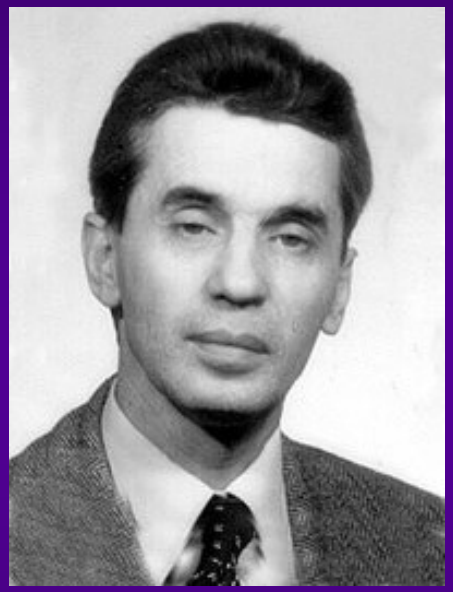
Overview of the Cellular Neural Networks (CNN)

Hyongsuk

- Visiting Research Scholar, UC Berkeley
- Professor, Chonbuk National University
- Email: jbnu.ac.kr

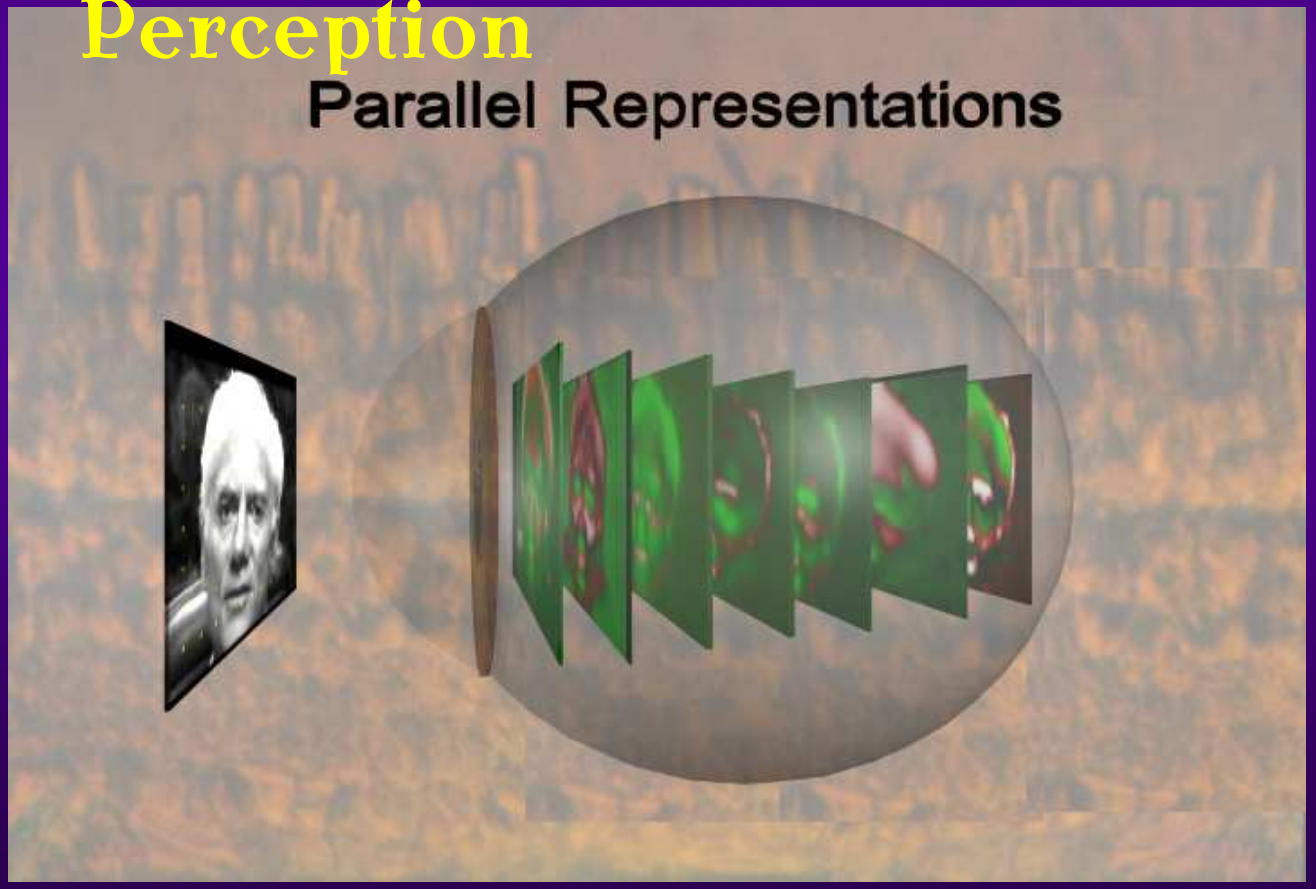
Leon O. Chua : University of California, Berkeley
Tamas Roska: Hungarian Academy of Science



- L. O. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Tr. on Circuits Systems*, vol.35, pp. 1257-1272, 1988.
- L. O. Chua and L. Yang, "Cellular neural networks: applications," *IEEE Tr. on Circuits Systems*, vol. 35, pp. 1273-1290, 1988.
- T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer", *IEEE Tr. on Circuits Systems II*, CAS-40, pp. 163-173, 1993.

Analogic Parallel Processor (CNN)

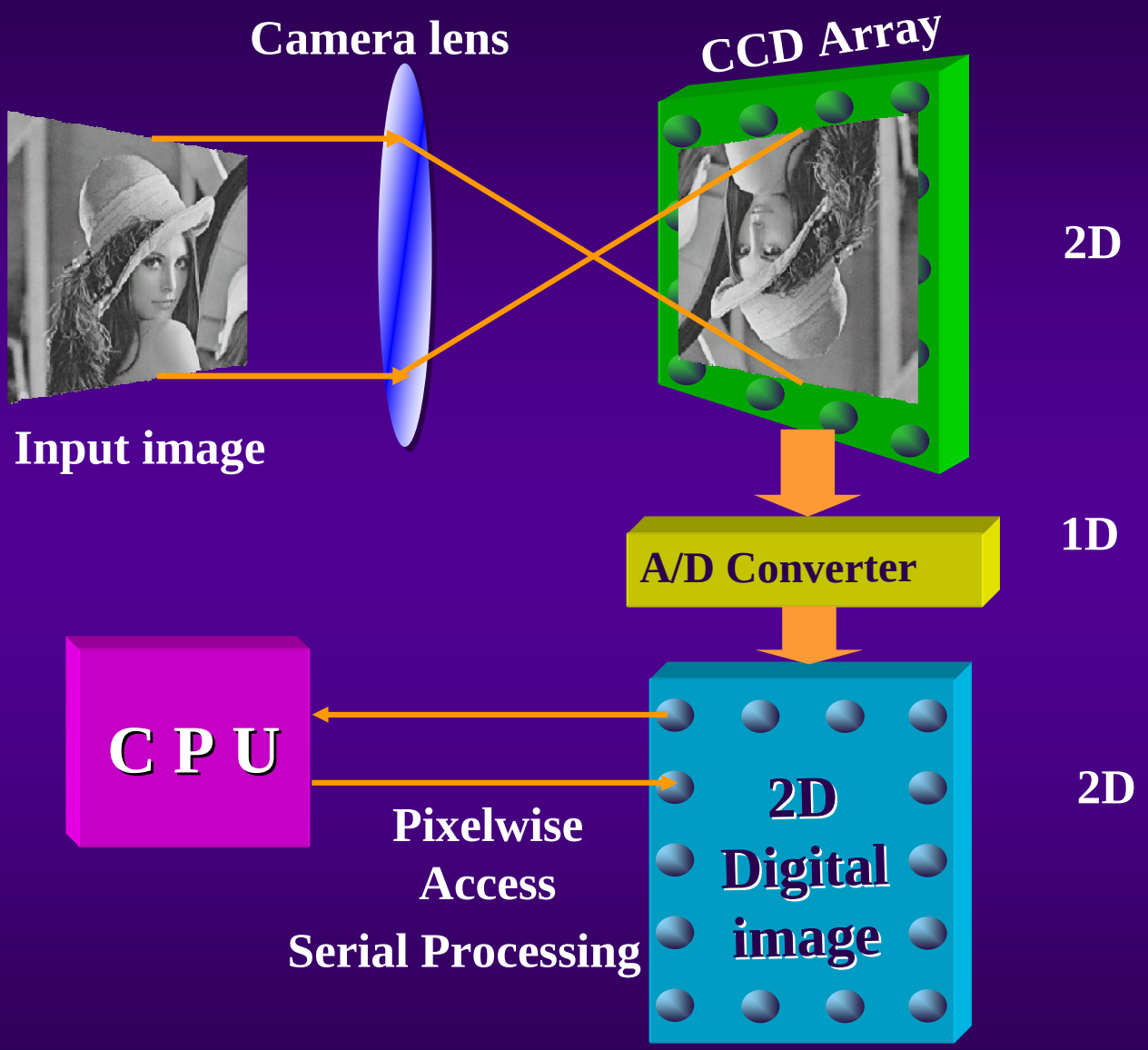
Human Visual Perception



In Retina

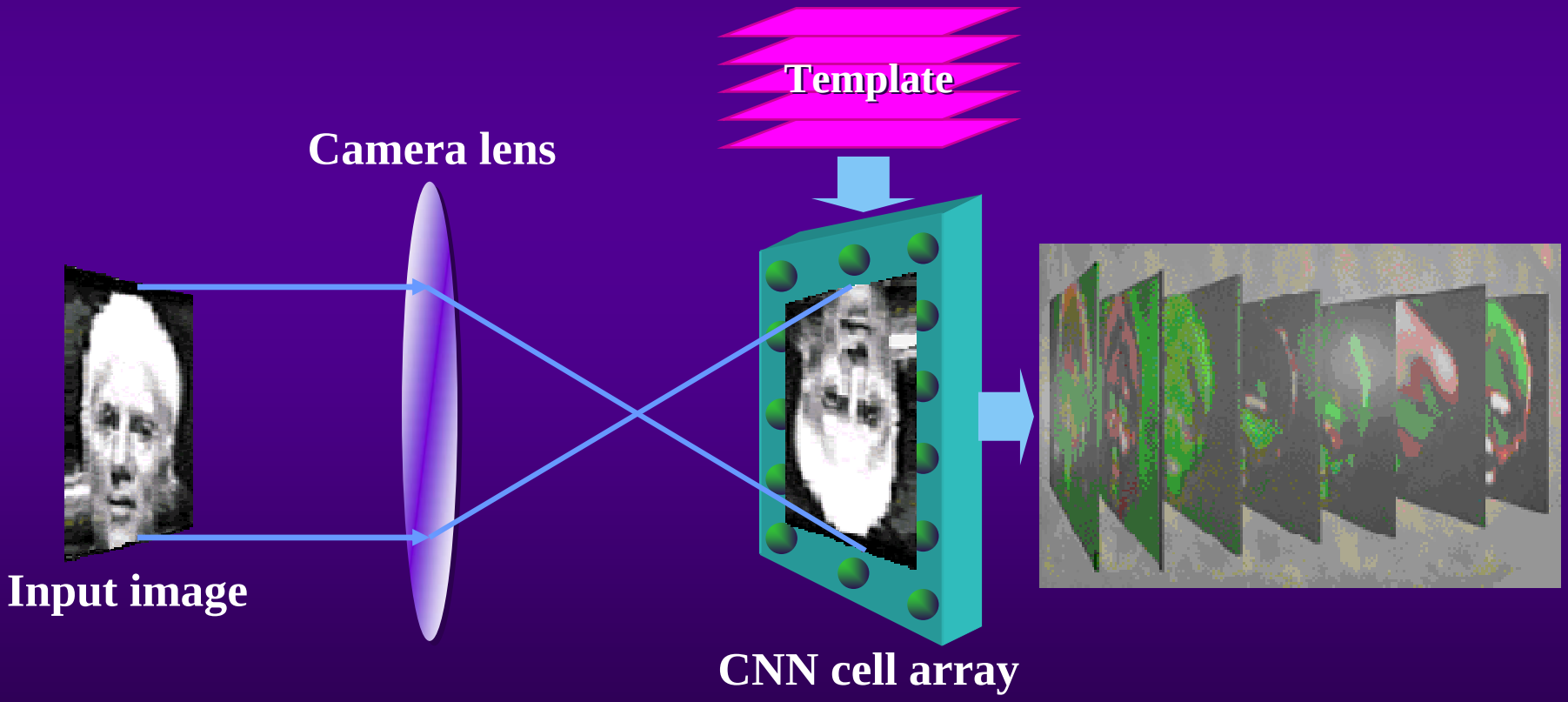
Horizontal Cells
Bipolar Cells
Amacrine Cells
Ganglion Cells

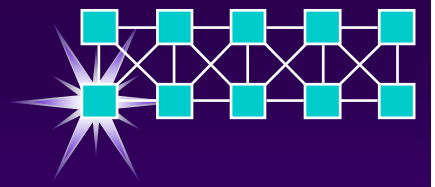
Digital Image Processing



* Conventional digital computer is inadequate for image processing

An Engineering Model of Bionic Eye, The CNN





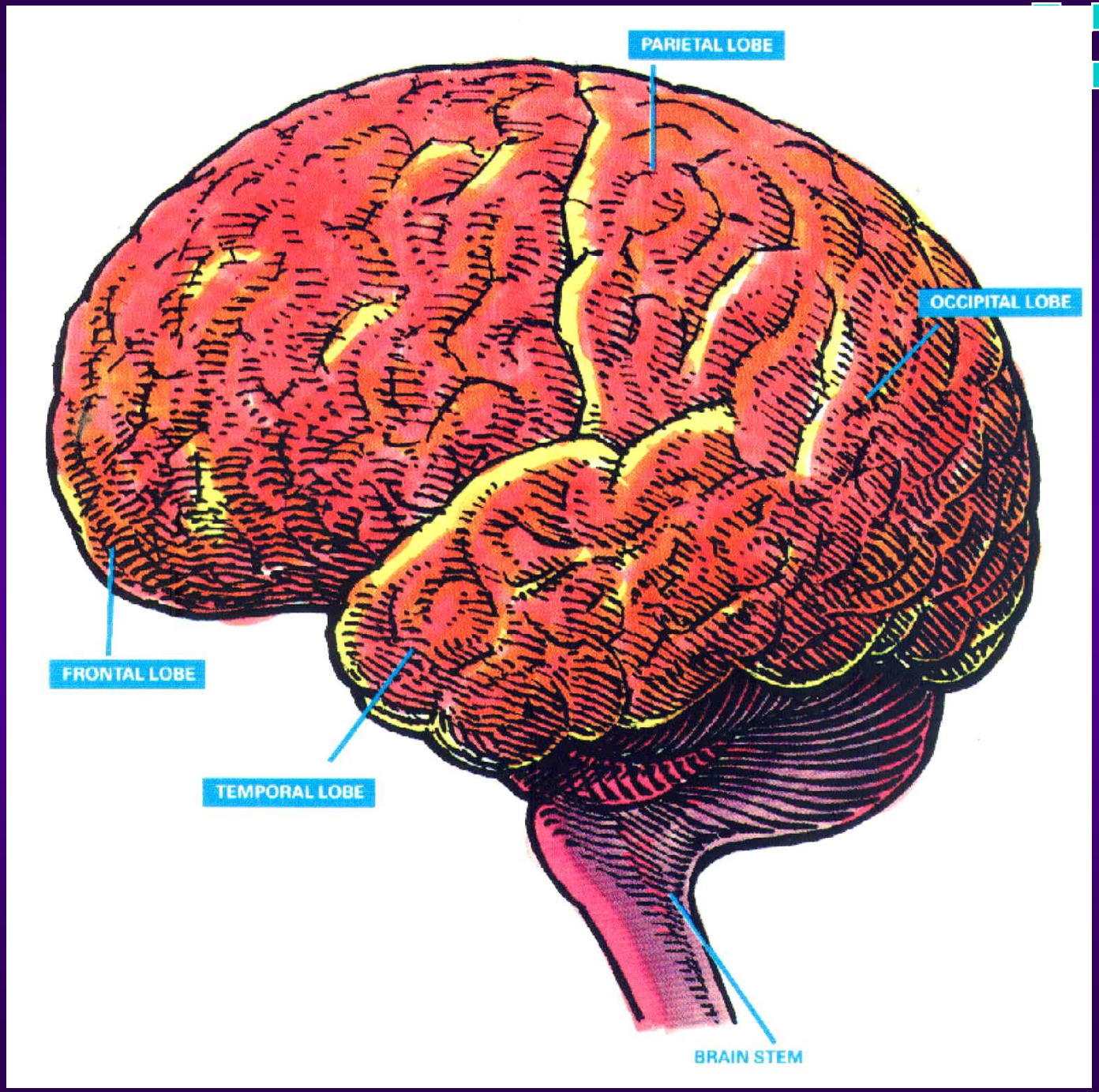
The **CNN**

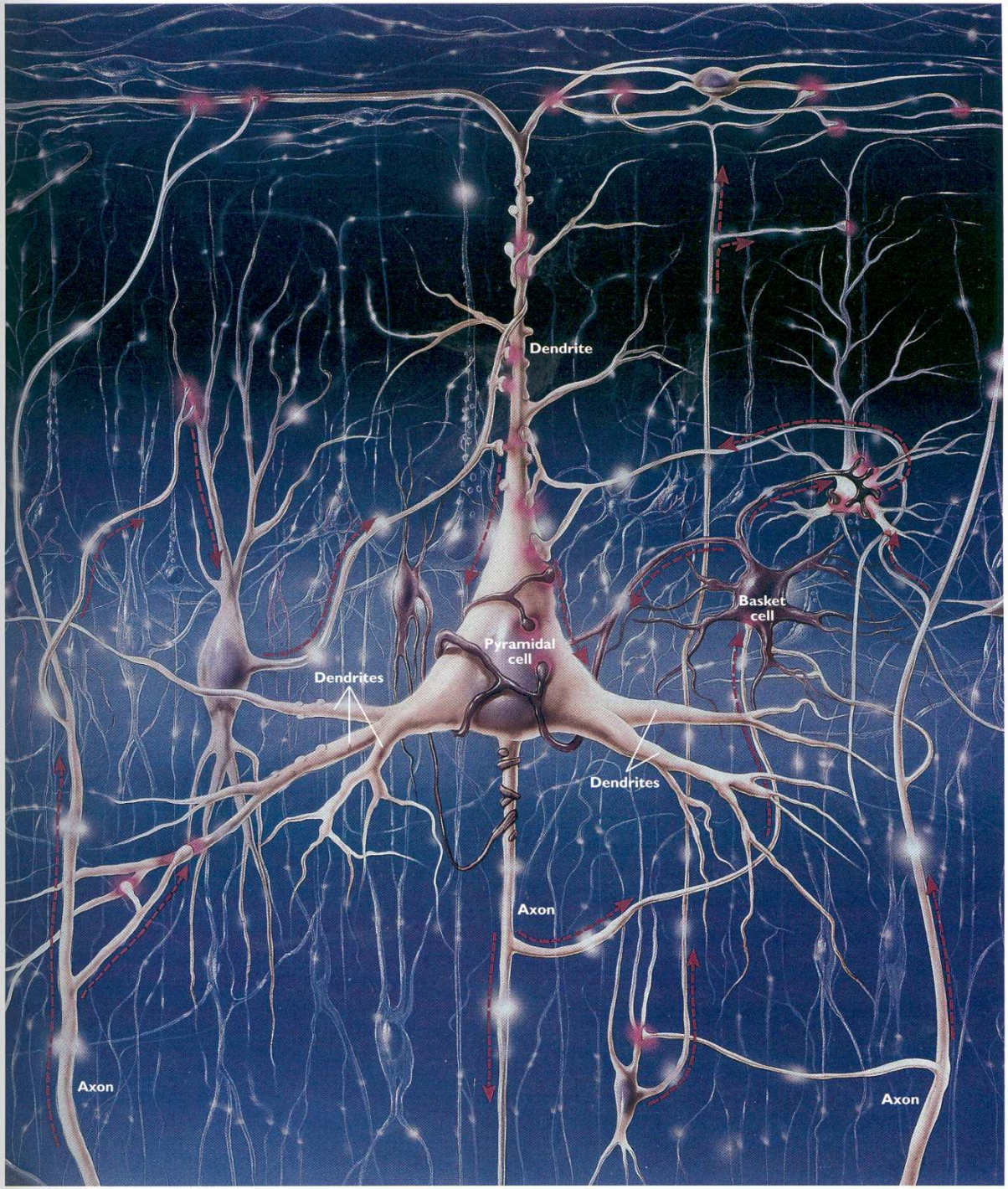
Architecture

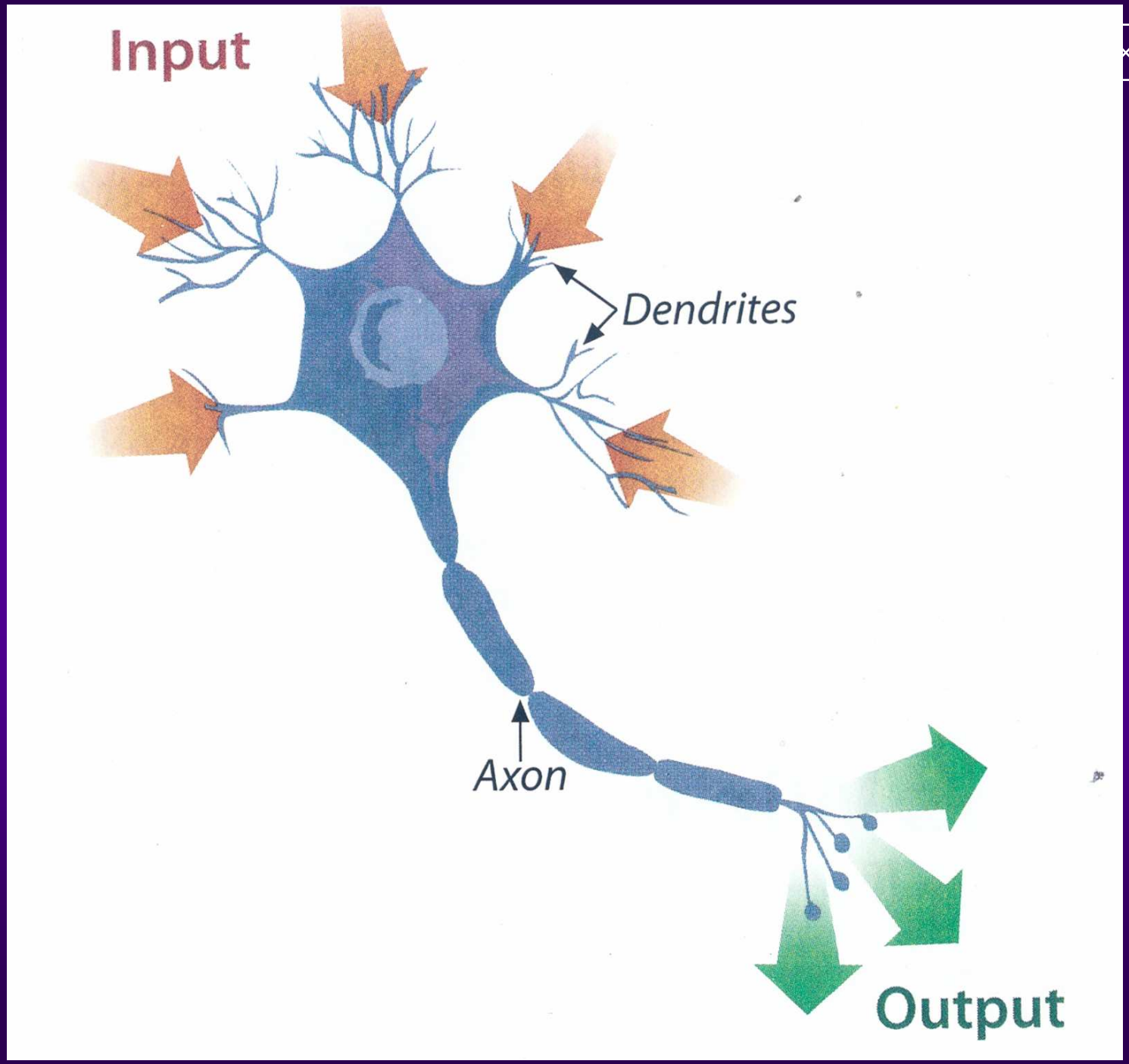
is similar to the

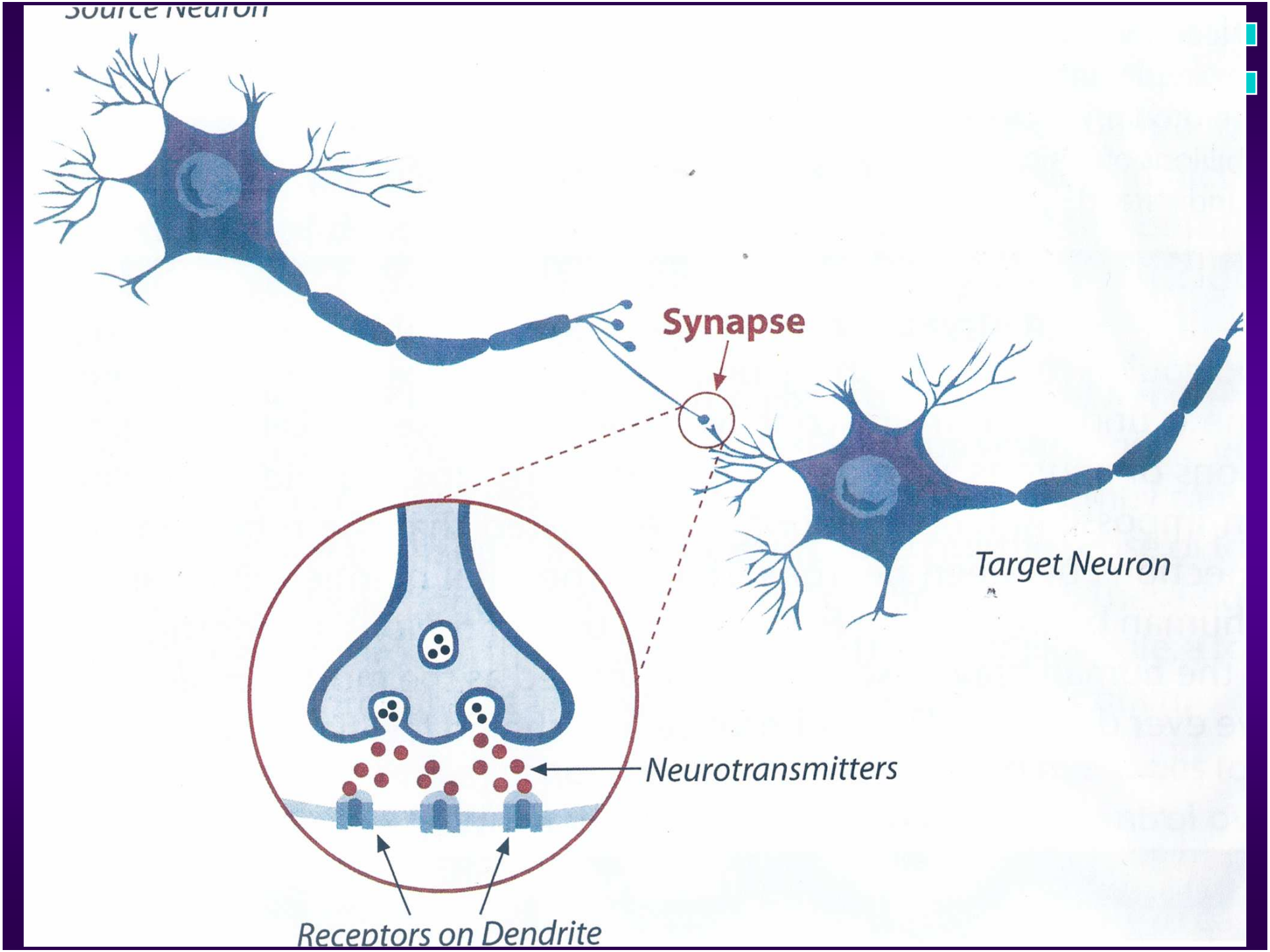
Brain Synaptic

Architecture

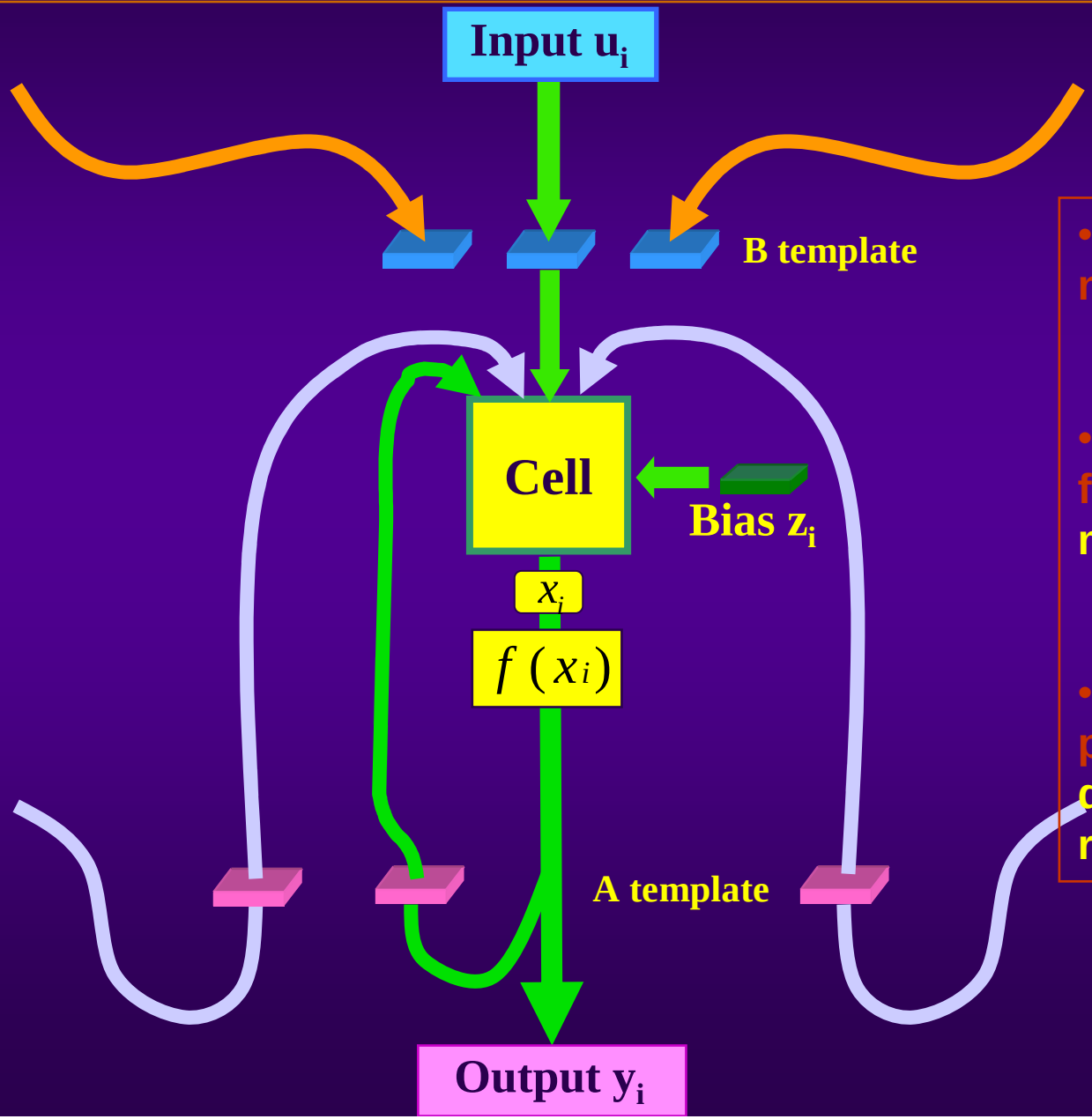

PARIETAL LOBE
OCCIPITAL LOBE
FRONTAL LOBE
TEMPORAL LOBE
BRAIN STEM


Dendrite
Basket cell
Pyramidal cell
Dendrites
Dendrites
Axon
Axon
Axon


Input
Dendrites
Axon
Output


Source Neuron
Synapse
Target Neuron
Neurotransmitters
Receptors on Dendrite

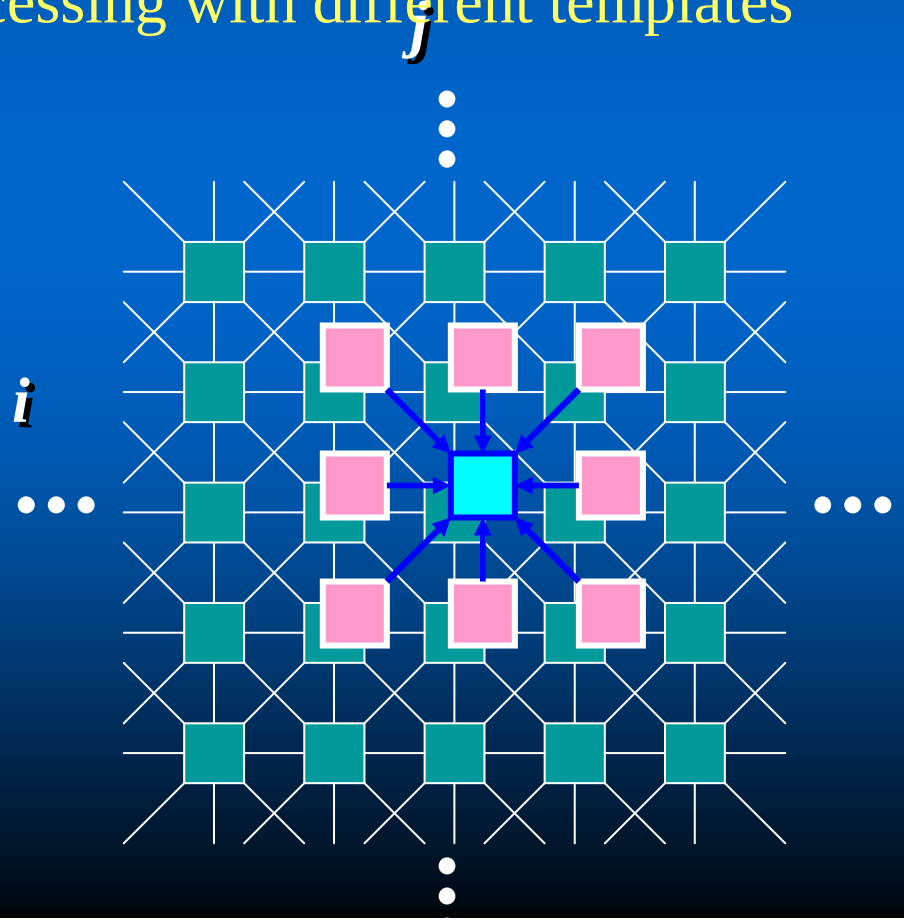
1D Structure of the CNN



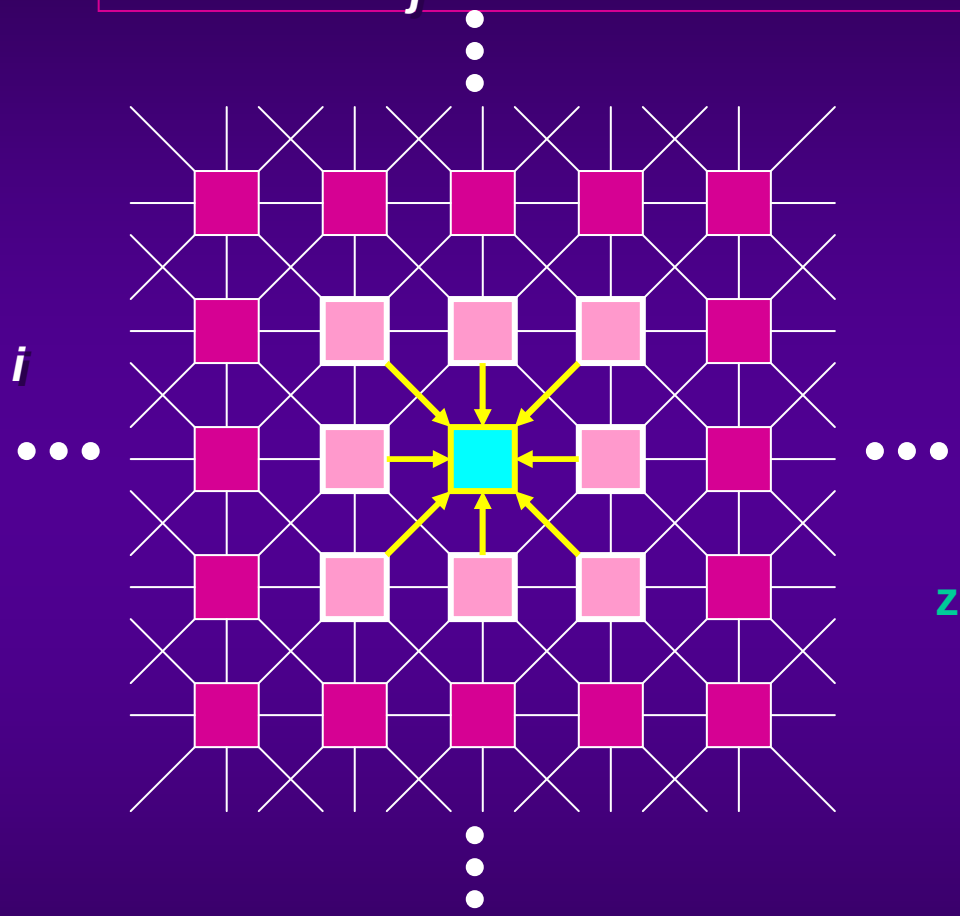
- Local connections with neighboring cells
- Image and the output is fed to the cell input after multiplied by a template.
- Different template (19 parameters) produces different processing results.

2D Cellular Neural Networks

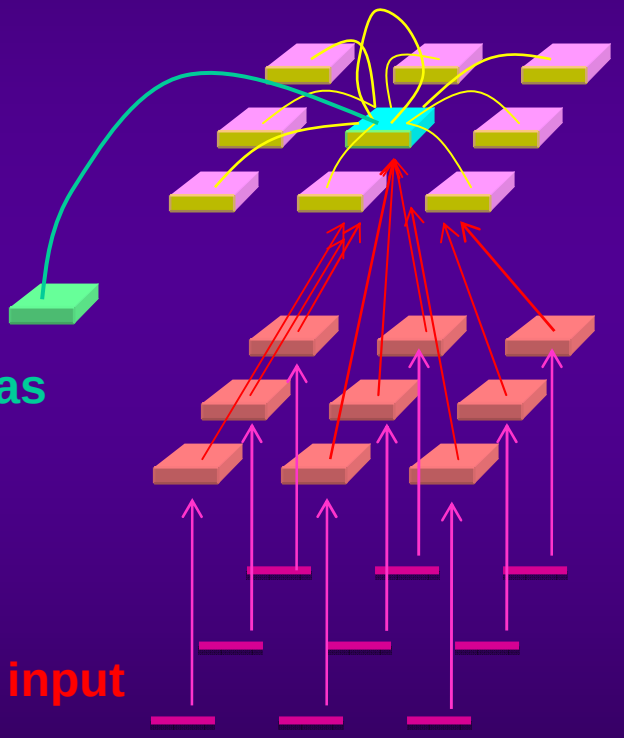
- The Neural Model to mimic the human visual system
- Parallel processing
- Interconnections among neighboring cells
- Diverse processing with different templates



Cellular Neural/Nonlinear Network (CNN)



x_{ij} - state/ y_{ij} - output



z_{ij} - bias

u_{ij} - input

Template - the program of the network: [A B z]

CNN Equation

State Equation

$$\frac{dx_{ij}}{dt} = -x_{ij} + z_{ij} + \sum b_{kl}u_{kl} + \sum a_{kl}f(x_{ij})$$

Z

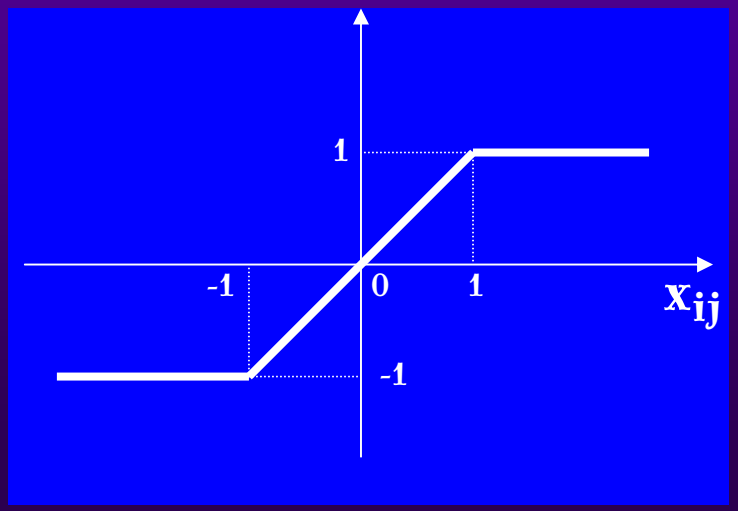
B

b_9	b_8	b_7
b_6	b_5	b_4
b_3	b_2	b_1

A

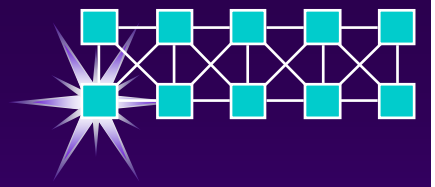
a_9	a_8	a_7
a_6	a_5	a_4
a_3	a_2	a_1

$$y_{ij} = f(x_{ij})$$



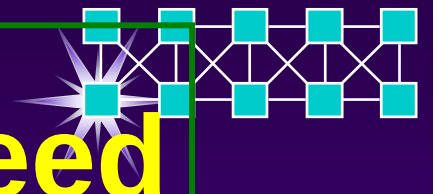
Output Equation

$$y_{ij} = f(x_{ij})$$



Each set of templates instructs the CNN to perform a specific task.

CNN Computation Speed



A typical **CNN**

template can be implemented on a

CNN Universal Chip

(0.18µ technology) in approximately

1 nano second.

Information on how to run CNN simulator “CANDY”

1. Visit the EE129 home page at
   http://www-inst.eecs.berkeley.edu/~ee129/fa09/

2. Choose “CNN Simulator” on the front page

3. Set up the simulator by clicking
   “CANDY Setup”

4. Download the manual by clicking the menu,
   “CNN Simulator, CANDY”, and then
   follow the instructions in the manual.

Examples of Simple Image Processing

< Edge Detection >

< Input Image >

< Output Image >

< Templates >

B TEMPLATE:

0 0 0
0 2 0
0 0 0

BIAS: -0.5

A Template:

-1.0 -1.0 -1.0

-1.0 8.0 -1.0

-1.0 -1.0 -1.0

Examples of Simple Image Processing

< Edge Detection >

< Input Image >

< Templates >

B TEMPLATE:

0 0 0
0 2 0
0 0 0

BIAS: -0.5

A TEMPLATE:

-1.0 -1.0 -1.0

-1.0 8.0 -1.0

-1.0 -1.0 -1.0

< Output : >

Examples of Simple Image Processing

< Median >

< Input Image >

< Templates >

B TEMPLATE:
0 0 0
0 1 0
0 0 0

NONLIN_D:
x-u#
d d d
d 0 d
d d d

d 0 2 0.01 0.5 2.5 -0.5
BIAS: 0.0

< Output Image >

Examples of Simple Image Processing

< Median >

< Input Image >

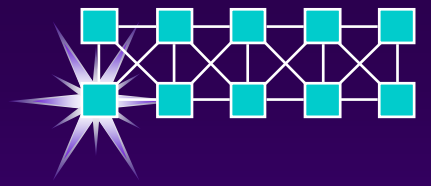
< Templates >

B TEMPLATE:
0 0 0
0 1 0
0 0 0

NONLIN_D:
x-u#
d d d
d 0 d
d d d

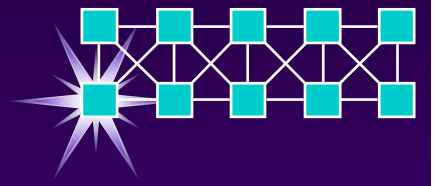
d 0 2 0.01 0.5 2.5 -0.5
BIAS: 0.0

< Output : >



Representative Image Processing Primitives

Edge Detection CNN



Cloning Templates

z : -0.5

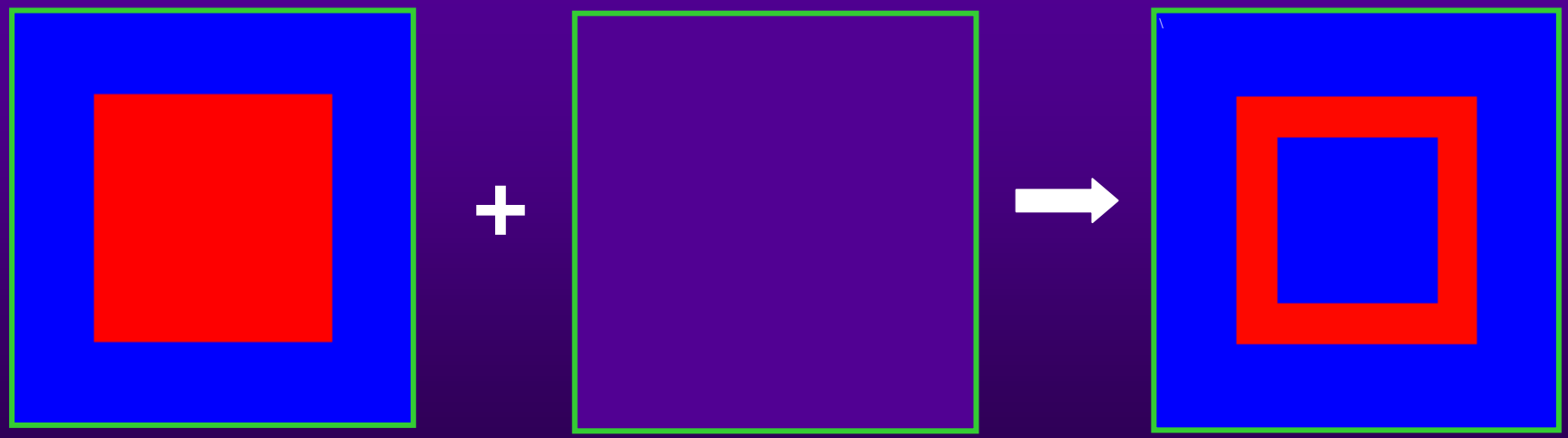
B :

-1	-1	-1
-1	8	-1
-1	-1	-1

A :

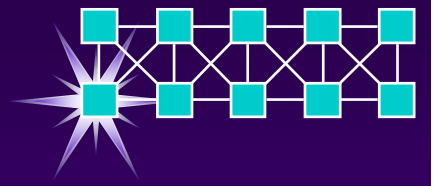
0	0	0
0	2	0
0	0	0

Example



Input Image **Initial Image** **Output Image**

Corner Detection CNN



Cloning Templates

z : -8.5

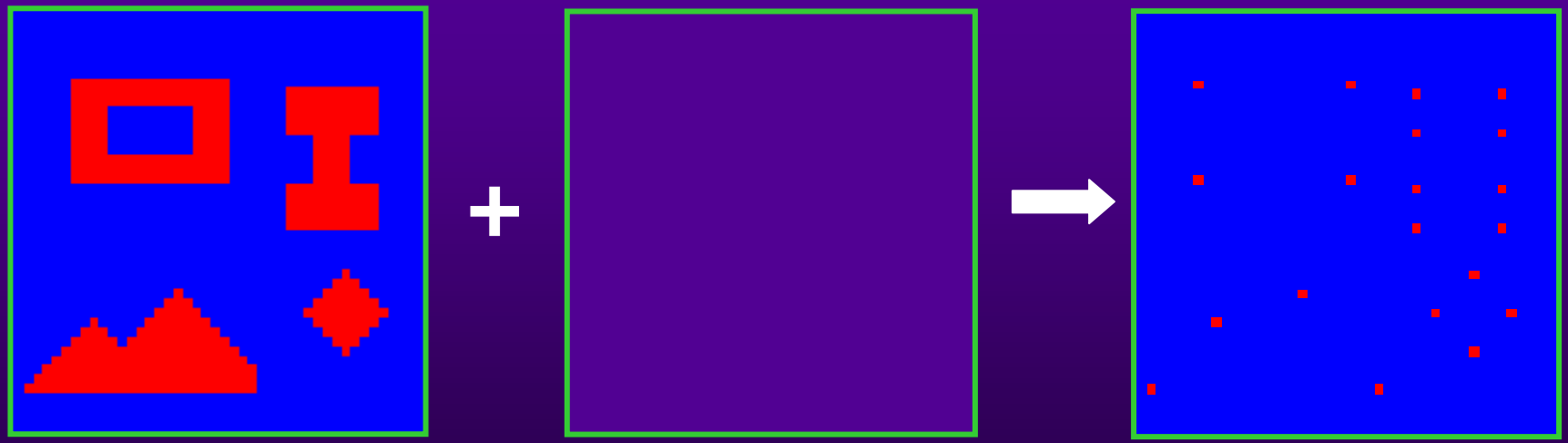
B :

-1	-1	-1
-1	8	-1
-1	-1	-1

A :

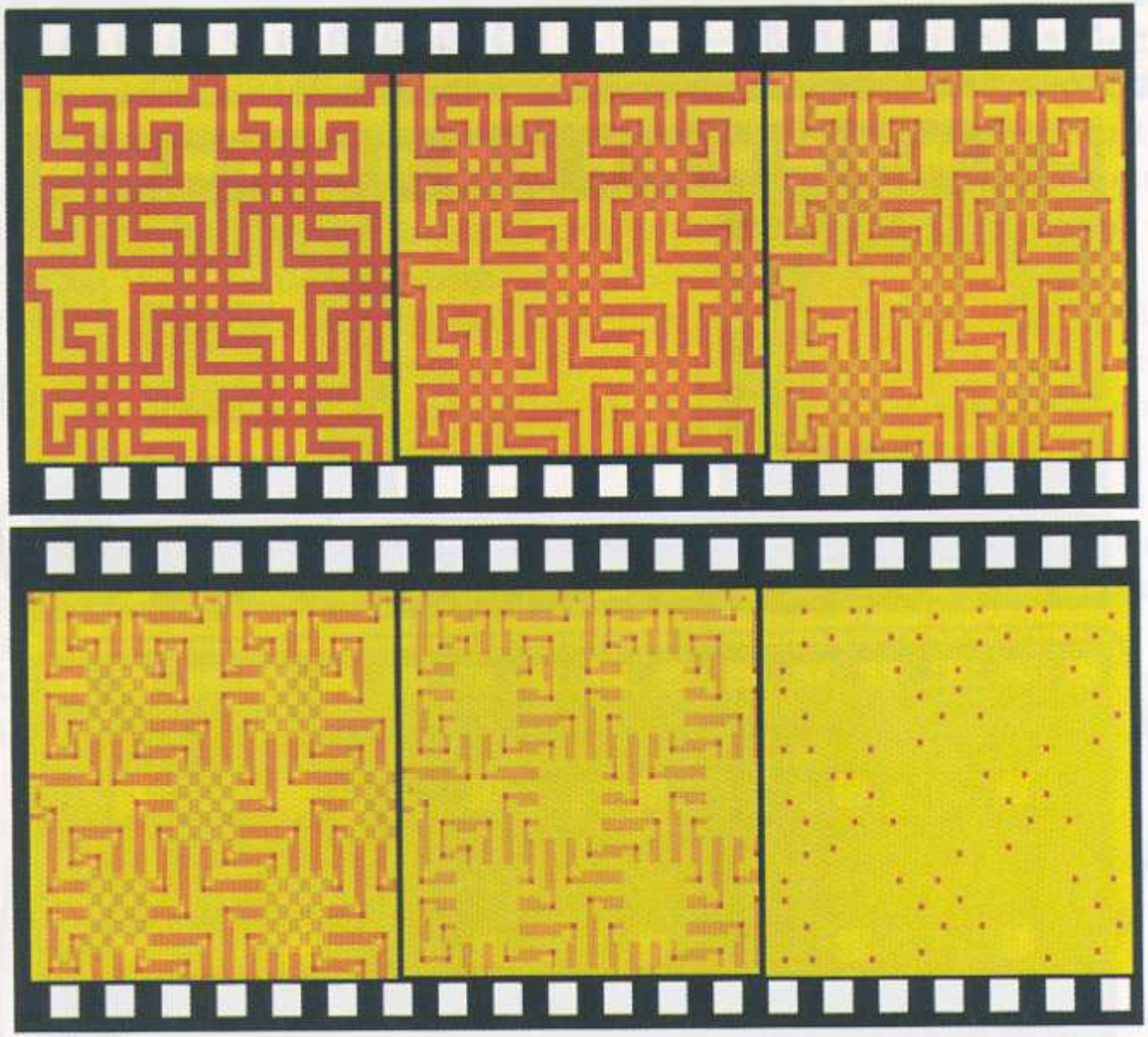
0	0	0
0	2	0
0	0	0

Example



Input Image **Initial Image** **Output Image**

Corner detector



Name: **CORNER DETECTION CNN**

Task Prescription: *Extract convex corners of a binary image. A black pixel (red in pseudo-color) is a convex corner if and only if it is surrounded by 5 or more white pixels (blue in pseudo-color).*

Cloning Template

z : -8.5

B :

-1	-1	-1
-1	8	-1
-1	-1	-1

A :

0	0	0
0	2	0
0	0	0

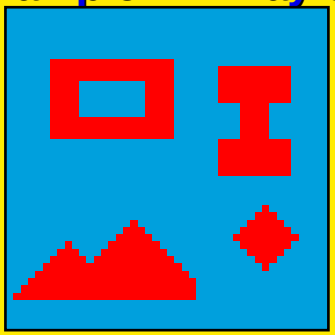
Boundary

Fixed: $x_{i*j*} = 0$, $u_{i*j*} = 0$
*(i*j* denotes boundary cells)*

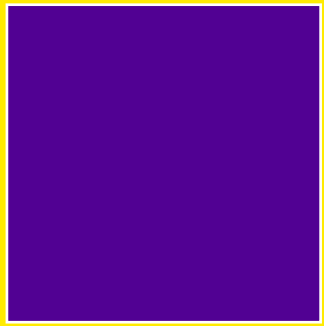
Initial State

$x_{ij}(0) = 0$ (white in pseudo-color)

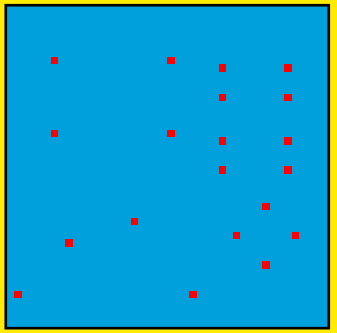
Example 1: Array Size = 44 x 44



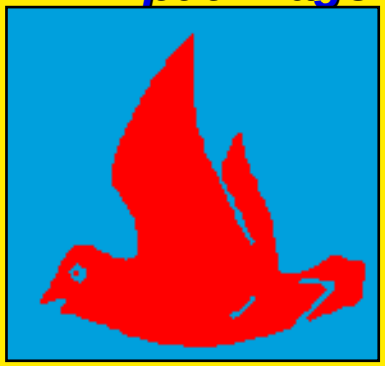
Input Image

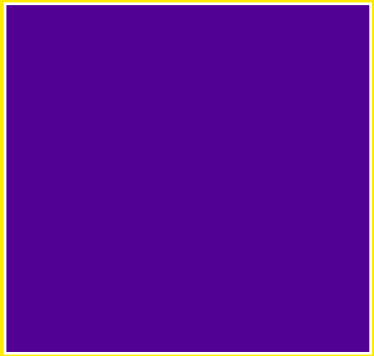


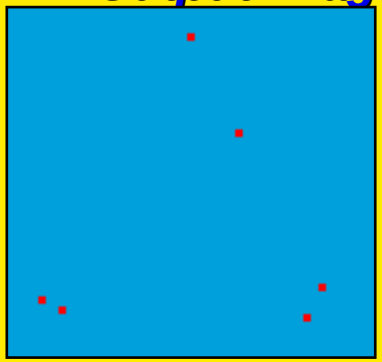
Initial State



Output Image

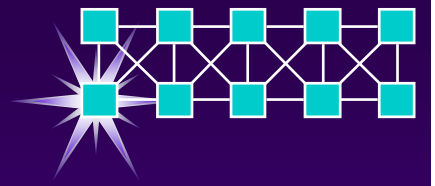






Example 2: Array Size = 140 x 140

Point Extraction CNN



Cloning Templates

z : -8

B :

-1	-1	-1
-1	1	-1
-1	-1	-1

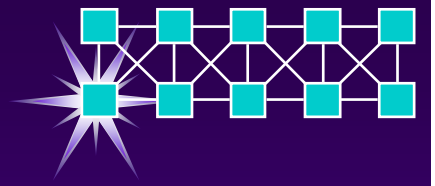
A :

0	0	0
0	1	0
0	0	0

Example

Input Image **Initial Image** **Output Image**

Hole-Filling CNN



Cloning Templates

z : -1

B :

0	0	0
0	4	0
0	0	0

A :

0	1	0
1	3	1
0	1	0

Example

Input Image **Initial Image** **Output Image**

Name: **SHADOW PROJECTION CNN**

Task Prescription: ***Project onto the left the shadow of all objects illuminated from the right.***

Cloning Template

z : 0

B :

0	0	0
0	2	0
0	0	0

A :

0	0	0
0	2	2
0	0	0

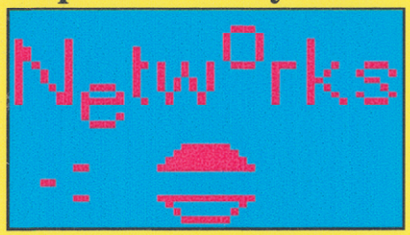
Boundary Condition

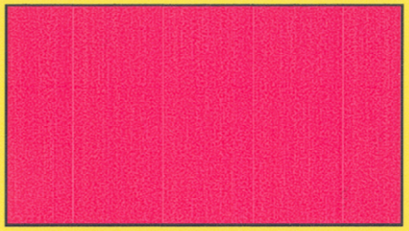
Fixed: $x_{i^*j^*} = 0$, $u_{i^*j^*} = 0$
(i*j* denotes boundary cells)

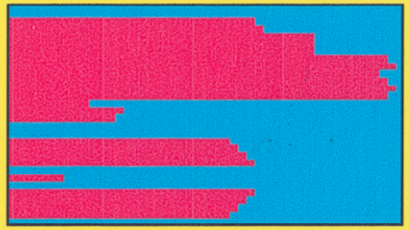
Initial State

$x_{ij}(0) = 1$ ***(red in pseudo-color)***

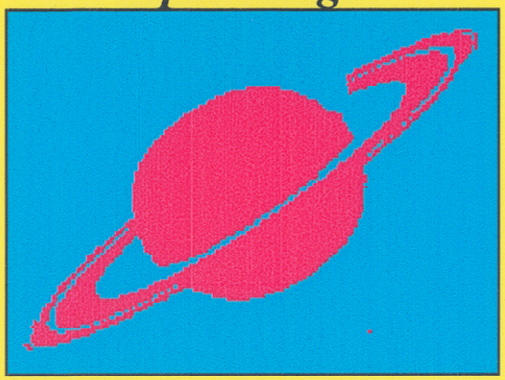
Example 1: Array Size = 30 x 48

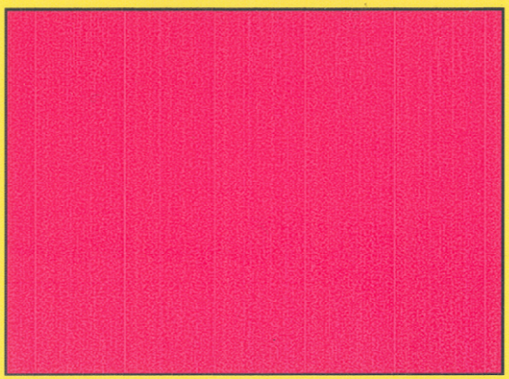


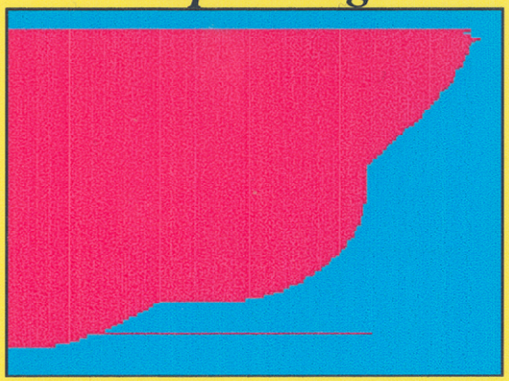




Input Image ***Initial State*** ***Output Image***







Example 2: Array Size = 176 x 176

Selected Objects Extraction CNN

Cloning Templates

z :

0

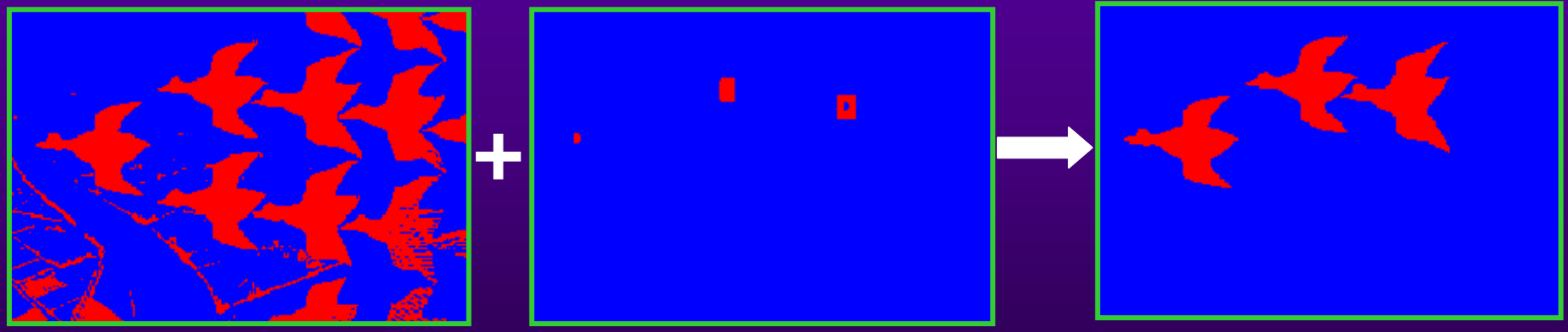
B :

0	0	0
0	1.75	0
0	0	0

A :

0.25	0.25	0.25
0.25	1	0.25
0.25	0.25	0.25

Example



Input Image **Initial Image** **Output Image**

Name: HORIZONTAL TRANSLATION CNN

Task Prescription: *Shift a binary image by one-pixel to the right. Rightmost pixels of all image rows are discarded whereas leftmost pixels of all rows are filled with the background color. For left translation, simply rotate* **B** *by* 180^0 *about the center.*

Cloning Template

z :

0

B :

0	0	0
1	0	0
0	0	0

A :

0	0	0
0	1	0
0	0	0

Boundary Condition

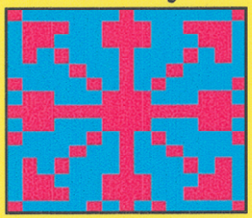
Fixed: $x_{i^*j^*} = 0, \quad u_{i^*j^*} = -1$

*(i*j* denotes boundary cells)*

Initial State

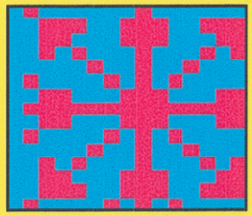
$x_{ij}(0) = 0$ *(white in pseudo-color)*

Example 1: Array Size = 16 x 16

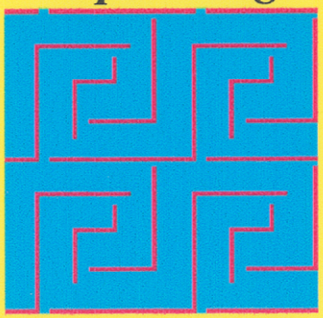


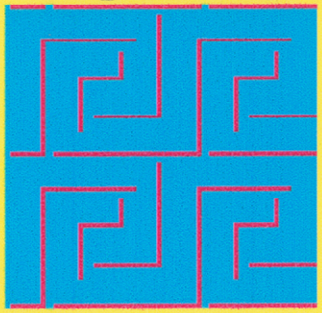
Input Image

Initial State



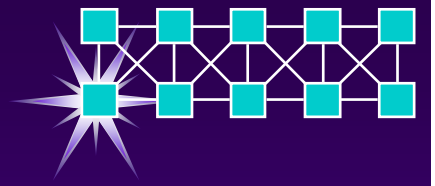
Output Image





Example 2: Array Size = 64 x 64

Point Removal CNN



Cloning Templates

z : -1

B :

1	1	1
1	8	1
1	1	1

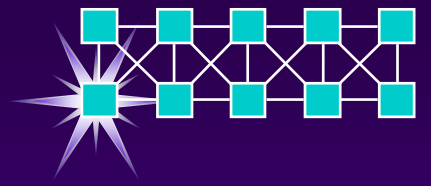
A :

0	0	0
0	1	0
0	0	0

Example

Input Image **Initial Image** **Output Image**

Erosion CNN



Cloning Templates

z : -4.5

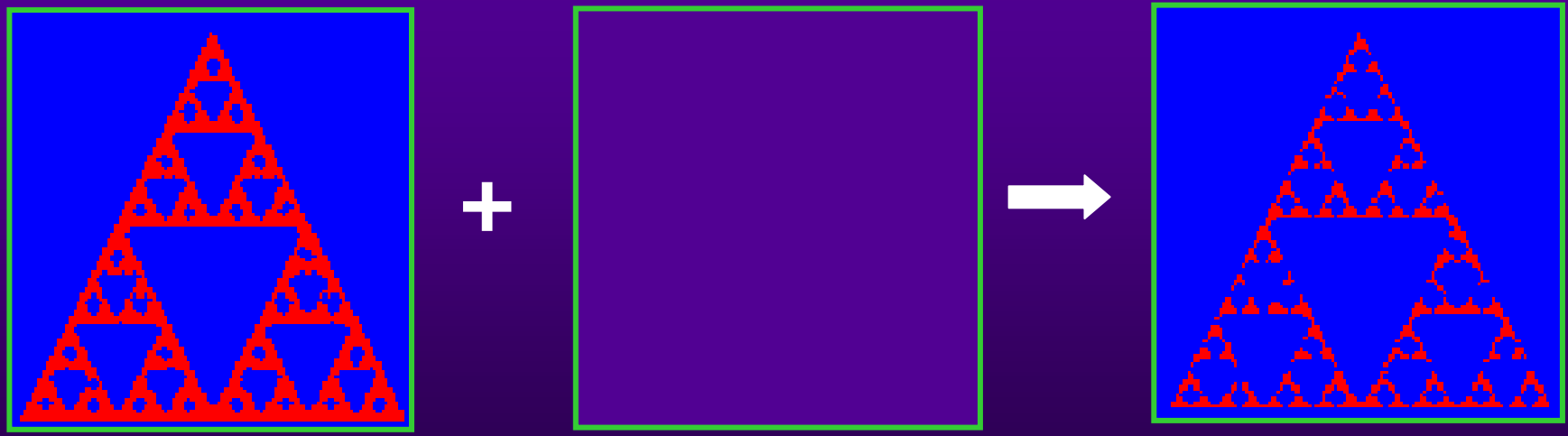
B :

0	1	0
1	1	1
0	1	0

A :

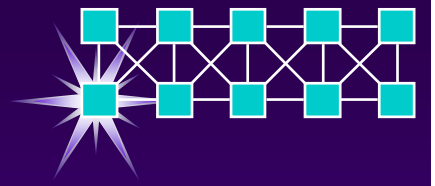
0	0	0
0	2	0
0	0	0

Example



Input Image **Initial Image** **Output Image**

Dilation CNN



Cloning Templates

z : 4.5

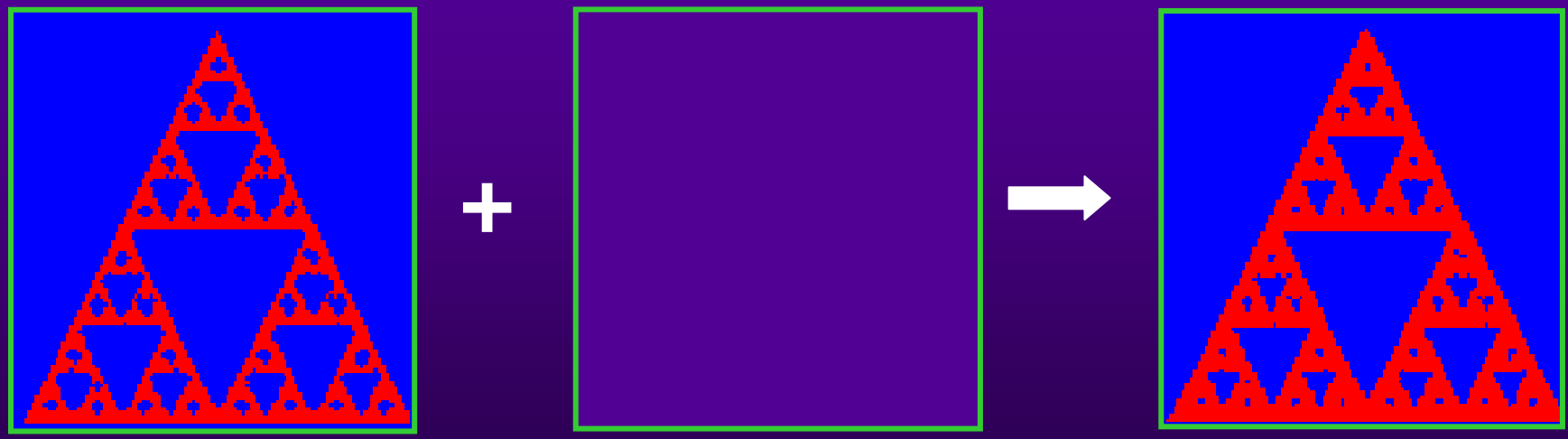
B :

0	1	0
1	1	1
0	1	0

A :

0	0	0
0	2	0
0	0	0

Example



Input Image **Initial Image** **Output Image**

Logic “NOT” Operation CNN

Cloning Templates

z : 0

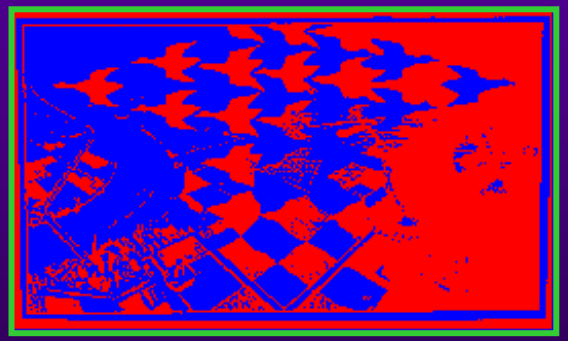
B :

0	0	0
0	-2	0
0	0	0

A :

0	0	0
0	1	0
0	0	0

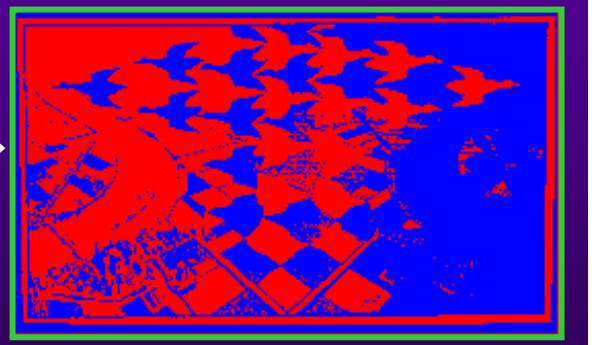
Example



Input Image

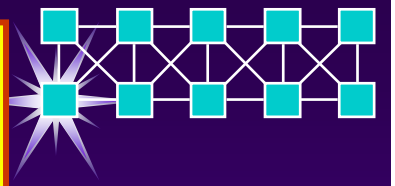
+

Initial Image



Output Image

Logic “AND” Operation CNN



Cloning Templates

z : -1

B :

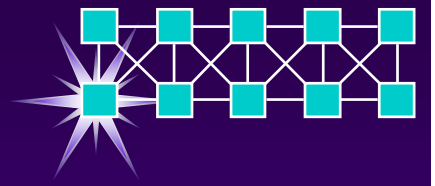
0	0	0
0	1	0
0	0	0

A :

0	0	0
0	2	0
0	0	0

Example

Input Image **Initial Image** **Output Image**



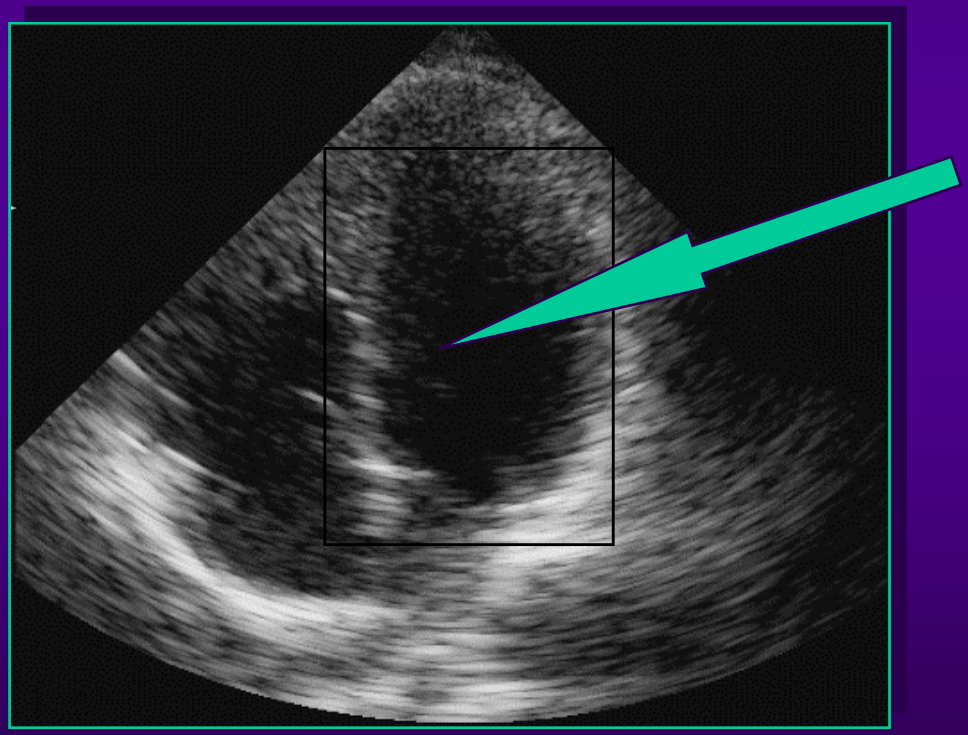
Application Examples

Echocardiogram

Motivation:

Detection of shapes and boundaries

Qualitative and quantitative analyzes

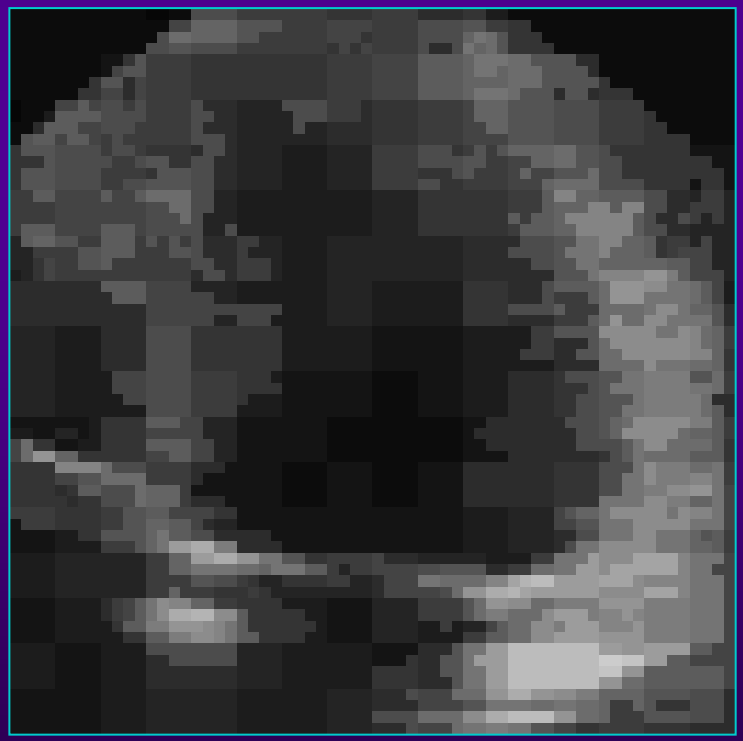


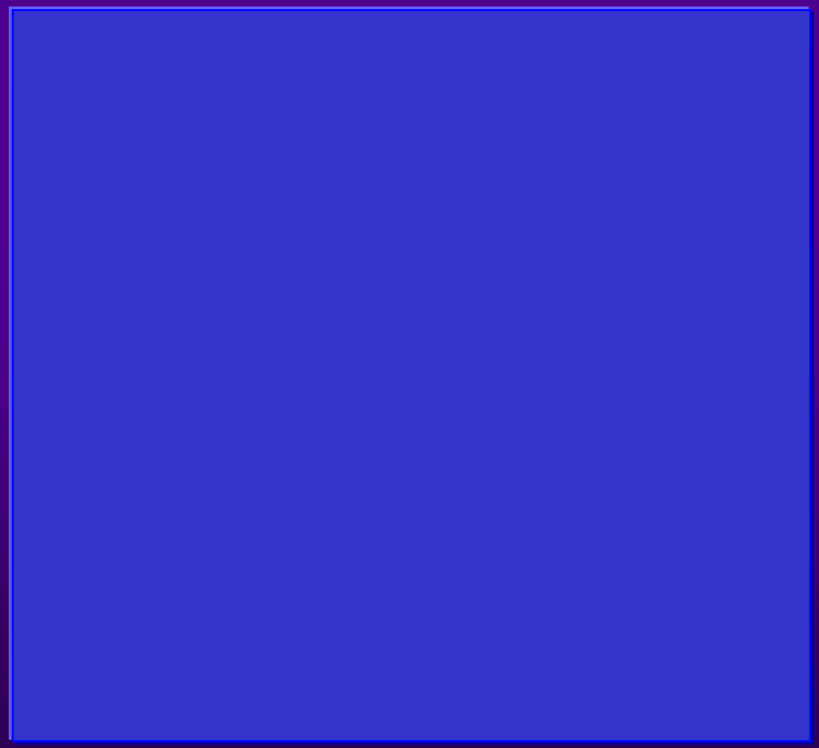
Human hart

Experiment:

Detection of the internal boundary of the left ventricle

Measurement results



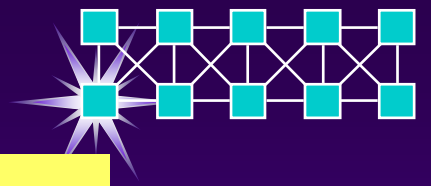


Separation of the stationary and moving parts

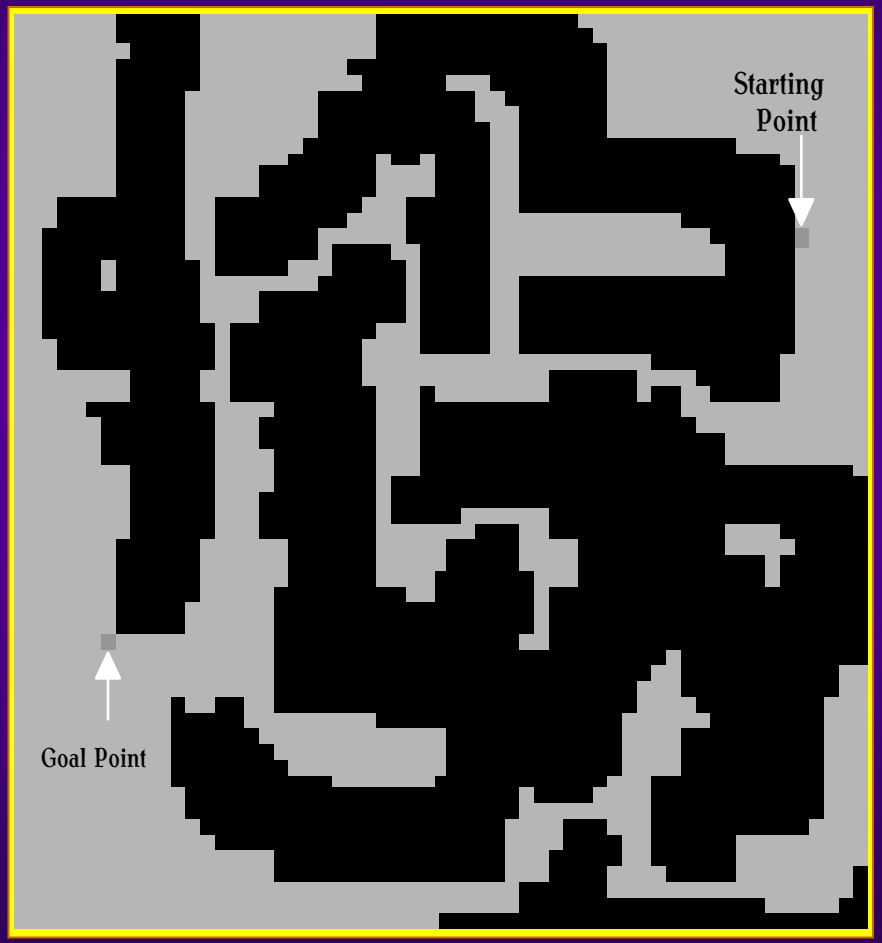
Original image parts

Moving parts

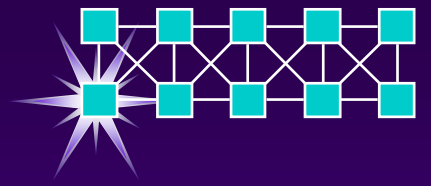
Stationary

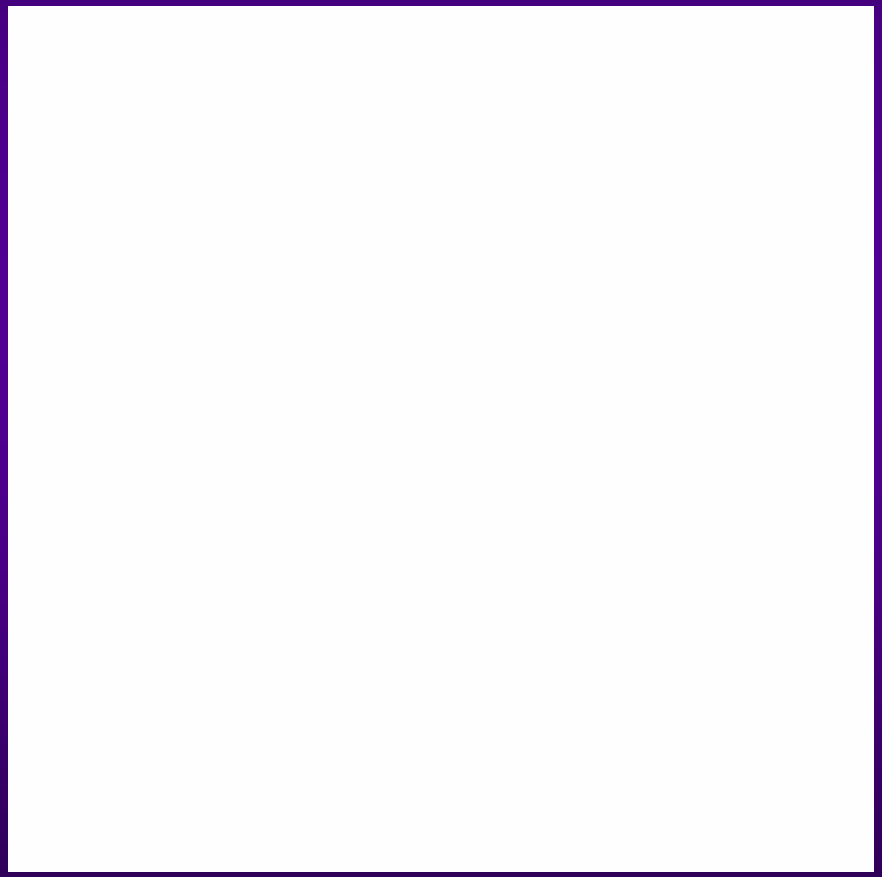


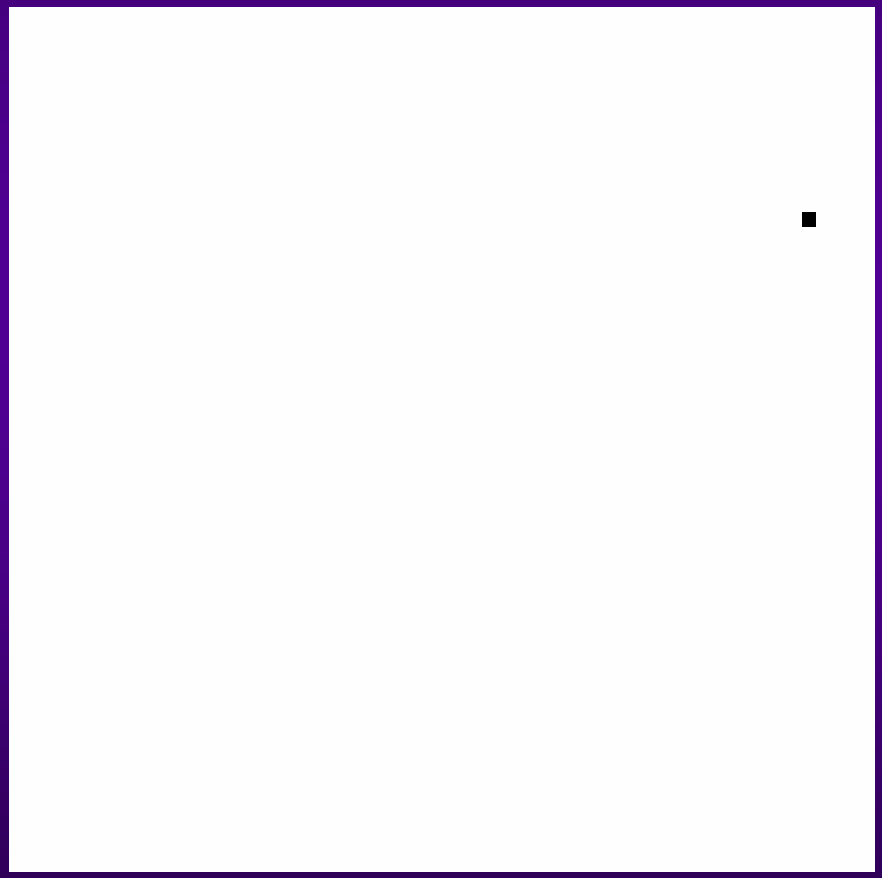
Optimal Path Finding with Space Variant Metric Weights via Multilayer CNN-UM


Starting
Point
Goal Point

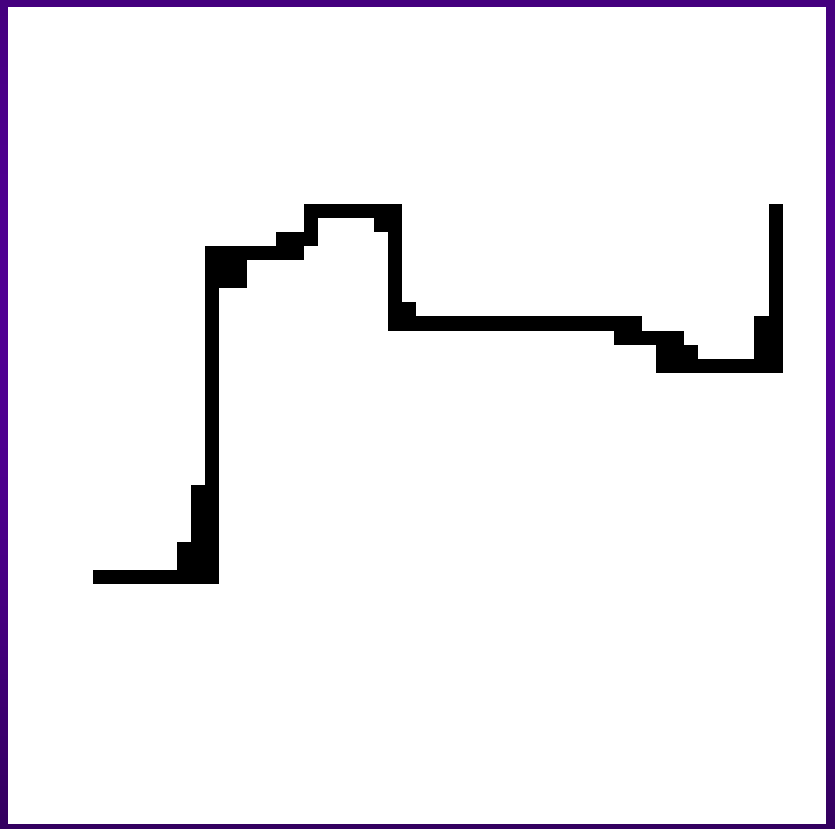
Dynamic Path Finding

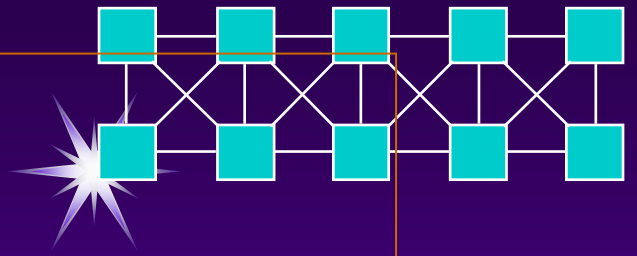






Determined Path





Target Tracking

Dim Airplane Target Detection

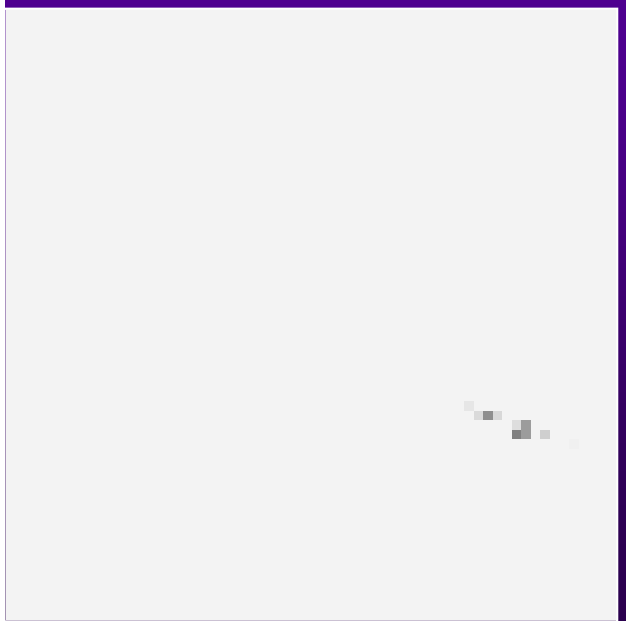
Original Image Sequence ↓

With Detected Target ↓

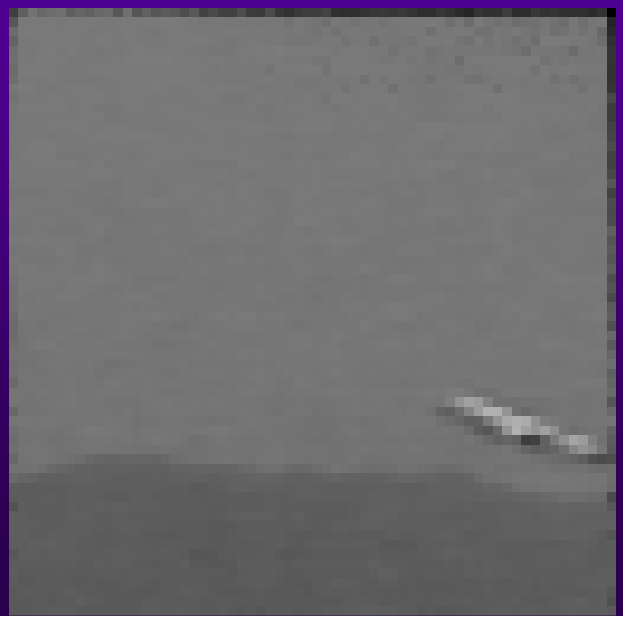
Dim Airplane Target Initiation & Tracking

Original Image Sequence ⇒

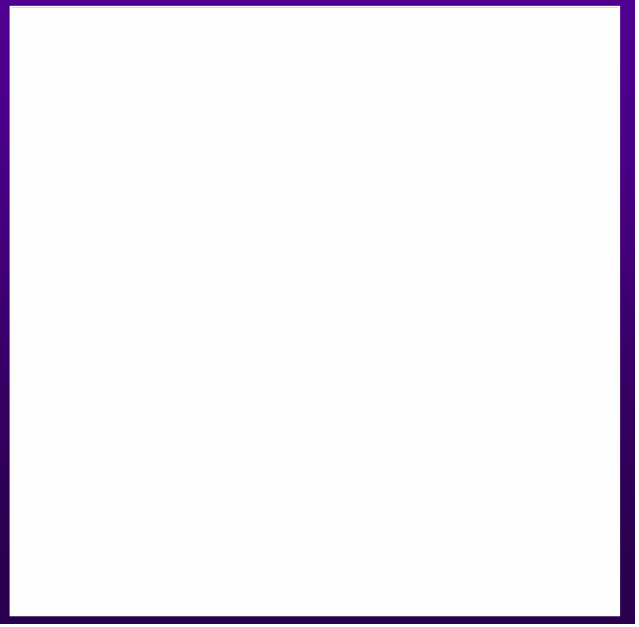
Confidence of Target Existence ↓



With Detected Target ↓

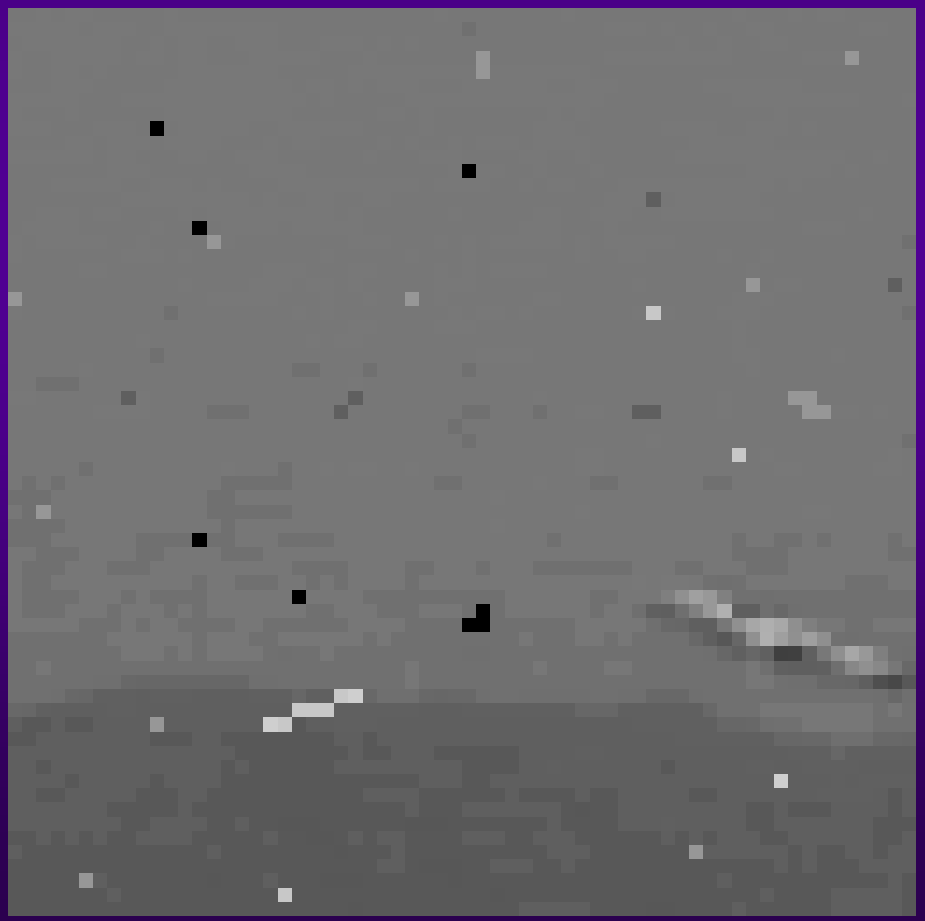


Trajectory of Detected Target ↓



Dim Noisy Airplane Target Detection

Original Noisy Image ↓

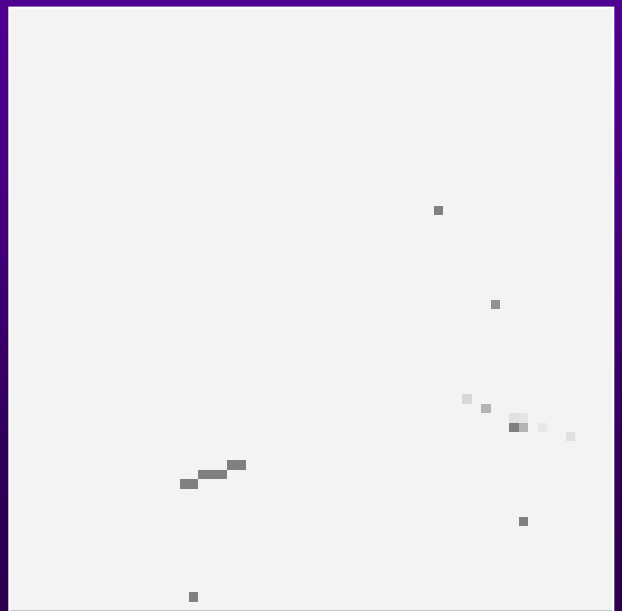


With Detected Target ↓

Dim Noisy Target Initiation & Tracking

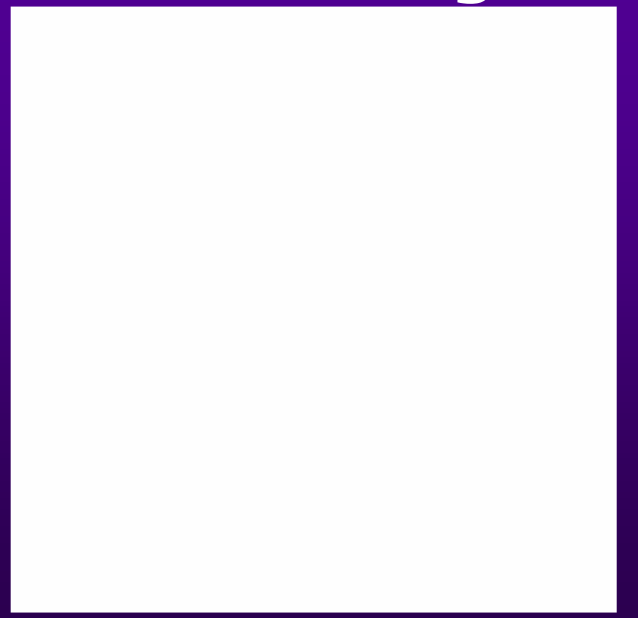
Original Noisy Image ⇒

Confidence of Target Existence ↓



With Detected Target ↓

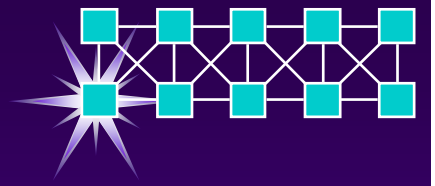
Trajectory of Detected Target ↓



Temporally Missing & Backward Moving Target Detection

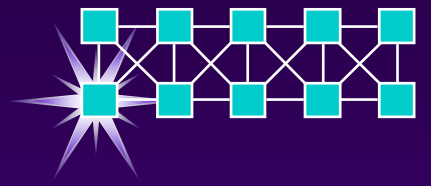
Original Image Sequence ↓

With Detected Target ↓



Hardware Implementation Issue

CNN Equation



State Equation

$$\frac{dx_{ij}}{dt} = -x_{ij} + z_{ij} + \sum b_{kl}u_{kl} + \sum a_{kl}f(x_{ij})$$

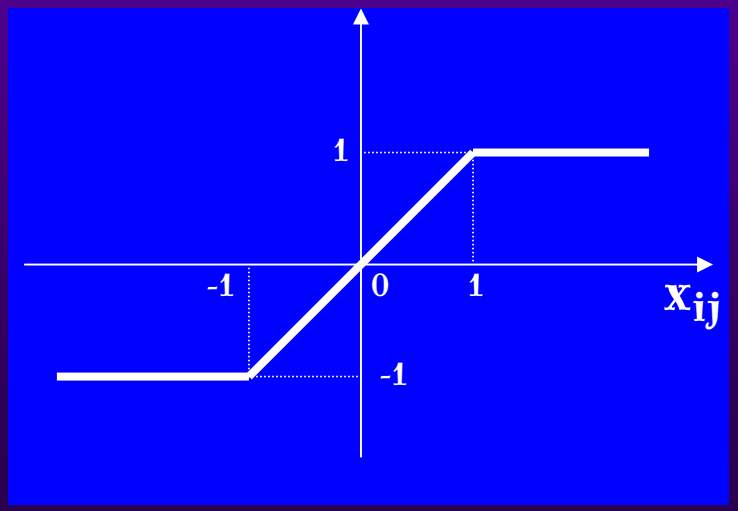
z

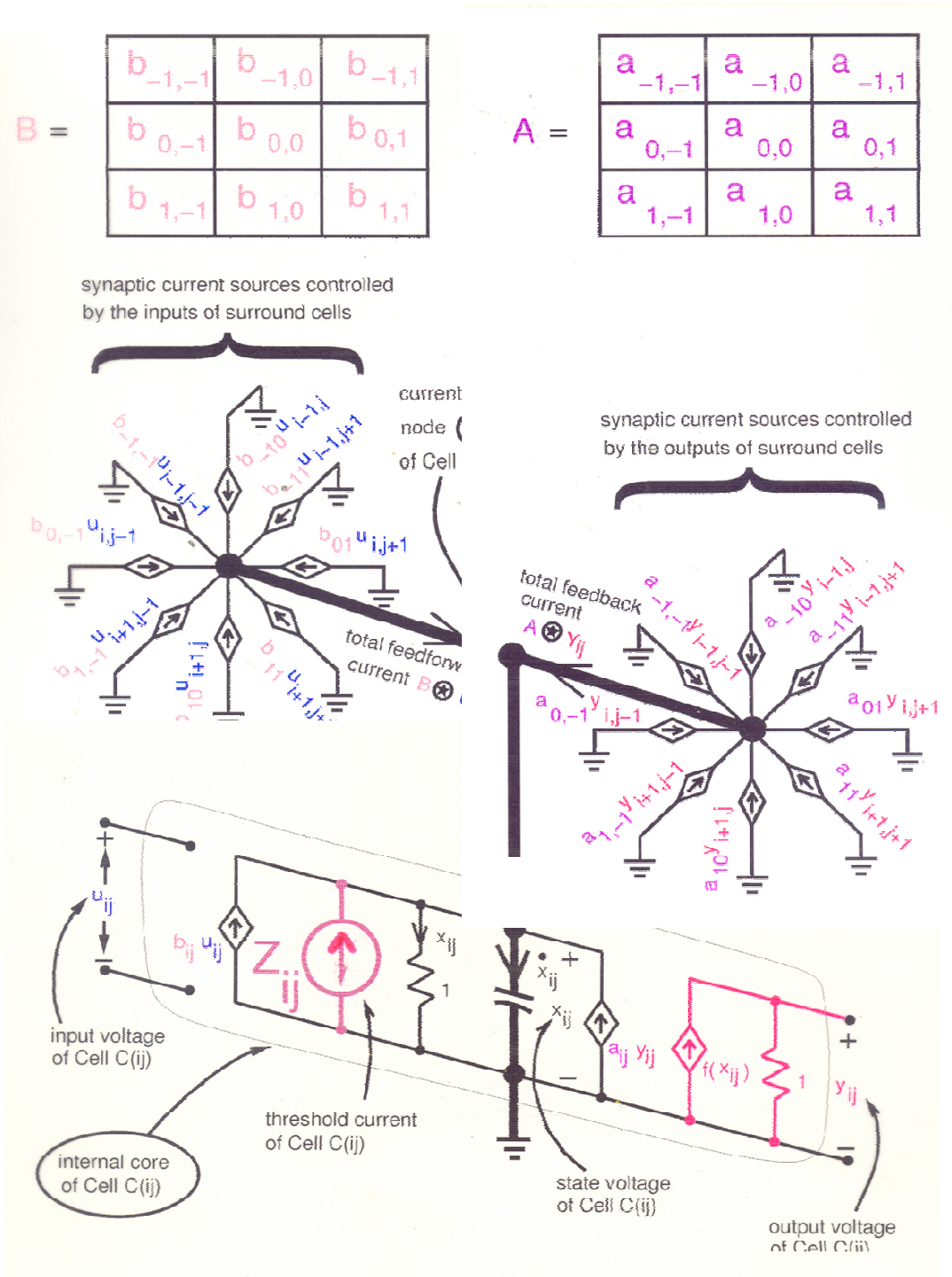
$$B = \begin{bmatrix} b_9 & b_8 & b_7 \\ b_6 & b_5 & b_4 \\ b_3 & b_2 & b_1 \end{bmatrix} \qquad A = \begin{bmatrix} a_9 & a_8 & a_7 \\ a_6 & a_5 & a_4 \\ a_3 & a_2 & a_1 \end{bmatrix}$$

$$y_{ij} = f(x_{ij})$$

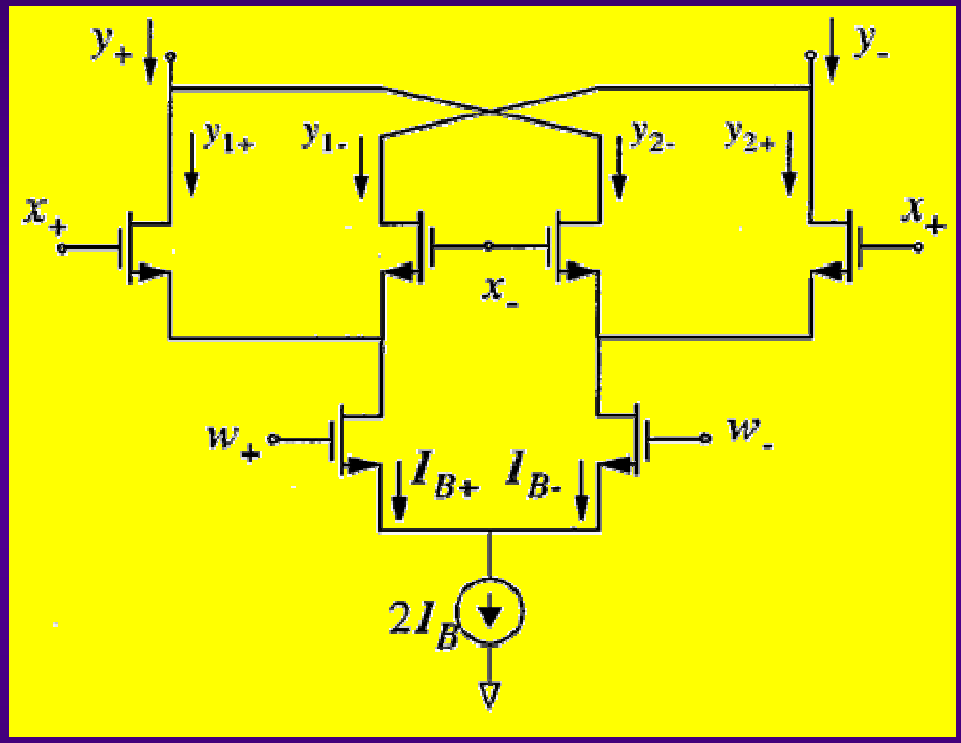
Output Equation

$$y_{ij} = f(x_{ij})$$



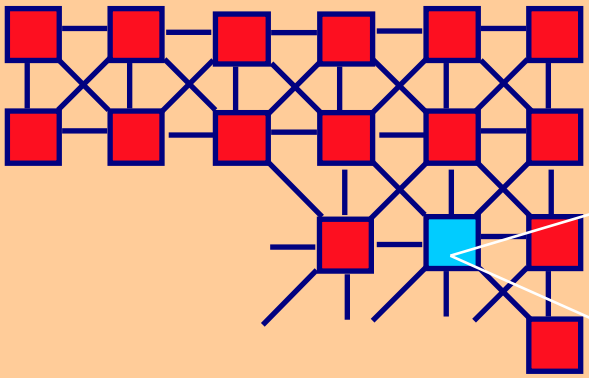

B =
$b_{-1,-1}$ $b_{-1,0}$ $b_{-1,1}$
$b_{0,-1}$ $b_{0,0}$ $b_{0,1}$
$b_{1,-1}$ $b_{1,0}$ $b_{1,1}$
A =
$a_{-1,-1}$ $a_{-1,0}$ $a_{-1,1}$
$a_{0,-1}$ $a_{0,0}$ $a_{0,1}$
$a_{1,-1}$ $a_{1,0}$ $a_{1,1}$
synaptic current sources controlled by the inputs of surround cells
current node of Cell
total feedforw current B
synaptic current sources controlled by the outputs of surround cells
total feedback current A
input voltage of Cell C(ij)
threshold current of Cell C(ij)
internal core of Cell C(ij)
state voltage of Cell C(ij)
output voltage of Cell C(ij)

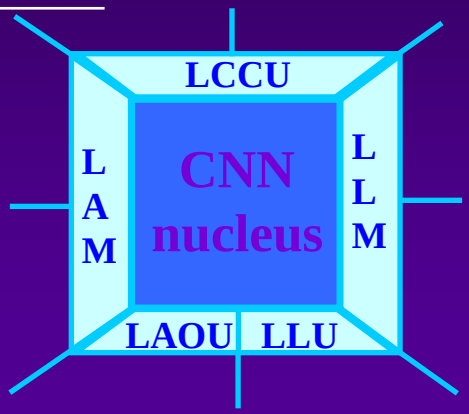
Analog Multiplier

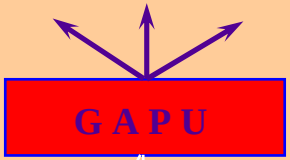


$$\frac{dx_{ij}}{dt} = -x_{ij} + z_{ij} + \sum b_{kl}u_{kl} + \sum a_{kl}f(x_{ij})$$

CNN Universal Machine (CNN-UM)







GAPU: Global Analogic Programming Unit

LAM: Local Analogue Memory
LLM: Local Logic Memory
LCCU: Local Communication and Control Unit
LAOU: Local Analogue Output Unit
LLU: Local Logic Unit

APR: Analog Instruction Register
LPR: Logic Program Register
SCR: Switch Configuration Register
GACU: Global Analogic Control Unit

$[A_1\ B_1\ z_1], [A_2\ B_2\ z_2], \ldots$

"Analogic (analog+logic) algorithm"

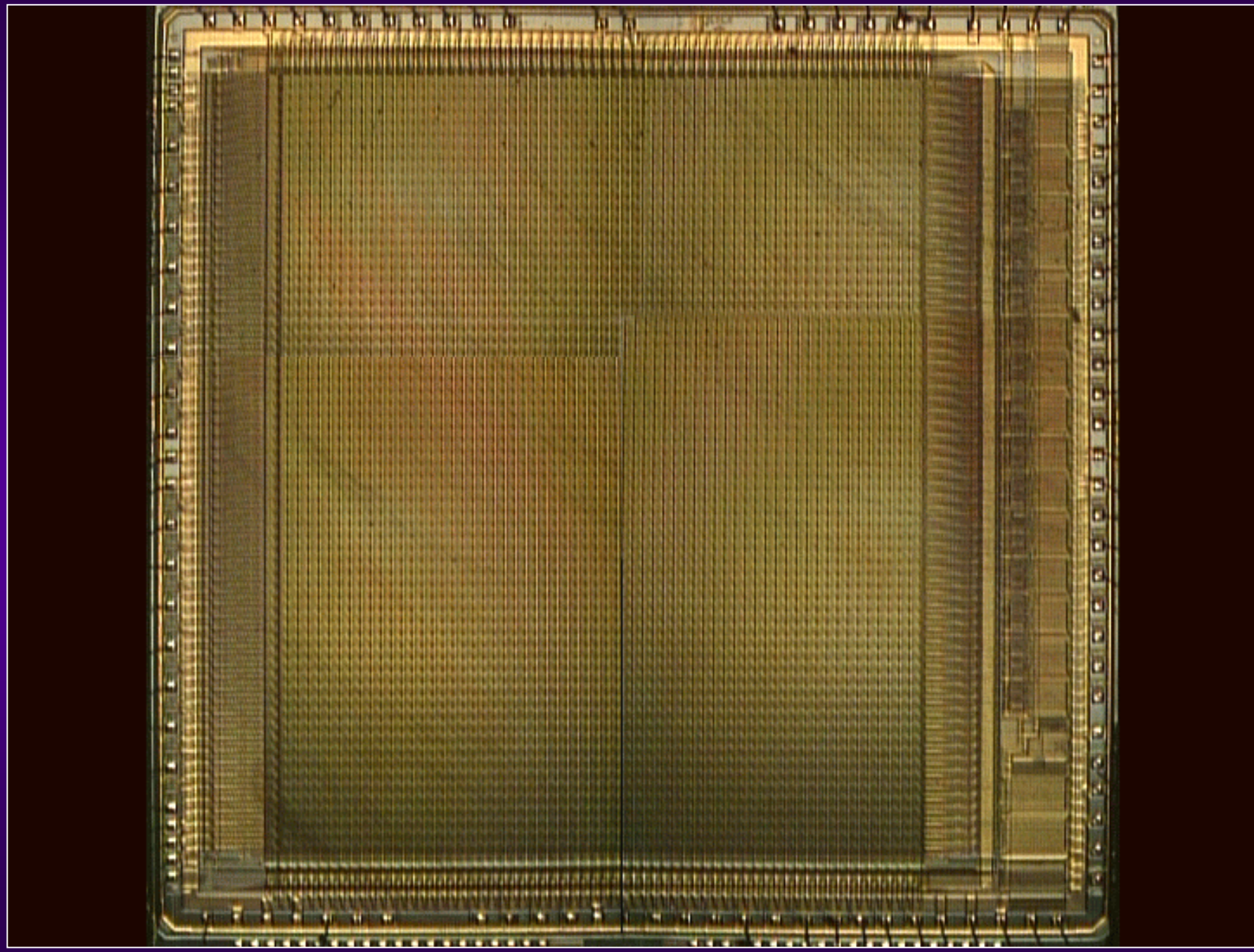
Cellular Neural Network (CNN)

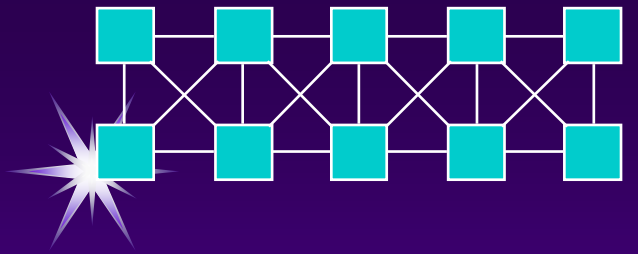












Q- eye

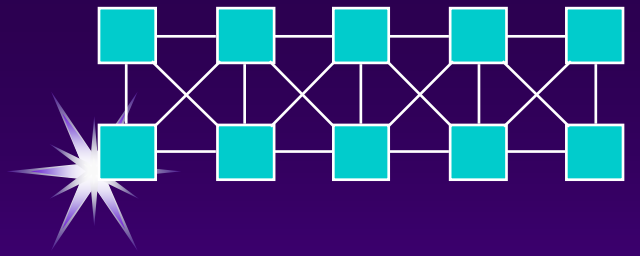
is the name of a

CNN Chip

from The

Anafocus Company

AnaFocus
perceiving the analogue world



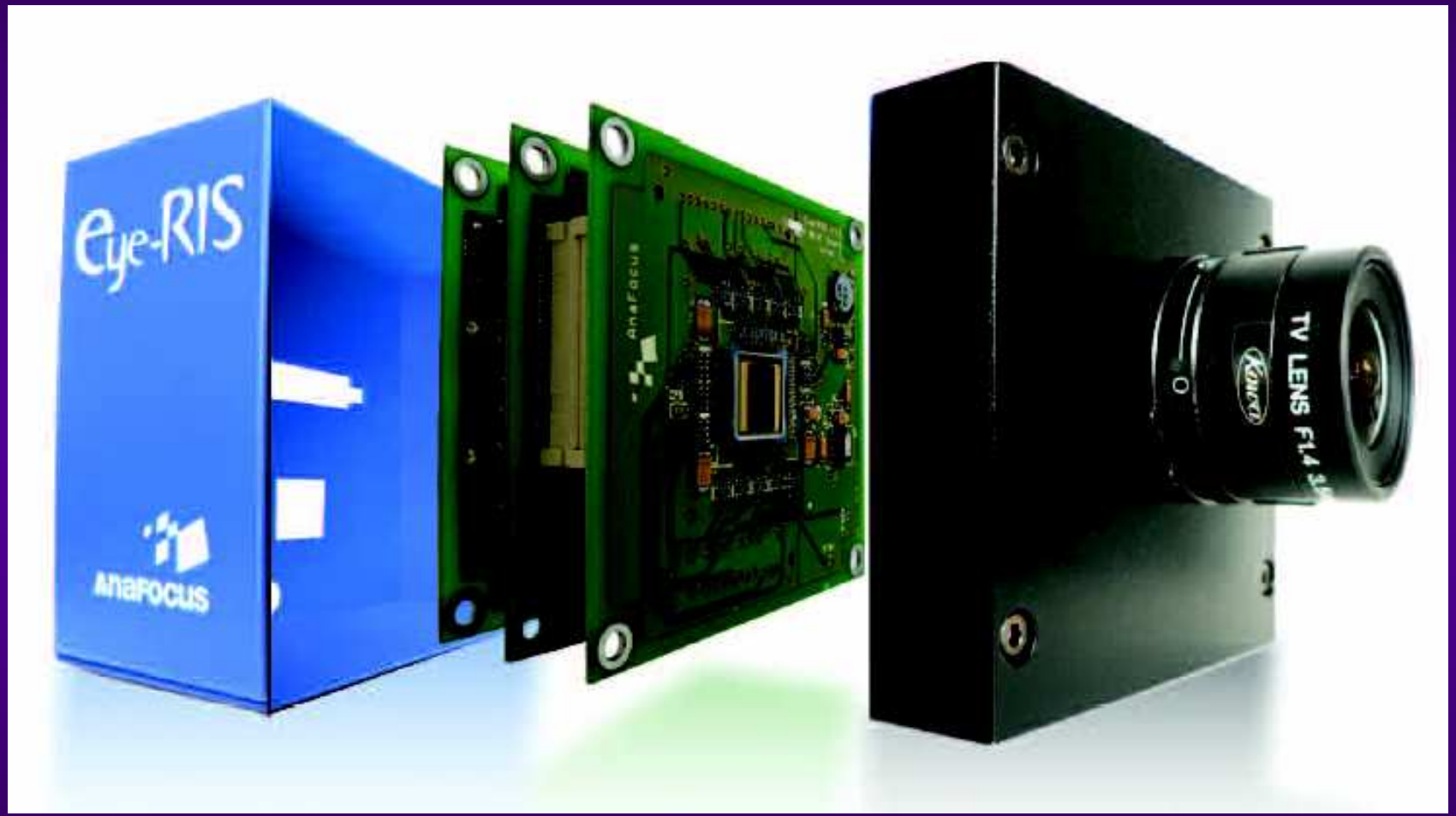
The Q-Eye™ Cellular Sensor-Processor Array Development within the Eye-RIS™ Standalone Vision System

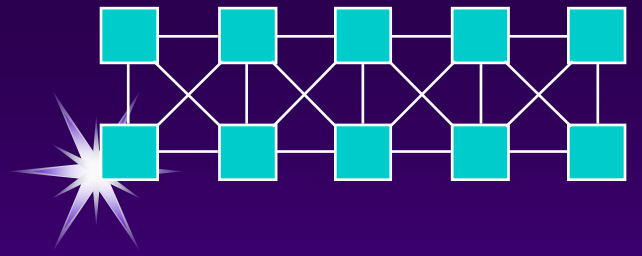
AnaFocus Co.

Angel Rodriguez Vazquez, CEO (founder and managing director)

Seville, Spain

(2009)


eye-RIS
AnaFocus
AnaFocus
TV LENS F1.4

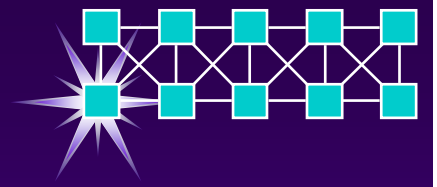


QCIF

The standard resolution for tele-video systems is

176×144

The commercial



Q - Eye

CNN Chip

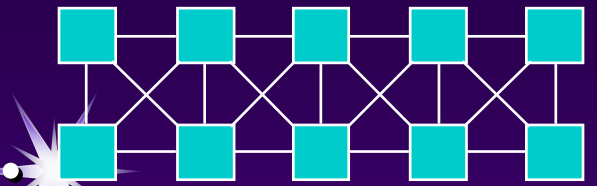
(fabricated in 2007)

has

176 X 144

= 25,344

CMOS cells.



Multi-core Video Analytics Engine (MVE™) Development Based on Cellular Processor Arrays

Eutecus, Inc.

Csaba Rekeczky, CTO and President
(founder and managing director)

Berkeley, CA, USA

(2009)

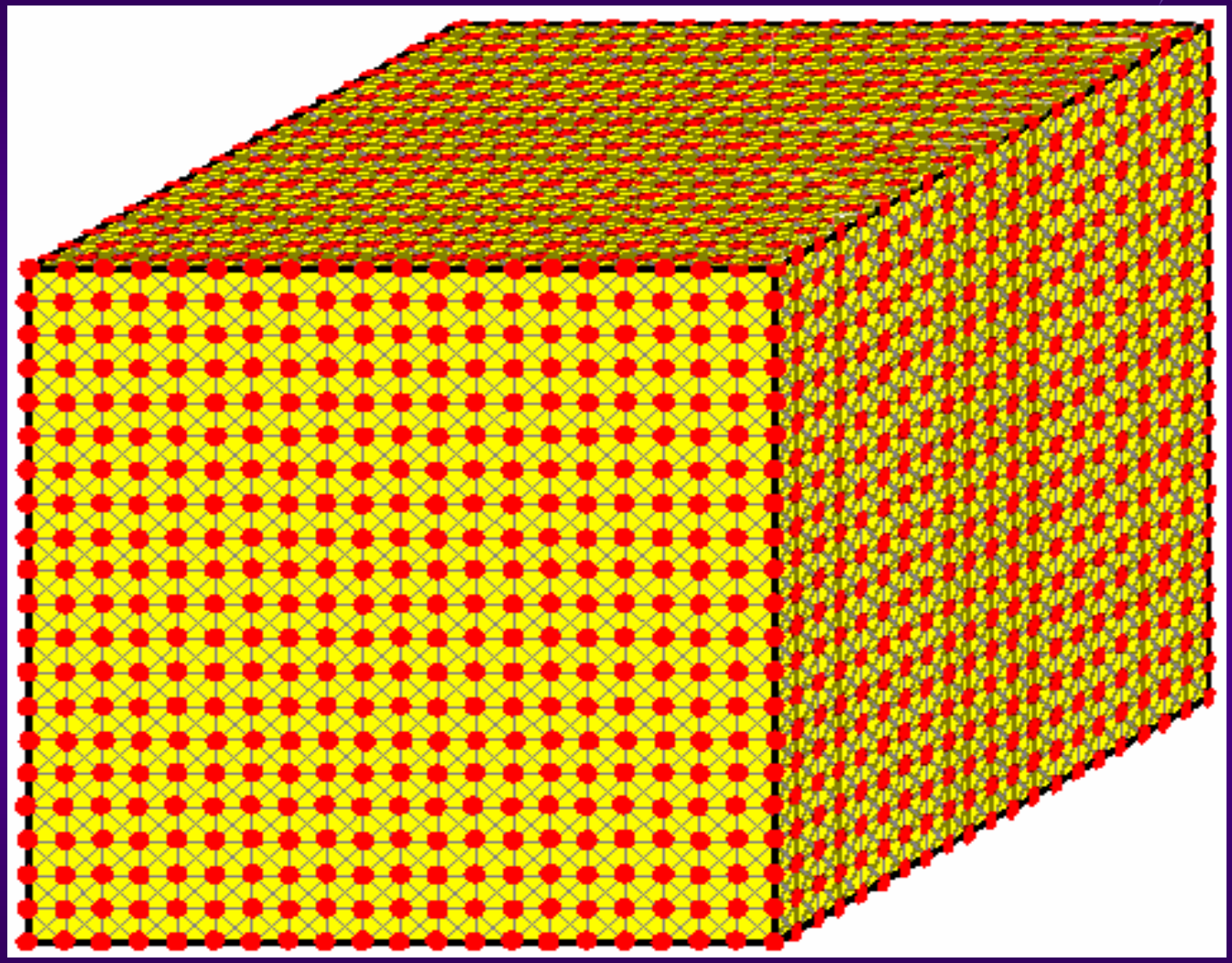
VISCUBE
3D CNN Vision Cube

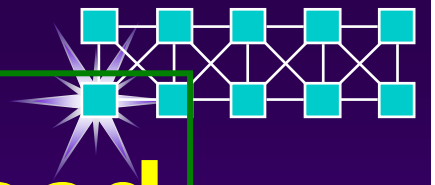
Eutecus Inc. Berkeley, California

www.eutecus.com

BAA N00173-08C-4005

Financed by ONR





CNN Computation Speed

A typical **CNN**

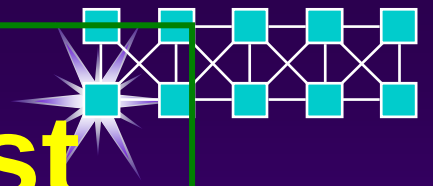
template can be implemented on a

CNN Universal Chip

(0.18μ technology) in approximately

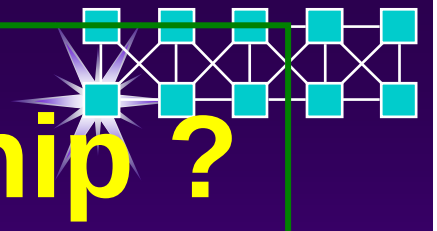
1 nano second.

CNN Chip is very Fast



A typical *instruction* can be implemented on a ***CNN chip*** in less than the time for *light* to travel 1 foot.

How Fast is the CNN Chip ?



The

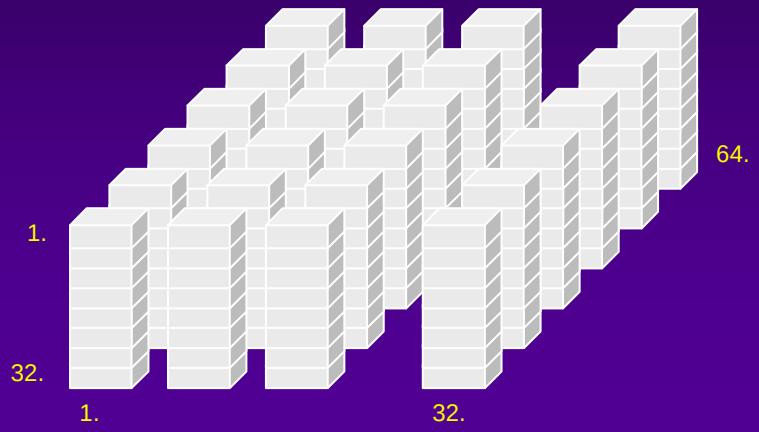
CNN Universal Chip

can execute

one billion instructions

in *one second*.

IBM Cellular Supercomputer 2002



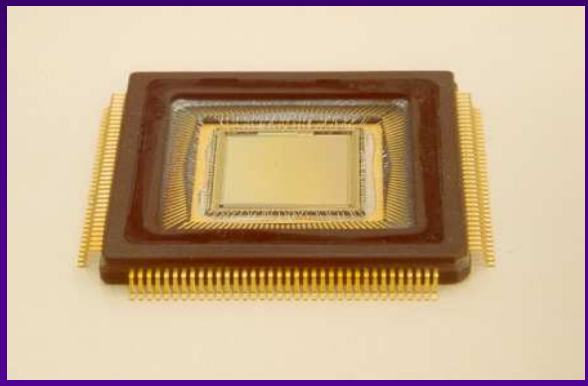
65536 (32 x 32 x 64) Power PC

Computing Power ~ 12 x 10^{12} (TeraFLOPS)

A = 65536 x 1.06 cm^2 = 6.9468 m^2

P = 491 kW

CNN Universal Chip

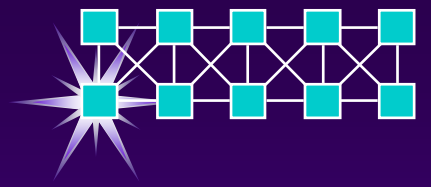


128 x 128 processor with optical input

Computing Power ~ 12 x 10^{12} (TeraOPS)

A = 1.4 cm^2

P = 4.5 W

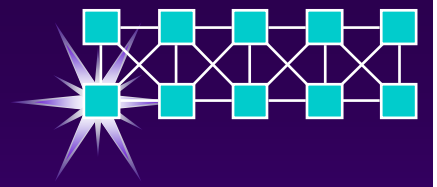


In what sense
is the
CNN Universal Chip
a
Supercomputer ?

The CNN Universal Chip is Functionally Equivalent to the IBM Cellular Supercomputer

The 128 x 128 **CNN universal chip** (circa 2003) is functionally equivalent to the **IBM Cellular Supercomputer** (circa 2002) in the sense that there are currently many **mission-critical** problems which can only be solved **in real time** by using this high-end IBM supercomputer, or by using the **CNN chip**.

CNN



offers 3 orders of magnitude higher performances in

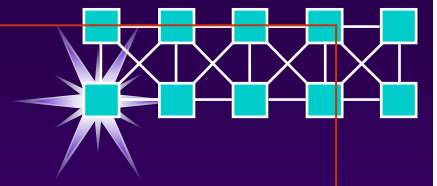
Speed

Power

Area

Cellular Computer Architecture is the Future

For *nanoscale computers*, communicating with far away processors is very inefficient because it takes many clock cycles to implement a computing instruction. Future nanocomputing calls for a *cellular multi-core architecture* where each core processor communicates only with its local neighbors, thereby enabling massively-parallel computations during each clock cycle.



SUMMARY

1) Parallel structure and Parallel Processing

2) Analog Processing

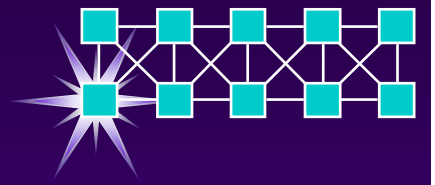
3) A/D convertor is not needed

4) Terra operation (10^12) speed

5) Processing is determined by the given template

Potential application areas

- Vision System of AGV
- Vision System of Robots
- Television image enhancement
- Industrial quality control
- Security
- Traffic control
- Medical imaging
- Image Compression
- Target Tracking
- Cruise Missile



Thank You Very Much