Representation of the quantity of visual items in the primate prefrontal cortex," *Science*, vol. 297, no. 5587, pp. 1708–1711, Sep. 2002.

[2] B. Butterworth, *What Counts: How Every Brain is Hardwired for Math*. New York, NY, USA: Free Press, 1999.

[3] K. Devlin, *The Math Gene: How Mathematical Thinking Evolved and Why Numbers are Like Gossip*. New York, NY, USA: Basic Books, 2001.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer-Verlag, 2006.
[5] D. C. Ciresan, 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.
[6] M. Ishikawa, "Learning of modular structured networks," *Artif. Intell.*, vol. 75, no. 1, pp. 51–62, May 1995.
[7] M. Ishikawa, "Structural learning with forgetting," *Neural Netw.*, vol. 9, no. 3, pp. 509–521, Apr. 1996.
[8] W. Fulton, *Algebraic Topology: A First Course*. New York, NY, USA: Springer-Verlag, 1995.
[9] Y. Matsumoto, *An Introduction to Morse Theory*. Providence, RI, USA: AMS, 2001.
[10] R. Ghrist, *Elementary Applied Topology*. North Charleston, SC, USA: CreateSpace, 2014.
[11] 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.
[12] Y. Baryshnikov and R. Ghrist, "Target enumeration via Euler characteristic integrals," *SIAM J. Appl. Math.*, vol. 70, no. 3, pp. 825–844, 2009.
[13] Y. Baryshnikov and R. Ghrist, "Euler integration over definable functions," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 21, pp. 9525–9530, 2010.
[14] J. Curry, R. Ghrist, and M. Robinson, "Euler calculus with applications to signals and sensing," in *Proc. Symp. Appl. Math.*, vol. 70, pp. 75–146, 2012.
[15] 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.
[16] A. Hirata *et al.*, "Geometric frustration of icosahedron in metallic glasses," *Science*, vol. 341, no. 6144, pp. 376–379, 2013.
[17] K. Miura and T. Aoki, "Hodge–Kodaira decomposition of evolving neural networks," *Neural Netw.*, vol. 62, pp. 20–24, Feb. 2015.
[18] K. Miura and T. Aoki, "Scaling of Hodge–Kodaira decomposition distinguishes learning rules of neural networks," in *Proc. IFAC CHAOS*, 2015, pp. 175–180.
[19] H. Edelsbrunner and J. L. Harer, *Computational Topology: An Introduction*. Providence, RI, USA: AMS, 2009.
[20] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*. New York, NY, USA: Springer-Verlag, 2010.
[21] 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.
[22] 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.
[23] 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.
[24] 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, Aug. 2011.
[25] 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.
[26] O. Knill. (2012). "A graph theoretical Poincare–Hopf theorem." [Online]. Available: http://arxiv.org/abs/1201.1162

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. He is currently 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. 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.