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 Condition

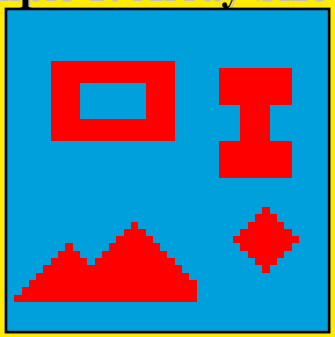
Fixed: $x_{i*j*} = 0$, $u_{i*j*} = 0$

*(i*j* denotes boundary cells)*

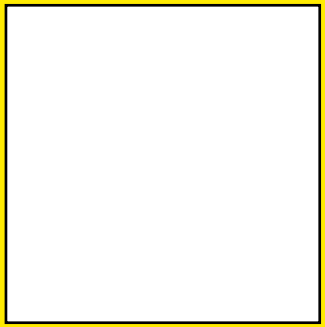
Initial State

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

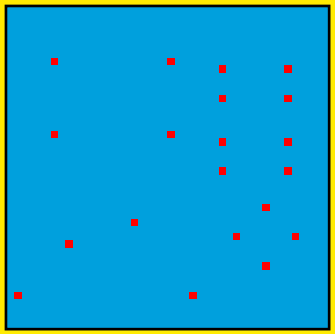
Example 1: Array Size = 44 x 44



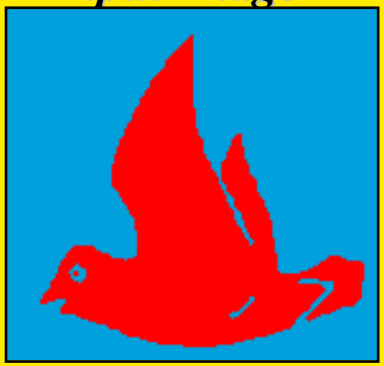
Input Image

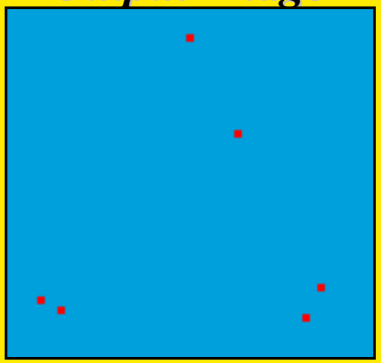


Initial State



Output Image





Example 2: Array Size = 140 x 140

Name: RIGHT EDGE DETECTION CNN

Task Prescription: *Extract right edges of all binary objects which are at least two-pixel wide along the horizontal direction.*

Cloning Template

z : -2

B :

0	0	0
1	1	-1
0	0	0

A :

0	0	0
0	1	0
0	0	0

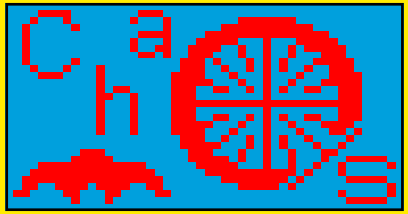
Boundary Condition

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

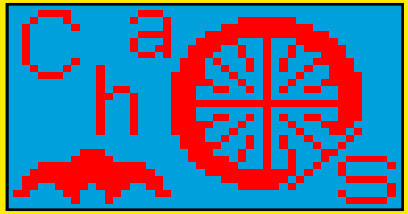
Initial State

$x_{ij}(0) = u_{ij}$ = *input image*

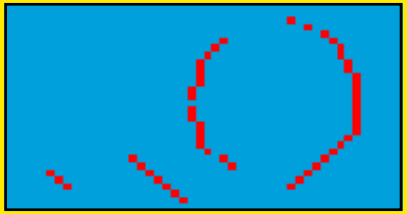
Example 1: Array Size= 30 x 48



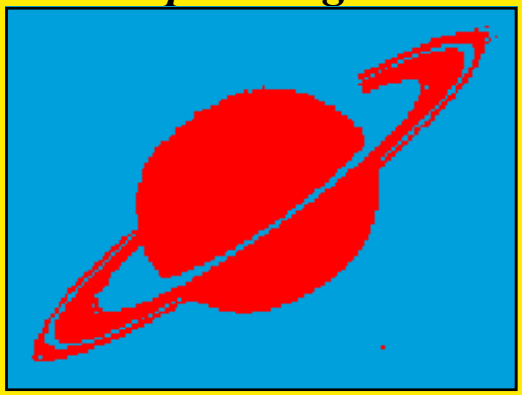
Input Image

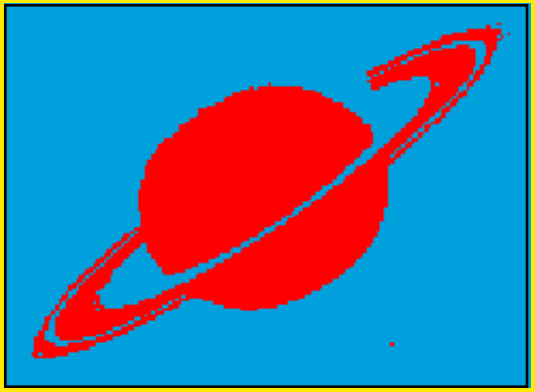


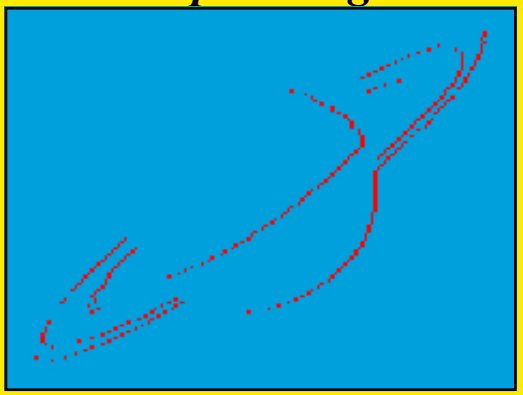
Initial State



Output Image







Example 2: Array Size = 176 x 176

Name:

EROSION CNN

Task Prescription:

Peel off all boundary pixels of binary image objects. Pixels are considered to belong to object boundary if the structuring element, coded by the ***B*** *template, does not fit completely within the object. EROSION is the morphological complement of DILATION. For the structuring* ***B*** *template shown below, the CNN simply* ***peels off*** *from the input image* ***one layer*** *of black pixels (red in pseudo color) on the* ***perimeter*** *of all black objects.*

Cloning Template

z : -4.5

B :

0	1	0
1	1	1
0	1	0

A :

0	0	0
0	2	0
0	0	0

Boundary Condition

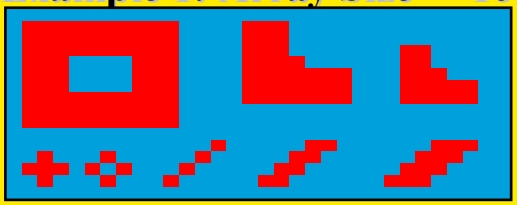
Fixed: $x_{i^*j^*} = -1$, $u_{i^*j^*} = -1$

(i^*j^* denotes boundary cells)

Initial State

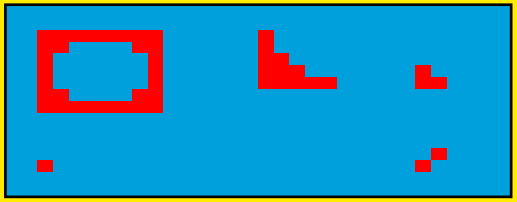
$x_{ij}(0) = 0$ *(white in pseudo-color)*

Example 1: Array Size = 16 x 32

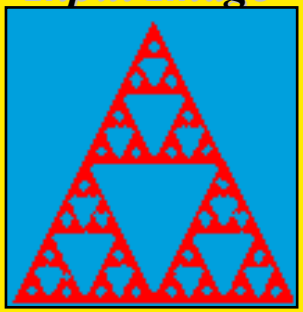


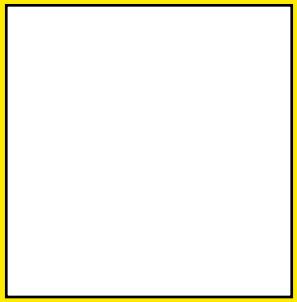
Input Image

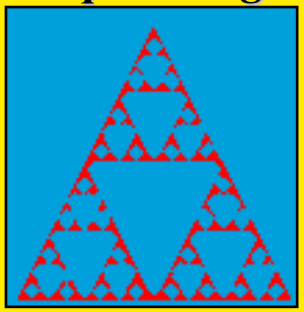
Initial State



Output Image







Example 2: Array Size = 100 x 100

Name:

DILATION CNN

Task Prescription: *Grow a layer of pixels around objects in a binary image in a way determined by structuring element coded by the **B** template. DILATION is the morphological complement of EROSION. For the structuring **B** template shown below, the CNN simply **adds** onto the input image **one layer** of black pixels (red in pseudo color) on the **perimeter** of all black objects.*

Cloning Template

z : 4.5

B :

0	1	0
1	1	1
0	1	0

A :

0	0	0
0	2	0
0	0	0

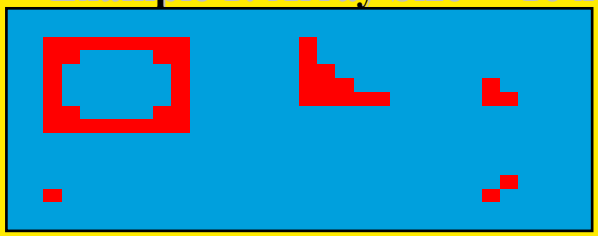
Boundary Condition

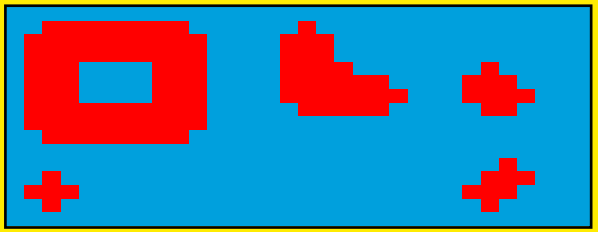
Fixed: $x_{i*j*} = -1$, $u_{i*j*} = -1$
*(i*j* denotes boundary cells)*

Initial State

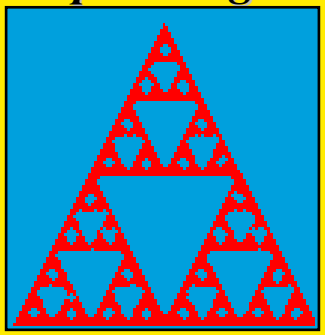
$x_{ij}(0) = 0$ (white in pseudo-color)

Example 1: Array Size = 16 x 32

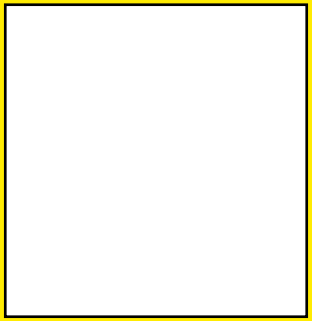




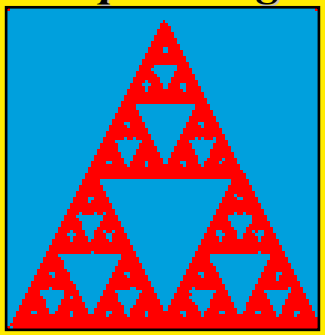
Input Image



Initial State



Output Image



Example 2: Array Size = 100 x 100

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

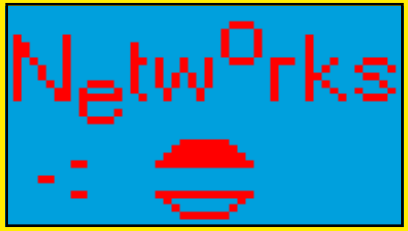
Fixed: $x_{i*j*}=0$, $u_{i*j*}=0$

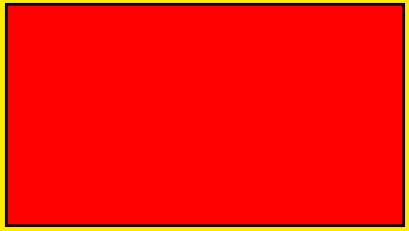
*(i*j* denotes boundary cells)*

Initial State

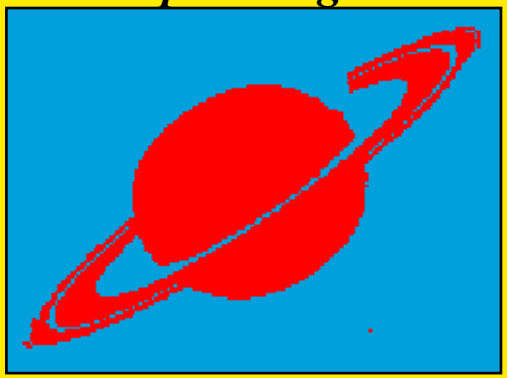
$x_{ij}(0) = 1$ *(red in pseudo-color)*

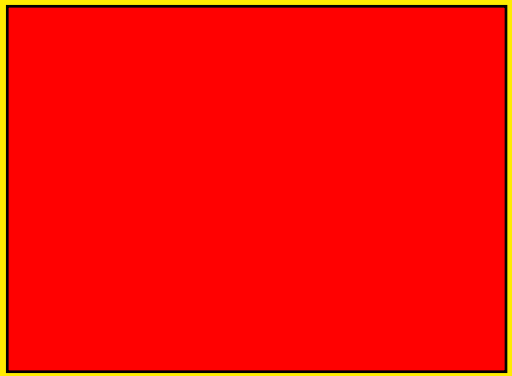
Example 1: Array Size = 30 x 48

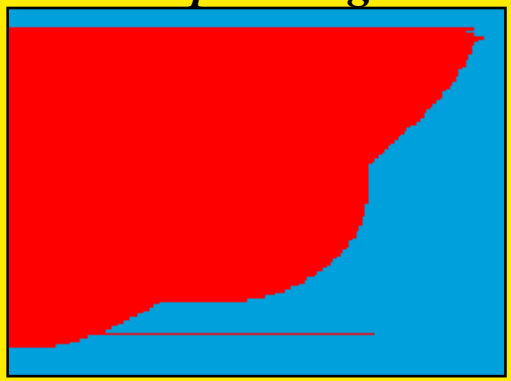




Input Image ***Initial State*** ***Output Image***







Example 2: Array Size = 176 x 176

Name: HORIZONTAL HOLE DETECTION CNN

Task Prescription: *Detect the number of horizontal holes from each horizontal row of a binary image* **P** *(loaded as initial state) where objects are coded as "red" pixels and the background is coded in "blue". A string of adjacent "blue" pixels in a horizontal row "i" of* **P** *is called a horizontal hole of horizontal row "i" if each end is terminated by at least one "red" pixel. The prescribed task of this CNN is first to replace each adjacent group of blue (respectively, red) pixels into a single blue (respectively, red) pixel and then to push the resulting subset of contracted blue (respectively, red) pixels into adjacent alternating rows towards the right boundary of the image. The number of horizontal holes is found by counting the number of blue pixels in the contracted string.*

Cloning Template

z :

0

B :

0	0	0
0	0	0
0	0	0

A :

0	0	0
1	2	-1
0	0	0

Boundary Condition

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

Initial State

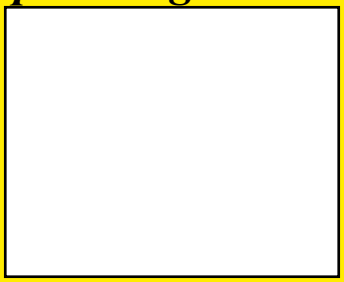
$x_{ij}(0)$ = *a given binary image*

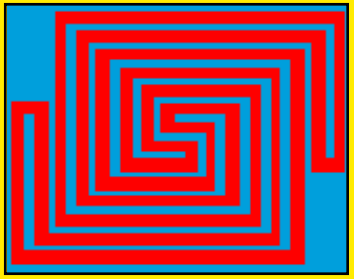
Example 1: Array Size = 30 x 48

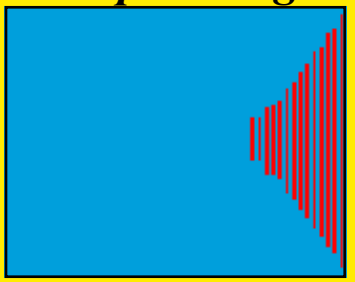
Input Image: arbitrary

Initial State

Output Image







Example 2: Array Size = 100 x 100

Name: VERTICAL HOLE DETECTION CNN

Task Prescription: *Detect the number of vertical holes from each vertical column of a binary image* **P** *(loaded as initial state) where objects are coded as "red" pixels and the background is coded in "blue". A string of adjacent "blue" pixels in a vertical column "i" of* **P** *is called a vertical hole of vertical column "i" if each end is terminated by at least one "red" pixel. The prescribed task of this CNN is first to replace each adjacent group of blue (respectively, red) pixels into a single blue (respectively, red) pixel and then to push the resulting subset of contracted blue (respectively, red) pixels into adjacent alternating columns towards the lower boundary of the image. The number of vertical holes is found by counting the number of blue pixels in the contracted string.*

Cloning Template

z :

0

B :

0	0	0
0	0	0
0	0	0

A :

0	1	0
0	2	0
0	-1	0

Boundary Condition

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

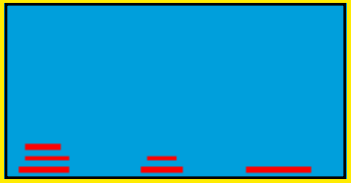
Initial State

$x_{ij}(0)$ = *a given binary image*

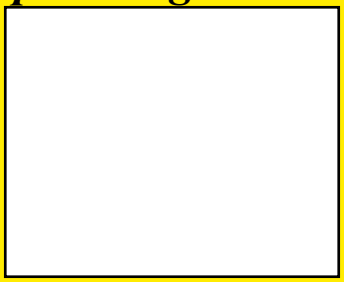
Example 1: Array Size = 30 x 48

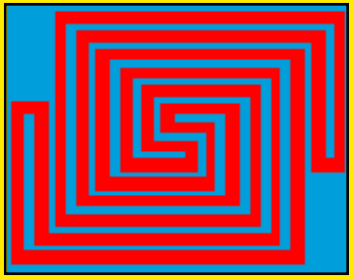
Input Image: arbitrary

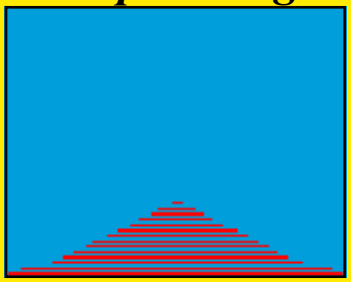
Initial State



Output Image







Example 2: Array Size = 100 x 100

Name: DIAGONAL HOLE DETECTION CNN

Task Prescription: *Detect the number of diagonal holes from each diagonal line of a binary image* **P** *(loaded as initial state) where objects are coded as "red" pixels and the background is coded in "blue". A string of adjacent "blue" pixels in a diagonal line "i" of* **P** *is called a diagonal hole of the line "i" if each end is terminated by at least one "red" pixel. The prescribed task of this CNN is first to replace each adjacent group of blue (respectively, red) pixels into a single blue (respectively, red) pixel and then to push the resulting subset of contracted blue (respectively, red) pixels into adjacent alternating rows and columns towards right and lower boundaries of the image. The number of diagonal holes is found by counting the number of blue pixels in the contracted string.*

Cloning Template

z :

0

B :

0	0	0
0	0	0
0	0	0

A :

1	0	0
0	2	0
0	0	-1

Boundary Condition

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

Initial State

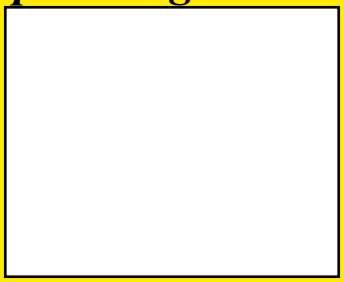
$x_{ij}(0)$ = *a given binary image*

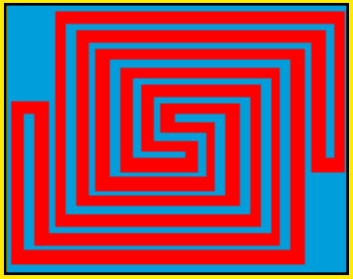
Example 1: Array Size = 30 x 48

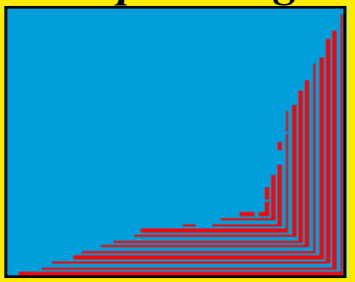
Input Image: arbitrary

Initial State

Output Image







Example 2: Array Size = 100 x 100

Name: DIAGONAL LINE DETECTION CNN

Task Prescription: *Extract all lines which have a slope of 45° from skeletonized binary images.*

Cloning Template

z : -1.5

B :

-2	0	2
0	0	0
2	0	-2

A :

0	0.75	0
0.75	2	0.75
0	0.75	0

Boundary Condition

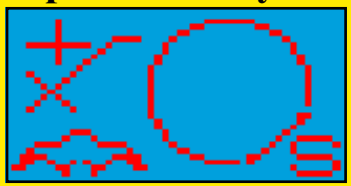
Fixed: $x_{i*j*}=0$, $u_{i*j*}=0$

*(i*j* denotes boundary cells)*

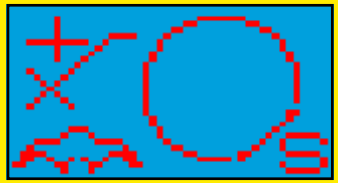
Initial State

$x_{ij}(0) = u_{ij}$ = *input image*

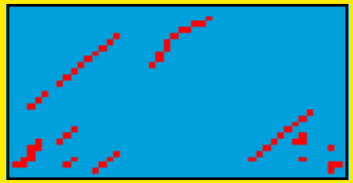
Example 1: Array Size = 30 x 48



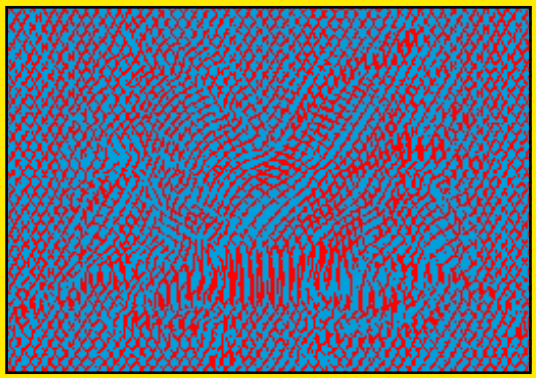
Input Image

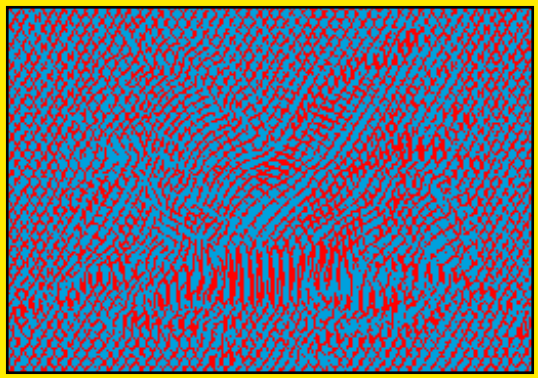


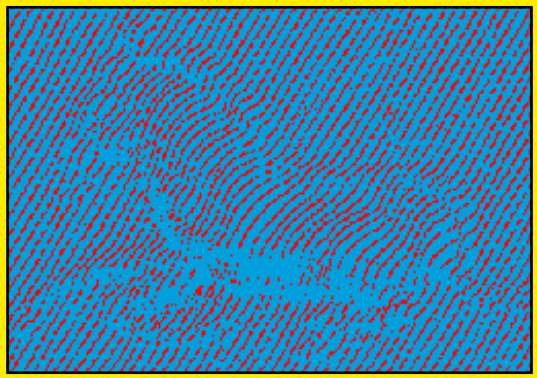
Initial State



Output Image







Example 2: Array Size = 256 x 256

Name: SELECTED OBJECTS EXTRACTION CNN

Task Prescription: *Extract marked objects from a binary image loaded as input. An object is any set of 8-connected black (red in pseudo-color) pixels. Markers are black pixels of another binary image loaded as CNN initial state. An object is considered to be marked if one of its elements overlaps with a marker.*

Cloning Template

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

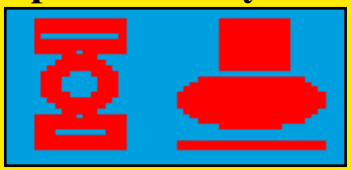
Boundary Condition

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

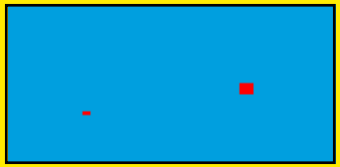
Initial State

$x_{ij}(0)$ = *another binary image*

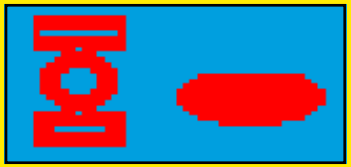
Example 1: Array Size = 30 x 48



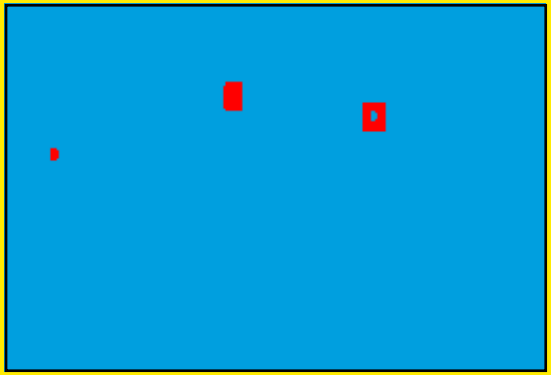
Input Image

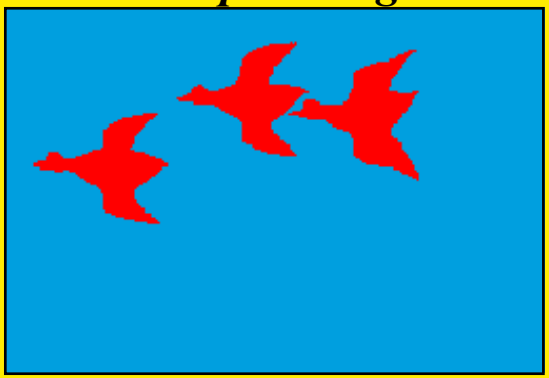


Initial State



Output Image





Example 2: Array Size = 176 x 176