

Recunoașterea de flăcări în materiale video

Flame Recognition



Comsa maria denisa

materie: Procesarea imaginilor

Universitatea “Lucian Blaga” din Sibiu, Facultatea de Stiinte

Cuprins

[Recunoașterea de flăcări în materiale video 2](#_Toc95847834)

[(Flame Recognition) 2](#_Toc95847835)

[Descrierea problemei: 2](#_Toc95847836)

[Detalii despre tema si aleasa si motivatia acesteia: 2](#_Toc95847837)

[Componentele programului: 2](#_Toc95847838)

[Mediul de programare folosit: 2](#_Toc95847839)

[Limbaj de programare folosit: 2](#_Toc95847840)

[Solutia propusa pentru rezolvare, cu accent pe concept si instrumente: 2](#_Toc95847841)

[1. detectionByColor() 3](#_Toc95847842)

[2. detectionByMovement() 3](#_Toc95847843)

[3. showFire() 3](#_Toc95847844)

[BIBLIOGRAFIE: 4](#_Toc95847845)

# Recunoașterea de flăcări în materiale video

# (Flame Recognition)

## Descrierea problemei:

**Recunoașterea de flăcări în materiale video (Flame Recognition). Detecția vizuală a focului poate fi aplicată în condiții în care metodele convenționale de detectare nu se pot aplica, cum ar fi recunoașterea de flăcări în materiale video filmat. Se cere detectarea flăcărilor în secvențe video folosind informații de culoare și mișcare. Se pot folosi segmentare bazată pe culoare, eroziune, region growing, optical flow.**

## Detalii despre tema si aleasa si motivatia acesteia:

Detectarea sau recunoasterea flacarilor a devenit un subiect importand in sfera detectarii de obiecte. In zilele noastre, majoritatea locurilor si cladirilor folosesc metode conventionale de prevenire a aparitilor incendiilor, cum ar fi: detectoarele de fum, detectoarele de caldura, si fotodetectoarele. Cu toate acestea, senzorii detectoarelor mentionate anterior pot avea o acuratete mai scazuta daca sunt plasate in zone cu imprejurimi mai aglomerate.

Odata cu evolutia tehnologiei de procesare a imaginilor a fost facuta posibila incercarea folosirii acesteia in detectia de flacari, folosind informatii de culoare si detectie a miscarii in materiale video.

## Componentele programului:

Proiectul este format dintr-o clasa principala numita FlameRecognition in care am implementat metodele detectionByColor(), detectionByMovement() si showFire(). De asemenea, am creat o clasa ajutatoare numita ImgProcHelper.

## Mediul de programare folosit:

* Intellij Community Edition

## Limbaj de programare folosit:

* Java, utilizand libraria OpenCV

## Solutia propusa pentru rezolvare, cu accent pe concept si instrumente:

Pentru rezolvarea fiecarei cerinte a problemei am realizat cate o metoda denumita in functie de problema pe care o trateaza, de exemplu pentru detectarea flacarilor in functie de culoare, am creeat metoda detectionByColor().

Clasa principala FlameRecognition implementeaza urmatoarele metode:

# detectionByColor()

private boolean detectionByColor() {  
 VideoCapture fire\_video = new VideoCapture("Slow Motion Fire Background Fire Backgrounds.mp4");  
 int nrRed = 0;  
 if (fire\_video.isOpened()) {  
 while (true) {  
 fire\_video.read(frame);  
 if (frame.empty()) break;  
  
 Imgproc.GaussianBlur(frame, blur, new Size(21, 21), 0);  
 Imgproc.cvtColor(blur, hsv, Imgproc.COLOR\_BGR2HSV);  
  
 Scalar upper = new Scalar(18, 50, 50);  
 Scalar lower = new Scalar(35, 255, 255);  
  
 Core.inRange(hsv, upper, lower, mask);  
 Core.bitwise\_and(frame, hsv, output, mask);  
 nrRed = Core.countNonZero(mask);  
 ImgProcHelper.resizeImage(frame);  
 ImgProcHelper.resizeImage(output);  
  
 if (nrRed > 100) {  
 ImgProcHelper.addText(frame, "FLAME RECOGNIZED");  
 }  
 HighGui.imshow("Frame", frame);  
 HighGui.imshow("output", output);  
 int key = HighGui.waitKey(20);  
 if (key == 27)  
 break;  
 }  
 }  
 HighGui.destroyAllWindows();  
 return nrRed > 100;  
}

* se creeaza obiectul fire\_video in care vom stoca video-ul folosind ca exemplu al tratarii problemei
* incepem citirea video-ului folosind un while infinit (pana la momentul apasarii tastei esc, semnalat aici de if(key==27) break;)
* Se citeste video-ul frame cu frame, fiecare frame fiind o matrice de imagine
* Aplicam un blur imaginilor pentru a elimina zgomotul, astfel tratarea problemei va merge mult mai usor
* In continuare convertim colorspace-ul BGR in HSV, un proces comun in detectarea culorilor, deoarece HSV este mult mai robust in contextul schimbarilor externale de lumina. Acest lucru inseamna ca in cazul schimbarilor minore in lumina externe (cum ar fi umbrele), valorile nuantelor curente variaza mai putin decat in cazul valorile RGB.
* Se declara doua valori scalare, upper si lower, care reprezinta gama de culori a focului HSV (Hue, Saturation, Value)
* Intr-o matrice numita mask, stocam informatiile de culoare din range-ul mentionat anterior gasite in matricea hsv.
* In valoarea nrRed stocam numarul de pixeli gasiti anterior, numarul de pixeli creste odata cu aparitia flacarilor.

# detectionByMovement()

private boolean detectionByMovement() {  
  
 List<MatOfPoint> cnts = new ArrayList<>();  
 VideoCapture fire\_video = new VideoCapture("Slow Motion Fire Background Fire Backgrounds.mp4");  
 if (fire\_video.isOpened()) {  
 while (true) {  
 fire\_video.read(frame);  
 if (frame.empty()) {  
 break;  
 }  
  
 fire\_video.read(frame);  
  
 Imgproc.*cvtColor*(frame, firstFrame, Imgproc.*COLOR\_BGR2GRAY*);  
 Imgproc.*GaussianBlur*(firstFrame, firstFrame, new Size(21, 21), 0);  
  
 while (fire\_video.read(frame)) {  
  
 Imgproc.*cvtColor*(frame, gray, Imgproc.*COLOR\_BGR2GRAY*);  
 Imgproc.*GaussianBlur*(gray, gray, new Size(21, 21), 0);  
  
 Core.*absdiff*(firstFrame, gray, frameDiff);  
 Imgproc.*threshold*(frameDiff, thresh, 25, 255, Imgproc.*THRESH\_BINARY*);  
  
 Imgproc.*erode*(thresh, thresh, new Mat(), new Point(-1, -1), 2);  
 Imgproc.*findContours*(thresh, cnts, new Mat(), Imgproc.*RETR\_EXTERNAL*, Imgproc.*CHAIN\_APPROX\_SIMPLE*);  
  
 for (MatOfPoint cnt : cnts) {  
 if (Imgproc.*contourArea*(cnt) < 500) {  
 continue;  
 }  
 return true;  
 }  
 }  
 int key = HighGui.*waitKey*(20);  
 if (key == 27)  
 break;  
 }  
 }  
 return false;  
}

* ca in metoda anterioara, citim video-ul si stocam fiecare frame
* convertim formatul video-ului in grayscale si aplicam un blur primului frame din video
* in continuare calculam diferenta dintre primul frame si frame-ul curent “grey”
* apoi intr-o matrice de tip threshold pastram pixelii din locurile unde frame-ul curent e diferit de frame-ul anterior, in valoarea specificata de threshold
* folosim un filtru de eroziune pentru a elimina zgomotul, iar informatia o stocam intr-un array de puncte
* dezavantajul a ceea ce am realizat pana acum consta in faptul ca in cazul miscarilor lente, se primeste foarte putina informatie, datorita schimbarilor mici intre frame-uri, uneori algoritmul nu primeste deloc rezultate

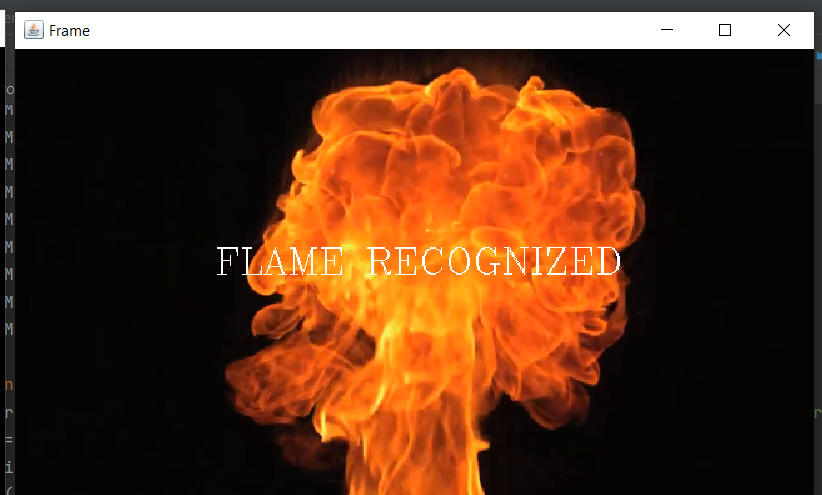
# showFire()

public void showFire() {  
 if (detectionByMovement()) {  
 if (detectionByColor()) {  
 System.*out*.println("Flame detected");  
 } else {  
 System.*out*.println("No colour detected");  
 }  
 } else {  
 System.*out*.println("No movement detected");  
 }  
}

* ajuta la apelarea functiilor anterioare, in interiorul carora exista si afisarea video-urilor modificate
* in cazul in care metodele nu detecteaza culoare, sau miscare afiseaza un mesaj in consola

## Rezultat:

La rularea aplicatiei va aparea urmatoarea fereastra, care reprezinta video-ul citit cu textul specific detectiei de flacari(in cazul existentei).



## BIBLIOGRAFIE:

* <https://www.tutorialspoint.com/java_dip/eroding_dilating.htm>
* <https://books.google.ro/books?id=sDNTEAAAQBAJ&pg=PA173&lpg=PA173&dq=what+is+%5B18,50,50%5D+%5B35,255,255%5D&source=bl&ots=JJR6kXknqr&sig=ACfU3U3zZhoaZMwpcxpzWwfJ6HyURWcm9g&hl=en&sa=X&ved=2ahUKEwigvtXQkoL2AhXNyqQKHfVTAaQQ6AF6BAgMEAM#v=onepage&q=what%20is%20%5B18%2C50%2C50%5D%20%5B35%2C255%2C255%5D&f=false>
* <https://en.wikipedia.org/wiki/Thresholding_(image_processing)>
* <https://www.quora.com/Why-do-we-convert-RGB-to-HSV#:~:text=The%20reason%20we%20use%20HSV,relatively%20lesser%20than%20RGB%20values>.