Universitatea POLITEHNICA din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Aplicație de identificare a ceasului pe baza imaginilor cu ceas analogic

Proiect ASPICV

Coordonatori:
Prof. Dr. Ing. Laura-Maria FLOREA
Drd. Ing. Andrei-Mircea RACOVIŢEANU

Masteranzi TAID:
Ing. Gheorghe-Iulian CHIVU
Ing. Alexandra-Elena ȘERBAN

Anul 2022

Cuprins

Cerința Problemei 4

1. Pașii necesari rezolvării problemei 5
1.1. Pre-procesarea imaginii 5
1.2. Transformata Hough Probabilistică 6
1.3. Clasificarea liniilor 7
1.4. Calcularea unghiurilor 8
1.5. Corelare grade - timp 9

2. Rezultate experimentale 10

Concluzii 13
Concluzii generale 13
Dezvoltări viitoare 13
Bibliografie 14

Anexa A. Codul sursa 15

Cerinta Problemei

Am ales problema 5.

Enunț: Aplicație care să spună cat este ceasul pe baza unei poze cu un ceas analogic (acela cu orar și minutar). Programul vostru trebuie să:

- Primească poza cu un ceas
- Să identifice liniile aferente minutarului și orarului
- Calculeze ora pe baza unghiului dintre linii și verticală. (Ex.: la ora 1:10, orarul are un unghi de aproximativ 30 de grade față de verticală, in timp ce minutarul are 60)

Nu vă faceti griji de o precizie extremă aici, sau de poze greu de prelucrat. Țineti ceasul în prim plan, să ocupe aproape toată imaginea. Transformata Hough ar trebui să vă ajute la acest proiect.

Capitolul 1
Pașii necesari rezolvării problemei

In acest capitol se vor discuta pașii realizați pentru a rezolva exercițiul ales.

1.1 Pre-procesarea imaginii

Pentru a putea utiliza cu succes transformata Hough din subcapitolul 1.2 trebuie realizată o procesare a imaginii inițiale. Pre-procesarea este formată din următorii pași:

- **Transformarea imaginii in nuanțe de gri.**

  Am realizat acest pas deoarece în cazul problemei nu avem nevoie de informațiile de culoare din imagine. Nu dorim să aplicăm transformata Hough pe fiecare plan de culoare în parte (RGB).

- **Binarizarea și negarea imaginii.**

  În continuare în cadrul imaginii există informație care influențează rezultatul transformatei Hough (în cadrul unei imagini cu nivele de gri apar detecții de linii false). Pentru a reduce aceste linii false am redus informația din imagine printr-o binarizare. Aceasta a fost realizată utilizând pragul adaptiv Otsu.

  Deoarece imaginile utilizate au fost considerate ca având limbile minutarului și orarului de nunațe închise am realizat în urma binarizării o negare pixel cu pixel. Imaginea rezultată având liniile minutarului și orarului de nivel 255 (alb) pe un fundal de nivel 0 (negru). Operația a fost necesară deoarece transformata Hough consideră nivelul de 0 fundal și valorile pixelilor de 255 ca nivel de interes.

- **Filtrarea conținutului imaginii**

  Pentru a extrage din imagine numai zona de interes, centrală, am creat o mască binară de dimensiunea imaginii inițiale [1] și am realizat operația logică 'AND' pixel cu pixel între imaginea inițială și mască. În Figura 1.1 se poate vizualiza operația descrisă [2].

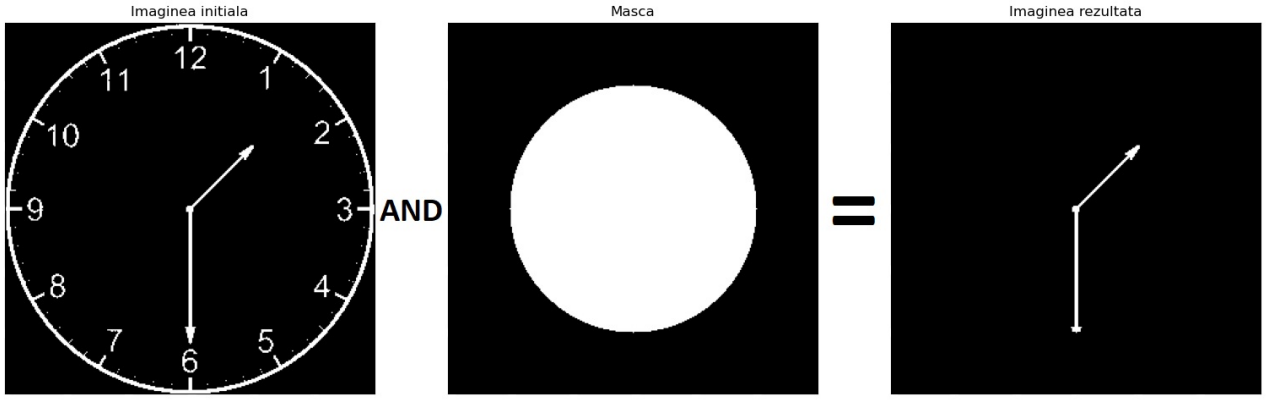

Figura 1.1: Filtrarea conținutului imaginii

- **Erodarea conținutului imaginii**

  Am folosit funcția morfologică de erodare pentru a subția limbile ceasului, proces necesar pentru a reduce numărul de linii multiple pentru fiecare limbă identificate de transformata Hough și eliminarea detaliilor nedorite care nu au fost eliminate de masca (Figura 1.2).

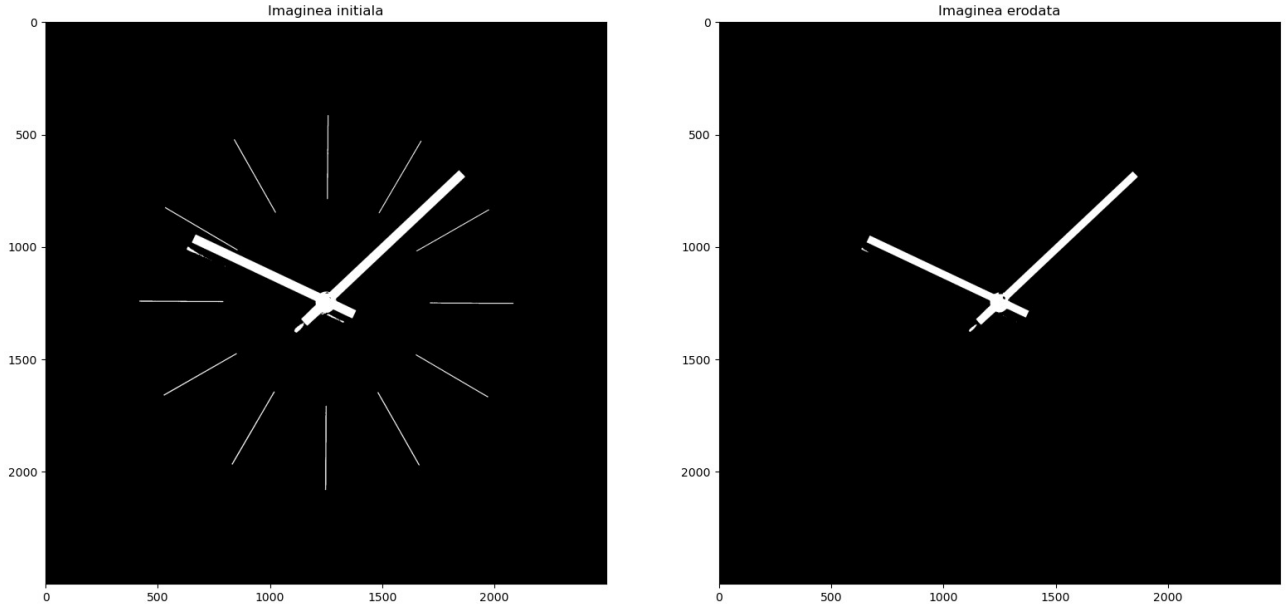

Figura 1.2: Erodarea conținutului imaginii

1.2 Transformata Hough Probabilistică

În cadrul problemei este necesară identificarea orarului și minutarului din imagine pentru a stabili timpul, prin urmare am considerat utilizarea transformării Hough care este folosită la detecția de forme geometrice: linii, cercuri, elipse. Pentru implementarea acesteia am folosit exemplul din OpenCV [3].

Transformata Hough Standard returnează coordonatele liniilor fără a menționa dimensiunea acestora, întrucât în problema noastră lungimea acestora este importantă pentru a clasifica limbile (Subcapitolul 1.3) am utilizat Transformata Hough Probabilistică. Aceasta din urmă returnează un vector de dimensiune 4 cu urmatoarea structură $(x_{start}, y_{start}, x_{end}, y_{end})$, acestea fiind coordonatele liniilor determinate.

În Figurile 1.3 1.4 sunt reprezentate cu rosu liniile identificate de cele doua tipuri de transformate Hough.

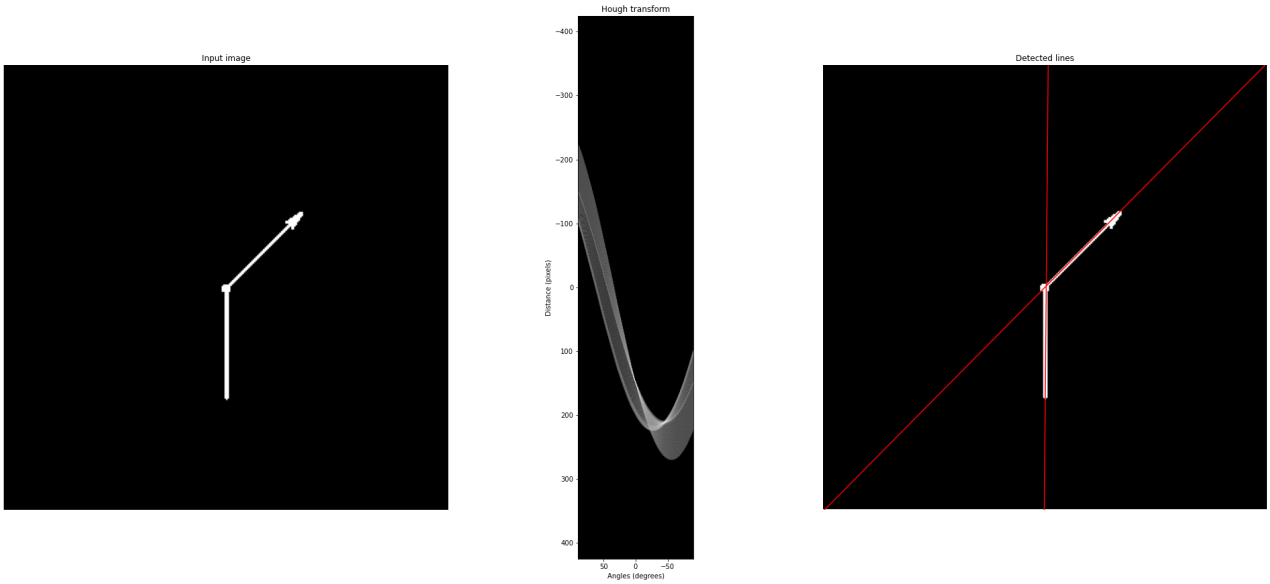

Figura 1.3: Rezultat Transformata Hough Standard

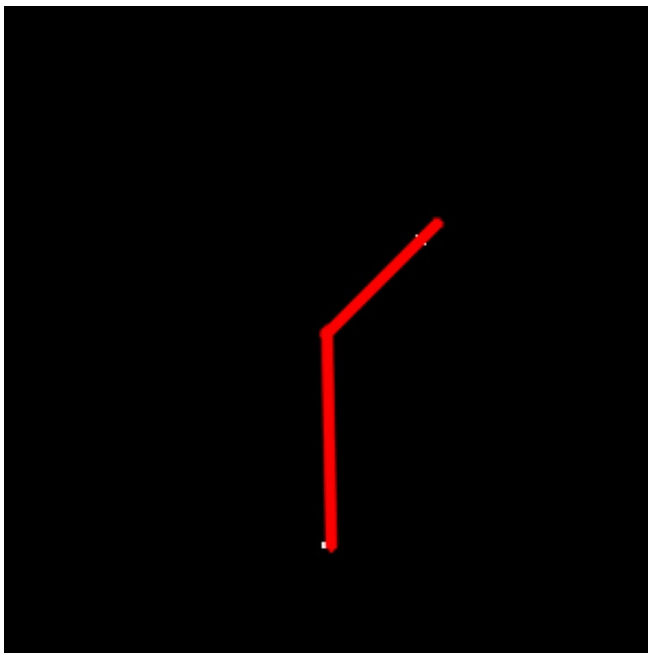

Figura 1.4: Transformata Hough Probabilistică

1.3 Clasificarea liniilor

Utilizând vectorii obținuți în (Subcapitolul 1.2) am calculat lungimile tuturor liniilor identificate de transformata Hough Probabilistică.

Se cunoaște că limba minutarului este de o dimensiune mai mare decât a orarului lucru ce ne permite clasificarea acestora. Dacă transformata Hough Probabilistică identifică mai mult de două linii înseamnă ca gradul de erodare din (Subcapitolul 1.1) nu a fost suficient de mare. Prin urmare, pentru a determina limbile vom considera în funcție de lungimile calculate că lungimea maximă corepunde minutarului și lungimea minimă corespunde orarului. Restul de linii sunt considerate fals identificate și nu se vor utiliza în continuare.

1.4 Calcularea unghiurilor

Pentru a putea calcula unghiurile corespunzătoare liniilor am stabilit o referință după cum se poate observa în Figura 1.5. Pentru fiecare dintre cele două linii se va calcula cu ajutorul teoremei cosinusului (Ecuatia 1.1) unghiul dintre referința stabilită și linia determinată în (Subcapitolul 1.3) după cum se observă în Figura 1.6.

$$a^2 = b^2 + c^2 - 2ab \cdot cos(A) \tag{1.1}$$

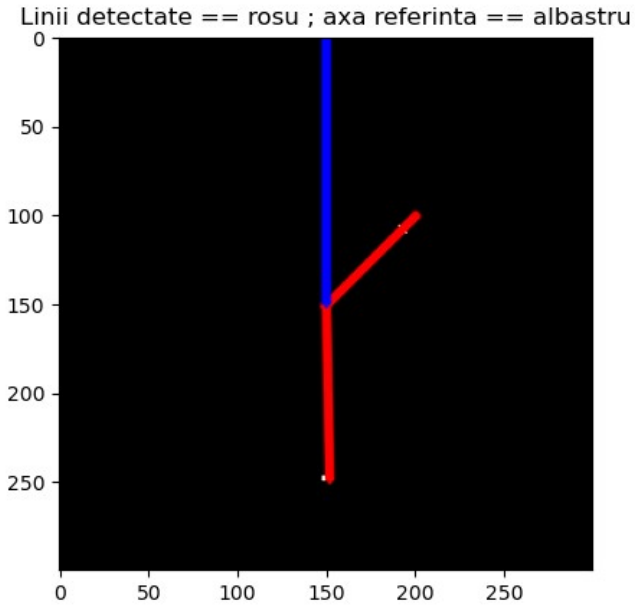

Figura 1.5: Axa referinta

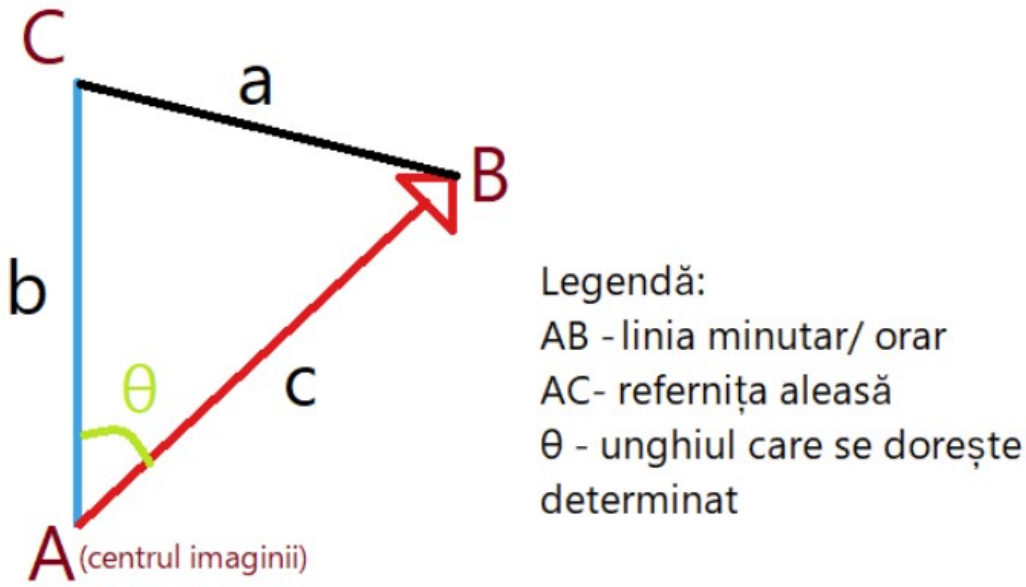

Figura 1.6: Teorema cosinusului

Dacă linia determinată se regăsește în partea stângă a referinței, unghiul θ returnat este eronat după cum se poate observa în Figura 1.7.

Prin urmare, dacă limba se află în partea stângă a referinței valoarea unghiului este de (360 - valoarea calculată).

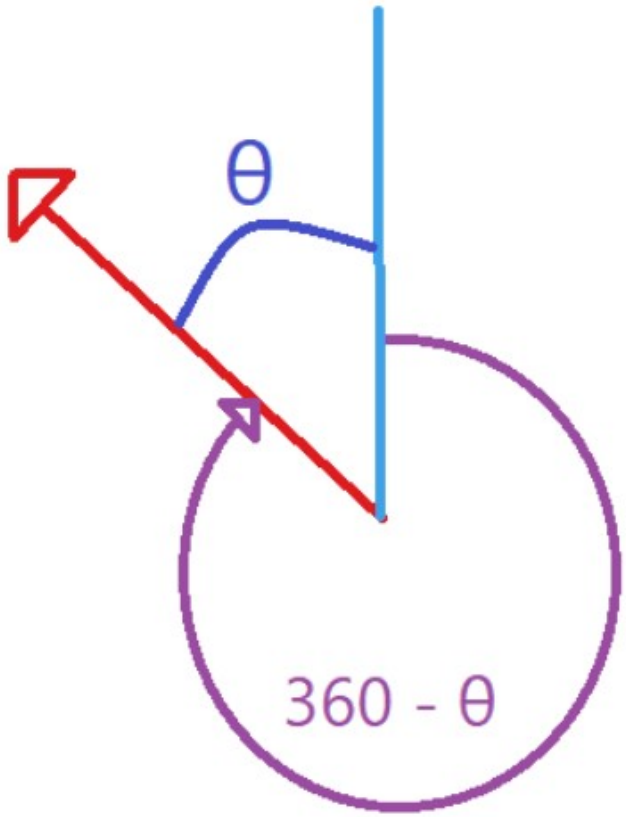

Figura 1.7: Corectie unghi θ

1.5 Corelare grade - timp

Un ceas are 12 ore și un cerc are 360 de grade. Prin urmare, putem corela că o oră este echivalentul a 30 de grade de pe cerc.

În cazul minutelor cele 360 de grade corespund pentru 60 de minute, deci trecerea unui minut poate fi reprezentată prin adăugarea a 6 grade.

Capitolul 2
Rezultate experimentale

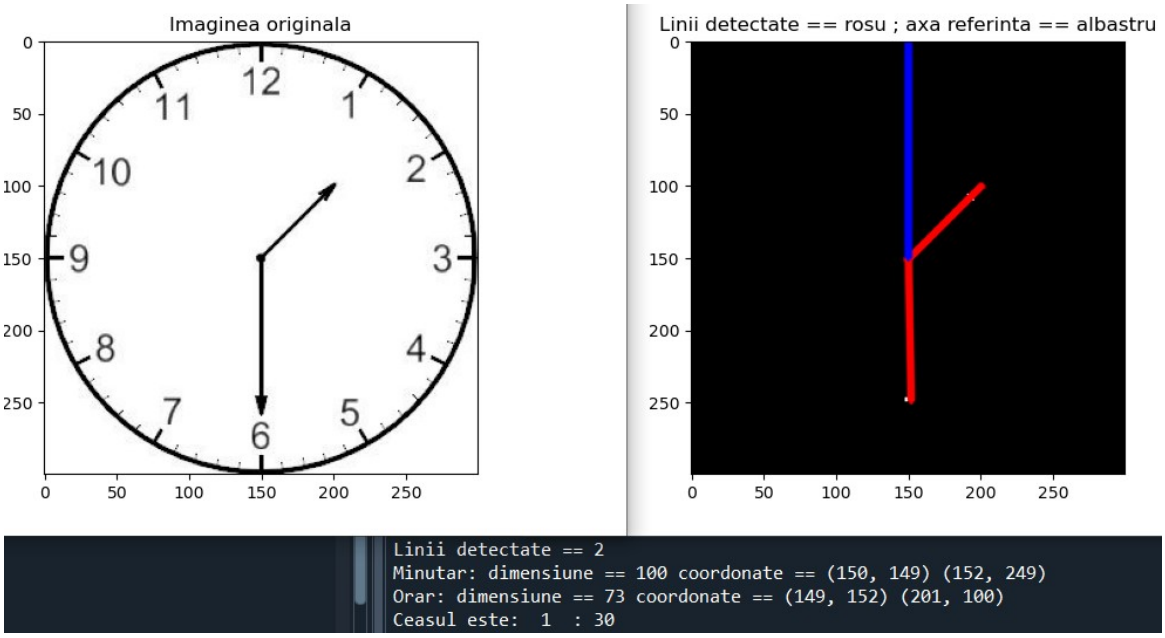

Figura 2.1: Experiment 1

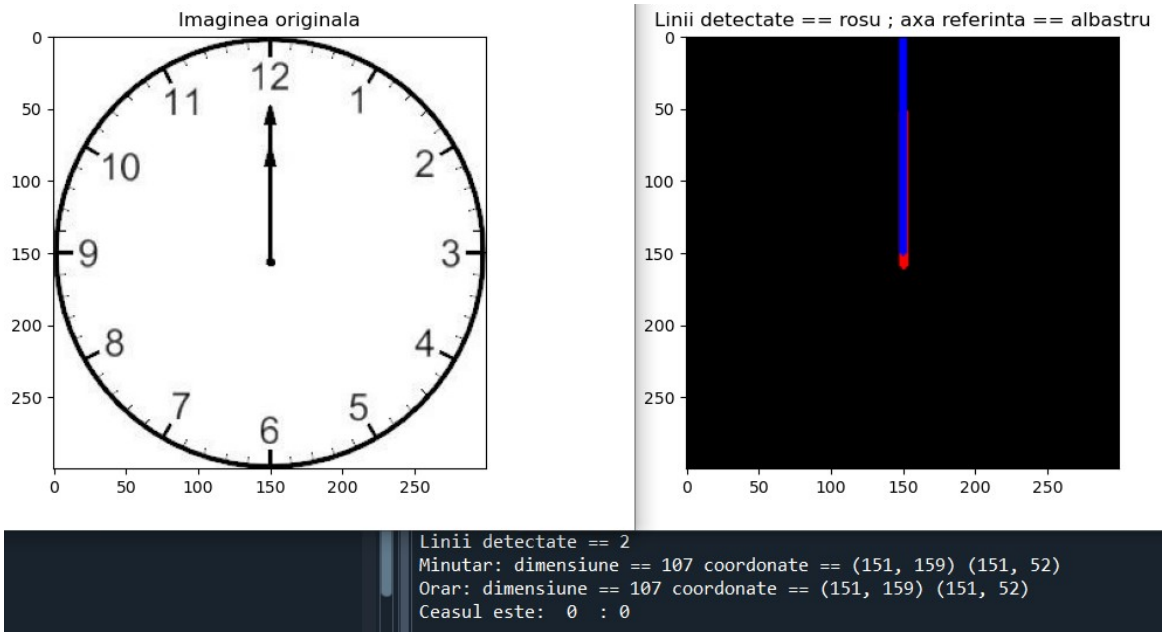

Figura 2.2: Experiment 2

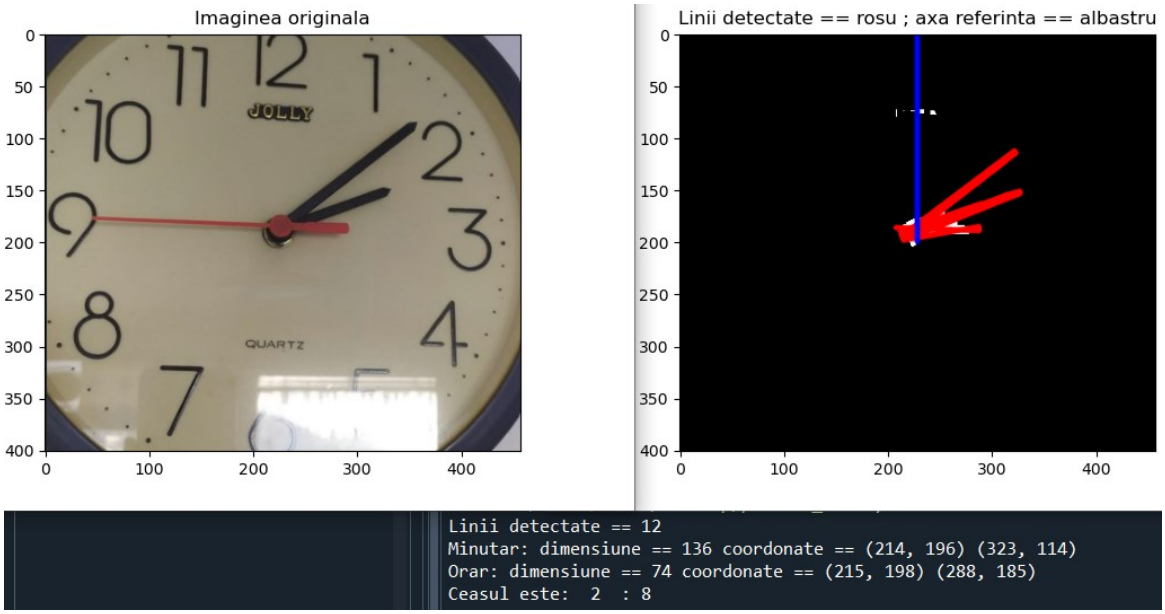

Figura 2.3: Experiment 3

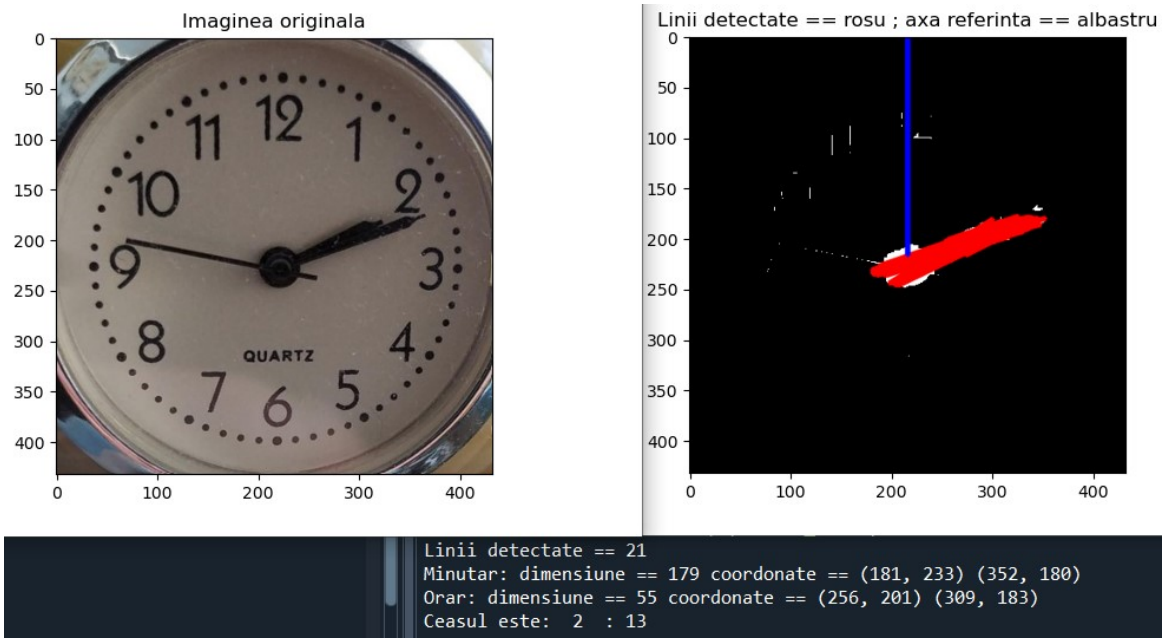

Figura 2.4: Experiment 4

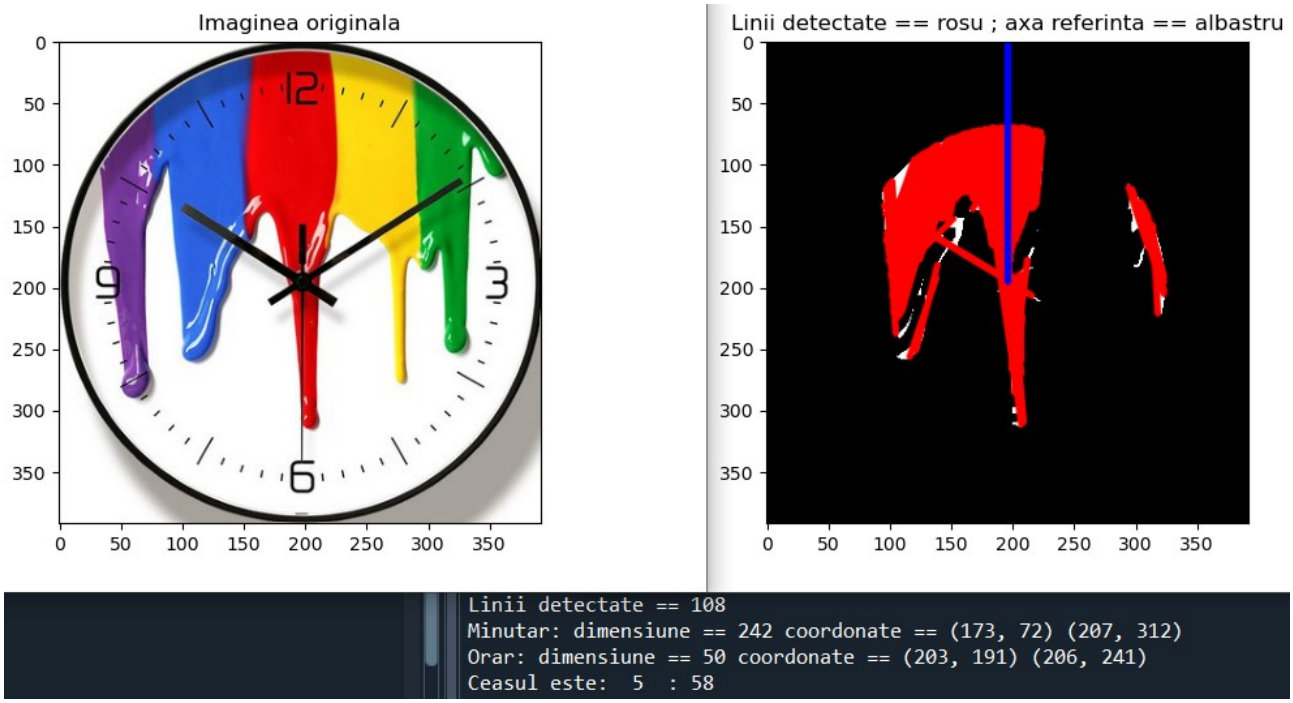

Figura 2.5: Experiment 5

Pentru primele experimente am folosit imagini usor de procesat (Figura 2.1, Figura 2.2). Algoritmul creat identifica corect timpul chiar și pentru cazurile în care limbile minutarului și orarului sunt suprapuse.

Următoarele experimente au fost realizate pe imagini ale unor ceasuri analogice reale (Figura 2.3, Figura 2.4). Pentru acestea a fost folosită în cadrul operației de erodare o fereastră glisantă de dimensiune mai mare (6,6) față de experimentele precedente (2,2) pentru a obține un nivel de erodare mai mare care filtrează cu succes secundarul.

În Figura 2.5 avem un experiment în care algoritmul creat nu reușește să determine corect timpul din cauza fundalului ceasului. În urma conversiei RGB în nivele de gri fundalul colorat al ceasului are un nivel apropiat de limbile acestuia. Iar în urma binarizarii folosind pragul adaptiv Otsu, o parte din acest fundal este clasificat ca obiect de interes.

Pentru a realiza mai multe teste se pot extrage imagini din următoarea sursă [4].

Concluzii

Concluzii generale

După cum se poate observa în cadrul (Capitolului 2) implementarea problemei funcționează foarte exact pentru imagini în care ceasul este obiectul principal, centrat în imagine și fundalul ceasului este mai deschis decât limbile acestuia.

Dezvoltări viitoare

- Crearea unei măști inteligente utilizate în cadrul imaginilor în care ceasul nu este situat central.
- Realizarea unui algoritm de erodare adaptivă în funcție de numărul de linii identificate de transformata Hough.
- Optimizarea parametrilor reglabili din transformata Hough Probabilistică (threshold, minLineLength, maxLineGap) [3].
- Extinderea algoritmului de pre-procesare pentru a se adapta la ceasuri cu limbi albe.

Bibliografie

[1] Masking Function. Lane Detection Tutorial in OpenCV Python using Hough Transform. https://machinelearningknowledge.ai/lane-detection-tutorial-in-opencv-python-using-hough-transform/.

[2] Image Arithmetic and Logical operations in OpenCV with Python. https://datahacker.rs/005-image-arithmetic-and-logical-operations-in-opencv-with-python/.

[3] OpenCV. Hough Line Transform. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[4] Clock time without seconds animation. https://depositphotos.com/94313000/stock-video-clock-time-without-seconds.html.

Anexa A
Codul sursa

```
import numpy as np
from matplotlib import pyplot as plt
from skimage import io, color

from skimage.filters import threshold_otsu

import cv2
import math

def preprocess(img):
    img_pro = np.copy(img)

    # conversie nivele de gri si gama 0-255, uint 8
    try:
        img_pro = color.rgb2gray(img_pro)
        img_pro = np.array(img_pro *255, dtype = 'uint8') #conversie uint8

    except Exception:
        pass

    # binarizare
    thr = threshold_otsu(img_pro)
    img_pro[img_pro >= thr] = 255
    img_pro[img_pro < thr] = 0

    # invert image color
    img_pro = ~img_pro

    # create a mask that selects only the center of the image
    h, w = img_pro.shape

    center = (int(w/2), int(h/2))
    # use the smallest distance between the center and image walls  and devide that by 3
    radius = int (min(center[0], center[1], w-center[0], h-center[1]) / 1.5)

    mask = np.ones((h, w), dtype = "uint8")
    cv2.circle(mask, center, radius, 255, -1)

    mask[mask != 255] = 0

    #bitwise and with mask
    img_pro = cv2.bitwise_and(img_pro, mask)

    #erodare imagine
    kernel = np.ones((2,2), np.uint8)
    #kernel[0, 0] = 0
    #kernel[2, 2] = 0
    #kernel[2, 0] = 0
    #kernel[0, 2] = 0
    img_pro = cv2.erode(img_pro, kernel, iterations=1)

    return img_pro

def hough_transform(img):
    # Copy edges to the images that will display the results in RGB
    cdstP = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)

    #
    linesP = cv2.HoughLinesP(img, 1, np.pi / 180, 50, None, 50, 10)

    if linesP is not None:
        for i in range(0, len(linesP)):
            l = linesP[i][0]
```

```
64              cv2.line(cdstP, (l[0], l[1]), (l[2], l[3]), (255,0,0), 3, cv2.LINE_AA)
65
66      return cdstP, linesP
67
68  def compute_dist(coordonate_linii):
69      lungimi = []
70      if coordonate_linii is not None:
71          for i in range(0, len(coordonate_linii)):
72              l = coordonate_linii[i][0]
73
74              lungime = math.sqrt ((l[2] - l[0])**2 + (l[3] - l[1])**2)
75              lungimi.append(lungime)
76
77      max_value = max(lungimi)
78      index_minutar = lungimi.index(max_value)
79
80      min_value = min(lungimi)
81      index_orar = lungimi.index(min_value)
82
83      return lungimi, index_minutar, index_orar
84
85  def plot_axa_referinta(plot_linii):
86      h, w, c = plot_linii.shape
87      #center = (int(w/2), int(h/2))
88      cv2.line(plot_linii, (int(w/2), int(h/2)), (int(w/2), 0), (0,0,255), 3, cv2.LINE_AA)
89
90      plt.figure(), plt.imshow(plot_linii), plt.title("Linii detectate == rosu ; axa referinta
        == albastru")
91
92  def calcul_unghi(img, coordonate_linie) :
93      #dimensiuni imagine
94      h, w = img.shape
95      #stabilire capat linie
96      d1 = np.zeros([2])
97      #calcul dintre centru si coordonate
98      d1[0] = abs(w//2 - coordonate_linie[0])
99      d1[1] = abs(h//2 - coordonate_linie[1])
100     d11 = d1[0]+d1[1]
101     d2 = np.zeros(2)
102     d2[0] = abs(w//2 - coordonate_linie[2])
103     d2[1] = abs(h//2 - coordonate_linie[3])
104     d22 = d2[0]+d2[1]
105
106     if d11 > d22 :
107       x = coordonate_linie[0]
108       y = coordonate_linie[1]
109     else:
110       x = coordonate_linie[2]
111       y = coordonate_linie[3]
112
113     #calcul valori lungimii laturi triunghi
114
115     a= math.sqrt((w/2-x)**2+(0-y)**2)
116
117     b = h/2
118
119     c = math.sqrt((w/2-x)**2+(h/2-y)**2)
120
121     #determinare unghi cu teorema cosinus
122     # a^2 = b^2+c^2 - 2bc*cosA
123
124     cosA = (b**2 + c**2 - a**2) / (2*b*c)
125
126     #calcul unghi A in radiani
127     A = math.acos(cosA)
128
129     # calcul unghi A in grade
130     unghiA = A*180/math.pi
131
132     # determinare semicerc linie (stanga/dreapta)
133
134     if x < w/2 :
135         unghiA = 360 - unghiA
136
137     return unghiA
138
```

```
139 def calcul_ceas (unghi_ora, unghi_minut):
140
141     # o ora este echivalatenta cu 360 grade / 12 ore = 30 de grade
142     ora = unghi_ora//30
143     # un minut este echivalent cu 360 grade / 60 minute = 6 grade
144     minut = round(unghi_minut/6)
145     return ora, minut
146
147
148 #main
149 if __name__=="__main__":
150
151     # incarca imagine
152     img = io.imread('clock_1.jpg')
153
154     # afisare imagine originala
155     plt.figure(), plt.imshow(img), plt.title("Imaginea originala")
156
157     # preprocesare imagine
158     img = preprocess(img)
159
160     # afisare imagine preprocesata
161     plt.figure(), plt.imshow(img, cmap='gray'), plt.title("Imaginea preprocesata")
162
163     #apelare transformata hough
164     plot_linii, coordonate_linii = hough_transform(img)
165
166     #afisare linii detectate
167     plt.figure(), plt.imshow(plot_linii), plt.title("Linii detectate (rosu)")
168     print("Linii detectate == " + str(coordonate_linii.shape[0]))
169
170
171     #determinare distante si clasificare minutar / orar
172     lungimi, min_i, ora_i = compute_dist(coordonate_linii)
173
174     print("Minutar: dimensiune == " + str(int(lungimi[min_i])) + " coordonate == (" + str(
        coordonate_linii[min_i][0][0]) + ", " + str(coordonate_linii[min_i][0][1]) + ") ("  + str(
        coordonate_linii[min_i][0][2]) + ", "  + str(coordonate_linii[min_i][0][3]) + ") "  )
175     print("Orar: dimensiune == " + str(int(lungimi[ora_i])) + " coordonate == (" + str(
        coordonate_linii[ora_i][0][0]) + ", " + str(coordonate_linii[ora_i][0][1]) + ") ("  + str(
        coordonate_linii[ora_i][0][2]) + ", "  + str(coordonate_linii[ora_i][0][3]) + ") "  )
176
177     #plot axa referinta
178     plot_axa_referinta(plot_linii)
179
180     #determinare unghiuri
181     unghi_minut = calcul_unghi(img, coordonate_linii[min_i][0])
182     unghi_ora = calcul_unghi(img, coordonate_linii[ora_i][0])
183
184     #determinare ceas
185     ora, minut = calcul_ceas(unghi_ora, unghi_minut)
186
187     print("Ceasul este: ", int(ora)," :", int(minut))
```