Laborator 1 SRF: Etapa de preprocesare.
Introducere în Processing IDE.

Scopul lucrării

Acest laborator are ca obiective înțelegerea de către studenți a proceselor care pot avea loc în etapa de preprocesare, precum și familiarizarea studentului cu mediul de lucru (Processing IDE). La finalul acestuia, studenții trebuie să poată folosi tipurile de variabile ale acestui mediu de dezvoltare, să folosească obiecte din clasa PImage, precum și să gestioneze evenimente apărute ca urmare a interacțiunii cu utilizatorul. Programul de test pe care se face toată această recapitulare a noțiunilor de bază din programarea grafică este unul simplu, de calcul al histogramei unei imagini pe care utilizatorul o poate modifica dinamic în timpul rulării, și se bazează pe noțiuni învățate anterior la discipline precum "Prelucrarea imaginilor".

Tot ca urmare a realizării acestui laborator studenții vor înțelege importanța etapei de pre-procesare în fluxul de pași care duce la recunoașterea unui obiect de către calculator și vor fi capabili să îmbunătățească contrastul unei imagini folosindu-se de algoritmul de egalizare a histogramei.

1. Breviar teoretic:

Processing este un mediu de dezvoltare (IDE) folosit la programarea aplicațiilor grafice. Acesta folosește o sintaxă ușoară (Java) pentru a crea animații, desene și aplicații interactive. Inițial, Processing a fost prototipul pentru mediul de dezvoltare al Arduino.

Processing este potrivit în special pentru creatia artistica grafică și procesarea grafică de date, dar simplitatea sa de utilizare îl face și un bun suport pentru învățarea logicii programării. Unul din avantajele sale este că poate fi folosit pe mai multe platforme: Macintosh, Windows, Linux, BSD și Android.

1.1. Tipuri de date în Processing:

A. Tipuri de date primitive:

A1. boolean

Memorează o valoare de adevăr într-o variabilă.

Ex.:

```
boolean a=false;
if(!a) rect(30,20,50,50);
```

A2. byte

Memorează un număr întreg în intervalul [-128, 127]

A3. char

Memorează un caracter alfanumeric. Ocupă 2 octeți de memorie RAM.

Ex:

```
char m='A';
```

A4. color

Memorează o variabilă de tip culoare.

Exemple:

```
color c1;
c1=color(200,200,200,255)
c1=#C8C8C8FF // (doar pentru culori opace)
c1=0xFFC8C8C8 // (prima data este trecut canalul alpha)
c1=get(i,j)  // (citeste valoarea culorii pixelului aflat la coordonatele i si j
```

Funcții care se bazează pe culori în Processing IDE:

```
background(r,g,b,alpha) // setează culoarea fundalului
stroke(r,g,b,alpha) // fixează culoarea contururilor
noStroke()       // figurile vor fi desenate fără contur
fill(r,g,b,alpha)           // fixează culoarea de umplere a formelor
noFill()

float r=red(c1);           // citirea nivelului de roșu dintr-o culoare c1
float g=green(c1);  // citirea nivelului de verde dintr-o culoare c1
float b=blue(c1);  // citirea nivelului de albastru dintr-o culoare c1
```

A5. Alte tipuri de date primitive:

- float (32bit)
- double (64bit)
- int (32bit)
- long (64bit)

B. Tipuri de date compuse:

B1. Array:

Exemple:

```
int[] numere=new int[3];
numere[0]=90;
numere[1]=91;
numere[2]=92;
int[] numere={90,91,92};
```

B2. String (șir de caractere)

Exemplu:

```
String str1="Abcd";
char[] str1={'A','b','c','d'};
if(str1.equals("Abcd")){
  println("Da, cele doua siruri sunt egale");
}
```

B3. Liste: ArrayList, IntList, FloatList, StringList

Exemplu:

IntList x;
x.sort(), x.reverse(), x.shuffle(), x.min(), x.max(),
x.get(nth), x.set(nth), x.append(element), x.add(nth,element);
x.remove(nth)

B4. PImage

Este o clasă folosită la reprezentarea imaginilor ca matrici de pixeli.
Un obiect din clasa PImage are următoarele date-membru:

- pixels[]: ne oferă acces la culorile tuturor pixelilor din obiectul imagine
- width : memorează lățimea unei imagini
- height: memorează înălțimea unei imagini

Exemplu:

```
PImage img;
c=img.pixels[j*width+i]
```

Un obiect din clasa PImage are urătoarele metode:

- resize: redimensineaza imaginea
- get: citeste culoarea pixelului aflat la pozitia i,j
- set: fixeaza culoarea pixelului aflat la pozitia i,j
- mask: foloseste o alta imagine ca masca
- filter: transforma culorile pixelilor din imagine in niveluri de gri
- copy: copiaza imaginea
- save: salveaza imaginea pe disc

- img=loadImage("fisier.jpg");

1.2.Interactiunea cu utilizatorul

Processing IDE oferă multiple posibilități de interacțiune cu utilizatorul. Poate citi atât date de la tastatură precum și de la mouse.

A. Date citite de la mouse:

Pentru a citi poziția mouse-ului se pot citi folosi variabilele mouseX si mouseY, care memorează coordonatele cursorului relativ la originea ecranului

Exemplu:

```
void draw(){
        frameRate(12);
        println(mouseX+":"+mouseY);
}
```

Alte variabile utile sunt pmouseX, pmouseY, care reprezintă coordonatele cursorului la momentul precedent (cu un cadru înainte).

Variabila booleană mousePressed indică dacă mouse-ul este apăsat la momentul verificării variabilei. Variabila mouseButton ne indică ce buton a fost apăsat ultima dată. Poate avea valorile LEFT, CENTER sau RIGHT.

B. Date primite de la tastatură

Variabila keyPressed, de tip boolean (true, false), indică dacă o tastă a fost apăsată sau nu. Variabila key memoreaza ultima tastă apăsată, fie sub formă de caracter (Ex. 'a'), fie sub forma unui cod ASCII. Poate memora si taste speciale (ALT, CONTROL, SHIFT, UP, DOWN, LEFT, RIGHT).

Evenimente sunt categorii speciale de funcții, care sunt apelate atunci când are loc o acțiune. Se execută o singură dată după apariția evenimentului, prin generarea unei așa-zise întreruperi politicoase. Exemple de evenimente gestionate în Processing:

```
void mousePressed() {
}

void mouseReleased() {
}

void mouseMoved(){
}

void mouseDragged(){
}

void keyPressed(){
}

void keyReleased(){
}
```

1.3**. Etapa de pre-procesare**:

Un sistem complet de recunoaștere a formelor urmează o serie de etape, plecând de la etapa de achiziție a unei imagini până la cea de identificare a obiectelor din imaginea respectivă. Una din etapele importante ale acestei serii este etapa de pre-procesare, care după cum îi spune numele, urmează imediat după etapa de achiziție și înaintea etapei de procesare propriu-zisă. Exemple de algoritmi de pre-procesare care pot îmbunătăți performanțele unui sistem de recunoaștere a formelor sunt: egalizarea histogramei, detecția conturului, eliminarea zgomotului, etc.

Egalizarea histogramei

Această metodă crește de obicei contrastul global al multor imagini, mai ales când imaginea este reprezentată de o gamă restrânsă de valori de intensitate. Prin această ajustare, intensitățile pot fi distribuite mai bine pe histogramă utilizând în mod egal întreaga gamă de intensități. Acest lucru

permite ca zonele cu contrast local mai scăzut să obțină un contrast mai mare. Egalizarea histogramei realizează acest lucru prin răspândirea eficientă a valorilor intensității foarte populate care sunt utilizate pentru a degrada contrastul imaginii.

Metoda este utilă în imaginile cu fundal și prim-plan care sunt ambele luminoase sau ambele întunecate. În special, metoda poate duce la vederi mai bune ale structurii osoase în imagini cu raze X și la detalii mai bune în fotografiile care sunt fie supra sau sub-expuse. Un avantaj cheie al metodei este că este o tehnică destul de simplă, adaptabilă la imaginea de intrare și un operator inversabil. Deci, în teorie, dacă funcția de egalizare a histogramei este cunoscută, atunci histograma originală poate fi recuperată. Calculul nu este intensiv de calcul. Un dezavantaj al metodei este că este nediscriminatorie. Poate crește contrastul zgomotului de fundal, scăzând în același timp semnalul utilizabil.

În imagistica științifică, unde corelația spațială este mai importantă decât intensitatea semnalului (cum ar fi separarea fragmentelor de ADN cu lungime cuantificată), raportul mic semnal/zgomot împiedică de obicei detecția vizuală.

Egalizarea histogramei produce adesea efecte nerealiste în fotografii; cu toate acestea, este foarte util pentru imagini științifice precum imaginile termice, prin satelit sau cu raze X, adesea aceeași clasă de imagini la care s-ar aplica culori false. De asemenea, egalizarea histogramei poate produce efecte nedorite (cum ar fi gradientul de imagine vizibil) atunci când este aplicată imaginilor cu adâncime de culoare scăzută. De exemplu, dacă este aplicat la o imagine de 8 biți afișată cu o paletă de gri de 8 biți, aceasta va reduce și mai mult profunzimea culorii (numărul de nuanțe unice de gri) a imaginii.
Egalizarea histogramei va funcționa cel mai bine atunci când este aplicată imaginilor cu o adâncime de culoare mult mai mare decât dimensiunea paletei, cum ar fi datele continue sau imaginile în scară de gri pe 16 biți.

Algoritm:
Pas1. Calculare tablou histograma prin parcurgerea tuturor pixelilor din imagine
Pas2. Calculare tablou acumulator, in care fiecare element i este egal cu suma tuturor elementelor de la 0 la i din tabloul histograma
Pas3. Schimbarea valorilor pixelilor folosindu-ne de tabloul acumulator

2. Desfășurarea lucrării:

În acest laborator vom folosi unul din proiectele oferite ca și exemplu în mediul Processing, exemplu pe care îl vom deschide în felul următor:

1) Se rulează aplicatia Processing.
2) Din meniul programului vom selecta **File→Exemples...→Topics→Image Processing→Histogram**.

Histograma unei imagini arată numărul total de pixeli alocat fiecărui ton de gri care apare în imagine. Aceasta ajută la analizarea și la corectarea contrastului unei imagini. Dacă se dorește a se calcula histograma atunci imaginea trebuie să fie în nuanțe de gri (grayscale).

Codul sursă al programului-exemplu este următorul:

Histogram

```
1
2  size(640, 360);
3
4  PImage img = loadImage("frontier.jpg");
5  image(img, 0, 0);
6  int[] hist = new int[256];
7
8  // Calcul histograma
9  for (int i = 0; i < img.width; i++) {
10   for (int j = 0; j < img.height; j++) {
11     int bright = int(brightness(get(i, j)));
12     hist[bright]++;
13   }
14 }
15
16 // Se gaseste valoarea maxima
17 int histMax = max(hist);
18
19 stroke(255);
20
21 for (int i = 0; i < img.width; i += 2) {
22   int which = int(map(i, 0, img.width, 0, 255));
23   // y reprezinta o locatie intre inceputul si sfarsitul imaginii
24   int y = int(map(hist[which], 0, histMax, img.height, 0));
25   line(i, img.height, i, y);
26 }
27
```

După rularea programului se va afișa următoarea imagine:

Se observă cum peste imagine a fost suprapus graficul histogramei folosind linii albe. Fiecare poziție pe orizontală a indică o valoare a luminanței, iar lungimea liniei reprezintă numărul de pixeli care au o anumită valoare a luminanței (0, 1, 2 etc., până la valoarea maximă de 255).

Cerințe :

1) **Analizați programul de mai jos și explicați funcționarea lui.**
2) **Desenați un cerc alb în jurul poziției mouse-ului.**
3) **Schimbați culoarea cercului la fiecare apăsare a tastei "c" din alb în negru și înapoi în alb. La apăsarea mouse-ului schimbați culoarea pixelilor din imagine deasupra cărora se află cercul în momentul apăsării: dacă cercul este alb, faceți și pixelii respectivi albi. Dacă cercul este negru, faceți și pixelii respectivi negri.**
4) **Reinițializați imaginea la apăsarea tastei r.**
5) **Colorați toți pixelii de pe linia pe care se află mouse-ul atunci când se apasă pe acesta. Inchideți luminozitatea pixelilor dacă cercul mouse-ului (brush-ul) este negru, și deschideți luminozitatea pixelilor dacă cercul este alb.**
6) **Implementați algoritmul de egalizare a histogramei explicat în acest laborator (pagina 4).**

Rezolvare:

1) Pentru a modifica imaginea în timp real Processing lucrează dinamic grație funcției draw() care este executată în buclă în timpul funcționării programului. Vom avea nevoie și de metoda "setup" care inițializează imaginea și mărimea ferestrei de afișare o singură dată. După cum spuneam, după executarea funcției setup(), metoda "draw" funcționează ca o buclă continuă, deci analizează imaginea la fiecare pas.

Setup:

```
1
2  PImage img;
3
4  void setup() {
5    size(640, 360);
6
7    img = loadImage("frontier.jpg");
8    image(img, 0, 0);
9  }
```

Draw:

```
10
11 void draw() {
12   int[] hist = new int[256];
13
14   for (int i = 0; i < img.width; i++) {
15     for (int j = 0; j < img.height; j++) {
16       int bright = int(brightness(get(i, j)));
17       hist[bright]++;
18     }
19   }
20
21   int histMax = max(hist);
22
23   stroke(255);
24
25   for (int i = 0; i < img.width; i += 2) {
26     int which = int(map(i, 0, img.width, 0, 255));
27     int y = int(map(hist[which], 0, histMax, img.height, 0));
28     line(i, img.height, i, y);
29   }
30 }
31
```

2) În continuare vom desena un cerc lângă poziția cursorului, adăugând metoda “circle” în draw.

```
12 void draw() {
13
14   circle(mouseX, mouseY, 30); // circle(x, y, diametru);
15
```

Se poate observa că cercul rămâne desenat chiar și după ce se schimbă poziția cursorului. Acest lucru se produce pentru că imaginea este inițializată în setup(). Pentru a rezolva problema este nevoie să mutăm imaginea în draw().

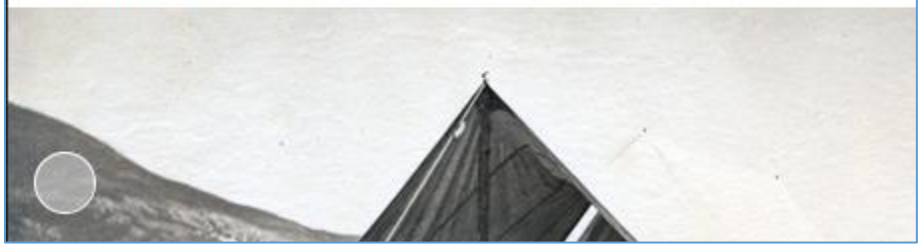
```
5  void setup() {
6
7    size(640, 360);
8    img = loadImage("frontier.jpg");
9
10 }
11
12 void draw() {
13
14   image(img, 0, 0);
```

Histogram

Pentru a colora o figură geometrică (cercul) se va utiliza metoda “fill” înainte circle.

```
14
15   fill(255, 255, 255, 100); // fill(red, green, blue, alpha);
16   circle(mouseX, mouseY, 30); // circle(x, y, diametru)
```

Histogram



3) La fiecare apăsare de tastă (c) se va schimba nuanța cercului din alb (deschis) în negru (închis) și la apăsarea cursorului se vor modifica pixelii în funcție de nuanța cercului.
Se vor adăuga următoarele:

```
2  PImage img;
3  int nuanta;
4
5  void setup() {
6
7    size(640, 360);
8    nuanta = 255;
9
10 }
```

```
12 void draw() {
13
14   image(img, 0, 0);
15
16   fill(nuanta, nuanta, nuanta, 100); // fill(red, green, blue, alpha);
17   circle(mouseX, mouseY, 30); // circle(x, y, diametru)
18
19   if(mousePressed) {
20     color culoare = color(nuanta); //color(gray);
21     img.set(mouseX, mouseY, culoare);
22   }
23
```

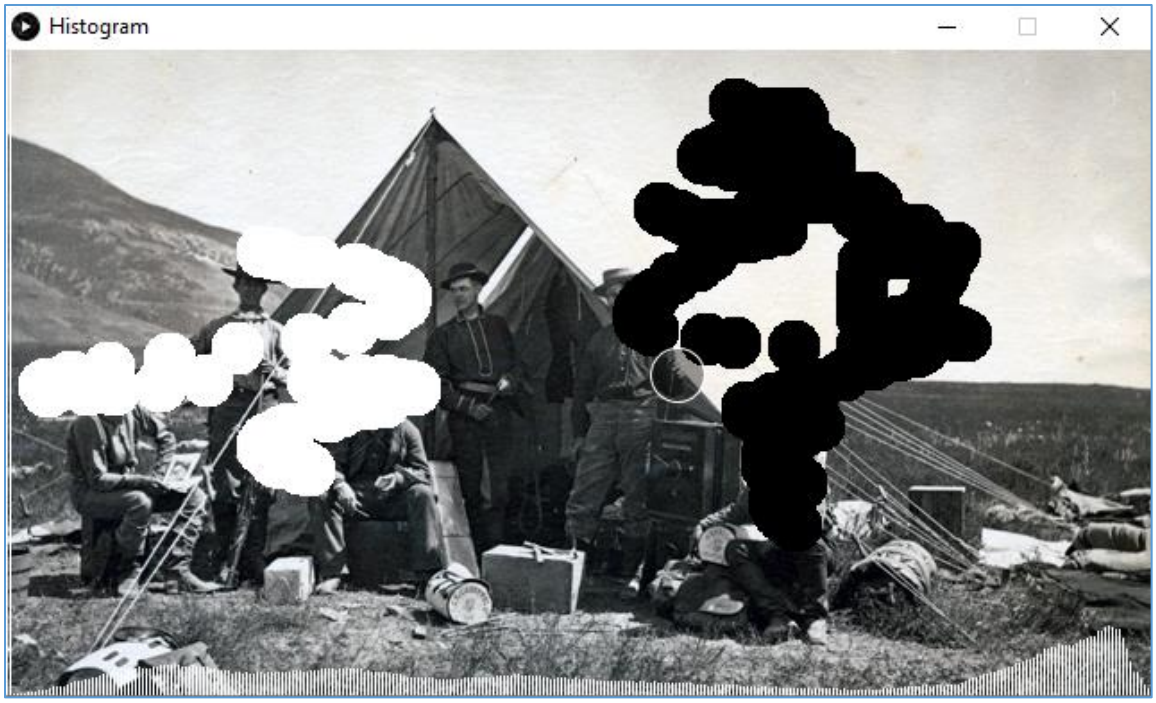
```
39
40 void keyPressed() {
41   if(key == 'c') {
42     nuanta = 255 - nuanta;
43   }
44 }
```

Pentru a schimba valoarea pixelilor in funcție de mărimea cercului se va adăuga:

```
3 int nuanta;
4 int raza = 15;
```

```
17   fill(nuanta, nuanta, nuanta, 100); // fill(red, green, blue, alpha);
18   circle(mouseX, mouseY, 2*raza); // circle(x, y, diametru)
19
20   if(mousePressed) {
21     color culoare = color(nuanta); //color(gray);
22     for (int i = 0; i < img.width; i++) {
23       for (int j = 0; j < img.height; j++) {
24         double dist = sqrt((mouseX - i) * (mouseX - i) + (mouseY - j) * (mouseY - j));
25         if(dist < raza) {
26           img.set(i, j, culoare);
27         }
28       }
29     }
30   }
```

Se poate observa ca simultan, s-a schimbat și histograma.



4) La apăsarea tastei “r” se va reinițializa poza (refresh).

```
52 void keyPressed() {
53   if(key == 'c') {
54     nuanta = 255 - nuanta;
55   }
56   if(key == 'r') {
57     img = loadImage("frontier.jpg");
58   }
59 }
```

5)

```
20   if(mousePressed) {
21     for(int i = 0; i < img.width; i++) {
22       for(int j = mouseY - raza; j < mouseY + raza; j++) {
23         color culoare = img.get(i,j);
24         int k=1;
25         if(nuanta == 0) {
26           k=-1;
27         }
28         culoare = color(red(culoare) + k, green(culoare) + k, blue(culoare)+k);
29         img.set(i, j, culoare);
30       }
31     }
32     //color culoare = color(nuanta); //color(gray);
33     //for (int i = 0; i < img.width; i++) {
34     //  for (int j = 0; j < img.height; j++) {
35     //    double dist = sqrt((mouseX - i) * (mouseX - i) + (mouseY - j) * (mouseY - j));
36     //    if(dist < raza) {
37     //      img.set(i, j, culoare);
38     //    }
39     //  }
40     //}
41   }
```

Anexe. Program complet:

PImage img;
int nuanta;
int raza = 15;

```
void setup() {

  size(640, 360);
  nuanta = 255;
  img = loadImage("frontier.jpg");
}

void draw() {

  image(img, 0, 0);

  fill(nuanta, nuanta, nuanta, 100); // fill(red, green, blue, alpha);
  circle(mouseX, mouseY, 2*raza); // circle(x, y, diametru)

  if(mousePressed) {
    for(int i = 0; i < img.width; i++) {
      for(int j = mouseY - raza; j < mouseY + raza; j++) {
        color culoare = img.get(i,j);
        int k=1;
        if(nuanta == 0) {
          k=-1;
        }
        culoare = color(red(culoare) + k, green(culoare) + k, blue(culoare)+k);
        img.set(i, j, culoare);
      }
    }
  }

  int[] hist = new int[256];

  for (int i = 0; i < img.width; i++) {
    for (int j = 0; j < img.height; j++) {
      int bright = int(brightness(get(i, j)));
      hist[bright]++;
    }
  }

  int histMax = max(hist);

  stroke(255);

  for (int i = 0; i < img.width; i += 2) {
    int which = int(map(i, 0, img.width, 0, 255));
    int y = int(map(hist[which], 0, histMax, img.height, 0));
    line(i, img.height, i, y);
```

```
  }
}

void keyPressed() {
  if(key == 'c') {
    nuanta = 255 - nuanta;
  }
  if(key == 'r') {
    img = loadImage("frontier.jpg");
  }
}
```