Computer Vision

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

Obiective

- Procesarea de imagini cu clase
- Realizarea unei interfete in QT cu:
  - Butoane pentru incarcare, salvare si procesare imagine
  - Posibilitatea alegerii de catre utilizator a unei culori
  - Slider

Gasim toti pixelii cu o anumita culoare

- Realizam un algoritm de identificare a pixelilor dintr-o imagine care au o culoare data sau una apropiata ei
- INPUT: o imagine, o culoare si un prag de toleranta
- OUTPUT: imaginea in care pixelii asemanatori culorii date sub toleranta stabilita sunt albi
- Restul sunt de culoare neagra.

Identificarea pixelilor

- Imaginea de intrare: `image`
- Imaginea rezultata: `rez`
- Tipul imaginii de iesire: CV_8U

```
// Proceseaza imaginea. Intoarce o imagine alb-negru cu un singur canal.
Mat process(Mat &image)
{
    // creeaza o imagine cu aceeasi marime ca cea de input, dar un singur canal
    rez.create(image.rows,image.cols,CV_8U);

    // folosim iteratori
    Mat_<Vec3b>::const_iterator it = image.begin<Vec3b>();
    Mat_<Vec3b>::const_iterator itend = image.end<Vec3b>();
    Mat_<uchar>::iterator itout = rez.begin<uchar>();

    // pentru fiecare pixel
    for ( ; it!= itend; ++it, ++itout)
    {
        //calculam distanta de la culoarea pixelului la culoarea tinta
        if (getDistance(*it) < minDist)
            *itout= 255;//alb
        else
            *itout= 0;//negru
    }

    return rez;
}
```

Distanta intre doua culori

```
// Calculam distanta de la culoarea tinta
int getDistance(Vec3b& color)
{
    return abs(color[0]-target[0])+
                        abs(color[1]-target[1])+
                        abs(color[2]-target[2]);
}
```

- Argumentul primit consta intr-un vector de 3 valori trimis de la pixelul curent din metoda `process`.
- `target` contine valorile culorii tinta.
- Distanta utilizata este cea cu numele de Manhattan.
- Se poate utiliza cea Euclidiana.

Clasa ColorDetector

- Elementele private din cadrul clasei.
- Metoda `getDistance` este apelata numai in cadrul acestei clase, prin urmare se poate considera `private`.

```
class ColorDetector
{
    private:
        // distanta prag
        //sub aceasta distanta, se considera ca cele doua culori coincid
        int minDist;

        // culoarea de detectat
        Vec3b target;

        // imaginea alb-negru rezultata
        Mat rez;

        // Calculam distanta de la culoarea tinta
        int getDistance(Vec3b& color)
        {
            return abs(color[0]-target[0])+
                               abs(color[1]-target[1])+
                               abs(color[2]-target[2]);
        }
```

Constructorul clasei

```
public:

    //constructor
    ColorDetector()
    {//setam valori pentru parametrii "prag" si "culoare tinta" - valorile pot fi si modificate
        minDist = 100;
        target[0]= target[1]= target[2]= 0;
    }
```

- In constructor punem niste valori standard pentru distanta minima si culoare.
- Acestia pot fi modificati prin metodele de tip set si get pe care le vom defini in continuare.

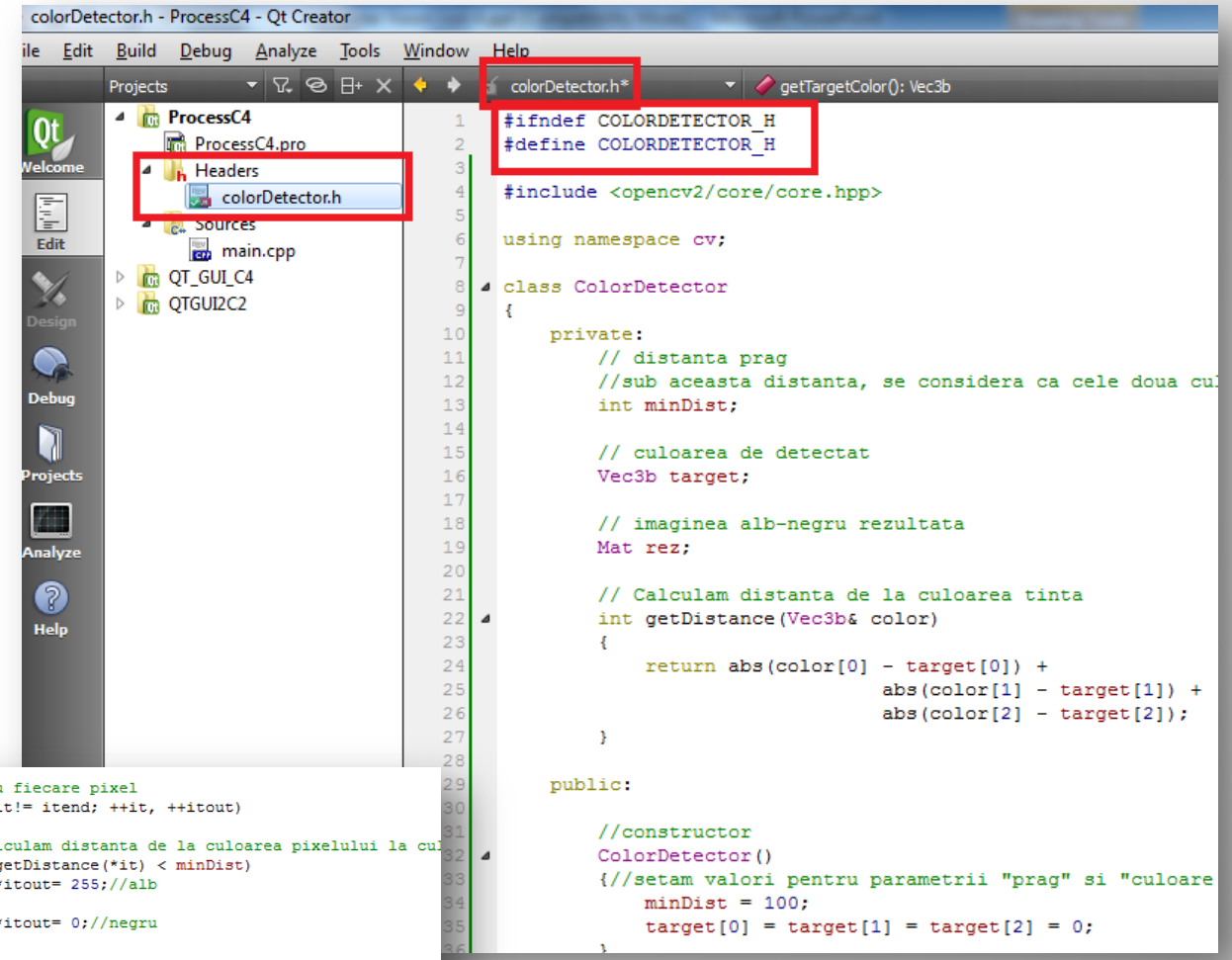
set & get

- Toate acestea sunt de tip `public`.
- Stabilim valoarea prag a distantei si intoarcem valoarea sa.
- 2 metode de a stabili culoarea tinta
  - se alege cea convenabila
- Metoda pentru a intoarce culoarea tinta

```
// Metode set si get

// Setarea pragului distanta.
// Acesta trebuie sa fie pozitiv
// Altfel, il facem 0.
void setColorDistanceThreshold(int distance)
{
    if (distance < 0)
        distance = 0;
    minDist = distance;
}

// Intoarce valoarea pragului distanta
int getColorDistanceThreshold()
{
    return minDist;
}

// Seteaza culoarea de detectat
void setTargetColor(uchar red, uchar green, uchar blue)
{
    target[0] = blue;
    target[1] = green;
    target[2] = red;
}

// Seteaza culoarea de detectat
void setTargetColor(Vec3b color)
{
      target = color;
}

// Gets the color to be detected
Vec3b getTargetColor()
{
      return target;
}
```

Clasa ColorDetector

- Clasa se poate defini in acelasi fisier cu metoda main sau in fisier separat header (*.h)
- Incheiem fisierul header cu `#endif`.

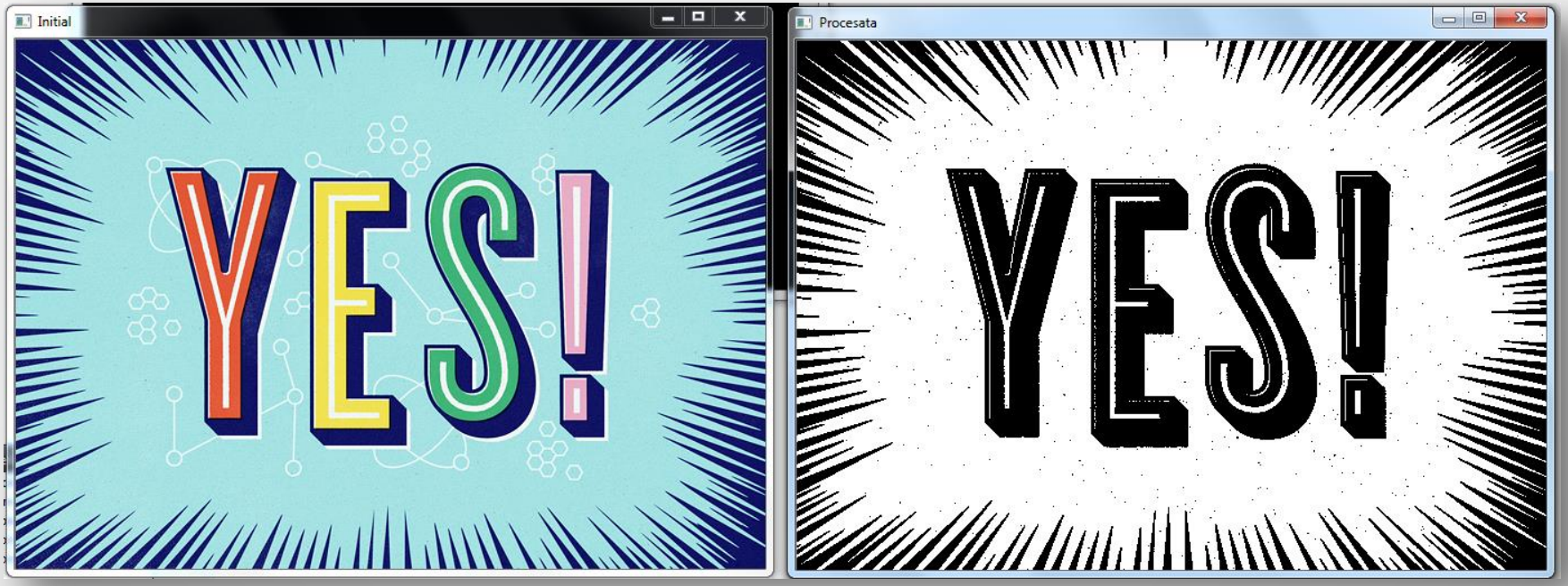


main.cpp

- Includem clasa ColorDetector
- Cream o instanta a clasei
- Citim o poza
- Stabilim culoarea cautata
- Procesam imaginea
- Afisam imaginile
  - Initiala
  - Procesata

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

#include "colorDetector.h"

using namespace cv;

int main()
{
    // Cream obiectul care proceseaza imaginea
    ColorDetector cdetect;

    // Citim imaginea
    Mat poza= imread("D:/poza.jpg");
    if (!poza.data)
            return 0;

    // stabilim culoarea tinta
    cdetect.setTargetColor(170,255,255); // albastru deschis
    Mat rez = cdetect.process(poza);

    namedWindow("Initial");
    imshow("Initial",poza);

    namedWindow("Procesata");
    imshow("Procesata",rez);

    waitKey();

    return 0;
}
```

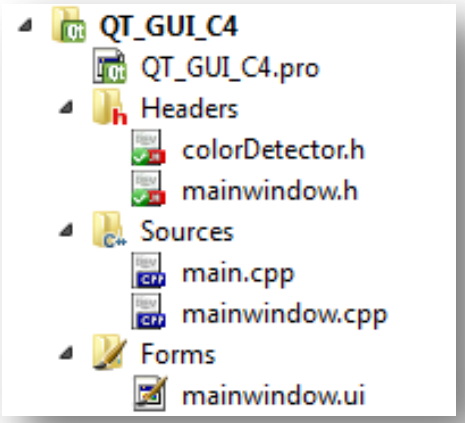
Rezultatul



A se revedea prezentarea de la cursul 2

Acelasi lucru cu GUI

- O interfata care sa ii permita utilizatorului sa dea valori parametrilor de intrare.
- Pastram clasa definita anterior la fel.
- Cream in QT un proiect de tip Widgets Application.
- Vom avea o structura a proiectului precum cea de mai jos
  - Am adaugat si colorDetector.h

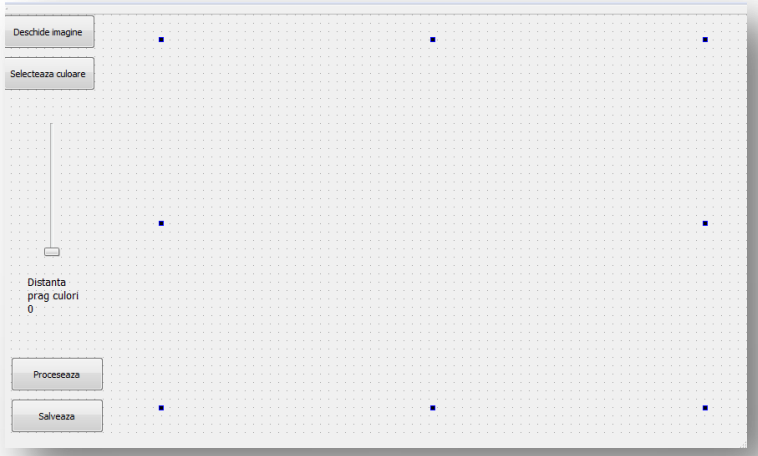


Stabilirea interfetei grafice

- Adaugam in mainwindow.ui componentele urmatoare:

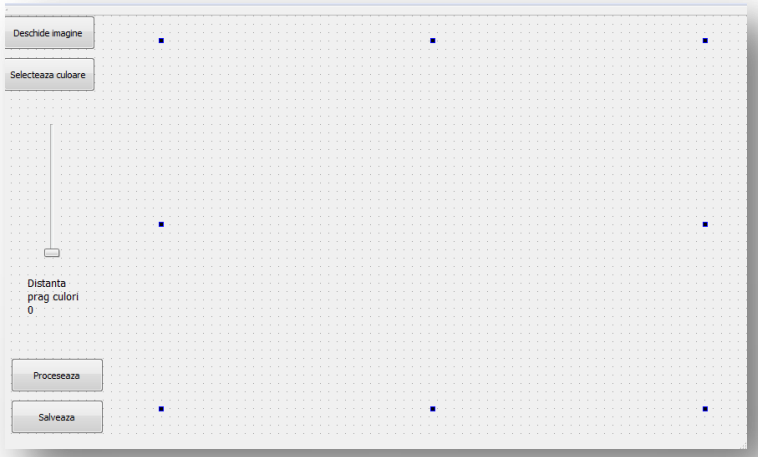
Stabilirea interfetei grafice

- Primul buton urmeaza sa fie folosit pentru a incarca o imagine de pe calculator
- Al doilea buton va deschide o fereastra care sa ii permita utilizatorului sa aleaga o culoare
- Slider-ul ofera posibilitatea stabilrii pragului distantei
- Label-ul va contine informatia cu privire la valoarea preluata din slider



Stabilirea interfetei grafice

- Label-ul mare din partea dreapta va contine imaginea
- Butonul Proceseaza modifica imaginea
  - Devine va lid dupa ce se incarca o imagine
- Ultimul buton deschide dialogul pentru salvarea imaginii rezultate
  - Acesta va deveni valid doar dupa apasarea butonului “Proceseaza”



Stabilirea interfetei grafice

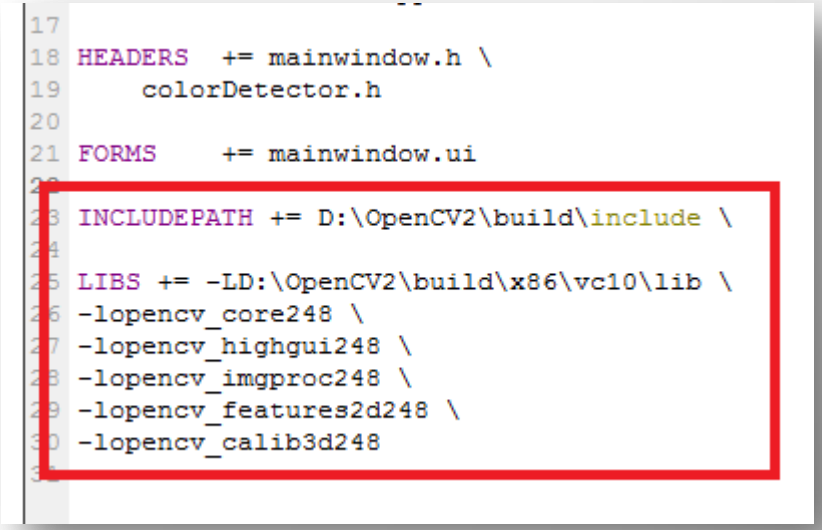
- Pentru toate componentele din GUI cu exceptia etichetelor generam functii de tratare a evenimentelor

```
private slots:

    void on_verticalSlider_valueChanged(int value);
    void on_pushButton_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_2_clicked();
    void on_pushButton_4_clicked();
```

- Captura este din fisierul mainwindow.h si contine sloturile care sunt adaugate automat cand se selecteaza actiunile corespunzatoare in mainwindow.ui.

Pachete necesare

- In **mainwindow.h** importam pachetele care ne sunt necesare in cadrul proiectului.
- Nu uitam sa facem conexiunile in cadrul fisierului **.pro**.



Variabilele din mainwindow.h

- Variabilele de care avem nevoie in interfata sunt incluse in **mainwindow.h** in zona **private**.
- Cu `ui` controlam obiectele din interfata.
- poza si rez reprezinta imaginile de intrare si iesire din algoritm
  - De cea de a doua avem nevoie pentru a salva imaginea rezultata pe calculator
- cDetect se ocupa de procesarea imaginii
- pozaFormatQT este folosita pentru a vedea daca trebuie sa facem transformare din Mat in QT.

```
private:
    Ui::MainWindow *ui;
    Mat poza;
    Mat rez;
    ColorDetector cDetect;
    bool pozaFormatQT;//retinem in ce format este poza in momentul curent
```

mainwindow.cpp

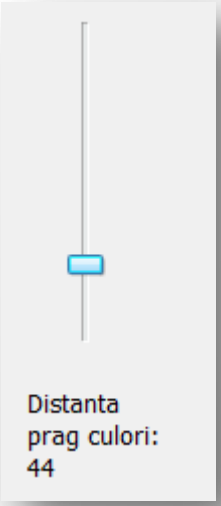
- In constructor dezactivam ultimele doua butoane si initializam variabila pozaFormatQT cu false.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("Procesare imagine");

    //dezactivam butonul de procesare pana cand incarcam o imagine
    ui->pushButton_3->setEnabled(false);

    //dezactivam butonul de salvare pana cand se obtine o imagine noua
    ui->pushButton_4->setEnabled(false);

    //la inceput citim poza in format Mat
    pozaFormatQT = false;
}
```

Functia pentru slider

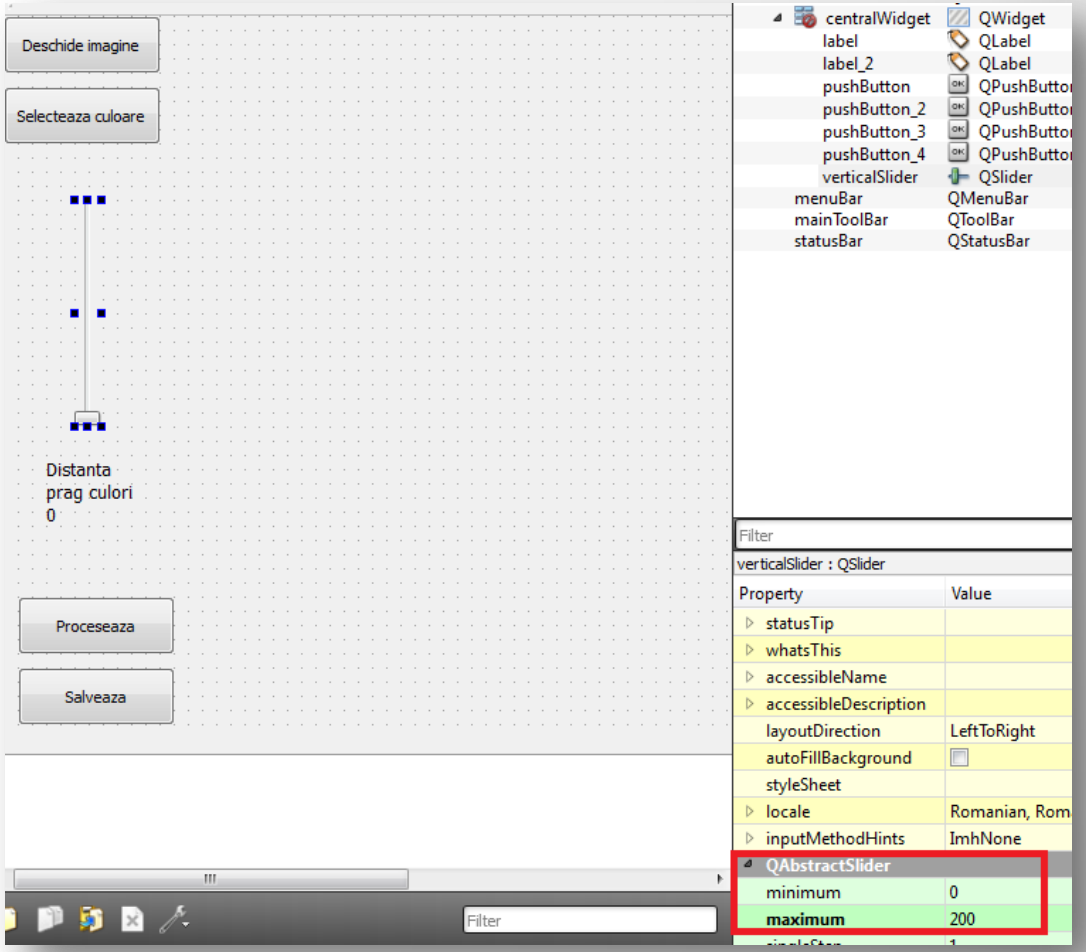
- Functia de mai jos intoarce valoarea curenta din slider.
- Construim un string la care adaugam valoarea pentru ca utilizatorul sa poata observa ce valoare selecteaza.
  - Stringul este trimis catre primul label
- Valoarea este trimisa catre pragul din cDetect

```
void MainWindow::on_verticalSlider_valueChanged(int value)
{
    //formam un string la care se adauga valoarea de la slider
    QString cdt("Distanta prag culori: ");
    cdt.append(QString::number(value));

    //Stringul este trimis la eticheta de sub slider
    this->ui->label->setText(cdt);

    //valoarea din slider este transmisa valorii prag din cDetect
    cDetect.setColorDistanceThreshold(value);
}
```



Slider

- Implicit, slider-ul ia valori intre 0 si 100
- Daca vrem sa stabilim alte limite, schimbam valorile la sectiunea din dreapta
- Se pot schimba si alte detalii legate de interfata

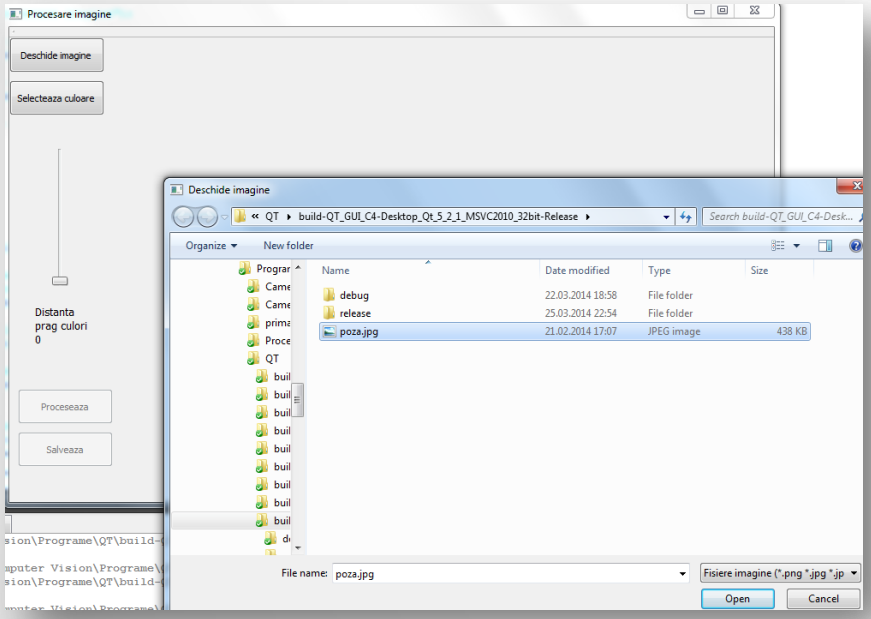


Incarcarea pozei

- Se citeste poza
- Se activeaza butonul de procesare
- Schimbam canalele din BGR in RGB pt QT
- Punem poza in partea dreapta a ferestrei

```
//butonul de incarcare a pozei
void MainWindow::on_pushButton_clicked()
{
    //dialogul pentru citirea pozei
    QString fileName = QFileDialog::getOpenFileName(this,
                       tr("Open image"), ".",
                       tr("Image Files (*.png *.jpg *.jpeg *.bmp)"));
    //citim imaginea in Mat
    poza= imread(fileName.toLatin1().data());

    //activam butonul de procesare
    if (poza.data)
        ui->pushButton_3->setEnabled(true);

    //pregatim poza pentru transformare in formatul QT
    //si retinem ac lucru in pozaFormatQT
    cvtColor(poza, poza, CV_BGR2RGB);
    pozaFormatQT = true;

    //trecem poza in QImage
    QImage img = QImage((uchar*) poza.data, poza.cols,
                       poza.rows, poza.step, QImage::Format_RGB888);

    //trimitem poza catre label-ul mare din dreapta
    ui->label_2->setPixmap(QPixmap::fromImage(img));
    ui->label_2->resize(ui->label_2->pixmap()->size());
}
```

Incarcarea pozei

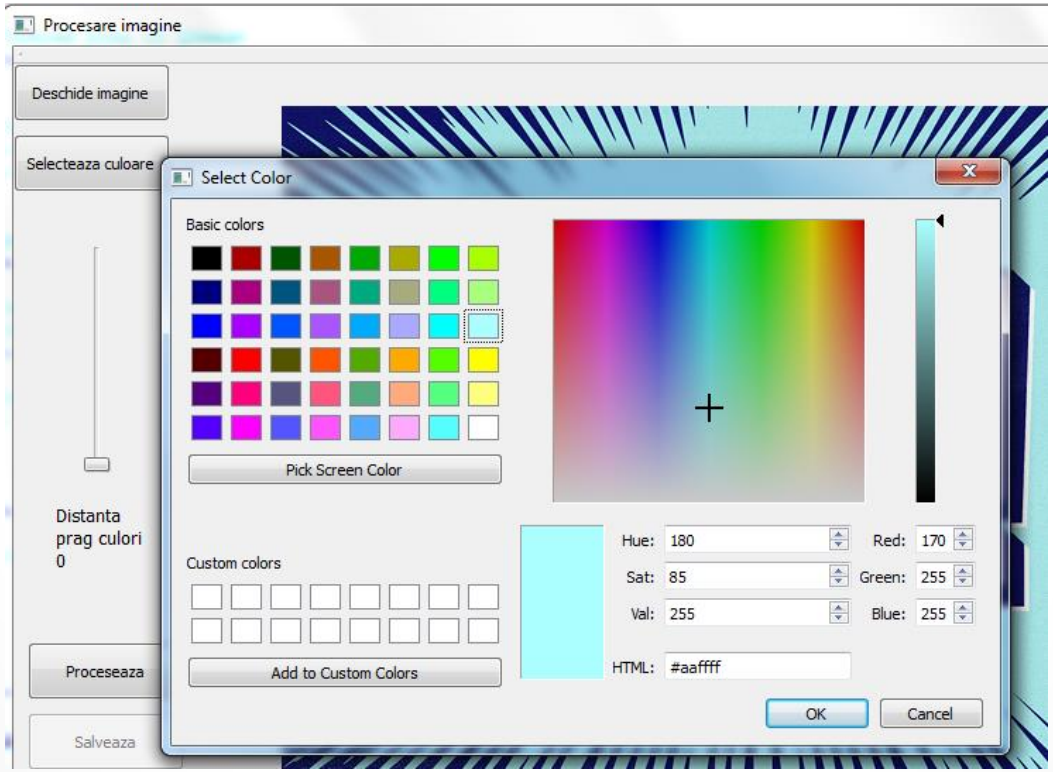


Alegerea culorii tinta

- Alegerea culorii se face prin dialogul oferit de QT
- Culoarea este apoi trimisa catre metoda specifica lui `cDetect`.

```
//butonul pentru alegerea culorii
void MainWindow::on_pushButton_2_clicked()
{
    //dialogul de alegere a culorii
    //culoarea pe care este pus cursorul este albastra
    QColor color = QColorDialog::getColor(Qt::blue, this);

    //culoarea este setata ca si tinta in cDetect
    if (color.isValid())
        cDetect.setTargetColor(color.red(), color.green(), color.blue());
}
```

Alegerea culorii tinta



Butonul de procesare

- Verificam daca poza este in format QT si in caz afirmativ o trecem inapoi in BGR
  - Verificarea este necesara pentru cazul in care utilizatorul apasa de mai multe ori butonul de procesare

```
//procesare
void MainWindow::on_pushButton_3_clicked()
{
    //daca poza este in format QT, o trecem inapoi in format Mat
    if(pozaFormatQT)
    {
        cvtColor(poza, poza, CV_RGB2BGR);
        pozaFormatQT = false;
    }

    //procesam poza si rezultatul este pus in rez
    rez = cDetect.process(poza);

    //rez este alb-negru, deci se retine intr-un singur canal
    QImage img= QImage((uchar*) rez.data, rez.cols, rez.rows, rez.step, QImage::Format_Indexed8);
    ui->label_2->setPixmap(QPixmap::fromImage(img));

    //activam bunotul de salvare
    if(!ui->pushButton_4->isEnabled())
        ui->pushButton_4->setEnabled(true);
}
```

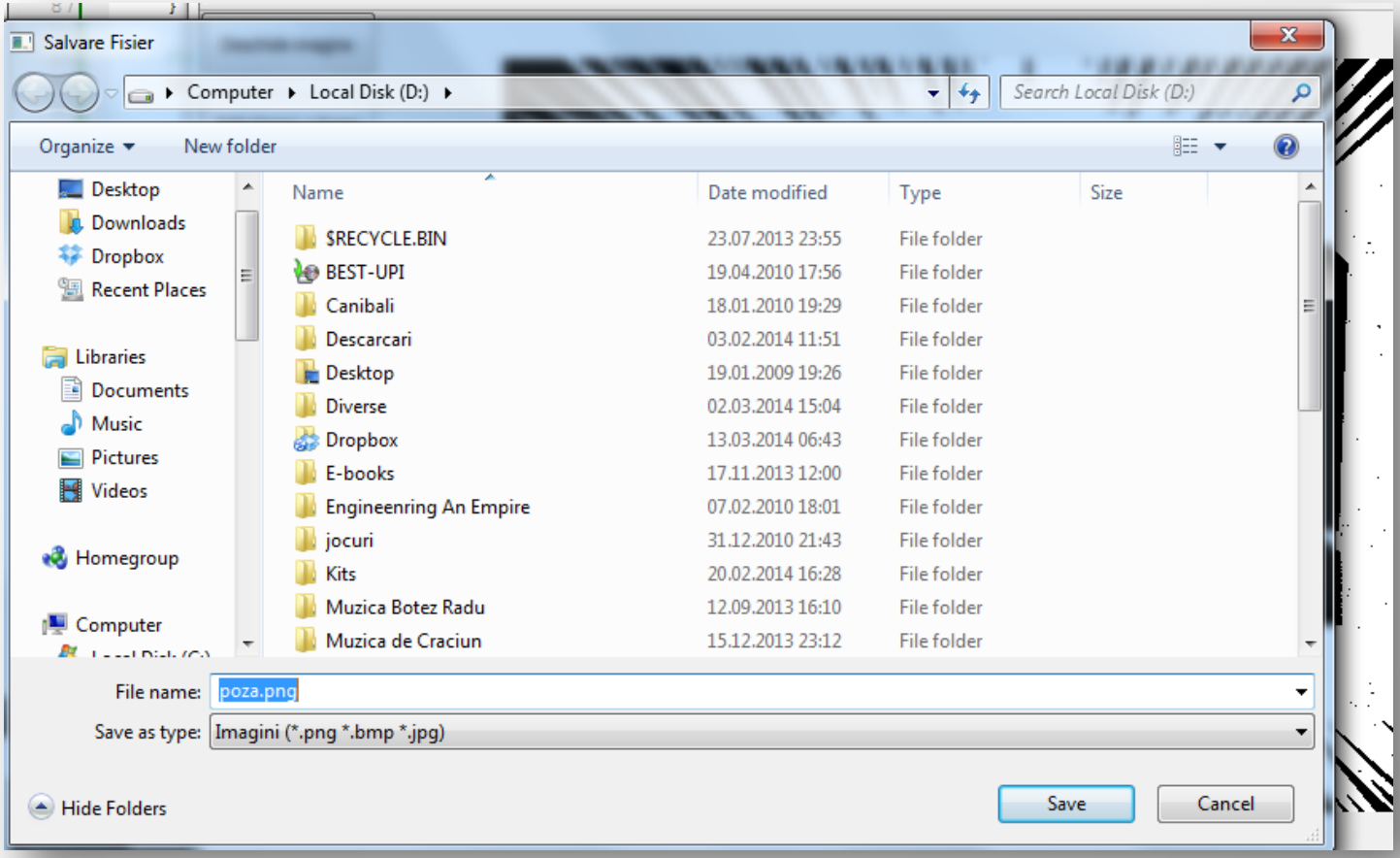
Butonul de procesare

Butonul de salvare

- Dialog de salvare a pozei unde se stabileste calea catre fisier si numele fisierului.
- Stabilim acel Qfile cu imaginea care urmeaza sa fie salvata.

```
//salvare
void MainWindow::on_pushButton_4_clicked()
{
    //dialogul pentru salvare
    QString fileName = QFileDialog::getSaveFileName(this, tr("Salvare Fisier"),
                                "D:/poza.png",
                                tr("Imagini (*.png *.bmp *.jpg)"));
    QFile f(fileName);
    f.open(QIODevice::WriteOnly);

    QImage img= QImage((uchar*) rez.data, rez.cols,
                       rez.rows, rez.step, QImage::Format_Indexed8);
    QPixmap::fromImage(img).save(&f, "PNG");
}
```

Butonul de salvare



Exercitii

- La proiectul anterior adaugati butoane radio pentru a permite utilizatorului sa aleaga din mai multe distante posibile:
  - Manhattan
  - Euclidiana
  - Chebyshev
  - Minkowski
- Scoateti butonul de procesare si realizati procesarea imaginii imediat ce se elibereaza clicul de la mouse de pe slider. Modificarea se face doar daca valoarea de la slider este diferita de 0.