# Computer Vision

Catalin Stoean
catalin.stoean@inf.ucv.ro
http://inf.ucv.ro/~cstoean

# Operatii morfologice Obiective

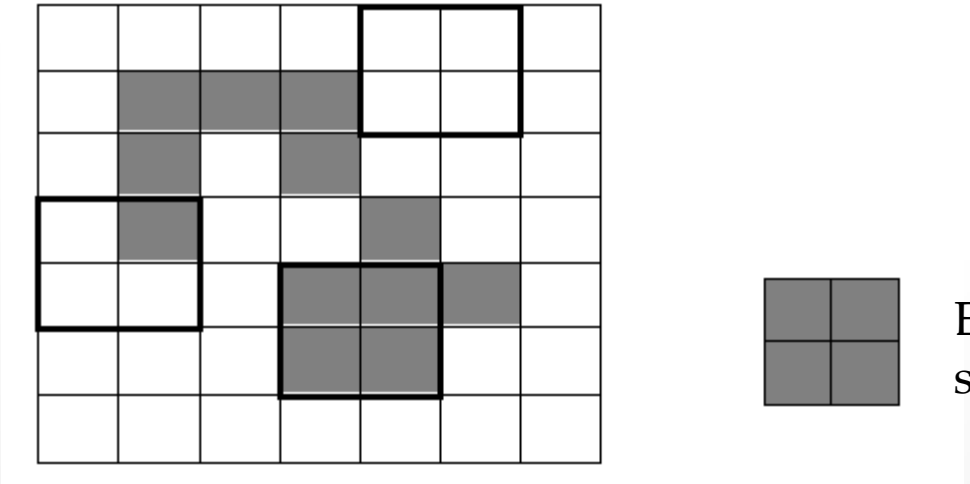
- Erodarea si dilatarea imaginilor folosind filtrare morfologica
- Inchiderea imaginii cu operatori morfologici
- Detectare de margini si colturi cu operatori morfologici
- Segmentarea imaginilor folosind watersheds

# Filtrarea morfologica

- Este aplicata in special pentru imagini binare.
- Imaginile binare pot contine imperfectiuni.
- De regula, imaginile binare se obtin din thresholding si sunt distorsionate de zgomot.
- Filtrarea morfologica intentioneaza sa elimine aceste imperfectiuni
- Filtrarea morfologica defineste o serie de operatori pentru a examina o imagine folosind un element cu o forma predefinita.

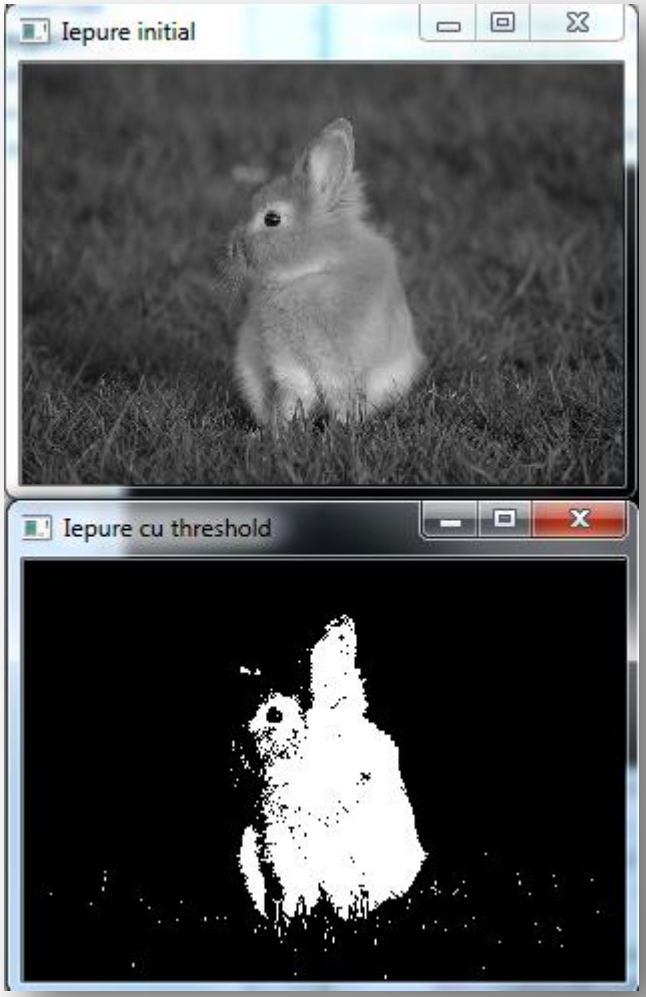
# Filtrarea morfologica

- Modul in care acest element intersecteaza vecinatatea unui pixel determina rezultatul operatiei.
- Forma elementului poate fi una oarecare, insa cele mai des utilizate sunt patrat, cerc sau romb cu originea in centru.



# Pregatirea imaginii

- Citim o imagine in alb-negru, ii aplicam un thresholding astfel incat fundalul sa fie negru, iar prim planul alb.
  - Acesta este standardul in morfologie.
- Cel mai des se foloseste THRESH_BINARY_INV.
  - In cazul de fata prim planul a fost confundat cu fundalul, atunci am folosit THRESH_BINARY.




```
Mat poza = imread("D:/iepure.jpg", 0);

namedWindow("Iepure initial", WINDOW_NORMAL);
resizeWindow("Iepure initial", 300, 210);
imshow("Iepure initial", poza);

threshold(poza,poza,120,255, THRESH_BINARY);

namedWindow("Iepure cu threshold", WINDOW_NORMAL);
resizeWindow("Iepure cu threshold", 300, 210);
imshow("Iepure cu threshold", poza);
```

# Thresholding

Amintim

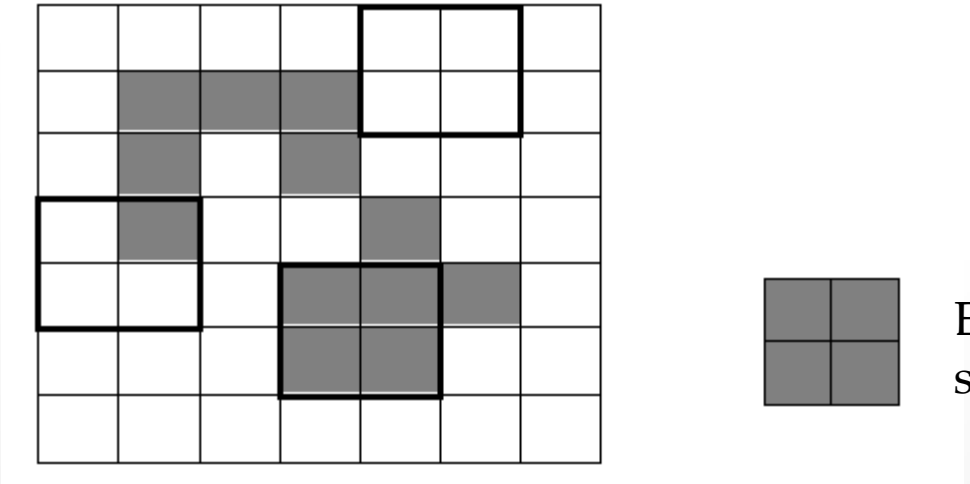
- `THRESH_BINARY` (cea utilizata anterior)

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- dst vine de la destinatie, src de la sursa, thresh este pragul, maxval este al patrulea parametru din functia threshold.
- Daca intensitatea unui pixel este mai mare decat pragul (thresh), noua intensitate devine MaxVal.
  - In cazul anterior, totul a fost facut negru
- Altfel, 0 (alb)

# Aplicarea morfologiei

- Se verifica fiecare pixel al imaginii cu elementul structural.
- Cand originea elementului structural se potriveste cu un anumit pixel, intersectia sa cu imaginea defineste o multime de pixeli pe care o anumita operatie morfologica se aplica.



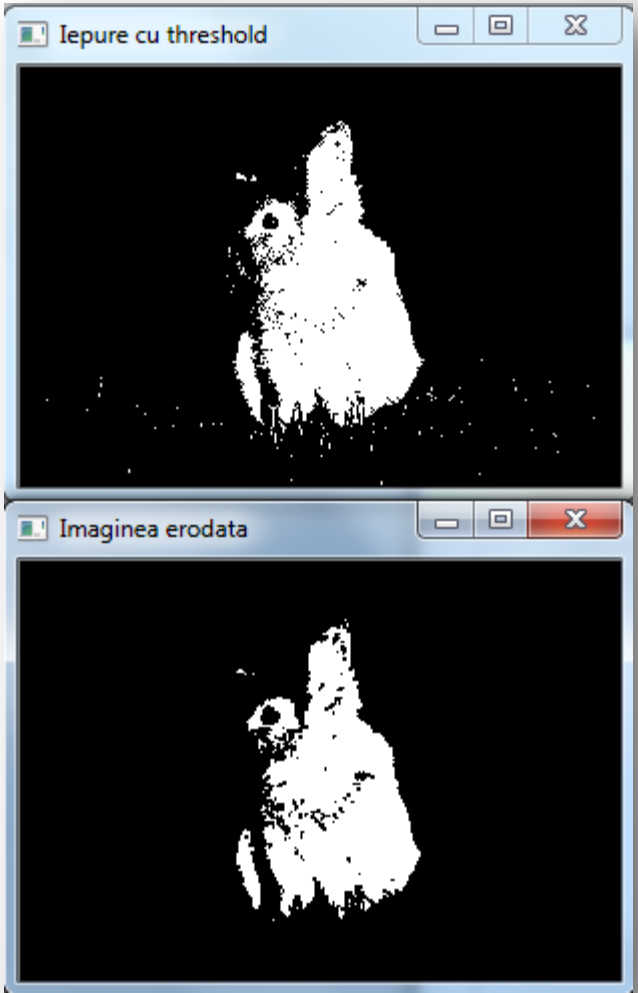
# Elementul structural

- Este o imagine mica binara (o mica matrice de pixeli cu valori de 1 si/sau 0)
- Dimensiunea matricei da marimea elementului structural
- Valorile de 0 si 1 specifica forma elementului structural
- Originea elementului este de obicei unul din pixelii matricei (cel mai des este cel din centru), insa poate fi si in afara matricei.

# Erodarea

```
Mat erodata;
erode(poza, erodata, Mat());

// Afisam imaginea erodata
namedWindow("Imaginea erodata", WINDOW_NORMAL);
resizeWindow("Imaginea erodata", 300, 210);
imshow("Imaginea erodata",erodata);
```

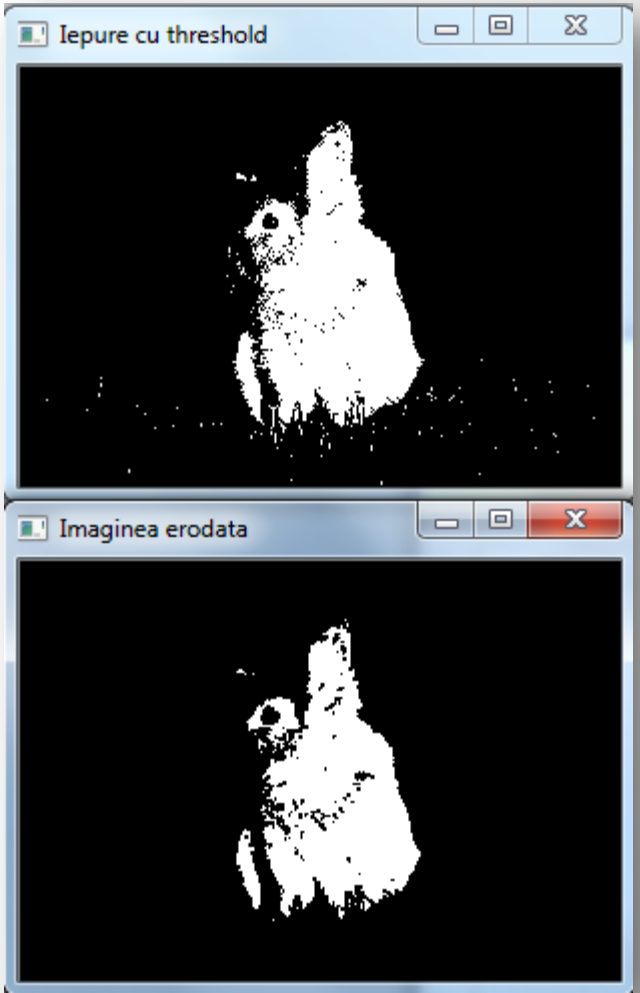
- Filtrul opereaza asupra unei multimi (vecinatati) de pixeli definita de elementul structural.
- Cand se aplica asupra unui pixel, originea elementului structural se aliniaza cu locatia pixelului si toti pixelii care intersecteaza elementul structural sunt inclusi in multime.
- Eroziunea inlocuieste pixelul curent cu valoarea minima aflata in multimea de pixeli.
-



# Erodarea

```
Mat erodata;
erode(poza, erodata, Mat());

// Afisam imaginea erodata
namedWindow("Imaginea erodata", WINDOW_NORMAL);
resizeWindow("Imaginea erodata", 300, 210);
imshow("Imaginea erodata",erodata);
```

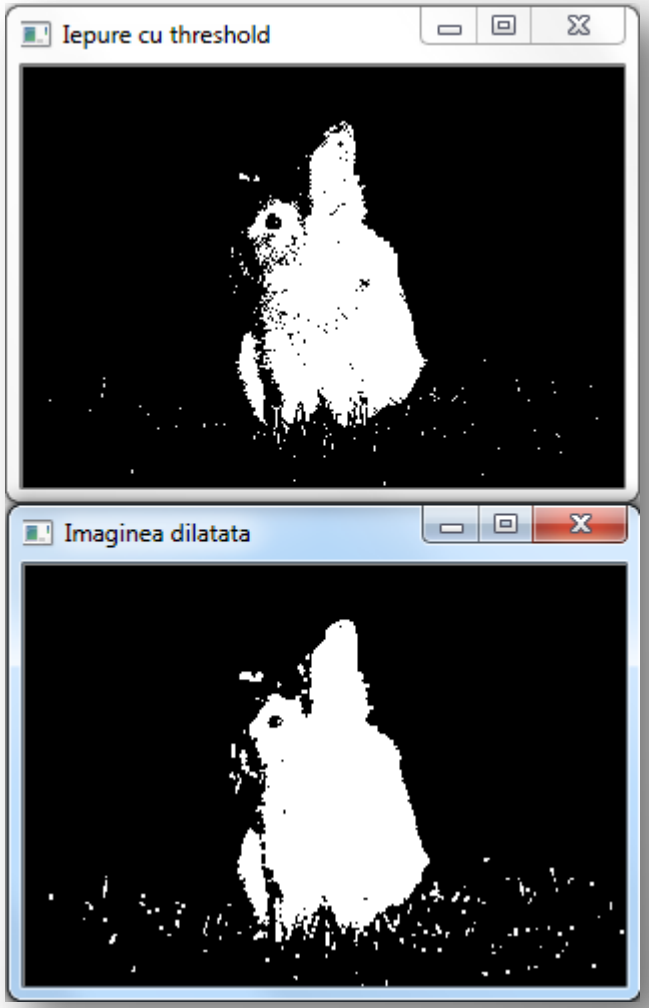
- Cum imaginea este binara, fiecare pixel este inlocuit ori cu 0 (negru), ori cu 255 (alb).
- Daca elementul structural pus peste un anumit pixel atinge fundalul, adica un punct din intersectie este negru, atunci pixelul curent este trimis in fundal.
- Marimea iepurelui este redusa.



# Dilatarea

- Este complementarul erodarii, adica inlocuieste pixelul curent cu valoarea maxima din multimea de pixeli intersectata de obiectul structural.
- Daca obiectul structural atinge prim planul imaginii, atunci pixelul curent devine alb.
- Prim planul este acum mai mare decat in imaginea initiala.

```
// Dilatarea imaginii
Mat dilatata;
dilate(poza, dilatata, Mat());

// Afisam imaginea dilatata
cv::namedWindow("Imaginea dilatata", WINDOW_NORMAL);
resizeWindow("Imaginea dilatata", 300, 210);
cv::imshow("Imaginea dilatata",dilatata);
```

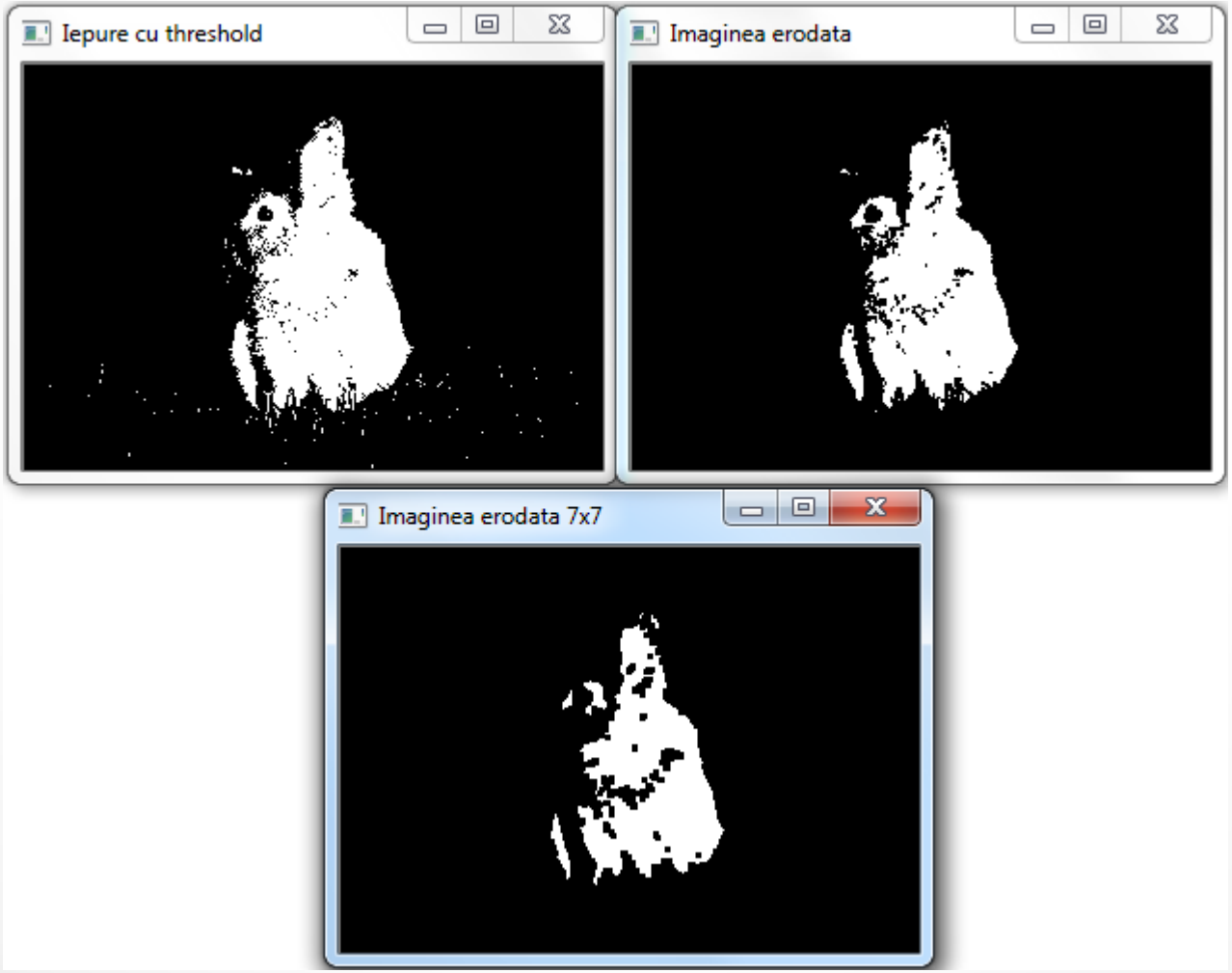


# Erodarea si dilatarea

- Predefinit, OpenCV foloseste un element structural in forma de patrat de marime 3x3.
  - Acesta se obtine din apelarea cu o matrice goala Mat() in argumentul functiei.
- Se poate defini insa si un alt element structural prin specificarea marimii si a formei intr-o matrice nenula.
- Un element structural de 7x7 pixeli se defineste ca mai jos:

```
Mat element(7, 7, CV_8U, cv::Scalar(1));
erode(poza, erodata, element);

// Afisam imaginea erodata 7x7
namedWindow("Imaginea erodata 7x7", WINDOW_NORMAL);
resizeWindow("Imaginea erodata 7x7", 300, 210);
imshow("Imaginea erodata 7x7",erodata);
```

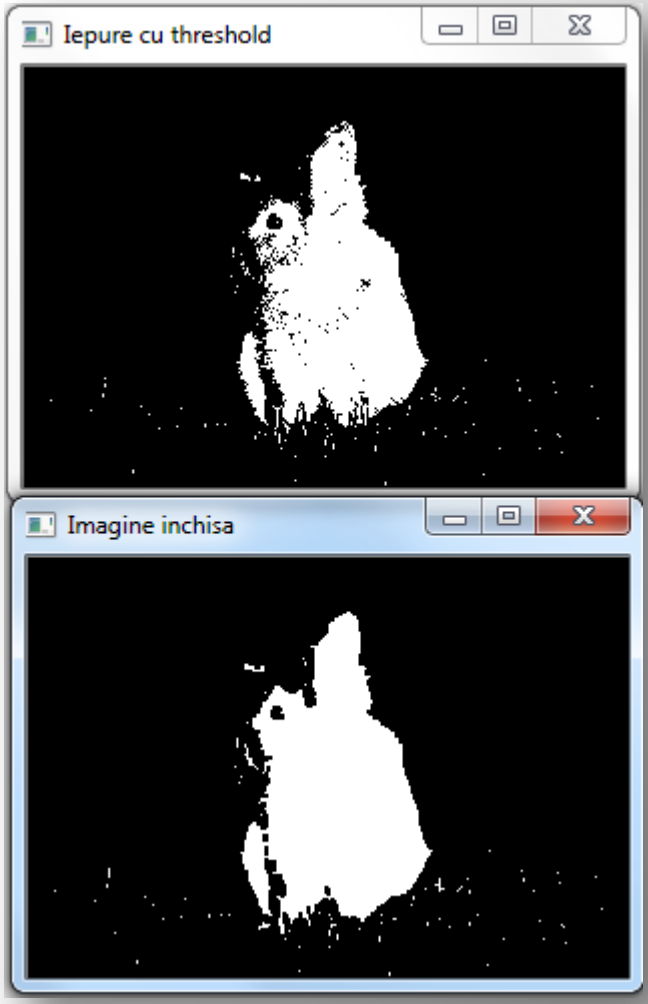
# Initial vs erodare vs erodare 7x7



# Inchiderea imaginii cu operatori morfologici

- Inchiderea este definita ca erodarea imaginii dilatate.
- Acelasi efect ar putea fi obtinut apeland:

```
dilate(image,result,cv::Mat());
erode(result,result,cv::Mat());
```

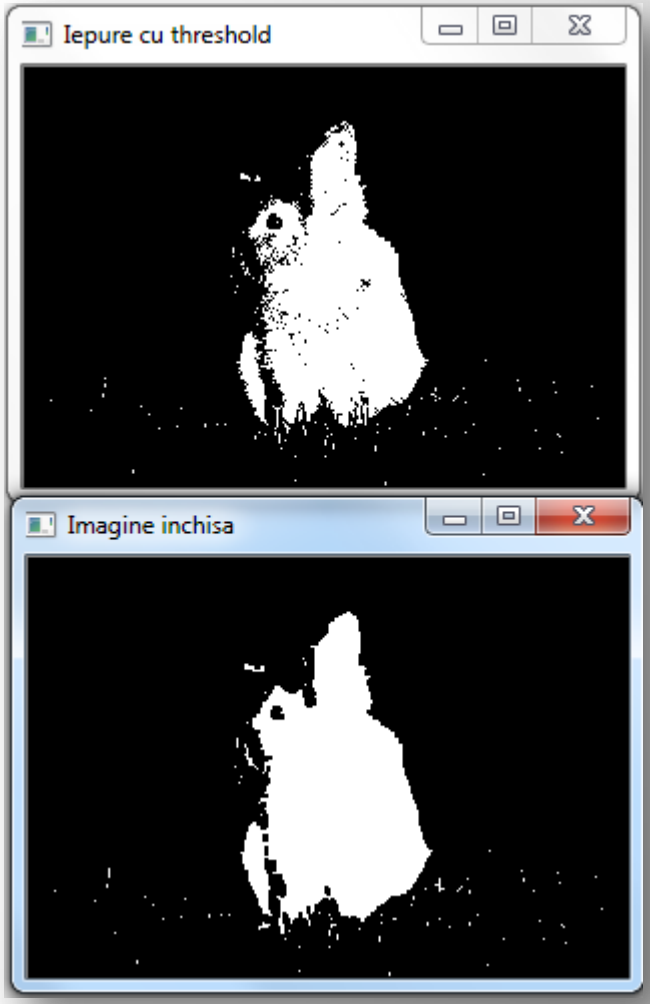



```
Mat element5(5,5,CV_8U,cv::Scalar(1));
Mat inchisa;
morphologyEx(poza, inchisa, MORPH_CLOSE, element5);

namedWindow("Imagine inchisa", WINDOW_NORMAL);
resizeWindow("Imagine inchisa", 300, 210);
imshow("Imagine inchisa", inchisa);
```

# Inchiderea imaginii cu operatori morfologici

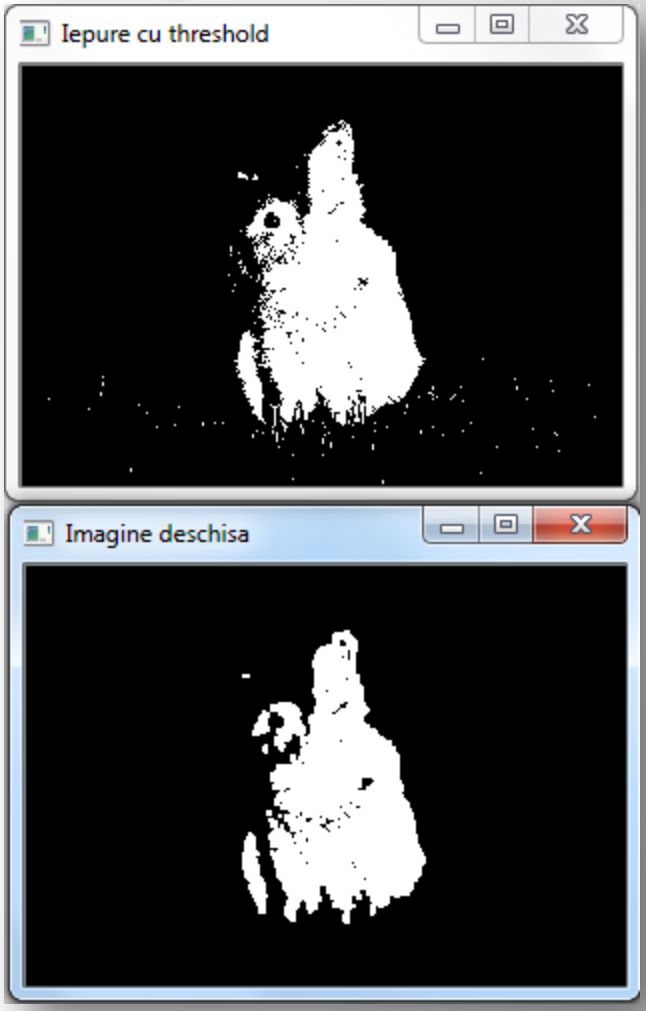
- Micile puncte din cadrul prim planului (din iepure) au fost eliminate
- Conecteaza obiecte adiacente

```
Mat element5(5,5,CV_8U,cv::Scalar(1));
Mat inchisa;
morphologyEx(poza, inchisa, MORPH_CLOSE, element5);

namedWindow("Imagine inchisa", WINDOW_NORMAL);
resizeWindow("Imagine inchisa", 300, 210);
imshow("Imagine inchisa", inchisa);
```



# Deschiderea imaginii cu operatori morfologici

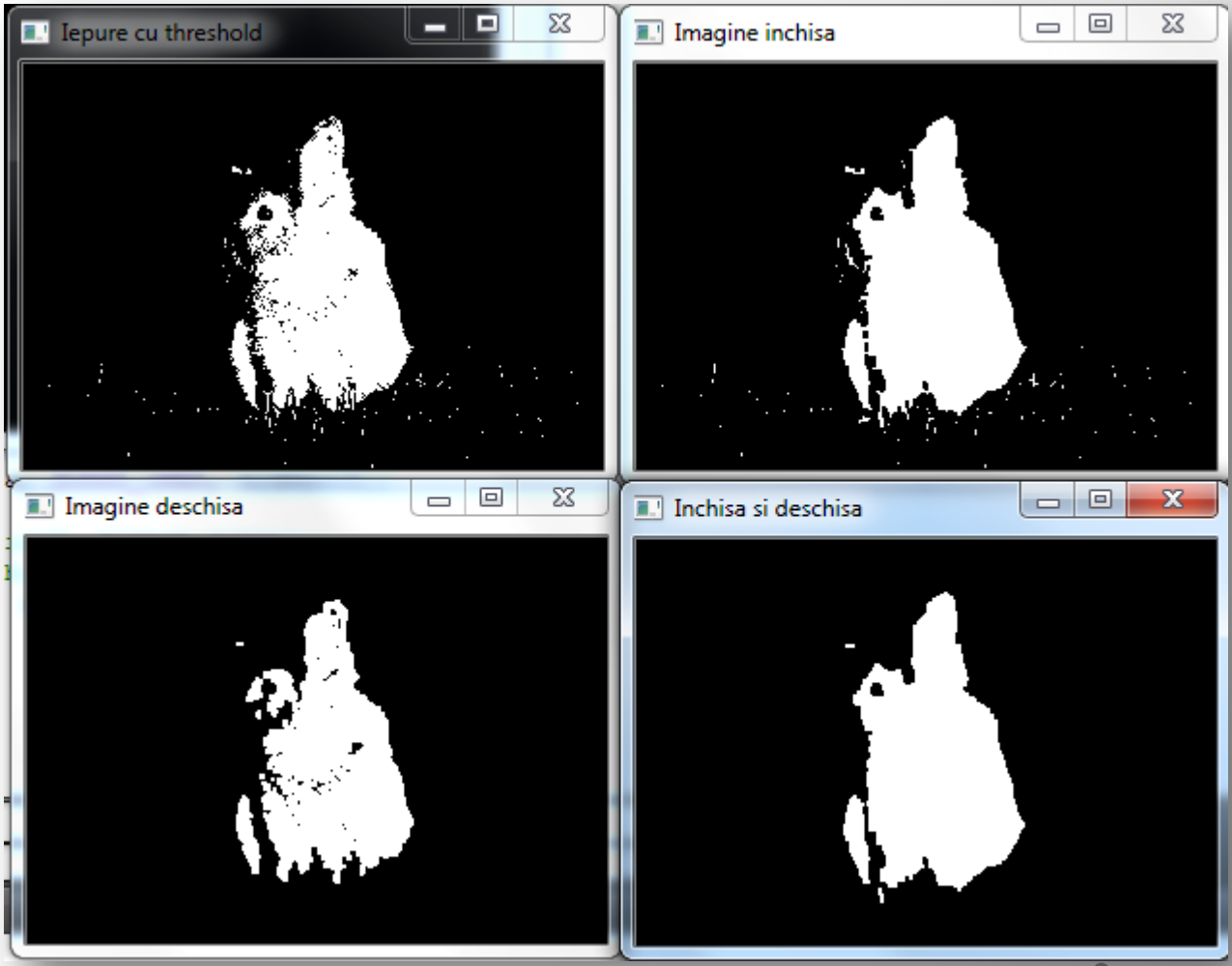
- Este definita ca dilatarea unei imagini erodate.
- Se poate obtine apeland intai rodarea, apoi dilatarea.
- Elimina obiecte albe mici din imagine




```
Mat deschisa;
morphologyEx(poza, deschisa, MORPH_OPEN, element5);

namedWindow("Imagine deschisa", WINDOW_NORMAL);
resizeWindow("Imagine deschisa", 300, 210);
imshow("Imagine deschisa", deschisa);
```

# Inchidere, apoi deschidere

- Ambele filtre se folosesc in detectarea de obiecte.
- Este avantajos sa se foloseasca secvential
- Daca sunt apelate de mai multe ori, nu se obtine nimic nou



# Detectare de margini cu operatori morfologici

- Ne intoarcem la imaginea cu cladirea din cursurile precedente pentru a obtine marginile.
- Definim o clasa Caracteristici cu urmatoarele elemente de tip privat:

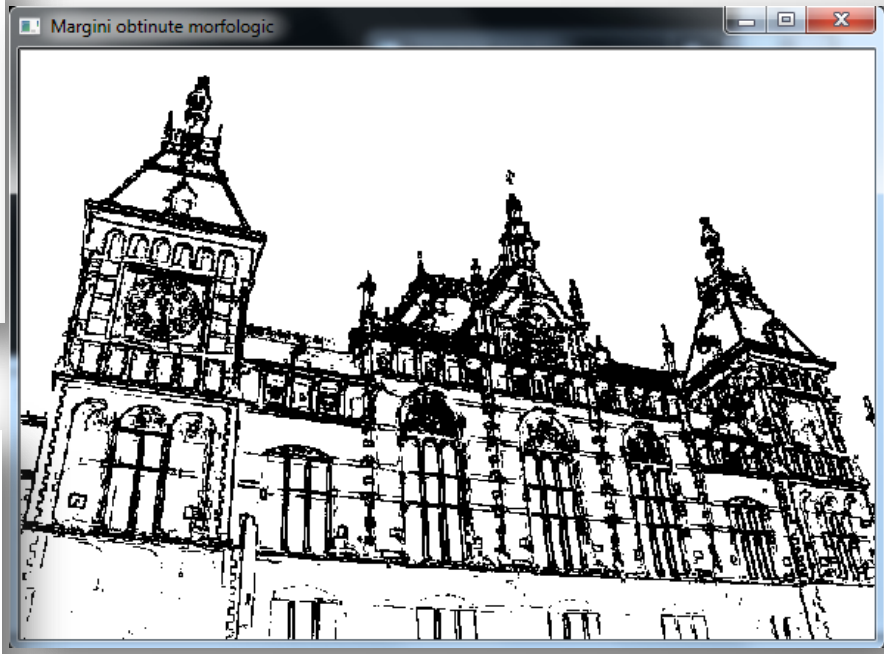
```
class Caracteristici
{
    private:
        int threshold;// pragul pentru a obtine o imagine binara
        // elemente structurale folosite in detectarea colturilor
        Mat cruce;
        Mat romb;
        Mat patrat;
        Mat x;

        void applyThreshold(Mat& result)
        {
            if (threshold>0)//pentru a face fundalul negru, de obicei aplicam THRESH_BINARY_INV
                cv::threshold(result, result, threshold, 255, THRESH_BINARY_INV);
        }
```

# Detectare de margini cu operatori morfologici

- In zona public adaugam o metoda care sa detecteze marginile din imagine folosind aceeasi metoda `morphologyEx`.

```
Mat getEdges(Mat &image)
{
    // Obtinem imaginea gradient
    Mat result;
    morphologyEx(image,result,MORPH_GRADIENT,Mat());

    // Aplicam threshold pentru a obtine o imagine binara
    applyThreshold(result);

    return result;
}
```

```
Caracteristici morf;
morf.setThreshold(40);
// gasim marginile
Mat margini;
margini= morf.getEdges(poza2);

namedWindow("Margini obtinute morfologic", WINDOW_NORMAL);
imshow("Margini obtinute morfologic", margini);
```



# Detectare de colturi cu operatori morfologici

- Nu avem implementare directa in OpenCV.
- Vom folosi si elemente structurale cu forme diferite de patrat.
  - Patrat
  - Romb
  - Cruce
  - X
- Toate vor fi definite in constructorul Caracteristici.
-

```
Caracteristici()
{
    threshold = 0;

    cruce = Mat(5,5,CV_8U,Scalar(0));

    romb = Mat(5,5,CV_8U,Scalar(1));
    patrat = Mat(5,5,CV_8U,Scalar(1));
    x = Mat(5,5,CV_8U,Scalar(0));

    // Element structural in forma de cruce
    for (int i=0; i<5; i++)
    {
        cruce.at<uchar>(2,i)= 1;
        cruce.at<uchar>(i,2)= 1;
    }

    // Element structural in forma de romb
    romb.at<uchar>(0,0)= 0;
    romb.at<uchar>(0,1)= 0;
    romb.at<uchar>(1,0)= 0;
    romb.at<uchar>(4,4)= 0;
    romb.at<uchar>(3,4)= 0;
    romb.at<uchar>(4,3)= 0;
    romb.at<uchar>(4,0)= 0;
    romb.at<uchar>(4,1)= 0;
    romb.at<uchar>(3,0)= 0;
    romb.at<uchar>(0,4)= 0;
    romb.at<uchar>(0,3)= 0;
    romb.at<uchar>(1,4)= 0;

    // Element structural in forma de X
    for (int i=0; i<5; i++)
    {
        x.at<uchar>(i,i)= 1;
        x.at<uchar>(4-i,i)= 1;
    }
}
```

# Detectare de colturi cu operatori morfologici

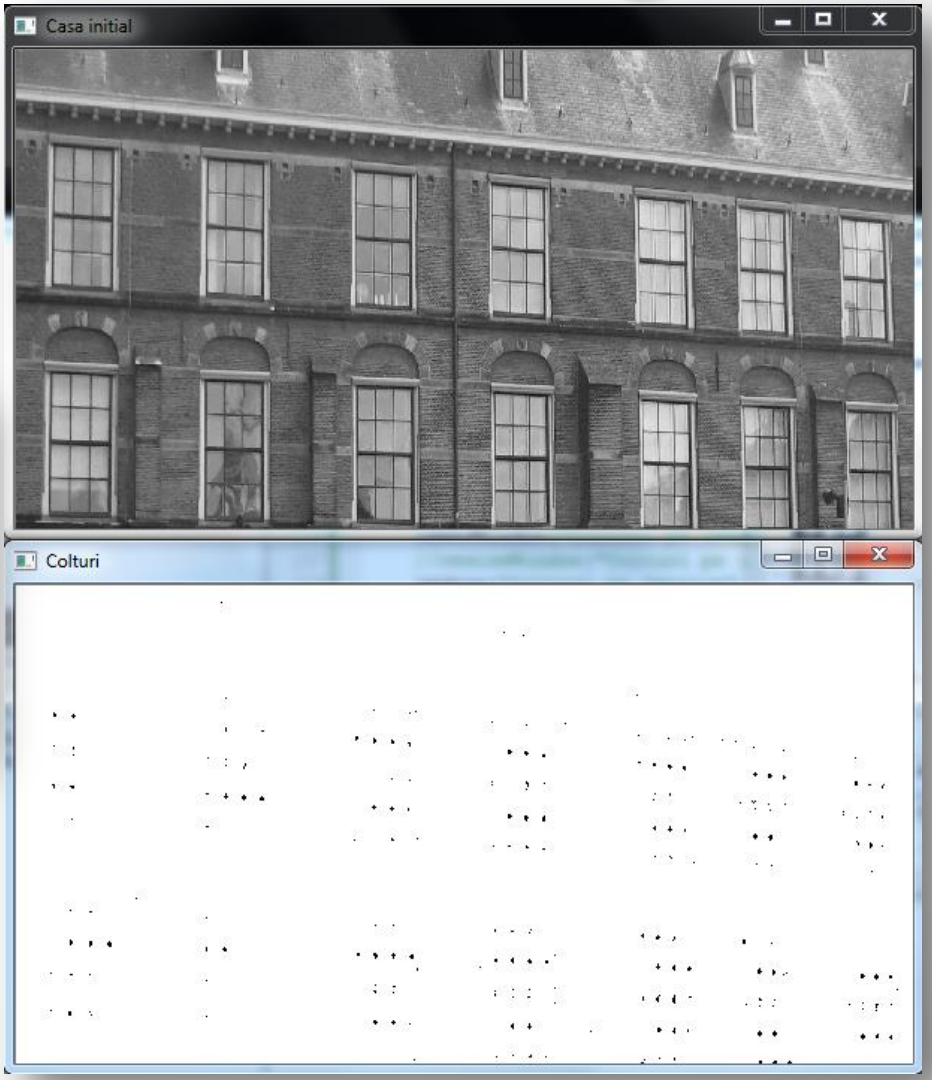
- Aplicam elementele structurale in cascada, urmand sa obtinem o harta a colturilor.

```
Mat getCorners(const Mat &poza)
{
    Mat rez;

    // Dilatam cu cruce
    dilate(poza,rez,cruce);

    // Erodam cu romb
    erode(rez,rez,romb);

    Mat rez2;
    // Dilatare cu X
    dilate(poza,rez2,x);

    // Erodare cu patrat
    erode(rez2,rez2,patrat);

    // Colturile se obtin prin diferenterea
    //  celor 2 imagini rezultate
    absdiff(rez2,rez,rez);

    // Aplicam threshold pentru a obtine o imagine binara
    applyThreshold(rez);

    return rez;
}
```

# Detectare de colturi cu operatori morfologici

- main.cpp

```
namedWindow("Casa initial", WINDOW_NORMAL);
//resizeWindow("Casa initial", 600, 320);
imshow("Casa initial", poza2);

// Obtinem colturi
Mat colturi;
colturi = morf.getCorners(poza2);
// Display the corner on the image
namedWindow("Colturi", WINDOW_NORMAL);
resizeWindow("Colturi", 600, 320);
imshow("Colturi", colturi);
```



# Detectare de colturi cu operatori morfologici

- Desenam pe imaginea initiala in locatiile unde sunt detectati pixeli negri in imaginea binara obtinuta dupa procesarile morfologice.

In clasa Caracteristici

```
void drawOnImage(Mat& binary, Mat& image)
{
    Mat_<uchar>::const_iterator it = binary.begin<uchar>();
    Mat_<uchar>::const_iterator itend = binary.end<uchar>();

    for (int i = 0; it!= itend; ++it,++i)
    {
      if (*it == 0)//daca pixelul curent este negru
              circle(image, Point(i%image.step, i/image.step), 5, Scalar(255,0,0));
    }
}
```

In main.cpp

```
morf.drawOnImage(colturi, poza2);
namedWindow("Colturi pe imagine", WINDOW_NORMAL);
resizeWindow("Colturi pe imagine", 600, 320);
imshow("Colturi pe imagine", poza2);
```

# Detectare de colturi cu operatori morfologici

- Rezultatul afisat pentru imaginea initiala:

# Segmentarea imaginilor folosind watersheds

- Termenul watershed se refera la cursul (controlat al) unui râu, iar metafora are urmatorul sens in segmentarea imaginilor:
- O imagine alb-negru este vazuta ca o suprafata unde
- intensitati ridicate reprezinta varfuri si dealuri
- Intensitati scazute reprezinta văi.
- Incepem prin umplerea fiecarei vai izolate cu apa de culori diferite.
- Crestem nivelurile apelor pana cand apele incep sa se uneasca, moment in care se construiesc bariere (diguri).

# Segmentarea imaginilor folosind watersheds

- Se continua cresterea nevilurilor apelor pana cand varfurile sunt acoperite.
- Rezultatul este dat de barierele create.
- Folosim functia `watershed`.
- Ca intrare pentru functie avem o imagine cu intregi pe 32 de biti in care fiecare pixel nenul reprezinta o eticheta.
- Ideea este sa marcam anumiti pixeli ai imaginii care se stie ca apartin unei regiuni date.
- De la aceasta etichetare initiala, algoritmul determina regiunile la care apartin ceilalti pixeli.

# Segmentarea imaginilor folosind watersheds

- Vom crea imaginea marker ca fiind alb-negru si apoi o vom converti la o imagine cu intregi.

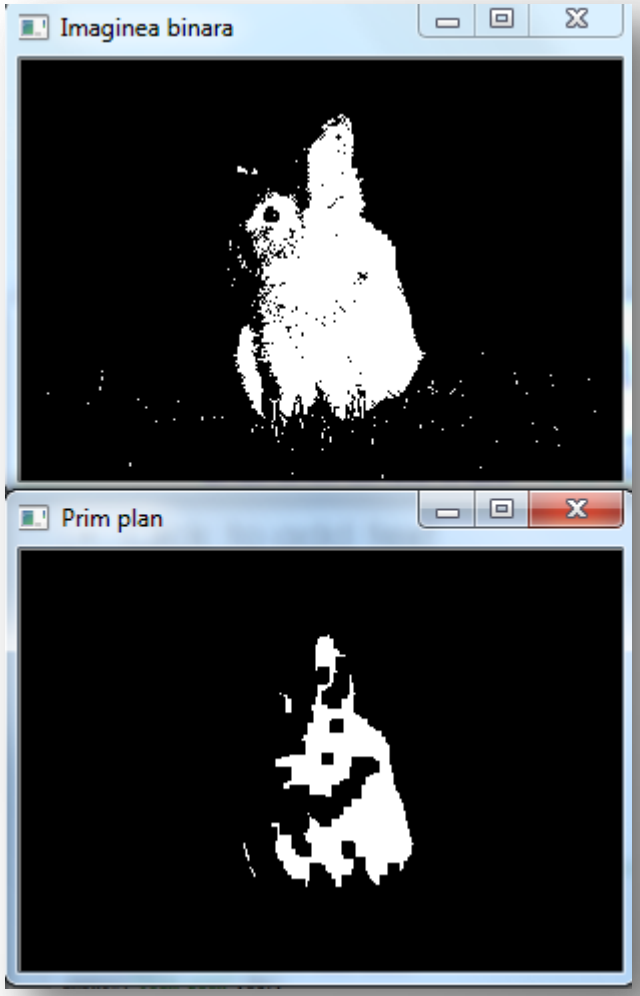
```
class WatershedSegmenter
{
    private:
        Mat markers;//imaginea marker

    public:

        void setMarkers(Mat& markerImage)
        {
            // Convertim la o imagine de intregi
            markerImage.convertTo(markers,CV_32S);
        }

        Mat process(Mat &image)
        {
            //aplicam algoritmul watershed
            watershed(image, markers);
            return markers;
        }

        // Intoarcem rezultatul sub forma unei imagini
        Mat getSegmentation()
        {
            Mat rez;
            // toate valorile mai mari decat 255 vor deveni 255
            markers.convertTo(rez, CV_8U);
            return rez;
         }

         // Intoarcem doar conurul sub forma unei imagini
        Mat getWatersheds()
        {
            Mat rez;
            markers.convertTo(rez, CV_8U, 255, 255);
            return rez;
        }
};
```

# Segmentarea imaginilor folosind watersheds

- Citim imaginea de mai devreme cu iepurele in format color, o transformam in binar si ii aplicam un thresholding.
- Va trebui sa extragem pixelii care se refera la prim plan (iepure) si pe cei care formeaza fundalul (iarba).
- Vom eroda imaginea pentru a pastra numai pixelii care apartin obiectelor importante (aici, iepurele).
  - Erodarea se aplica de 6 ori folosind un element structural standard.

# Segmentarea imaginilor folosind watersheds

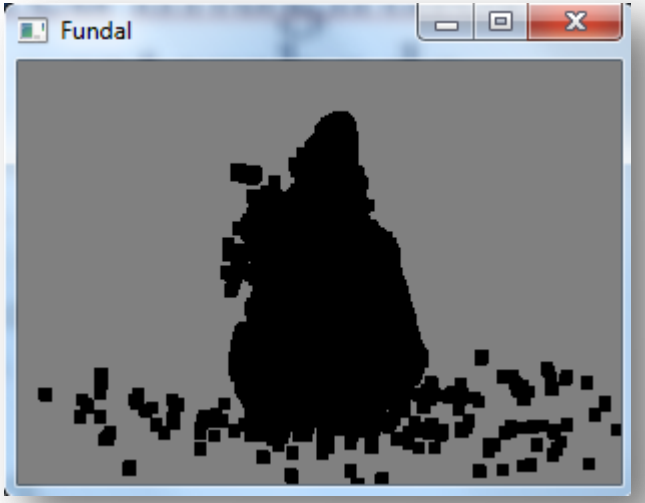
```
Mat poza = imread("D:/iepure.jpg");//citim poza color

// Transformam poza in binar
Mat binara;
cvtColor(poza, binara, CV_BGR2GRAY);
threshold(binara, binara, 120, 255, THRESH_BINARY);

// Afisam poza binara
namedWindow("Imaginea binara", WINDOW_NORMAL);
resizeWindow("Imaginea binara", 300, 210);
imshow("Imaginea binara", binara);

// Eliminam zgomot si obiecte mici
Mat fg;
//erodarea se aplica de 6 ori, iar punctul este central (-1, -1)
erode(binara, fg, Mat(), Point(-1,-1), 6);

// Afisam imaginea cu prim plan
namedWindow("Prim plan", WINDOW_NORMAL);
resizeWindow("Prim plan", 300, 210);
imshow("Prim plan",fg);
```



# Segmentarea imaginilor folosind watersheds

- Marcam pixelii de prim plan cu 255 (alb) si pe cei de fundal cu 128 (gri), iar pe restul (cei despre care nu stim ce reprezinta) cu 0 (negru).
- Aplicam o dilatare mare (tot de 6 ori) asupra imaginii binare initiale.

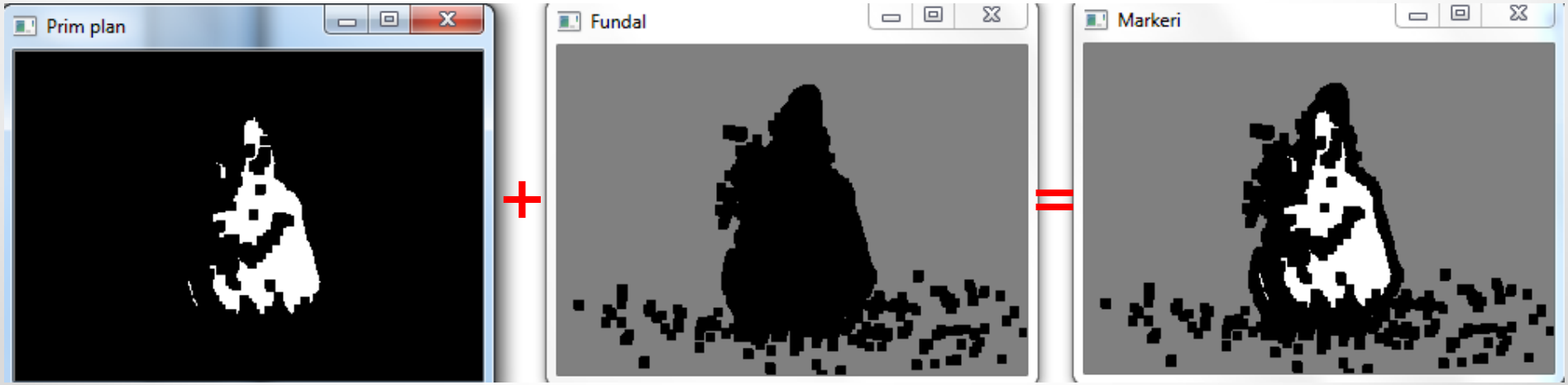
```
// Identificam fundalul
Mat bg;
dilate(binara, bg, Mat(), Point(-1,-1), 6);
threshold(bg, bg, 1, 128, THRESH_BINARY_INV);

// Afisam background-ul
namedWindow("Fundal", WINDOW_NORMAL);
resizeWindow("Fundal", 300, 210);
imshow("Fundal",bg);
```



# Segmentarea imaginilor folosind watersheds

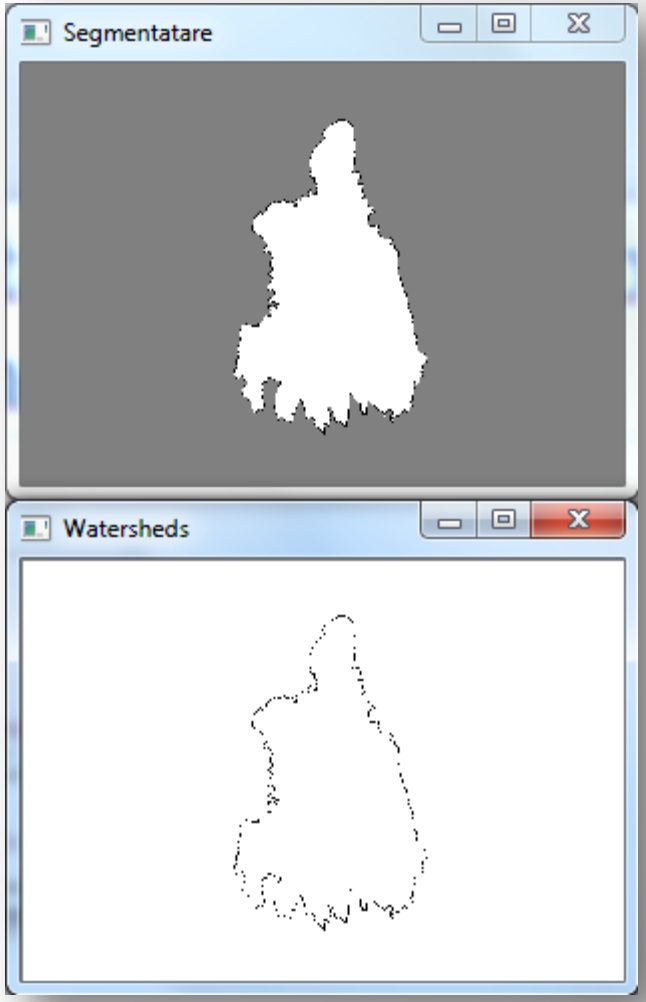
- Combinam apoi imaginile pentru a forma imaginea marker. Aceasta va fi folosita ca intrare in algoritmul watershed.

```
// Imaginea cu markeri
Mat markers(binara.size(), CV_8U, Scalar(0));
markers = fg + bg;
namedWindow("Markeri", WINDOW_NORMAL);
resizeWindow("Markeri", 300, 210);
imshow("Markeri",markers);
```



# Segmentarea imaginilor folosind watersheds

```
WatershedSegmenter segment;

// Set markers and process
segment.setMarkers(markers);
segment.process(poza);

// Display segmentation result
namedWindow("Segmentatare", WINDOW_NORMAL);
resizeWindow("Segmentatare", 300, 210);
imshow("Segmentatare",segment.getSegmentation());

// Display watersheds
namedWindow("Watersheds", WINDOW_NORMAL);
resizeWindow("Watersheds", 300, 210);
imshow("Watersheds",segment.getWatersheds());
```



# Proiecte 1/2

1. Faceti un proiect cu GUI care sa permita incarcarea unei imagini si sa se aplice apoi dilatarea si erodarea pentru aceasta. Pentru acestea sa se poata stabili marimea elementului structural si sa se poata alege din mai multe forme posibile (patrat, romb, cruce, x etc).
2. Faceti un proiect cu GUI care sa permita incarcarea unei imagini si sa se detecteze apoi pe imagine colturile. Pentru aceasta sa se poata stabili de catre utilizator marimea elementului structural.

# Proiecte 2/2

3. Faceti un proiect cu GUI care sa permita incarcarea unei imagini si sa se detecteze contururile elementelor din prim plan. Thresholding-ul pentru imaginea initiala trebuie sa se poata selecta de catre utilizator ca fiind binar sau invers binar. Utilizatorul trebuie sa poata stabili si marimea elementului structural.