Computer Vision

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

Identificare obiecte, utilizare camera web

- Transformata Hough pentru identificare de cercuri intr-o poza
- Adaugarea unui Trackbar in aplicatie (inclusiv VS)
- Detectarea unui obiect de o anumita culoare folosind HSV
- Aplicatii folosind camera web

Transformata Hough

- Identifica intr-o figura obiecte care sunt in forma de cerc.
- HoughCircles (InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double **minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)
  - **image** – imagine in grayscale, pe 8 biti, un singur canal.
  - **circles** – Vector de cercuri. Fiecare cerc este un vector de 3 elemente float.
  - **method** – singura posibila deocamdata este CV_HOUGH_GRADIENT .
  - **dp** – parametru legat de rezolutia de acumulare. Daca dp=1, acumulatorul are aceeasi rezolutie ca image. Daca dp=2 , acumulatorul are lungimea si inaltimea injumatatite.
    - Cu cat este mai mare, cu atat vectorul de acumulari este mai mic

Transformata Hough

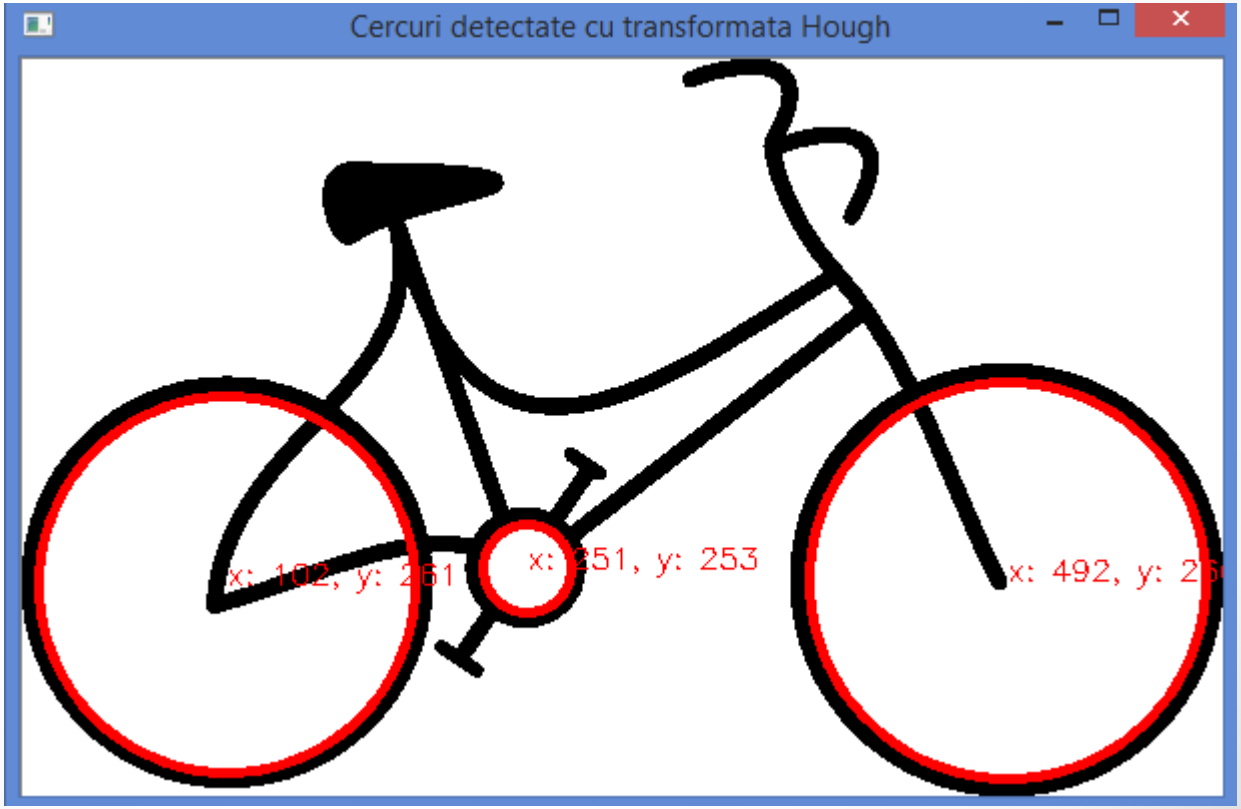
- HoughCircles (InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double **minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)
  - **minDist** – Distanta minima intre centrele cercurilor detectate.
    - Daca este prea mic, poate gasi mai multe cercuri vecine pentru acelasi cerc.
    - Daca este prea mare, unele cercuri pot sa fie omise.
  - **param1** – Este pragul superior care urmeaza sa fie trimis metodei Canny().
  - **param2** – Este pragul acumulatorului pentru centrele cercurilor. Cu cat este mai mic, cu atat mai multe cercuri false sunt detectate.
  - **minRadius** – raza minima.
  - **maxRadius** – raza maxima.

Transformata Hough

```
void detecteazaCercuri(Mat &poza)
{
    Mat pozaNecolor;
    cvtColor( poza, pozaNecolor, CV_BGR2GRAY );
    // aplicam filtrare pentru a reduce zgomotul
    GaussianBlur( pozaNecolor, pozaNecolor, Size(5, 5), 2 );

    vector<Vec3f> cercuri;
    // Transformata Hough gaseste cercurile
    HoughCircles( pozaNecolor, cercuri, CV_HOUGH_GRADIENT, 1, 7, 200, 80, 0, 0 );

    Mat copiePoza;
    poza.copyTo(copiePoza);

    /// Desenam cercurile gasite
    for( size_t i = 0; i < cercuri.size(); i++ )
    {
        Point center(cvRound(cercuri[i][0]), cvRound(cercuri[i][1]));
        int radius = cvRound(cercuri[i][2]);
        // centrul cercului
        putText(copiePoza, "x: " + to_string((int)cercuri[i][0]) + ", y: " + to_string((int)cercuri[i][1]),
            center, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0,0,255), 1, 8, false );
        // cercul
        circle( copiePoza, center, radius, Scalar(0,0,255), 3, 8, 0 );
    }

    namedWindow( "Cercuri detectate cu transformata Hough", CV_WINDOW_AUTOSIZE );
    imshow( "Cercuri detectate cu transformata Hough", copiePoza );
}
```

Transformata Hough

```
int main()
{
    Mat poza;
    poza = imread( "E:/Test/bici.png");

    detecteazaCercuri(poza);

     waitKey(0);
  return 0;
}
```



Adaugarea unui Trackbar in aplicatie

- int createTrackbar(const string& **trackbarname**, const string& **winname**, int* **value**, int **count**, TrackbarCallback **onChange**=0, void* **userdata**=0)
  - **trackbarname** – numele trackbar-ului.
  - **winname** – numele ferestrei in care va fi pus trackbar-ul.
    - Poate fi una noua sau una existenta care contine deja componente.
  - **value** – pointer optional catre o variabila care stabileste pozitia slider-ului.
  - **count** – valoarea maximala pe slider. Valoarea minima este 0.

Adaugarea unui Trackbar in aplicatie

- int createTrackbar(const string& **trackbarname**, const string& **winname**, int* **value**, int **count**, TrackbarCallback **onChange**=0, void* **userdata**=0)
  - **onChange** – pointer catre functia care este apelata cand se schimba valoarea pe slider.
    - Prototipul sau este void func(int,void*);
      - primul parametru este pozitia trackbar-ului, iar al doilea este reprezentat de datele utilizatorului (de la urmatorul parametru).
    - Daca acest parametru lipseste, valoarea variabilei se actualizeaza si nu se apeleaza nimic.
  - **userdata** – date trimise prin argumentul anterior. Prin acestea, se poate evita utilizarea variabilelor globale

Adaugarea unui Trackbar in aplicatie

- Setam variabile globale

```
int dp = 1;
int minDist = 7;
int param1 = 200;
int param2 = 80;
int minRadius = 0;
int maxRadius = 0;
Mat pozaNecolor, poza;
vector<Vec3f> cercuri;
```

- Modificam putin si in main()

```
int main()
{
    poza = imread( "E:/Test/bici.png");
    detecteazaCercuri();

    waitKey(0);
    return 0;
}
```

```
void deseneazaCercuri()
{
    HoughCircles( pozaNecolor, cercuri, CV_HOUGH_GRADIENT, dp, minDist, param1, param2, minRadius, maxRadius );
    Mat copiePoza;
    poza.copyTo(copiePoza);
    /// Desenam cercurile gasite
    for( size_t i = 0; i < cercuri.size(); i++ )
    {
        Point center(cvRound(cercuri[i][0]), cvRound(cercuri[i][1]));
        int radius = cvRound(cercuri[i][2]);
        // centrul cercului
        putText(copiePoza, "x: " + to_string((int)cercuri[i][0]) + ", y: " + to_string((int)cercuri[i][1]),
            center, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0,0,255), 1, 8, false );
        // cercul
        circle( copiePoza, center, radius, Scalar(0,0,255), 3, 8, 0 );
    }
    imshow( "Control Hough", copiePoza );
}

void detecteazaCercuri()
{
    namedWindow("Control Hough", CV_WINDOW_AUTOSIZE);

    createTrackbar("dp", "Control Hough", &dp, 10);
    createTrackbar("minDist", "Control Hough", &minDist, 50);
    createTrackbar("param1", "Control Hough", &param1, 255);
    createTrackbar("param2", "Control Hough", &param2, 255);
    createTrackbar("minRadius", "Control Hough", &minRadius, 255);
    createTrackbar("maxRadius", "Control Hough", &maxRadius, 255);
    cvtColor( poza, pozaNecolor, CV_BGR2GRAY );
    // aplicam filtrare pentru a reduce zgomotul
    GaussianBlur( pozaNecolor, pozaNecolor, Size(5, 5), 2 );
    while(true)
    {
        faraZero();
        deseneazaCercuri();
        if(waitKey(50) > 0)
            break;
    }
}
```

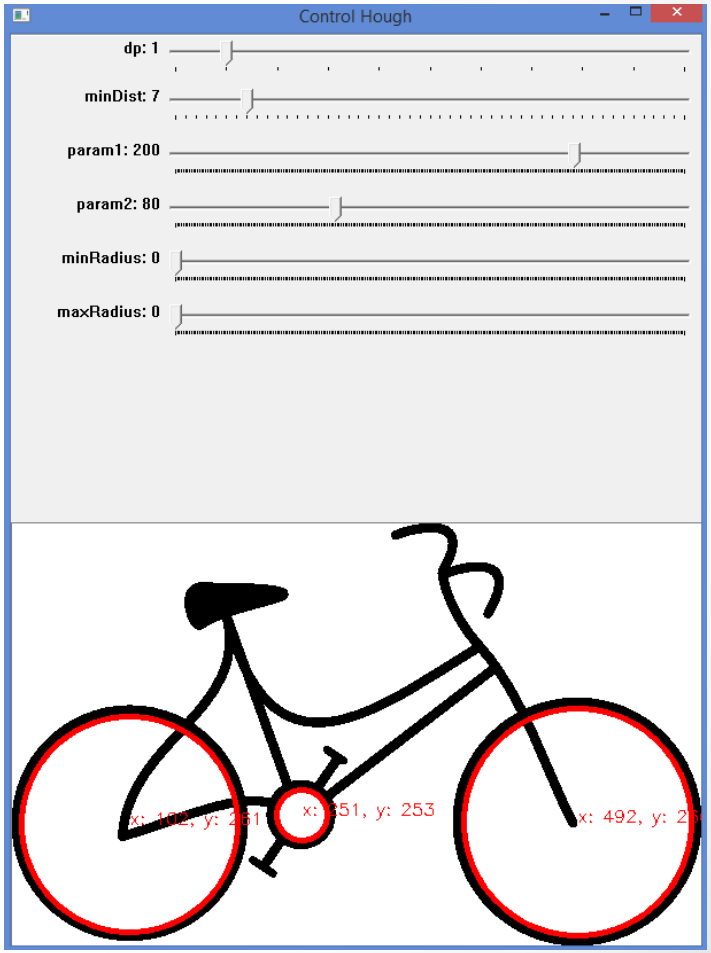
Adaugarea unui Trackbar in aplicatie I

- Anumiti parametri din functia HoughCircles nu pot avea valoarea zero:
  - dp
  - minDist
  - param1
  - param2
- Din acest motiv modificam valoarea, daca este introdusa, in 1.

```
void faraZero()
{
    if(dp == 0)
        dp = 1;
    if(minDist == 0)
        minDist = 1;
    if(param1 == 0)
        param1 = 1;
    if(param2 == 0)
        param2 = 1;
}
```

Adaugarea unui Trackbar in aplicatie I

- In varianta I am utilizat un ciclu infinit care putea fi oprit numai de atingerea unei taste de catre utilizator.
- Nu s-a facut uz de “TrackbarCallback”.



Adaugarea unui Trackbar in aplicatie II

- Apeleaza functia modificare de cate ori se modifica vreo valoare la slide-uri.
- Nu transmit valorile prin parametri pentru ca variabilele sunt globale.

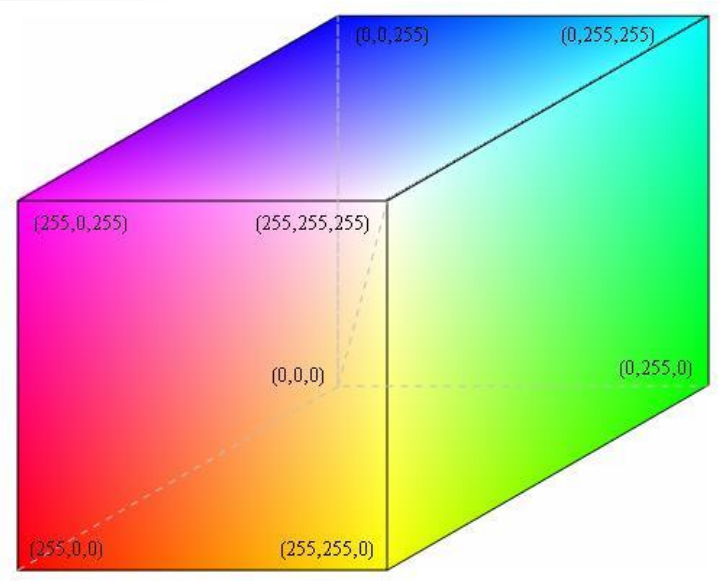
```
void modificare(int, void*)
{
    faraZero();
    deseneazaCercuri();
}

void detecteazaCercuri2()
{
    namedWindow("Control Hough", CV_WINDOW_AUTOSIZE);

    createTrackbar("dp", "Control Hough", &dp, 10, modificare);
    createTrackbar("minDist", "Control Hough", &minDist, 50, modificare);

    createTrackbar("param1", "Control Hough", &param1, 255, modificare);
    createTrackbar("param2", "Control Hough", &param2, 255, modificare);

    createTrackbar("minRadius", "Control Hough", &minRadius, 255, modificare);
    createTrackbar("maxRadius", "Control Hough", &maxRadius, 255, modificare);

    cvtColor( poza, pozaNecolor, CV_BGR2GRAY );
    // aplicam filtrare pentru a reduce zgomotul
    GaussianBlur( pozaNecolor, pozaNecolor, Size(5, 5), 2 );
    deseneazaCercuri();
}
```

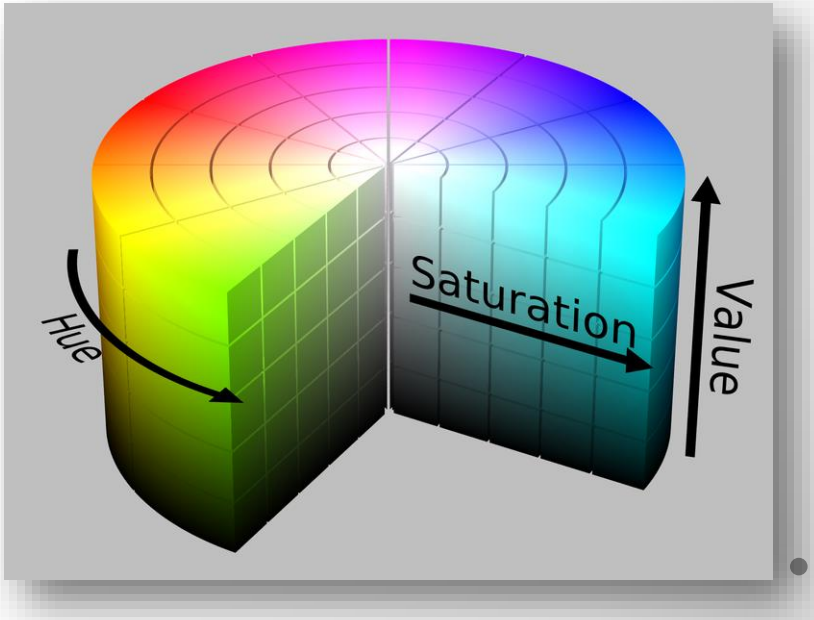
Adaugarea unui Trackbar in aplicatie II

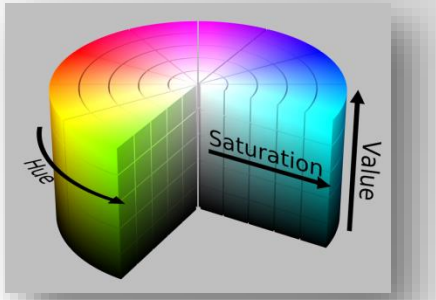
- Rezultatul variantei a doua este identic cu cel anterior.
- Pentru a vedea si un exemplu in care se utilizeaza TrackbarCallback, vizitati
  - http://opencv-srf.blogspot.ro/2011/11/track-bars.html

Detectarea unui obiect de o anumita culoare folosind HSV

- Vom utiliza canalul HUE al spatiului de culori HSV.
- Spatiul HSV este mai potrivit pentru segmentarea imaginilor pe baza de culori.







HSV

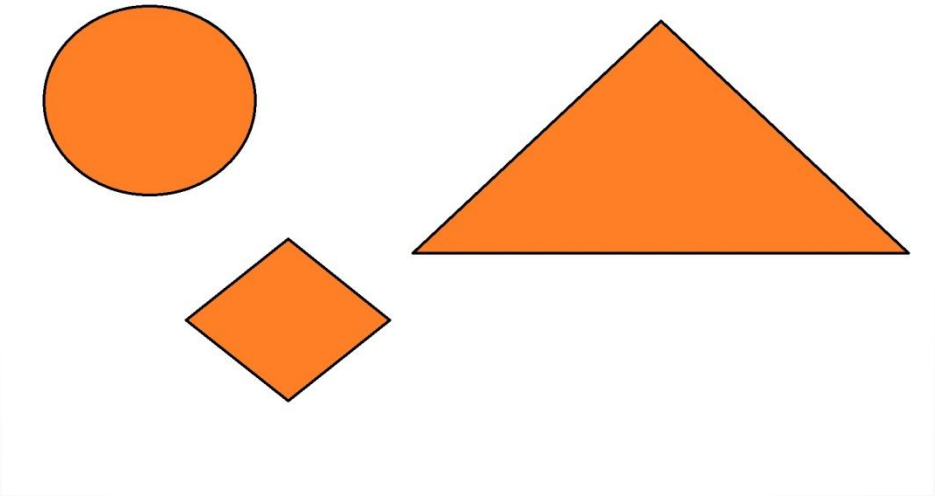
- In HSV avem o reprezentare cilindrica fata de cubul din RGB.
- In HSV se incepe de la rosu primar (0), se merge spre verde (120), apoi albastru (240) si inapoi la rosu (360)
  - toate culori primare la acele limite.
- Centrul cilindrului este gri.
- HSV vine de la HUE (culoare), saturatie (umbra, nivel de gri) si valoare (luminozitate)
- Spatiul culorilor din HSV seamana cu modul in care oamenii percep culorile.

Detectarea unui obiect de o anumita culoare folosind HSV

- Limitele de valori pentru HSV sunt urmatoarele:
- HUE: [0, 179]
- Saturatie: [0, 255]
  - Cat din culoare este amestecat cu alb.
- Valoare:[0, 255]
  - Cat din culoare este amestecat cu negru
- Valorile HUE (orientativ)
  - Portocaliu 0-22
  - Galben 22- 38
  - Verde 38-75
  - Albastru 75-130
  - Violet 130-160
  - Rosu 160-179

Detectarea unui obiect de o anumita culoare folosind HSV

- Pentru a gasi intervalul potrivit pentru un obiect, ar trebui sa observam exact culoarea sa.
- Consideram poza de mai jos:



```
void trackBars()
{
    namedWindow("Control", CV_WINDOW_AUTOSIZE);
    //Create trackbars in "Control" window
    createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
    createTrackbar("HighH", "Control", &iHighH, 179);

    createTrackbar("LowS", "Control", &iLowS, 255); //Sat (0 - 255)
    createTrackbar("HighS", "Control", &iHighS, 255);

    createTrackbar("LowV", "Control", &iLowV, 255);//Val (0 - 255)
    createTrackbar("HighV", "Control", &iHighV, 255);
    cout<<"iLowH = "<<iLowH<<endl;
}

void cautaObiectInPoza()
{
    while(true)
    {
        Mat imgHSV, imgThresholded;
        cvtColor(poza, imgHSV, COLOR_BGR2HSV);

        inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded);

        //aplicam deschidere pentru a elimina obiectele mici din fundal
        erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
        dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );

        //aplicam deschidere pentru a elimina obiectele mici din prim plan
        dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
        erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
        namedWindow("Cauta obiect", CV_WINDOW_AUTOSIZE);
        imshow("Cauta obiect", imgThresholded);
        if(waitKey(50) > 0)
            break;
    }
  }
```

Detectarea unui obiect de o anumita culoare folosind HSV

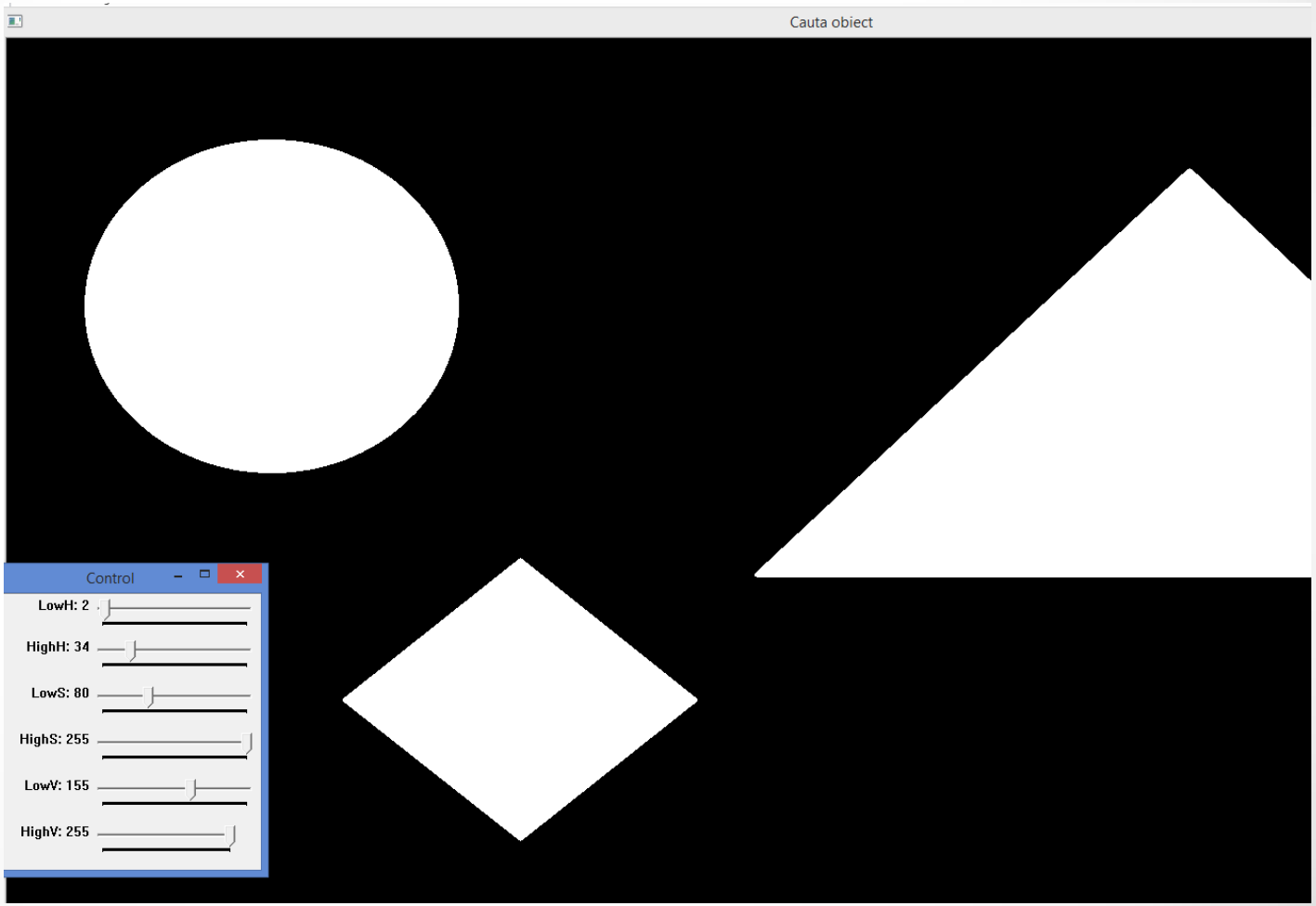
- void inRange(InputArray **src**, InputArray **lowerb**, InputArray **upperb**, OutputArray **dst**)
- Verifica daca fiecare element din src se gaseste intre lowerb si upperb.
  - Daca da, locatia respectiva ia valoarea 255 (alb)
  - Altfel, 0 (negru)

- In main():

```
trackBars();
cautaObiectInPoza();
```

Detectarea unui obiect de o anumita culoare folosind HSV

- Am aplicat cautare fina in slidere pana cand am identificat obiectele.



Aplicatii folosind camera web

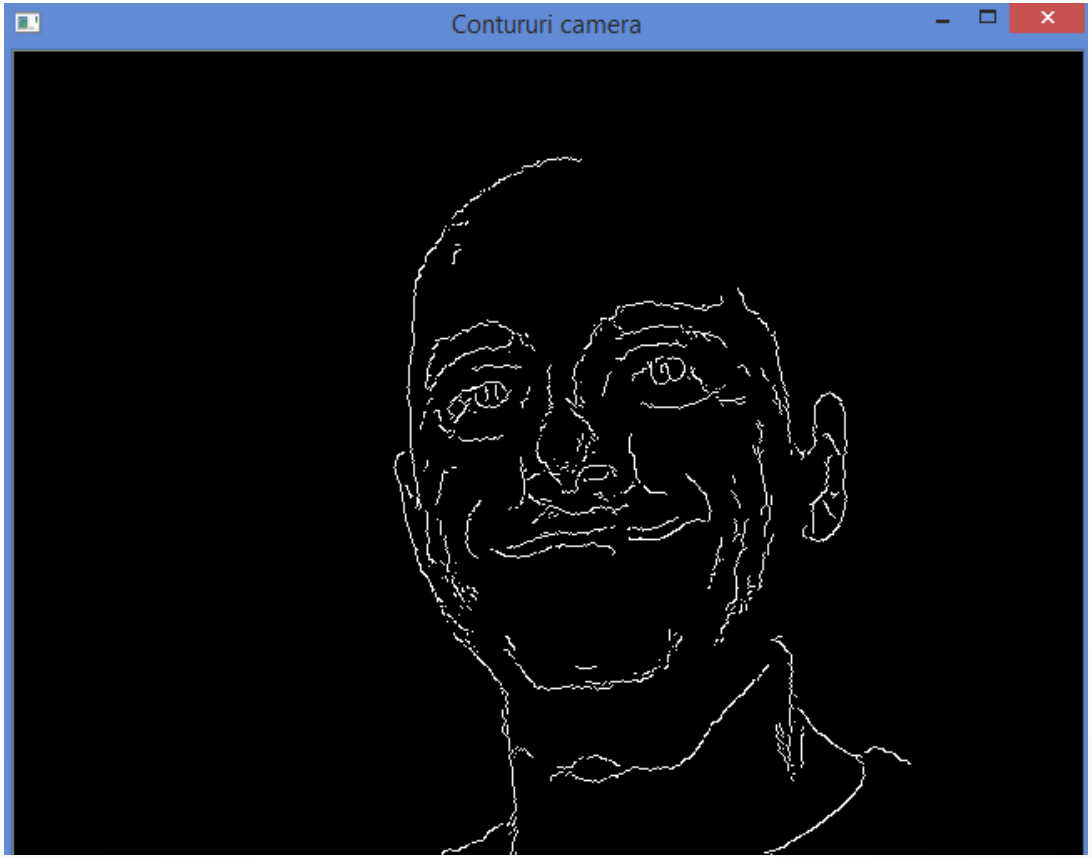
- Redare simpla de la camera web:

```
int main()
{
    VideoCapture cap(0); // open the default camera
    if ( !cap.isOpened() )  // daca nu se deschide camera, incheiem programul
    {
         cout << "Camera web nu poate fi pornita." << endl;
         return -1;
    }

    namedWindow("Redare camera",WINDOW_AUTOSIZE);
    while(true)
    {
        Mat frame;
        cap >> frame; // citim frame de la camera
        imshow("Redare camera", frame);
        if(waitKey(30) >= 0) break;
    }

    waitKey(0);
    return 0;
}
```

Contururi din camera web

```
void contururiCamera(Mat& frame, Mat& contururi)
{
    cvtColor(frame, contururi, CV_BGR2GRAY);
    GaussianBlur(contururi, contururi, Size(7,7), 1.5, 1.5);
    Canny(contururi, contururi, 0, 30, 3);
    imshow("Contururi camera", contururi);
}

int main()
{
    VideoCapture cap(0); // deschidem camera

    if ( !cap.isOpened() )  // daca nu se deschide camera, incheiem programul
    {
         cout << "Camera web nu poate fi pornita." << endl;
         return -1;
    }

    Mat camera;
    namedWindow("Contururi camera", WINDOW_AUTOSIZE);

    while(true)
    {
        Mat frame;
        cap >> frame; // citim frame de la camera
        contururiCamera(frame, camera);

        if(waitKey(30) >= 0) break;//cand atingem orice tasta
    }
   waitKey(0);
   return 0;
}
```

Contururi din camera web



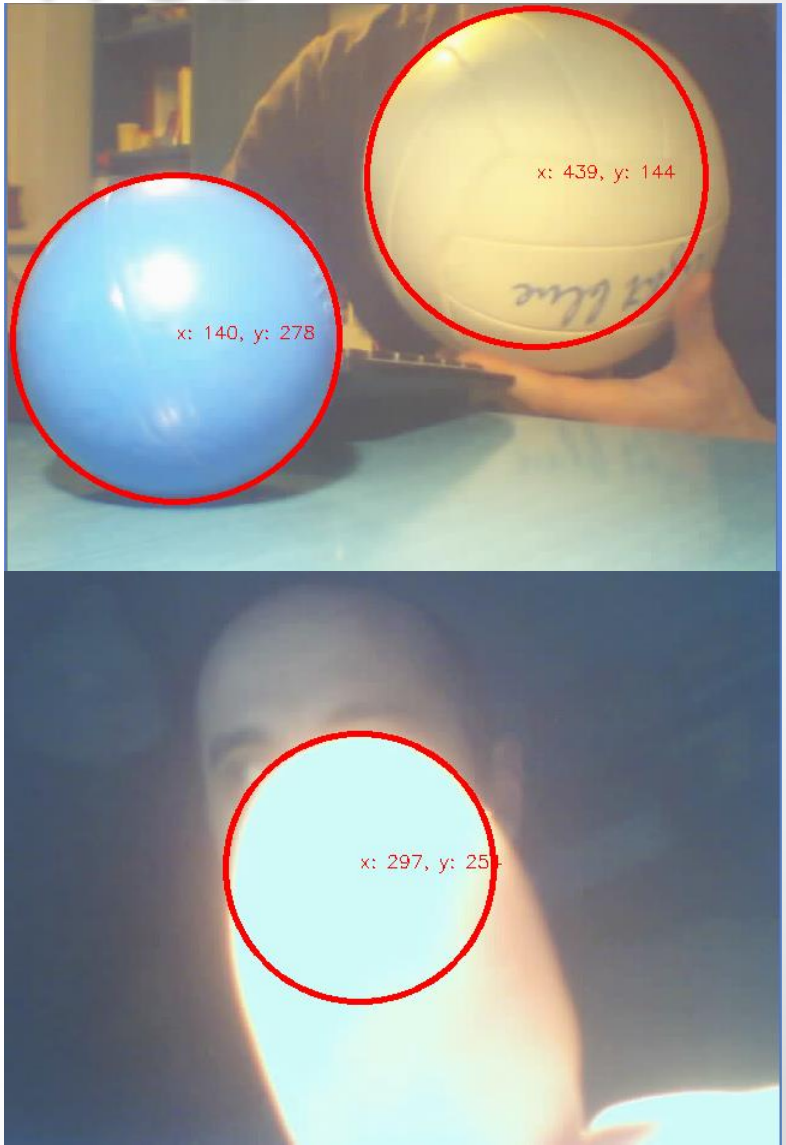
Cercuri detectate din camera web

- Pentru web cam, avem deja un while(true), deci nu ma avem nevo de altul pentru a actualiza valorile din slidere.

```
void deseneazaCercuri(Mat &poza)
{
    HoughCircles( pozaNecolor, cercuri, CV_HOUGH_GRADIENT, dp, minDist, param1, param2, minRadius, maxRadius );
    Mat copiePoza;
    poza.copyTo(copiePoza);
    /// Desenam cercurile gasite
    for( size_t i = 0; i < cercuri.size(); i++ )
    {
        Point center(cvRound(cercuri[i][0]), cvRound(cercuri[i][1]));
        int radius = cvRound(cercuri[i][2]);
        // centrul cercului
        putText(copiePoza, "x: " + to_string((int)cercuri[i][0]) + ", y: " + to_string((int)cercuri[i][1]),
            center, FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0,0,255), 1, 8, false );
        // cercul
        circle( copiePoza, center, radius, Scalar(0,0,255), 3, 8, 0 );
    }
    imshow( "Control Hough", copiePoza );
}

void detecteazaCercuri(Mat& poza)
{
    namedWindow("Control Hough", CV_WINDOW_AUTOSIZE);

    createTrackbar("dp", "Control Hough", &dp, 10);
    createTrackbar("minDist", "Control Hough", &minDist, 50);
    createTrackbar("param1", "Control Hough", &param1, 255);
    createTrackbar("param2", "Control Hough", &param2, 255);
    createTrackbar("minRadius", "Control Hough", &minRadius, 255);
    createTrackbar("maxRadius", "Control Hough", &maxRadius, 255);
    cvtColor( poza, pozaNecolor, CV_BGR2GRAY );
    // aplicam filtrare pentru a reduce zgomotul
    GaussianBlur( pozaNecolor, pozaNecolor, Size(5, 5), 2 );

    faraZero();
    deseneazaCercuri(poza);
}
```

Cercuri detectate din camera web

```
int main()
{
    VideoCapture cap(0); // deschidem camera

    if ( !cap.isOpened() )  // daca nu se deschide camera, incheiem programul
    {
         cout << "Camera web nu poate fi pornita." << endl;
         return -1;
    }

    while(true)
    {
        Mat frame;
        cap >> frame; // citim frame de la camera
        detecteazaCercuri(frame);
        if(waitKey(30) >= 0) break;//cand atingem orice tasta
    }
    waitKey(0);
    return 0;
}
```



Detectarea unui obiect de o anumita culoare folosind HSV

```
int main()
{
    VideoCapture cap(0); // deschidem camera

    if ( !cap.isOpened() )  // daca nu se deschide camera, incheiem programul
    {
         cout << "Camera web nu poate fi pornita." << endl;
         return -1;
    }
    trackBars();//pentru cautaObiectContururi

    Mat camera;
    int anteriorX = -1, anteriorY = -1;//pentru a trasa liniile in cautaObiectContururi/4
    Mat frame;
    cap >> frame; // citim un prim frame pentru a stabili marimea pozelor
    Mat imgLines = Mat::zeros( frame.size(), CV_8UC3 );//cream o imagine neagra in care sa trasam linii
    while(true)
    {
        cap >> frame;
        cautaObiectContururi(frame, imgLines, anteriorX, anteriorY);
        if(waitKey(30) >= 0) break;//cand atingem orice tasta
    }
    waitKey(0);
    return 0;
}
```

Detectarea unui obiect de o anumita culoare folosind HSV

- Transformam imaginea din BGR in HSV
- Aplicam deschidere si inchidere pentru a scapa de obiectele mici
- Afisam imaginea rezultata

```
void cautaObiectContururi(Mat& frame, Mat& imgLines, int& xAnterior, int& yAnterior)
{
    Mat imgHSV, imgThresholded;
    cvtColor(frame, imgHSV, COLOR_BGR2HSV);
    //ponderam imaginea pentru a detecta obiectul de culoarea dorita
    inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded);

    //aplicam deschidere pentru a elimina obiectele mici din fundal
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
    dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );

    //aplicam inchidere pentru a elimina obiectele mici din prim plan
    dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );

    imshow("Thresholded Image", imgThresholded); //afisam imaginea ponderata
```

Detectarea unui obiect de o anumita culoare folosind HSV

- Continuarea metodei din slide-ul precedent.

```
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
findContours( imgThresholded, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
if(contours.size() > 0)
{
    vector<Point2f>center( contours.size() );
    vector<float>radius( contours.size() );
    //caut cercul cu cea mai mare raza
    size_t k = -1;
    double razaMax = 0;
    for( size_t i = 0; i < contours.size(); i++ )
    {
        minEnclosingCircle( (Mat)contours[i], center[i], radius[i] );
        if(razaMax < radius[i])
        {
            razaMax = radius[i];
            k = i;//cercul k ne intereseaza
        }
    }
    int posX = (int)center[k].x;
    int posY = (int)center[k].y;
    if (xAnterior >= 0 && yAnterior >= 0 && posX >= 0 && posY >= 0)
    {
    //Linie albastra de la punctul anterior la cel curent
    line(imgLines, Point(posX, posY), Point(xAnterior, yAnterior), Scalar(255,0,0), 2);
    }

    xAnterior = posX;
    yAnterior = posY;
}

frame = frame + imgLines;//punem liniile peste imagine
imshow("Original", frame); //afisam imaginea originala
}
```

Detectarea unui obiect de o anumita culoare folosind HSV

- Se utilizeaza contururi pentru a gasi centrul cercului care cuprinde cel mai mare obiect detectat.
  - Util in cazul in care mai scapa obiecte mici
- Din centrul cercului din frame-ul precedent pana la centrul cercului din frame-ul curent trasam o linie.
- La final, poza cu linii se uneste cu frame-ul initial.

Detectarea unui obiect de o anumita culoare folosind HSV

- Cu slider-ele controlate prin trackbars stabilim manual care este obiectul de o anumita culoare care ne intereseaza.

```
void trackBars()
 {
     namedWindow("Control", CV_WINDOW_AUTOSIZE);

     createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
     createTrackbar("HighH", "Control", &iHighH, 179);

     createTrackbar("LowS", "Control", &iLowS, 255); //Sat (0 - 255)
     createTrackbar("HighS", "Control", &iHighS, 255);

     createTrackbar("LowV", "Control", &iLowV, 255);//Val (0 - 255)
     createTrackbar("HighV", "Control", &iHighV, 255);
 }
```

Detectarea unui obiect de o anumita culoare folosind HSV