

Neural implementation of shape-invariant touch counter based on Euler calculus

Keiji Miura, Tohoku Univ.

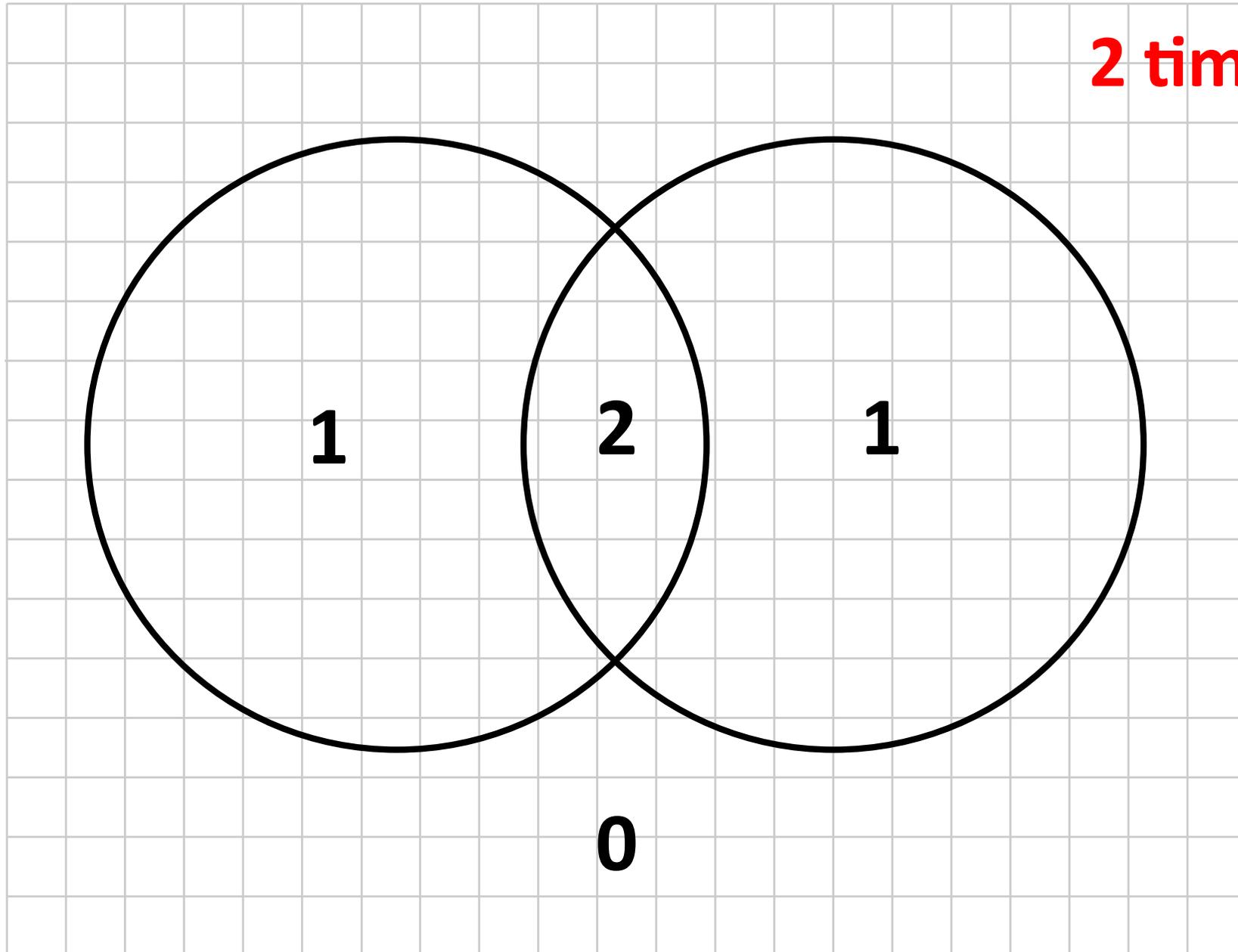
Kazuki Nakada, Univ. of Electro-Communications

Topics in Differential Geometry and its Discretizations

Jan 10, 2015

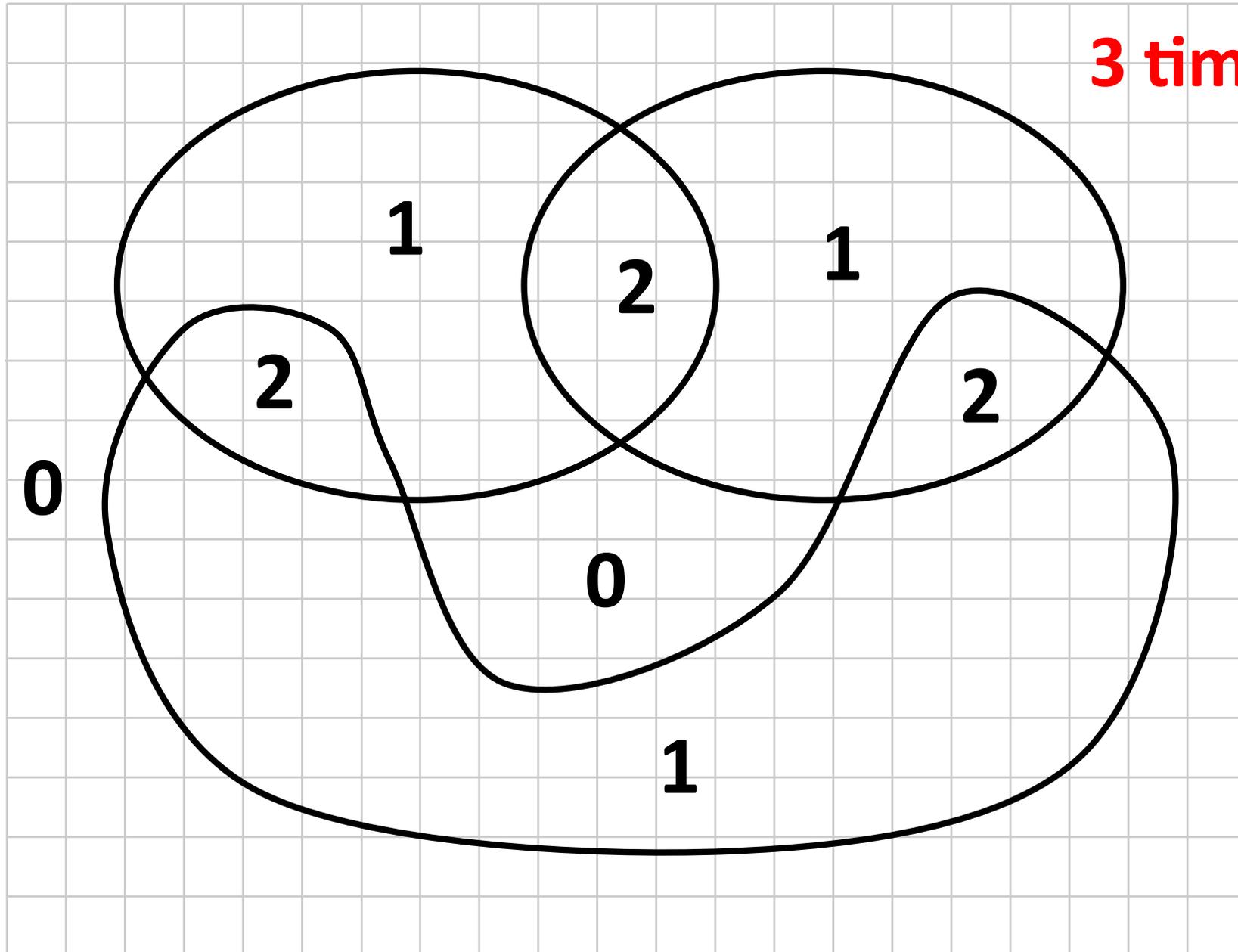
Goal : shape-invariant touch count

(No time resolution)



Goal : shape-invariant touch count

(No time resolution)



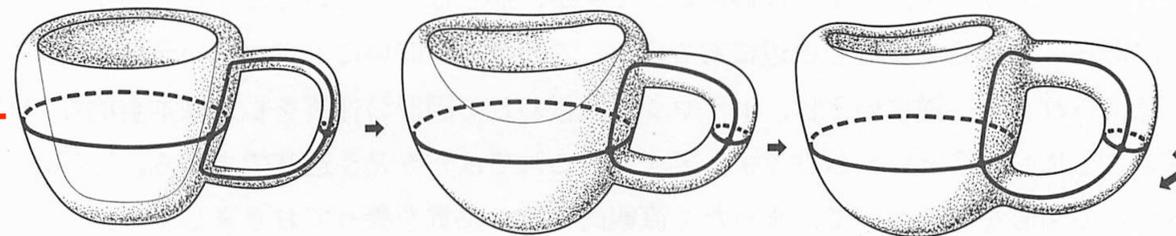
Goal : touch count

1. Invariance to finger shapes
 - Topology (Euler Calculus, Ghrist group)
2. Speed up
 - Parallelization with neural networks

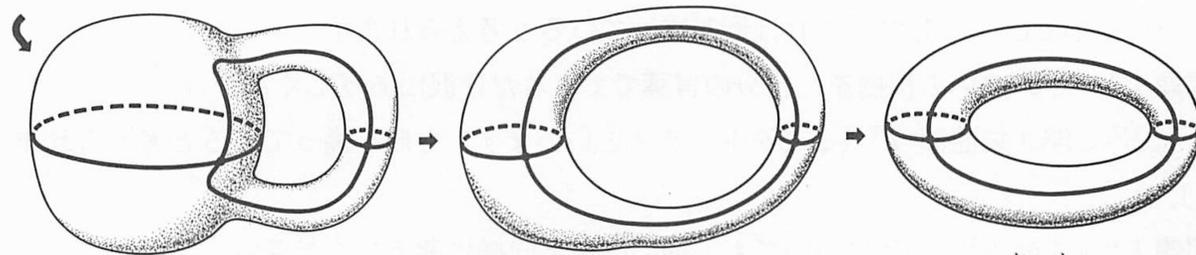
Topology gives invariants under morphing

- β_0 = #connected components (in a binary image)
- β_1 = #holes
- $\chi = \beta_0 - \beta_1$

$$\begin{aligned}\beta_0 &= 1 \\ \beta_1 &= 1 \\ \chi &= 1 - 1 = 0\end{aligned}$$



Coffee cup



doughnuts

$$\begin{aligned}\beta_0 &= 1 \\ \beta_1 &= 1 \\ \chi &= 1 - 1 = 0\end{aligned}$$

$$\beta_0(\text{[Diagram 1]}) = 1$$

The diagram shows a rectangular box containing a single large oval. Inside the oval is the number '1'. To the right of the oval, outside the box, is the number '0'.

$$\beta_0(\text{[Diagram 2]}) = 1$$

The diagram shows a rectangular box containing a large oval. Inside this large oval is a smaller oval. The number '1' is located to the left of the inner oval, and the number '0' is inside the inner oval. To the right of the large oval, outside the box, is the number '0'.

$$\beta_0(\text{[Diagram 3]}) = 3$$

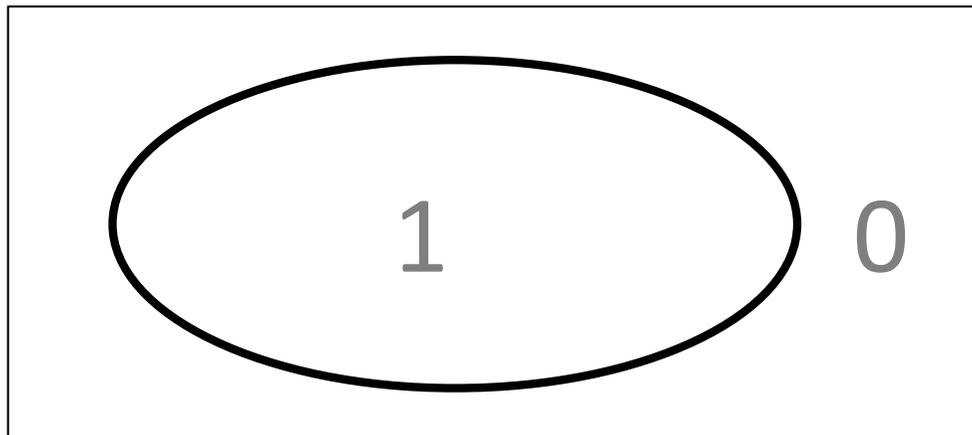
The diagram shows a rectangular box containing three separate ovals. Two ovals are positioned at the bottom, each containing the number '1'. A third oval is positioned at the top center, containing the number '1'. To the right of the top oval, outside the box, is the number '0'.

$$\beta_1 \left(\begin{array}{c} \text{[Diagram: A large oval containing the number 1, with the number 0 to its right.]} \end{array} \right) = 0$$

$$\beta_1 \left(\begin{array}{c} \text{[Diagram: A large oval containing a smaller oval containing the number 0, with the number 1 to the left of the inner oval and the number 0 to the right of the outer oval.]} \end{array} \right) = 1$$

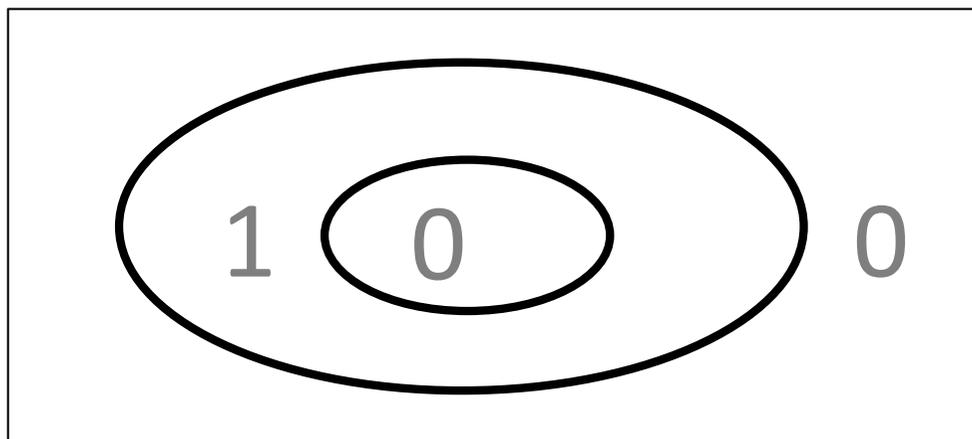
$$\beta_1 \left(\begin{array}{c} \text{[Diagram: Three separate ovals, each containing the number 1, arranged in a triangle. The number 0 is to the right of the top oval.]} \end{array} \right) = 0$$

$\chi($



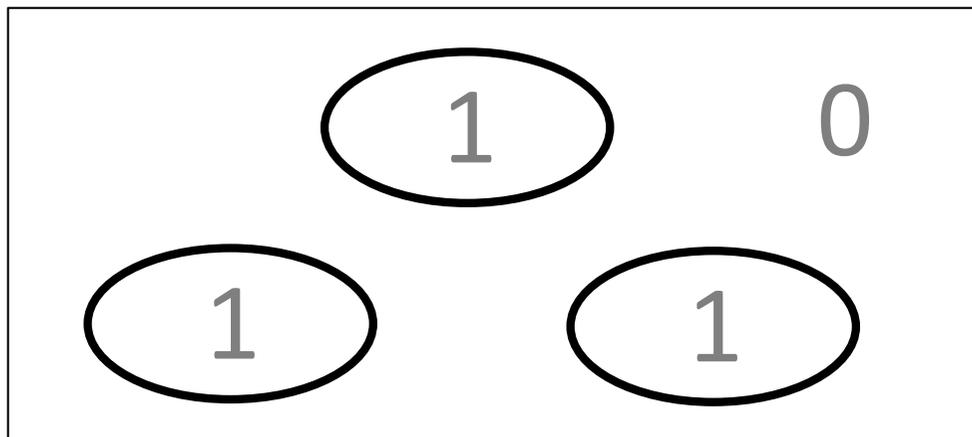
$)=1$

$\chi($



$)=0$

$\chi($



$)=3$

Euler Integral for multivalued image

(Ghrist, 2009-)

Generalized integral:

- One touch: $\int(f_1)dx = \chi(\boxed{\textcircled{1}_0}) = 1$

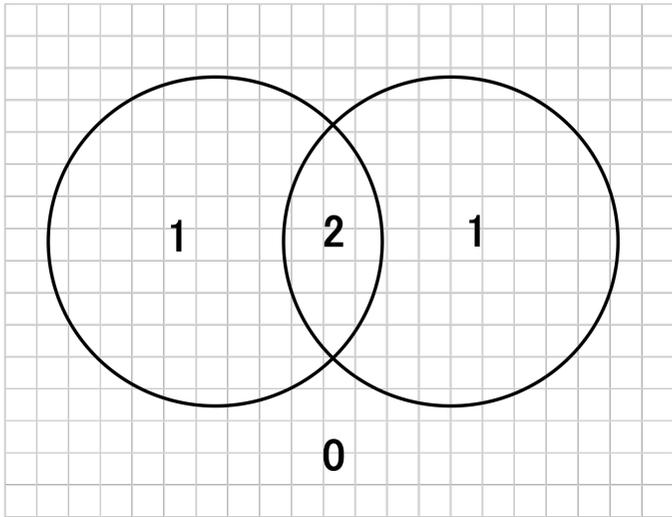
cf. $\chi_{\text{binary image}} := (\#\text{connected components}) - (\#\text{holes})$

- More touches: the integral is defined by a sum of indicator functions (binary images):

$$\int(\sum f_i)dx = \sum(\int f_i dx) = \text{Touch number!}$$

$$\chi(\boxed{\text{1 2 1}_0}) = 2\chi(\boxed{\textcircled{1}_0}) = 2$$

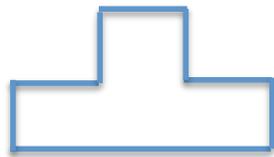
Example 1



$$\equiv \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{matrix} \text{[Yellow Rectangle]} & \text{Height} > 0 \\ + \\ \text{[Yellow Square]} & \text{Height} > 1 \end{matrix}$$

Automation with level sets



Decompose into
binary indicator
functions



Level sets



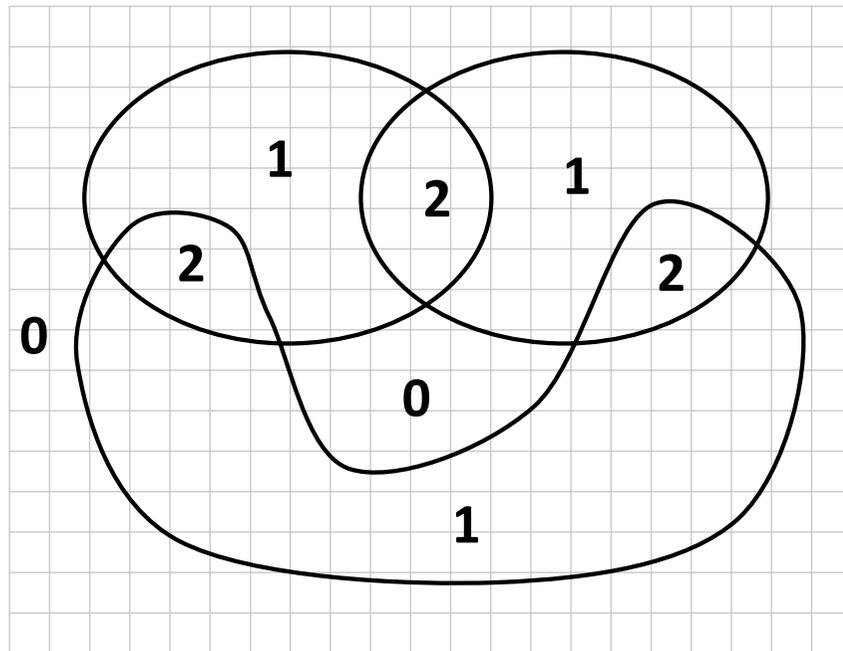
$$\begin{aligned}
 \chi\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) &= \chi\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) + \chi\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) \\
 &= \beta_0\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) - \beta_0\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}\right) + 1 \\
 &+ \beta_0\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) - \beta_0\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}\right) + 1 \\
 &= 1 - 1 + 1 + \underline{1 - 1 + 1} + 0 + 0 + \dots \\
 &= 2.
 \end{aligned}$$

Formula:

Alexander Duality

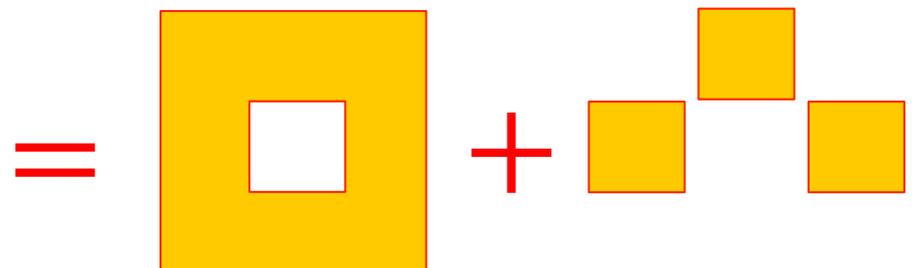
$$\chi(h) = \sum_{s=0}^{\infty} \chi(h > s) = \sum_{s=0}^{\infty} [\beta_0(h > s) - \beta_0(h \leq s) + 1]$$

Example 2



\equiv

0	0	0	0	0
0	1	2	1	0
0	2	0	2	0
0	1	1	1	0
0	0	0	0	0



Circuit Simulation on Matlab/Simulink

$$\begin{aligned}
 \chi(h) &= \chi(h > 0) + \chi(h > 1) + 0 + 0 + \dots \\
 &= [\beta_0(h > 0) - \beta_1(h > 0)] + [\beta_0(h > 1) - \beta_1(h > 1)] \\
 &= [\beta_0(h > 0) - \beta_0(h \leq 0) + 1] + [\beta_0(h > 1) - \beta_0(h \leq 1) + 1] \quad (\text{Alexander Duality}) \\
 &= [1 - 2 + 1] + [3 - 1 + 1] = 3
 \end{aligned}$$

Upper-level module that calls β_0 :

a

$$\beta_0(h > 0) = \beta_0 \left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right) = 1$$

b

$$\beta_0(h \leq 0) = \beta_0 \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \right) = 2$$

c

$$\beta_0(h > 1) = \beta_0 \left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right) = 3$$

d

$$\beta_0(h \leq 1) = \beta_0 \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \right) = 1$$

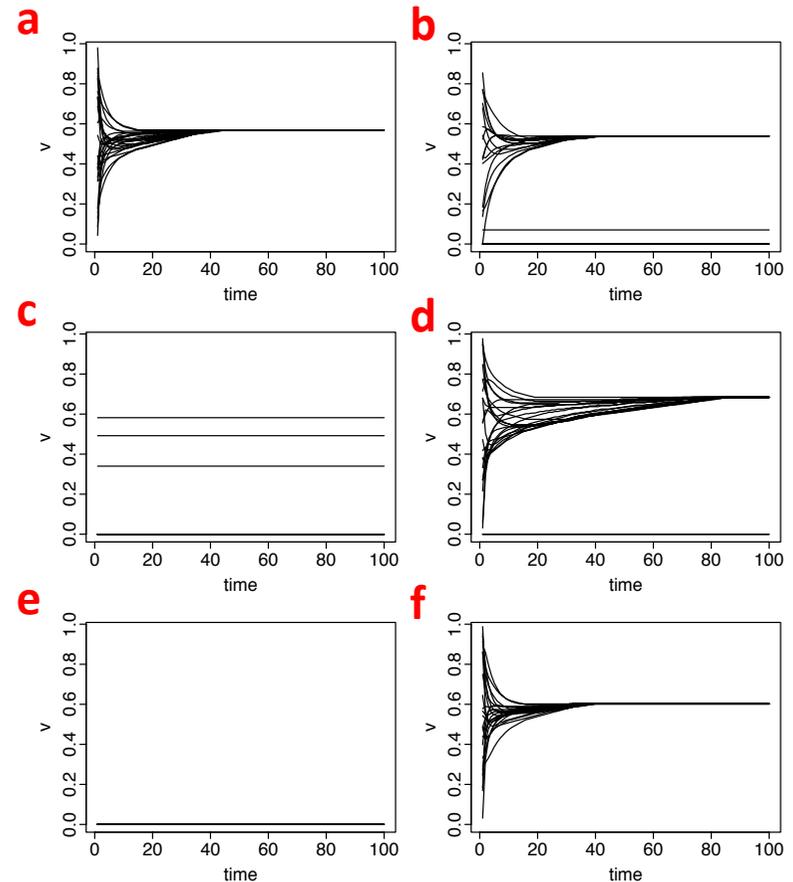
e

$$\beta_0(h > 2) = \beta_0 \left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right) = 0$$

f

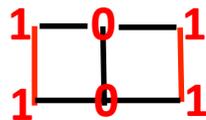
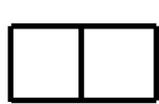
$$\beta_0(h \leq 2) = \beta_0 \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \right) = 1$$

Inside β_0 : (recurrent neural network)



Algorithm to compute β_0 (binary image)

Example with 2x3 pixels:



$$\Delta_{ex} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

▪ pixel values (0 or 1)
determines edges
(Hebbian rule)

▪ Matrix representation of a graph
▪ -#edge on a node on diagonal



$$\Delta v = 0$$



$$v_0 = \begin{pmatrix} a \\ a \\ b \\ b \end{pmatrix}$$



$$\beta_0 = 2$$

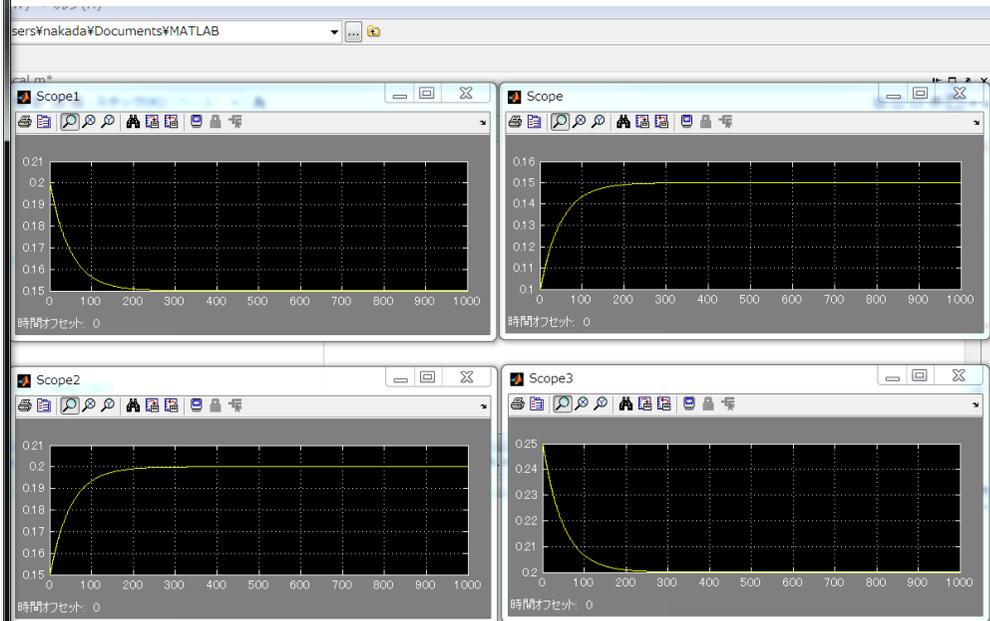
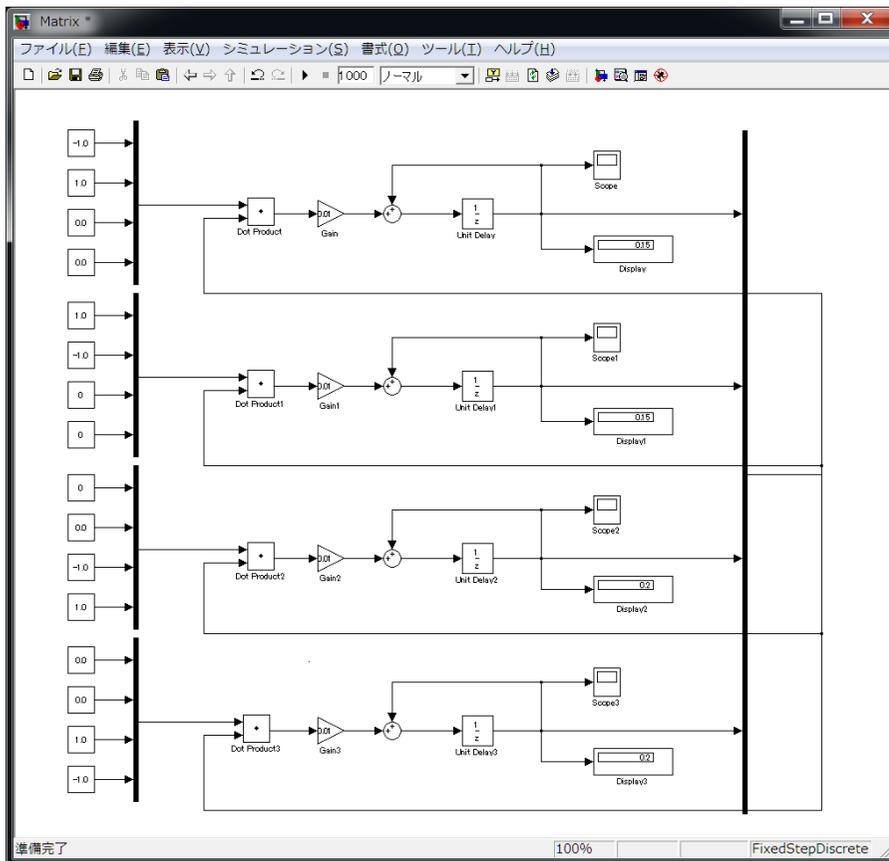
Parallelizable!
(Next slide)

Parallelized Implementation for β_0

$$\Delta_{ex} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \Rightarrow \boxed{(I + \alpha\Delta)^n v_0} \Rightarrow v_{final} = \begin{pmatrix} a \\ a \\ b \\ b \end{pmatrix}$$

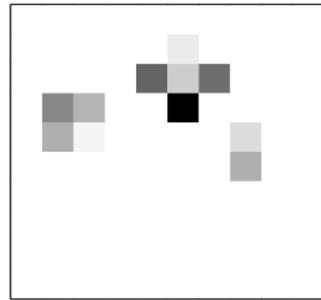
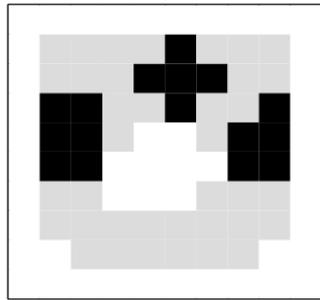
Parallelized!

Neurons interact and average their states.

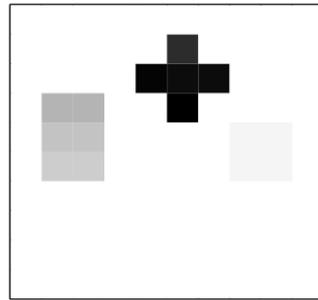


MATLAB HDL Coder \rightarrow HDL \rightarrow FPGA

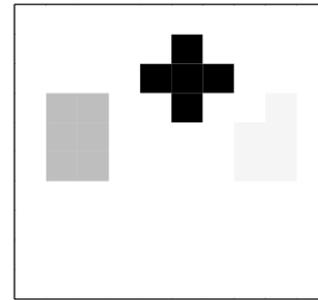
A larger example (10 x 10)



t=0

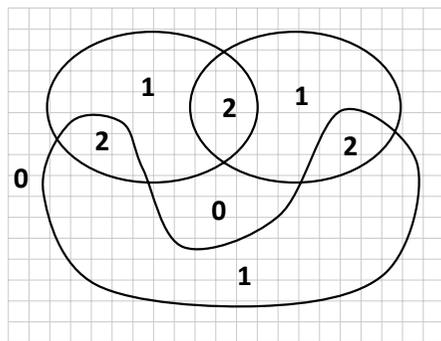


t=20

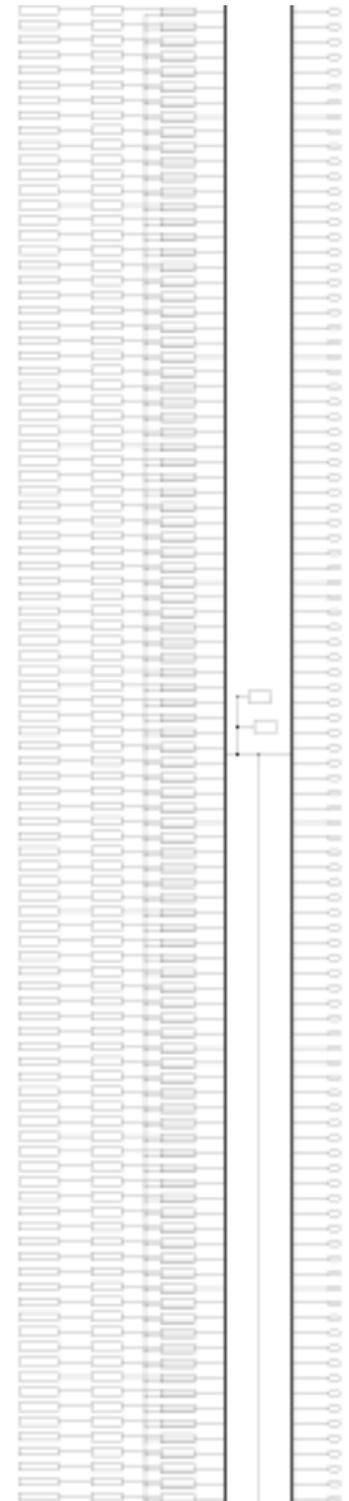
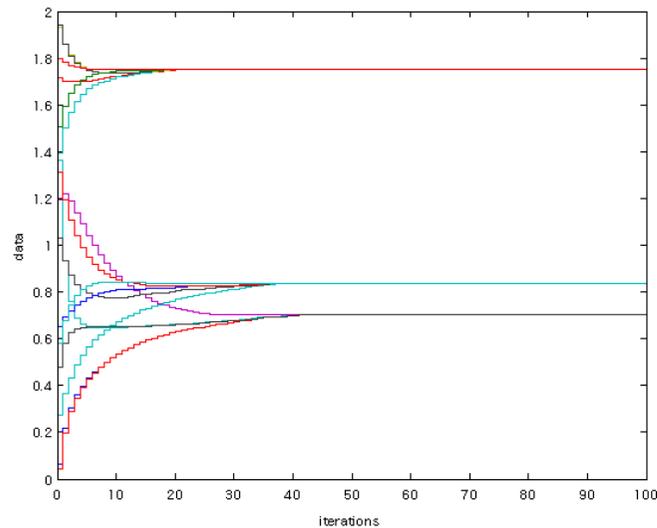


t=40

$\cdot || \cdot$



Level set: h=2



Summary

- We propose a parallelized algorithm of shape-invariant touch counter.
 - We use Euler Calculus, a topological method, to realize shape and position invariance.
 - To accelerate, we parallelized the recursive computation of connected component counters, which are elemental sub-modules.

Discussion

- Useful when no time resolution
- The matrix is sparse: $O(4N)$
- Iterative computation applicable to general β_i
- Integral is unique as it satisfies additivity axiom of measure: $\chi(A \cup B) = \chi(A) + \chi(B) - \chi(A \cap B)$
- Larger problems and parameter tunings for the future works.