0444547011 4547070 4547112 4547059 0671112615 0631112615 0990112615 0996802615 Набрать в час дня

*// public static Image mat2Image(Mat src) {  
//  
// if (src != null) {  
// int cols = src.cols();  
// int rows = src.rows();  
// int elemSize = (int) src.elemSize();  
// byte[] data = new byte[cols \* rows \* elemSize];  
// int type;  
// switch (src.channels()) {  
// case 1:  
// type = BufferedImage.TYPE\_BYTE\_GRAY;  
// break;  
// case 3:  
// type = BufferedImage.TYPE\_3BYTE\_BGR;  
// // bgr to rgb  
// byte b;  
// for (int i = 0; i < data.length; i = i + 3) {  
// b = data[i];  
// data[i] = data[i + 2];  
// data[i + 2] = b;  
// }  
// break;  
// default:  
// return null;  
// }  
// BufferedImage bimg = new BufferedImage(cols, rows, type);  
// bimg.getRaster().setDataElements(0, 0, cols, rows, data);  
// return SwingFXUtils.toFXImage(bimg, null);  
// }  
// return null;  
// }*

*// String url = "src/main/java/sample/my\_foto.jpg";  
// File file = new File(url);  
// FileInputStream fileIn = new FileInputStream(file);  
// //Можно просто ссылку писать и будет работать.  
// Image image = new Image(fileIn);  
//  
//// ImageView imageView = new ImageView(raster);  
// ImageView colorView = new ImageView(image);  
// ImageView bhView = new ImageView(image);  
//  
// ColorAdjust colorAdjust = new ColorAdjust();  
// colorAdjust.setBrightness(-0.8);  
// colorAdjust.setContrast(-0.7);  
//  
// bhView.setEffect(colorAdjust);  
//  
// primaryStage.setScene(new Scene(new VBox(bhView, colorView)));  
// primaryStage.setTitle("MyFoto");  
// FlowPane flowPane = new FlowPane();  
// flowPane.setAlignment(Pos.TOP\_CENTER);  
// Scene scene = new Scene(flowPane, 1200, 800);  
// primaryStage.setScene(scene);  
// flowPane.getChildren().addAll(bhView, colorView);  
// primaryStage.show();*

*// WritableImage raster = new WritableImage(image.getPixelReader(), (int) image.getWidth(), (int) image.getHeight());  
// PixelWriter pixelWriter = raster.getPixelWriter();  
// PixelReader pixelReader = raster.getPixelReader();  
  
  
// BufferedImage bufferedImage= ImageIO.read(file);  
// WritableRaster writableRaster=bufferedImage.getRaster();  
//  
//  
// Object data=writableRaster.getDataElements(0,0,null);  
// ColorModel colorModel=bufferedImage.getColorModel();  
// Color color=new Color(colorModel.getRGB(data),true);  
// writableRaster.setDataElements(100,100,data);  
// bufferedImage.setData(writableRaster);  
// ImageIO.write(bufferedImage,"jpg",file);  
  
  
// for (int i = 0; i < raster.getHeight(); i++) {  
// for (int j = 0; j < raster.getWidth(); j++) {  
// Color c = pixelReader.getColor(j, i);  
// if (c.getOpacity() < 1) {  
// pixelWriter.setColor(j, i, Color.WHITE);  
// }  
// if (c.getRed() > 0 || c.getGreen() > 0 || c.getBlue() > 0) {  
// pixelWriter.setColor(j, i, Color.BLACK);  
// }  
// }  
// }  
  
// BufferedImage image1  
  
// for (int i = 0; i < raster.getWidth(); i++) {  
// int[] pixel= writableRaster.getPixel(i,0,new int[4]);  
// pixel[0]=1;  
// pixel[1]=8;  
// pixel[2]=9;  
// writableRaster.setPixel(i,0,pixel);  
// }  
// writableRaster.getPixels(0,0,bufferedImage.getHeight(),bufferedImage.getWidth(),new int[4 \* bufferedImage.getHeight()\* bufferedImage.getWidth()]);  
// writableRaster.setPixels(0,0,bufferedImage.getHeight(),bufferedImage.getWidth(),new int[4 \* bufferedImage.getHeight()\* bufferedImage.getWidth()]);*

*// Найти треугольник начало  
// JFrame window = new JFrame();  
// JLabel screen = new JLabel();  
// ImageIcon imageIcon;  
// window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
// window.setVisible(true);  
//  
  
// Image image = Images.getImage().getImage();  
// Mat img = imageToMat(image);  
// if (img.empty()) {  
// System.out.println("Не удалось загрузить изображение");  
// return;  
// }  
// Mat destination = new Mat(img.rows(), img.cols(), img.type());  
// //Преобразует изображение из одного цветового пространства в другое,  
// // где исходное изображение хранится в двух плоскостях.  
// // На данный момент эта функция поддерживает только преобразование YUV420 в RGB.  
// Imgproc.cvtColor(img, destination, Imgproc.COLOR\_BGR2GRAY);  
//  
// //Выполнить автоматическое выравнивание гистограммы для 8-битного изображения в  
// // градациях серого можно с помощью метода equalizeHist()  
// Imgproc.equalizeHist(destination, destination);  
// //Размытие по Гауссу реализуется с помощью статического метода GaussianBlur() из класса Imgproc  
// // В первом параметре указывается исходное изображение (глубина CV\_8U, CV\_16U, CV\_16S, CV\_32F или CV\_64F),  
// // а во втором — матрица, в которую будет записан результат операции. В параметре ksize указываются размеры ядра фильтра.  
// // Ширина и высота ядра могут различаться, но оба они должны быть положительными и нечетными.  
// Imgproc.GaussianBlur(destination, destination, new Size(5, 5), Core.BORDER\_DEFAULT);  
//  
// //Выделить границы объектов на изображении позволяет статический метод Canny() из класса Imgproc  
// Imgproc.Canny(destination, destination, 50, 200);  
// //Кратко Отбрасывает четные строки и столбцы Элементы исходного изображения должны состоять из одного канала  
// // и иметь глубину цвета 8 битов или 32 бита (вещественное число).  
// // Параметр thresh задает пороговое значение, а параметр maxval — максимальное значение  
// // (используется только в алгоритмах THRESH\_BINARY и THRESH\_BINARY\_INV).  
// // В параметре type можно указать следующие алгоритмы (константы из класса Imgproc):  
// //THRESH\_BINARY — если src(x, y) больше thresh, то dst(x, y) будет иметь значение maxval, в противном случае 0.  
// Imgproc.threshold(destination, destination, 0, 255, Imgproc.THRESH\_BINARY);  
// ArrayList<MatOfPoint> contours = new ArrayList<MatOfPoint>();  
// Mat edges = new Mat();  
// Imgproc.Canny(destination, edges, 80, 200);  
// //Находит контуры в бинарном изображении.  
// // Функция извлекает контуры из бинарного изображения с помощью алгоритма CITE: Suzuki85.  
// // Контуры — полезный инструмент для анализа формы, обнаружения и распознавания объектов  
// Imgproc.findContours(destination, contours, new Mat(),  
// Imgproc.RETR\_EXTERNAL, Imgproc.CHAIN\_APPROX\_SIMPLE);  
//  
// for (int i = 0, j = contours.size(); i < j; i++) {  
// System.out.println(Imgproc.contourArea(contours.get(i)));  
// Rect r = Imgproc.boundingRect(contours.get(i));  
// System.out.println("boundingRect = " + r);  
// double len = Imgproc.arcLength(new MatOfPoint2f(contours.get(i).toArray()), true);  
// System.out.println("arcLength = " + len);  
// if (14.8 < (len \* len) / Imgproc.contourArea(contours.get(i)) && Imgproc.contourArea(contours.get(i)) < 200) {  
// if (12 < (len \* len) / Imgproc.contourArea(contours.get(i)) && Imgproc.contourArea(contours.get(i)) < 14.8) {  
// Imgproc.rectangle(img, new Point(r.x, r.y), new Point(r.x + r.width - 1, r.y + r.height - 1), new Scalar(0, 0, 255));  
// MatOfByte matOfByte = new MatOfByte();  
// Imgcodecs.imencode(".png", img, matOfByte);  
// imageIcon = new ImageIcon(matOfByte.toArray());  
// screen.setIcon(imageIcon);  
// window.getContentPane().add(screen);  
// window.pack();  
// }  
// }  
// }*