Laborator 2 SRF: Detecția obiectelor pe baza culorii. Folosirea fluxurilor video

Scopul lucrării
Această lucrare ne va arăta cum putem detecta obiectele grafice pe baza culorii lor, diferite de culoarea backgroundului, cum să le urmărim și cum să le marcăm. Fiecare obiect va fi văzut ca un conglomerat de pixeli (blob) de culoare asemănătoare. Vom modifica apoi poziția obiectului urmărit pentru a vedea dacă ordinatorul sesizează schimbarea poziției acestuia.
Tot în această lucrare vom învăța cum pot fi folosite în general librăriile de funcții în Processing și vom începe cu folosirea librăriei Video, care ne permite să achiziționăm fluxuri video, fie direct de la webcam, fie dintr-un fișier local.

1. Breviar teoretic:

Libraria Video este una dintre cele mai utile librării de funcții care pot fi integrate în mediul de dezvoltare Processing. Clasele definite în aceasta se ocupă cu preluarea si prelucrarea fluxurilor video de la webcam sau dintr-un fișier local. Clasa Capture tratează un flux video de la webcam, în timp ce clasa Movie folosește un flux video la un fișier local. Video este o librărie de funcții cu aplicații în preluarea și prelucrarea fluxurilor video, acestea fiind de tip webcam sau fișier video.
Clasa Movie încarcă respectivul clip video ales de utilizator, având la dispoziție diferite funcții de control, precum read(), play(), loop(), noLoop(), etc, iar clasa Capture se ocupă cu preluarea în timp real a semnalului video oferit de webcam-ul atașat la PC.

1.1.Clasa Capture:

Pentru folosirea corectă a clasei Capture, trebuie urmați următorii pași:

Pas1: Scrierea liniei de comandă folosită la importarea librăriei.

import processing.video.;*

Pas2: Declararea unui obiect cu numele video care să aparțină clasei Capture:

Capture video;

Pas3: Initializările de variabile în funcția setup():

nume_webcam=Capture.list()[0]
video=new Capture(this, dimx, dimy,nume_webcam)

Pas4: Pornirea fluxului video:

video.start();

Pas5: Incarcarea unui nou cadru:

La această etapă putem apela la două metode diferite:

A. Prima metodă, bazată pe o funcție-eveniment separată, care se activează doar atunci când primim un nou cadru de la fluxul video:

```
void captureEvent(Capture video){
        video.read();
}
```

B. A doua metodă, în cadrul funcției draw() verificăm fiecare cadru în parte:

```
if(video.available()){
        video.read();
}
```

Pas6: Afisarea imaginii
image(video,0,0)

1.2. Clasa Movie

Clasa Movie are același principiu de utilizare. Rolul ei este cu preluarea și folosirea fluxurilor video provenind de la fișiere. Pașii necesari utilizării unui obiect aparținând acestei clase sunt asemănători cu cei exemplificați anterior:

Pas1: Importarea librăriei
import processing.video.*;

Pas2: Declararea variabilei obiect:
Movie video;

Pas3: Initializarea variabilelor obiect in functia setup():
video=new Movie(this, nume_fisier_film)

Pas4: Pornirea fluxului video:
video.play(); (daca videoul este rulat o singura data)

video.loop(); (daca videoul se repeta)

Pas5: Încărcarea unui nou cadru:

Metoda 1: folosirea unei funcții eveniment separate:

```
void movieEvent(Movie video){
        video.read();
}
```

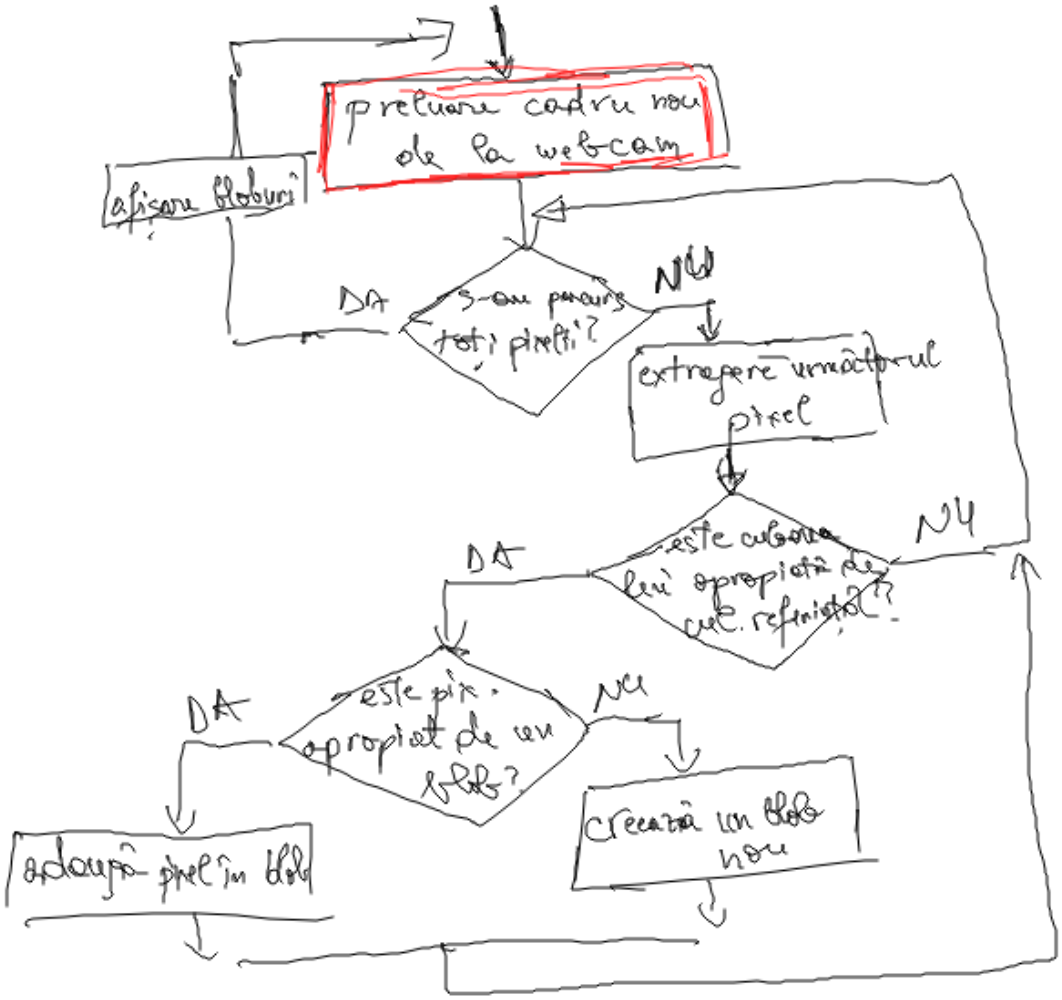
Metoda 2: testarea in cadrul functiei draw():

```
if(video.available()){
        video.read();
}
```

Pas6: După achiziție, urmează prelucrarea sau afișarea cadrului curent:
image(video,0,0)

1.3. **Algoritmul de detecție a blob-urilor**

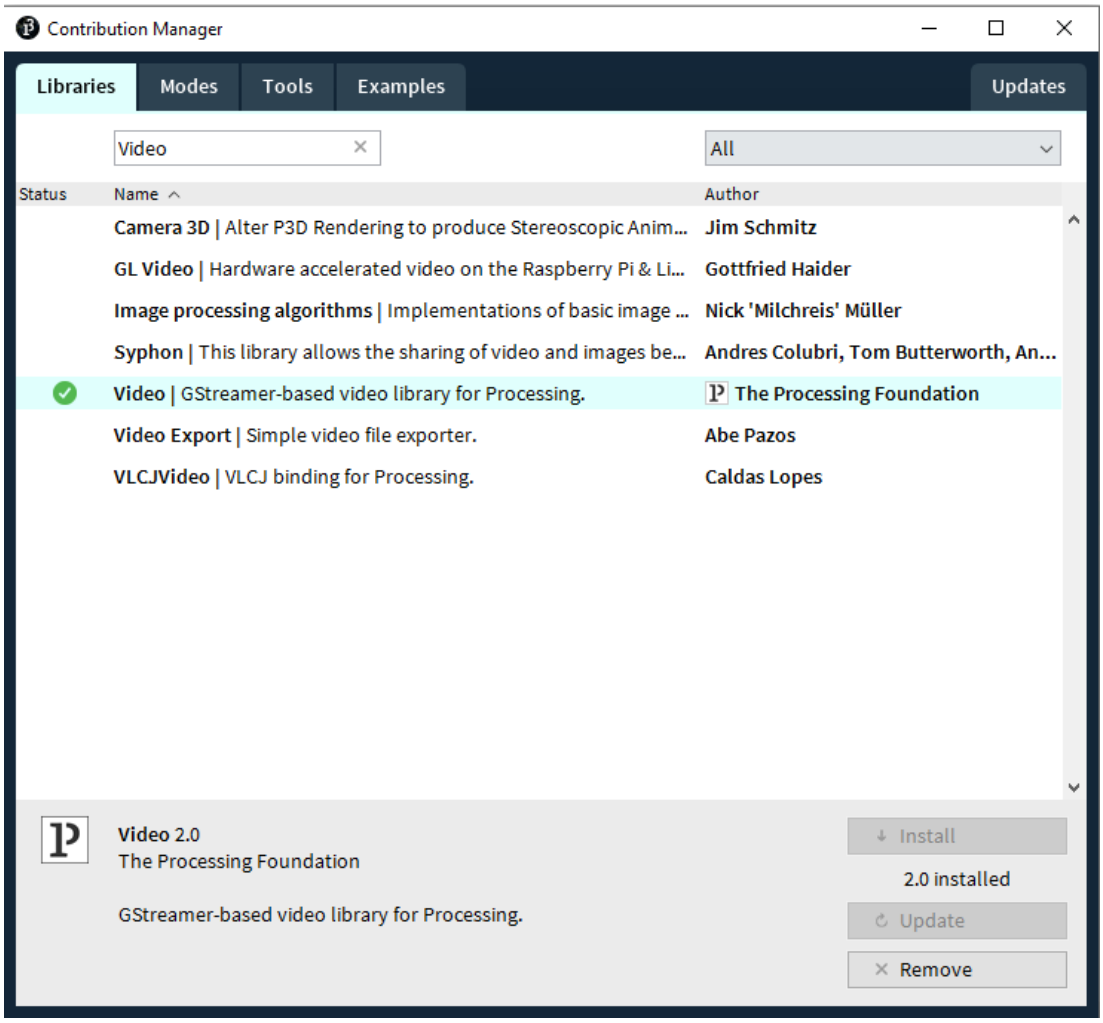
Blobul este o grupare de pixeli învecinați care au proprietăți asemănătoare (culoare sau luminozitate). Algoritmul de detecție a blob-urilor dintr-un cadru primit de la webcam este afișat în imaginea de mai jos.



2. Desfășurarea lucrării:

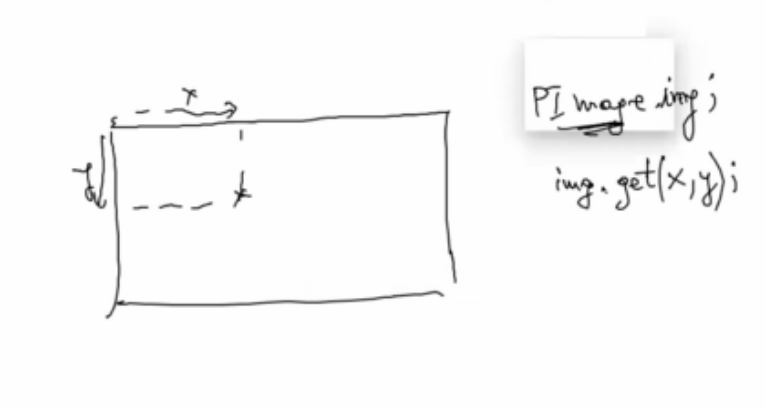
Pentru această lucrare trebuie să instalăm în cadrul programului Processing, librăria Video.

Pentru a realiza acest pas vom deschide Processing, iar din bara de meniuri vom ***alege Sketch -> Import library -> Add Library*** și vom căuta si instala librăria ***Video***.



Librăria Video definește două clase importante: Capture (pentru stream-urile video primite de la webcam) și Movie (pentru fisiere de tip video).

În laboratorul numărul 1 s-a lucrat cu obiecte de tip **PImage**. Să ne reamintim că dacă avem un obiect ***img*** de tip PImage, putem culege culoarea unui pixel de la pozitia x si y folosind funcția **img.get(x,y),** în timp ce funcția **img.set(x,y,culoare)** ne permite să setam culoarea acelui pixel.



Cadrele dintr-un stream video se comportă fiecare exact ca și un obiect de tip PImage. Fiecare asemenea cadru are aceleași variabile-membru și metode precum obiectele de tip PImage.

Se pornește de la programul următor:

Program principal - **BlobTracking.pde**

```
// Daniel Shiffman
// http://codingtra.in
// http://patreon.com/codingtrain
// Code for: https://youtu.be/1scFcY-xMrI

import processing.video.*;

Capture video;

color trackColor;
float threshold = 25;// threshold este pragul pentru distanta intre culori
float distThreshold = 50;// distThreshold este pragul de detectie pentru distanta intre 2 pixeli pentru a fi considerati in acelasi obiect

ArrayList<Blob> blobs = new ArrayList<Blob>();

void setup() {
  size(640, 360);// Dimensiunea ferestrei
  String[] cameras = Capture.list();// Lista dispozitivelor capture disponibile
  printArray(cameras);
  video = new Capture(this, 640, 360, cameras[1]);// Selectati o camera functionala din lista dispozitivelor
  video.start();
  trackColor = color(255, 0, 0);// Initial culoarea de urmarire este setata ca fiind rosu, dar dupa ce apasam click se va modifica
}


void captureEvent(Capture video) {
  video.read();
}

/* Functia keypressed() este apelata de fiecare data cand o tasta este apasata.
Tasta care a fost apasata este pastrata in variabila key. */
void keyPressed() {
  if (key == 'a') { // Cand se apasa tasta 'a' se mareste distanta de threshold cu 5
    distThreshold+=5;
  } else if (key == 'z') { // Cand se apasa tasta 'z' se micsoreaza distanta de threshold cu 5
    distThreshold-=5;
  }
  if (key == 's') { // Cand se apasa tasta 's' se mareste threshold-ul cu 5
    threshold+=5;
  } else if (key == 'x') { // Cand se apasa tasta 'x' se micsoreaza threshold-ul cu 5
    threshold-=5;
  }
  println(distThreshold);
}

void draw() {
  video.loadPixels();
  image(video, 0, 0);

  blobs.clear();
```

```
  // Parcurgem fiecare pixel
  for (int x = 0; x < video.width; x++ ) {
    for (int y = 0; y < video.height; y++ ) {
      int loc = x + y * video.width;
      // Care este culoarea curenta?
      color currentColor = video.pixels[loc];

      // Se preiau componentele de culoare de la pixelul curent analizat
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      // Se preiau componentele de culoare de la pixelul de referinta(care trebuie urmarit)
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);

      float d = distSq(r1, g1, b1, r2, g2, b2);// Se calculeaza patratul distantei intre cele 2 culori

      if (d < threshold*threshold) {

        boolean found = false;
        for (Blob b : blobs) { // Parcurgem fiecare element al tabloului blobs
          if (b.isNear(x, y)) { // Daca blob-ul este aproape de pozitia pixelului curent
            b.add(x, y); // Adaugam pixelul curent la blob
            found = true;
            break;
          }
        }

        if (!found) {
          Blob b = new Blob(x, y);
          blobs.add(b);
        }
      }
    }
  }

  for (Blob b : blobs) { // Parcurgem fiecare element al tabloului blobs
    if (b.size() > 500) { // Daca marimea blob-ului este mai mare decat 500
      b.show();// Este afisat dreptunghiul blob-ului
    }
  }

  textAlign(RIGHT);
  fill(255);
  text("distance threshold: " + distThreshold, width-10, 25);
  text("color threshold: " + threshold, width-10, 50);
}


// Functii distanta fara radacina patrata pentru optimizare
float distSq(float x1, float y1, float x2, float y2) {
  float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1);
  return d;
}


float distSq(float x1, float y1, float z1, float x2, float y2, float z2) {
  float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) +(z2-z1)*(z2-z1);
```

```
  return d;
}

void mousePressed() {
  // Salveaza culoarea de unde este apasat mouse-ul in variabila trackColor
  int loc = mouseX + mouseY*video.width;
  trackColor = video.pixels[loc];
}
```

Clasa Blob - **Blob.pde**

```
// Daniel Shiffman
// http://codingtra.in
// http://patreon.com/codingtrain
// Code for: https://youtu.be/1scFcY-xMrI

class Blob {
  float minx;
  float miny;
  float maxx;
  float maxy;

  ArrayList<PVector> points;

  Blob(float x, float y) {
    minx = x;
    miny = y;
    maxx = x;
    maxy = y;
    points = new ArrayList<PVector>();
    points.add(new PVector(x, y));
  }

  void show() { //Functia show() construieste dreptunghiul blob-ului
    stroke(0);
    fill(255);
    strokeWeight(2);
    rectMode(CORNERS);
    rect(minx, miny, maxx, maxy);

    for (PVector v : points) {
      //stroke(0, 0, 255);
      //point(v.x, v.y);
    }
  }


  void add(float x, float y) {
    points.add(new PVector(x, y));
    // Dupa ce adaugam pixelul la blob trebuie sa reverificam limitele x,y
    minx = min(minx, x);
    miny = min(miny, y);
    maxx = max(maxx, x);
    maxy = max(maxy, y);
  }
```

```
 float size() { // Se calculeaza suprafata dreptunghiului care incercuieste blob-ul
  return (maxx-minx)*(maxy-miny);
 }

 boolean isNear(float x, float y) {

  // Cel mai apropiat punct al strategiei blob
  float d = 10000000;
  for (PVector v : points) {
   float tempD = distSq(x, y, v.x, v.y);
   if (tempD < d) {
    d = tempD;
   }
  }

  if (d < distThreshold*distThreshold) {
   return true;
  } else {
   return false;
  }
 }
}
```

Cerințe

1. Explicați codul de mai sus folosind comentarii în cod.

2. Să se înlocuiască în cod .pixels[loc] cu o metodă directă (dar mai înceată) folosind x și y (să nu mai trebuiască să calculăm variabila loc);

3. Să putem modifica de la tastatură (folosind tastele c și d) suprafața minimă pe care trebuie să o aibă un obiect ca să fie încercuit;

4. Faceți dreptunghiul de selecție transparent;

5. Modificați chenarul dreptunghiului de selecție în culoarea verde;

6. Găsiți funcțiile care calculează pătratul distanțelor (între culori sau pixeli) și modificați codul pentru a lucra direct cu distanțe (în loc să memorăm pătratul acestora).

Rezolvare:

2. Pentru a nu mai calcula variabila loc, putem folosi direct metoda get() pentru a obține culoarea.

În programul principal BlobTracking vom modifica următoarele linii de cod din metoda draw():

```
for (int x = 0; x < video.width; x++ ) {
  for (int y = 0; y < video.height; y++ ) {
    int loc = x + y * video.width;
    // Care este culoarea curenta?
    color currentColor = video.pixels[loc];
```

Modificările sunt următoarele:

```
for (int x = 0; x < video.width; x++ ) {
   for (int y = 0; y < video.height; y++ ) {
     // Care este culoarea curenta?
     color currentColor = video.get(x,y);
```

3. Pentru a putea modifica de la tastatură suprafața minimă pentru încercuirea unui obiect, în primul rând în programul principal vom declara global o variabilă surface.

```
int surface = 500;
```

Vom folosi această variabilă pentru a decide când se va afișa dreptunghiul și vom modifica următoarea linie de cod din metoda draw():

```
if (b.size() > 500) { // Daca marimea blob-ului este mai mare decat 500
```

Se va modifica cu:

```
if (b.size() > surface) { // Daca marimea blob-ului este mai mare decat surface
```

La sfârșitul metodei draw() se va adăuga următoarea linie:

```
text("surface threshold: " + surface, width-10, 75);
```

În continuare în metoda keyPressed() vom adăuga următoarele linii de cod:

```
if (key == 'c') {
   surface+=50;
  } else if (key == 'd') {
    surface-=50;
  }
```

Cand se va apăsa tasta c suprafata se va mării cu 50, iar când se va apăsa tasta d se va micșora cu 50.

4. Pentru a modifica dreptunghiul de selecție vom modifica in clasa Blob metoda show().

Mai precis vom modifica linia de cod:

```
fill(255);
```

Aceasta devenind:

```
fill(255,0);//continut dreptunghi
```

În cazul de față metoda fill() are următoarele sintaxe:

```
fill(gray)
fill(gray, alpha)
```

Parametrii:

gray = float: număr care specifica valoarea dintre alb și negru

alpha = float: opacitatea umplerii

Pentru un dreptunghi de selecție transparent am ales alpha egal cu 0.

5. Pentru a modifica culoarea dreptunghiului de selecție vom modifica tot în clasa Blob metoda show().

Se va modifica linia:

```
stroke(0);
```

Cu:

```
stroke(0,255,0);//contur dreptunghi
```

În acest caz metoda stroke() are următoarele sintaxe:

```
stroke(gray)
stroke(v1, v2, v3)
```

Parametrii:

gray = float: specifică valoarea dintre alb și negru

v1 = float: valoarea de roșu

v2 = float: valoarea de verde

v3 = float: valoarea de albastru

Pentru un dreptunghi de selecție de culoare verde am folosit valoarea 0 pentru canalul roșu, 255(maxim) pentru canalul verde și 0 pentru canalul albastru.

6. Funcțiile care calculează pătratul distanțelor se află în programul principal.

```
float distSq(float x1, float y1, float x2, float y2) {
  float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1);
  return d;
}

float distSq(float x1, float y1, float z1, float x2, float y2, float z2) {
  float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) +(z2-z1)*(z2-z1);
  return d;
}
```

Pentru a calcula doar distanțele vom folosi sqrt, codul devenind:

```
float distSq(float x1, float y1, float x2, float y2) {
  float d = sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
  return d;
}

float distSq(float x1, float y1, float z1, float x2, float y2, float z2) {
  float d = sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) +(z2-z1)*(z2-z1));
  return d;
}
```

Deoarece am modificat pătratul distanțelor cu distanțele pure va fi nevoie să mai modificăm următoarele linii de cod:

În metoda draw()

```
    if (d < threshold*threshold) {
```

Se va înlocuii cu:

```
if (d < threshold) {
```

În clasa Blob în metoda isNear()

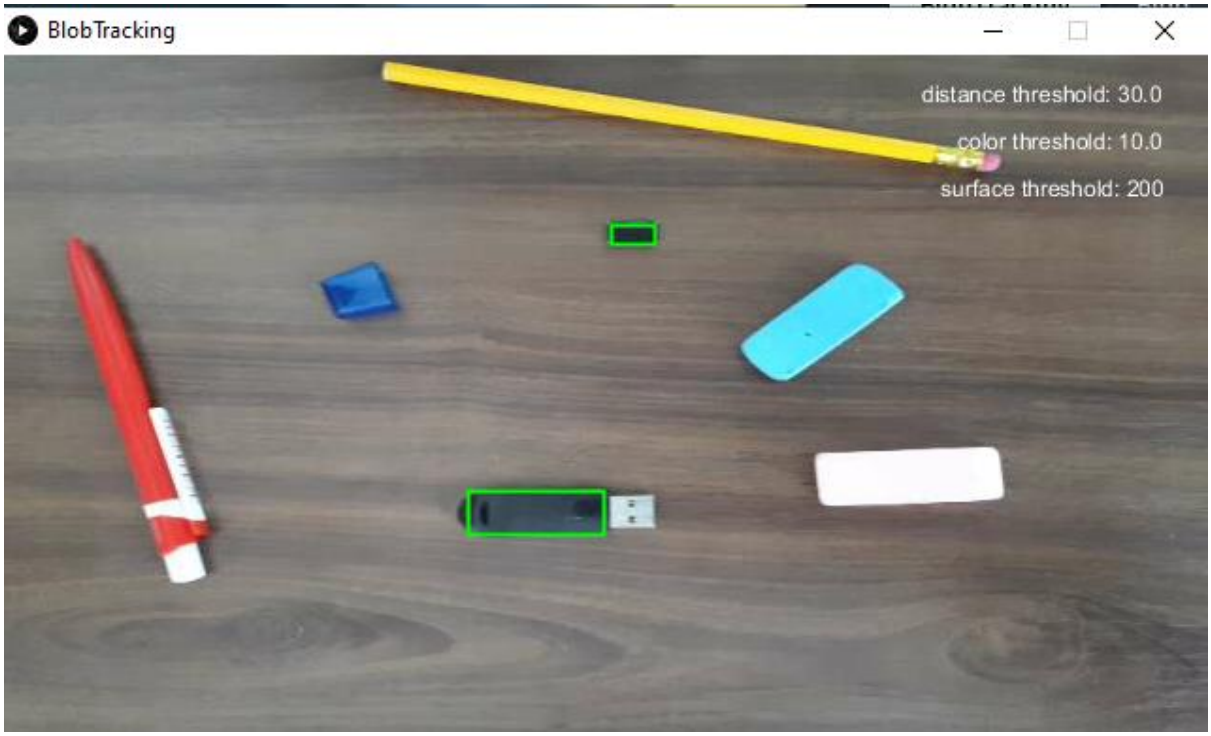
```
if (d < distThreshold*distThreshold) {
```

Se va înlocuii cu:

```
if (d < distThreshold) {
```

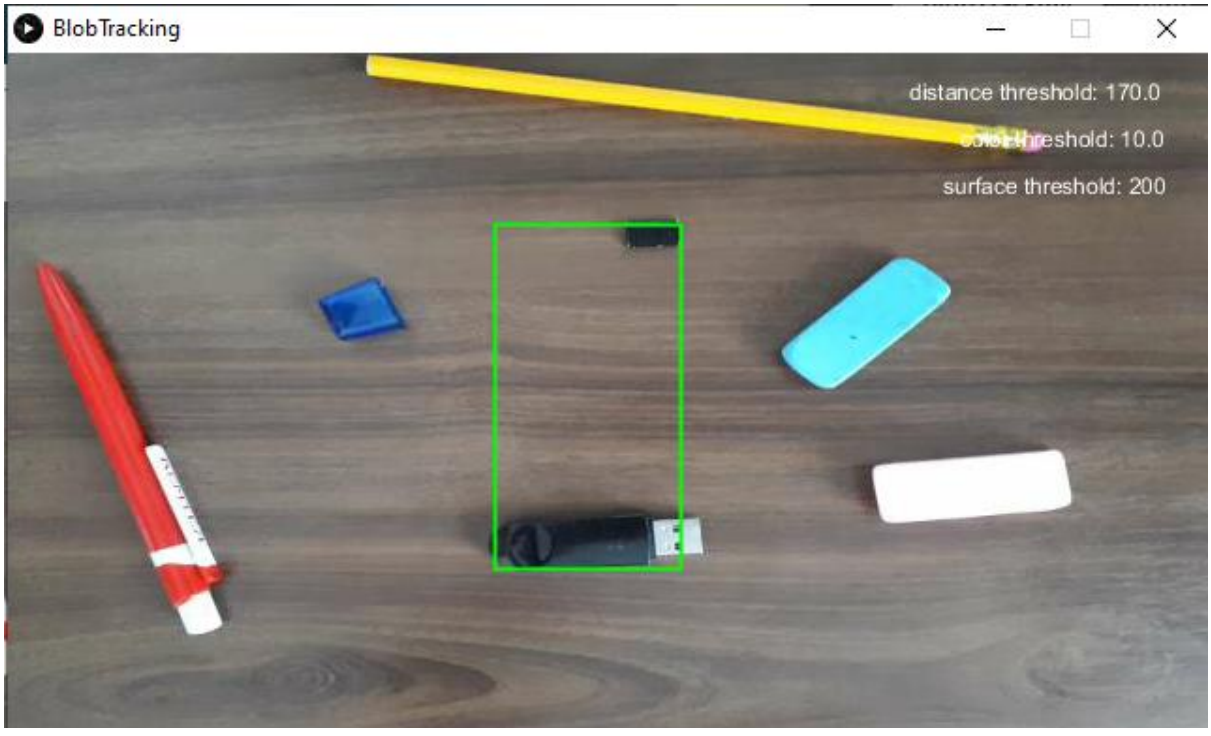
Rezultate finale:

Acesta este rezultatul după ce am dat click pe partea neagră a stick-ului, pentru a selecta culoarea negru ca și culoare de referință.



Pentru a detecta doar chenarul de la stick am mărit suprafața de threshold la 300.

Mărind distance thresholdului la 170 cele 2 blob-uri s-au unit într-un singur blob.



Folosind un color threshold de 80 și apăsând pe pixul roșu din stânga putem observa că si obiectul din dreapta a căpătat un chenar în dreptul său.

Pentru a putea diferenția culorile celor 2 obiecte am redus color threshold.