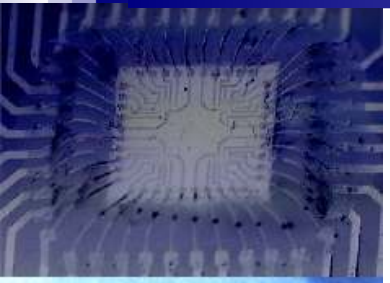
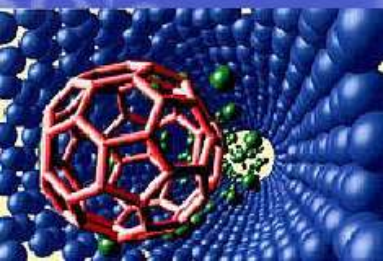






# Algoritmusok implementálása CNN-UM-en

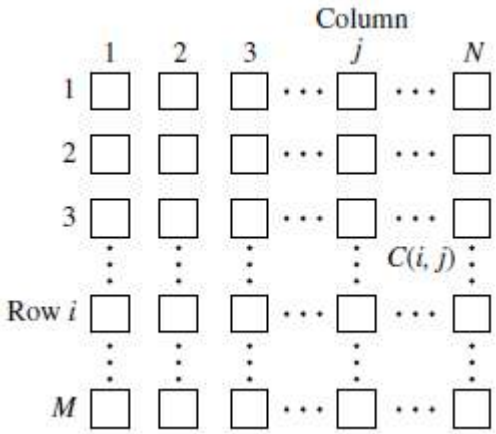
*András Horváth*

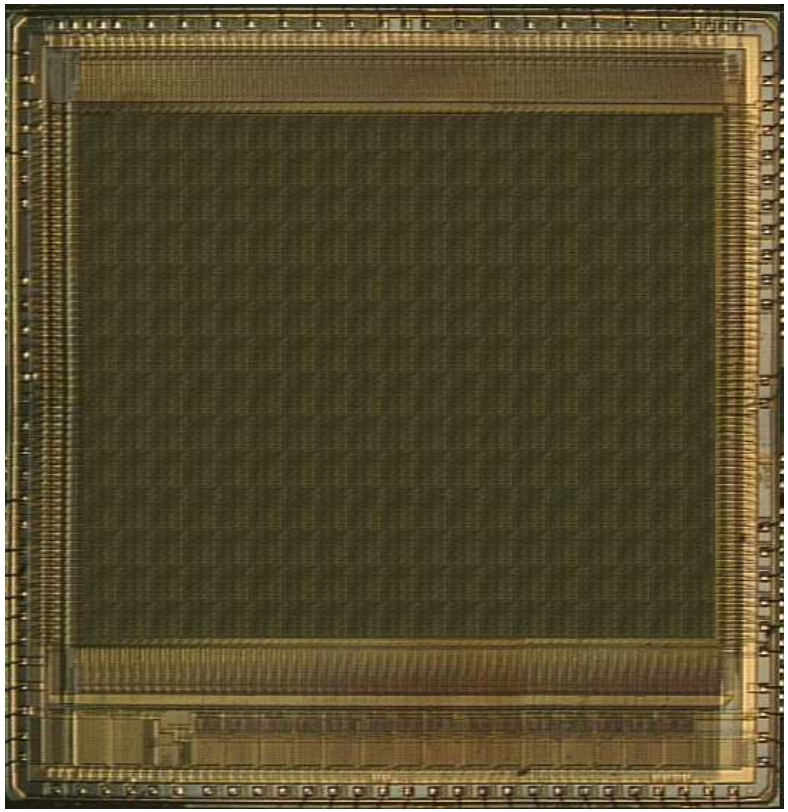
# A CNN architektúra

M*N-es cellák reguláris hálózata

C(i,j)
*(i, j )*, *i* = 1, 2, . . . , *M*, *j* = 1, 2, . . . , *N*

*M nem szükségszerűen egyenlő* N

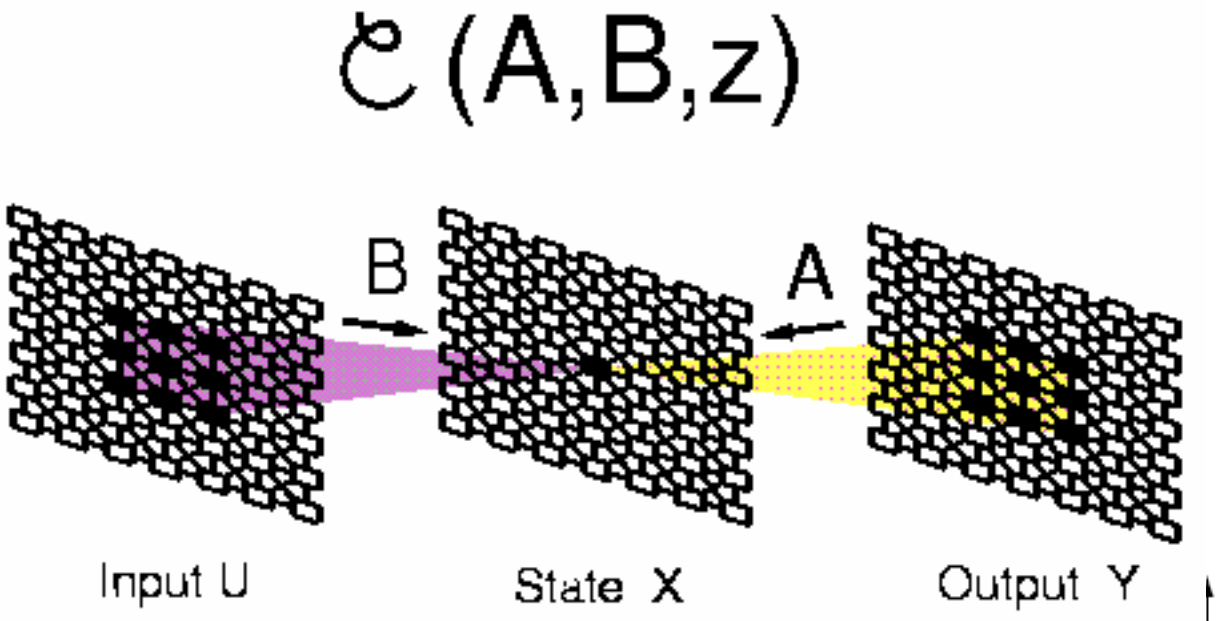


# CNN állapotegyenlet

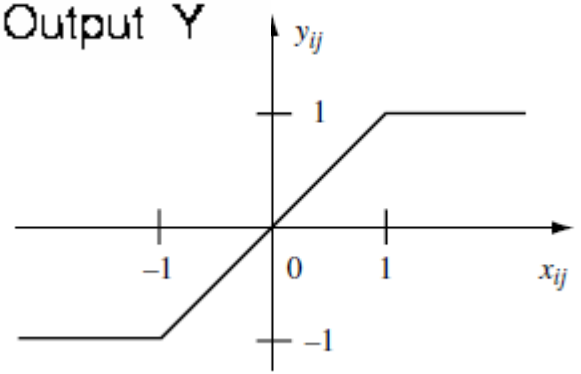
*1 State equation*

$$\dot{x}_{ij} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)\,y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)\,u_{kl} + z_{ij}$$




*2 Output equation*

$$y_{ij} = f(x_{ij}) = \frac{1}{2}|x_{ij}+1| - \frac{1}{2}|x_{ij}-1|$$



# CNN-UM

CNN univerzális gép:

Algortimus tervezéséhez (Turing teljes), feladatok megvalósításához (milyen feladat megoldására hatékony)

Nem minden algoritmus valósítható meg egyenlet template-tel

Minden algoritmus megvalósítható tempalte-ek sorozataként.

Egymás után csatolt lépések és elágazások megvalósítására van szükség.

# CNN-UM

Egyéb kiegészítő infrastruktúrákra van szükség az algoritmusok megvalósításához

El kell tárolnunk az egyes utasítások eredményeit és később azokat vissza kel ltudnunk tölteni és kominbálni más eredményekkel

Memóriára van szükség:

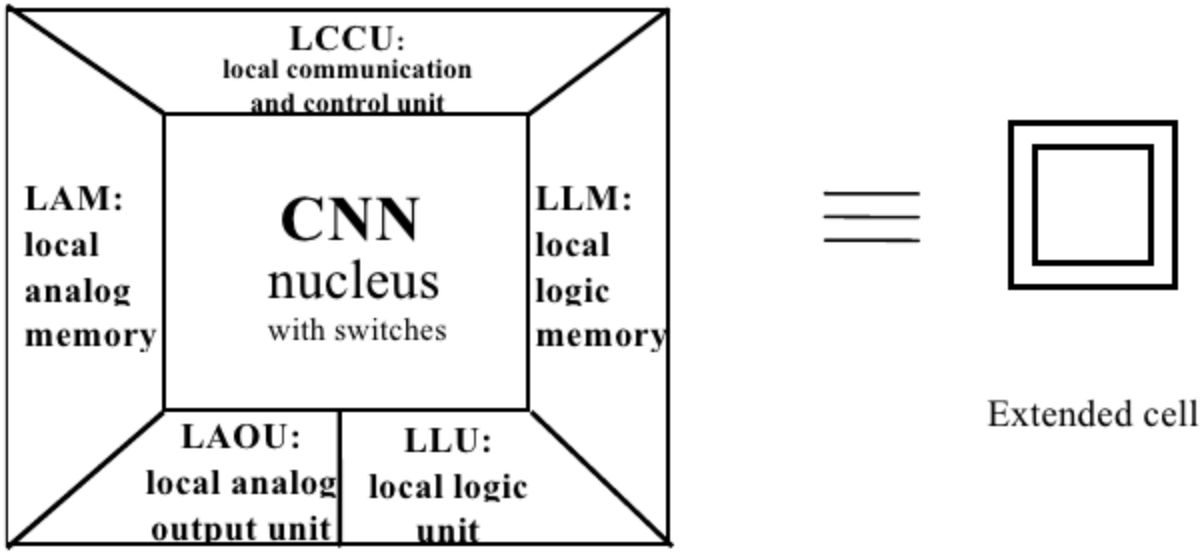
- lokális memória, elég a köztes eredmények tárolásához

- globális memória: Az utasítás sorozat leírásához, minden cellához eljut

# Extended CNN cell in the CNN-UM

Folytonos időben, folytonos állapotváltozóval, az ismert nemileáris dinamika

+Lokális memória

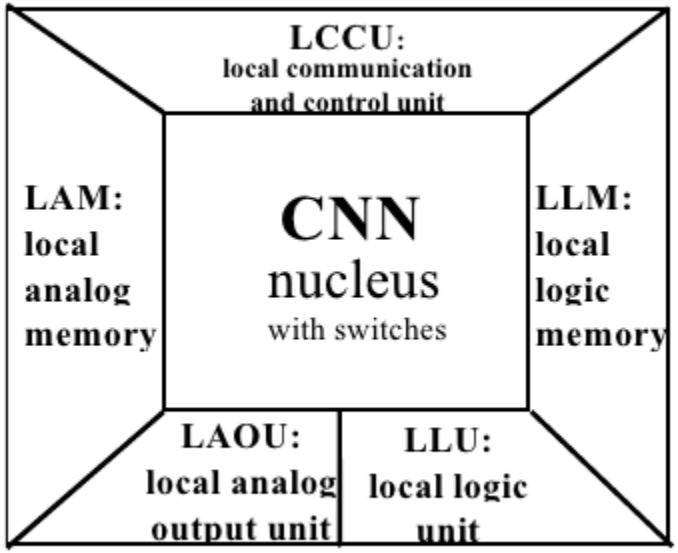


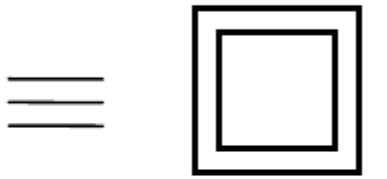
# LAOU

Local Analog Output Unit: analóg memória

Bemenetet, kimenetet kezeli a szürkeskálás bemenetekre, kimenetekre, állapotokra

Kombinálja (összeadás, kivonás) a lokálisan tárolt adatokat egy kimeneti vagy bemeneti képpé



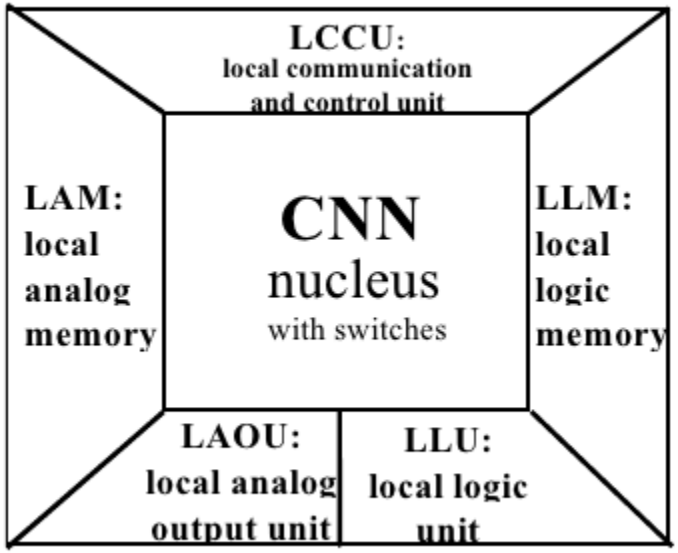
Extended cell

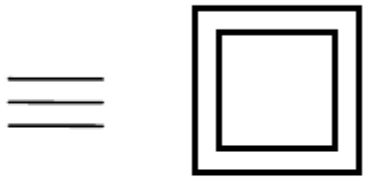
# LLU

Local Logic Unit

Bemenetet, kimenetet kezeli a bináris (fekete/fehér) bemenetekre, kimenetekre, állapotokra

Kombinálja (logikai műveletek: AND,OR...) a lokálisan tárolt adatokat egy kimeneti vagy bemeneti képpé



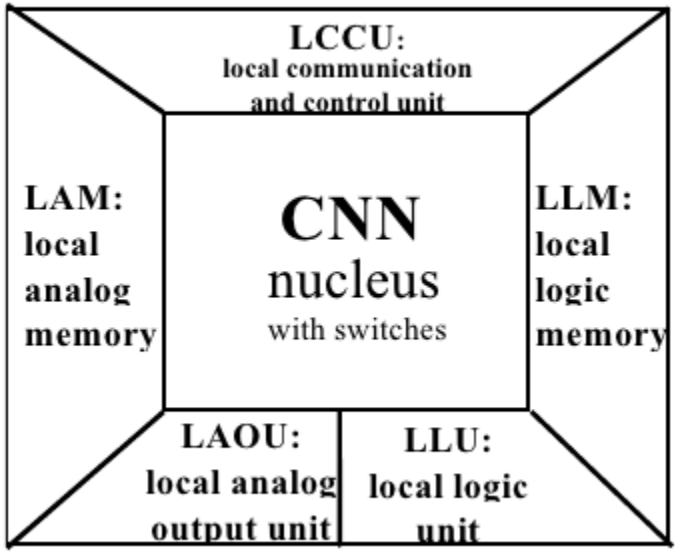



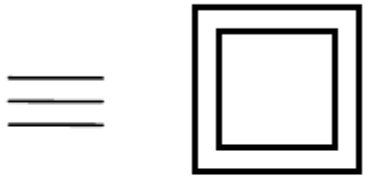
Extended cell

# LCCU

Local Communication and Control Unit.

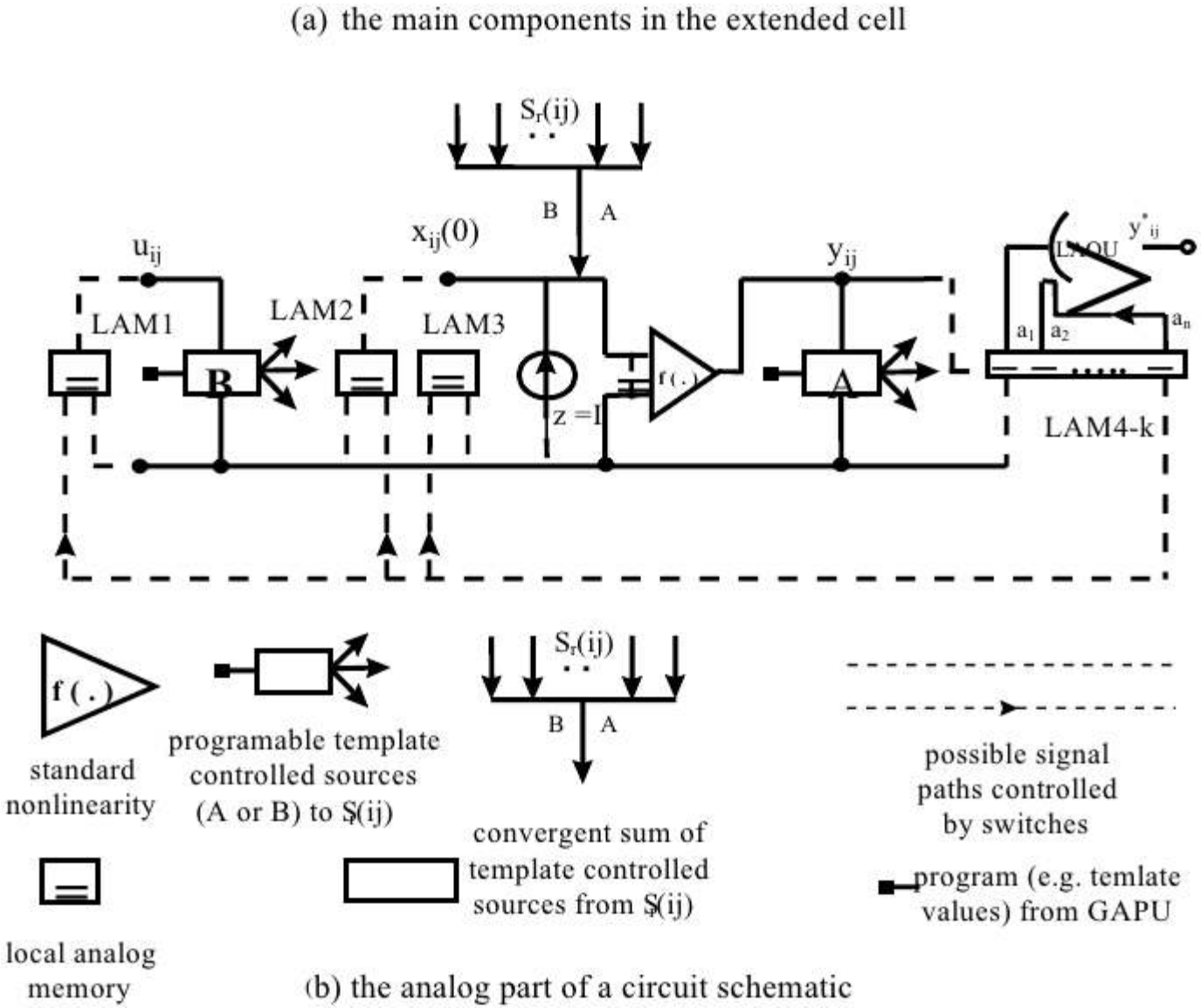
Communication Unit:Fogadja az utasításokat a Global Analog Programming Unit (GAPU)-tól és beállítja a szükséges paramétereket



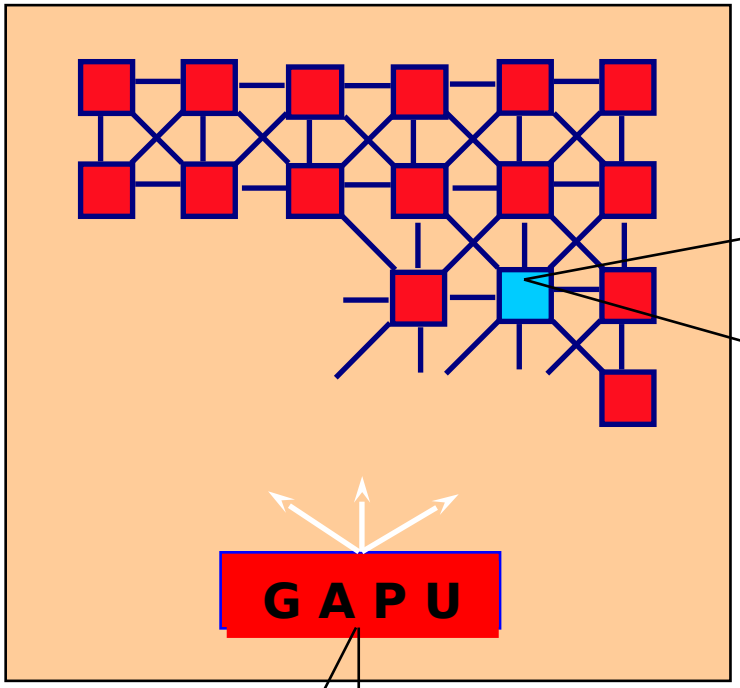



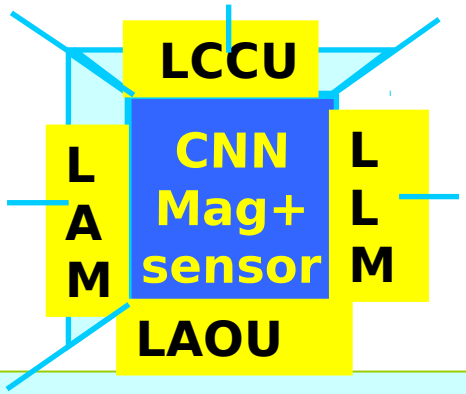
Extended cell

# CNN Cella áramköri modellje



# CNN Universal Machine (CNN-UM)




**GAPU**: Global Analogic Programming Unit

**LAM: Local Analog/Arithmetic Memory**
**LLM: Local Logic/Binary Memory**
**LCCU: Local Communication and Control Unit**
**LAOU: Local Analog/Arithmetiv Output Unit**
**LLU: Local Logic Unit**

**APR: Analog Program instr. Register**
**LPR: Logic Program Register**
**SCR: Switch Configuration Register**
**GACU: Global Analogic Control Unit**


# GAPU

Négy további részből áll:

- Analóg program regiszter
- Logikai program regiszter
- Switch configuration regiszter
- Global analog control unit

Globálisan kezeli az utasításokat, ezek elindítását, ütemezését (nincs közös órajel), elküldi az utasítások paramétereit és fogadja/feldolgozza a kimeneteket.


# Közös órajel nélkül?

Nincs közös órajel a cellák között, az indítás 'egyszerre' történik, de nincs kényszerített szinkronizáció

A számítás szinkron módon, de közös órajel nélkül történik

A futási idő becslése::

Minden jel korlátos, a korlátok ismertek

Tudunk korlátot becslést adni a tranziens időszak maximumára

Ez egy rendkívül hasznos trend az eszközfejlesztésben, nincs szükség közös frekvenciára, nem kell a frekvenciákat összehangolni

# Az Alpha nyelv

Standard nyelv, ami leírja az algoritmikus lépéseket

**Visual Feature Detection**
**(α-language, version 2.1)**

```
        FUNCTION BARS-UP;
xLoad (LLM1, BarsUpTest);
        LLM3:= LLM1;
        HOLLOW(LLM1,LLM1,LLM2,10,-1);
        LOGXOR(LLM2,LLM3,LLM1,10,-1);
        HORDIST(LLM1,LLM1,LLM2,10,-1);
        RECALL(LLM2,LLM3,LLM1,10,-1);
xSAVE(RESULT,LLM1);
        ENDFUNCT;
```

**Compiled Analogic Macro Code**
in hexadecimal format

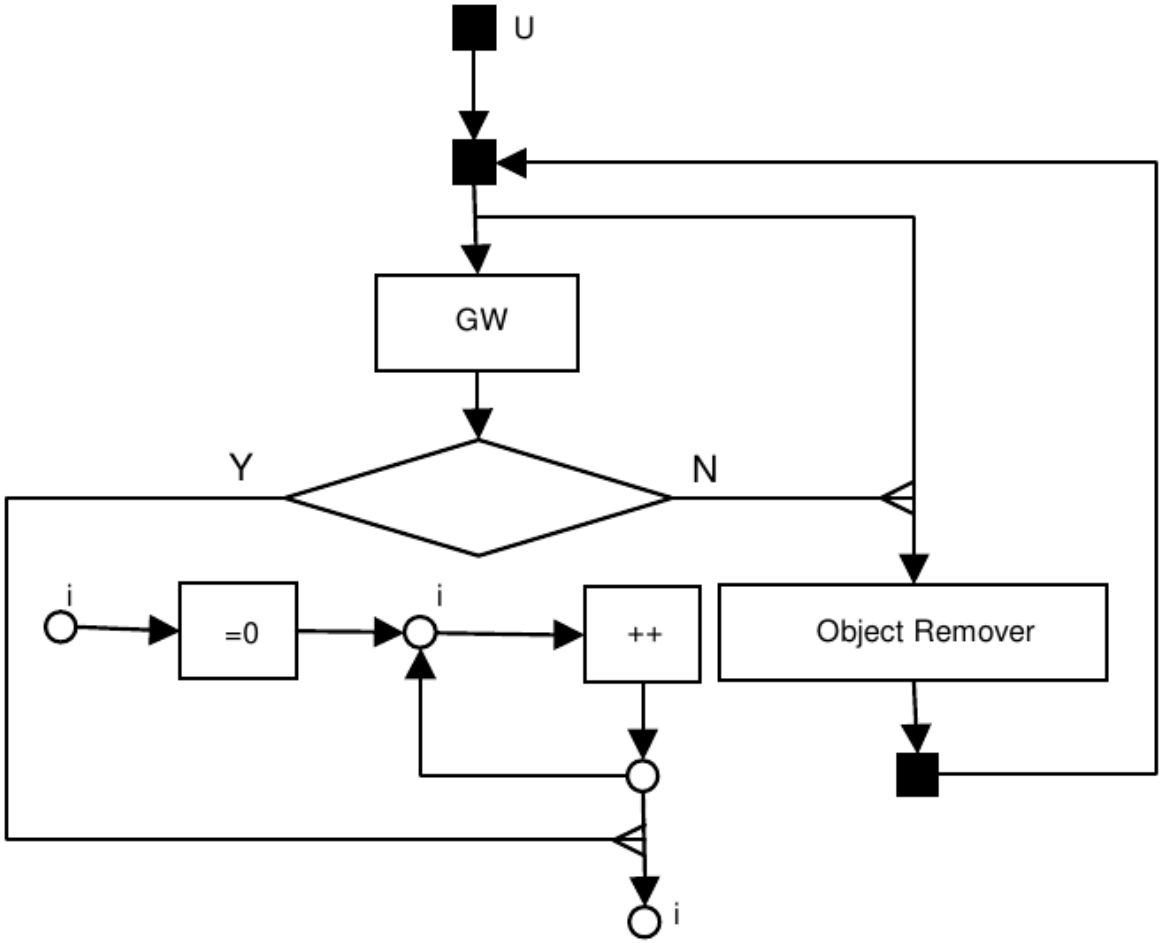
| hexa | binary | code |
|---|---|---|
| 12h | 0000 0000 0001 0010 | COPY |
| 8h | 0000 0000 0000 1000 | B2C_L2L |
| FFC0h | 1111 1111 1100 0000 | >FFC0 |
| 1h | 0000 0000 0000 0001 | 1 |
| 62h | 0000 0000 0001 0010 | LOADT |
| FFA0h | 1111 1111 1010 0000 | >FFA0 |
| 1h | 0000 0000 0000 0001 | 1 |
| 62h | 0000 0000 0001 0010 | LOADT |
| FF80h | 1111 1111 1010 0000 | >FF80 |
| 2h | 0000 0000 0000 0010 | 2 |

# Flow-chart, Diagram

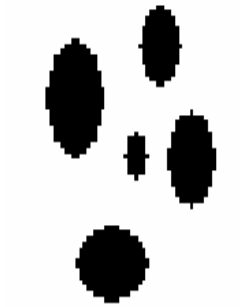
Könnyebb megérteni és átlátni az algoritmikus lépéseket

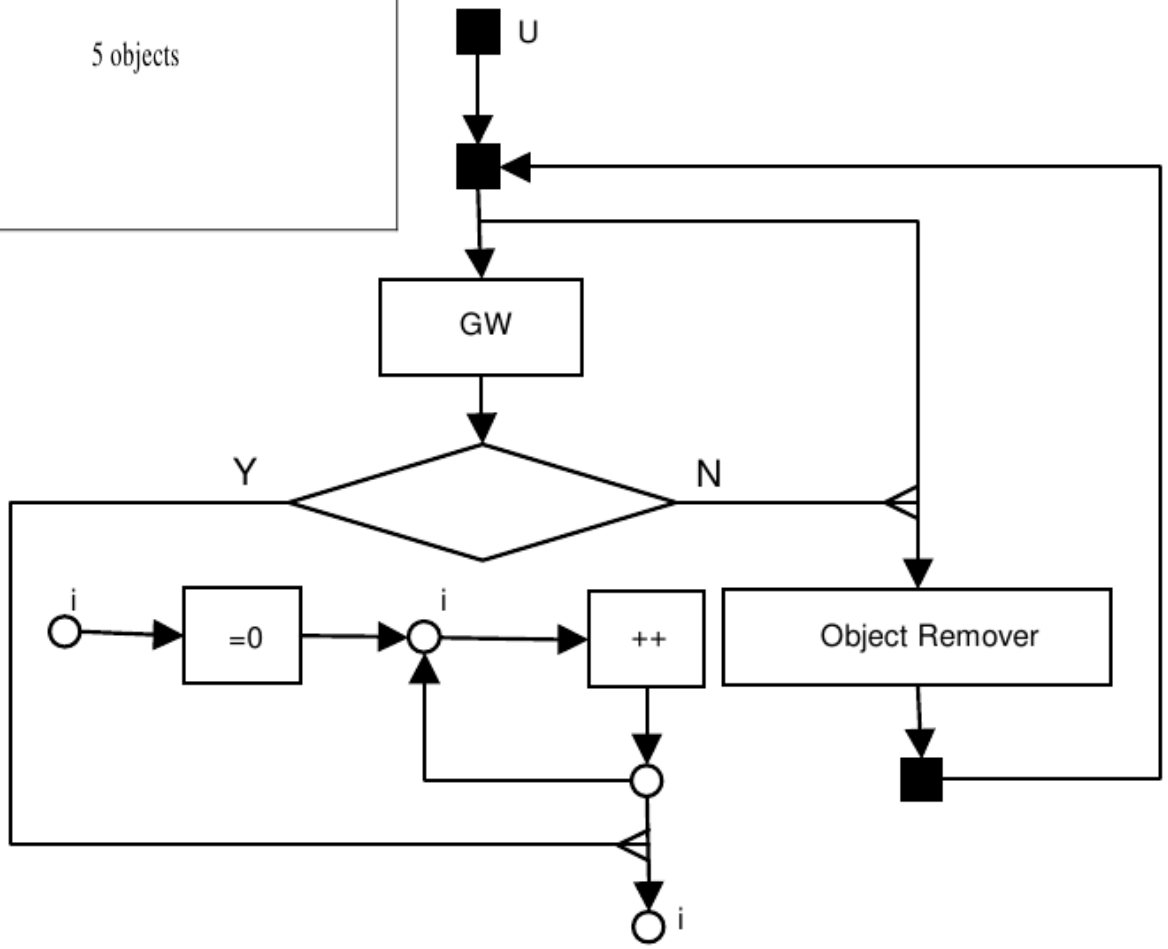
Komplex eljárásoknál nagy helyet igényel, nehéz átlátni

Kevésbé tömör leírás



# Objektum számlálás

| Input image | Output (scalar) |
| --- | --- |
|  | 5 objects |



Fides et Ratio

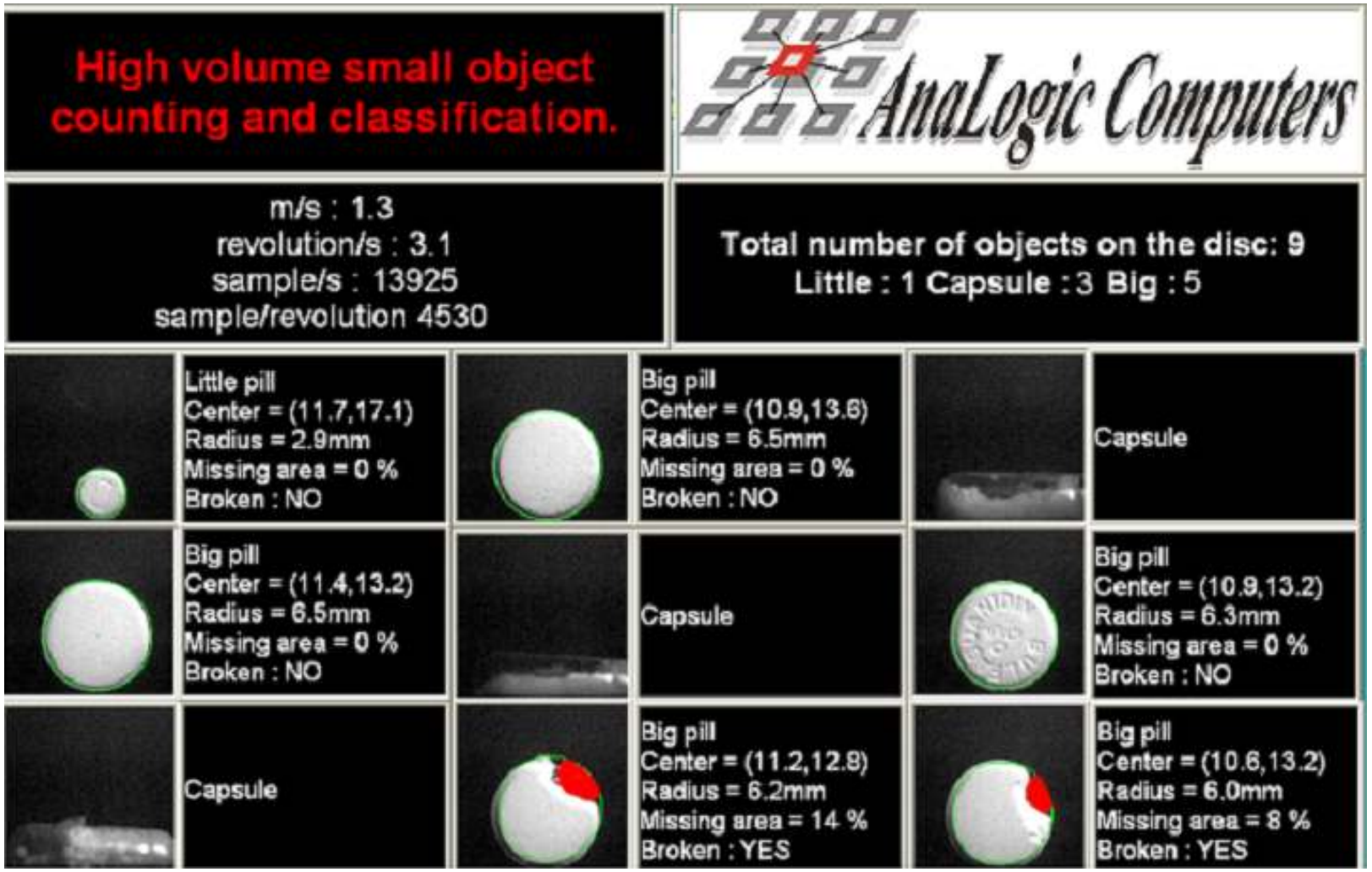
# Application example: Pills calssificaiton – iwth high speed

Aim: identification of pills on a conveyor belt

- 1000-10,000 decisions per sec

Fides et Ratio

# User Interface



# Application Example

Fides et Ratio

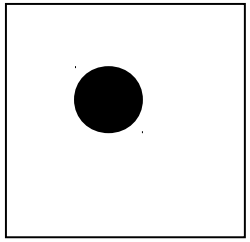
# 'Diameter' Calculation

Horizontal length

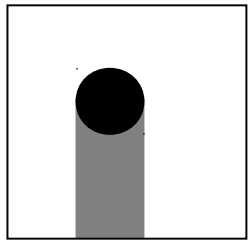
Vertical projection

Vertical length

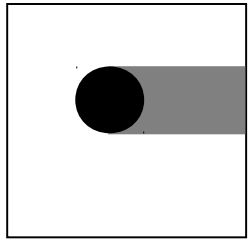
Horizontal projection



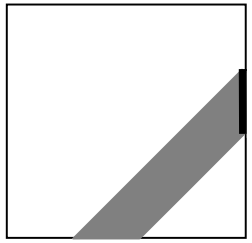
Original image



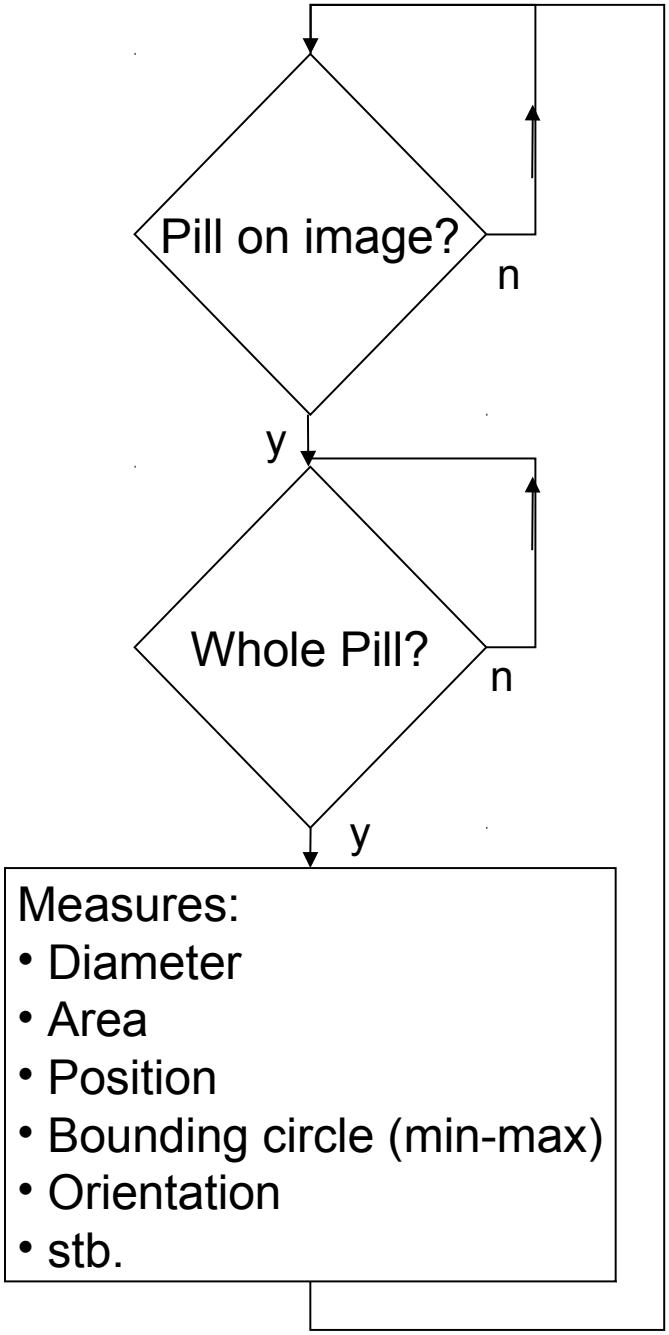
Vertical projection



Horizontal projection



Diagonal projection




# Detection of a Pill

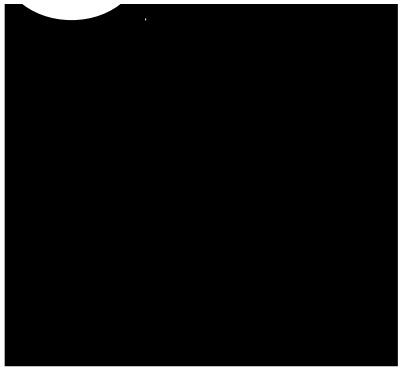
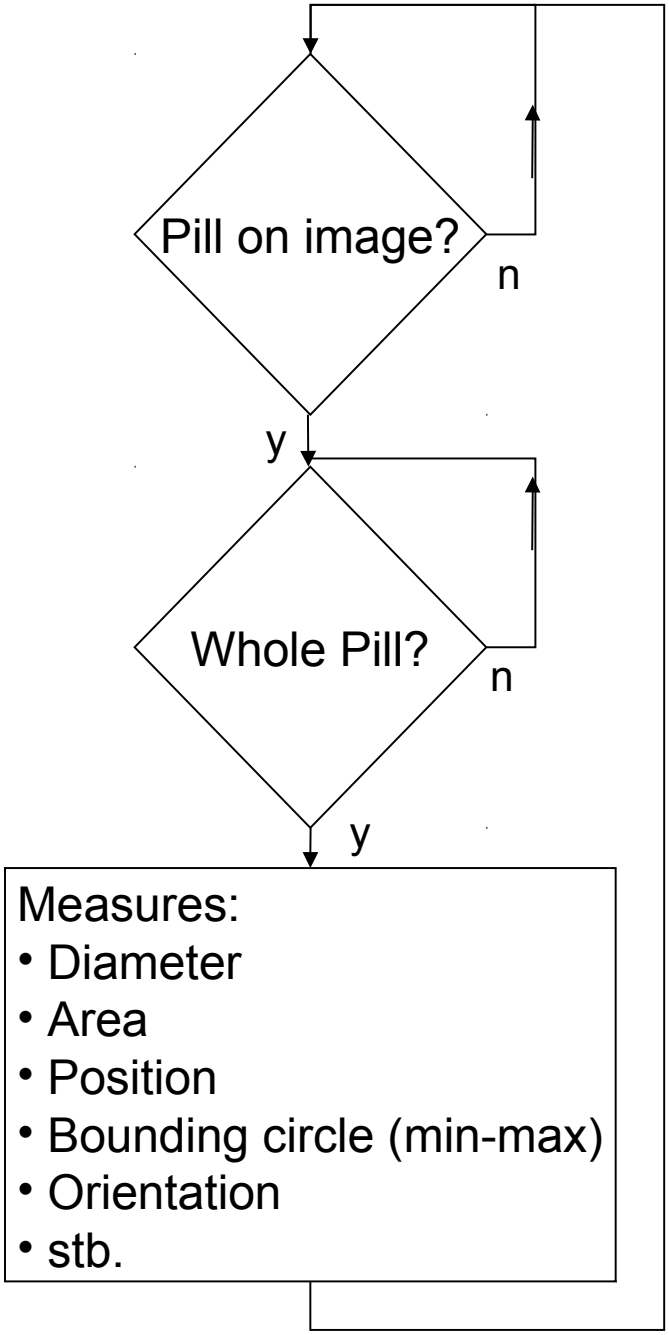


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

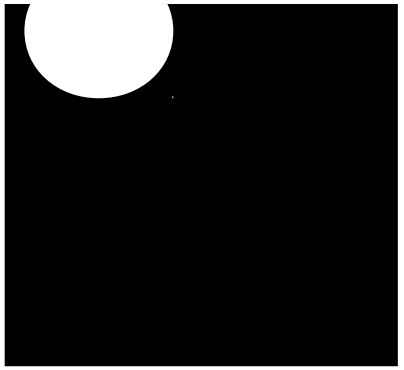
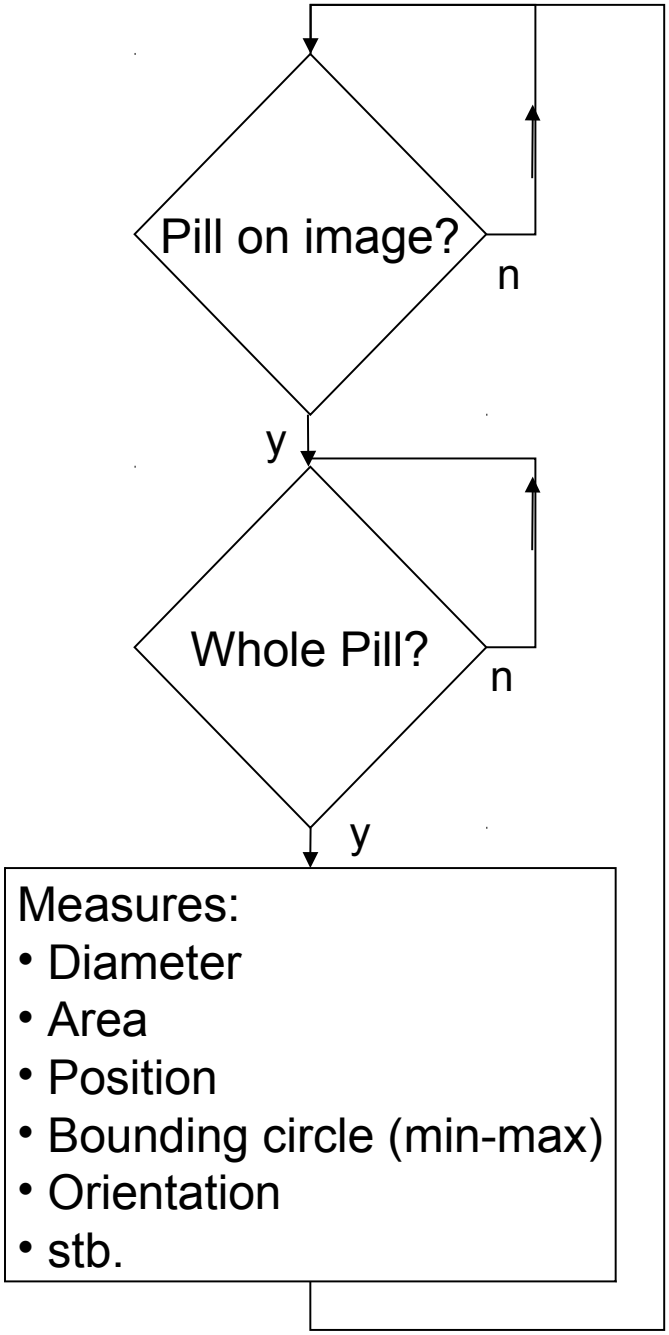


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

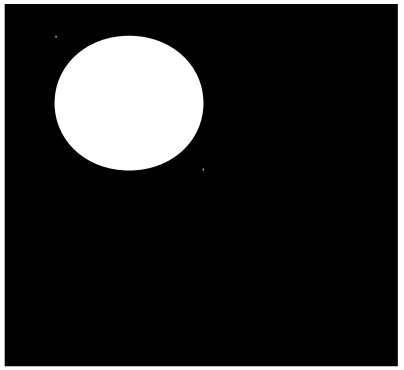
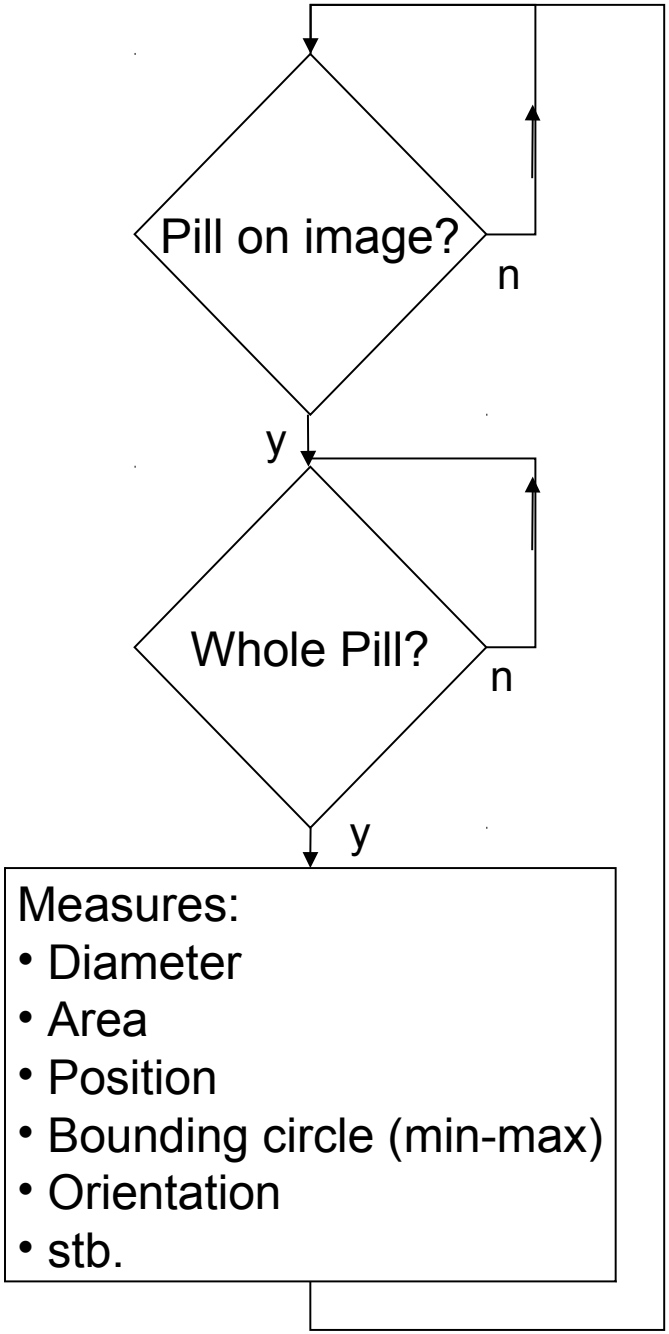


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

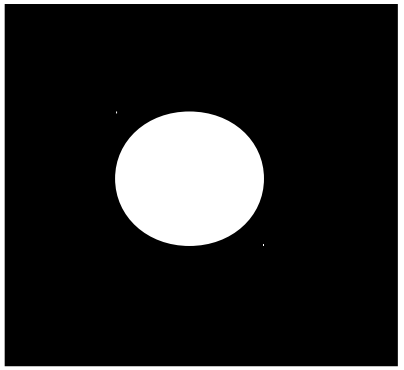
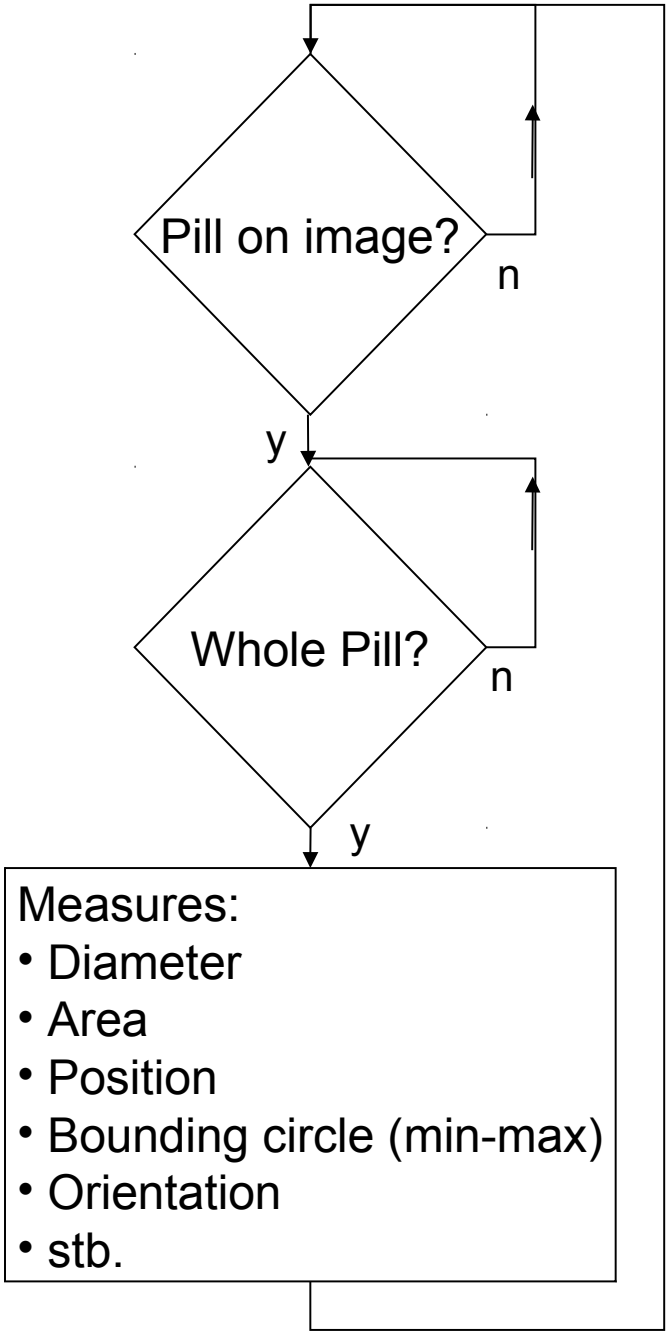


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

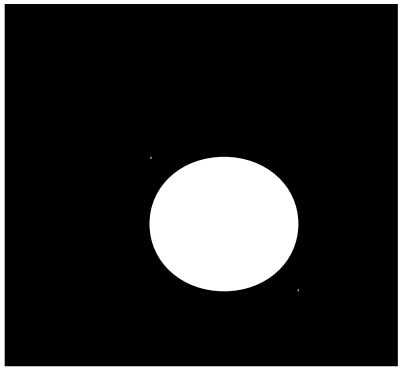
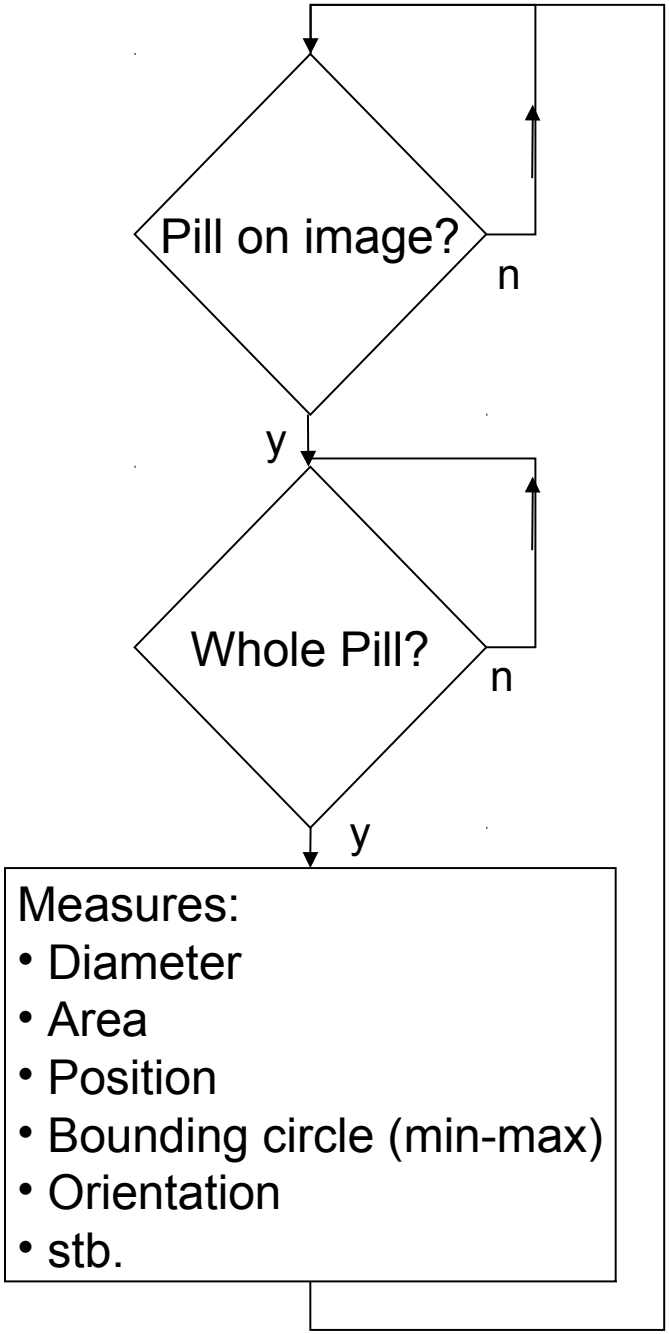


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

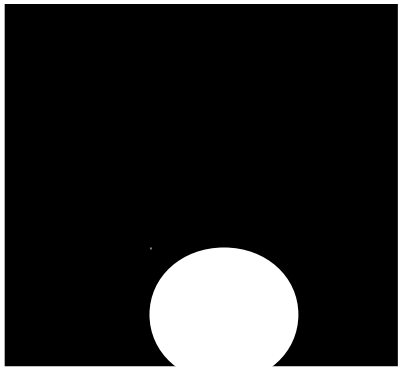
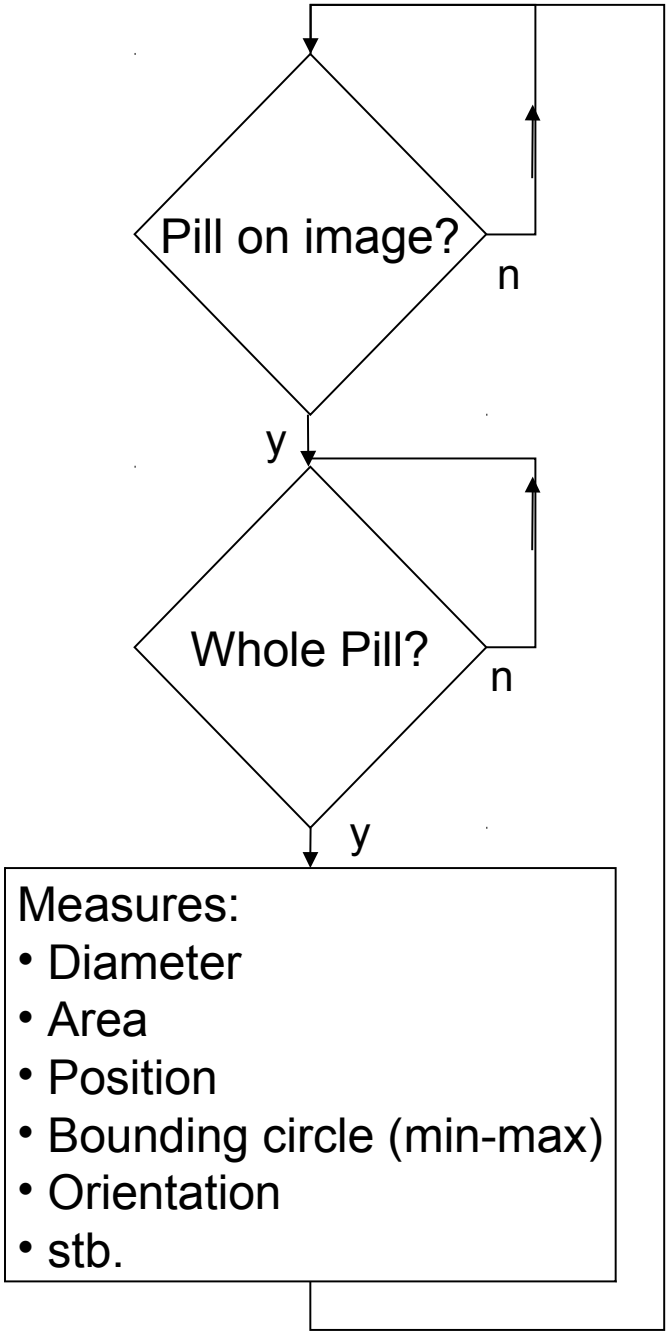


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

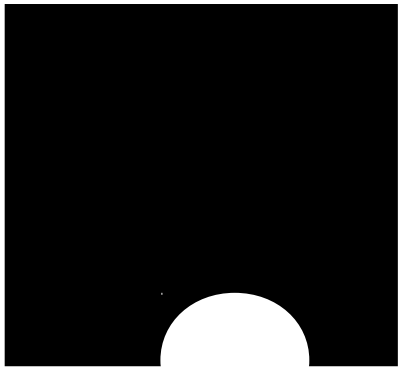
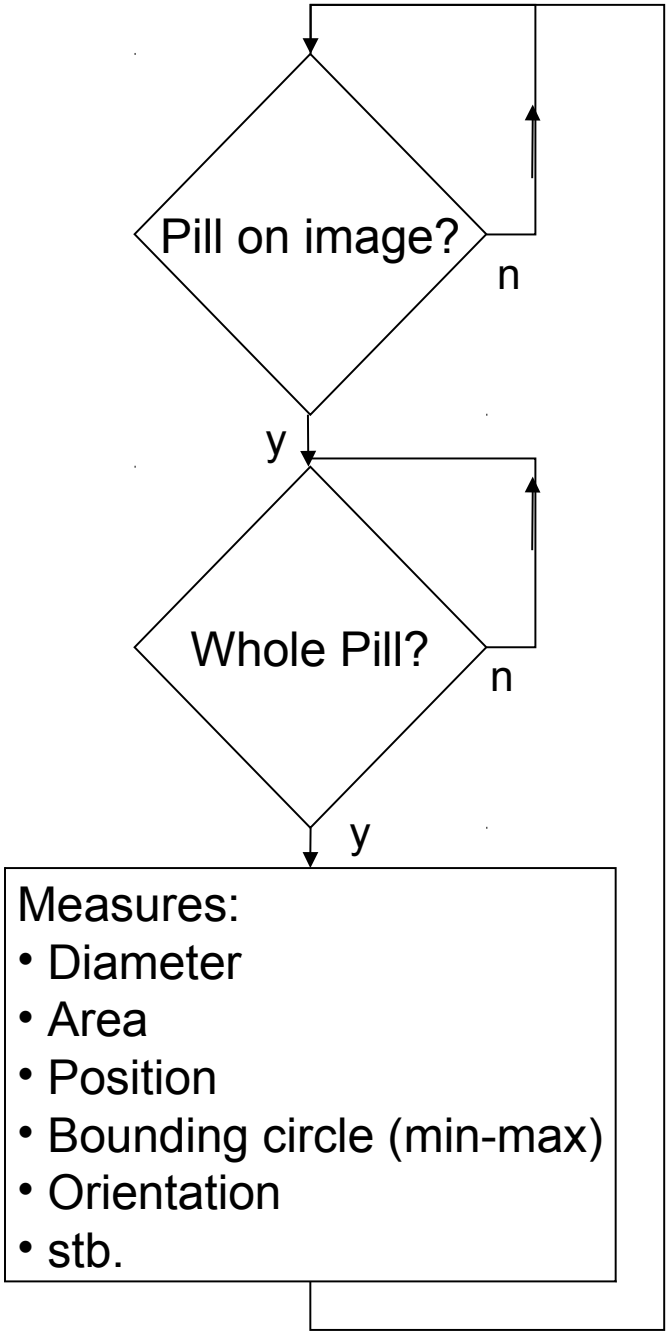
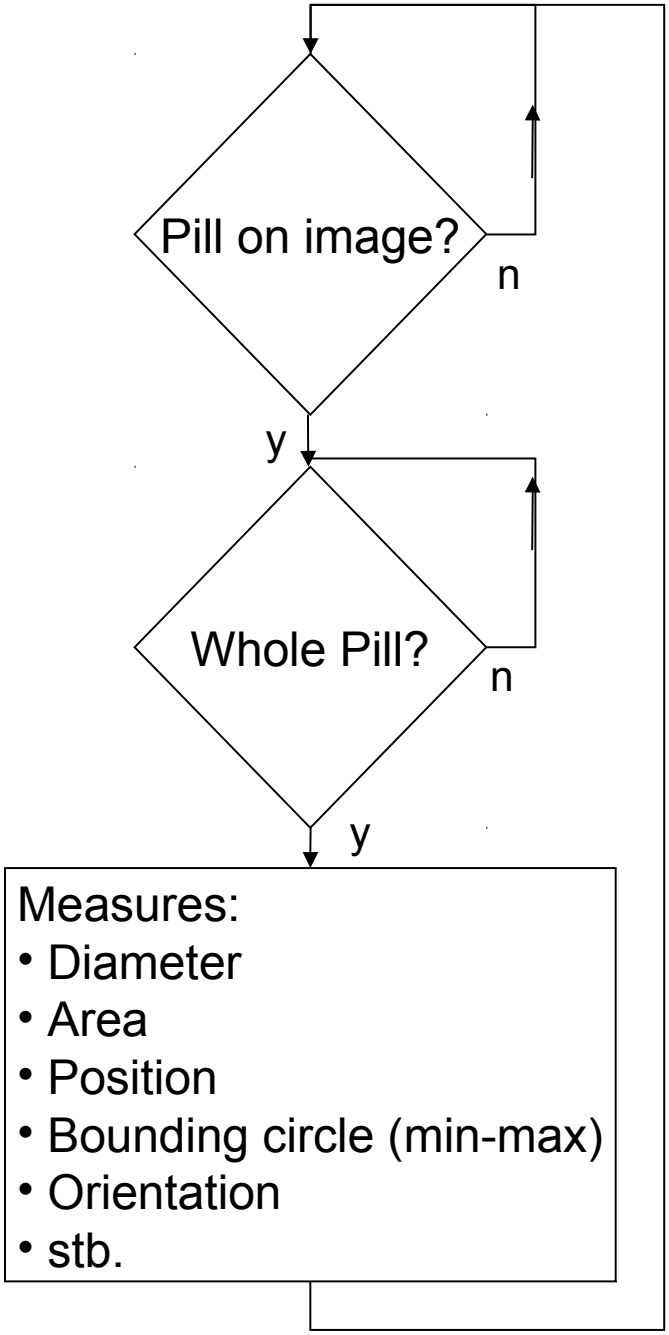


Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

Image sequence from ACE16k
*Black – Background*
*White- Pill*




# Detection of a Pill

Image sequence from ACE16k
*Black – Background*
*White- Pill*

# Whole Pill on Image?

Method 1 (~10 microsec):

Readout of the bottom row

If there is a pill, there are white pixels in the bottom row

•

Method 2 (~10 microsec):

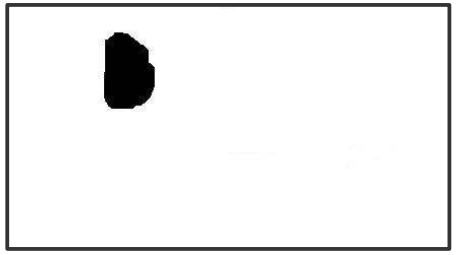
Readout of the bottom row

If the are no white (not too many) pixels in the bottom row, the whole pill can be seen on the image

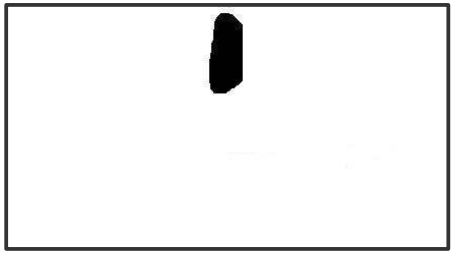
# Objektum számlálás

Bemeneti kép

Érdekes részlet



Object one



Object Two

Object Three

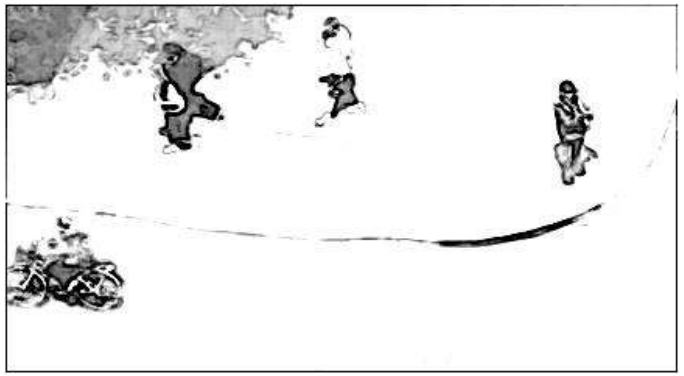
Object Four

# Objektum számlálás

Original image

Cropped image

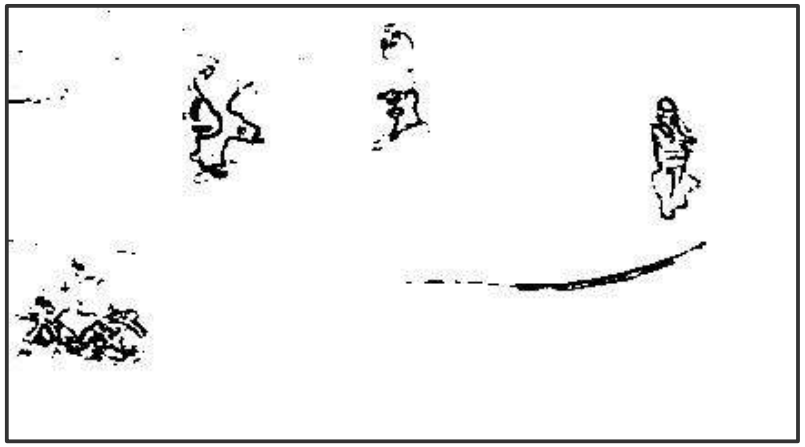
Input image



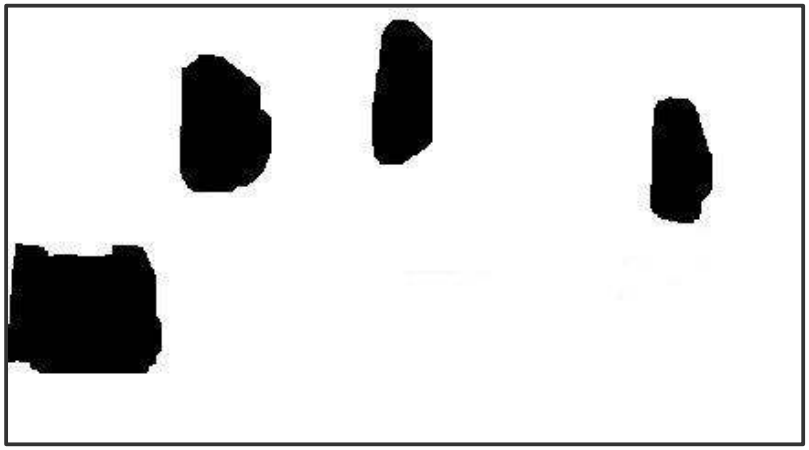
Local feature map

Fides et Ratio

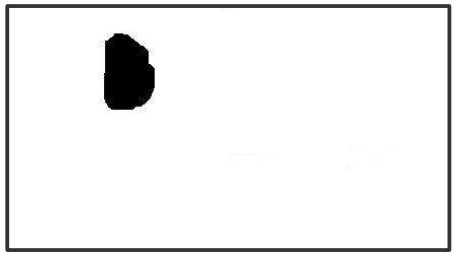
# Objektum számlálás



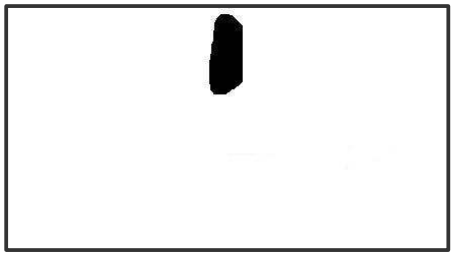
Spatial features map



Extracted blobs



Object one

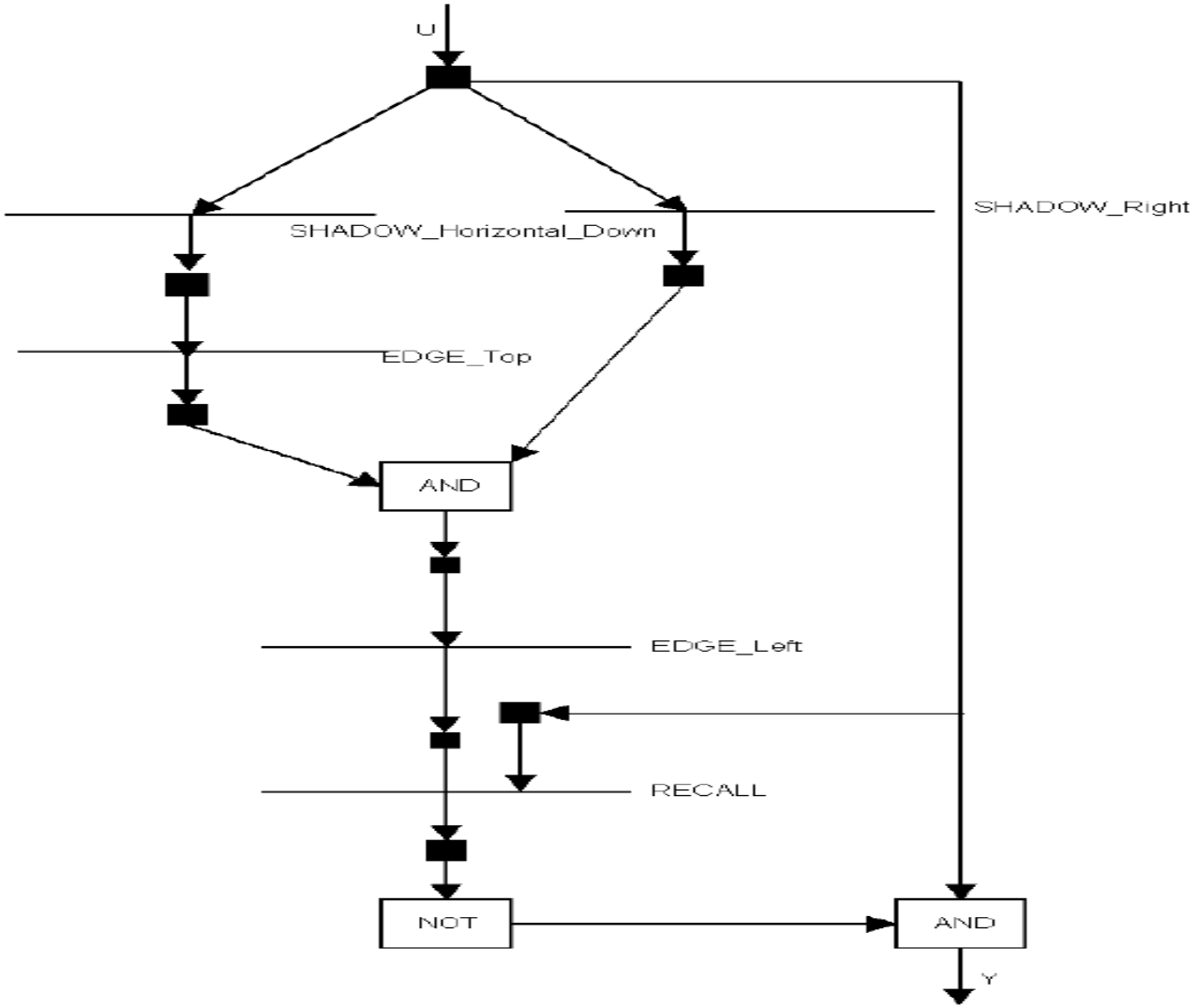


Object Two

Object Three

Object Four

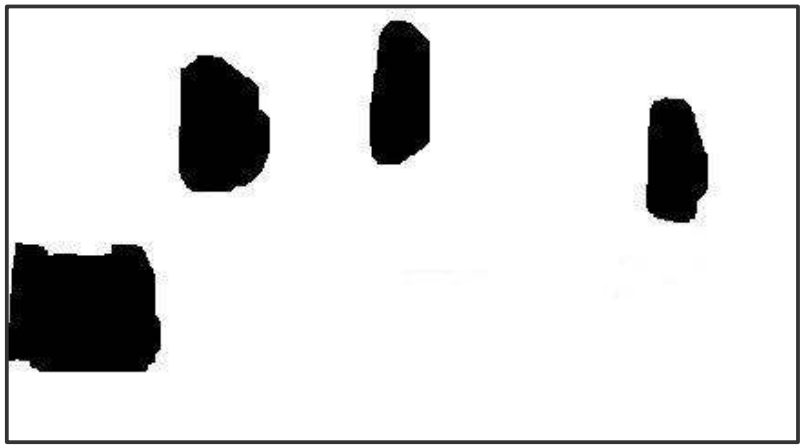
# Példa: Objektum számlálás



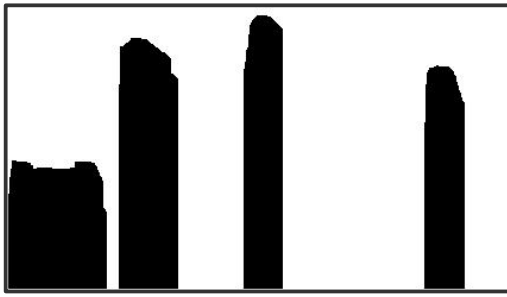
# Labeling

Egyesével leválasztani az objektumokat a képről

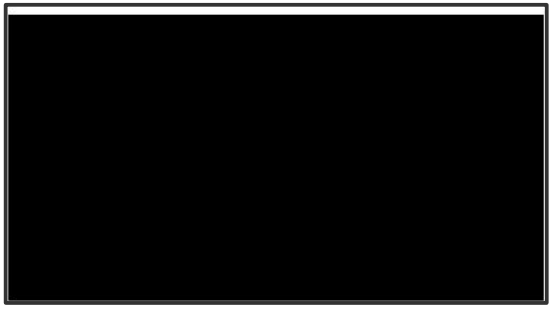
Shadow right



Objektumok



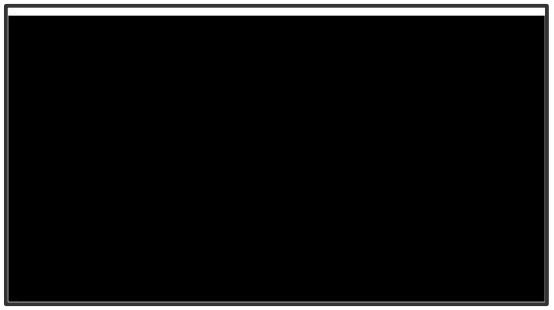
Shadow down



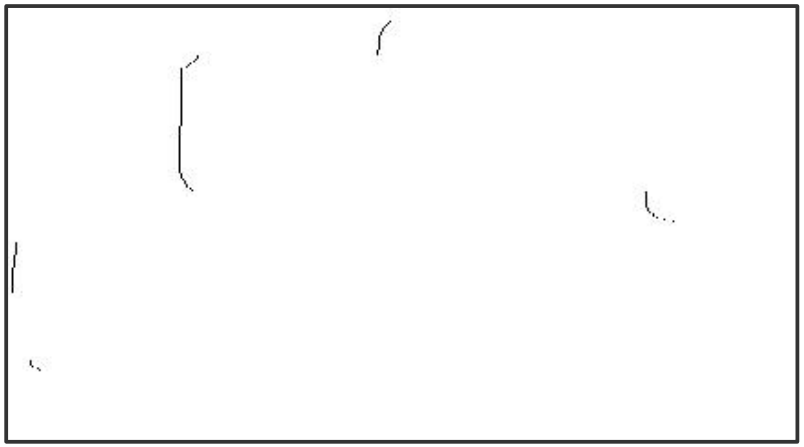
Shadow left,right and down

# Labeling

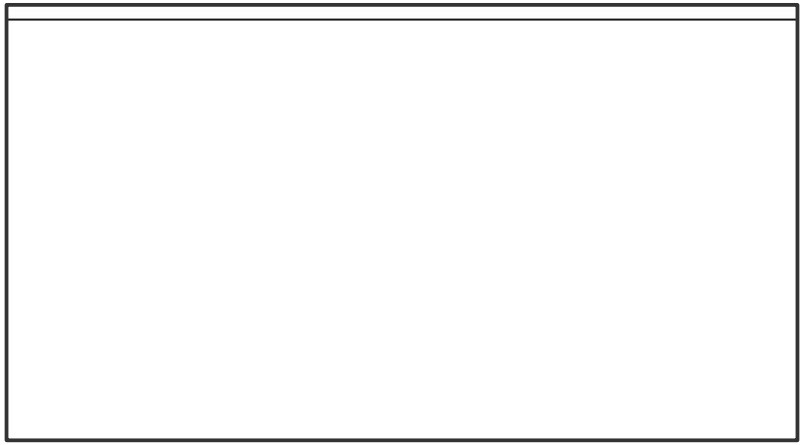
Shadow right



Shadow left,right and down



Ledt edge detection

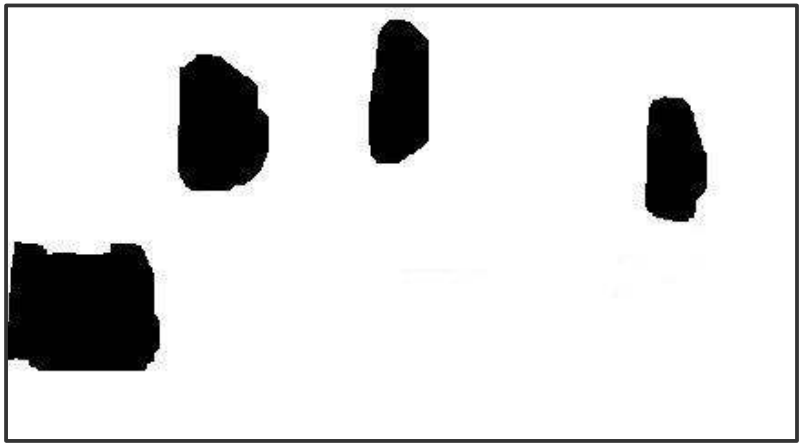


Top edge detection

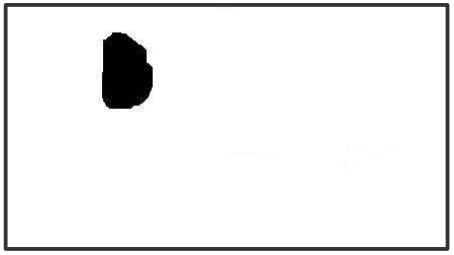
A jobb élek és a felső élek kombinálásával (LOGAND) visszakapunk egyetlen pixelt, ami jobb felső elemhez tartozik

# Labeling

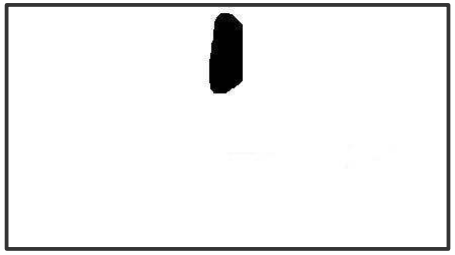
Egyesével visszaadja az objektumokat a képről



Blobs



Object one



Object Two

Object Three

Object Four

# Szimulátor

MatCNN:

http://cnn-technology.itk.ppke.hu/

Mit kell megadnunk:

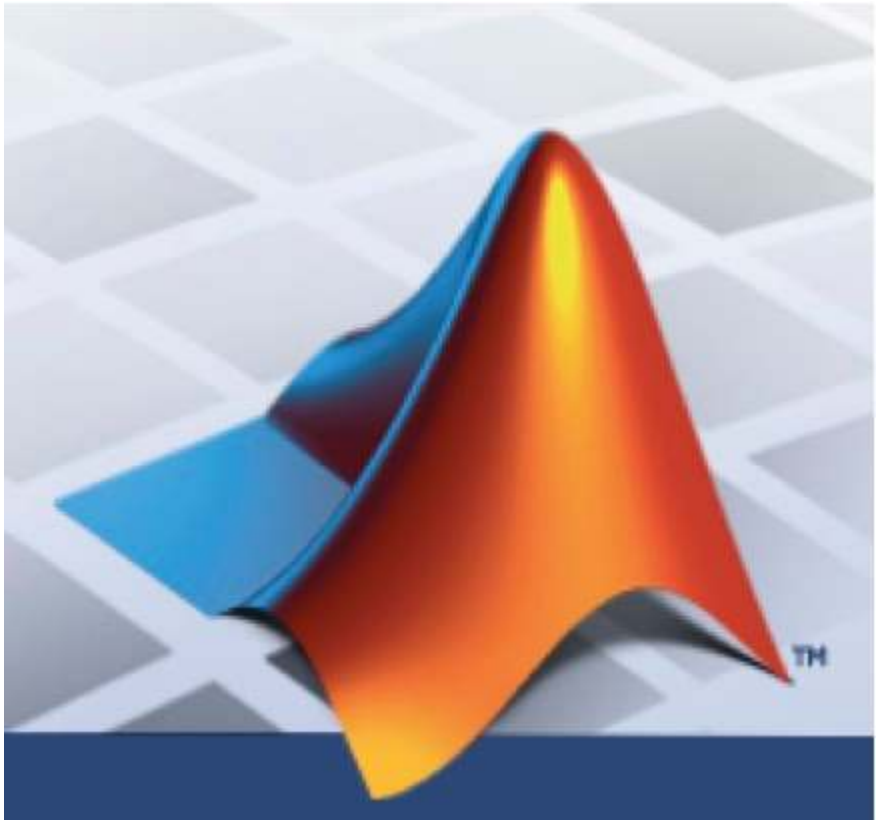
Kezdeti érték

Bemenet

Boundary condition

Futási idő

Numerikus megoldás




# Szimulátor

Szimulátor inicializálása:

**cnn_setenv;**

Kép betöltése:

**INPUT1 = lbmp2cnn('piclib/BABOON.bmp');**

Bemenet betöltése:

**mCNN.INPUT1 = INPUT1;**

Kezdeti állapot betöltése:

**mCNN.STATE = INPUT1;**


# Szimulátor

Peremfeltételek megadása:

**mCNN.Boundary = 2**

-1 és 1 között constans (az adott színnek megfelelő)

2 – zero flux

3 periódikus


# Szimulátor

Numerikus megoldás beállítása:

**mCNN.TimeStep = 0.1;**

**mCNN.IterNum = 10;**

Template library megadása**:**

**mCNN.TemGroup = 'temlib';**

**loadTem('GRADT');**

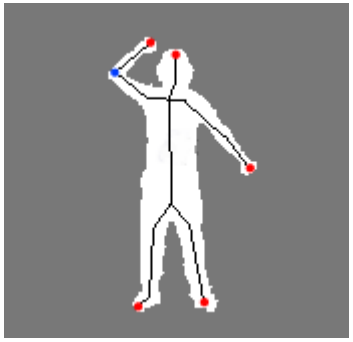
Nem kell megadnunk az A,B,Z értékeket

Szimuláció indítása és eredmény megjelenítése:

**runtem;**

**cnnshow(mCNN.OUTPUT);**

# Example: skeletonization



input

output

# Example: skeletonization

*SKELBW1:*

$\mathbf{A}_1 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_1 =$

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 5 | -1 |
| 0 | -1 | 0 |

$z_1 =$ -1

*SKELBW2:*

$\mathbf{A}_2 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_2 =$

| | | |
|---|---|---|
| 2 | 2 | 2 |
| 0 | 9 | 0 |
| -1 | -2 | -1 |

$z_2 =$ -2

*SKELBW3:*

$\mathbf{A}_3 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_3 =$

| | | |
|---|---|---|
| 0 | 1 | 1 |
| -1 | 5 | 1 |
| 0 | -1 | 0 |

$z_3 =$ -1

*SKELBW4:*

$\mathbf{A}_4 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_4 =$

| | | |
|---|---|---|
| -1 | 0 | 2 |
| -2 | 9 | 2 |
| -1 | 0 | 2 |

$z_4 =$ -2

# Example: skeletonization

*SKELBW5:*

$\mathbf{A}_5 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_5 =$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | 1 |
| 0 | 1 | 1 |

$z_5 =$ -1

*SKELBW6:*

$\mathbf{A}_6 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_6 =$

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 9 | 0 |
| 2 | 2 | 2 |

$z_6 =$ -2

*SKELBW7:*

$\mathbf{A}_7 =$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_7 =$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| 1 | 5 | -1 |
| 1 | 1 | 0 |

$z_7 =$ -1

*SKELBW8:*

$\mathbf{A}_8 =$

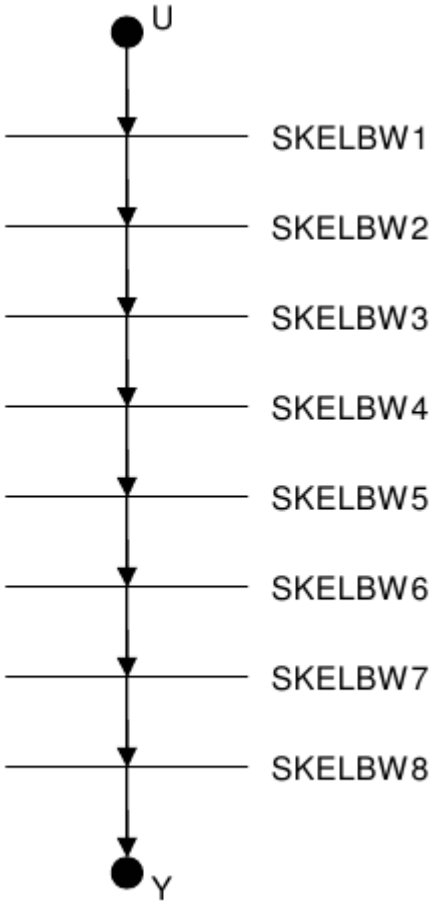
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

$\mathbf{B}_8 =$

| | | |
|---|---|---|
| 2 | 0 | -1 |
| 2 | 9 | -2 |
| 2 | 0 | -1 |

$z_8 =$ -2

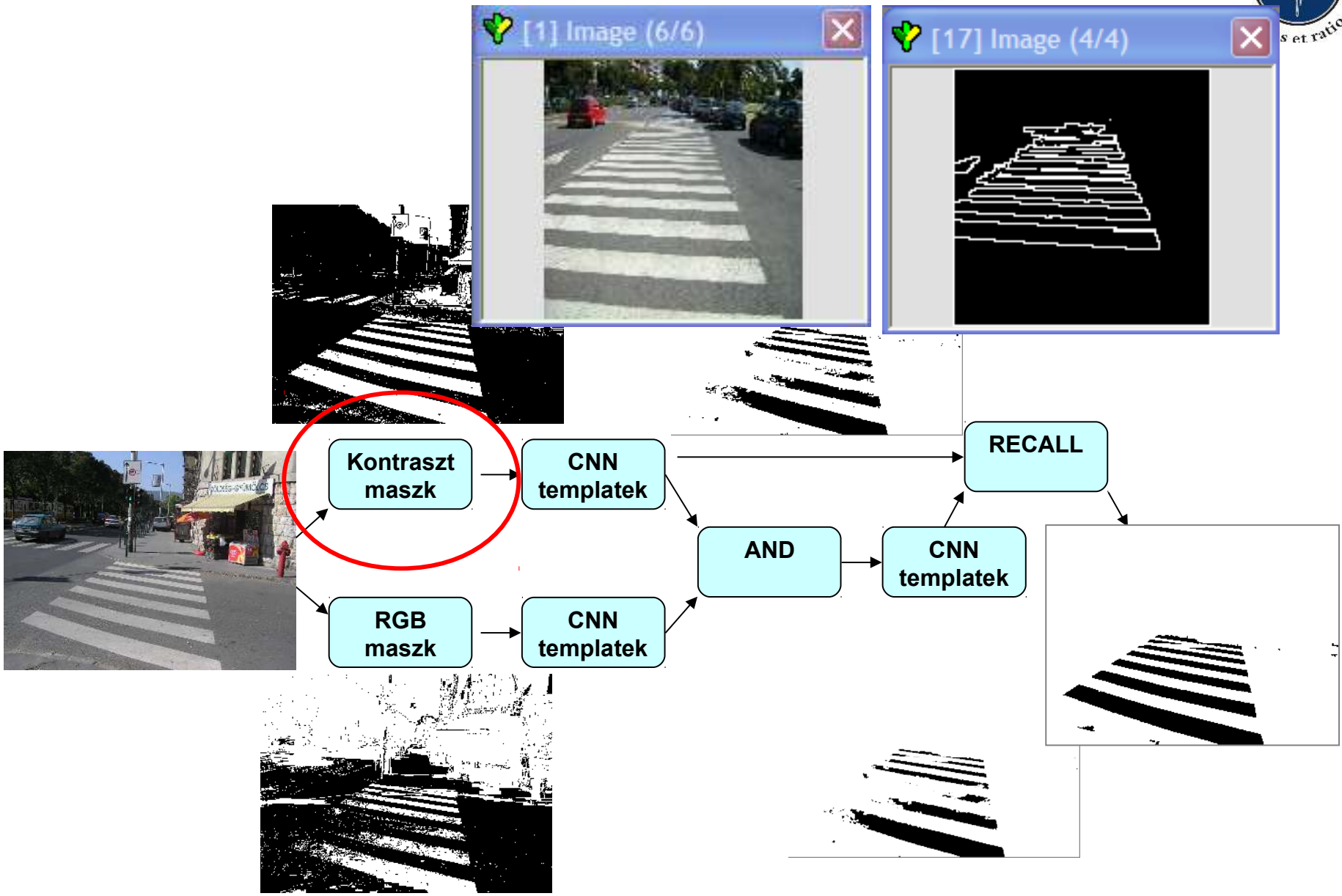
# skeletonization



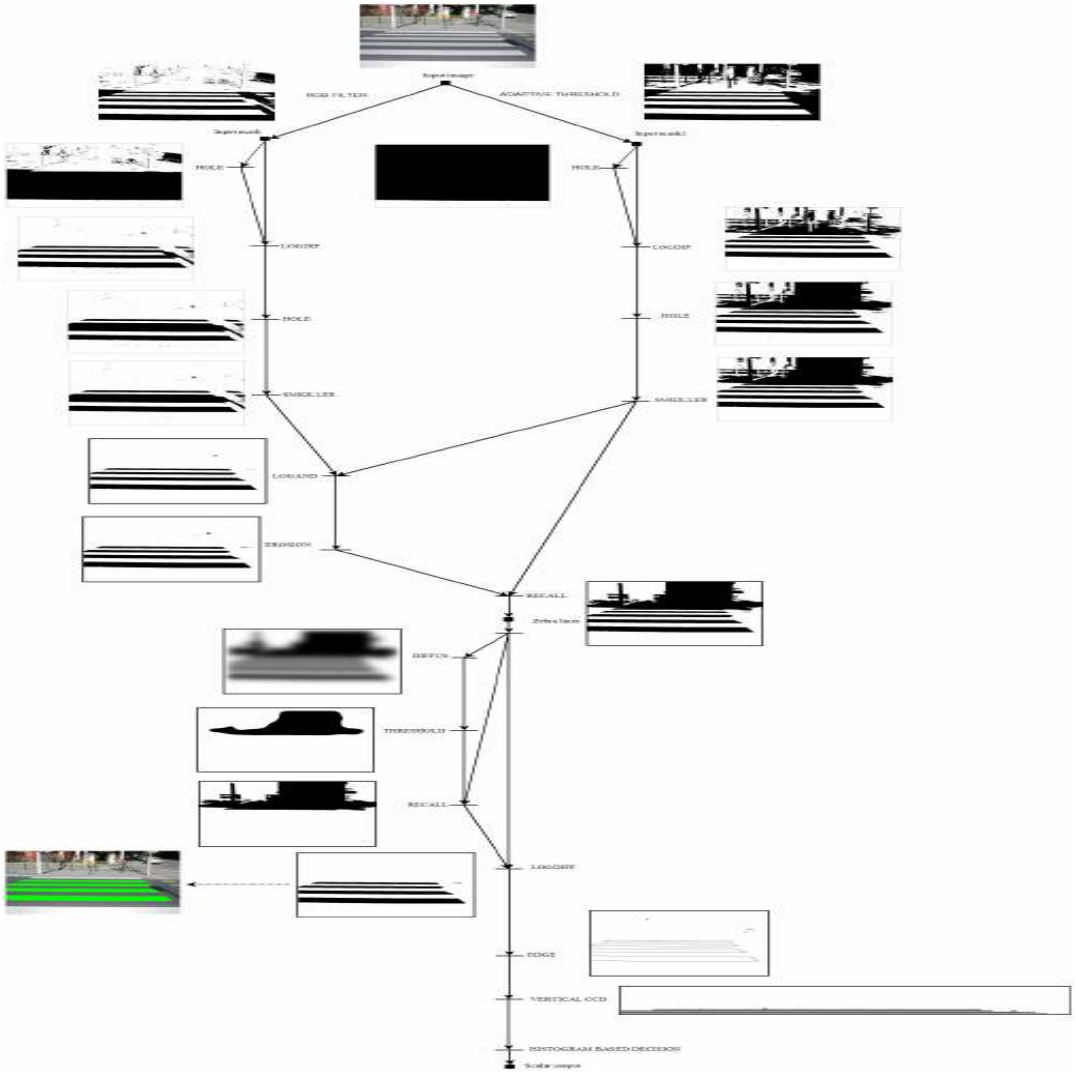

input

output

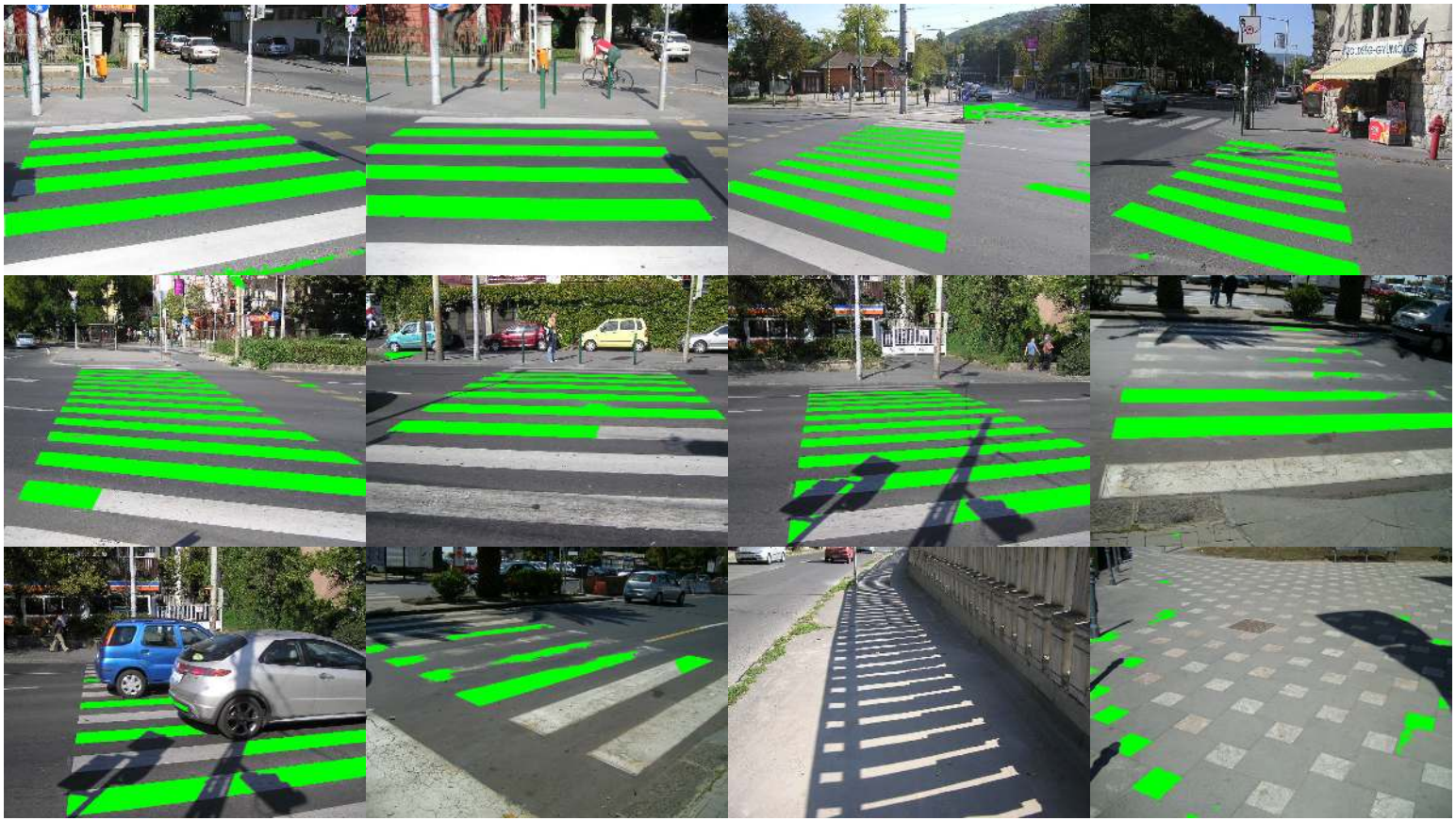
# Application example- Bionic Eyeglass



# Application example- Bionic Eyeglass



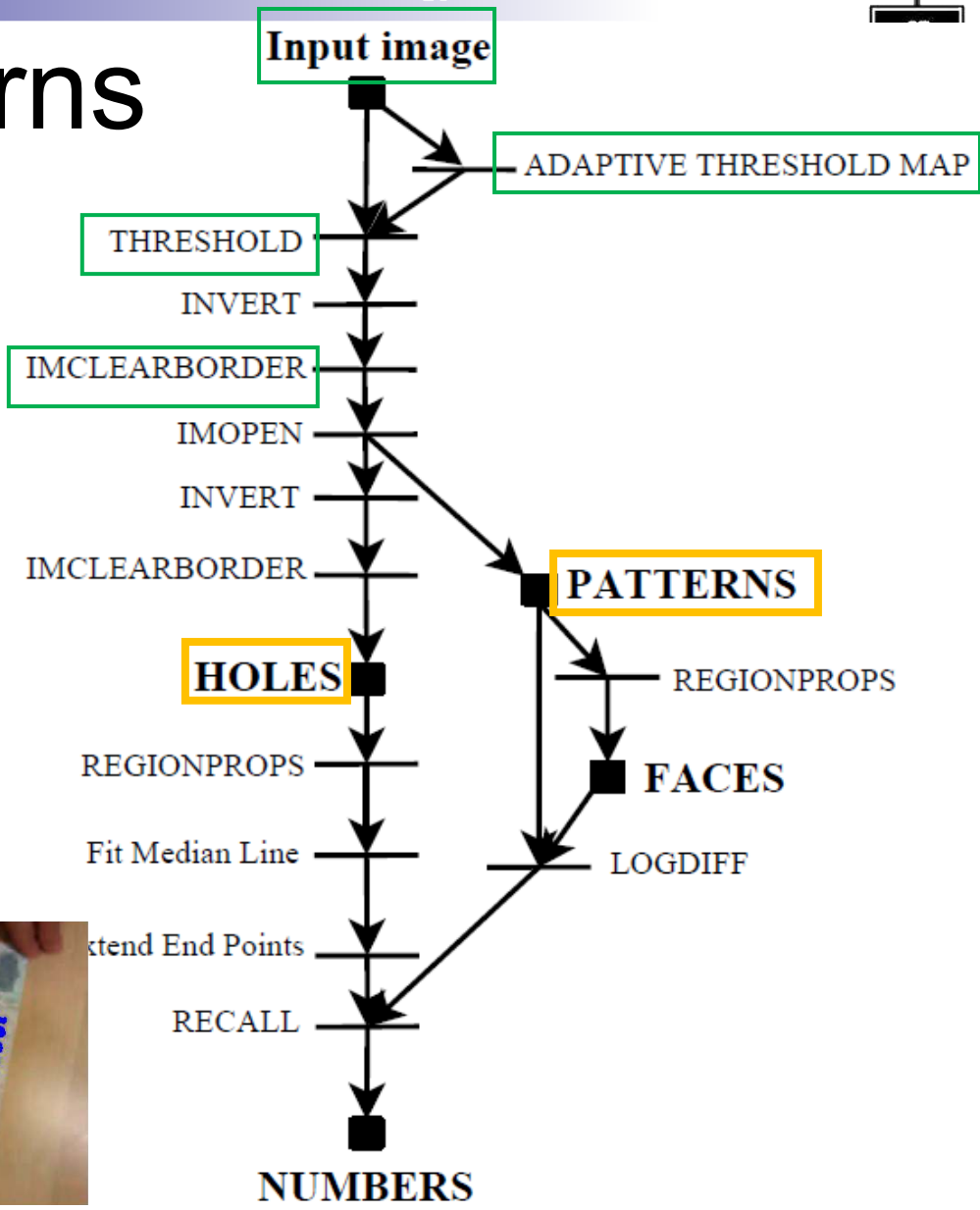
# Application example- Bionic Eyeglass



# Detecting patterns

## An example

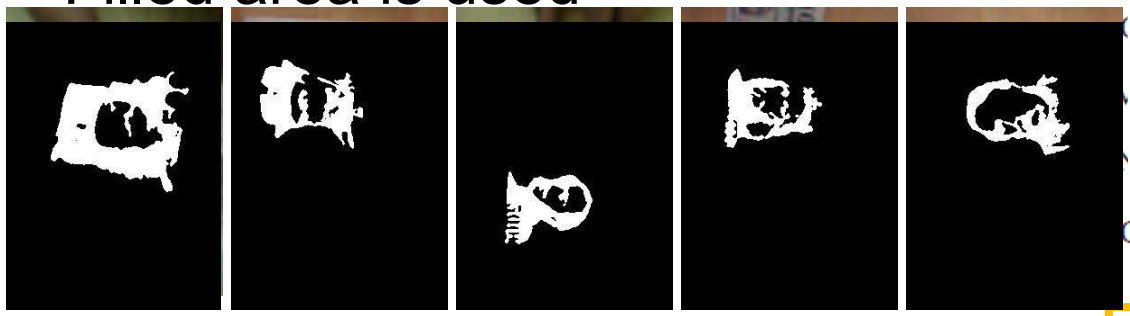
## Patterns and holes



# Detecting patterns

## Portraits

Filled area is used



## Numbers

similarity in: orientation, size, eccentricity → zeros

Extending